Measuring Valence Through Physiological Reactions

A pilot experiment

Erik Aas Borge

Master of Science in Mechanical Engineering
Submission date: June 2017
Supervisor: Martin Steinert, MTP

Norwegian University of Science and Technology
Department of Mechanical and Industrial Engineering

Acknowledgements

This thesis represents some of my final work at Norwegian University of Science and Technology. It has been a fun, but fuzzy journey. Foremost, I would like to give my thanks to my supervisor, Professor Martin Steinert, for guidance throughout the spring semester. Furthermore, a special thank you to Jørgen A. B. Erichsen, Andreas Wulvik and Stephanie Balters for valuable and much appreciated input and for their willing to help. Thanks to Helge Garsmark for good collaboration and to all the guys at P314 for their childishness and willing to sacrifice themselves for this research. Lastly, a huge thanks to my girlfriend (and partly psychiatrist), Lovisa, for her endless support and eminent ability to keep me calm, focused and generally alive the last few months.

This page is intentionally left blank.

Abstract

The much-debated topic of physiologically differentiating emotions has been of great interest to the research community in recent times. While many studies focus on facial reactions, the rest of the body is rather unexplored. This thesis takes upon the challenge of experimentally piloting and testing whether we can measure positive and negative emotional valence changes through physiological reactions in a user interaction setting. 34 adult subjects played variations of Tetris while collecting electromyography (EMG) data on the neck and forearm, and leaning distance in a seated position. Three scenarios were designed with different levels of valence and arousal (high valence-low arousal, low valence-high arousal, high valence-high arousal), upon which physiological data was analyzed. Valence was used as a within-subjects factor via self-report measurements by the Russell Affect Grid (Valence-Arousal grid) and Positive Affect Negative Affect Schedule (PANAS). Results showed that there was no statistically significant difference in physiological data between the scenarios ($p = .761$, $p = .191$ and $p = .093$ for EMG neck, EMG arm and leaning distance, respectively). However, the data showed tendencies in increased leaning distance from negative to positive valence. Furthermore, the thesis focuses on how one without much knowledge within the field of human experimentation can prototype and iterate one's way to such an experimental design, applying product development models to achieve this.

Keywords: Emotions, Physiological Reactions, Valence

This page is intentionally left blank.

Sammendrag

Det mye omdiskuterte temaet om å fysiologisk differensiere følelser har vært av høy interesse i forskningsmiljøet i senere tid. Mens mange studier fokuserer på reaksjoner i ansiktsområdet, er resten av kroppen heller lite utforsket. Denne oppgaven tar for seg utfordringen med å eksperimentelt pilot-teste og utforske om vi kan måle positiv og negativ emosjonell valens-endringer gjennom fysiologiske reaksjoner i en brukerinteraksjonssituasjon. 34 voksne deltagere spilte variasjoner av Tetris mens elektromyografi (EMG) data på nakken og underarmen, samt lenedistanse i en sittende posisjon ble målt. Tre scenario ble designet med forskjellig nivå av valens og arousal (høy valens-lav arousal, lav valens-høy arousal, høy valens-høy arousal), hvor de fysiologiske dataene ble analysert i henhold til dette. Valens ble brukt som en innad-i-deltager-faktor via selvrapporteringsverktøyene "Russel Affect Grid" (valens-arousal-grid) og "Positive Affect Negative Affect Schedule" (PANAS). Resultatene viste at det var ingen statistisk signifikant forskjell i fysiologisk data mellom scenarioene ($p = .761$, $p = .191$ og $p = .093$ for EMG nakke, EMG arm og lenedistanse, respektivt). Derimot viste dataene tendenser til økt lenedistanse fra negativ til positiv valens. Videre fokuserer oppgaven på hvordan man uten mye forhåndskunnskaper innen eksperimentering med mennesker kan prototype og iterere seg frem til et slikt eksperimentelt design, ved bruk av produktutviklingsmetodikk til å oppnå dette.

This page is intentionally left blank.

Table of Contents

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

List of Figures

Abbreviations

EMG	=	Electromyography
PANAS	=	Positive Negative Affect Schedule
PA	=	Positive Affect
NA	=	Negative Affect
S1/S2/S3	=	Scenario 1/2/3
ANOVA	=	Analysis of Variance
SD	=	Standard Deviation
Mdn	=	Median
χ^2	=	Chi-square
p	=	Significance level
ε	=	Epsilon, correction variable for ANOVA

This page is intentionally left blank.

Chapter 1

Introduction

1.1 Background

Human emotions are complex. Every day we communicate, and every day we continuously interpret the feelings of one another. Though this can seem fairly simple, it is a non-verbal, unconscious action that we just do, often without reflecting upon how we do it or what it is that makes up these interpretations. It could be easy to tell, even with high accuracy, that a person is happy, but possibly problematic to know exactly what it is that person does that creates this impression. A smiling face could be one answer, but facial expressions are just a part of one component of human emotion (Desmet, 2005). In product development and design, the continuous strive to design better user experiences and interfaces signifies that capturing the human emotional element is of great importance (Balters and Steinert, 2015). Developers have and are still straining to design products that elicit a deeper meaning in users. To many, an iPhone is not just a cell phone, and a Harley Davidson is certainly not just a motorcycle. This illustrates that our relationship with the world around us is not only driven by the rational, but also the affective. Desmet (2005) wrote,

> *"So far, little is known about how people respond emotionally to products and what aspects of design or interaction trigger emotional responses."*

This highlights a need for a deeper understanding of these concepts, and is the fundament of which this thesis is built upon.

1.2 Scope

In user experience design the term valence is of interest. Valence can be said to be how pleasant or unpleasant an experience or interaction is, or the degree of "good"-ness or "bad"-ness perceived. Valence is not something that is restricted to specific feelings, but rather groups of feelings, where feelings such as anger and hostility represent negative-

and inspiration and joy represent positive valence. When designing products one would often want to evoke positive and pleasant feelings, or positive valence, in the user. The challenge is then to understand how one can capture and interpret this concept. How can we measure valence? Attempting to make sense of the complex concept of emotions, Desmet (2005) suggests that these consist of four components: Behavioral reactions (e.g. retreating), expressive reactions (e.g. smiling), subjective feelings (e.g. feeling amused) and physiological reactions (e.g. heart pounding). A number of well-established tools for measuring subjective feelings exist, e.g. the Affect Grid (Russel et al., 1989) measuring pleasure-displeasure and arousal-sleepiness in a two-dimensional grid, and the Positive Affect Negative Affect Schedule (PANAS) (Watson et al., 1988) measuring just that by rating scales on a set of 20 words representing positive and negative feelings. While these are easy-to-use tools that can be constructed as desired to measure a particular emotion or set of emotions, they require conscious awareness of the emotional state one is in. Also, labelling emotions and rating them on a scale is not something people normally do, and some could find this difficult. Hereby, this thesis seeks to explore other, more objective ways to measure valence. By using the existing subjective measurement tools the Affect Grid and PANAS as a basis, the mission of the author is to find a relation between these and the last component of emotions: physiological reactions. How can we physically observe valence? How do human reactions represent positive or negative feelings?

This relates to the thematic problem description for this thesis:

> *Experimentally piloting and testing whether we can measure valence changes (stimulus check via PANAS and Affect Grid) in a user interaction setting via EMG and posture.*

1.3 Outline

Firstly, the thesis introduces a theoretical background for the work done as well as earlier relevant research within this field. An experimental design with constructed hypotheses is then presented, where muscle tension and posture is measured through electromyography (EMG) and distance sensors. Results from the experiment are presented and analyzed with the ultimate goal of answering if these reactions could function as objective measurements of valence. Lastly, the thesis reflects around and concludes upon the work conducted.

Chapter 2

Theoretical Background

2.1 Affective engineering

Affective Engineering, also known as *Kansei Engineering* (KE), is a concept that is based on the desire of designing products that elicit positive emotions in the consumer. It stems from Japan in the 1970s, where the term was introduced by Professor M. Nagamachi (Dahlgaard et al., 2008), which explains the term *Kansei* as *"a consumer's psychological feeling and image regarding a new product"* (Nagamachi, 1995). Dahlgaard et al. (2008) describes that the terminology of the word itself can be translated as *emotionality, sensuality or sensitivity*. In short, and in the context of product development, one can describe KE as the intention of implementing and interpreting human affect in the design process to develop better products. The approach became popular in the Japanese car industry in the 1980s and has since then been adapted to a broad number of industries. There also exist methodologies that support this vision, such as the Design Thinking model (Brown, 2008), where empathizing with the users for which a product is designed to better take this into consideration when designing is in focus. This illustrates that the desire for implementing human emotion and affect into product development processes is high. What emotions are will be discussed in the following section.

2.2 Human Emotions

As mentioned in the introduction, emotions are complex. While we constantly feel them and interpret others' emotions and can generally understand what they are, the concept seems to be highly intangible and difficult to define. In his component process model, Scherer (1987) defines an emotion as *"an episode of interrelated, synchronized changes in the states of all or most of the five organismic subsystems in response to the evaluation of an external or internal stimulus event as relevant to major concerns of the organism"*. A rather complex description. A sheer number of attempts to precisely define emotions have been made and dates all the way back to Darwin. However, the literature seems to have agreed upon that several ingredients comprise emotions, and Desmet (2005) states that they are best treated as multifaceted phenomena that consist of the components mentioned in the introduction of this thesis: behavioural reactions, expressive reactions, physiological reactions and subjective feelings. Worth noting is the distinction between emotions and feelings in this statement, where feelings exist as one element of emotion, representing the subjective experience of something. Recalling the explanation of valence in the introduction, this is a way to categorize groups of feelings. However, the components of emotions necessarily have to be interconnected to some degree, as one for example often smiles when feeling happy. Hence, valence could be visible in other components of emotions. Furthermore, Balters and Steinert (2015) state that emotions are no longer considered in relation to random behavioral patterns varying between individuals, but rather as repeatedly occurring patterns, suggesting that a systematic approach to explore these patterns could enable quantification of emotions and how they manifest in human behaviour. To facilitate this type of research, one has to consider two things: (1) capturing the emotion itself and (2) capturing how the emotion propagates in behaviour or reactions. In relation to the component of interest in this thesis, physiological reactions, as of now no established objective way of measuring valence exist. In the next section, some of the established subjective measurement tools for valence is presented.

2.3 Subjectively Measuring Valence

In 1980, Russell introduced the Circumplex Model of Affect **(Fig. 2.1a)**, which was later to be called The Affect Grid (Russel et al., 1989). The grid takes the form of a 9x9 matrix **(Fig. 2.1b)** that represent the two dimensions, the horizontal signifying the valence- (unpleasant-pleasant) and the vertical signifying the arousal (sleepiness-arousal) axis. It assesses specific emotions not as independent dimensions, but rather as combinations of the dimensions of valence and arousal, where for example excitement is a combination of pleasant feelings and arousal, and depression is a combination of unpleasant feelings and sleepiness. Compared to other subjective measurement tools, like questionnaires, the Affect Grid presents a fast and simple way of reporting one's affective state. While the argument on discretizing emotions is ongoing, this tool still serves as a sufficient tool within the context of this thesis. As will be presented in **Chap. 3**, the conducted experiment designed three scenarios with three different levels of valence. While some will argue on exactly where different emotions should be placed in the grid, the difference in the valence dimension between the scenarios unarguably implies that one scenario is experienced as more positive than the other, and hence this is sufficient information to capture the subjective part of affect necessary in the context of this thesis. As we also will see later, the arousal dimension is of interest for the experiment, as it was conducted with another Masters student assessing this topic.

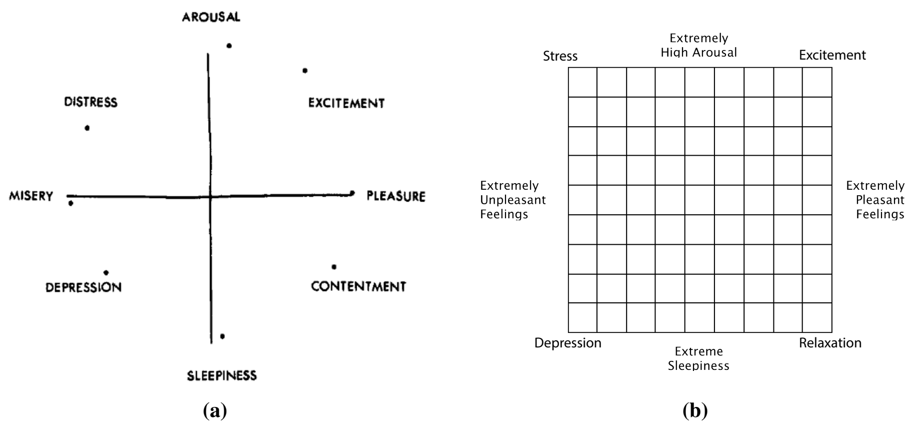


Figure 2.1: The original circumplex model (left) and the developed Affect Grid (right) (Russell (1980);Russel et al. (1989))

Further out in the 80s, Watson et al. (1988) defined a measuring scale according to the valence dimension of affect, the PANAS (Positive Affect Negative Affect Schedule), a scale that uses a series of positively and negatively loaded adjectives to determine the "valence state" of a person. There are ten words for the Positive Affect (PA)- and ten for the Negative Affect (NA) category, which are to be rated according to how a particular person has that feeling, on a scale with the following scale points: 1. Very Slightly or not at all, 2. A little, 3. Moderately, 4. Quite a bit, 5. Extremely. The words are presented in **Tab. 2.1**. The subscales, PA and NA, contain the words 1, 3, 5, 9, 10, 12, 14, 16, 17, and 19, and 2, 4, 6, 7, 8, 11, 13, 15, 18, and 20, respectively. The subjective rating of each word is summed up in each subscale with a value between 1 and 5, according to the scale points. Worth noting is the mean momentary (PA) score of **29.7** (std.dev. 7.9) and mean (NA) of **14.8** (std.dev. 5.4). That is, the score of a given person at a given point in time. These numbers are based on the response from a set of college students at Southern Methodist University in Texas in 1988 (Watson et al., 1988), and Crawford and Henry (2004) later confirmed the validity of the scale also for demographic variances.

Similar to the PANAS, the modified Differential Emotions Scale (mDES) measures positive and negative emotions with 18 words, 10 in the positive and 8 in the negative subscale. The positive emotions are amusement, awe, compassion, contentment, gratitude, hope, interest, joy, love, and pride, and the negative are anger, contempt, disgust, embarrassment, fear, guilt, sadness, and shame (Cohn et al., 2009).

One could reflect upon the composition of words in such a survey and which words that together best describes positive and negative feelings. Specific words can also be suspected not to only lie in the valence dimension, like *jittery* in the PANAS, which could very well also fit in the arousal dimension, considering the two-factor structure of the Affect Grid. However, for its proven reliability the PANAS is chosen as a supplementary measurement for valence in this thesis.

#	**Word**
1.	Interested
2.	Distressed
3.	Excited
4.	Upset
5.	Strong
6.	Guilty
7.	Scared
8.	Hostile
9.	Enthusiastic
10.	Proud
11.	Irritable
12.	Alert
13.	Ashamed
14.	Inspired
15.	Nervous
16.	Determined
17.	Attentive
18.	Jittery
19.	Active
20.	Afraid

Table 2.1: Words of the PANAS questionnaire

- Viktig: Det finnes enda ikke klare objektive mål på valens. Det er dette som er greia med oppgaven.

overgang: valens i kroppsspråk/fys.reak.?

2.4 Physiological measurements

Despite the intention behind KE of implementing affect in product development, Balters and Steinert (2015) state that most of the existing KE tools focus on deriving insights from triangulation of external factors. They further suggest that a better approach could be to measure and quantify human physiological response to model and estimate emotion reactivity, and through this enable a deeper understanding of the relationship between emotions and how they are expressed through behaviour. Furthermore, Ekman (1994) states that emotions are associated with particular patterns of facial expressions, which together with Balters and Steiner's similar statement on behavioural patterns suggest that it may be possible to capture these. In the scope of this thesis, this means to measure both the emotions, i.e. valence, and the reactions this may lead to, with the ultimate goal of finding a connection between them. This brings us to another of Desmet's components of emotion, physiological reactions. Several studies have assessed the topic of emotions being possible to distinguish in terms of their associated patterns of autonomic nervous system (ANS) activity (Levenson (2003), Christie and Friedman (2004)). Ax (1953) studied the physiological differentiation between anger and fear, on which Stemmler et al. (2001) followed up, measuring several physiological variables like electrocardiogram (ECG), blood pressure, skin temperature and electromyography (EMG, muscle tension). However, many studies focus on specific emotions and how to distinguish them. This thesis takes on a broader perspective, assessing groups of feelings relating to valence. Within the field of valence and physiological reactions, there have been some studies which focus on facial activity. Hazlett (2006) did a study on measuring valence through facial EMG during a video game play experience. He found that the zygomaticus muscle, which controls smiling, was found to be significantly more active during positive events as compared to negative. Cacioppo et al. (1986) found that facial EMG differentiated both the valence and the intensity of the affective reaction, while, interestingly, no change in facial expression could not be determined by visual inspection. Considering the thematic of the research done within this area, the reactions of the rest of the body seem rather unexplored, and since there's more to the human body than just the face, this could potentially function as an excellent channel for emotional communication (Coulson (2004); Montepare et al. (1999)).

This thesis will consider EMG measurements in two locations, the *Platysma* muscle on the neck, and the *Flexor Digitorum Superficialis* muscle on the arm **(Fig. 2.2)**. The Platysma muscle is a sheet of muscle that runs from the chest and shoulder muscles upwards along the neck. It is activated when moving the corners of the mouth downwards, as in an expression of surprise or fright. The Flexor Digitorum Superficialis is located along the lower side of the forearm and runs from the fingers to the elbow. It is activated when one tenses the arm by making a fist (HowToMedia, n.d.). The contraction of muscles accompanies electrical impulses within the muscle and can be measured by electrodes placed on three places on and around the muscle (Merletti and Parker, 2004). The potential difference between a (+) and (-) electrode is measured in millivolts, and a third neutral electrode, placed on electrically unrelated tissue, is applied as a baseline (Balters and Steinert, 2015).

In addition to muscle tension, posture is measured. While some studies have assessed the subjective perception of emotions through posture (e.g. Coulson (2004); Dael et al. (2012)), i.e. how one interprets emotions by observation of another, this thesis seeks to explore this from another point of view by measuring posture with emotion as input. This is measured as leaning distance in a seated position. That is, how much a person leans forward.

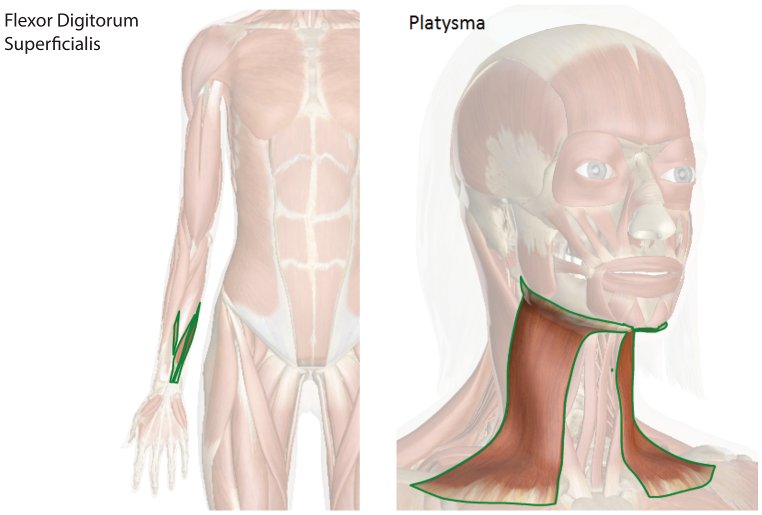


Figure 2.2: The Flexor Digitorum Superficialis (left) and Platysma (right) muscles. (HowToMedia, n.d.)

2.5 Application of theoretical background

As far as theoretical knowledge about the topics presented in this chapter go, the author has limited experience with human experimentation, and this type of research is not completely within the main field of the author. To cope with challenges regarding how one can set out to conduct this kind of research, the experiment presented in the next chapter has been subject to models within product development. The wayfaring approach, as proposed by Steinert and Leifer (2012), allowed for fast iterations, prototyping and testing of experiments which quickly generated new learnings on how to test, how to relate to human experimentation and generally what worked and what didn't. Furthermore, ideas from set-based engineering (Sobek et al., 1999) and agile development (Takeuchi and Nonaka, 1998) have been applied to enable progress in a field where the author's knowledge is limited.

Chapter 3

Experimental Setup

The following chapter presents the experimental approach and setup used by the author to study how the mentioned physiological reactions could vary with valence. The author and another master student have been working on this setup together, as the intention of the two allowed a merge of two experiments. Hereby, this chapter is written by both students. As some tools and theory was applied by one student only, this causes some information from Chapter 2 to be duplicated. The abstract of this student's master thesis is presented to enable an overview of his research:

> *Human-Computer Interaction systems are rapidly changing. In such interactions, human performance might be essential for the system to function in the best possible way. Thus the need to optimize for interaction design that take human emotion in to account are advancing. Great challenges are involved in capturing emotions, for instance that human emotion are complex, and difficult to quantify. In this thesis theory about body language and emotions are adapted in an attempt to introduce a new tool for Affective Engineering. This is done through developing a pilot chair sensor setup and test its capabilities towards capturing emotional states. A pilot experiment setup is used to test association between posture sensed by the chair and level of mental activation or arousal. Statistical analyzes on the data recorded is done to compare established measures of arousal with posture data from the chair. The analyzes show some tendencies toward association between movement and arousal, but more comprehensive analyzes are needed to introduce the setup as a new tool for affective research. The main take away from this thesis would be the experimental framework that was developed, which with small modification can be used for testing tools to be used in interaction studies.*
>
> Helge Soltvedt Garsmark, 2017

The combined intentions of the experiment was hereby to study how physiological reactions vary with changes in affect, including both valence and arousal. The goal was to design three scenarios to achieve three different levels of affect within the two dimensions, and then evaluate how the physiological data relates with the two. Furthermore, the setup of this experiment was not only result based. It was also a pre-study and an exploration on how one can study physiological reactions and emotional states in a user interaction setting. As such, this experiment is also provided as a pilot and a framework for how one can achieve this. This chapter introduces hypotheses, the complete setup for the experiment as well as the chosen measurement methods and input variables.

3.1 Hypotheses

In the following section a number of hypotheses are presented. All of the hypotheses are listed with their corresponding null-hypotheses. The hypotheses relates to relative change in valence and arousal.

3.1.1 Valence Hypotheses

These valence hypotheses relates to the established subjective measurements of valence and the goal was to explore the relationship between the objective measurements of physiological data, EMG and leaning distance, and the established subjective measurements, the Affect Grid and PANAS. The first hypothesis is regarding valence and its effect on muscle tension:

H1: "Changes in valence induce changes in muscle tension"
H1null: "Changes in valence does not induce changes in muscle tension"

The second hypothesis is regarding valence and its effect on the level of forward leaning, i.e. leaning distance:

H2: "Changes in valence affects the level of forward leaning"
H2null: "Changes in valence does not affect the level of forward leaning"

Hypothesis H1 was tested with EMG data which was compared with the data from the PANAS and the Affect Grid **section 3.7**. Hypothesis H2 was tested with data from a Lidar distance sensor, which was then compared with the same control measurements.

3.1.2 Arousal Hypotheses

The following arousal hypotheses relates to established measurements of arousal. The goal was to test if another objective measurement gathered from features from the chair setup were associated with the established subjective and objective measurements. The first hypothesis regarding arousal is connected to the effects arousal has on number of posture changes in a chair.

H3: "Changes in arousal affect the number of position changes in a chair in a given time period "
H3null: "Changes in arousal does not affect the number of position changes in a chair in a given time period"

The second hypothesis is regarding arousal and its effect on the level of forward leaning:

H4: "Changes in arousal affect the level of forward leaning"
H4null: "Changes in arousal does not affect the level of forward leaning"

Hypotheses H3 and H4 were tested with data from the chair that was processed through machine learning to classify positions and level of forward leaning. This data was then compared with both the subjective self report AD-ACL-questionnaire and objective HRV measurements. H4 was also tested with the Lidar distance sensor data.

3.2 Independent Variables

Since there was no direct way to vary the affect of the subjects, stimuli was used to achieve the desired emotional state. The stimuli represents the independent variable, where this was tuned to guide subjects in the wanted directions of affect. Following are the different means of stimuli used, and a short explanation of how they were adjustable. How they were implemented in this experiment is described in **section 3.9**.

3.2.1 Difficulty

The task, in this case TETRIS (described in 3.6), varied in difficulty. The manipulated version of the game allowed for adjustable speed, reversing the controls and to decide the order of the pieces, including which pieces to be included at all.

3.2.2 Lights

This was a visual stimuli that was in the form of a LED-strip around the LCD-screen. It could vary in intensity and color, be turned on/off and the speed of all these variations could be controlled.

3.2.3 Music

This was an audible stimuli. There are infinite aspects of music that could be controlled. The aspects that were given most thought in this experiment were tempo, pitch and familiarity.

3.2.4 Feedback

This was both a visual and audible stimuli and was controlled to match the performance and desired affect level. The feedback consisted of several elements, including a continuously changing feedback bar on the screen, score of the game and score relative to other people. It could differentiate from positive to negative loaded with both color and sound, and could be enabled and disabled.

3.3 Dependent Variables

For this experiment several dependent variables were used, both subjective and objective. Heart rate variability (HRV) and all of the subjective variables are established as measurements of the affect dimensions. EMG and the posture measurements (both chair and leaning distance) are the proposed new measurements for valence and arousal. The intention was to compare the EMG and leaning data with the established valence variables, and all the posture data with the established arousal variables. The variables are presented in **table 3.1**.

Variable	**Measurement**	**Subjective or objective**
Valence	Panas	Subjective
Valence and arousal	Affect Grid	Subjective
Arousal	AD-ACL	Subjective
Arousal	ECG - HRV	Objective
Proposed valence	EMG - Muscle tension	Objective
Proposed valence	Chair posture data	Objective
Proposed arousal	Chair posture data	Objective

Table 3.1: Dependent variables

3.4 Physical Setup

In the physical aspects of the experiment setup it was important to have a minimum of external "noise" to bias the data. In this experiment the physical setup was constructed by a mock-up room room **Fig. 3.1** where the test subjects participate without any human interaction after the initial briefing and sensor connection **Fig. 3.11**. That being said, the room was in a busy hallway close to a hallway door that was often being opened and slammed, that lead to some interrupting sounds and vibrations. Also the room requires to have a window open and a lot of outside sounds could be heard by the subjects. The intention behind this experiment does not require the setup to be "white-room" and totally isolated from external factors, which also would have been difficult in the scope of the project. But it is was supposed to be a lot more "neutral" compared to an in-situ setup whitch is a setup that is supposed to represent a situation that is as close to a real life situation as possible.

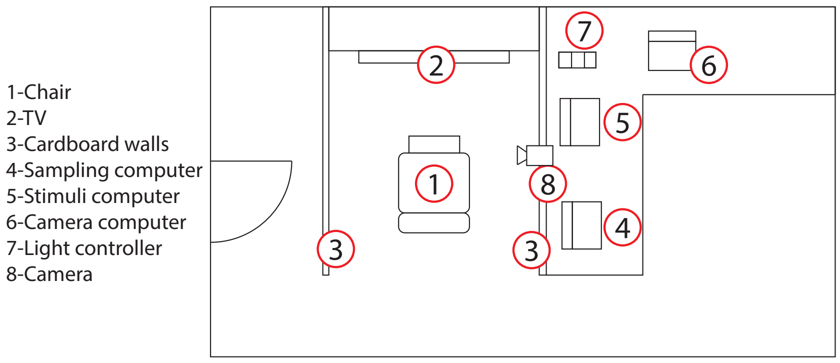


Figure 3.1: Room layout.

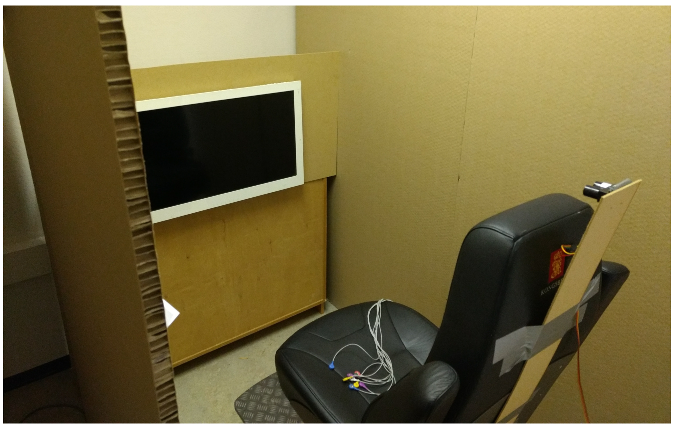
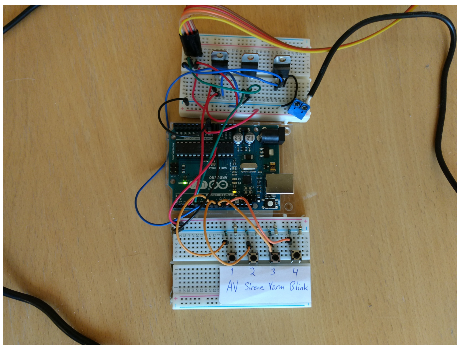


Figure 3.2: Overview of the physical setup

The subjects were placed in the chair, where they were initially briefed and connected to the physiological sensors **Fig. 3.11**. The subjects were given a wireless headset for sound stimuli and they interacted with the experiment interface with a wireless Xbox One-controller **Fig. 3.3a**. The interface was presented to the subjects on a 32" LCD screen with surrounding LED-strips for light stimuli **3.3b**. These LED-strips were controlled with an Arduino controlled button circuit **Fig. 3.3c**. Information sheets about how to use the Russel Affect Grid **(Appendix B)** were placed on the subjects left hand side. The subjects were isolated from the rest of the room with two cardboard walls to eliminate distractions from external effects in the room **Fig. 3.2**. Behind the cardboard wall on the chairs right hand side, the computer setup was placed **Fig. 3.3d**, this is where the experiment was controlled and monitored. This is explained in detail in the next section. A camera was placed on the top of the wall to the right of the subject for video recording **Fig. 3.1**.

(a) Subject in chair, with headset and controller

(b) LCD-screen with surrounding LED-strips



(c) Arduino controlled light circuit

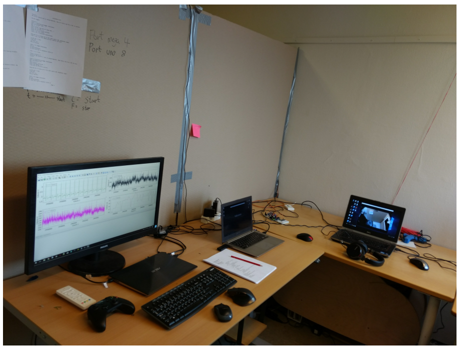

(d) The computer setup

Figure 3.3: Other aspects in the physical setup

3.5 Computer Setup

The vast amount of data collection and impulses required great control of the experimental environment. Three computers were used to ensure complete overview and control **(Fig. 3.4)**. The sampling computer was set up to gather all sensory data from the subjects, meaning ECG, EMG and posture both from the chair sensors and the Lidar distance sensor. The sensor platform, consisting of four Arduinos, provided the sensory data over a serial connection with software on the sampling computer capturing this and displaying it in realtime on screen. This allowed for substantial control of the input data, especially in the first phase of the experiment where the connection of the electrode pads for ECG and EMG sometimes didn't provide a good data stream. If some of the electrodes weren't connected well enough this could be easily seen and corrected. The sensor platform in itself will be fully explained and discussed in **section 3.7**. The stimuli computer functioned solely to provide the test subjects with the user interface part of the experiment. Here, the interface software, which will be explained in detail in (section 3.8), guided the user through the whole experiment on the TV screen. This made it possible to avoid interference with the subjects during testing, and at the same time ensured the exact same user experience for every subject. The third computer, the camera computer, was used to do video recordings as well as allowing visual observation of the subjects.

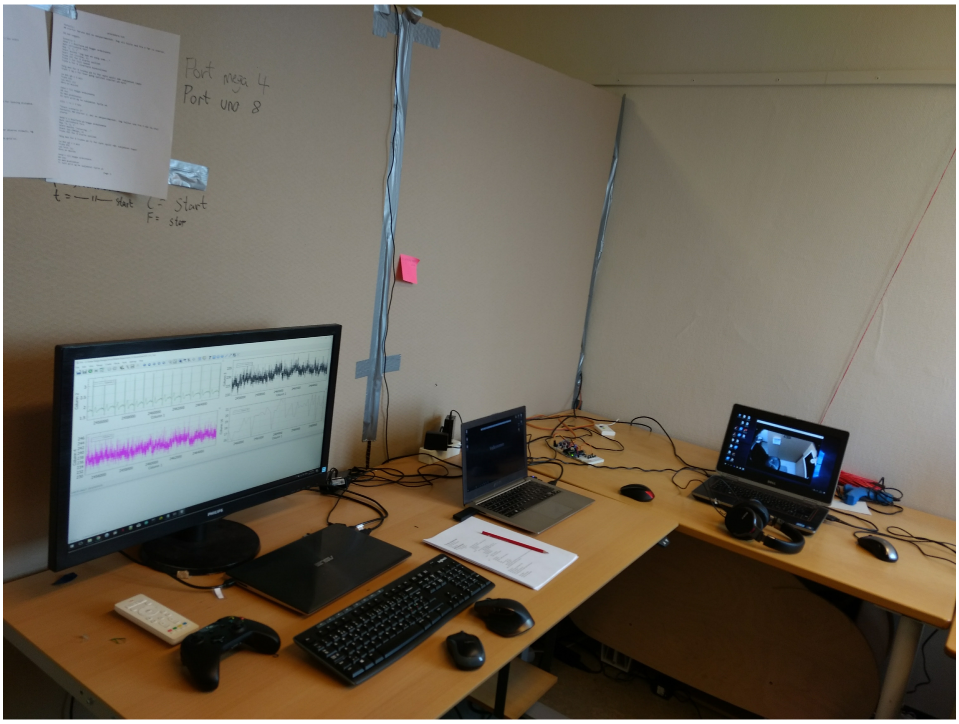
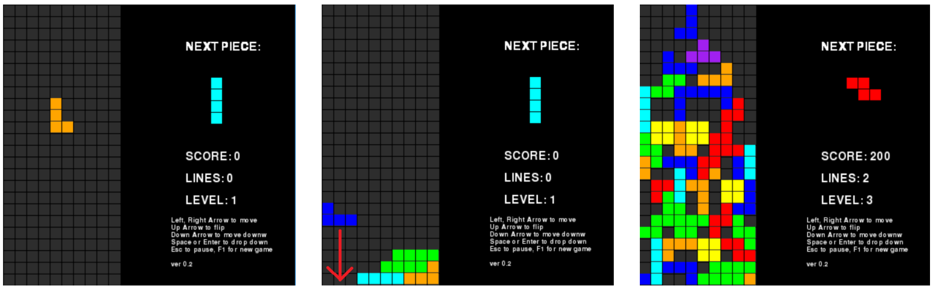


Figure 3.4: Overview of the computer setup

3.6 TETRIS

The traditional game TETRIS was applied as the main task of the experiment. TETRIS is a simple game where a grid defines the playing area. Pieces put together in different shapes fall down from the top one by one **Fig. 3.5a**, and the goal of the game is to puzzle these pieces together at the bottom so that you cover a full line with pieces across the grid (**Fig. 3.5b**). This line then disappears, and you receive points for each line. The game is over when the stack of bricks reaches the top of the grid **Fig. 3.5c**.



(a) Pieces fall down from the top **(b)** Making complete lines gives points **(c)** Game over when the stack reaches the top

Figure 3.5: Tetris

The TETRIS game used in this experiment was a clone of the original game, downloaded from GitHub. It was coded from scratch in the programming language Python and uses a series of functions from Pygame, which is a graphical interface pack compatible with Python. The fact that the whole game is constructed by just under 600 lines of code made i fairly easy to tweak for the purposes of the experiment, and design several versions of the game to achieve the different levels of affect intended. Moreover, it allowed for fast prototyping of the game and made it easy to test changes and see how they affected pilot subjects. The final versions are explained further in **section 3.9**.

3.7 Sensor platform and Measurements

In this section the different measurements, both subjective and objective, are described. Also the sensors and tools used to collect some of these measurements are described in this section.

3.7.1 Affect - self report measurements

To be able to have some control measurements in the experiment, some subjective measurements of affect was needed. The Russel Affect Grid is, as mentioned in **Chapter 2**, a established agent of measuring both dimensions of affect (arousal and valence). However we felt it was needed to include at least one more control measurement of both dimensions, and PANAS and AD-ACL was chosen for their proven reliability. Following are some short explanation of these measurements, a more detailed explaination of all three is included in

Russel Affect Grid

The grid consist of a 9x9 matrix, the horizontal dimension representing valence and the vertical representing arousal. This is described by Russel et al. (1989) as a map representation of feelings. The top right quadrant represents feelings of excitement, the bottom right quadrant feelings of relaxation, the bottom left feelings of depression and sadness, and lastly the top left quadrant distressing and tense feelings.

PANAS

The Positive and Negative Affect Schedule (PANAS) (Watson et al., 1988) is a questionnaire to measure valence, similar to the horizontal axis in the Russel Affect Grid. The subjects are presented with a mix of positively and negatively loaded adjectives. The subjects are to indicate the extent they feel these feelings on a scale with the following scale points: 1. Very slightly or not at all, 2. A little, 3. Moderately, 4. Quite a bit, 5. Extremely. The words are presented in **table 3.2**. The sub scales, *Positive affect (PA)* and *Negative affect (NA)*, contain the words 1, 3, 5, 9, 10, 12, 14, 16, 17, and 19, and 2, 4, 6, 7, 8, 11, 13, 15, 18, and 20, respectively. The subjective rating of each word is summed up, and the mean momentary scores of PA and NA are 29.7 and 14.8 respectively.

no.	Word
1.	Interested
2.	Distressed
3.	Excited
4.	Upset
5.	Strong
6.	Guilty
7.	Scared
8.	Hostile
9.	Enthusiastic
10.	Proud
11.	Irritable
12.	Alert
13.	Ashamed
14.	Inspired
15.	Nervous
16.	Determined
17.	Attentive
18.	Jittery
19.	Active
20.	Afraid

Table 3.2: Words of the PANAS questionnaire

AD ACL

The Activation-Deactivation Adjective Check List (AD ACL) is also a questionnaire where the subjects are presented with adjectives which they are to indicate on a scale their extent of feeling the specific feeling. But rather than being a measurement of valence like PANAS AD ACL is a measurement of arousal. The scale points in the Ad ACL questionnaire are: 1. Definitely do not feel, 2. Cannot decide, 3. Feel slightly, 4. Definitely feel

Number	Word
1	Active
2	Placid
3	Sleepy
4	Jittery
5	Energetic
6	Intense
7	Calm
8	Tired
9	Vigorous
10	At-rest
11	Drowsy
12	Fearful
13	Lively
14	Still
15	Wide-awake
16	Clutched-up
17	Quiet
18	Full-of-pep
19	Tense
20	Wakeful

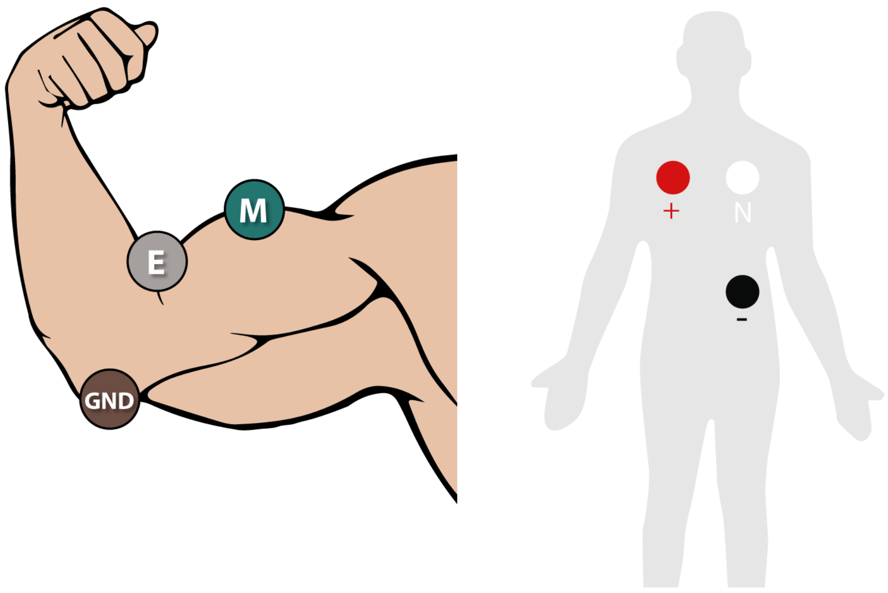
Table 3.3: Word of the AD-ACL questionnaire

3.7.2 Chair sensors and measurements

Force sensitive resistors (FSRs), were embedded in the chair. They acted as pressure sensors where the the resistance varied depending on the pressure applied. The intention of the chair sensor setup in this experiment was to provide measurements of the subjects posture during the experiment. This included sitting position, number of changes in sitting position in a given period of time, and in which degree the subjects was leaning forward. The forward leaning was also measured by using a Lidar distance sensor placed on the back of the chair, behind the subjects' heads. The Lidar sensor was chosen, even with some issues with stability in measurements, mainly because the only other option taht was available in this time scope was a HC-SR04 ultra sonic sensor, but the ultra sonic distance measurements gave sky high peak readings. In comparison the Lidar is highly reliable, although it has fairly low accuracy for the purpose of this experiment ($\pm 2.5cm$). All the above mentioned data was collected using an Arduino Mega, which had a sampling rate of 5 Hz.

3.7.3 Biometric sensors and measurements

The biometric sensors used in this experiment included two sets of electromyography (EMG) electrodes and one set of electrocardiography (ECG) electrodes. The EMG is an established method of measuring muscle tension (Merletti and Parker, 2004) and was used to measure muscle tension in the subjects' Flexor Digitorum Superficialis muscle on the arm and the Platysma muscle on the neck. Electrical impulses appear during a muscle contraction and hereby generates a voltage difference between the electrodes. The more contraction, the more voltage difference. ECG is a proven tool for measuring the muscle functions of the heart (Yakut et al., 2014). Also the ECG measurements was acquired by using three surface electrodes, the hearts muscle generates an electrical signal periodically (Yakut et al., 2014), this is represented by an analogical signal in volts by the ECG.



(a) EMG **(b)** ECG

Figure 3.6: Electrode placement (Hacks, 2017)

Both the EMG and the ECG signals were gathered using an Arduino Uno with an e-health shield **Fig. 3.7** at a sampling rate of 380 Hz to provide sufficient resolution of the ECG data. The e-health shield provided a simple way of gathering physiological data with the Arduino platform. Besides EMG and ECG it also has the capabilities to do other measurements such as blood pressure, body temperature and skin conductivity.

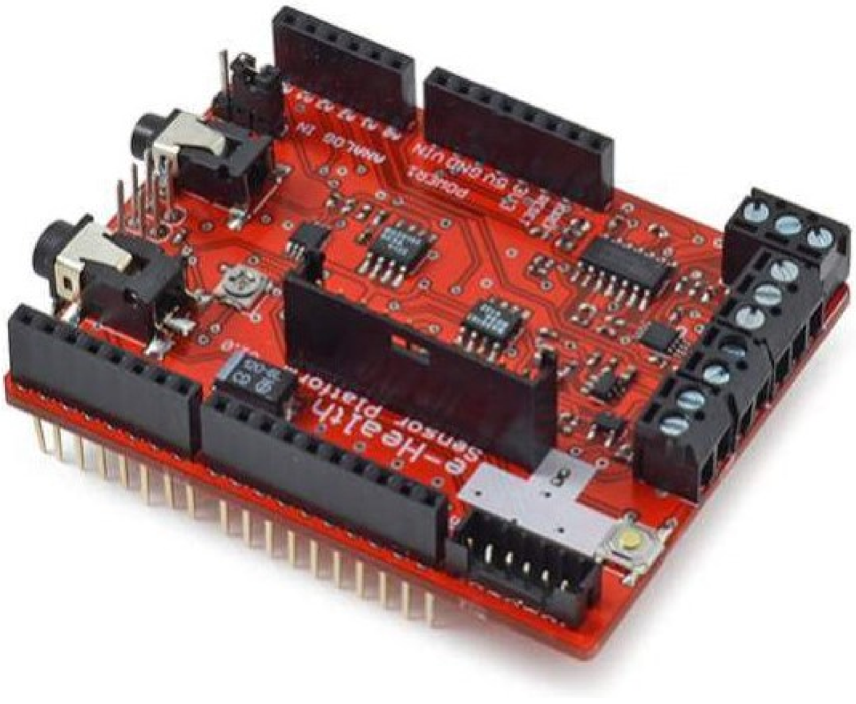
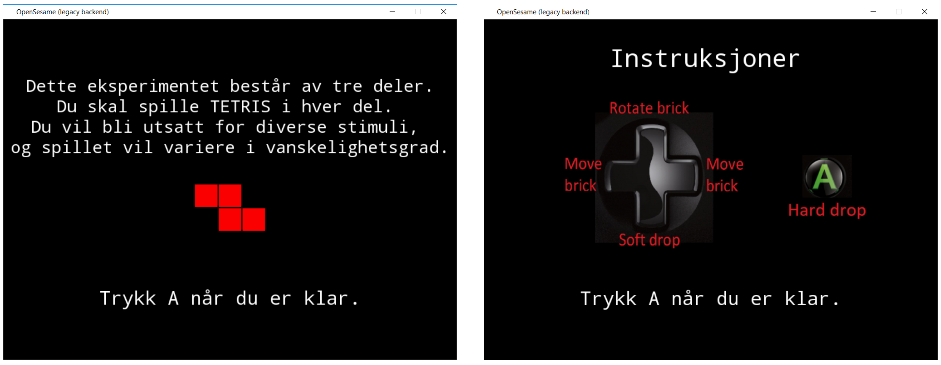


Figure 3.7: The e-Health Sensor Platform

3.8 User Interface

How the test subject is interacting with the tasks of an experiment is a vital part of every research that includes studying human behaviour. The interface created for this experiment was designed to facilitate easy interpretation of the necessary information, as well as focusing on consistency for all subjects. Furthermore, it was put effort to facilitate for a non biased response, as some of the main output used for data analysis was subjective (Affect grid, PANAS and AD-ACL). The interface was designed in a graphical experiment builder software, OpenSesame. This is a simple, free and open-source software for designing the graphical user interface (GUI) of experiments (Mathôt et al., 2012). OpenSesame provided tools for making the whole experiment, screen by screen (**Fig. 3.8**). **Appendix A** shows the GUI in its entirety. The subjects simply clicked their way through by using the controller. It also supports implementation of python scripts, which made it easy to integrate the Tetris code into the GUI. Even more convenient was the opportunity to code the affect grid(**Fig. 3.9a**) and the PANAS and AD-ACL (**Fig. 3.9b**) questionnaires in python as well, making the whole experiment on screen. This was done from scratch **Appendix B, C, D**. OpenSesame also provides data logging from scripts which allowed easy extraction of the subjects' answers, outputting all the data to a *.csv* file, ready for analysis. There were several other advantages of using this software. Firstly, it enabled complete separation between the subjects and the experimenters, which kept distractions and biases towards their evaluation of themselves to a minimum. It also provided continuity for the subjects, as well as allowing them to proceed in their own time without missing any information. Furthermore, the subjects were presented with the exact same information in the exact same order every test run. This not only provided consistency for the subjects, but also relieved the experimenters of the task of presenting all this information, freeing time to monitor and to keep an overview of the experiment.



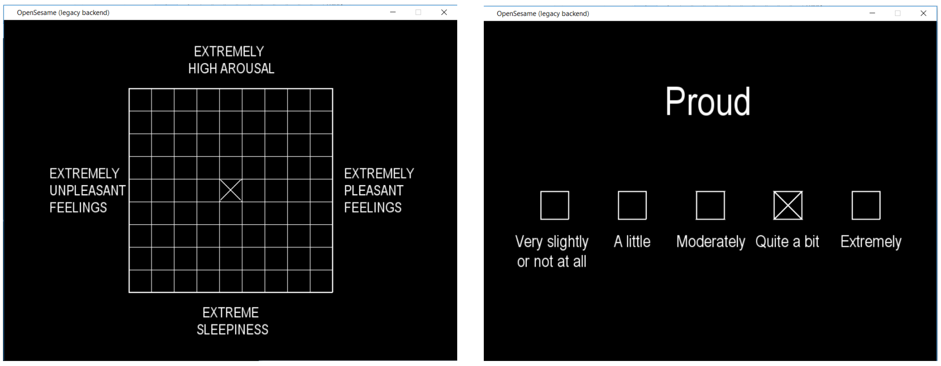
Dette eksperimentet består av tre deler.
Du skal spille TETRIS i hver del.
Du vil bli utsatt for diverse stimuli,
og spillet vil variere i vanskelighetsgrad.

Trykk A når du er klar.

Trykk A når du er klar.

(a) Information **(b)** Instructions

Figure 3.8: The GUI


OpenSesame (legacy backend)
EXTREMELY HIGH AROUSAL
EXTREMELY UNPLEASANT FEELINGS
EXTREMELY PLEASANT FEELINGS
EXTREME SLEEPINESS
OpenSesame (legacy backend)
Proud
Very slightly or not at all
A little
Moderately
Quite a bit
Extremely

(a) The affect grid, as presented to the subjects **(b)** One of 20 words in the PANAS questionnaire

Figure 3.9: Parts of the subjective evaluation screens

3.9 Procedure

This section presents the complete procedure for the experiment, including detailed explanation of the three scenarios (S1-S3) mentioned in the beginning of this chapter. It discusses how and why the various impulses were applied to achieve different levels of affect. As mentioned, the goal was to vary the affect of the subjects in three levels. These were as follows **(Table 3.4, Fig. 3.10)**:

Scenario no.	**Desired level of affect**
S1	Low arousal, Positive valence
S2	High arousal, Negative valence
S3	High arousal, Positive valence

Table 3.4: The three desired levels of affect

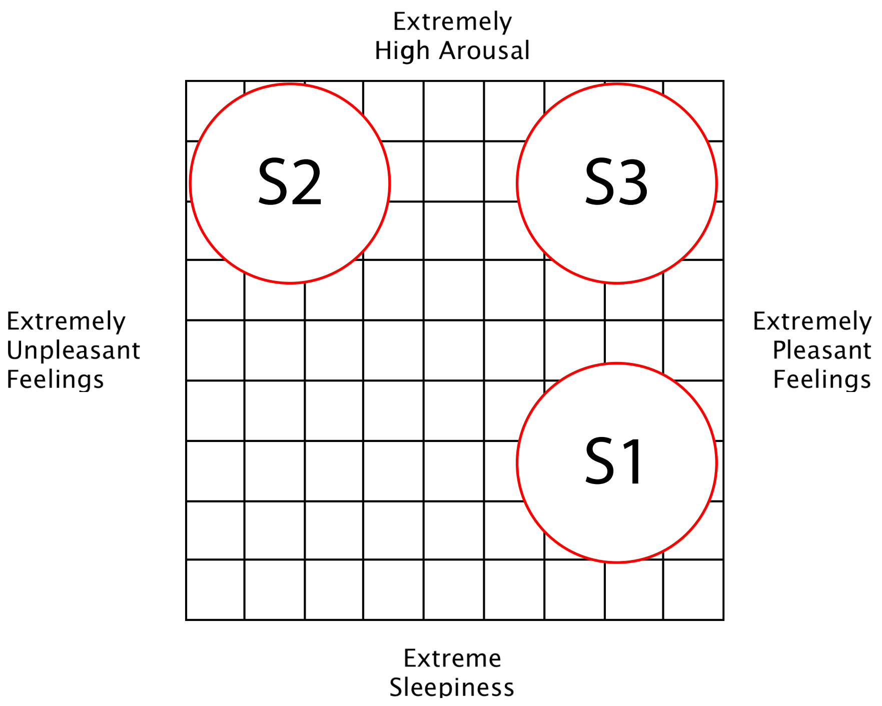


Figure 3.10: The desired placement of the three scenarios in the affect grid.

By achieving these three levels of affect this would have facilitatet both the authors' needs at the same time. The span in valence (S2 and S3) enables analysis according to this dimension and could at the same time confirm the change in physiological reactions also for low arousal (S1). The same goes for analysis of arousal, with span in this dimension (S1 and S3) and confirmation with negative valence (S2).

The scenarios were presented in the same order for every subject, with S1, considered as the "calmest" one, first. The reason for this was to avoid lingering effects from the high arousal scenarios (S2 and S3) into the low arousal scenario (S1), which was experienced during pilot testing. Another observation made in the pilot tests was an elevated arousal level in the beginning of the experiment. Many subjects seemed to get an increased arousal level by being wired up with electrodes and at all be excited, and even a bit nervous, to be a part of a research experiment. This confliced with the desired low arousal state, but the lingering effects of having S2 and S3 first was experienced to have a larger impact on arousal. Each scenario had a duration of 5 minutes, with approximately 2 minutes of questionnaires after each and a two minute break in between to "reset" the emotional state. Including introduction and instructions the experiment had a total duration of about 30 minutes.

3.9.1 Initiation

The subject was welcomed to the experiment by reading and signing a standard consent form **Appendix J**. After this all the nine electrodes were placed on the subject's body. Three in the front upper body area for ECG, three on the right forearm and three on the neck for EMG (**Fig. 3.11**). As both experimenters were males an image was presented to the female subjects, which showed the placement of the ECG electrodes. The correct placement of the electrodes was easily checked as real time sensor data was shown on the sampling computer. For the sake of simplicity the rest of the electrodes were placed by one of the experimenters. The subject was then given the headset and controller. Some information was conveyed personally, such as informing about the two information sheets linked to the affect grid (**Appendix B**). After commencing the experiment the subjects were asked to report their initial state in an affect grid on screen. This was intended both for the subject to get to know the grid, but also to generate a starting point on how this person was feeling before starting.

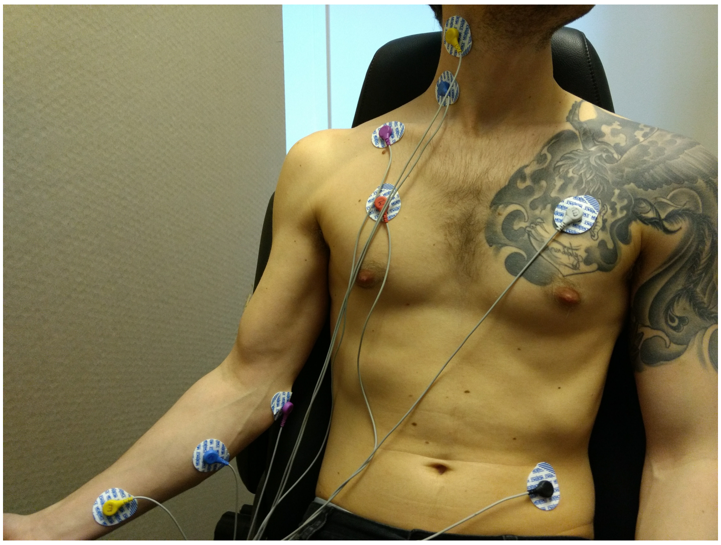


Figure 3.11: Connected sensor electrodes.

3.9.2 Scenario 1

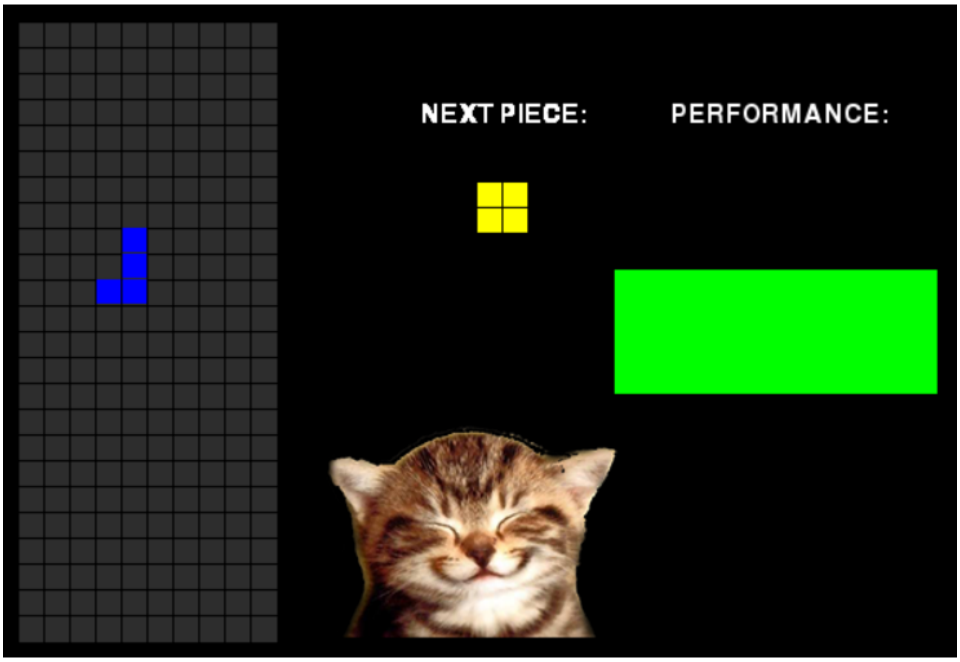


Figure 3.12: TETRIS, scenario 1.

Before starting the first scenario, instructions about the game layout and controls were given to the subjects. The layout of this situation included a pleasant picture of a cat. Cats are well known to be cute creatures and pilot testing indicated that the cat evoked pleasant feelings. In this scenario the layout did not include a score or a counter of how many lines the subjects had accomplished in the game, as the experience was intended to be as calm and as little competitive as possible. A performance bar was placed to the right of the game, actively giving the subjects feedback on their performance. The stimuli used in this scenario were as follows:

Difficulty

The speed of the game was set to a relative low setting, pilot testing showed that this was comfortable for the subjects, and far from being a challenging factor. To achieve the desired relaxed emotional state, the three pieces most of the pilot subjects found challenging were removed, this left only four pieces which gave the game a pleasant and calming effect. The game was too simple and unchallenging for the subjects to get aroused, yet it was fulfilling for the subjects to perform well.

Lights

The lights was set to a steady setting with a warm color and low intensity. The light was not supposed to stimulate on its own, it was a complimentary feature to make the visual experience from the screen more pleasant and less intense.

Music

The music the subjects were presented with in this scenario was slow paced and calming.

Feedback

In this scenario the subjects were only presented with positive feedback. Every time they cleared a line a positive sound was played, and the performance bar only moved in the positive direction, upwards. On the positive side, the bar had a green color, often associated with correct actions. This was done to keep the subjects from getting bored and unfulfilled.

3.9.3 Scenario 2

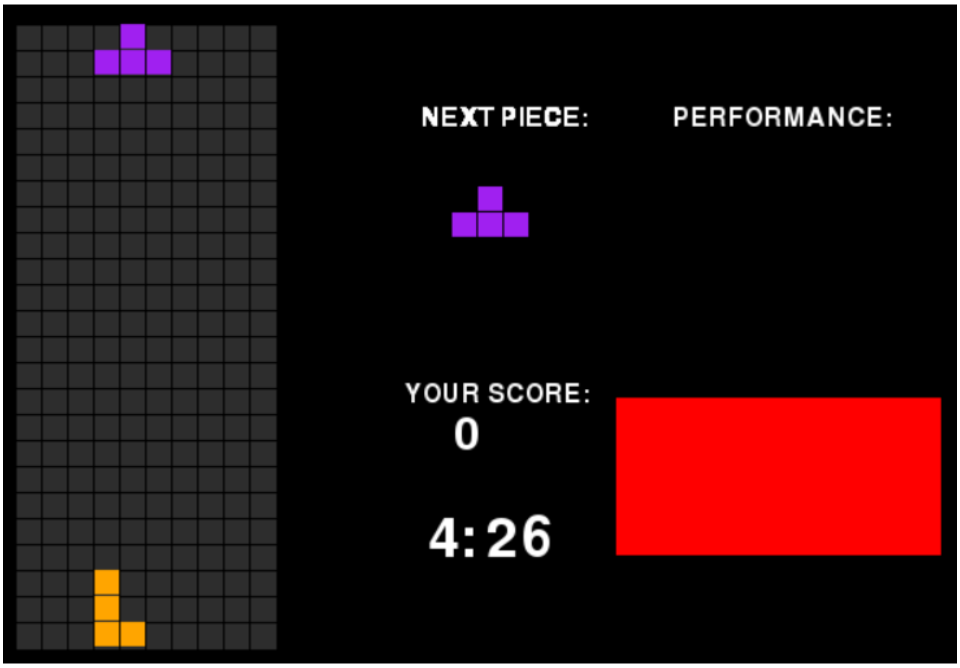


Figure 3.13: TETRIS, scenario 2.

Also before starting the second scenario instructions were given. The layout in this scenario did not include a picture of a cat, but here the score was presented in addition to the time remaining. Both the score and the remaining time were included to evoke the feeling of needing to perform, resulting in the desired arousal level. In addition, the subjects were given information stating that this scenario was a part of a competition, utilizing people's competitive spirit to achieve even higher arousal.

Difficulty

The speed was by default set at a really high level, and most pilot subjects found it to be too fast to be able to perform well. The speed also increased after 3 minutes to a level that even high performing Tetris players would find too high. The input controls were inverted so move right became move left and vice versa. The controls were switched back to normal after two and a half minutes, and then back to inverted after four minutes. This lead to a lot of frustration because when finally the controls were familiarized they were switched again. The pieces in this scenario included all seven pieces, however the order was manipulated in a way so that the three pieces the pilot subjects found most challenging appeared at a higher rate, and the most helpful piece at a much lower rate.

Lights

The lights in this scenario were strobing fast through four colors similar to police car flashers, red, blue, green and white. The intensity was set to the highest possible setting to make it as unpleasant and annoying as possible. This stimuli was kept in spite of one of the pilot subjects feeling it was over the top. Along with the difficulty, the lights appeared to be the most prominent traction stimuli for negative valence in this scenario. To clarify, the subjects were asked if they had any kind of epilepsy in the consent form before participation.

Music

Several kinds of music were subject to pilot tests for this scenario. Eventually a song was chosen based on feedback from pilot subjects. This song was "up-beat", powerful and ominous, similar to the music in a horror movie when the action level is on it's highest.

Feedback

In this scenario the subjects were not presented with any positive feedback, but given negative feedback both by an intense and negative buzz sound and the performance bar moving in the negative direction (downwards) when performing bad. On the negative side the bar had a red color, often associated with incorrect actions. Because of the scenario having a quite high level of difficulty, these negative feedback aspects were presented with high frequency, giving the subjects an even more negative experience.

3.9.4 Scenario 3

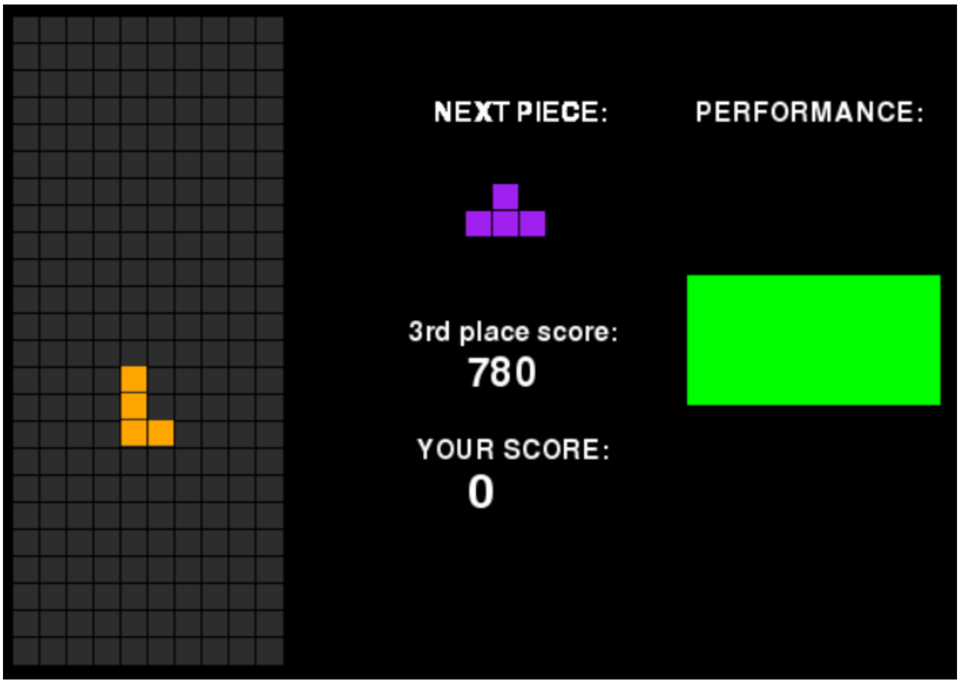


Figure 3.14: TETRIS, scenario 3.

Before starting the third scenario the subjects were again presented with the instructions and layout of the scenario. This time they were presented with the instruction about this scenario being part of a competition two times to make an even call to the competitive spirit to increase arousal levels. In this scenario the layout included a highscore list, starting at third place, this is more explained in the feedback description below.

Difficulty

The game speed wass set relatively high, but still achievable. The thought was to let the subjects make it, but only barely, to achieve a maximum positive valence and high-arousal effect. By giving the subjects a good fight the intention was for them to achieve an even greater feeling of accomplishment when they made it.

Lights

The lights in this scenario were intended to increase arousal without compromising positive valence. Though the flashing lights in S2 were perceived as very negative due to the fast flashes, flashing colors in a slower pace seemed to avoid getting negative valence. Quite the opposite, this switching of colors seemed to only reinforce positive feelings as the scenario in itself was positive. The intervals between each color was set to 961 ms to accompany the beat of the music.

Music

Several songs were tested for this scenario before the final version of the experiment, and various up-beat instrumental songs were perceived as only moderately arousing. What seemed to do the trick was the association people had with the song, which could often push them towards both higher arousal and more positive valence at the same time. This is possibly a bit risky, as associations could differ greatly from person to person. To minimize this risk, a song that lies in the hearts of many 90s children's memories is chosen, as the majority of the participants in the experiment was born in the early 90s. The song is called "Sandstorm" and is performed by Darude.

Feedback

The feedback of this scenario was exclusively positive. The bar was used in the same way as in S1, going up for each line taken, whilst playing a positive "ping" sound. In addition to this a highscore list was shown. As in **Fig. 3.14**, the game started by showing the 3rd place score. When the subject passed this score, the screen turned black with the text *"YOU ARE NOW IN 3RD PLACE!"* flashing in blue and green in the center, and a corresponding "level up" sound was played. The score on the right changed to show the 2nd place score, and further to 1st place score, also with a flashing text and sound in between. When the 1st place score was beaten, the text *"YOU ARE NOW IN THE LEAD!"* was shown. After recommencing the game, this text was also shown to the right instead of "x place score". The indications of increasing positioning on the highscore list evolved from being just the subtle change in the text to the right to a more pretentious and explicit notification, as subjects were often too deep into the game to notice this change.

Chapter 4

Results

This chapter presents the results from the data gathering conducted during the experiment. The data will first be shown in a descriptive format to give an overview of the different variables, and will furthermore be analyzed to test and evaluate the mentioned hypotheses. The total number of subjects participating in the experiment was 34, and the experiment was conducted from the 11th to the 18th of may. Since the experiment ran through several iterations of pilot testing, both with numbered and unnumbered subjects, the subject numbers span from 6 to 39. 27 of the subjects were males and 7 were females. 19 subjects were between 20 and 25 years of age and 15 between 26 and 30. Most of the subjects were recruited from the nearby university environment. The dataset can be viewed in its full form in **Appendix G**.

4.1 Descriptive results

The intention of this section is to give a brief overview of the measurements separately, from scenario to scenario. As mentioned, there were three of them, S1, S2 and S3. S1 was intended to be *positive valence - low arousal*, S2 *negative valence - high arousal* and S3 *positive valence - high arousal*. This allows for analyses of the data according to the two dimensions, and better observe which dimension that plays a role in the physiological measurements. The three scenarios yielded three affect states for each subject, which leaves a total of 102 measurements. In every graphical representation separating the scenarios from each other, S1 will have a green, S2 a red and S3 a blue color. If nothing else is specified, the EMG neck/arm and leaning distance measurements are measured in their respective average value in each scenario.

4.1.1 Affect grid

When using the Aaffect Grid (Russel et al., 1989), the subjects rates their experienced affect levels in the valence and arousal dimension on a scale from 1 to 9 in both dimensions. 1 valence represent *extremely unpleasant feelings*, and 9 *extremely pleasant feelings*. 1 arousal represent *extremely high arousal* and 9 *extreme sleepiness*. A summary of the scores is presented in **Table 4.1**. A heatmap of the results from the affect grid are shown in **Fig. 4.1**, with S1 in green, S2 in red and S3 in blue colors. Darker colors represent greater concentrations. Each participant's response in the valence dimension is shown in **Fig. 4.2**, and the distribution is shown in a boxplot in **Fig. 4.3**. In this plot, the mean value is symbolized with an X and the median with a line across the box. The box in itself contains 50 percent of the data.

	Valence			**Arousal**		
Scenario	**S1**	**S2**	**S3**	**S1**	**S2**	**S3**
Mean	6.97	3.50	6.85	4.47	2.48	2.79
Median	7	3	7	4	3	3
Std. dev.	1.15	1.42	1.17	1.40	1.18	0.87
Minimum	3	1	4	2	1	1
Maximum	9	8	9	8	6	4

Table 4.1: Affect grid valence and arousal. All data with N=34.

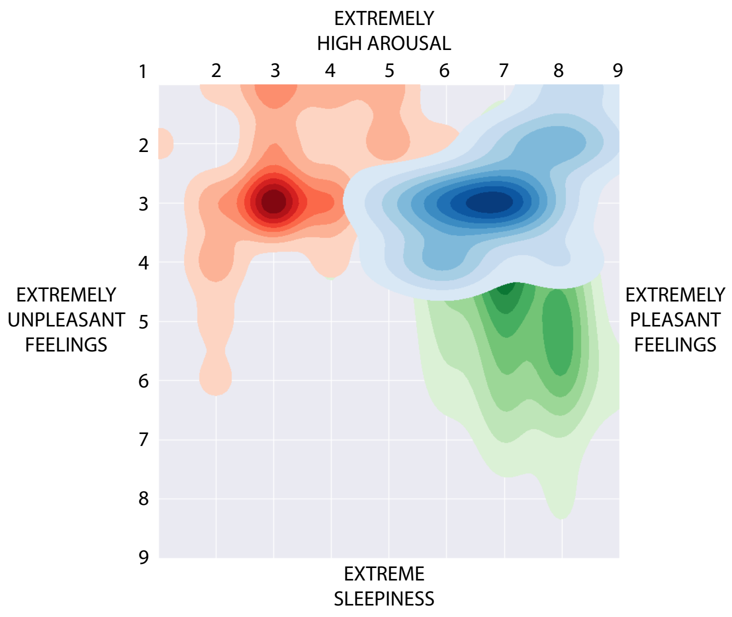


Figure 4.1: Affect Grid in all scenarios

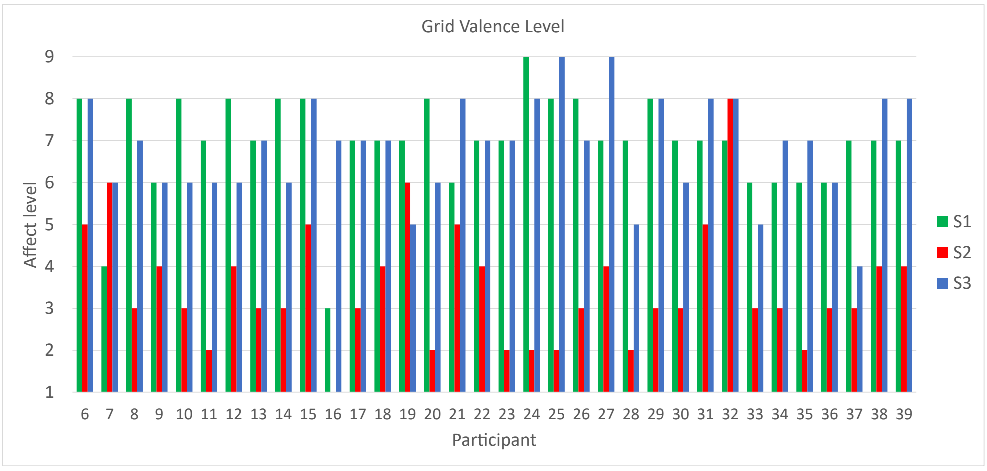


Figure 4.2: Valence level from the Affect Grid

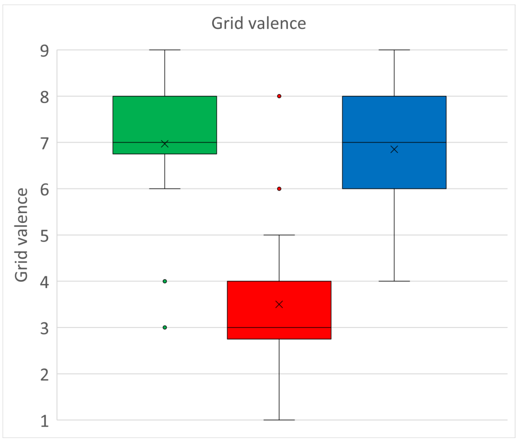


Figure 4.3: Valence level from the Affect Grid

4.1.2 PANAS

The *Positive and Negative Affect Schedule* (PANAS) (Watson et al., 1988) was used as a countermeasure for valence. As mentioned earlier, the PANAS consists of 20 words (**Tab. 3.2**) which describes different feelings. The subjects were to rate to what extent they had this feeling during the experiment, on a 5-point scale from *"Very slightly or not at all"* to *"Extremely"*, which corresponds to a value of 1 and 5, respectively. The scores are then added up to a PA and NA value, which yields a minimum and maximum value of 10 and 50. The higher PA the more positive affect and the higher NA the more negative affect. Since it is convenient to have only one value to relate to when conducting statistical analyses, a *total* PANAS value is also calculated and used in further investigation of the data. This scale has a minimum and maximum score of ±40, and is referred to as *PA-NA*. A summary of these scores is presented in **Table 4.2**. Their distributions are presented in a boxplot in **Fig. 4.4**.

	S1			**S2**			**S3**		
	PA	**NA**	**PA-NA**	**PA**	**NA**	**PA-NA**	**PA**	**NA**	**PA-NA**
Mean	31.6	16.6	15.1	30.7	26.3	4.4	36.8	16.6	20.2
Median	31.5	16.0	15.0	32.5	26.5	3.0	37.0	16.0	20.5
Std. dev.	7.0	5.5	7.1	6.6	6.3	9.2	5.8	4.5	6.9
Minimum	18	10	-1	18	10	-18	24	10	6
Maximum	44	37	29	48	37	24	48	26	38

Table 4.2: PANAS scores, PA, NA and PA-NA. All data with N=34.

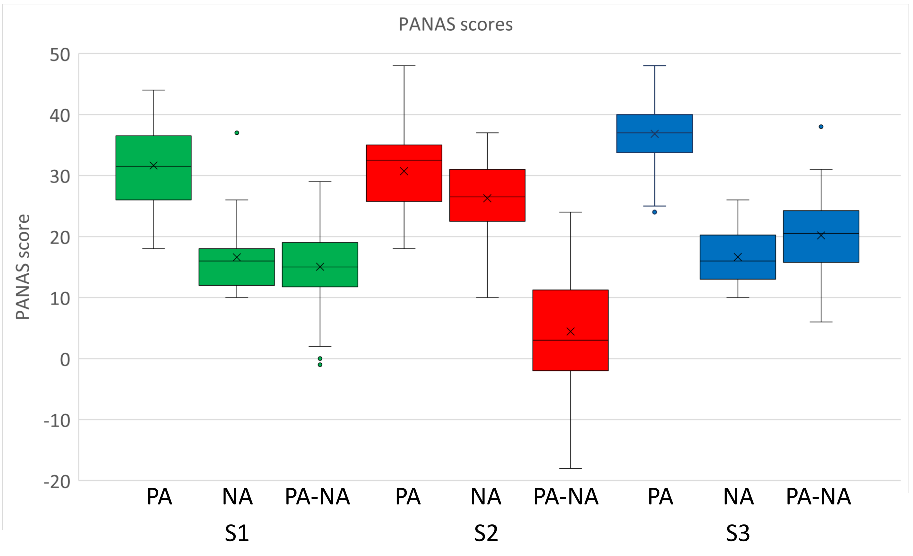


Figure 4.4: PANAS PA and NA scores for each scenario.

4.1.3 EMG

The raw EMG data was collected as datapoints over a time series during the whole experiment. Each scenario lasted 5 minutes, and the sampling frequency was around 380Hz. This resulted in about 114 000 data points for each of the scenarios. Since the valence values from the Affect Grid and PANAS represent the overall valence level in one scenario, the EMG values used for investigation of the hypotheses are mean values of these data points. Normally, the EMG is measured in Volts as the potential difference between two electrodes. However, the sensory setup for this experiment was designed to allow good flow and synchronization of data. For this purpose, the measured EMG values were sent from the EMG "slave" Arduinos through a simple RC circuit and read by the "master" Arduino as an analog signal (Section 3.7). The relation between the signal through the RC circuit compared to a raw input signal recorded directly from the "master" Arduino was tested and found to be alike, only with an offset:

$$RecordedEMG = OriginalEMG + constant \tag{4.1}$$

This was done by recording EMG data on the arm over a time period of about 20 seconds with three different degrees of muscle tension. First with complete relaxation, second holding a 0.5kg weight in the hand and third holding a 1kg weight in the hand. This is illustrated in **Fig. 4.5**, and the results from the test are presented in **Tab. 4.3**. From the table one can observe that the internal differences between the EMG values through the circuit and from direct input are reasonably similar. As far as accuracy with the use of surface electrodes and the way this was tested goes, it can be concluded that the two signals are of the same nature, that **Equation 4.1** can be verified, and that the constant in this equation is about -148. Furthermore, the unit of the data is in mV (millivolts). For the sake of not corrupting the data, it is treated "as is", that is, in it's original value. Although, it is nice to know that this relation is constant and that the data makes sense.

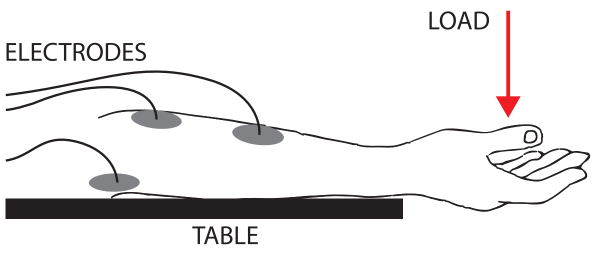


Figure 4.5: EMG calibration.

	Via circuit	Difference	Direct input	Difference
Relaxation (0 kg)	200.3	0	52.1	0
0.5 kg	206.4	**+6.1**	57.6	**+5.5**
1.0 kg	212.8	**+6.4**	63.6	**+6.0**

Table 4.3: EMG calibration for the RC circuit vs direct input from the Arduino.

The levels of EMG were experienced to vary from subject to subject, which may be caused by several factors, e.g. the placement of the electrodes and the thickness of the subjects' subcutaneous tissue layers (Farina et al., 2004), which heavily depend on the weight and physiology of the subject. The placement of the electrodes was not something that was accurately monitored during the experiment, which resulted in large variance in the data. However, the interesting effect to be analyzed in this thesis is each subject's individual change in muscle tension. Hence the between-subjects variation is not as much of importance. Worth noting is that one subject, no. 27, showed abnormally high EMG values. The values for this subject were around 600 mV, while the average median for all subjects was around 180 mV. Even though the mentioned RC circuit increased the EMG value by abound 148 mV, a value of 600 mV is unlikely. In addition, the subjects were asked to tense their neck and arm muscles after connecting the electrodes, and values rarely exceeded 250 mV. Also, during the experiment with this subject it was observed that the neck EMG and arm EMG values were running in phase, as if they were almost completely dependent on each other. Though it is not certain what exactly generated these high values, this is highly likely an error in the measurements, and of this reason subject 27's EMG values is removed from the dataset.

Neck EMG

A summary of the neck EMG values is represented in **Tab. 4.4**. **Fig. 4.6** shows an overview of the average neck EMG values from the dataset for each participant, separated on S1, S2 and S3. To enable overview in terms of variation in the data in each scenario, the data is also presented in a boxplot in **Fig. 4.7**.

	S1	S2	S3
Mean	188.9	189.5	189.9
Median	183.2	182.5	184.8
Std.dev.	17.3	17.7	17.5
Minimum	167.8	168.4	167.7
Maximum	224.3	232.7	227.1

Table 4.4: EMG neck values for S1, S2 and S3 in mV. All values with N=33.

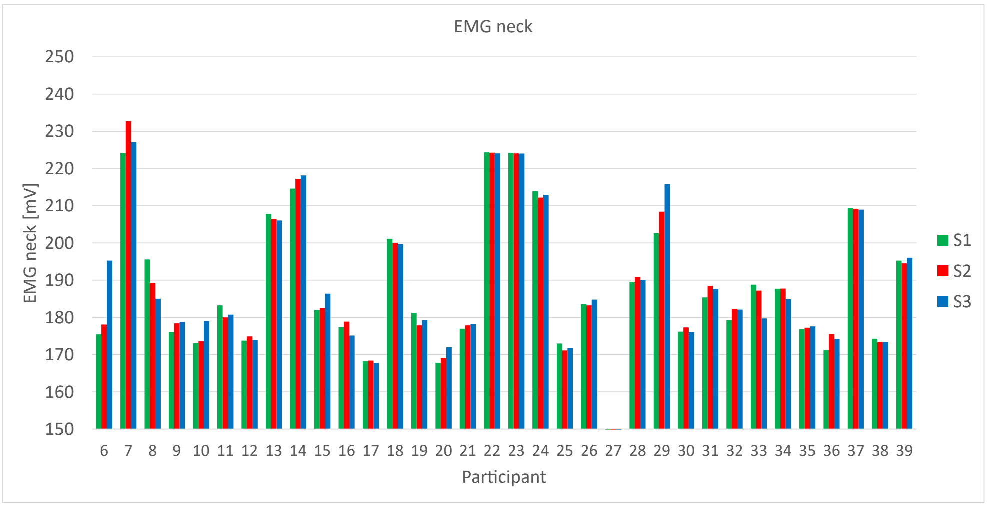


Figure 4.6: Neck EMG values for all participants, except from subject 27, in all scenarios

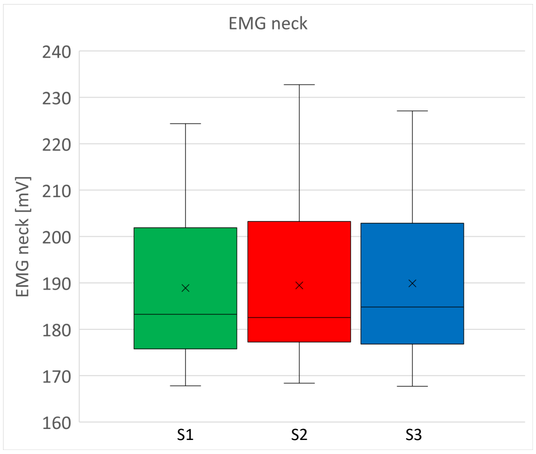


Figure 4.7: Boxplot of EMG neck values in the three scenarios.

Arm EMG

A summary of the arm EMG values is represented in **Tab. 4.5**. Subject 27's EMG values are still left out of the dataset. Similar to neck EMG, all remaining arm EMG values in the three scenarios are shown in **Fig. 4.8**. Further, the distributions of the EMG arm values are shown in a boxplot in **Fig. 4.9**.

	S1	S2	S3
Mean	204.7	204.6	207.1
Median	202.4	201.2	202.7
Std.dev.	15.5	15.1	16.7
Minimum	186.5	184.0	184.1
Maximum	248.9	248.2	248.0

Table 4.5: EMG arm values in S1, S2 and S3. All values with N=33.

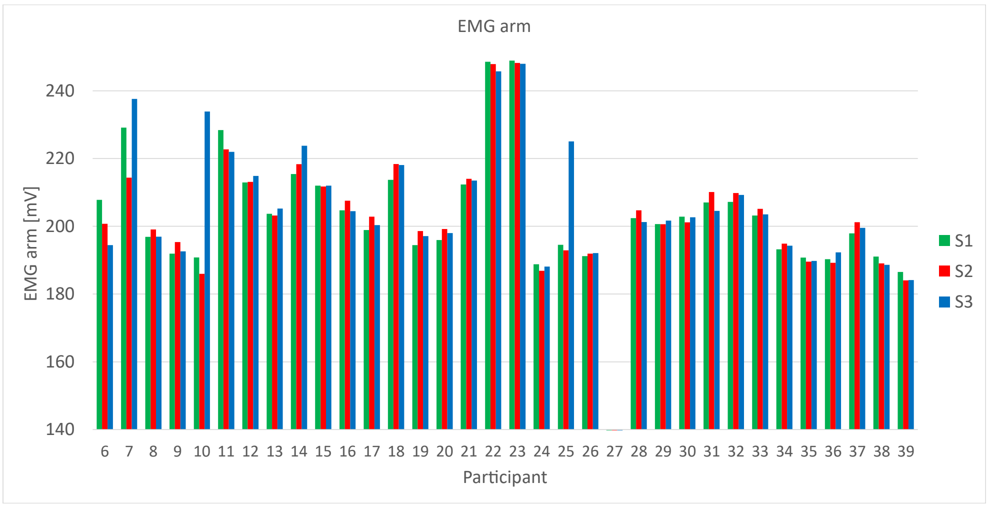


Figure 4.8: Arm EMG values for all participants, except from subject 27, in all scenarios.

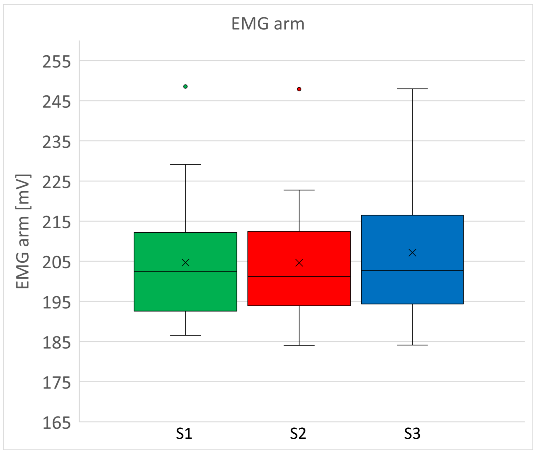


Figure 4.9: Boxplot of arm EMG values in the three scenarios

4.1.4 Leaning distance

The leaning distance input was measured by the Lidar distance sensor behind the head of the subjects. Of practical reasons the sensor was placed at a base distance of about 6 cm behind the head when a person rests the head on the back of the chair. A summary of the leaning distance data is shown in **Tab. 4.6**. A barchart for the leaning distances of all subjects in all scenarios is shown in **Fig 4.10**. The distribution for each scenario is shown in a boxplot in **Fig. 4.11a**. Note that subject 27 now is back in the dataset, and that this data now contains all 34 subjects. Worth noting is the decrease in leaning distance in S2 compared to S1 and S3. As shown in **Fig. 4.1**, this is also the scenario where the valence was significantly different from the other two. Taken into account the base distance from the sensor to the back of the head, the subjects leaned forward between 1 and 26cm.

	S1	**S2**	**S3**
Mean	19.3	18.4	19.4
Median	18.8	18.8	19.4
Std.dev.	4.5	5.0	5.3
Minimum	10.9	6.5	10.6
Maximum	30.1	29.0	32.0

Table 4.6: Leaning distance values in S1, S2 and S3 in cm. All values with N=34.

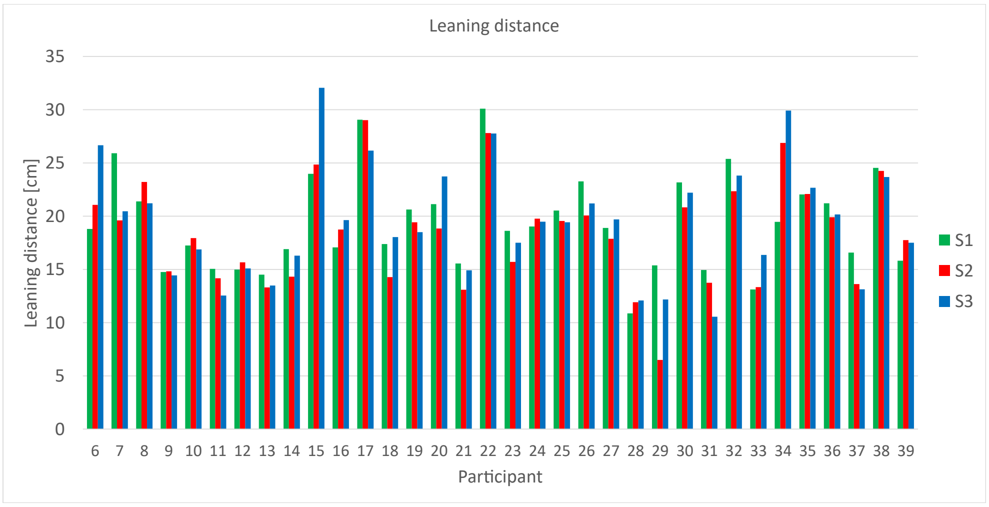
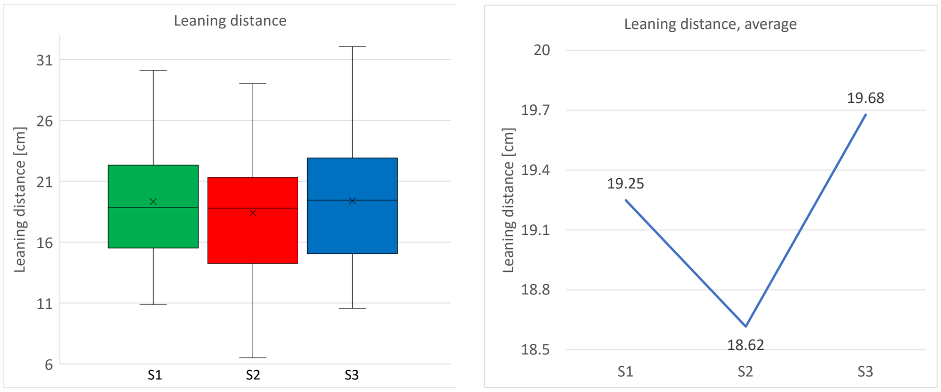


Figure 4.10: Leaning distance values in cm for all participants in all scenarios.



(a) Leaning distance boxplot **(b)** Average leaning distance in scenarios

Figure 4.11: Boxplot and linechart of leaning distance values in the three scenarios.

4.2 Statistical evaluation

The intention of this section is to use statistical analysis methods to relate the physiological data presented earlier to the valence data from the Affect Grid and PANAS, with the ultimate goal of evaluating the hypotheses. First there will be an evaluation of the subjective measurements, the Affect Grid and PANAS, and further the hypotheses will be tested and evaluated. Here, statistical tests will be run to check for statistical significant difference in the EMG neck/arm and leaning distance variables between the scenarios and, if there is, further tests will be run to determine the relationship between the variables and the valence measurements.

The "one-way repeated measures analysis of variance (ANOVA)" test is used to determine if there is a statistically significant difference between population means of three or more levels of a within-subjects factor. The levels are related because they contain the same subjects. The test has the following requirements:

1. One dependent continuous variable which is measured repeatedly under three or more different conditions

2. One within-subjects factor (independent variable) that consists of three or more categorical levels

3. There should be no significant outliers in the three or more levels

4. The distribution of the dependent variable should be approximately normally distributed in the three or more levels

5. The variances of the differences between all combinations of levels of the within-subjects factor must be equal (known as the assumption of sphericity)

In this context, EMG neck/arm and leaning distance values represent the dependent variable and the scenarios represent the conditions. The rest of the assumptions will be tested during the test in the variables' respective sections.

4.2.1 Evaluating the Grid and PANAS

No matter how established and validated subjective measurements are, it is interesting to see how these apply to an experiment like this. Especially how subjects relate to these types of questionnaires and observe how they rate themselves. To evaluate the use of these measurements in this particular experiment, a search for a relationship between them was conducted. This was done by doing a Mantel-Haenszel test of trend, which tests the linear association between two ordinal variables. It basically runs a Pearson correlation on the two variables and uses this to calculate the test statistic, given by **Equation 4.2**. Here, r is the Pearson correlation coefficient and n is the sample size. The main outputs of this test are the Pearson correlation coefficient, r, the significance of difference between the means, sig. (2-tailed), and the significance of the linear association between the variables, Asymp. Sig (2-sided). Since both the grid and PANAS variables are ordinal and the association between them is expected to be linear, both criteria for conducting this test are satisfied. In this test, all the 102 datapoints for affect level was used. The test was conducted between

Grid valence level and all variables of the PANAS, PA, NA and PA-NA. Their correlations are presented in **Table 4.7** and results from the Mantel-Haenszel test of trend are presented in **Table 4.8**. Significant relations are highlighted in yellow.

$$[H]M^2 = (n-1) * r^2 \tag{4.2}$$

		PA	**NA**	**PA-NA**
Grid valence	Pearson correlation	0.407**	-0.570**	0.678**
	Sig. (2-tailed)	2.2E-5	4.0E-10	5.3E-15
	N	102	102	102

Table 4.7: Affect grid valence and PANAS correlations. Significant results marked in yellow.

		Value	**df**	**Asymptotic Significance (2-sided)**
PA	Pearson Chi-square	219.991	216	0.412
	Likelihood Ratio	164.309	216	0.996
	Linear-by-Linear Association	16.750	1	4.3E-5
	N	102		
NA	Pearson Chi-square	249.701	200	.010
	Likelihood Ratio	186.486	200	.745
	Linear-by-Linear Association	32.812	1	1.0E-8
	N	102		
PA-NA	Pearson Chi-square	331.604	296	.076
	Likelihood Ratio	221.049	296	1.000
	Linear-by-Linear Association	46.360	1	9.8E-12
	N	102		

Table 4.8: Results from Mantel-Haenszel test of trend. Significant results marked in yellow.

These results imply that there is indeed a relation between the measured Grid valence values and the three PANAS variables. From the correlation table we can observe that the correlations are all statistically significant at the $p < 0.01$ level. The PA, NA and PA-NA correlations are moderate and strong with $r = 0.407$, $r = -0.570$ and $r = 0.678$, respectively. The Mantel-Haenszel test of trend showed a statistically significant linear association between the Grid values and all the PANAS variables. Higher positive Grid valence were associated with higher PANAS scores (except for NA, where there is a negative correlation, logically). This can also be viewed graphically in **Fig. 4.12**, where a fit line that represents the trend is added into the plot. What the test does not show is how well the subjects were able to rate their own affect level. However, this is always an uncertainty when working with subjective measurements, which is why this experiment was conducted, to explore potential objective measures of valence.

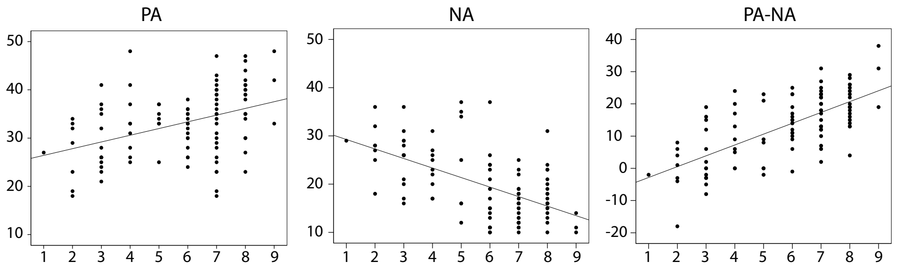


Figure 4.12: Grid valence vs PANAS scatterplot, with all 102 datapoints. From left: PA, NA, PA-NA.

4.2.2 Testing the EMG hypothesis: H1

As EMG data was recorded on both neck and arm, this hypothesis will be tested and evaluated for each of these measurements separately.

Neck EMG

As can be reviewed in **Fig. 4.7**, the EMG neck data had no significant outliers after removing subject 27. A Shapiro-Wilk test of normality revealed that the data was not normally distributed in the three scenarios ($p_{S1} = .002, p_{S2} = .001, p_{S3} = .001$, where the requirement is $p > .05$). Due to the violation of the requirement of normality, this test could not be conducted. Instead, a Friedman test was run. This is the non-parametric alternative to the one-way repeated measures ANOVA test, and it is used when the requirement of normality is violated. This test introduces a NULL and an Alternative hypothesis, H_0 and H_A:

H0: The distribution of EMG neck values in each scenario are the same.
HA: At least two of the groups' distributions differ (possibly with respect to location; e.g. median)

The results from the Friedman test showed that the EMG neck values decreased from S1 ($Mdn = 183.2$) to S2 ($Mdn = 182.5$), and increased to S3 ($Mdn = 184.8$), but the differences were not statistically significant ($\chi^2(2) = .545$, $p = .761$). Hence, there was no statistically significant difference in EMG neck values between the scenarios.

As there was no difference between all three scenarios, the author wanted to test also if there was any difference between the scenario pairs, that is S1 vs S2 and S2 vs S3. For this, a paired samples T-test was run. The T-test, similarly to the ANOVA, determines if the difference between observations is different from zero, only between a pair of observations. In this test, the author took some liberty in removing extreme outliers, as this was but a supplementary test to check for difference between pairwise scenarios. The results are shown in **(Tab. 4.9 and Tab. 4.10)**As can be observed, the valence difference elicited an increase of $0.554 \pm .474$ mV [mean$\pm$ standard error] from S1 to S2 and a decrease of $-0.094 \pm .402$ mV from S2 to S3, but was not statistically significant, $t(32) = .1.170, p = .251$ for S1 vs S2 and $t(29) = -.234, p = .817$ for S2 vs S3.

		Mean	N	Std. Dev	Std. Err. Mean
Pair 1	2EMGn_Mean	189.457	33	17.701	3.081
	1EMGn_Mean	188.903	33	17.288	3.009
Pair 2	3EMGn_Mean	189.185	30	17.605	3.214
	2EMGn_Mean	189.279	30	18.128	3.309

Table 4.9: Descriptive results paired samples T-test, Neck EMG

	Mean	SD	Std.Err.Mean	Lower	Upper	t	df	Sig.(2-tail)
2EMGn - 1EMGn	.554	2.723	.474	-.411	1.520	1.170	32	.251
3EMGn - 2EMGn	-.094	2.206	.402	-.917	.729	-.234	29	.817

Table 4.10: Paired samples T-test, S1 vs S2 and S2 vs S3, Neck EMG

Arm EMG

In the EMG arm variables, two outliers were discovered, subject 22 and 23. As can be reviewed in **Fig. 4.8**, these subjects' values were higher than the rest in all three scenarios. It is suspected that this error is due to EMG measurement factors, discussed in **Section 4.1.3**, and has to do with placement of electrodes, subcutaneous tissue layers etc. Furthermore, these subjects' values were then assumed to be equally higher than the rest in all three scenarios. Thus, these outliers were dealth with by subtracting their values with 34, which is approximately 10 less than their difference from the mean EMG arm value for all subjects in S1. This way, they ended up just above the mean values for all subjects, maintaining their rank, and their difference between the scenarios were maintained, which was the interesting factor for this analysis. Now there were no outliers and the data was normally distributed in each scenario, as assessed by boxplot and Shapiro-Wilk test of normality ($p_{S1} = .055, p_{S2} = .438, p_{S3} = .168$, where the requirement is $p > .05$). The assumption of sphericity was not met, as assessed by Mauchly's test of sphericity, $\chi^2(2) = 35.098$, $p = 2.4E - 8$ (requirement: $p > .05$). Epsilon (ε) was 0.596, as calculated according to Greenhouse and Geisser (1959), and was used to correct the one-way repeated measures ANOVA for the absence of sphericity in the data. The test did not elicit statistically significant changes in EMG arm values between the scenarios, $F(1.192, 38.148) = 1.774$, $p = .191$, with EMG arm values decreasing from S1 ($M = 202.61$, $SD = 11.05$ mV) to S2 ($M = 202.57$, $SD = 10.52$ mV), and increasing to S3 ($M = 205.08$, $SD = 13.30$ mV).

The paired samples T-test was run for the same reason as in the case of EMG neck. The data was normally distributed after outliers were removed. The results are shown in **(Tab. 4.11 and Tab. 4.12)**. As can be observed, the valence difference elicited an increase of $0.411 \pm .516$ mV [mean$\pm$ standard error] from S1 to S2 and a decrease of $-0.681 \pm .433$ mV from S2 to S3, but was not statistically significant, $t(31) = .797, p = .432$ for S1 vs S2 and $t(29) = -1.574, p = .126$ for S2 vs S3. The increase previously observed in the ANOVA from S2 to S3 ($M = 202.57$ for S2 and $M = 205.08$ for S3), was not apparent in this test, which is a consequence of removing the 4 outliers in this part of the T-test.

		Mean	N	Std. Dev.	Std. Err. Mean
Pair 1	2EMGa	204.324	32	15.218	2.690
	1EMGa	203.912	32	15.147	2.677
Pair 2	3EMGa	204.633	30	15.250	2.784
	2EMGa	205.315	30	15.180	2.771

Table 4.11: Descriptive results paired samples T-test, Arm EMG

	Mean	SD	Std.Err.Mean	Lower	Upper	t	df	Sig.(2-tail)
2EMGa - 1EMGa	.411	2.919	.516	-.641	1.463	.797	31	.432
3EMGa - 2EMGa	-.681	2.372	.433	-1.567	.204	-1.574	29	.126

Table 4.12: Paired samples T-test, S1 vs S2 and S2 vs S3, Arm EMG

4.2.3 Testing the leaning distance hypothesis: H2

As with the EMG data, the one-way repeated measures ANOVA test was conducted also for leaning distance. The dataset was here used with all subjects, N=34. There were no outliers and the data was normally distributed in each scenario, as assessed by boxplot (**Fig. 4.11a**) and Shapiro-Wilk test of normality ($p_{S1} = .397$, $p_{S2} = .619$, $p_{S3} = .572$, requirement: $p > .05$). The assumption of sphericity was met, as assessed by Mauchly's test of sphericity, $\chi 2(2) = 5.821$, $p = .054$ (requirement: $p > .05$). The test did not elicit statistically significant changes in leaning distance values between the scenarios, $F(2, 66) = 2.466$, $p = .093$, with leaning distance values decreasing from S1 ($M = 19.33$, $SD = 4.51$ cm) to S2 ($M = 18.42$, $SD = 4.96$ cm), and increasing to S3 ($M = 19.39$, $SD = 5.29$ cm).

The paired samples T-test was run as in the case of the EMG variables. The data was normally distributed after outliers were removed in S1 vs S2. In S2 vs S3, all datapoints were used. The results are shown in **(Tab. 4.13 and Tab. 4.14)**. As can be observed, the valence difference elicited a decrease of $-0.922 \pm .340$ cm [mean$\pm$ standard error] from S1 to S2 and an increase of $0.975 \pm .0.410$ cm from S2 to S3, and both cases were statistically significant, $t(31) = -2.711, p = .011$ for S1 vs S2 and $t(33) = 2.376, p = .023$ for S2 vs S3.

		Mean	N	Std.Dev.	Std.Err.Mean
Pair 1	2Lean	18.525	32	4.396	.777
	1Lean	19.448	32	4.598	.812
Pair 2	3Lean	19.392	34	5.298	.908
	2Lean	18.417	34	4.963	.851

Table 4.13: Descriptive results paired samples T-test, Leaning distance

	Mean	SD	Std.Err.Mean	Lower	Upper	t	df	Sig.(2-tail)
2Lean-1Lean	-.922	1.925	.340	-1.616	-.228	-2.711	31	.011
3Lean-2Lean	.975	2.392	.410	.140	1.809	2.376	33	.023

Table 4.14: Paired samples T-test, S1 vs S2 and S2 vs S3, Leaning distance

4.3 Evaluation of hypotheses

The statistical tests run in the previous section were done to explore potential relations in the physiologic data to valence. The results will be evaluated in this section.

4.3.1 H1: EMG neck

A one-way repeated measures ANOVA was attempted to analyze the EMG neck data. The requirement of normality was not met, hence the non-parametric Friedman test was run. The test elicited an insignificant result, $p = .761$, which is far away from the significance level of $p < .05$. Thus, the null hypothesis for this test (H0), that suggests that the distribution of EMG neck values in each scenario is the same, is retained. The T-test was run as a supplementary test to explore the difference between the pairs of scenarios, S1 vs S2 and S2 vs S3. Again, no significant difference was detected, $p = .251$ between S1 and S2, and $p.817$ between S2 and S3. Due to the lack of difference between scenarios, the null hypothesis for this thesis, in terms of neck muscle tension, is kept. Qualitatively, one can see from the median data from the Friedman test for each of the EMG neck variables (183.2, 182.5 and 184.8 mV for S1, S2 and S3, respectively) that the differences between them are minor (-0.7 mV from S1 to S2 and +2.3 mV from S2 to S3). Considering the somewhat unreliable aspect of the data when measuring muscle tension with surface electrodes, it is hard to favor this hypothesis with such small differences.

4.3.2 H1: EMG arm

The same ANOVA test was run on the EMG arm data. The requirement of normality was met. The requirement of spericity was not met, and this was corrected by Epsilon (ε). The result from the test was not statistically significant ($p = .191$), and hence the null hypothesis for this test, that says that the population means in the EMG arm data in the three scenarios are equal, is retained. Qualitatively, one can see from the mean data from the test (202.6, 202.6 and 205.1 mV for S1, S2 and S3, respectively) that the values in S1 and S2 are identical. There is however a change to S3, which could indicate a difference in arm EMG between 'negative excitement' (S2) and 'positive excitement' (S3). The T-test was run to explore this further, but showed no significant difference, $p = .432$ between S1 and S2, and $p.126$ between S2 and S3. There was taken some liberty regarding outliers in this test, which after inspection of the mean data was the cause of the difference first detected from the ANOVA. Due to the lack of difference between scenarios, the null hypothesis of this thesis, in terms of arm muscle tension, is kept.

4.3.3 Leaning hypothesis

The same ANOVA test was run on the leaning distance data. Here, all the requirements for the test were fulfilled, and all the data points were used. The result from the test was not statistically significant ($p = .093$), and hence the null hypothesis for this test, that says that the population means in the leaning distance data in the three scenarios are equal, is retained. Although this difference is not statistically significant, it is not too far away from it, and the measurements show tendencies. Mean leaning distance in each scenario (19.3, 18.4 and 19.4 cm for S1, S2 and S3, respectively) show increasing distance with more positive valence, also independent of arousal. This change was explored further in the T-test, which showed that there was indeed a change in leaning distance, both from S1 to S2 ($p = .011$) and S2 to S3 ($p = .023$). However, the effect size (d, calculated as Mean/Std.Dev. (Cohen, 1988)) is merely moderate in size, $d_1 2 = .48$ and $d_2 3 = .41$. Considering the accuracy of the distance sensor as well, no conclusions can be made in the favor of this hypothesis based on this data. Due to the lack of difference between scenarios from the ANOVA and the small change compared to sensor accuracy from the T-test, the null hypothesis of this thesis, in terms of leaning distance, is kept.

This page is intentionally left blank.

Chapter 5

Limitations and evaluations

After conducting this experiment, several aspects are subject to improvement. Furthermore, the experiment was made as a prototype on how one can specifically target and measure user interactions and explore how human emotions are expressed through physiological reactions. This chapter aims to reflect on these points and present thoughts the author has towards what worked well and what didn't.
The physical setup of this experiment was designed focusing on simplicity to explore how one can apply iterative product development methods in a scientific experiment setting, creating prototypes in a resource- and time-efficient way. The materials used were simple. This resulted in a flexible experiment environment that could be and was changed several times during pilot testing. Cardboard walls were used to set the layout of the room, which effectively isolated the subjects from the experimenters and made a small personal space without any physical or visual interference. That said, when working with humans there is more to this term than physical or visual disturbance, and the never-ending paradox of measuring without affecting is delicate. The cardboard wall set up functioned as a physical separator, but the subjects also knew, due to the obviously small area of the room, that the experimenters were right on the other side, watching. Considering the little separation and that affection was observed to some degree in some subjects, the author believes that more separation and distance could contribute to more elbow space for subjects, evidently providing more real data. This is something that should be taken into consideration when doing this kind of research.

Continuing with the computer setup, this was immense but well-functional. For the sake of having complete control of the experimental environment, one computer was used for data collection and one for the user interface. Considering the amount of data collected and the desire to have real time control of the data, this was a wise decision. Having the sensor data real time on screen made it easy to ensure that all the sensors and electrodes were functional, and if not, correct it before commencing the experiment. This also made it possible to pick up dysfunctional sensors during the experiment. The data synchronization was of decent quality. To start recording, a character was sent through a serial connection to the Arduino. Since two Arduinos had to get a start signal, this had to be done twice. This generated a minor offset of no more than about 0.1 seconds, which is more than enough for this experiment as we are looking at average data from 5 minutes of recordings. However, time sensitive research experiments would benefit from arranging the synchronization with higher accuracy. The sensor platform provided a lot of data, of which most could be considered to be of good quality. The exception from this was the EMG data, which, as mentioned, was sent through an RC circuit from the "slave" to the "master" Arduino. This generated an offset on the EMG value from the original. Though this was tested and confirmed to have a constant offset of about -148 millivolts, the data was still "manipulated". As such, the experiment would have benefited from having a sensor platform that could have sent the original EMG values directly to the master Arduino without manipulation, for example through serial communication. Though this would have generated a lower sample rate (because of more time waiting for data to arrive in the serial), maybe lower than the ECG measurements required, it would have been the real EMG values. Continuing with the EMG data, several factors affect this value when using surface electrodes, as mentioned before; subcutaneous tissue layers, placing of electrodes and more. One thing that should have been more controlled during this experiment was the placing. There were pre-determined areas to place the electrodes, but sometimes right after attaching, an EMG signal was not obtained and electrodes needed to be replaced to receive a signal. This was not reflected upon well enough during the conduction of the experiment, which together with earlier mentioned factors that affect this value resulted in a large variations in EMG values from subject to subject. That said, this experiment was conducted to explore whether there is a variation in muscle tension by increasing or reducing the valence, not quantifying exactly how much this variation would be. This, along with the inconvenience of using needle electrodes justifies this setup.

The user interface software used on the stimuli computer, OpenSesame, was indispensable for this experiment. In the pilot tests, the Affect Grid, the PANAS and the AD-ACL were given to the subjects on paper, making a lot of extra work to transfer this data to the computer. In addition, all the information was given manually by one of the experimenters. With all the information to be given, it was hard to make this process identical every time, as well as it making room for biasing the subjects depending on how the "initial relationship" between the experimenter and the subject was developed. By doing all this directly on the TV screen, the experiment ensured complete neutrality between subjects, as well as making it easier for the subjects to process all the information before moving on to the next step. In addition, this gave all the data gathered from the questionnaires directly in an Excel sheet, ready to be analyzed. Regarding user interaction, some implications when using the PANAS and AD ACL questionnaires were experienced. Some words were difficult to understand and this was solved by allowing subjects to ask the meaning of a word and getting an answer from one of the experimenters. This opened for some interaction that could have been avoided by having explanations on screen.

The stimuli used to provoke different affect levels worked quite well. As seen from the Affect Grid and PANAS measurements, this made a good baseline with the levels of valence necessary to analyze data according to these changes. The tools used as stimuli were relatively simple. Lights and sound made the baseline for this, along with the game itself with varying difficulty and feedback. A thought worth mentioning is that even though this experiment achieved a large spread in valence, there are more to this term than what can be affected externally with such tools used here, and subjectively rating oneself is difficult. I.e many subjects rated themselves at the most negative valence possible in the affect grid in S2. However, the author doubts that this rating would hold if they were to rate after considering the most negative experience they had in their life. It is also challenging to simulate extremely positive or negative feelings in a short experiment without it being somehow artificial. However, it is clear that the valence was changing and that this made it possible to analyze the data accordingly. Considering the simple setup, this emphasizes how one can easily do research on user interactions. Furthermore, one can ask how one can expect physiological reactions to change when sitting completely still and not really doing much. What was intended for this experiment was to explore the small differences in reactions that are not visible to the naked eye. Those may also be the most interesting findings. Not just the things people "do" that they are not consciously aware of, but also the things they "do" that they would not have noticed even if they tried. Furthermore, there was observed a reduction in leaning distance in S2, where the light was quite extreme. A potential weakness in this observation could come from people leaning back in an effort to "get as far away from the annoying lights as possible".

The subjects recruited for this experiment were all between 20 and 30 years of age. It was assumed beforehand that most of the people that were going to participate had knowledge about Tetris and how the game worked, and that many were likely to have played it before. This assumption appeared to be true. In fact, the similarity between subjects in terms of knowledge to Tetris made a fundament for similar and predictable results in terms of affect. However, it should be noted that Tetris may not work the same way for other parts of the population which do not have the same relationship to the game and prerequisites to play. If some know the game and others don't this could be a source for unpredictable affect levels. Therefore, if Tetris is to be used in future experimentation it is recommendable to test how the game in itself apply to people's emotion before using it.

The subjects were all students except for two, and they were all around the same age. As mentioned, the experiment benefits from this in terms of affect level. Although, when conducting an experiment like this to test physiological reactions, a larger spread in the subject population would have made the data more rich and extensive. Many of the students were friends, or acquaintances, of the author and the other student running the experiment. This could have affected and biased the interaction with the subjects, consciously or unconsciously. For more extensive research this is worth considering.

In this experiment, the data was gathered first, and then the statistical methods for analysis were chosen. This lead to some requirements for these methods not being fulfilled, for example, the requirement of normality. For further research conducting similar experiments, one could benefit from assuring during data collection that the data meets the requirements for the targeted statistical tests to be performed with the data afterwards.

External factors like time of day, weather, room temperature etc. were not controlled in this experiment. Some data of this kind was recorded, like how many cups of coffee the subjects had been drinking that day. Though this was not tested, it is suspected that some variables, like coffee, could have had some effect on for example arousal level. The validity of the results could have been extended by having more control of these external variables.

5.1 Implications and Future Work

As the experiment presented, as well as the results, had its limitations, this section briefly discusses and describes what would have been done if the limitations were accounted for, if the physiological data were to show significant change between the scenarios, what this would mean for the research and how this could be further studied.

As mentioned, The one-way repeadet measures ANOVA determines significant change on a dependent variable on a within-subjects factor, in this case valence. The analyses never made it to the point where the EMG and leaning data could actually be compared to the valence data from the Affect Grid and PANAS. To find the direction of the change in physiological data, if there had been a difference between scenarios, a post hoc test would have been conducted. This is basically a multiple paired-samples T-test, with correction to adjust the confidence intervals and p-values accounting for making multiple comparisons. This would have given an idea of the direction and magnitude of the change in EMG and leaning distance.

Furthermore, if a change in EMG or leaning distance with valence had been discovered, this would have to be analyzed also according to the arousal dimension. This experiment aimed consciously to achieve a reflected 'L' shape in the Affect Grid with the scenarios, i.e from top left to top right and then down right. This was for both the students conducting the experiment to achieve necessary variation in the valence and arousal dimension, but also to use the "third point" to potentially exclude effects that had to do with the dimension not being analyzed. In this case, S2 (low valence-high arousal) and S3 (high valence-high arousal) would have been the main focus, and S1 (high valence-low arousal) would have been the 'control measurement'. A change in EMG or leaning distance from S2 to S3 could then have been tested with a change from S1 to S2, to see if this change was independent of arousal or if it was simply a change between positive and negative excitement.

Being able to physically observe valence in the data gathered would certainly be of high interest to many. As mentioned in the theoretical background, the research community has been exploring emotional differentiation in physiological reactions and some interesting findings have been made, although this research have been focusing on the facial region. Findings on the rest of the body would establish also this as a channel for emotional communication, where the possibilities for future research are endless. Furthermore, this would have various applications within Affective Engineering, where developers are constantly striving to understand and facilitate for their users. Furthermore, as the equipment for measuring physiological reactions now is becoming more and more available and low-cost, an objective measurement tool for valence would open many doors to study also other reactions quite easily.

Chapter 6

Conclusion

The work of this thesis was done with the goal of exploring and testing how valence could be measured through physiological reactions, and how one can prototype an experimental setup for this purpose. Firstly, background theory within Affective Engineering, emotions, physiological reactions and measurement tools of valence was presented. Then followed a complete overview of the exploration and setup for the experiment conducted. A lot of effort was put into this part of the work, and prototyping and piloting were done in several iterations to ensure a robust setup and that the experiment affected people in the desired way. Although the recorded data revealed that there was no significant difference between the scenarios (except for leaning distance, but the effect was evaluated as a too small) the thesis shows that this kind of setup could, with some modifications, be applied to similar kind of research within this area. Furthermore, it demonstrates that it is possible, with simple materials and limited resources, to do this kind of research. As such, this study functions as an invitation and encouragement to others that would be interested in studying the connection between emotions and physiology. With today's accessibility of sensory equipment, the road is not that long.

This page is intentionally left blank.

Bibliography

Ax, A. F., 1953. The physiological differentiation between fear and anger in humans. Psychosomatic medicine 15 (5), 433–442.

Balters, S., Steinert, M., 2015. Capturing emotion reactivity through physiology measurement as a foundation for affective engineering in engineering design science and engineering practices. Journal of Intelligent Manufacturing, 1–23.

Brown, T., 2008. Design thinking. Harvard Business Review.

Cacioppo, J. T., Petty, R. E., Losch, M. E., Kim, H. S., 1986. Electromyographic activity over facial muscle regions can differentiate the valence and intensity of affective reactions. Journal of personality and social psychology 50 (2), 260.

Christie, I. C., Friedman, B. H., 2004. Autonomic specificity of discrete emotion and dimensions of affective space: A multivariate approach. International journal of psychophysiology 51 (2), 143–153.

Cohen, J., 1988. Statistical power analysis for the behavioral sciences lawrence earlbaum associates. Hillsdale, NJ, 20–26.

Cohn, M. A., Fredrickson, B. L., Brown, S. L., Mikels, J. A., Conway, A. M., 2009. Happiness unpacked: positive emotions increase life satisfaction by building resilience. Emotion 9 (3), 361.

Coulson, M., 2004. Attributing emotion to static body postures: Recognition accuracy, confusions, and viewpoint dependence. Journal of nonverbal behavior 28 (2), 117–139.

Crawford, J. R., Henry, J. D., 2004. The positive and negative affect schedule (panas): Construct validity, measurement properties and normative data in a large non-clinical sample. British Journal of Clinical Psychology 43 (3), 245–265.

Dael, N., Mortillaro, M., Scherer, K. R., 2012. Emotion expression in body action and posture. Emotion 12 (5), 1085.

Dahlgaard, J. J., Dahlgaard, J. J., Schütte, S., Ayas, E., Mi Dahlgaard-Park, S., 2008. Kansei/affective engineering design: A methodology for profound affection and attractive quality creation. The TQM Journal 20 (4), 299–311.

Desmet, P., 2005. Measuring emotion: Development and application of an instrument to measure emotional responses to products. Funology, 111–123.

Ekman, P., 1994. Strong evidence for universals in facial expressions: a reply to russell's mistaken critique.

Farina, D., Merletti, R., Enoka, R. M., 2004. The extraction of neural strategies from the surface emg. Journal of Applied Physiology 96 (4), 1486–1495.

Greenhouse, S. W., Geisser, S., 1959. On methods in the analysis of profile data. Psychometrika 24 (2), 95–112.

Hacks, C., 2017. e-health documentation.
URL `https://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical#step4_2`

Hazlett, R. L., 2006. Measuring emotional valence during interactive experiences: boys at video game play. In: Proceedings of the SIGCHI conference on Human Factors in computing systems. ACM, pp. 1023–1026.

HowToMedia, I., n.d. Muscular system. `http://www.innerbody.com/image/musfov.html`.

Levenson, R. W., 2003. Autonomic specificity and emotion. Handbook of affective sciences 2, 212–224.

Mathôt, S., Schreij, D., Theeuwes, J., 2012. Opensesame: An open-source, graphical experiment builder for the social sciences. Behavior research methods 44 (2), 314–324.

Merletti, R., Parker, P. A., 2004. Electromyography: physiology, engineering, and non-invasive applications. Vol. 11. John Wiley & Sons.

Montepare, J., Koff, E., Zaitchik, D., Albert, M., 1999. The use of body movements and gestures as cues to emotions in younger and older adults. Journal of Nonverbal Behavior 23 (2), 133–152.

Nagamachi, M., 1995. Kansei engineering: a new ergonomic consumer-oriented technology for product development. International Journal of industrial ergonomics 15 (1), 3–11.

Russel, J. A., Weiss, A., Mendelsohn, G. A., 1989. Affect grid: A single-item scale of pleasure and arousal. Journal of Personality and Social Psychology 57 (3), 493–502.

Russell, J., 1980. A circumplex of affect. Journal of Personality and Social Psychology 36, 1152–1168.

Scherer, K. R., 1987. Toward a dynamic theory of emotion. Geneva studies in Emotion 1, 1–96.

Sobek, D. K., Ward, A. C., Liker, J. K., 1999. Toyota's principles of set-based concurrent engineering. Sloan management review 40 (2), 67.

Steinert, M., Leifer, L. J., 2012. 'finding one's way': Re-discovering a hunter-gatherer model based on wayfaring. International Journal of Engineering Education 28 (2), 251.

Stemmler, G., Heldmann, M., Pauls, C. A., Scherer, T., 2001. Constraints for emotion specificity in fear and anger: The context counts. Psychophysiology 38 (02), 275–291.

Takeuchi, H., Nonaka, I., 1998. The new new product development game. Japanese Business: Part 1, Classics Part 2, Japanese management Vol. 2: Part 1, Manufacturing and production Part 2, Automotive industry Vol. 3: Part 1, Banking and finance Part 2, Corporate strategy and inter-organizational relationships Vol. 4: Part 1, Japanese management overseas Part 2, Innovation and learning 64 (1), 321.

Thayer, R. E., 1989. The biopsychology of mood and arousal: Oxford university press. New York.

Watson, D., Clark, L. A., Tellegen, A., 1988. Development and validation of brief measures of positive and negative affect: the panas scales. Journal of personality and social psychology 54 (6), 1063.

Yakut, O., Solak, S., Bolat, E. D., 2014. Measuring ecg signal using e-health sensor platform. In: International Conference on Chemistry, Biomedical and Environment Engineering, Antalya. pp. 71–75.

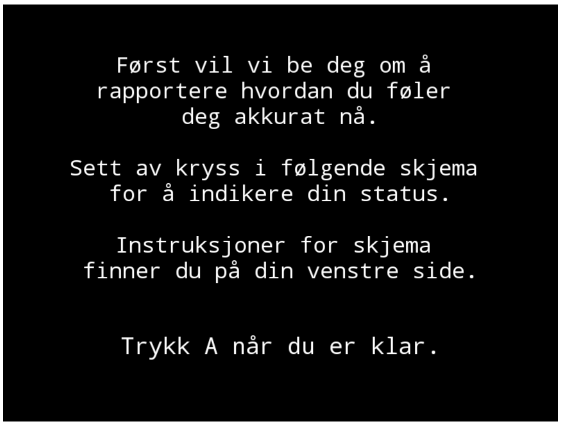
This page is intentionally left blank.

Appendix

This page is intentionally left blank.

Appendix A - Graphical User Interface (GUI)

Velkommen

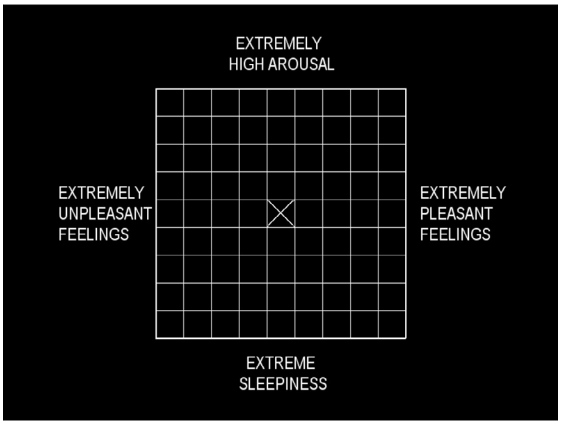


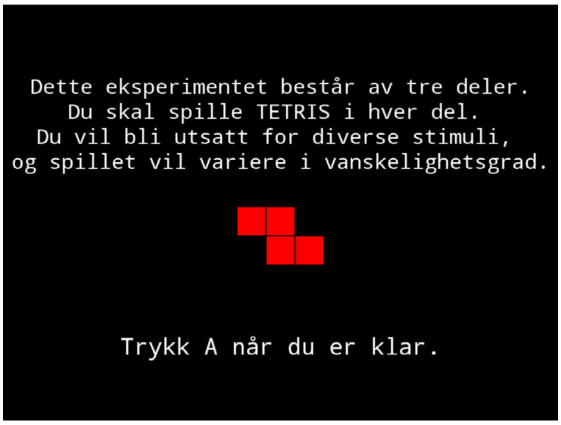
Først vil vi be deg om å rapportere hvordan du føler deg akkurat nå.

Sett av kryss i følgende skjema for å indikere din status.

Instruksjoner for skjema finner du på din venstre side.

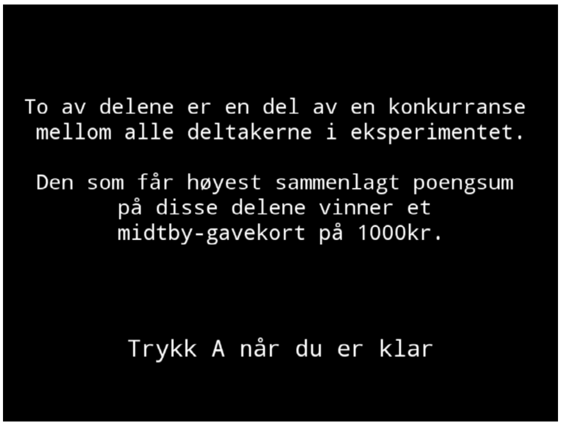
Trykk A når du er klar.





Dette eksperimentet består av tre deler.
Du skal spille TETRIS i hver del.
Du vil bli utsatt for diverse stimuli,
og spillet vil variere i vanskelighetsgrad.

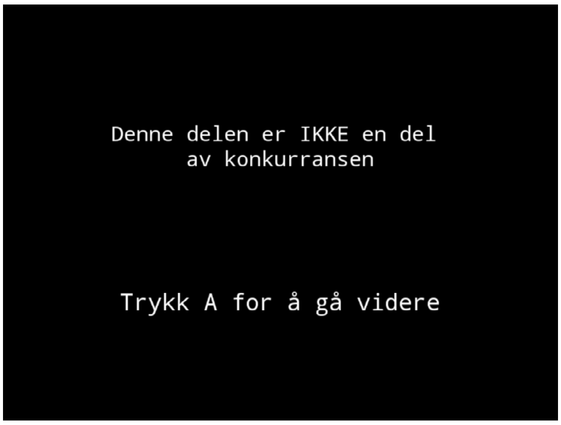
Trykk A når du er klar.



To av delene er en del av en konkurranse mellom alle deltakerne i eksperimentet.

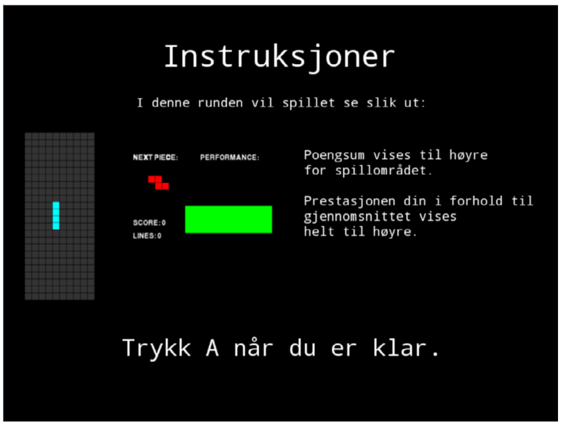
Den som får høyest sammenlagt poengsum på disse delene vinner et midtby-gavekort på 1000kr.

Trykk A når du er klar



Denne delen er IKKE en del av konkurransen

Trykk A for å gå videre



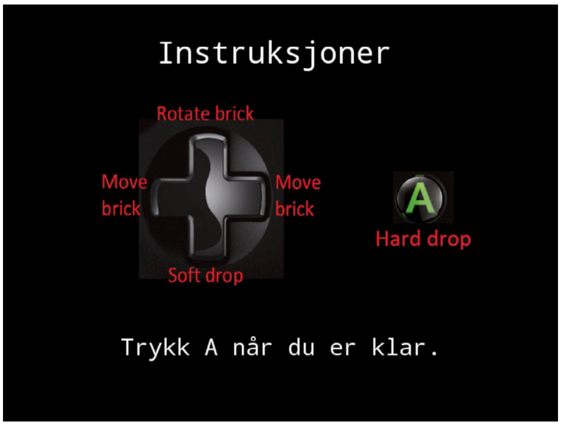
Instruksjoner

I denne runden vil spillet se slik ut:

Poengsum vises til høyre for spillområdet.

Prestasjonen din i forhold til gjennomsnittet vises helt til høyre.

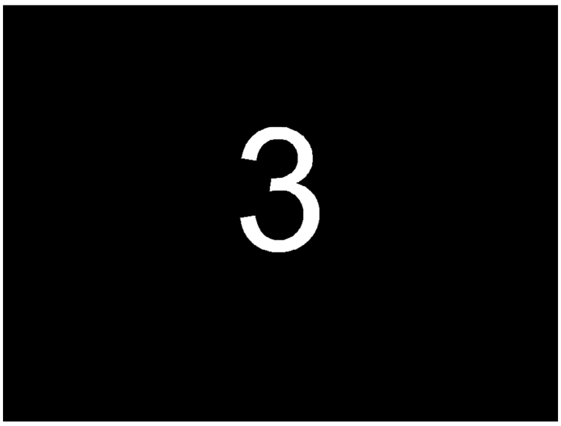
Trykk A når du er klar.



Instruksjoner

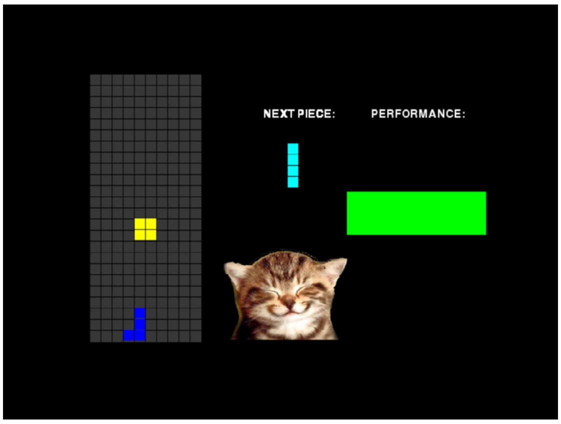
Trykk A når du er klar.

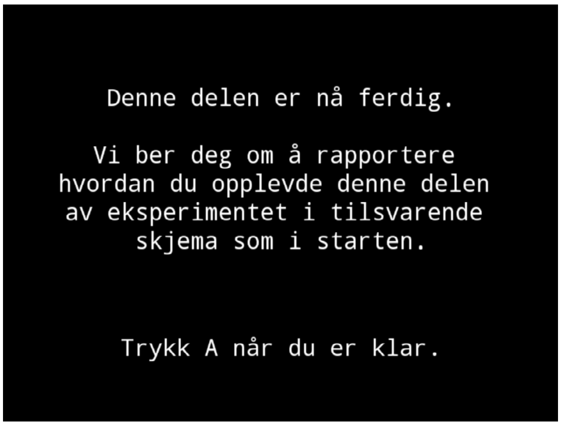
Vennligst vent...


3

2

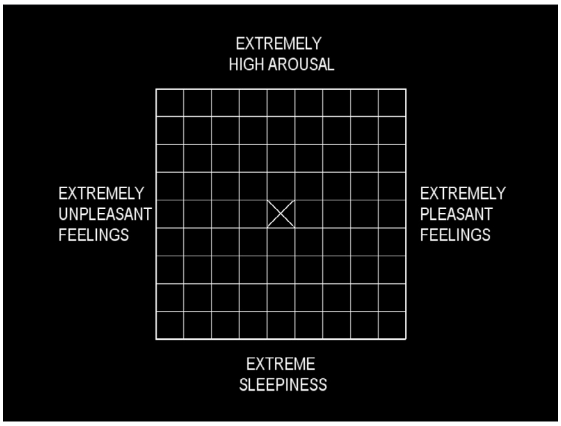
1

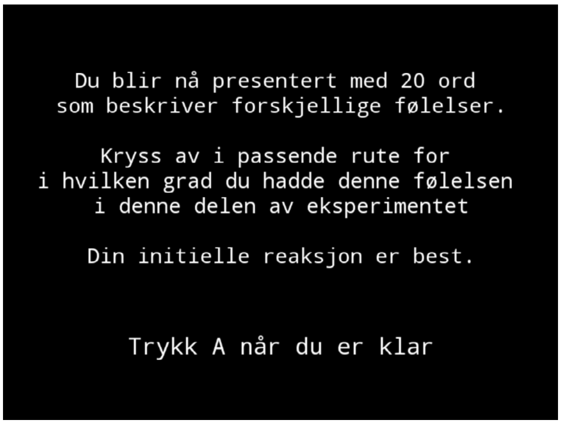



Denne delen er nå ferdig.

Vi ber deg om å rapportere hvordan du opplevde denne delen av eksperimentet i tilsvarende skjema som i starten.

Trykk A når du er klar.

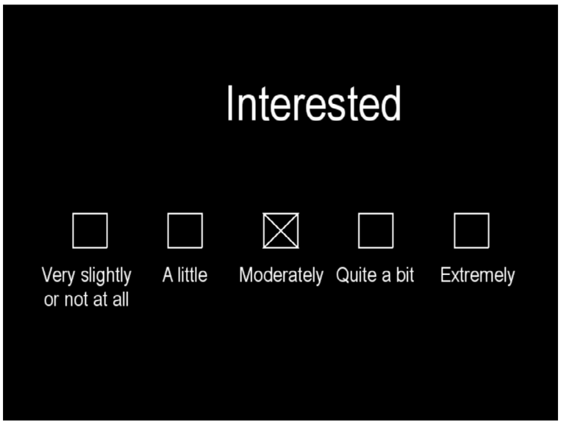



Du blir nå presentert med 20 ord som beskriver forskjellige følelser.

Kryss av i passende rute for i hvilken grad du hadde denne følelsen i denne delen av eksperimentet

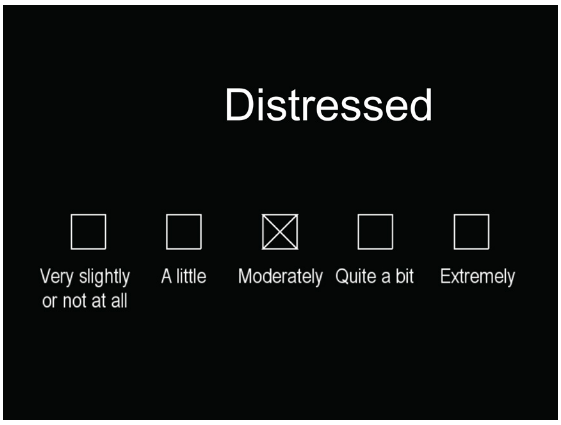
Din initielle reaksjon er best.

Trykk A når du er klar



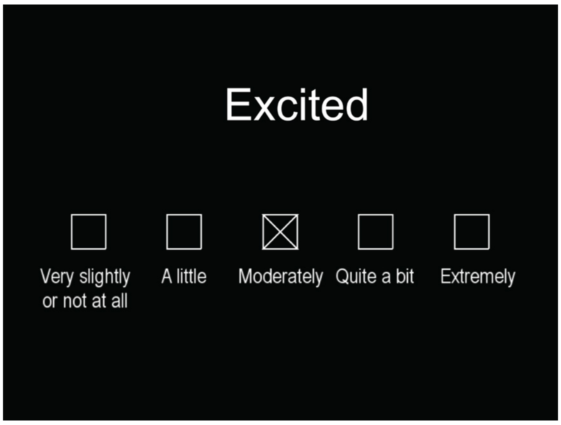
Interested

☐	☐	☒	☐	☐
Very slightly or not at all	A little	Moderately	Quite a bit	Extremely



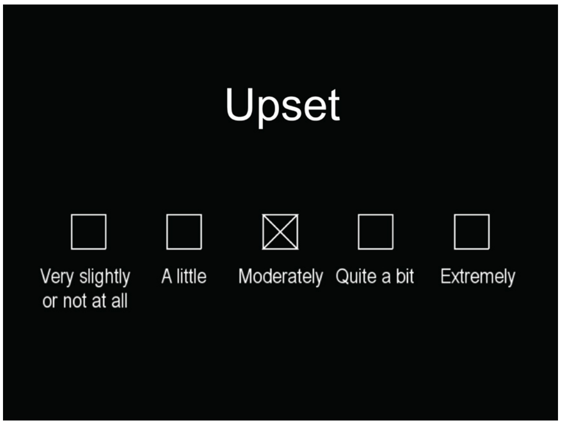
Distressed

☐	☐	☒	☐	☐
Very slightly or not at all	A little	Moderately	Quite a bit	Extremely



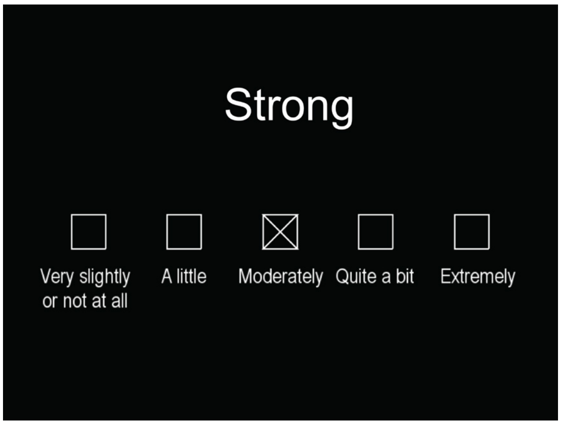
Excited

☐	☐	☒	☐	☐
Very slightly or not at all	A little	Moderately	Quite a bit	Extremely



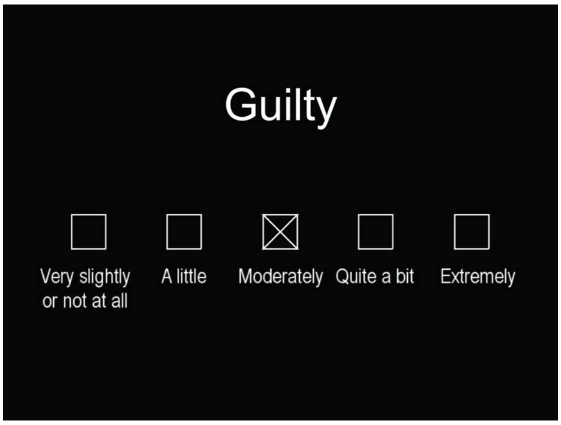
Upset

☐	☐	☒	☐	☐
Very slightly or not at all	A little	Moderately	Quite a bit	Extremely



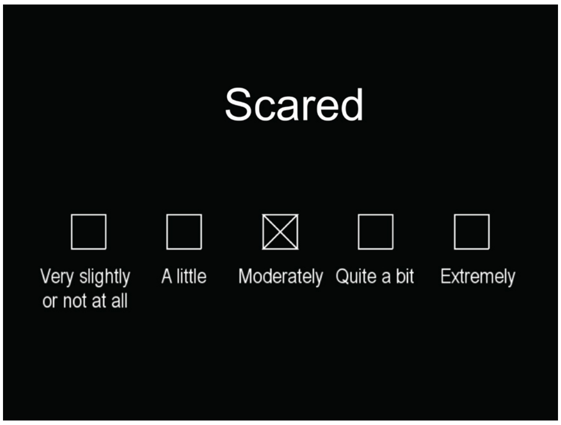
Strong

☐	☐	☒	☐	☐
Very slightly or not at all	A little	Moderately	Quite a bit	Extremely



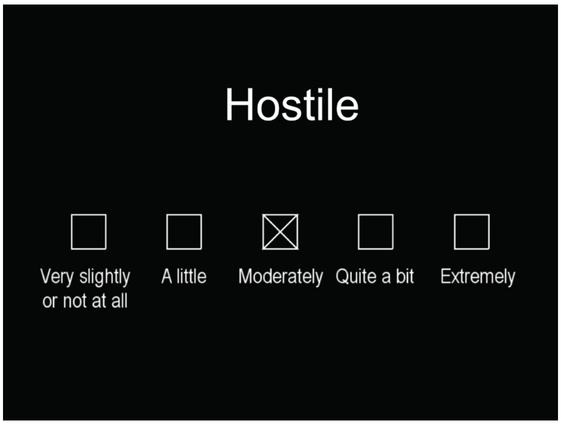
Guilty

☐	☐	☒	☐	☐
Very slightly or not at all	A little	Moderately	Quite a bit	Extremely



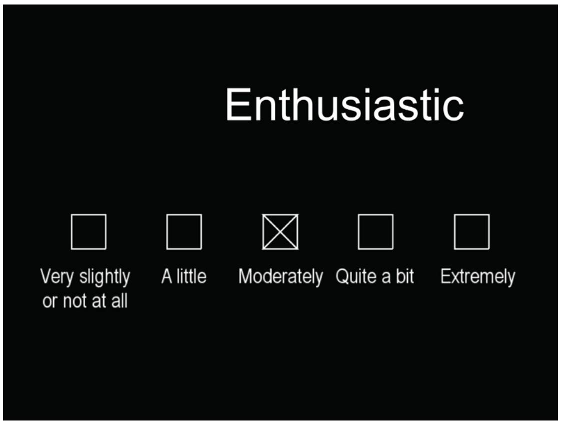
Scared

☐	☐	☒	☐	☐
Very slightly or not at all	A little	Moderately	Quite a bit	Extremely



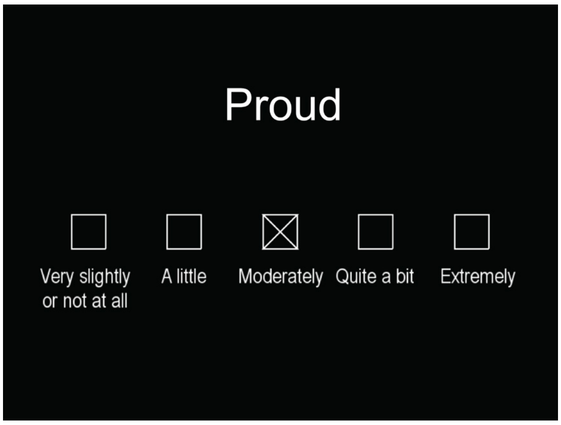
Hostile

☐	☐	☒	☐	☐
Very slightly or not at all	A little	Moderately	Quite a bit	Extremely



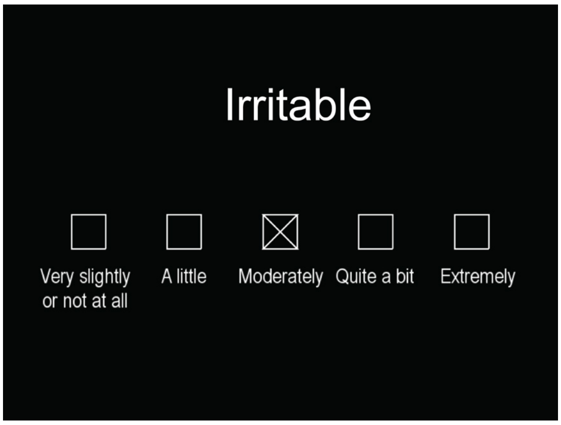
Enthusiastic

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely



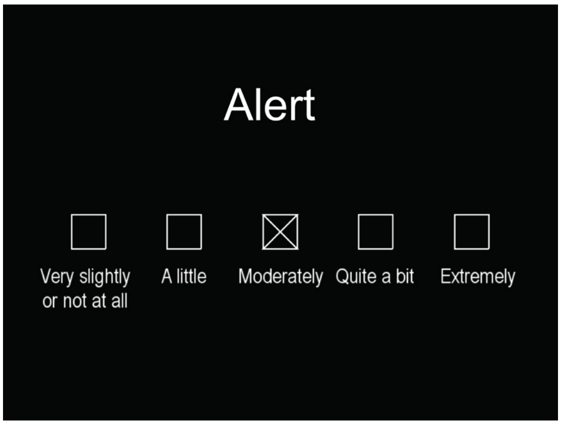
Proud

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely



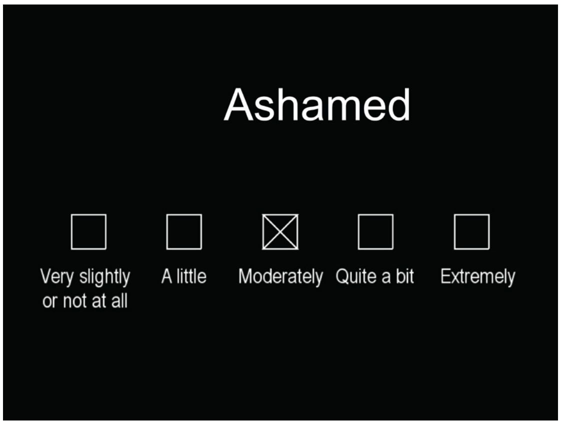
Irritable

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely



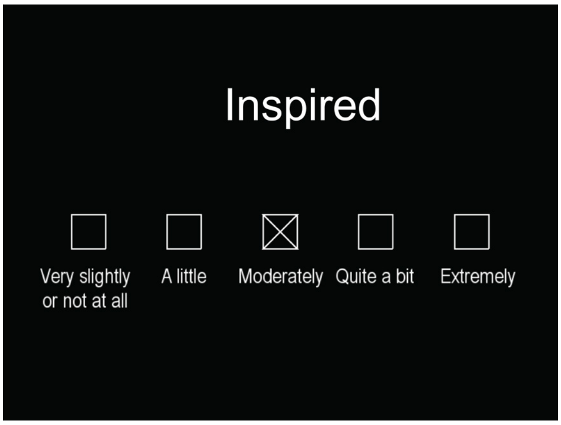
Alert

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely



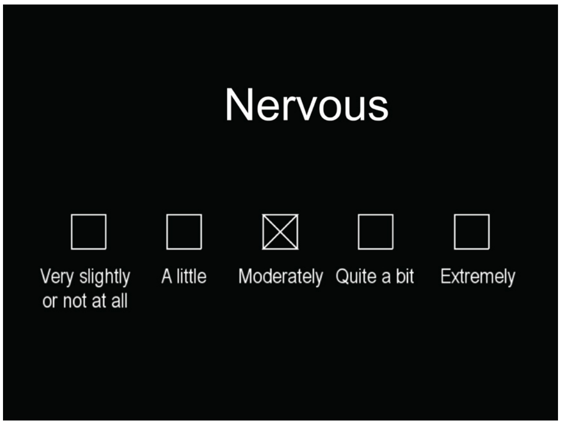
Ashamed

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely



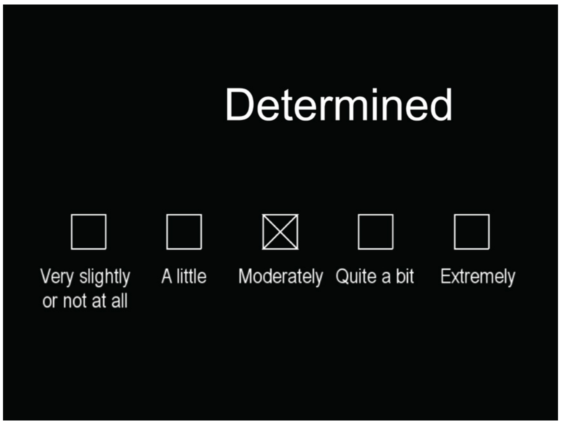
Inspired

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely



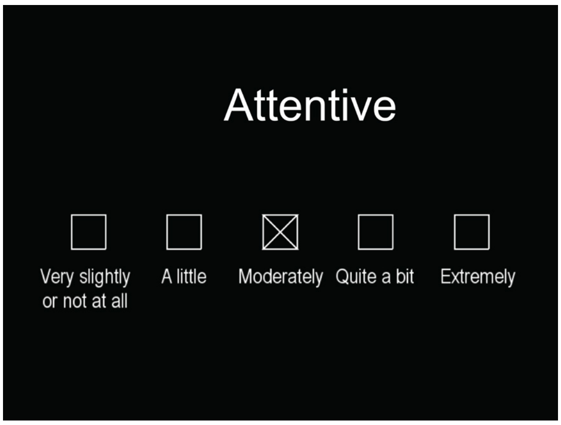
Nervous

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely



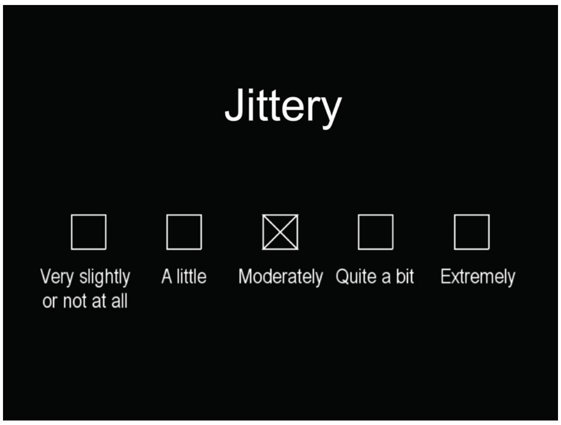
Determined

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely



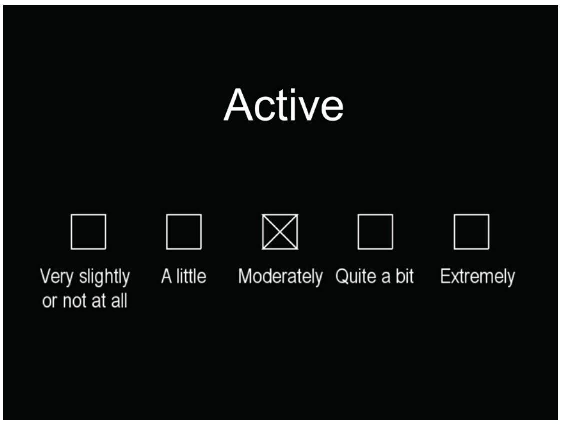
Attentive

Very slightly or not at all | A little | [✓ Moderately] | Quite a bit | Extremely



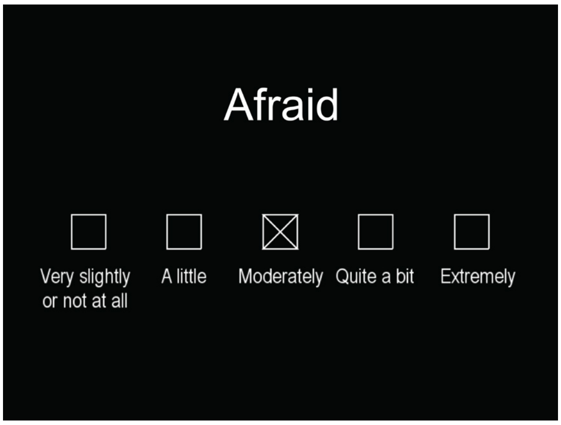
Jittery

Very slightly or not at all | A little | [✓ Moderately] | Quite a bit | Extremely



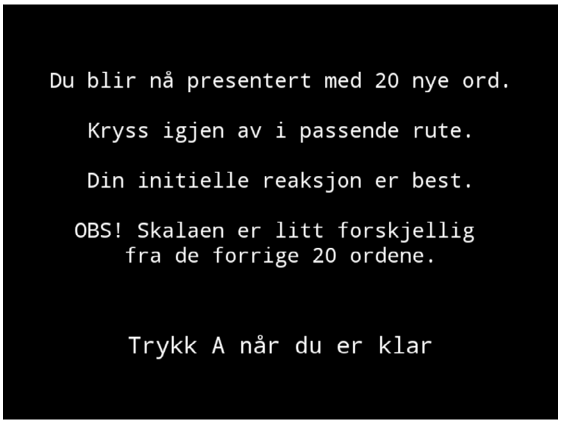
Active

Very slightly or not at all | A little | [✓ Moderately] | Quite a bit | Extremely



Afraid

Very slightly or not at all | A little | [✓ Moderately] | Quite a bit | Extremely



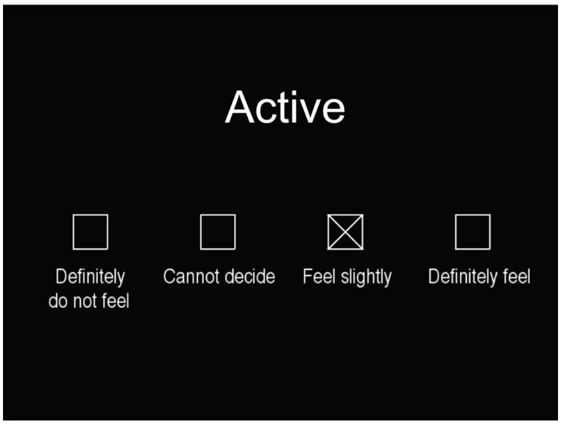
Du blir nå presentert med 20 nye ord.

Kryss igjen av i passende rute.

Din initielle reaksjon er best.

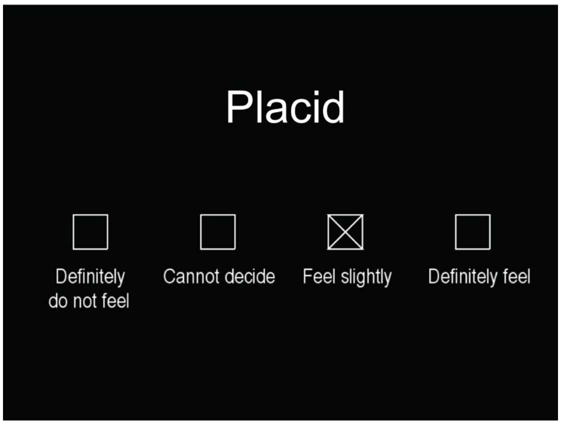
OBS! Skalaen er litt forskjellig
fra de forrige 20 ordene.

Trykk A når du er klar



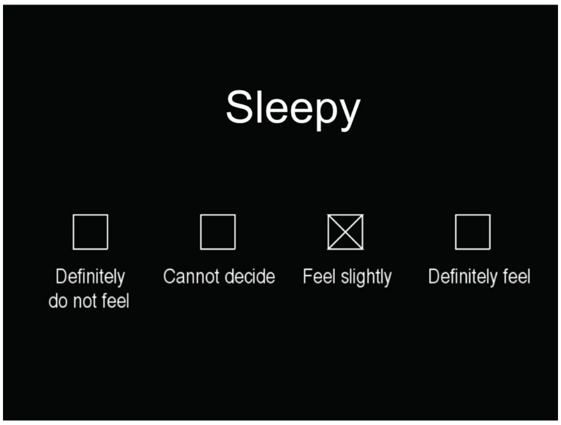
Active

Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel



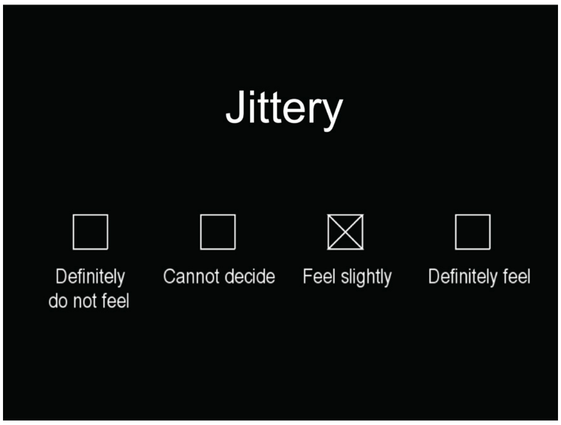
Placid

Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel



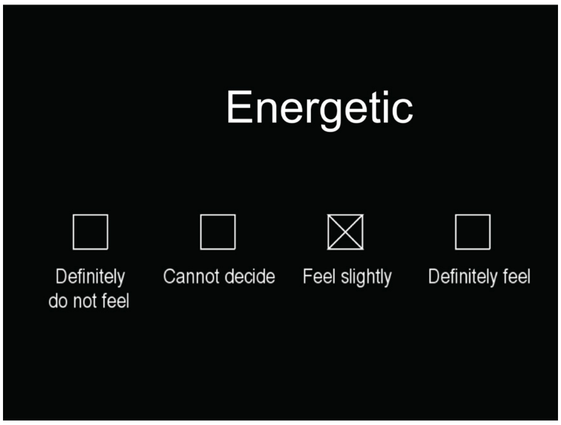
Sleepy

Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel



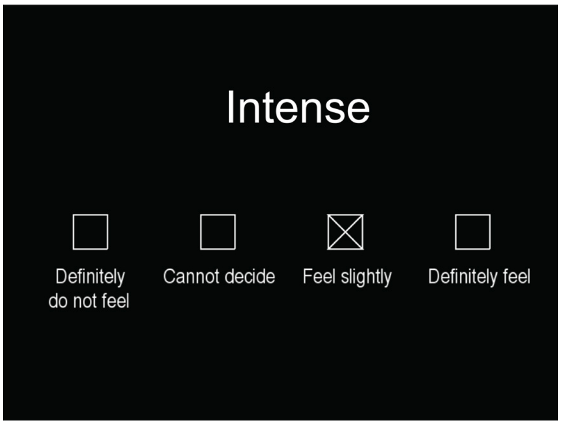
Jittery

Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel



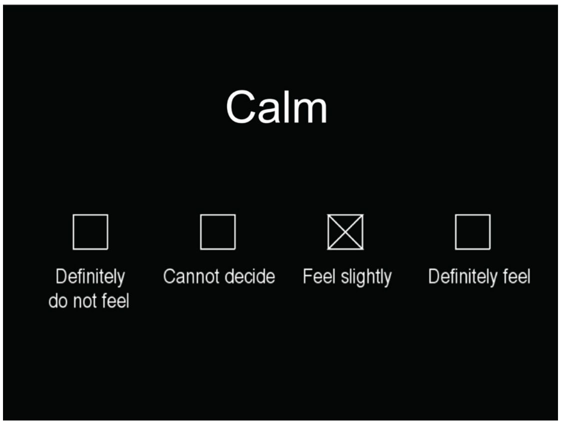
Energetic

Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel



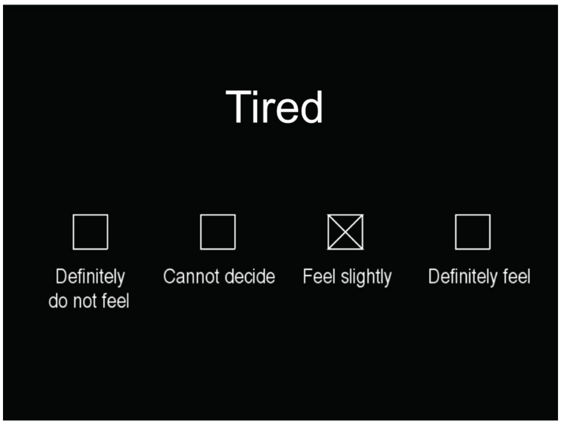
Intense

Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel



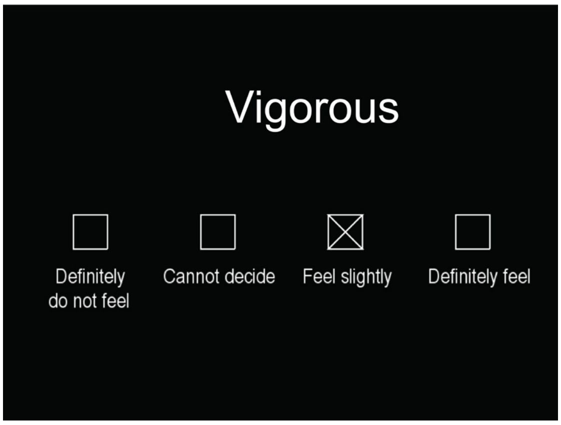
Calm

Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel



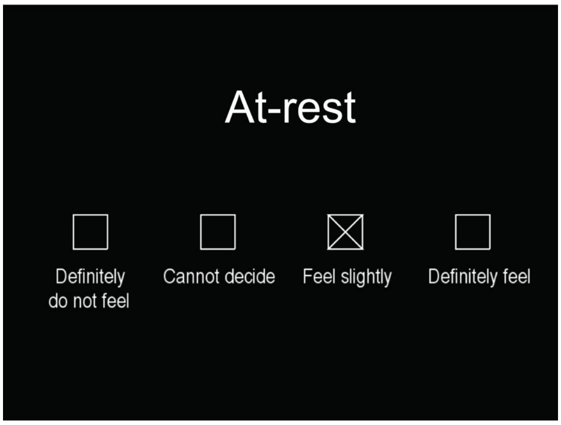
Tired

Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel



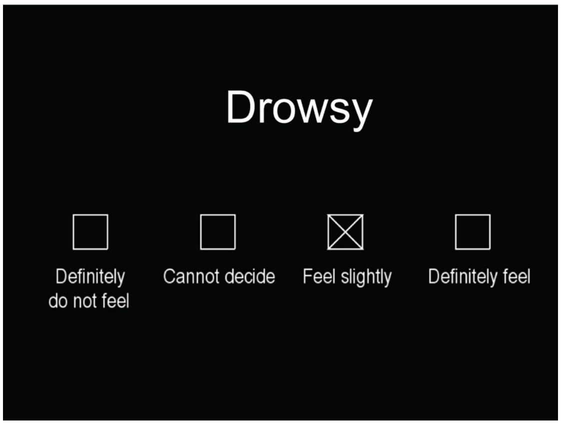
Vigorous

Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel



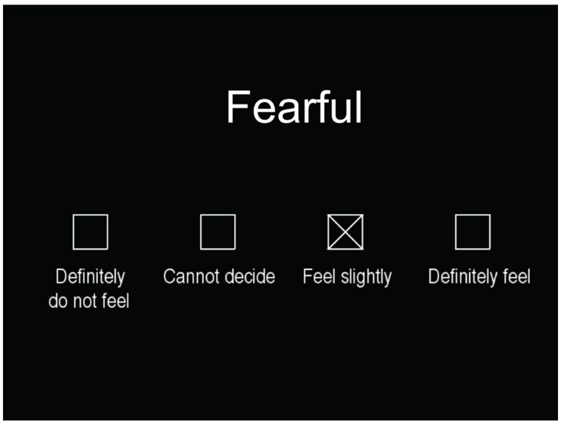
At-rest

Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel

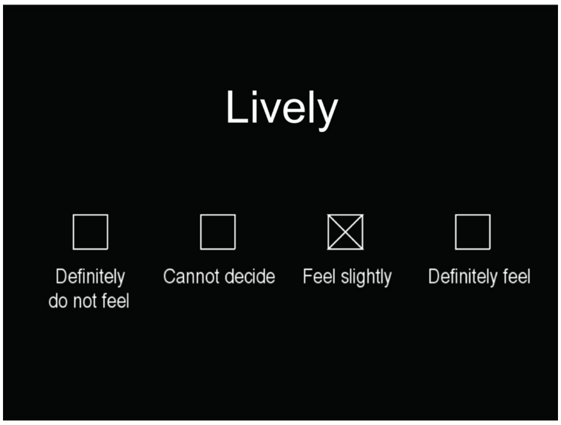


Drowsy

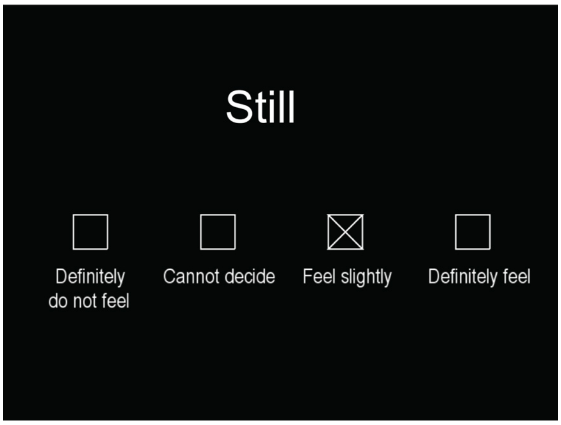
Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel


Fearful

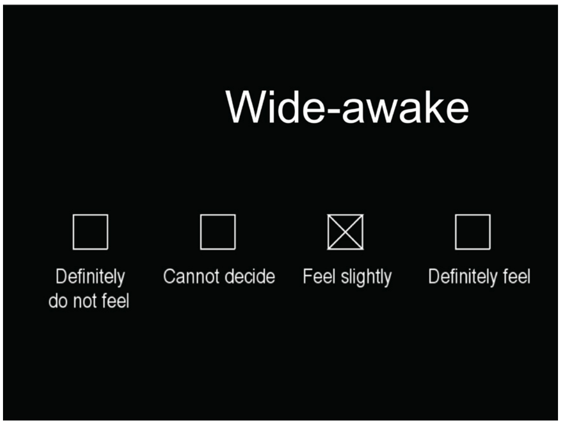
Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel


Lively

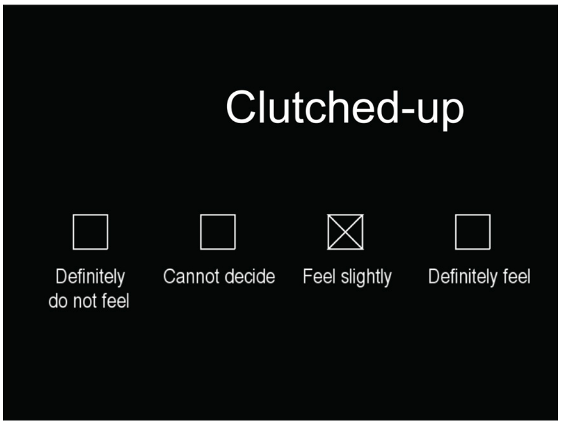
Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel


Still

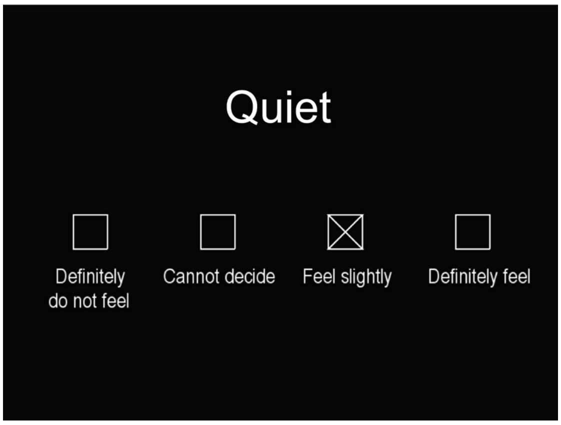
Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel


Wide-awake

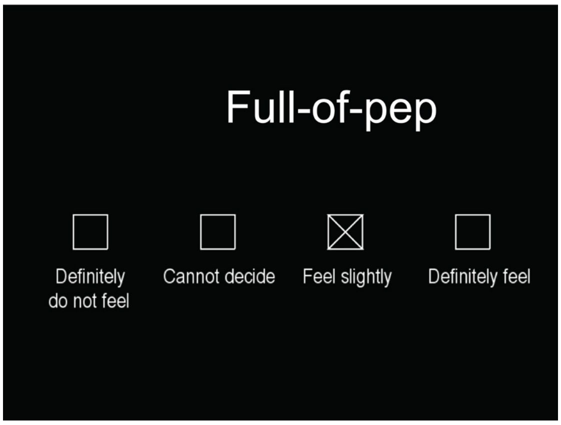
Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel


Clutched-up

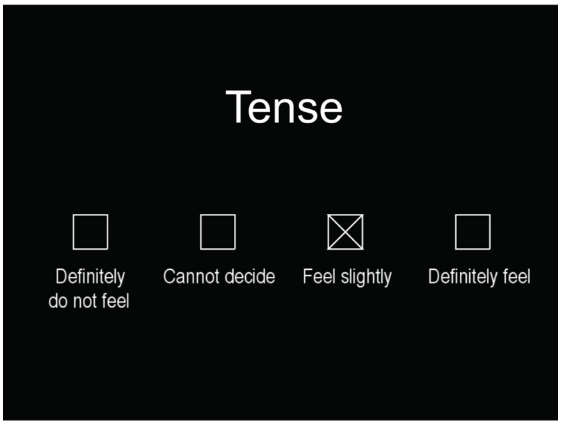
Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel


Quiet

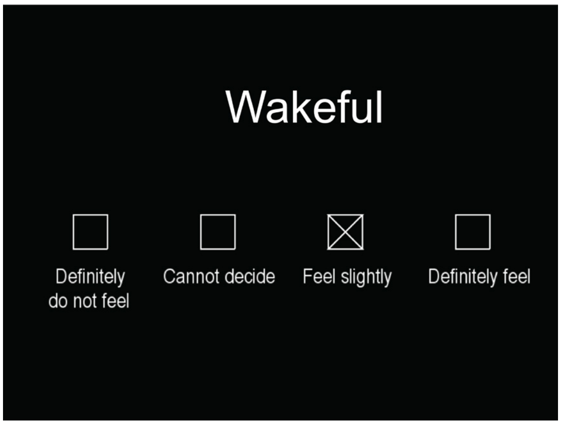
Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel


Full-of-pep

Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel

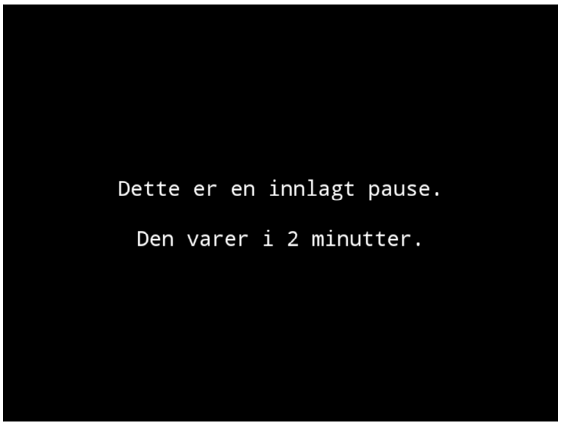

Tense

Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel



Wakeful

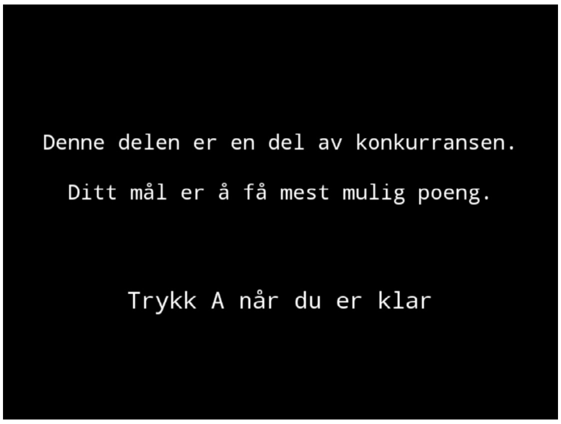
Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel



Dette er en innlagt pause.

Den varer i 2 minutter.

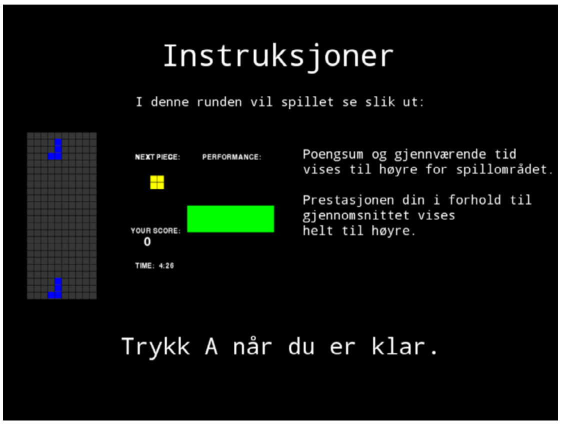
Gjør deg klar



Denne delen er en del av konkurransen.

Ditt mål er å få mest mulig poeng.

Trykk A når du er klar



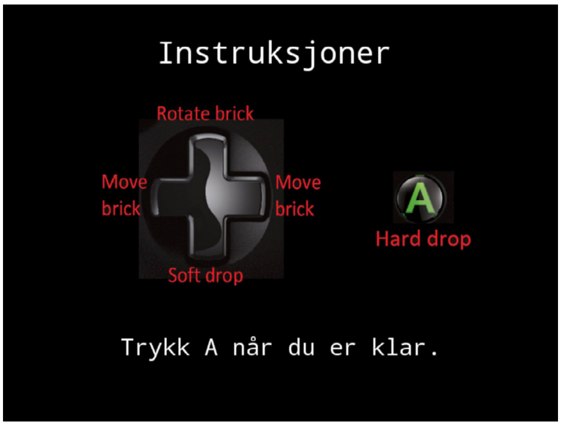
Instruksjoner

I denne runden vil spillet se slik ut:

Poengsum og gjenværende tid vises til høyre for spillområdet.

Prestasjonen din i forhold til gjennomsnittet vises helt til høyre.

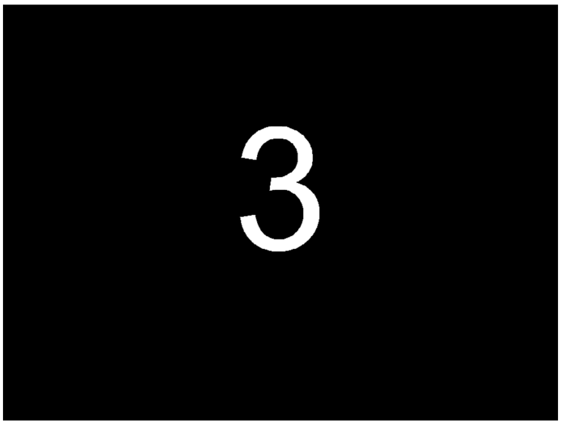
Trykk A når du er klar.



Instruksjoner

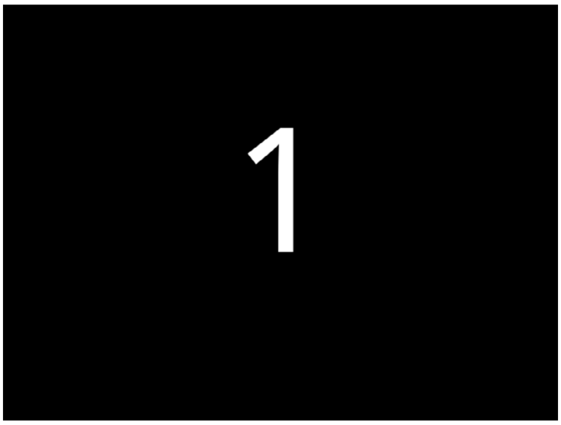
Trykk A når du er klar.

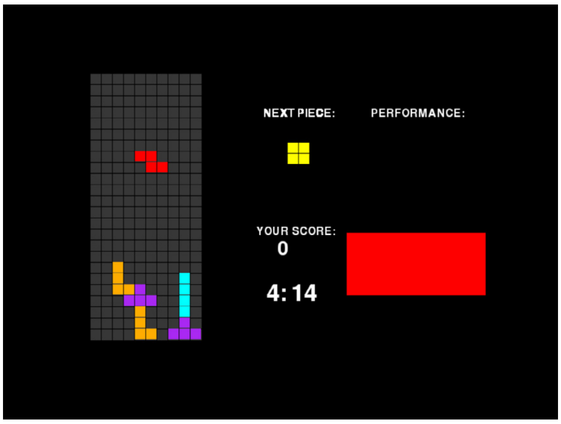
Vennligst vent...

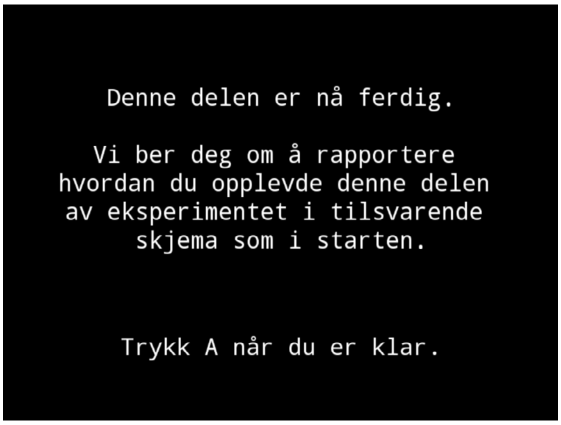


3

2

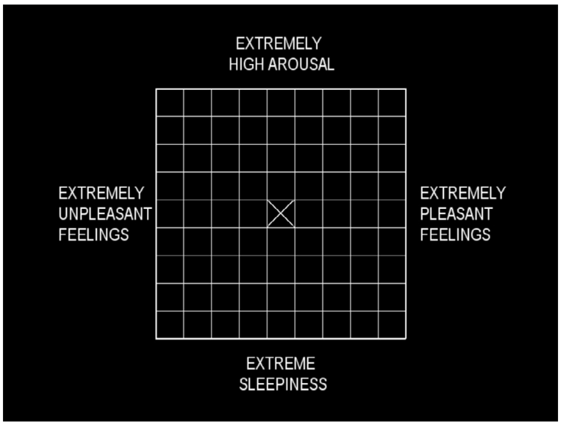

1

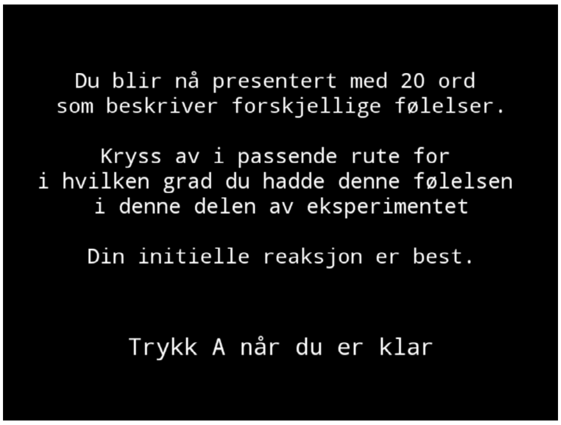



Denne delen er nå ferdig.

Vi ber deg om å rapportere hvordan du opplevde denne delen av eksperimentet i tilsvarende skjema som i starten.

Trykk A når du er klar.

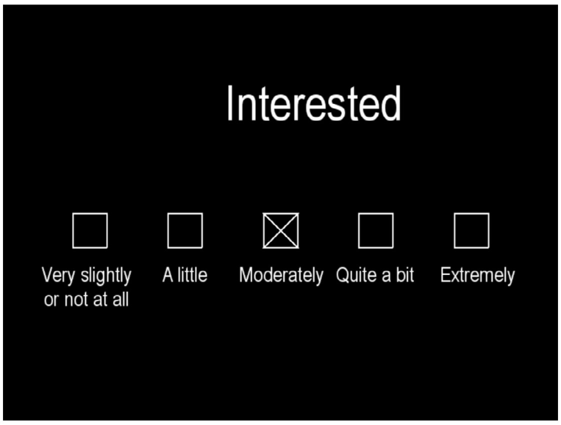



Du blir nå presentert med 20 ord som beskriver forskjellige følelser.

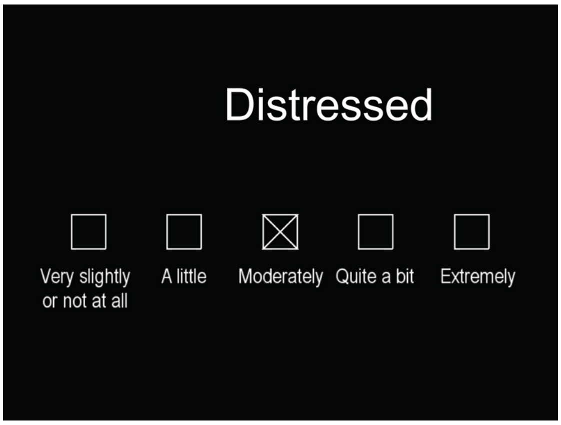
Kryss av i passende rute for i hvilken grad du hadde denne følelsen i denne delen av eksperimentet

Din initielle reaksjon er best.

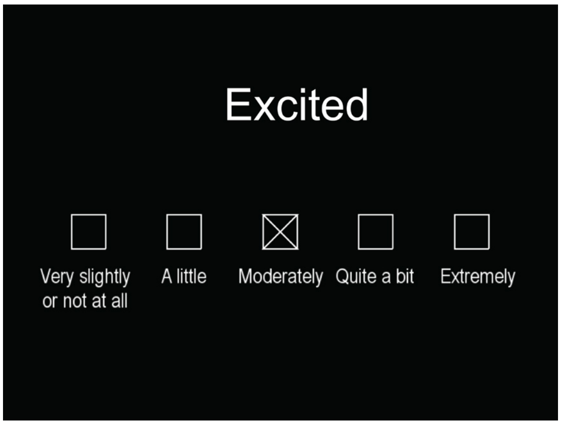
Trykk A når du er klar


Interested

Very slightly or not at all · A little · [✓ Moderately] · Quite a bit · Extremely

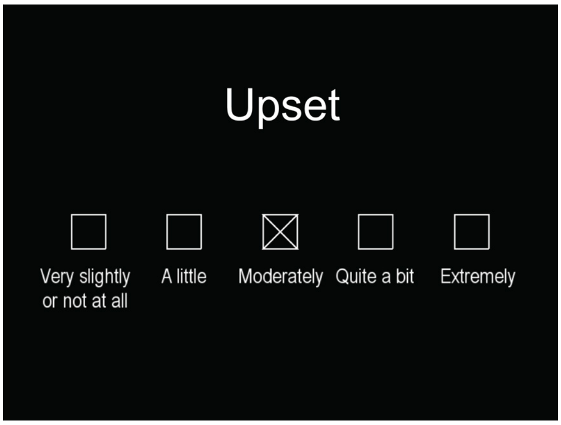

Distressed

Very slightly or not at all · A little · [✓ Moderately] · Quite a bit · Extremely



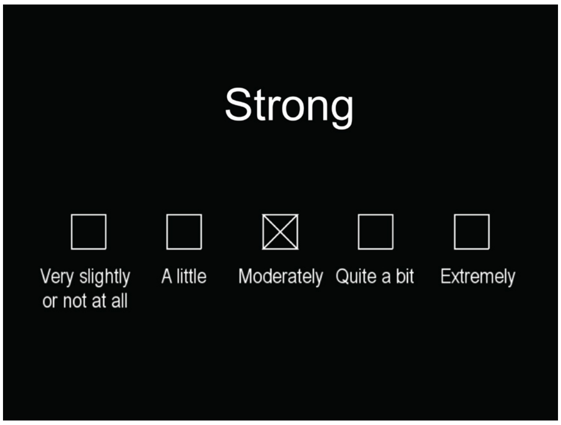
Excited

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely



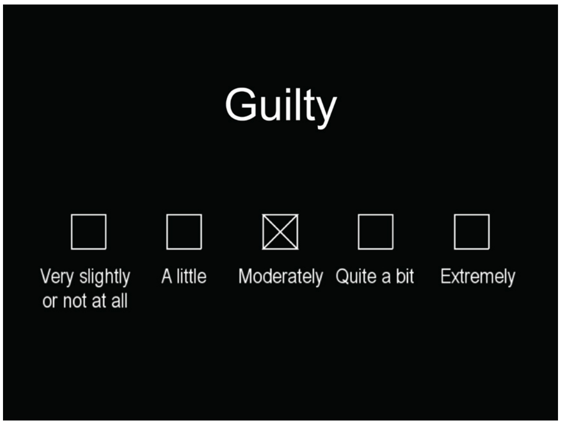
Upset

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely



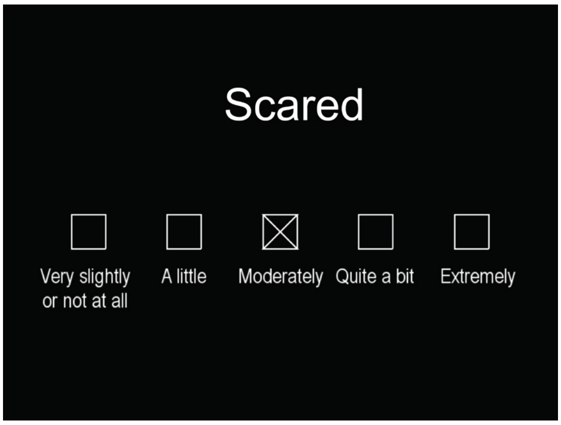
Strong

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely



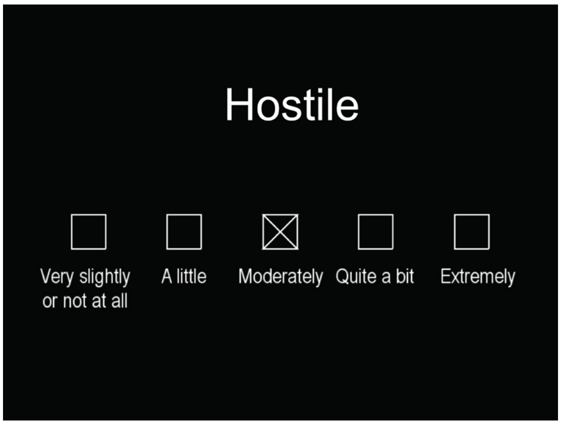
Guilty

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely



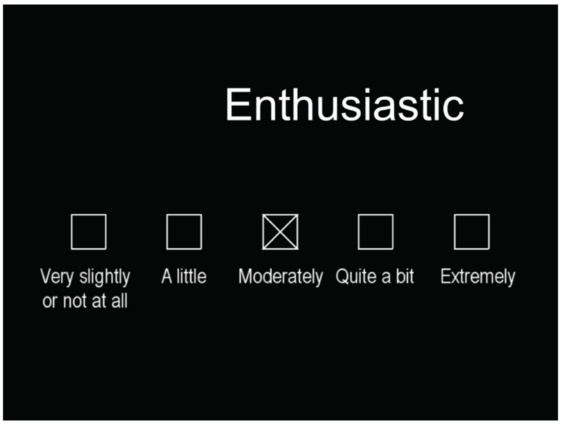
Scared

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely



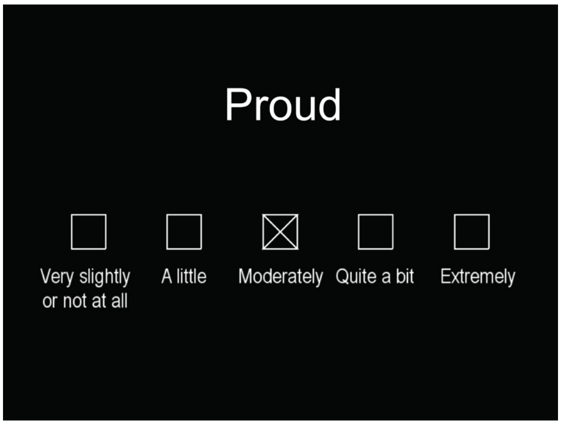
Hostile

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely



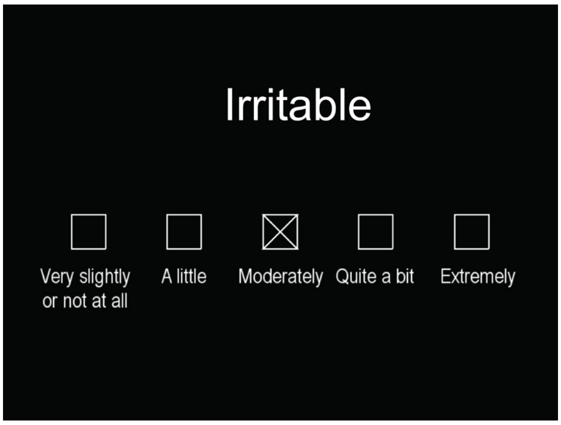
Enthusiastic

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely



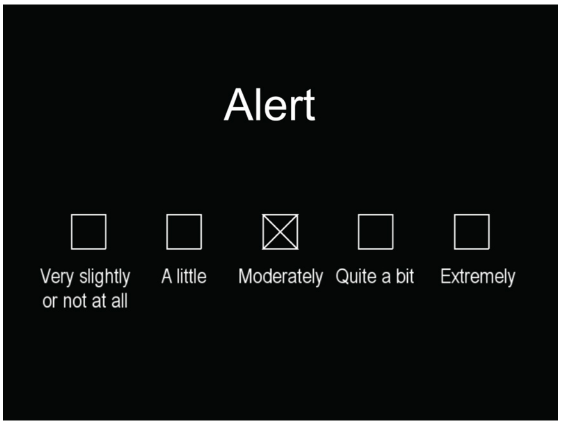
Proud

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely



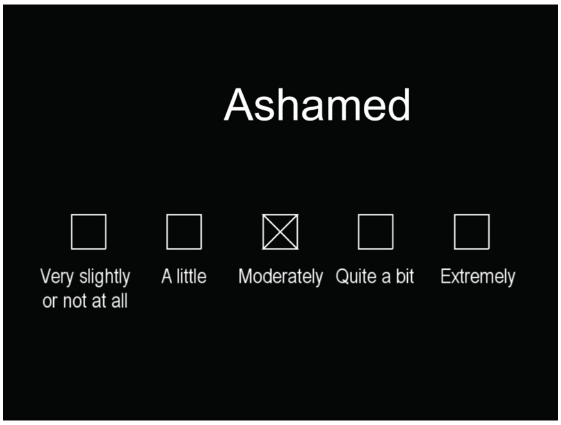
Irritable

Very slightly or not at all · A little · [✓ Moderately] · Quite a bit · Extremely



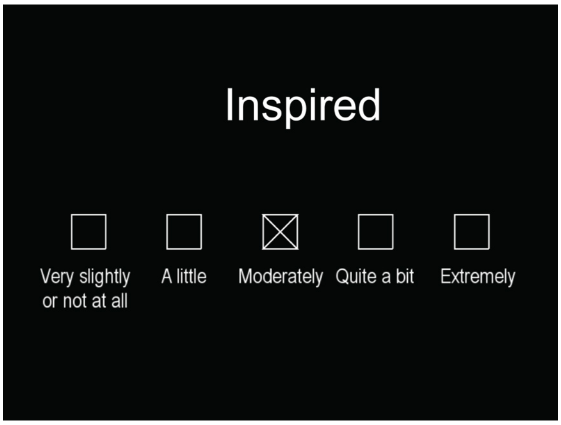
Alert

Very slightly or not at all · A little · [✓ Moderately] · Quite a bit · Extremely



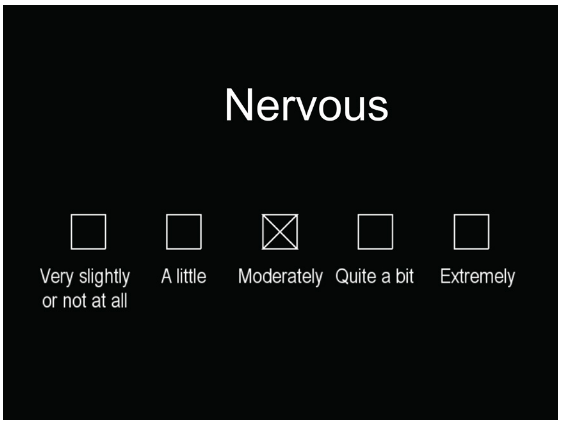
Ashamed

Very slightly or not at all · A little · [✓ Moderately] · Quite a bit · Extremely



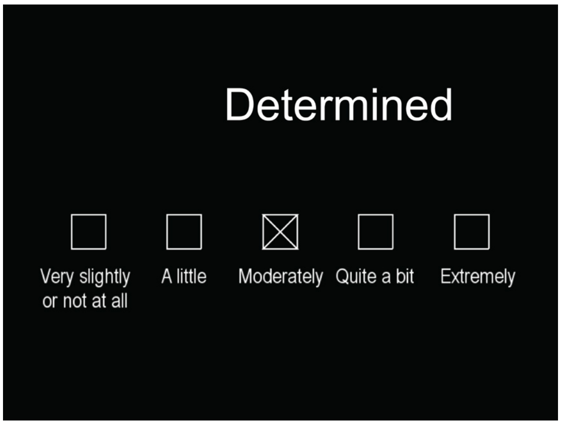
Inspired

Very slightly or not at all · A little · [✓ Moderately] · Quite a bit · Extremely



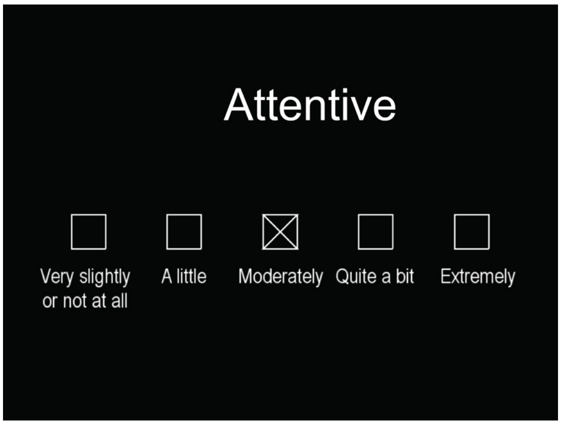
Nervous

Very slightly or not at all · A little · [✓ Moderately] · Quite a bit · Extremely



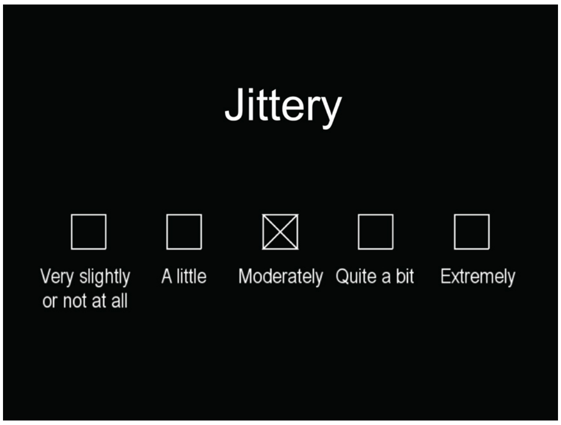
Determined

Very slightly or not at all · A little · [✓ Moderately] · Quite a bit · Extremely



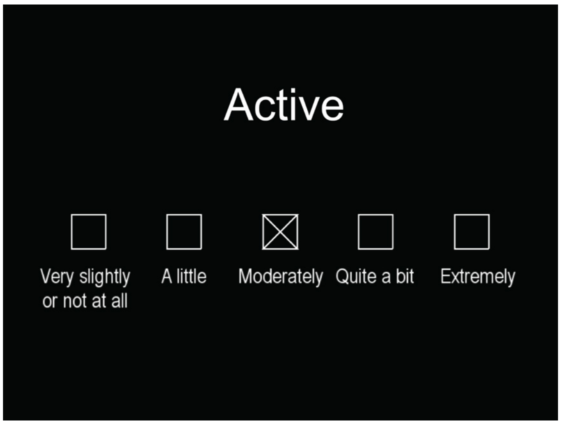
Attentive

Very slightly or not at all · A little · [✓ Moderately] · Quite a bit · Extremely

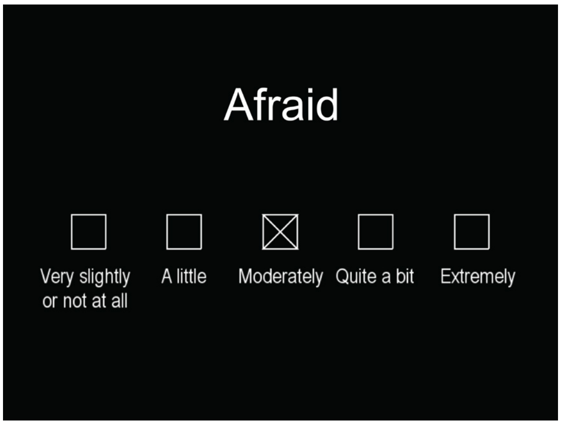


Jittery

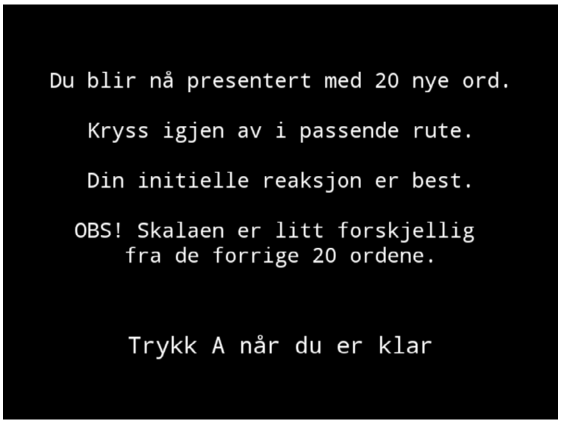
Very slightly or not at all · A little · [✓ Moderately] · Quite a bit · Extremely

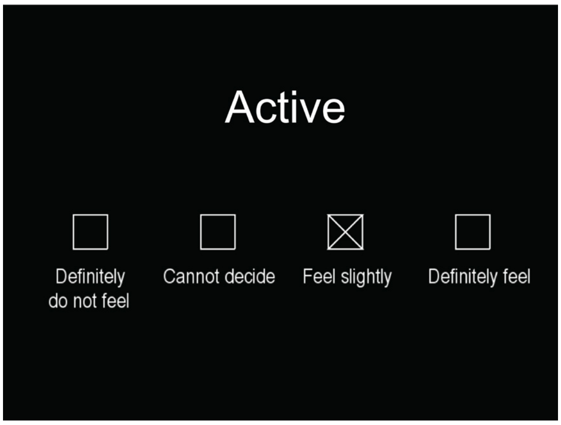

Active

Very slightly or not at all · A little · [✓ Moderately] · Quite a bit · Extremely

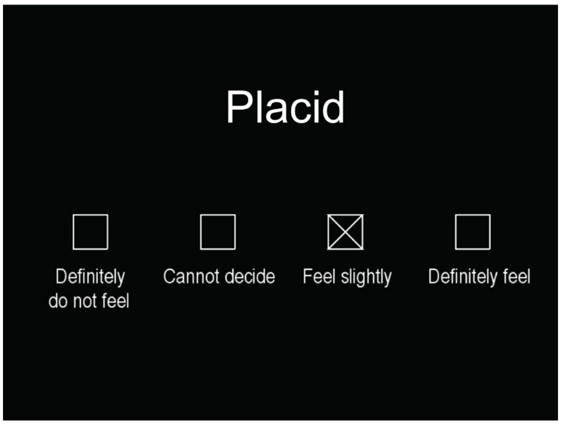

Afraid

Very slightly or not at all · A little · [✓ Moderately] · Quite a bit · Extremely

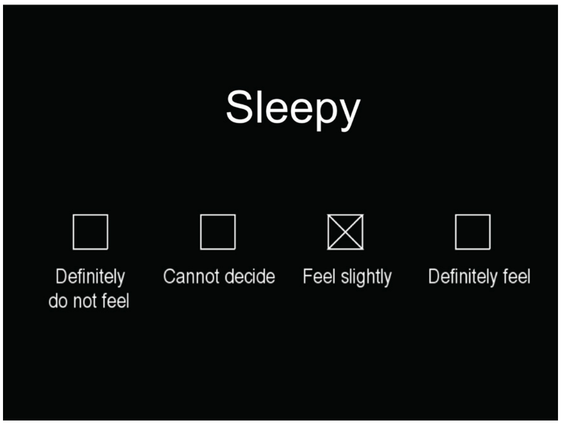

```
Du blir nå presentert med 20 nye ord.

Kryss igjen av i passende rute.

Din initielle reaksjon er best.

OBS! Skalaen er litt forskjellig
fra de forrige 20 ordene.


Trykk A når du er klar
```


Active

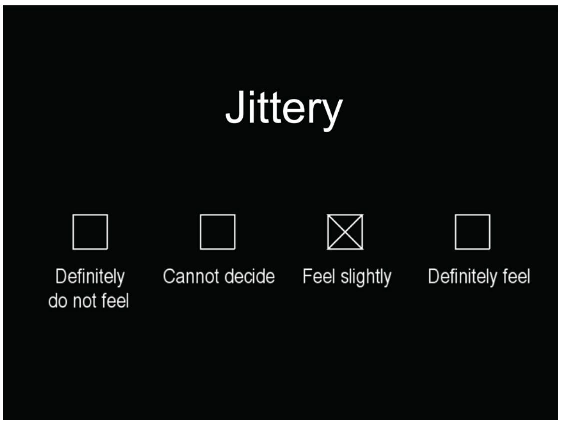
Definitely do not feel · Cannot decide · [✓ Feel slightly] · Definitely feel


Placid

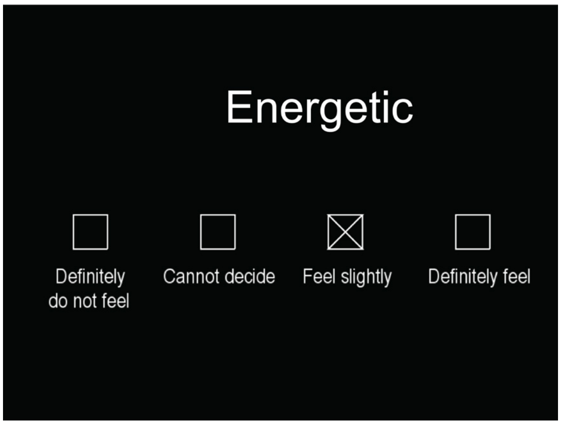
Definitely do not feel · Cannot decide · [✓ Feel slightly] · Definitely feel


Sleepy

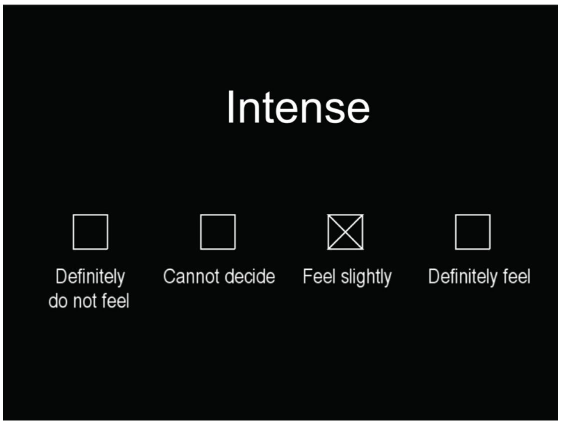
Definitely do not feel · Cannot decide · [✓ Feel slightly] · Definitely feel


Jittery

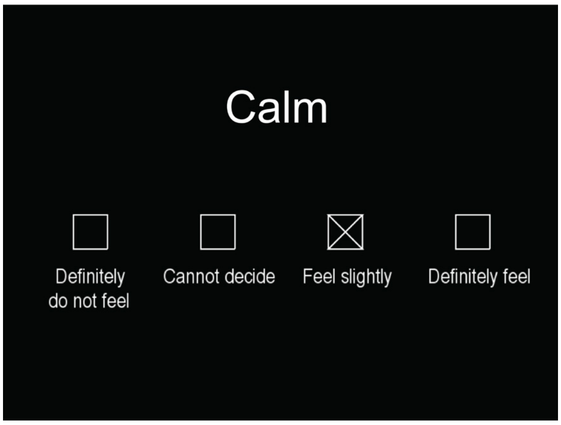
Definitely do not feel · Cannot decide · [✓ Feel slightly] · Definitely feel


Energetic

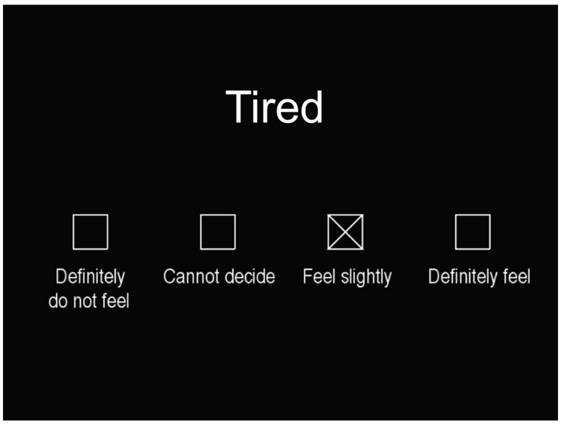
Definitely do not feel · Cannot decide · [✓ Feel slightly] · Definitely feel


Intense

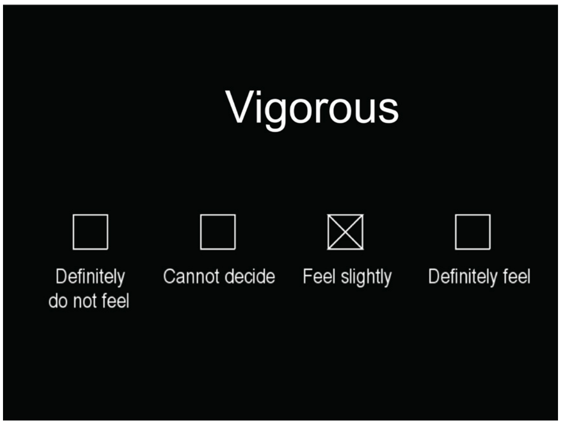
Definitely do not feel	Cannot decide	[✓ Feel slightly]	Definitely feel


Calm

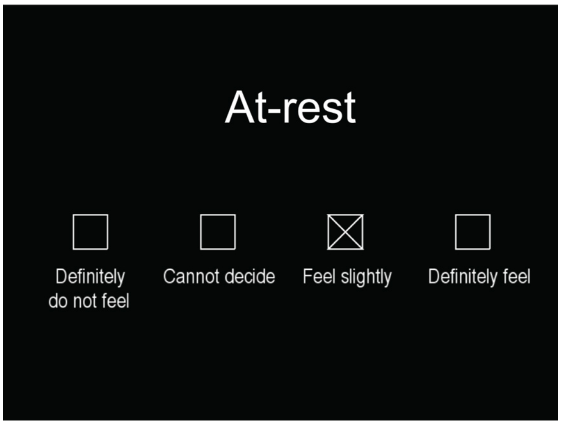
Definitely do not feel	Cannot decide	[✓ Feel slightly]	Definitely feel


Tired

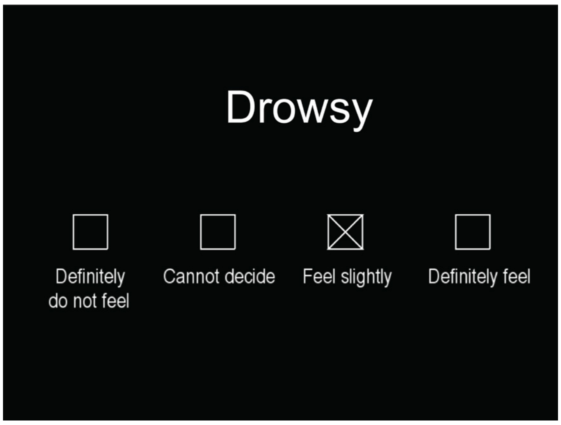
Definitely do not feel	Cannot decide	[✓ Feel slightly]	Definitely feel


Vigorous

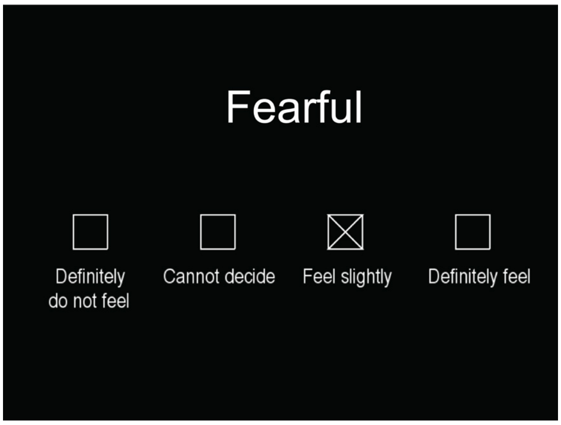
Definitely do not feel	Cannot decide	[✓ Feel slightly]	Definitely feel


At-rest

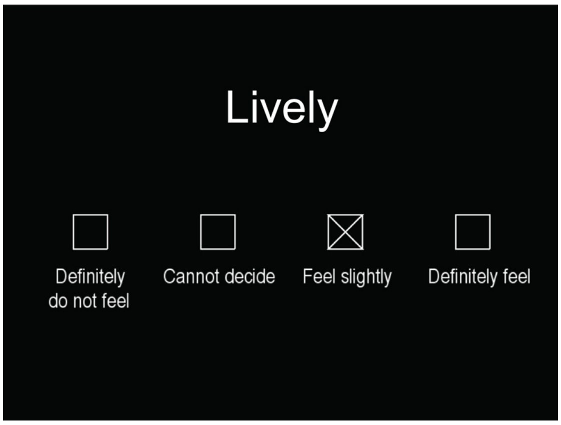
Definitely do not feel	Cannot decide	[✓ Feel slightly]	Definitely feel


Drowsy

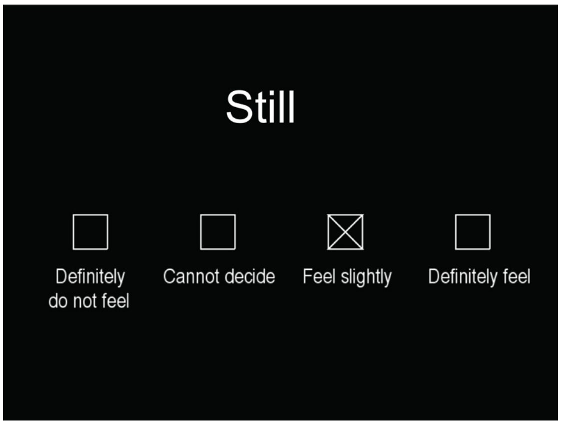
Definitely do not feel	Cannot decide	[✓ Feel slightly]	Definitely feel


Fearful

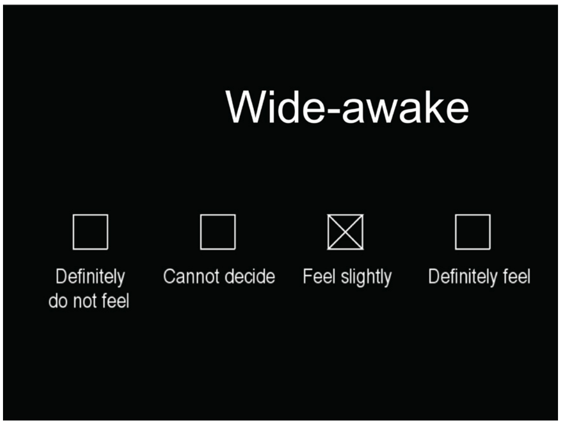
Definitely do not feel	Cannot decide	[✓ Feel slightly]	Definitely feel


Lively

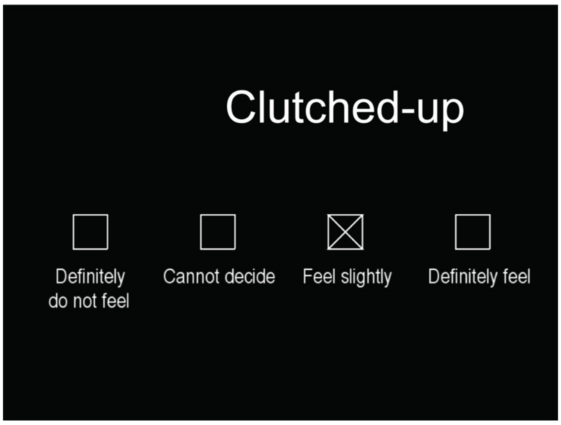
Definitely do not feel	Cannot decide	[✓ Feel slightly]	Definitely feel


Still

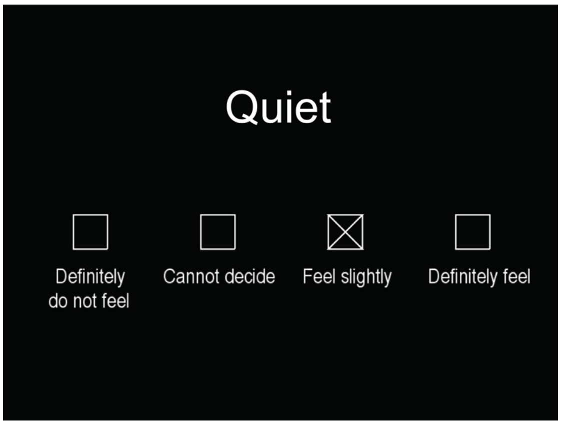
Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel


Wide-awake

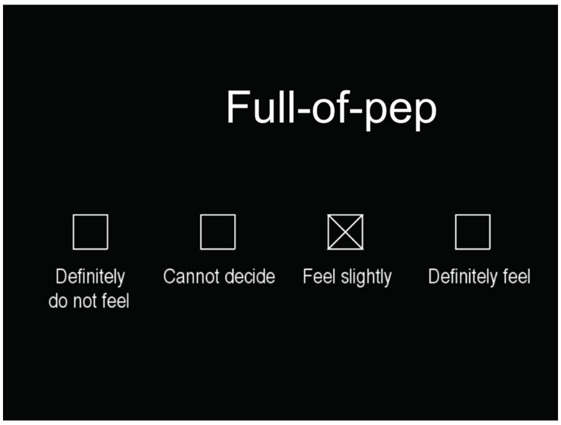
Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel


Clutched-up

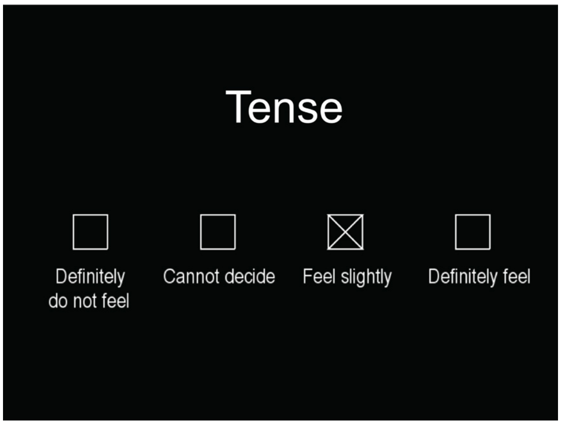
Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel


Quiet

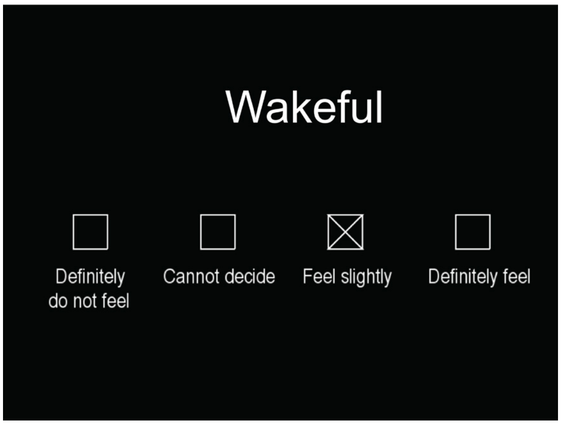
Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel


Full-of-pep

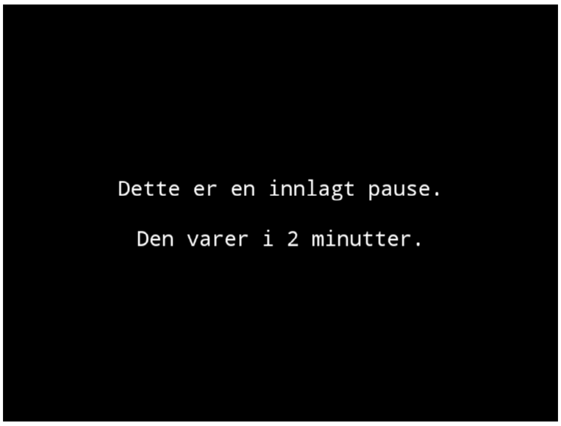
Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel


Tense

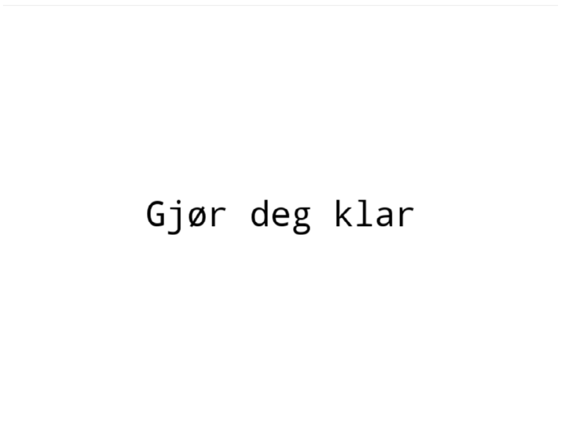
Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel


Wakeful

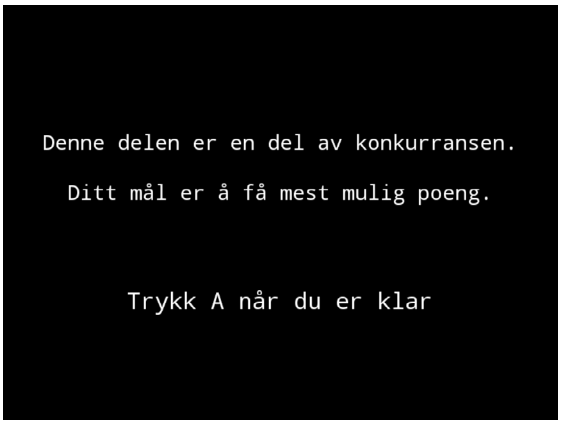
Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel


Dette er en innlagt pause.

Den varer i 2 minutter.



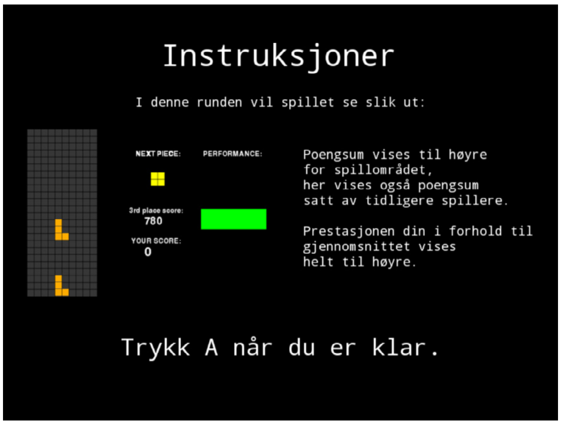
Gjør deg klar



Denne delen er en del av konkurransen.

Ditt mål er å få mest mulig poeng.

Trykk A når du er klar



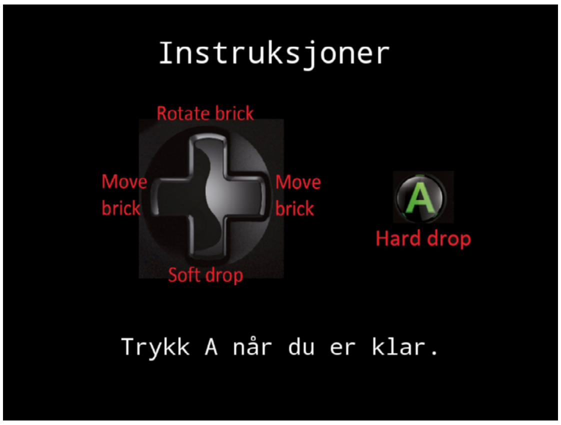
Instruksjoner

I denne runden vil spillet se slik ut:

Poengsum vises til høyre for spillområdet, her vises også poengsum satt av tidligere spillere.

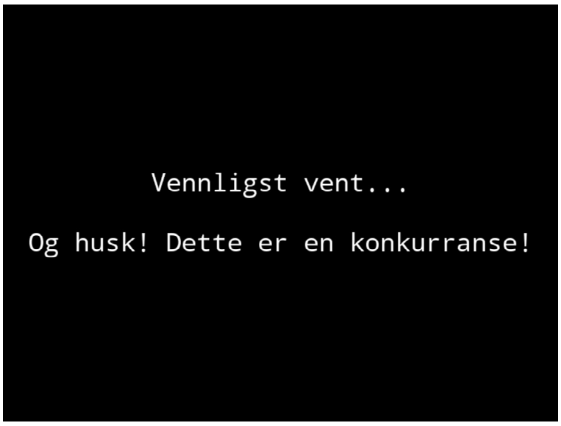
Prestasjonen din i forhold til gjennomsnittet vises helt til høyre.

Trykk A når du er klar.



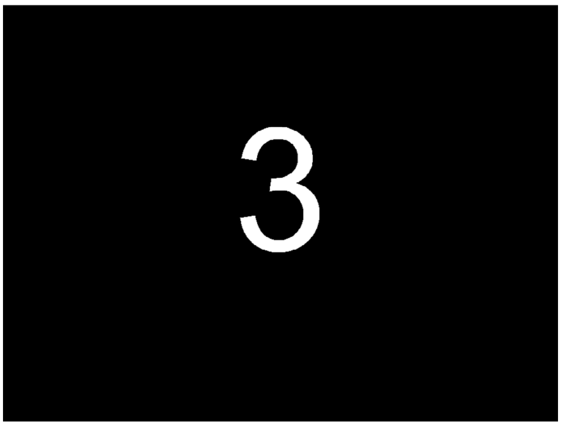
Instruksjoner

Trykk A når du er klar.



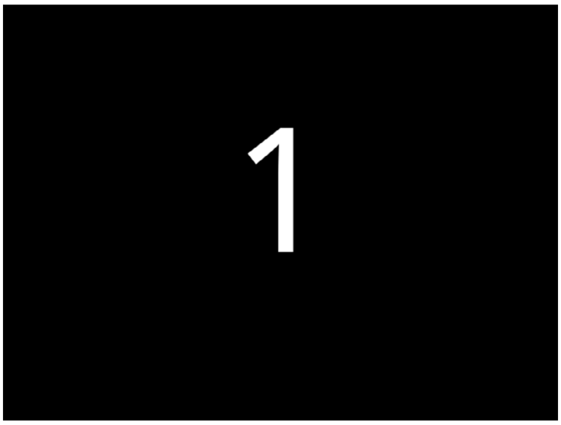
Vennligst vent...

Og husk! Dette er en konkurranse!

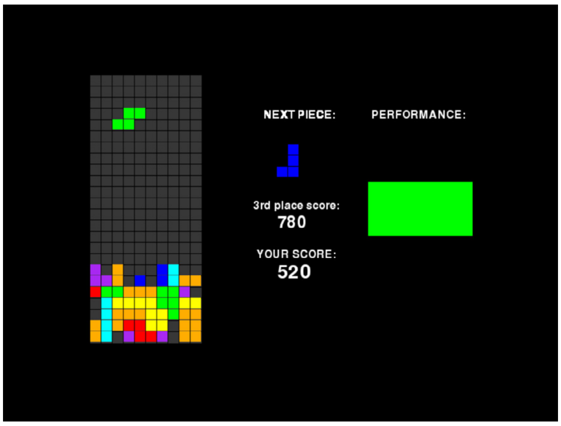


3

2



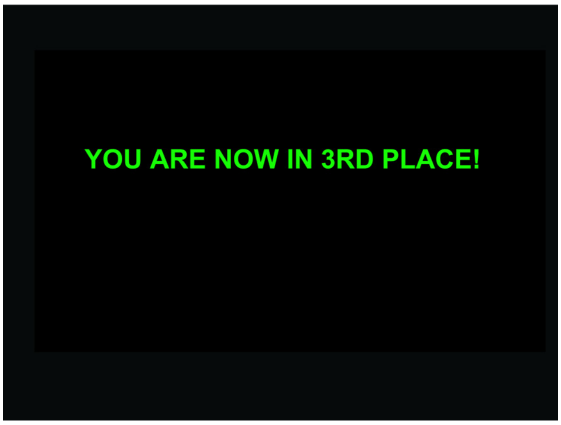
1

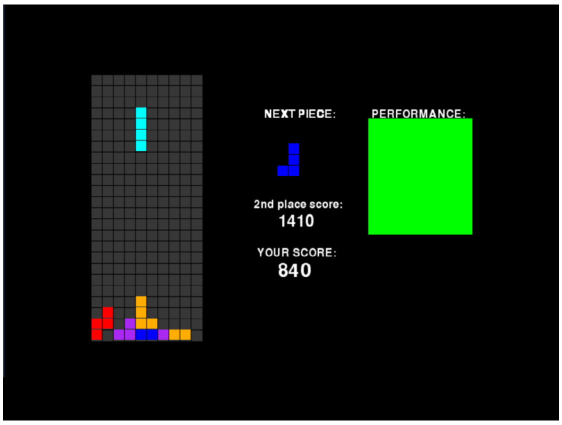

NEXT PIECE:

PERFORMANCE:

3rd place score:
780

YOUR SCORE:
520

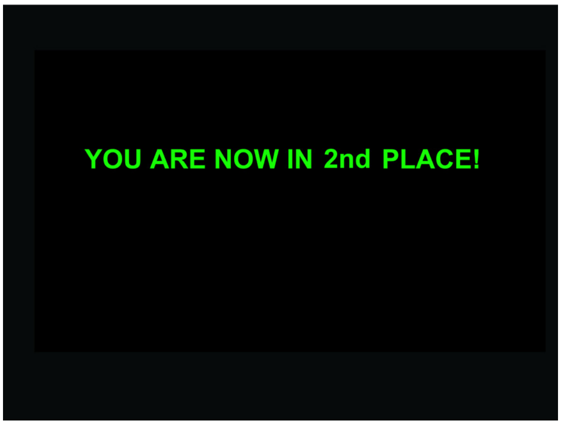

YOU ARE NOW IN 3RD PLACE!

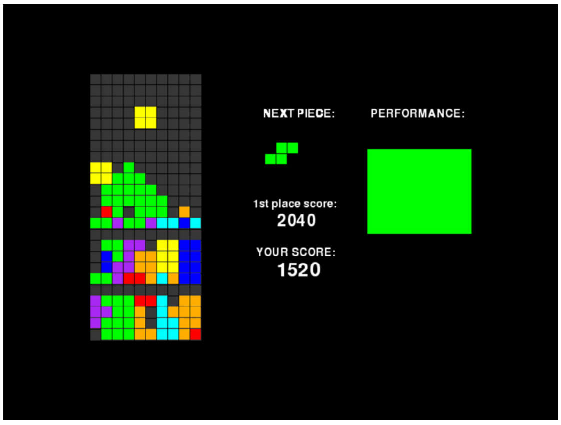

NEXT PIECE:

PERFORMANCE:

2nd place score:
1410

YOUR SCORE:
840

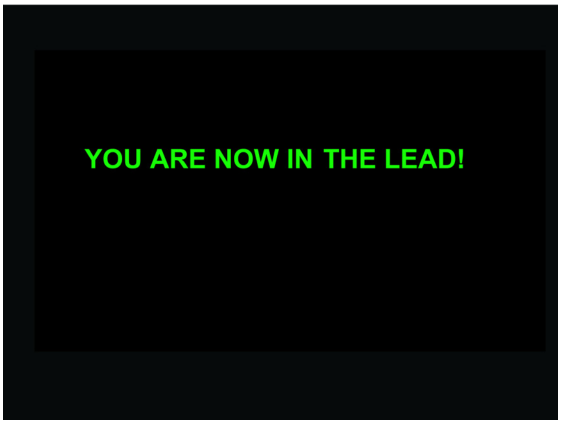

YOU ARE NOW IN 2nd PLACE!

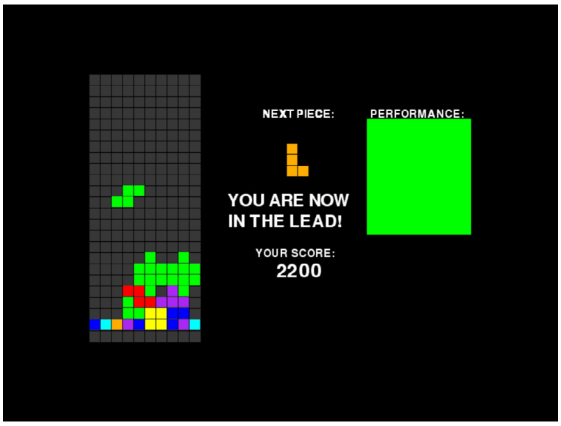

NEXT PIECE:

PERFORMANCE:

1st place score:
2040

YOUR SCORE:
1520

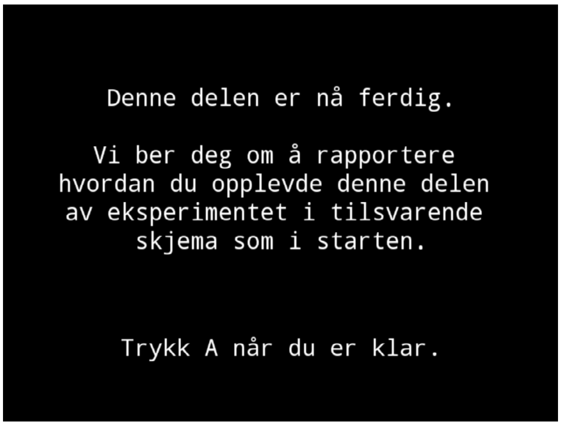

YOU ARE NOW IN THE LEAD!


NEXT PIECE:

PERFORMANCE:

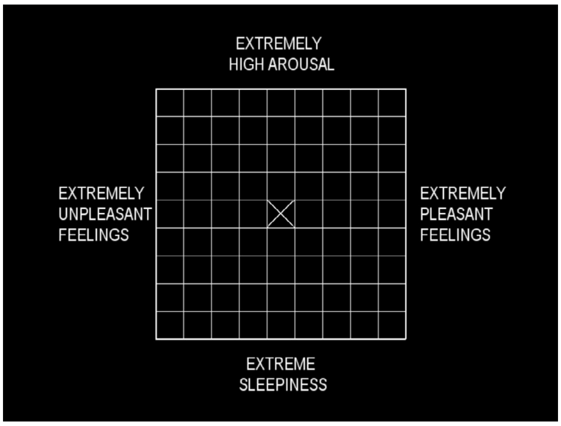
YOU ARE NOW IN THE LEAD!

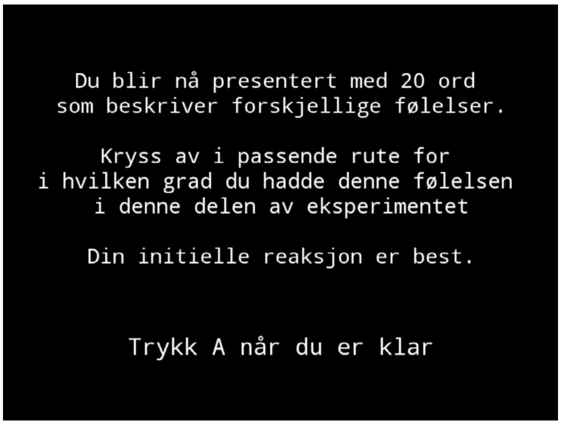
YOUR SCORE:
2200


Denne delen er nå ferdig.

Vi ber deg om å rapportere hvordan du opplevde denne delen av eksperimentet i tilsvarende skjema som i starten.

Trykk A når du er klar.

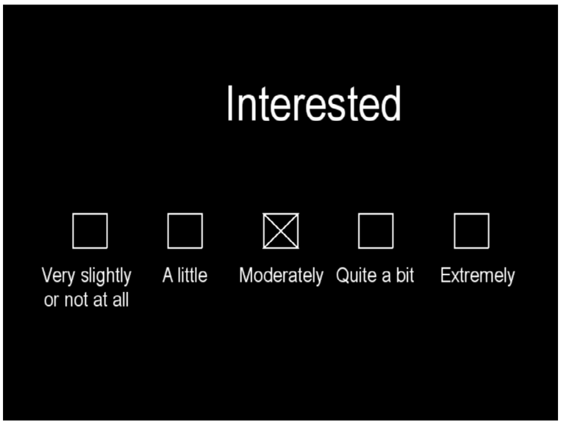



Du blir nå presentert med 20 ord
som beskriver forskjellige følelser.

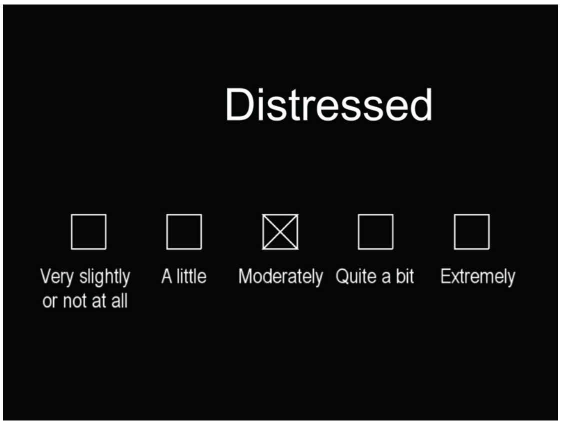
Kryss av i passende rute for
i hvilken grad du hadde denne følelsen
i denne delen av eksperimentet

Din initielle reaksjon er best.

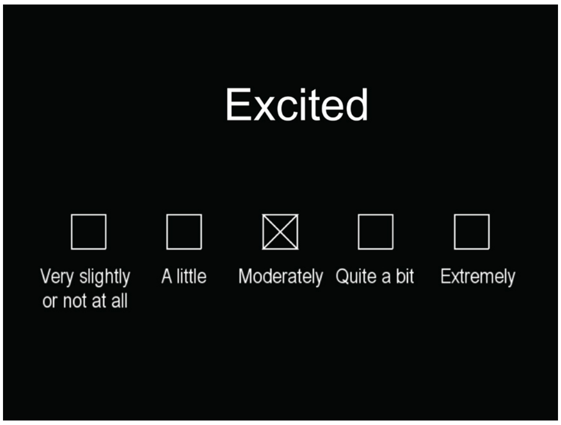
Trykk A når du er klar


Interested

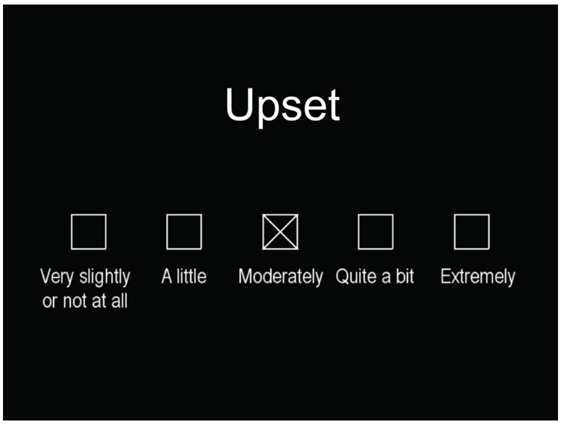
Very slightly or not at all | A little | [✓ Moderately] | Quite a bit | Extremely


Distressed

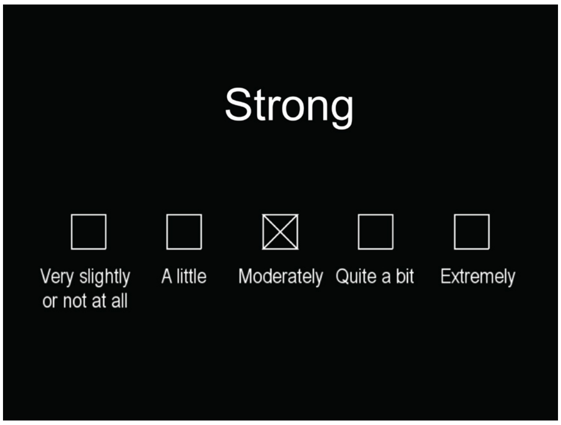
Very slightly or not at all | A little | [✓ Moderately] | Quite a bit | Extremely


Excited

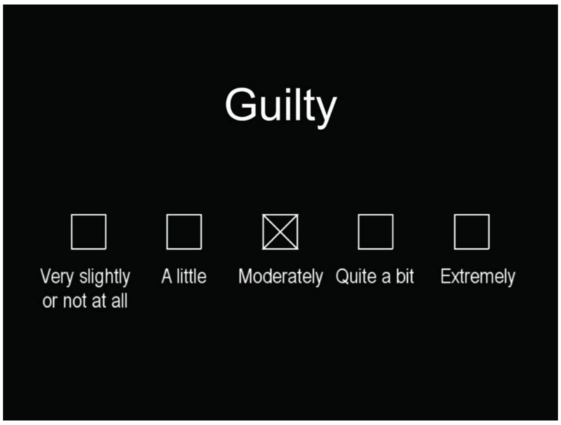
Very slightly or not at all | A little | [✓ Moderately] | Quite a bit | Extremely


Upset

Very slightly or not at all | A little | [✓ Moderately] | Quite a bit | Extremely


Strong

Very slightly or not at all | A little | [✓ Moderately] | Quite a bit | Extremely


Guilty

Very slightly or not at all | A little | [✓ Moderately] | Quite a bit | Extremely

Scared

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely

Hostile

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely

Enthusiastic

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely

Proud

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely

Irritable

☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely

Alert

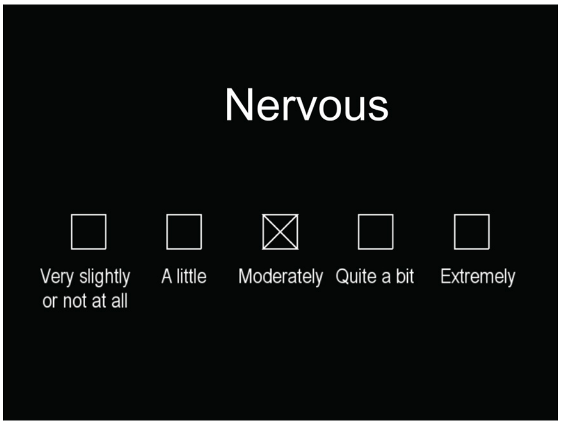
☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely

Ashamed

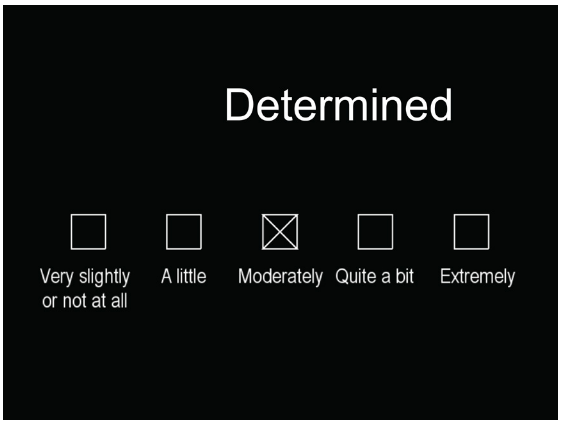
☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely

Inspired

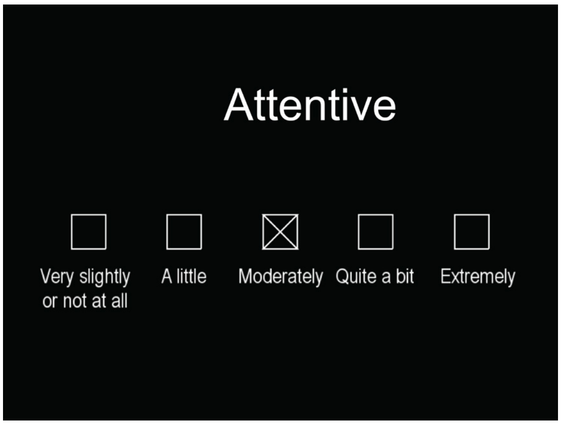
☐ Very slightly or not at all ☐ A little [✓ Moderately] ☐ Quite a bit ☐ Extremely


Nervous

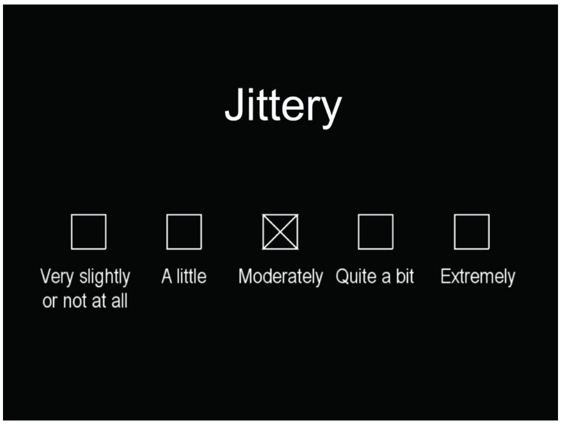
Very slightly or not at all | A little | [✓ Moderately] | Quite a bit | Extremely


Determined

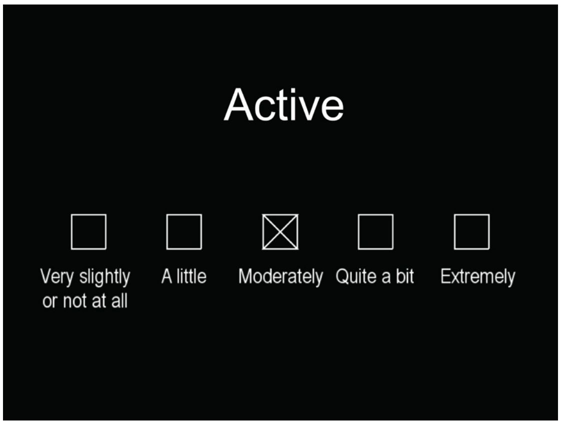
Very slightly or not at all | A little | [✓ Moderately] | Quite a bit | Extremely


Attentive

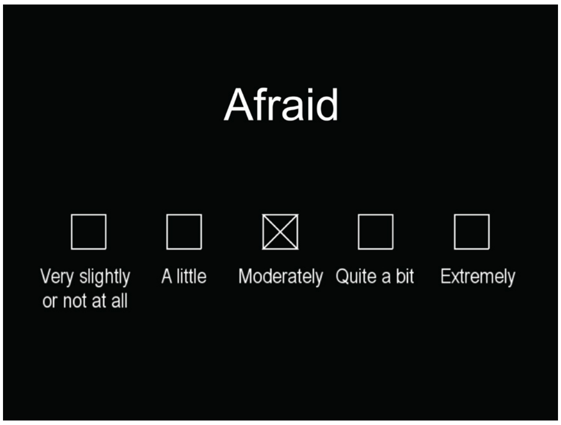
Very slightly or not at all | A little | [✓ Moderately] | Quite a bit | Extremely


Jittery

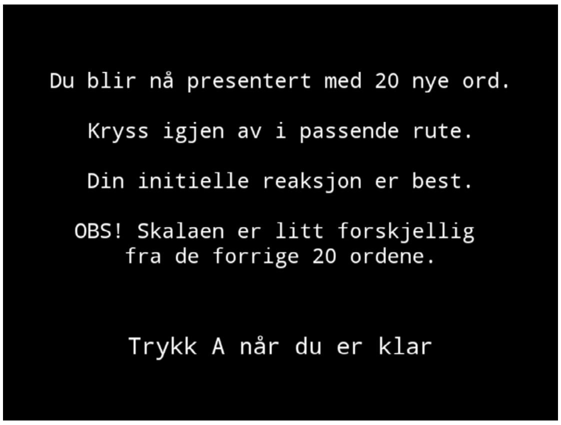
Very slightly or not at all | A little | [✓ Moderately] | Quite a bit | Extremely


Active

Very slightly or not at all | A little | [✓ Moderately] | Quite a bit | Extremely


Afraid

Very slightly or not at all | A little | [✓ Moderately] | Quite a bit | Extremely

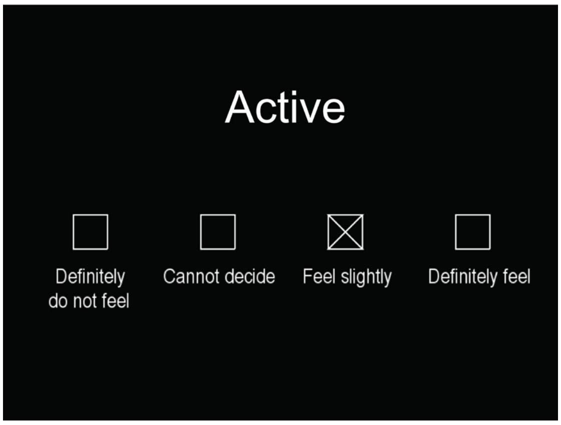

Du blir nå presentert med 20 nye ord.

Kryss igjen av i passende rute.

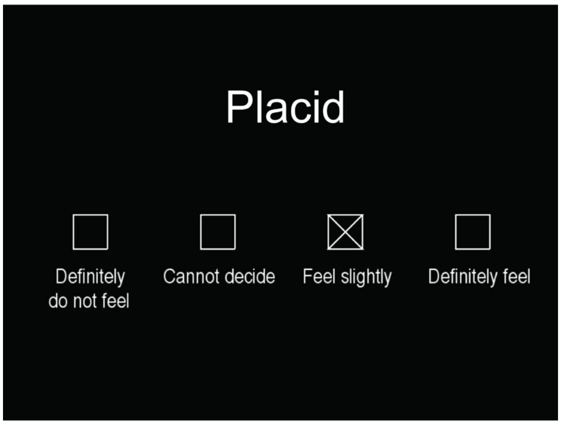
Din initielle reaksjon er best.

OBS! Skalaen er litt forskjellig fra de forrige 20 ordene.

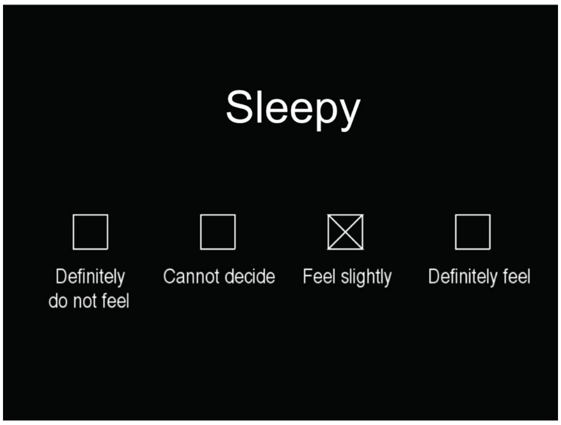
Trykk A når du er klar


Active

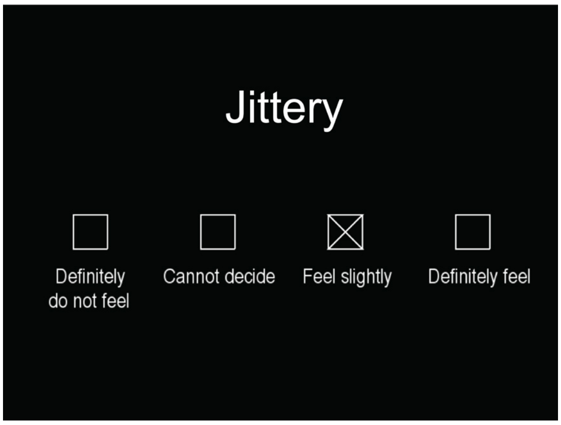
Definitely do not feel | Cannot decide | [✓ Feel slightly] | Definitely feel


Placid

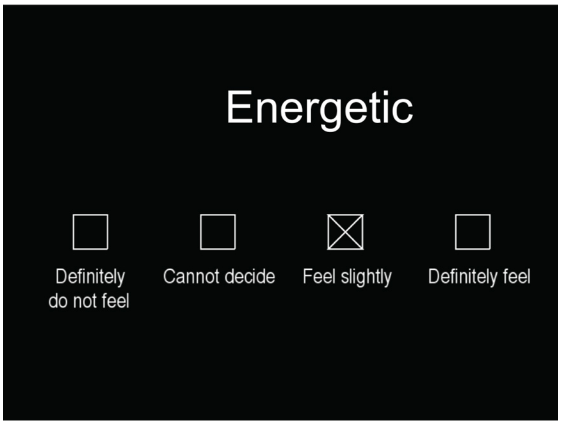
Definitely do not feel Cannot decide [✓ Feel slightly] Definitely feel


Sleepy

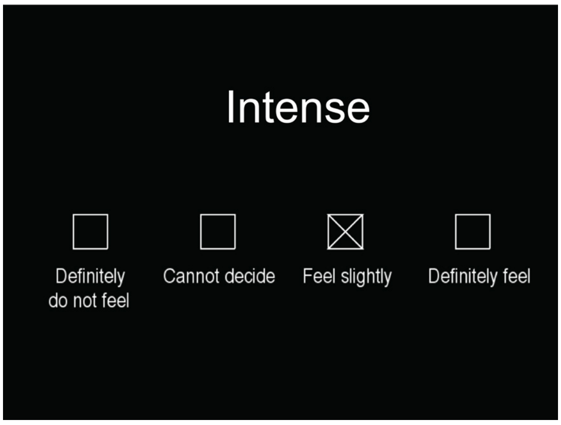
Definitely do not feel Cannot decide [✓ Feel slightly] Definitely feel


Jittery

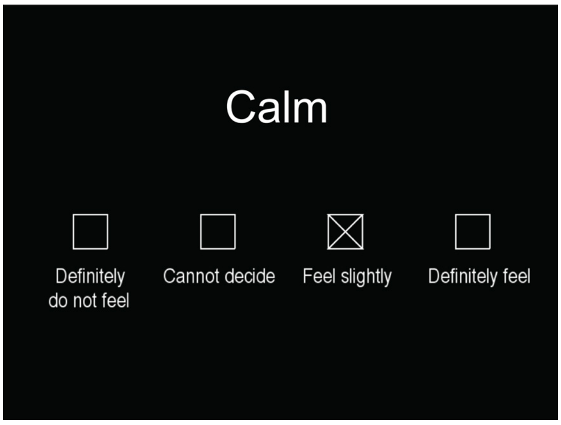
Definitely do not feel Cannot decide [✓ Feel slightly] Definitely feel


Energetic

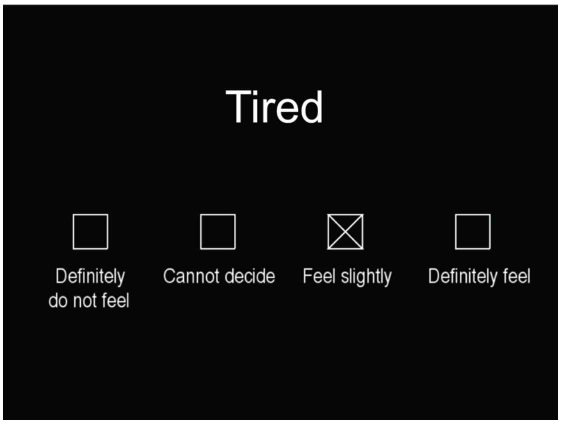
Definitely do not feel Cannot decide [✓ Feel slightly] Definitely feel


Intense

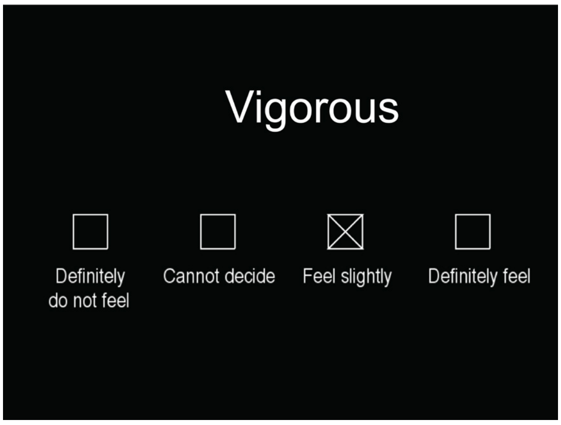
Definitely do not feel Cannot decide [✓ Feel slightly] Definitely feel


Calm

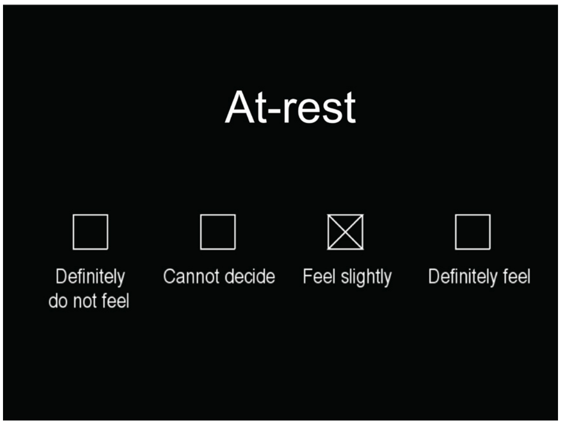
Definitely do not feel Cannot decide [✓ Feel slightly] Definitely feel


Tired

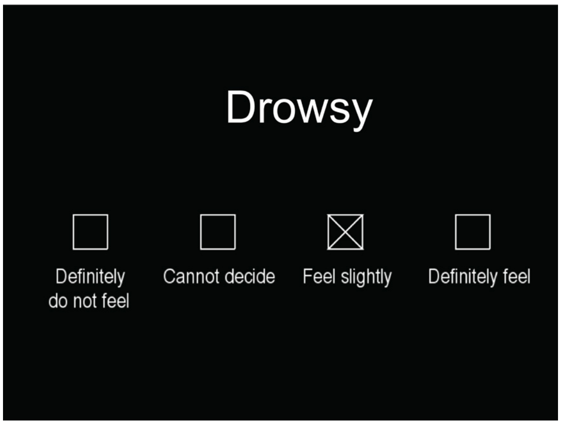
Definitely do not feel Cannot decide [✓ Feel slightly] Definitely feel


Vigorous

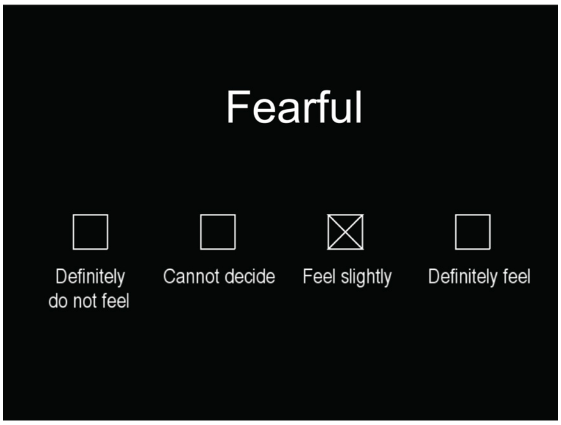
Definitely do not feel Cannot decide [✓ Feel slightly] Definitely feel


At-rest

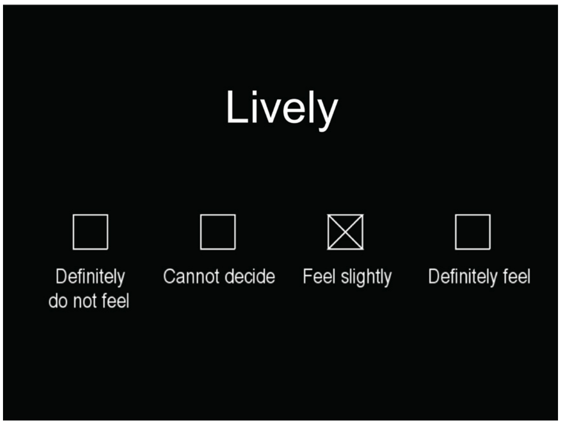
☐ Definitely do not feel ☐ Cannot decide [✓ Feel slightly] ☐ Definitely feel


Drowsy

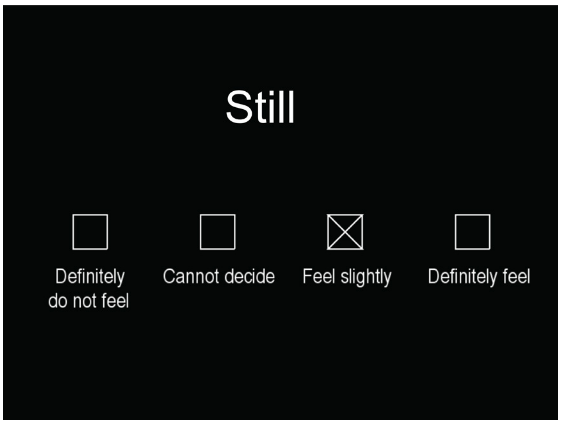
☐ Definitely do not feel ☐ Cannot decide [✓ Feel slightly] ☐ Definitely feel


Fearful

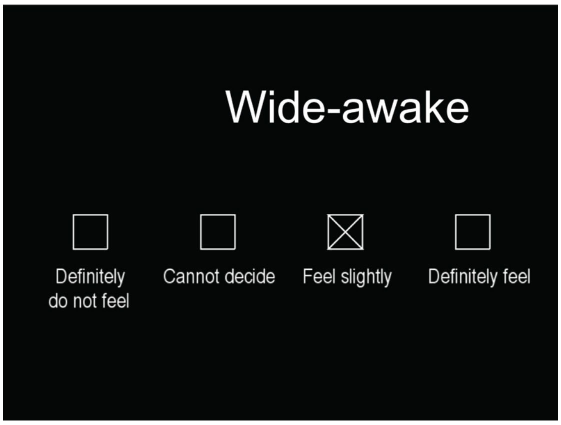
☐ Definitely do not feel ☐ Cannot decide [✓ Feel slightly] ☐ Definitely feel


Lively

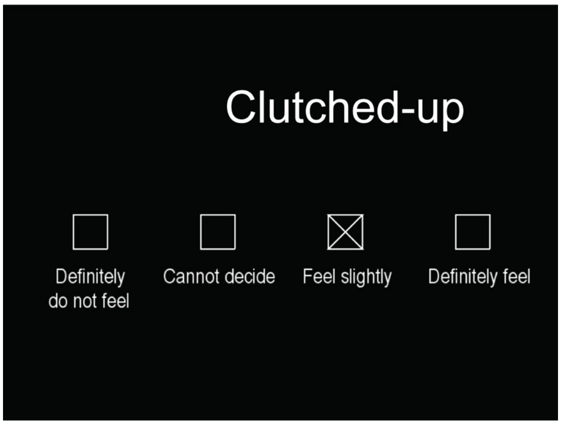
☐ Definitely do not feel ☐ Cannot decide [✓ Feel slightly] ☐ Definitely feel


Still

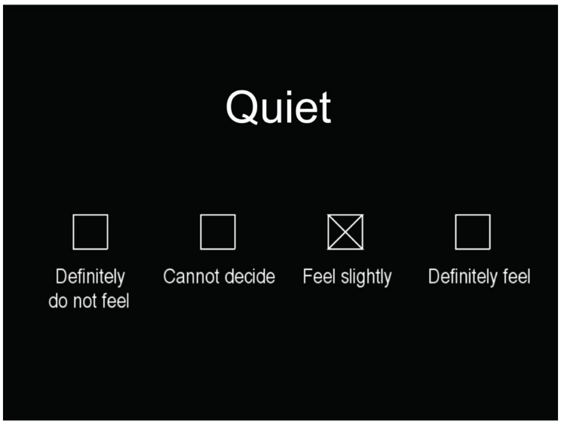
☐ Definitely do not feel ☐ Cannot decide [✓ Feel slightly] ☐ Definitely feel


Wide-awake

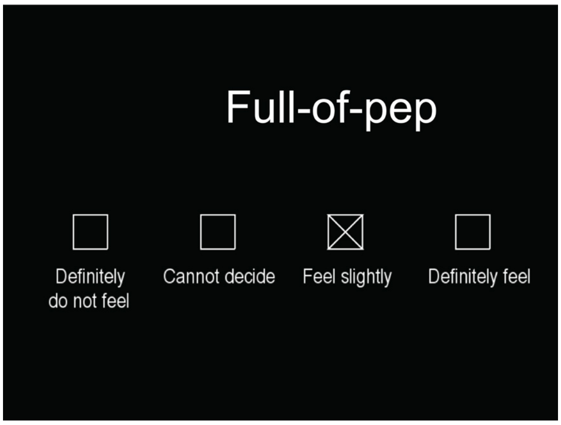
☐ Definitely do not feel ☐ Cannot decide [✓ Feel slightly] ☐ Definitely feel


Clutched-up

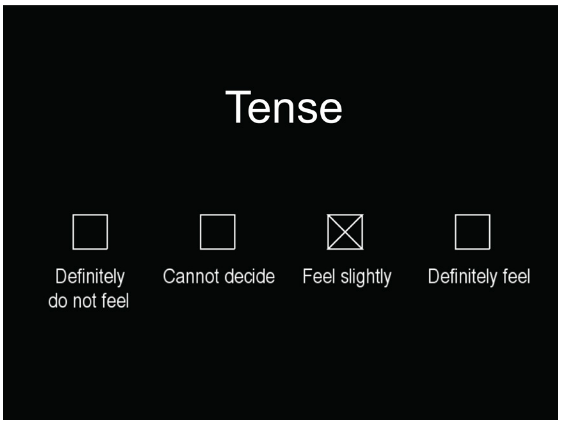
☐ Definitely do not feel ☐ Cannot decide [✓ Feel slightly] ☐ Definitely feel


Quiet

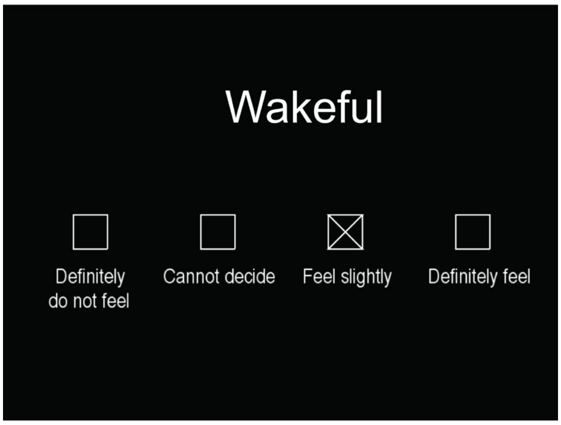
☐ Definitely do not feel ☐ Cannot decide [✓ Feel slightly] ☐ Definitely feel


Full-of-pep

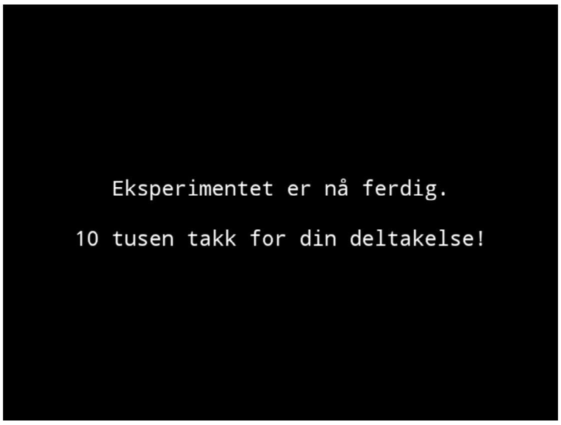
Definitely do not feel · Cannot decide · [✓ Feel slightly] · Definitely feel


Tense

Definitely do not feel · Cannot decide · [✓ Feel slightly] · Definitely feel


Wakeful

Definitely do not feel · Cannot decide · [✓ Feel slightly] · Definitely feel


```
Eksperimentet er nå ferdig.

10 tusen takk for din deltakelse!
```

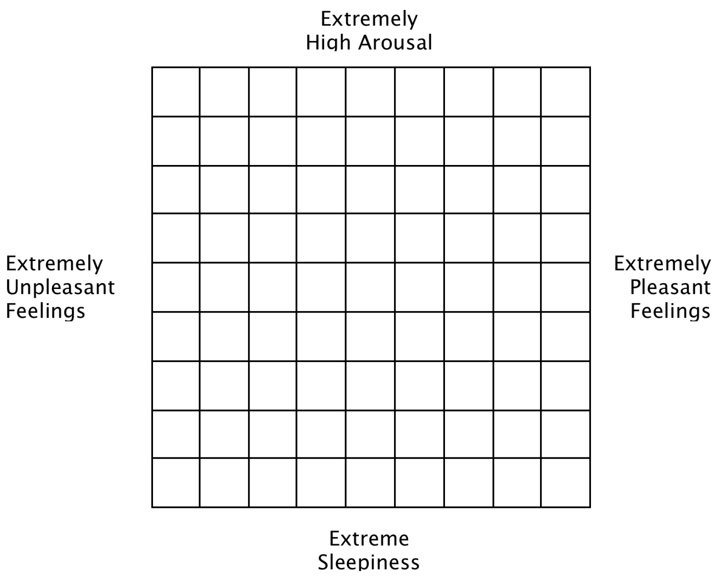
This page is intentionally left blank.

Appendix B - The Affect Grid

Originally from the appendix of (Russel et al., 1989).

Please use the affect grid below to describe how you feel right now.

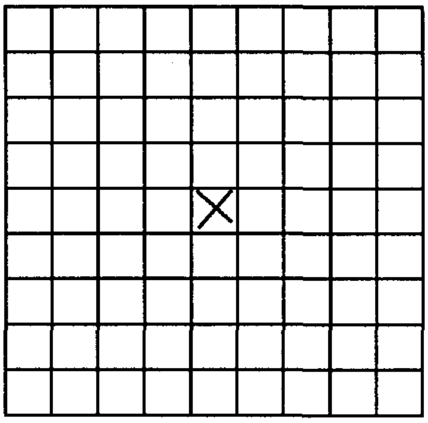
(For instructions on how to use the affect grid below, please refer to the following two pages)



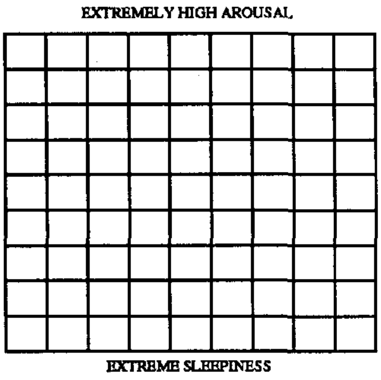
Appendix

The Affect Grid

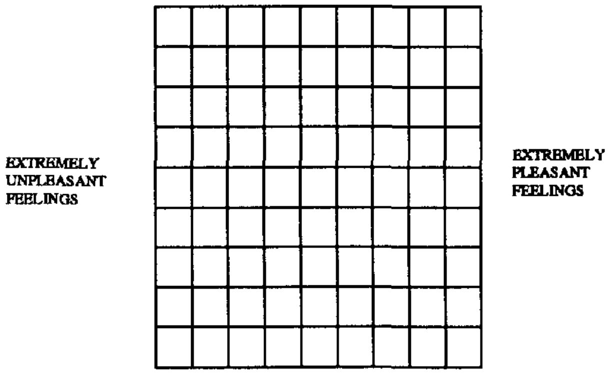
You use the "affect grid" to describe feelings. It is in the form of a square—a kind of map for feelings. The center of the square (marked by X in the grid below) represents a neutral, average, everyday feeling. It is neither positive nor negative.



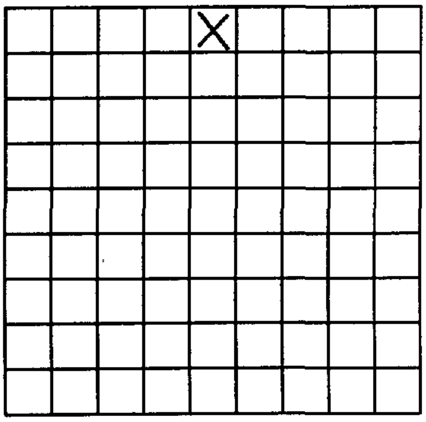
The vertical dimension of the map represents degree of arousal. Arousal has to do with how wide awake, alert, or activated a person feels—independent of whether the feeling is positive or negative. The top half is for feelings that are above average in arousal. The lower half for feelings below average. The bottom represents sleep, and the higher you go, the more awake a person feels. So, the next step up from the bottom would be half awake/half asleep. At the top of the square is maximum arousal. If you imagine a state we might call frantic excitement (remembering that it could be either positive or negative), then this feeling would define the top of the grid.



The right half of the grid represents pleasant feelings. The farther to the right the more pleasant. The left half represents unpleasant feelings. The farther to the left, the more unpleasant.

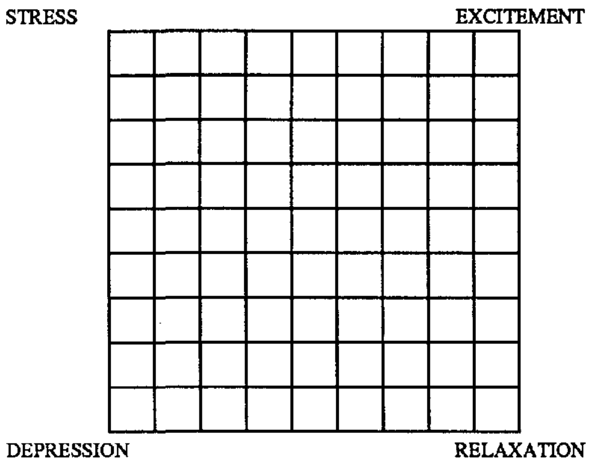


If the "frantic excitement" was positive it would, of course, fall on the right half of the grid. The more positive, the farther to the right. If the "frantic excitement" was negative, it would fall on the left half of the grid. The more negative, the farther to the left. If the "frantic excitement" was neither positive nor negative, then it would fall in the middle square of the top row, as shown below.



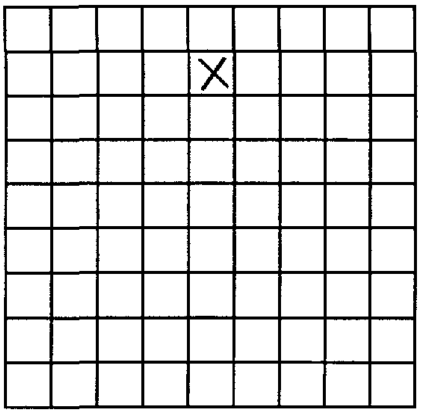
Other areas of the grid can be labeled as well. Up and to the right are feelings of ecstasy, excitement, joy. Opposite these, down and to the left, are feelings of depression, melancholy, sadness, and gloom.

Up and to the left are feelings of stress and tension. Opposite these, down and to the right, are feelings of calm, relaxation, serenity.

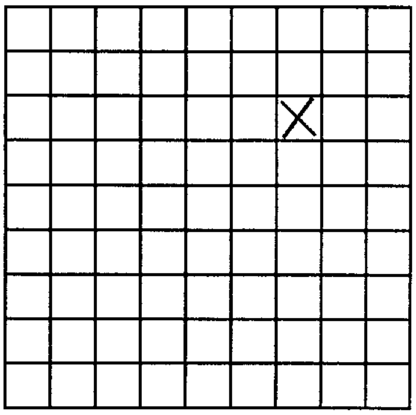


Feelings are complex. They come in all shades and degrees. The labels we have given are merely landmarks to help you understand the affect grid. When actually using the grid, put an X anywhere in the grid to indicate the exact shade and intensity of feeling. Please look over the entire grid to get a feel for the meaning of the various areas.

EXAMPLE: Suppose that you were just surprised. Suppose further that the surprise was neither pleasant nor unpleasant. Probably you would feel more aroused than average. You might put your mark as shown.



EXAMPLE: Suppose, instead, that you were only mildly surprised but that the surprise was a mildly pleasant one. You might put your mark as shown below.



Received October 22, 1987
Revision received January 13, 1989
Accepted February 10, 1989 ■

Affect Grid GUI Code

```
import pygame
import time
import sys
from pygame import K_SPACE, K_w, K_s, K_a, K_d
from pygame.locals import *

def get_pygame_events():
pygame_events = pygame.event.get()
return pygame_events

st = 1
affectresult = (0,0)
pygame.init()
sizex = 800
sizey = 600
screen = pygame.display.set_mode((sizex,sizey))

#colors
red = (255,0,0)
green = (0,255,0)
blue = (0,0,255)
black = (0,0,0)
white = (255,255,255)

myfont = pygame.font.SysFont("arial", 25)
extremely = myfont.render("EXTREMELY", 1, white)
extreme = myfont.render("EXTREME", 1, white)
highar = myfont.render("HIGH AROUSAL",1,white)
lowar = myfont.render("SLEEPINESS", 1, white)
unpleasant = myfont.render("UNPLEASANT", 1, white)
feelings = myfont.render("FEELINGS", 1, white)
pleasant = myfont.render("PLEASANT", 1, white)


pygame.draw.rect(screen, white, ((sizex/2)-180,(sizey/2)-180, 360, 360),


'''
x = 0
y = 0
for i in range(0,9):
for j in range(0,9):
pygame.draw.rect(screen, white, (200+i, 100+i, 40, 40), 1)
```

```
x += 40
x=0
y+=40
'''

offset = 40
theight = 400
twidth = 400
for i in xrange(9): pygame.draw.line(screen, white, (220+i*offset, 120),
for i in xrange(9): pygame.draw.line(screen, white, (220, 120+i*offset),

screen.blit(extremely, (335, 40))
screen.blit(highar, (325, 70))
screen.blit(extreme, (350, 500))
screen.blit(lowar, (340, 530))
screen.blit(extremely, (80, 255))
screen.blit(unpleasant, (80, 285))
screen.blit(feelings, (80, 315))
screen.blit(extremely, (600, 255))
screen.blit(pleasant, (600, 285))
screen.blit(feelings, (600, 315))

posx = 380
posy = 280

pygame.draw.line(screen,white,(posx,posy),(posx+40,posy+40))
pygame.draw.line(screen,white,(posx+40,posy),(posx,posy+40))

def move():
pygame.draw.rect(screen,black,(lastposx+2,lastposy+2,36,36),0)
pygame.draw.line(screen,white,(posx+2,posy+2),(posx+36,posy+36),2)
pygame.draw.line(screen,white,(posx+36,posy+2),(posx+2,posy+36),2)

while st == 1:
lastposx=posx
lastposy=posy
keys_pressed = get_pygame_events()
for event in keys_pressed:
if event.type == pygame.KEYDOWN:
if event.key == K_w:
posy-=40
if event.key == K_s:
posy+=40
if event.key == K_a:
posx-=40
```

```
if event.key == K_d:
posx+=40
if event.key == K_SPACE:
affectresult = (((posx-220)/40)+1, ((posy-120)/40)+1)
log.write(affectresult)
st = 0
if posx<220 or posx>540:
posx = lastposx
if posy<120 or posy>440:
posy = lastposy
move()
pygame.display.update()
```

This page is intentionally left blank.

Appendix C - PANAS

Originally from the appendix of (Watson et al., 1988).

Appendix

The PANAS

This scale consists of a number of words that describe different feelings and emotions. Read each item and then mark the appropriate answer in the space next to that word. Indicate to what extent [INSERT APPROPRIATE TIME INSTRUCTIONS HERE]. Use the following scale to record your answers.

1	2	3	4	5
very slightly or not at all	a little	moderately	quite a bit	extremely

______ interested	______ irritable
______ distressed	______ alert
______ excited	______ ashamed
______ upset	______ inspired
______ strong	______ nervous
______ guilty	______ determined
______ scared	______ attentive
______ hostile	______ jittery
______ enthusiastic	______ active
______ proud	______ afraid

We have used PANAS with the following time instructions:

Moment	(you feel this way right now, that is, at the present moment)
Today	(you have felt this way today)
Past few days	(you have felt this way during the past few days)
Week	(you have felt this way during the past week)
Past few weeks	(you have felt this way during the past few weeks)
Year	(you have felt this way during the past year)
General	(you generally feel this way, that is, how you feel on the average)

Received May 10, 1987
Revision received September 14, 1987
Accepted November 11, 1987 ■

PANAS GUI Code

```
import pygame
import time
import sys
from pygame import K_SPACE, K_w, K_s, K_a, K_d
from pygame.locals import *

def get_pygame_events():
        pygame_events = pygame.event.get()
        return pygame_events

wordcount = 0
st = 1
pygame.init()
sizex = 800
sizey = 600
screen = pygame.display.set_mode((sizex,sizey))
pos = 3
results = []

#colors
red = (255,0,0)
green = (0,255,0)
blue = (0,0,255)
black = (0,0,0)
white = (255,255,255)

myfont = pygame.font.SysFont("arial", 70)
myfont2 = pygame.font.SysFont("arial", 30)
alt1 = myfont2.render("Very slightly",1,white)
alt12 = myfont2.render("or not at all",1,white)
alt2 = myfont2.render("A little",1,white)
alt3 = myfont2.render("Moderately",1,white)
alt4 = myfont2.render("Quite a bit",1,white)
alt5 = myfont2.render("Extremely",1,white)
screen.blit(alt1, (55, 370))
screen.blit(alt12, (59, 405))
screen.blit(alt2, (230, 370))
screen.blit(alt3, (340, 370))
screen.blit(alt4, (480, 370))
screen.blit(alt5, (630, 370))


wordlist = ['Interested', 'Distressed', 'Excited','Upset','
   Strong','Guilty','Scared','Hostile','Enthusiastic','
```

```
    Proud','Irritable','Alert','Ashamed','Inspired','Nervous
    ','Determined','Attentive','Jittery','Active','Afraid']

def words(counter):
        return wordlist[counter]

def drawWord(counter):
        pygame.draw.rect(screen,black,(0,0,800,200),0)
        word = myfont.render(wordlist[counter], 1, white)
        screen.blit(word, (320, 100))

count=0
for count in xrange(5): pygame.draw.rect(screen, white,
    (100+count*(87.5+50),300,50,50),2);count+=1


posx = 375
posy = 300

pygame.draw.line(screen,white,(posx,posy),(posx+50,posy+50)
    ,3)
pygame.draw.line(screen,white,(posx+50,posy),(posx,posy+50)
    ,3)

def move():
        pygame.draw.rect(screen,black,(lastposx+2,lastposy
            +2,47,47),0)
        pygame.draw.line(screen,white,(posx,posy),(posx+50,
            posy+50),3)
        pygame.draw.line(screen,white,(posx+50,posy),(posx,
            posy+50),3)

def erase():
        pygame.draw.rect(screen,black,(100+2,300+2,47,47)
            ,0)
        pygame.draw.rect(screen,black,(237.5+2,300+2,47,47)
            ,0)
        pygame.draw.rect(screen,black,(375+2,300+2,47,47)
            ,0)
        pygame.draw.rect(screen,black,(512.5+2,300+2,47,47)
            ,0)
        pygame.draw.rect(screen,black,(650+2,300+2,47,47)
            ,0)

while wordcount<20:
```

```
lastposx=posx
lastposy=posy
lastpos=pos
keys_pressed = get_pygame_events()
for event in keys_pressed:
        if event.type == pygame.KEYDOWN:
                if event.key == K_a:
                        posx-=87.5+50
                        pos-=1
                if event.key == K_d:
                        posx+=87.5+50
                        pos+=1
                if event.key == K_SPACE:
                        result = pos
                        results.append(result)
                        print(words(wordcount)+str(
                           result))
                        if wordcount == 19:
                                PA = results[0]+
                                   results[2]+
                                   results[4]+
                                   results[8]+
                                   results[9]+
                                   results[11]+
                                   results[13]+
                                   results[15]+
                                   results[16]+
                                   results[18]
                                NA = results[1]+
                                   results[3]+
                                   results[5]+
                                   results[6]+
                                   results[7]+
                                   results[10]+
                                   results[12]+
                                   results[14]+
                                   results[17]+
                                   results[19]
                                print('PA='+str(PA)
                                   )
                                print('NA='+str(NA)
                                   )
                        wordcount+=1
                        erase()
                        pos=3
                        posx=375
                        move()
if posx<100 or posx>=700:
        posx = lastposx
        pos=lastpos

move()
if wordcount<20:
        drawWord(wordcount)
pygame.display.update()
```

Appendix D - The Activation-Deactivation Adjective Check List (AD ACL)

Originally from the appendix of (Thayer, 1989)

APPENDIX I

The Activation-Deactivation Adjective Check List (AD ACL)

The AD ACL is a multidimensional test of various transitory arousal states, including energetic and tense arousal (see Chapter 3). It has been used widely in many psychophysiological (e.g., Mackay, 1980) and psychological contexts,* and it has taken a variety of language forms (e.g., Bohlin & Kjellberg, 1973—Swedish version; Grzegolowska-Klarkowska, 1980—Polish version; Mackay et al., 1978—Anglicized version). Within the wider dimensions of energetic and tense arousal are four subscales—Energy (General Activation), Tiredness (Deactivation-Sleep), Tension (High Activation), and Calmness (General Deactivation).

The above parenthetical designations were given in the 1960s (Thayer, 1967), before the multidimensional arousal model was conceptualized in its present form. If these parenthetical names were to be modifed at the present time, they would be somewhat different. For example, the parenthetical name associated with Tiredness would probably now be *General Deactivation,* thus indicating that it is likely to represent the opposite pole from *General Activation.* Other names associated with Tension and Calmness might be High and Low Preparatory-Emergency Activation (or Arousal), thus indicating the likely function of these kinds of arousal.

The self-rating response format used in this test originally followed a format employed by Nowlis (1965) with the Mood Adjective Check List. This four-point self-rating system is slightly unconventional in comparison with the more usual three-, five-, or seven-point formats used in a number of other adjective checklists. Also, the verbal anchors of the AD ACL (as well as of the Mood Adjective Check List), although quite meaningful, are not completely symmetrical.

In order to determine if these somewhat unconventional features result in important differences, a study was recently completed to compare factor structures using different self-rating formats (Thayer, 1986). In this research, little difference was observed between the usual AD ACL format and others. Additional evidence for the validity of the AD ACL format may be found in other studies that employed it, and that obtained findings consistent with both mood and general arousal theories (Purcell, 1982; Watson & Tellegen, 1985). Therefore, the format most often employed with the AD ACL appears to be satisfactory. Alternatively, other more conventional

* In addition to studies reviewed in this book, see *Social Science Citation Index* with Thayer (1967, 1978a, 1986) as search references.

formats probably can be employed with little difference in results so long as the factor groupings are maintained.

Following is the AD ACL Short Form with the self-descriptive adjectives of Energy (A1), Tiredness (A2), Tension (B1), and Calmness (B2). Scoring is based on four possible points for each adjective. A common procedure in many studies has been to score only A1 and B1, since they are the best indications of energetic and tense arousal, respectively. A2 and B2 are particularly useful if the primary purpose of a study is to focus on the low arousal states of each dimension (Tiredness and Calmness). However, use of the full range of dimensions tends to reduce somewhat the strength of the relationships observed between arousal and other behaviors. This may be because people often do not make good discriminations of states of calmness, or it may occur because different processes underlie the pole opposites of each dimension (see Chapter 3).

AD ACL Short Form

Each of the words on the back describes feelings or mood. Please use the rating scale next to each word to describe your feelings *at this moment.*

EXAMPLES:

relaxed	[✓ vv]	v	?	no	If you circle the double check (vv) it means that you *definitely feel* relaxed *at the moment.*
relaxed	vv	[✓ v]	?	no	If you circle the single check (v) it means that you feel slightly relaxed *at the moment.*
relaxed	vv	v	[✓ ?]	no	If you circle the question mark (?) it means that the word does not apply or you cannot decide if you feel relaxed *at the moment.*
relaxed	vv	v	?	[✓ no]	If you circle the no it means that you are *definitely not* relaxed *at the moment.*

Work rapidly, but please mark all the words. Your first reaction is best. This should take only a minute or two.

(Back page)

[✓ vv]	v	?	no	:	definitely feel
vv	[✓ v]	?	no	:	feel slightly
vv	v	[✓ ?]	no	:	cannot decide
vv	v	?	[✓ no]	:	definitely do not feel

active	vv	v	?	no	drowsy	vv	v	?	no
placid	vv	v	?	no	fearful	vv	v	?	no
sleepy	vv	v	?	no	lively	vv	v	?	no
jittery	vv	v	?	no	still	vv	v	?	no
energetic	vv	v	?	no	wide-awake	vv	v	?	no
intense	vv	v	?	no	clutched-up	vv	v	?	no
calm	vv	v	?	no	quiet	vv	v	?	no
tired	vv	v	?	no	full-of-pep	vv	v	?	no
vigorous	vv	v	?	no	tense	vv	v	?	no
at-rest	vv	v	?	no	wakeful	vv	v	?	no

The AD ACL is scored by assigning 4, 3, 2, and 1, respectively to the "vv, v, ?," and "no" scale points, and summing or averaging the five scores for each subscale. (An appropriate cardboard template can be easily constructed.) In order of appearance, the subscale adjectives are as follows: Energetic (active, energetic, vigorous, lively, full-of-pep); Tired (sleepy, tired, drowsy, wide-awake, wakeful); Tension (jittery, intense, fearful, clutched-up, tense); Calmness (placid, calm, at-rest, still, quiet). Scoring for "wakeful" and "wide-awake" must be reversed for the Tiredness subscale. Also, if full bipolar dimensions of energetic and tense arousal are of interest (see above), Tiredness and Calmness scores must be reversed (but not wakeful and wide-awake, in this case) before summing the ten scores.

The AD ACL Long Form (Thayer, 1967, 1978a) includes additional activation adjectives as well as filler adjectives to disguise the purpose of the test. It contains the same instructions except that respondents are told that the test will take only a couple of minutes to complete. Based on previous analyses (Thayer, 1967, 1978a), the following adjectives are included on this form. The designations A1, A2, A3, and A4 after each significantly loaded activation adjective represent the subscales of Energy, Tiredness, Tension, and Calmness, respectively.

In order of appearance, the adjectives are: carefree, serious, peppy (A1), pleased, placid (A4), leisurely (A4), sleepy (A2), jittery (A3), intense (A3), grouchy, energetic (A1), egotistic, calm (A3, A4), suspicious, tired (A2), regretful, stirred-up (A3), warm-hearted, vigorous (A1), engaged-in-thought, at-rest (A4), elated, drowsy (A2), witty, anxious (A3), aroused, fearful (A3), lively (A1), defiant, still (A4), self-centered, wide-awake (A1, A2), skeptical, activated (A1), sad, full-of-pep (A1), affectionate, quiet (A4), concentrating, sluggish (A1, A2), overjoyed, quick (A1), nonchalant, quiescent (A4), clutched-up (A3), wakeful (A1, A2), rebellious, active (A1), blue, alert (A1), tense (A3). Since different numbers of activation adjectives are included in the four factors, these factor scores must be averaged instead of just summed if interfactor comparisons are to be made.

AD ACL GUI Code

```
import pygame
import time
import sys
from pygame import K_SPACE, K_w, K_s, K_a, K_d
from pygame.locals import *

def get_pygame_events():
        pygame_events = pygame.event.get()
        return pygame_events

wordcount = 0
st = 1
pygame.init()
sizex = 800
sizey = 600
screen = pygame.display.set_mode((sizex,sizey))
pos = 3
results = []

#colors
red = (255,0,0)
green = (0,255,0)
blue = (0,0,255)
black = (0,0,0)
white = (255,255,255)

myfont = pygame.font.SysFont("arial", 70)
myfont2 = pygame.font.SysFont("arial", 30)
alt1 = myfont2.render("Definitely",1,white)
alt12 = myfont2.render("do not feel",1,white)
alt2 = myfont2.render("Cannot decide",1,white)
alt3 = myfont2.render("Feel slightly",1,white)
alt4 = myfont2.render("Definitely feel",1,white)
screen.blit(alt1, (75, 370))
screen.blit(alt12, (65, 405))
screen.blit(alt2, (230, 370))
screen.blit(alt3, (430, 370))
screen.blit(alt4, (610, 370))


wordlist = ['Active', 'Placid', 'Sleepy','Jittery','
   Energetic','Intense','Calm','Tired','Vigorous','At-rest
   ','Drowsy','Fearful','Lively','Still','Wide-awake','
   Clutched-up','Quiet','Full-of-pep','Tense','Wakeful']
```

```
def words(counter):
        return wordlist[counter]

def drawWord(counter):
        pygame.draw.rect(screen,black,(0,0,800,200),0)
        word = myfont.render(wordlist[counter], 1, white)
        screen.blit(word, (320, 100))

count=0
for count in xrange(4): pygame.draw.rect(screen, white,
    (100+count*(133.33+50),300,50,50),2);count+=1


posx = 466.66
posy = 300

pygame.draw.line(screen,white,(posx,posy),(posx+50,posy+50)
    ,3)
pygame.draw.line(screen,white,(posx+50,posy),(posx,posy+50)
    ,3)

def move():
        pygame.draw.rect(screen,black,(lastposx+2,lastposy
            +2,47,47),0)
        pygame.draw.line(screen,white,(posx,posy),(posx+50,
            posy+50),3)
        pygame.draw.line(screen,white,(posx+50,posy),(posx,
            posy+50),3)

def erase():
        pygame.draw.rect(screen,black,(100+2,300+2,47,47)
            ,0)
        pygame.draw.rect(screen,black
            ,(283.33+2,300+2,47,47),0)
        pygame.draw.rect(screen,black
            ,(466.66+2,300+2,47,47),0)
        pygame.draw.rect(screen,black
            ,(649.99+2,300+2,47,47),0)

while wordcount<20:
        lastposx=posx
        lastposy=posy
        lastpos=pos
        keys_pressed = get_pygame_events()
```

```
for event in keys_pressed:
        if event.type == pygame.KEYDOWN:
                if event.key == K_a:
                        posx-=(133.33+50)
                        pos-=1
                if event.key == K_d:
                        posx+=133.33+50
                        pos+=1
                if event.key == K_SPACE:
                        result = pos
                        if wordcount == 14 or
                           wordcount == 19:
                                if result == 4:
                                        result = 1
                                elif result == 3:
                                        result = 2
                                elif result == 2:
                                        result = 3
                                elif result == 1:
                                        result = 4
                        results.append(result)
                        print(words(wordcount)+' '+
                           str(result))
                        if wordcount == 19:
                                A1 = results[0]+
                                   results[4]+
                                   results[8]+
                                   results[12]+
                                   results[17]
                                A2 = results[2]+
                                   results[7]+
                                   results[10]+
                                   results[14]+
                                   results[19]
                                B1 = results[3]+
                                   results[5]+
                                   results[11]+
                                   results[15]+
                                   results[18]
                                B2 = results[1]+
                                   results[6]+
                                   results[9]+
                                   results[13]+
                                   results[16]
                                print('A1='+str(A1)
                                                    )
                                                print('A2='+str(A2)
                                                    )
                                                print('B1='+str(B1)
                                                    )
                                                print('B2='+str(B2)
                                                    )
                                        wordcount+=1
                                        erase()
                                        pos = 3
                                        posx=466.66
                                        move()
                if posx<99 or posx>=700:
                        posx = lastposx
                        pos=lastpos

                move()
                if wordcount < 20:
                        drawWord(wordcount)
                pygame.display.update()
```

Appendix E - Arduino Schematics

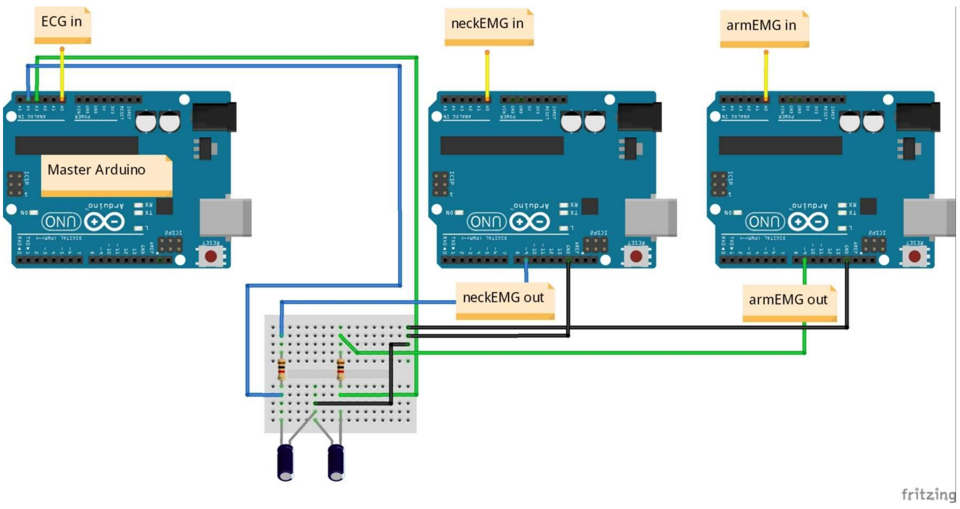


Figure 6.1: Schematics of biometric set up

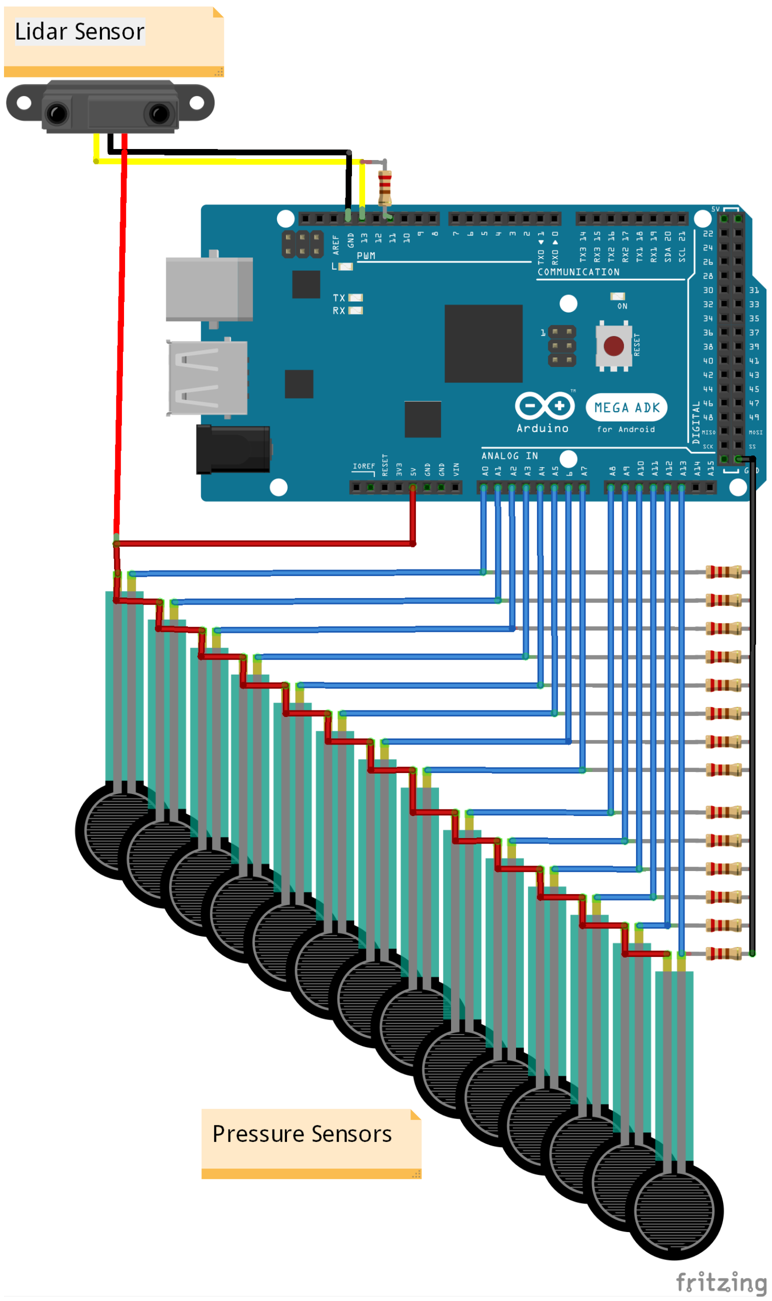


Figure 6.2: Schematics of chair set up

This page is intentionally left blank.

Appendix F - Arduino Code

Arduino code for biometric sensors

```
eHealth sensor platform for Arduino and Raspberry from Cooking-hacks.

Description:  "The e-Health Sensor Shield allows Arduino and
Raspberry Pi users to perform biometric and medical applications
by using 9 different sensors:  Pulse and Oxygen in Blood Sensor
(SPO2), Airflow Sensor (Breathing), Body Temperature, Electrocardiogram
Sensor (ECG), Glucometer, Galvanic Skin Response Sensor (GSR
- Sweating), Blood Pressure (Sphygmomanometer) and Patient Position
(Accelerometer)."

In this example we read the values in volts of ECG sensor and
show these values in the serial monitor.

Copyright (C) 2012 Libelium Comunicaciones Distribuidas S.L.
http://www.libelium.com

This program is free software:  you can redistribute it and/or
modify it under the terms of the GNU General Public License as
published by the Free Software Foundation, either version 3 of
the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.

Version 2.0
Author:  Luis Martin & Ahmad Saad Modified by Helge Soltvedt
Garsmark


#include <eHealth.h>

byte serialByte;
float ECG;
int EMGneck;
int EMGarm;
int i = 0;
//double distance;
// The setup routine runs once when you press reset:
void setup() {
  Serial.begin(115200);
  pinMode(A1,INPUT);
  pinMode(A4,INPUT);
}

// The loop routine runs over and over again forever:
void loop() {

  while (Serial.available()>0){
    serialByte=Serial.read();
    if (serialByte=='C'){
      Serial.println("Time , ECG , EMGneck , EMGarm");
      while(1){
        EMGneck = analogRead(A4);
        EMGarm = analogRead(A1);
        ECG = eHealth.getECG();
        //distance = analogRead(A1);


        //Serial.print("ECG value :  ");
        Serial.print(millis());
        Serial.print(";");
        Serial.print(ECG,5);
        Serial.print(";");
        Serial.print(EMGneck);
        Serial.print(";");
        Serial.println(EMGarm);
        //Serial.print(" V");
        //Serial.println("");
        delay(1);  // wait for a millisecond
        if (Serial.available()>0){
          serialByte=Serial.read();
          if (serialByte=='r')   {Serial.print("Situation
              ");
          Serial.print(i); Serial.println(" end"); delay
              (50);}
          else if (serialByte=='t')  { i += 1;
          Serial.print("Situation "); Serial.print(i);
          Serial.println(" start"); delay(50); }
          else if (serialByte=='F')  break;
        }
      }
    }
  }
}
```

Arduino code for chair sensors

```
    //#include <CapacitiveSensor.h>

byte serialByte;
int i = 0;
long previousMillis = 0;
int numSensors = 14;
#define LtrigPin 12
#define LmonPin 11
double distance;
//long rightarm;
//long leftarm;

//CapacitiveSensor   cs_4_2 =
CapacitiveSensor(4,2); // 2 is sensor pin
//CapacitiveSensor   cs_4_6 =
CapacitiveSensor(4,6); //6 is sensor pin



void setup() {
  pinMode(LtrigPin, OUTPUT);
  digitalWrite(LtrigPin, LOW);
  pinMode(LmonPin, INPUT);

  Serial.begin(115200);
}


void loop() {

while (Serial.available()>0){
  serialByte=Serial.read();
  if (serialByte=='C'){
     while(1){
         if (millis()-previousMillis>199){
              String dataString = "";
              Serial.print(millis());
              Serial.print(";");
              //Pressure 1-16
              // read three sensors and append to the
                 string:
              for (int analogPin = 0; analogPin <
                 numSensors;
              analogPin++) {
                int sensor = analogRead(analogPin);
                Serial.print(sensor);
                Serial.print(";");
            }

            //rightarm =  cs_4_2.capacitiveSensor(60);
            //leftarm =  cs_4_6.capacitiveSensor(60);
            //Serial.print(rightarm);
            //Serial.print(";");
            //Serial.print(leftarm);
            //Serial.print(";");

            distance = pulseIn(LmonPin, HIGH);
            distance = distance/10;
            Serial.print(distance);
            Serial.println(";");

            previousMillis = millis();
            if (Serial.available()>0){
                serialByte=Serial.read();
                if (serialByte=='r') {
                  Serial.print("Situation ");
                  Serial.print(i);
                  Serial.println(" end");
                  delay(50);
                  }
                else if (serialByte=='t') {
                  i += 1;
                  Serial.print("Situation ");
                  Serial.print(i);
                  Serial.println(" start");
                  delay(50);
                  }
                else if (serialByte=='F') {
                  break;
                  }
            }
         }
     }
   }
 }
}
```

This page is intentionally left blank.

Appendix G - Complete Dataset

Participant	Gender	Education	Age	Coffee	0Grid_x	0Grid_y	1EMGn_M	1EMGn_st	1EMGn_Va	1EMGn_M	1EMGn_(S	1EMGa_M
6	0	0	25	2	7	3	175.4591	3.083753	9.509535	175		207.8262
7	0	2	29	0	6	6	224.1257	2.629375	6.913612	224		229.1308
8	0	0	25	2	7	4	195.5588	2.781894	7.738934	195		196.8901
9	0	0	25	2	5	5	176.1089	2.284992	5.22119	176		191.9126
10	0	0	26	3	7	4	173.0589	2.34527	5.500289	173		190.7914
11	1	0	24	2	5	3	183.2389	2.714734	7.369779	183		228.4236
12	0	0	25	2	6	6	173.7692	2.361205	5.575288	174		212.9603
13	0	0	25	0	4	6	207.8031	2.586786	6.691461	208		203.7219
14	0	0	25	2	7	3	214.6144	2.989564	8.937492	214		215.4242
15	0	0	24	1	4	3	181.9698	2.805458	7.870597	182		211.9988
16	0	1	27	3	7	4	177.3393	2.936033	8.620287	177		204.7099
17	0	1	27	3	8	3	168.1937	2.720918	7.403397	168		198.9112
18	0	0	25	2	7	4	201.1221	2.865259	8.209708	201		213.7427
19	0	0	26	1	7	5	181.1925	2.609722	6.810648	181		194.466
20	0	0	25	2	5	5	167.8087	2.456662	6.03519	168		195.9458
21	0	0	25	2	7	4	176.979	3.639972	13.2494	177		212.3416
22	0	0	26	2	6	3	224.336	2.390642	5.715168	224		248.5631
23	0	0	27	2	6	5	224.2277	2.418467	5.848983	224		248.9014
24	1	0	26	1	7	4	213.9019	3.346579	11.19959	214		188.8192
25	0	0	26	3	5	5	172.9742	3.740395	13.99055	172		194.5568
26	0	0	26	2	4	2	183.5341	3.009155	9.055013	183		191.1929
27	0	1	25	1	5	5	576.3101	16.42362	269.7353	577		142.6393
28	0	2	26	0	7	7	189.5632	2.775958	7.705943	189		202.4218
29	1	0	24	1	6	4	202.6318	4.878639	23.80112	202		200.6659
30	0	2	25	3	6	6	176.1946	2.662227	7.087454	176		202.8263
31	0	0	28	2	7	3	185.3633	2.776762	7.710407	185		207.0207
32	1	0	25	1	3	4	179.3005	2.575687	6.634162	179		207.194
33	1	0	24	2	6	6	188.7848	2.666371	7.109534	189		203.1843
34	1	0	26	1	4	6	187.7064	3.312588	10.97324	187		193.2163
35	0	1	26	0	6	7	176.8357	2.484319	6.171841	177		190.7498
36	0	0	26	3	3	7	171.2247	2.456777	6.035754	171		190.293
37	0	0	24	2	6	4	209.3488	2.478581	6.143366	209		197.9149
38	0	0	24	2	6	7	174.2648	2.470446	6.103105	174		191.0804
39	1	0	25	0	5	7	195.2655	2.765131	7.645948	195		186.5437

1EMGa_st	1EMGa_Va	1EMGa_M	1EMGa_(S	1ECG_Mea	1ECG_std	1ECG_Var	1ECG_Med	1ECG_RR n	1ECG_RR s	1ECG_HR n	1ECG_HR s	1ECG_RMS
2.347485	5.510688	208		1.80147	0.211122	0.044572	1.77908	874	120	70	10	160
2.391404	5.718814	229		1.786415	0.23473	0.055098	1.77908	754	50	80	5.3	34
2.275372	5.177318	197		1.821456	0.245093	0.060071	1.78397	788	36	76	4	30
2.247401	5.050813	192		1.803884	0.468902	0.219869	1.73998	742	36	81	5.9	34
2.551987	6.512638	191		1.816996	0.22911	0.052491	1.78397	695	53	87	6.7	40
2.488064	6.19046	228		1.817652	0.244115	0.059592	1.78397	640	27	94	502	11
2.693976	7.257508	213		1.807458	0.308152	0.094958	1.73021	739	26	82	6	20
3.326508	11.06566	203		1.839347	0.441141	0.194605	1.73998	715	22	84	2.9	19
3.103935	9.634411	215		1.823429	0.416377	0.17337	1.73998	696	29	87	9.3	22
3.023188	9.139663	212		1.757977	0.289309	0.083699	1.71065	753	44	80	7	29
4.644112	21.56778	205		1.836252	0.418971	0.175537	1.74976	560	25	110	6.8	16
2.498066	6.240334	199		1.813893	0.327482	0.107244	1.78397	762	34	81	14	37
2.371349	5.623294	214		1.824747	0.29065	0.084477	1.80841	749	38	81	9.6	34
2.783387	7.747243	194		1.817534	0.378757	0.143457	1.78397	1030	76	58	5.4	77
2.301034	5.29476	196		1.820135	0.464033	0.215326	1.75464	804	39	75	4.6	34
2.476978	6.13542	212		1.763002	0.55625	0.309414	1.78397	861	70	70	10	110
2.408668	5.80168	248		1.807736	0.228454	0.052191	1.78886	1100	65	55	5.5	84
3.622137	13.11988	248		1.816461	0.159643	0.025486	1.80841	990	71	61	5.3	61
3.097781	9.596248	188		1.824387	0.469722	0.220639	1.79374	650	16	92	2.7	12
3.331205	11.09693	194		4.234101	1.367629	1.870409	5					
3.290286	10.82598	191		1.818601	0.275295	0.075787	1.77419	764	72	80	10	95
10.66548	113.7525	143		1.829401	0.692654	0.479769	1.78886					
3.241692	10.50856	202		1.824931	0.174594	0.030483	1.80352	891	48	68	4.5	43
3.3419	11.1683	200		1.793475	0.17916	0.032098	1.77419	717	41	84	6.2	29
3.672974	13.49074	202		1.80389	0.167314	0.027994	1.78397	1050	56	58	3.6	62
3.655623	13.36358	206		1.828638	0.208584	0.043507	1.80841	827	51	73	7.2	31
3.36703	11.33689	207		1.807244	0.256201	0.065639	1.78886	656	23	92	9	15
2.683179	7.199451	203		1.818731	0.147686	0.021811	1.79863	728	18	83	3.6	16
3.026065	9.157067	193		1.824029	0.132264	0.017494	1.80841	681	24	88	4.1	17
2.50378	6.268915	190		1.823205	0.235575	0.055496	1.80352	827	46	73	5.1	41
2.467592	6.089011	190		1.820615	0.174424	0.030424	1.76931	675	45	90	7.6	29
2.625421	6.892837	198		1.815647	0.306089	0.09369	1.77419	559	25	110	7.4	19
2.553836	6.522079	191		1.808746	0.205774	0.042343	1.78886	868	42	69	5.2	49
2.920707	8.530527	186		1.803828	0.178491	0.031859	1.76442	632	39	96	9.6	34

1ECG_NN5	1ECG_PNN	1ECG_RR t	1ECG_TINN	1ECG_VLF	1ECG_LF p	1ECG_HF p	1ECG_VLF	1ECG_LF	1ECG_HF	1ECG_LF n	1ECG_HF n	1ECG_LF/H
192	57	22.6	364.3	260	1400	2670	6.1	32	62	32.5	69.9	0.52
32	8.2	10.6	156.3	240	1100	242	15	69	16	79.2	18.1	4.4
31	8.3	8.7	126.5	42	590	257	4.7	67	29	68.7	29.6	2.3
52	14	10.3	150.7	57	560	168	7.3	71	21	72.9	22	3.3
61	14	9.9	154.3	270	950	153	20	69	11	79.6	12.9	6.2
0	0	7.2	101.9	51	300	28	13	79	7.3	90.9	8.41	11
5	1.3	6.7	102.8	36	210	77.8	11	64	24	70.2	26.3	2.7
3	0.72	5.6	84.8	13	98	81	6.7	51	42	51.5	42.8	1.2
13	3	7.6	110.5	50	310	82.8	11	70	19	75.7	20.1	3.8
29	7.4	11.5	171.3	140	1000	222	9.8	74	16	81.1	17.4	4.7
6	1.1	5.9	87.1	19	190	112	5.9	59	35	61.2	36.5	1.7
71	18	8.1	124.5	28	290	141	6.1	63	31	51.9	25.5	2
54	14	9.6	140.7	70	410	332	8.6	51	41	53.5	43	1.2
138	47	16.2	241.8	600	1500	744	21	53	26	63.5	30.8	2.1
41	11	9.4	145	170	680	332	15	58	28	64.9	31.5	2.1
42	12	8.6	124.6	42	390	408	5	46	49	34.4	36	0.95
147	55	14.9	235.3	53	880	804	3	51	46	48.7	44.3	1.1
101	34	13.3	192.7	700	1900	404	23	63	13	81.6	17.2	4.8
0	0	4	62.3	14	95	20.3	11	74	16	77.7	16.6	4.7
82	21	11.8	164.9	100	770	525	7.5	55	38	50	34.1	1.5
80	24	12.7	189.1	170	650	299	15	58	27	66.7	30.6	2.2
32	7.8	10	156.9	110	430	174	16	60	24	69.5	28.4	2.5
134	48	14.1	222.6	77	620	651	5.7	46	48	46.5	48.7	0.96
37	10	11.1	184.9	320	760	119	27	64	9.9	85.3	13.3	6.4
2	0.44	6.2	84.1	89	190	49.5	27	57	15	75.8	20.2	3.7
0	0	5.1	77.3	27	82	11.1	22	69	9.2	67.9	9.15	7.4
2	0.46	6.4	97.9	56	220	49.2	17	68	15	76.3	16.7	4.6
77	22	11.5	185.4	120	770	328	10	63	27	68.2	29.1	2.3
36	8.2	13.3	208.7	100	810	212	9	72	19	77.4	20.3	3.8
17	3.2	6.8	94.2	31	160	139	9.6	48	42	50.4	44.8	1.1
117	34	10.6	169.3	98	170	380	15	26	59	28.1	63.4	0.44
68	15	10.2	155.5	180	250	413	21	30	49	36	58.5	0.61

1ECG_(Seri	1Lean_mea	1Lean_std.	1Lean_Var	1Lean_Median	1Lean_(Ser	1Chair_lea	1Chair_std	1Chair_Var	1Chair_me	1Chair_NO	1Chair_(Se	1Grid_x
	18.79173	2.394676	5.734471	18.9		4				1		8
	25.90531	2.269929	5.152579	26		4				1		4
	21.38431	1.666396	2.776875	21.5		2.993179				8		8
	14.74577	1.994813	3.979277	14.7		3				1		6
	17.25041	1.689469	2.854306	17.3		2.997948				56		8
	15.05404	1.846756	3.410508	15		3.000679				99		7
	14.98406	2.052178	4.211435	15		2.634063				27		8
	14.50918	2.166642	4.694338	14.6		1.99865				1		7
	16.90221	1.635644	2.675333	16.9		3.880936				15		8
	23.97316	2.509822	6.299204	24		2.010246				13		8
	17.06435	4.187832	17.53794	17		2.902473				11		3
	29.05619	3.114665	9.701136	29.1		3				3		7
	17.3739	3.592463	12.90579	17.4		3				1		7
	20.62162	2.020648	4.08302	20.6		4				1		7
	21.12114	2.537052	6.436633	21.1		2.00684				5		8
	15.55962	1.713085	2.934659	15.6		4				1		6
	30.09275	2.141719	4.58696	30		2.369788				7		7
	18.6201	3.247329	10.54515	18.5		2.734133				3		7
	19.0269	1.936453	3.749851	19		2				1		9
	20.52882	1.989331	3.957438	20.6		2.000684				1		8
	23.26516	2.020838	4.083786	23.4		3.989071				45		8
	18.89667	2.133429	4.55152	18.8		2.998639				1		7
	10.8638	2.030105	4.121327	10.9		4				1		7
	15.37988	2.662201	7.087316	15.4		3.57974				25		8
	23.17089	2.724746	7.424242	23.1		2.99863				1		7
	14.93712	2.567247	6.590755	14.9		4				1		7
	25.37173	2.371816	5.62551	25.3		2.605938				3		7
	13.11516	2.25058	5.065112	13.1		3.97541				3		6
	19.4663	2.898746	8.402727	19.5		1.859589				1		6
	22.03762	3.401109	11.56755	22.1		1.327633				41		6
	21.2111	2.878452	8.285483	21.2		2.993146				8		6
	16.58077	2.058476	4.237323	16.5		2.560575				92		7
	24.52909	2.179095	4.748456	24.6		2.009582				7		7
	15.80842	2.521446	6.357689	15.7		3				1		7

1Grid_y	1PANAS_P	1PANAS_N	1PANAS_(S	1ADACL_A	1ADACL_A	1ADACL_B	1ADACL_B	1ADACL_(S	2EMGn_M	2EMGn_st	2EMGn_Va	2EMGn_M
4	35	16		13	9	10	12		178.0733	3.735474	13.95377	178
4	26	26		9	7	12	14		232.7258	2.463123	6.066975	233
4	27	12		9	15	7	18		189.2702	2.984219	8.90556	189
4	26	15		9	15	9	12		178.4187	2.440922	5.958102	178
5	30	17		11	11	7	15		173.5731	2.563328	6.570652	173
3	31	19		12	6	11	11		179.9798	2.825535	7.983651	180
8	23	10		8	12	5	15		174.9109	4.479319	20.0643	174
3	25	12		7	16	12	14		206.4244	2.376984	5.650053	206
6	38	14		14	13	9	18		217.195	3.1062	9.648479	217
6	42	23		19	12	15	16		182.5212	4.257125	18.12311	182
3	36	17		14	6	12	11		178.8838	2.659987	7.075532	179
3	42	15		16	6	10	13		168.3822	2.415705	5.83563	168
2	43	18		14	7	11	12		200.0448	2.945469	8.675787	200
7	23	10		9	12	5	18		177.8548	2.542588	6.464753	178
5	30	16		10	8	8	17		169.0196	2.310644	5.339077	169
3	33	23		15	6	17	9		177.8646	2.756166	7.596453	178
4	38	16		12	7	9	15		224.243	2.403534	5.776975	224
5	33	16		12	12	12	19		224.0724	2.492829	6.214198	224
6	33	14		12	9	10	14		212.2108	4.295935	18.45506	212
2	44	15		16	6	14	11		171.1312	2.333968	5.447405	171
6	41	23		17	7	11	16		183.2451	2.477703	6.139013	183
3	41	18		17	5	18	15		616.4878	9.773267	95.51674	617
5	23	11		8	15	5	17		190.8471	2.747619	7.54941	191
5	35	18		12	10	15	12		208.4328	3.576342	12.79022	208
6	18	11		5	15	6	15		177.3061	6.234632	38.87063	177
4	28	18		10	14	6	17		188.4606	3.135319	9.830228	188
4	29	23		12	8	12	13		182.2818	2.853068	8.139997	182
5	26	11		12	14	10	16		187.2192	2.68832	7.227063	187
6	34	11		14	13	6	16		187.7403	4.73654	22.43482	187
3	36	37		12	18	8	13		177.2177	2.412619	5.820728	177
4	32	17		14	8	12	10		175.5179	3.250396	10.56508	175
4	26	13		13	7	7	16		209.1824	2.598241	6.750856	209
4	30	12		14	15	7	17		173.3419	2.438028	5.943979	173
6	19	17		8	19	6	19		194.5121	2.709343	7.340538	194

2EMGn_(S	2EMGa_M	2EMGa_st	2EMGa_Va	2EMGa_M	2EMGa_(S	2ECG_Mea	2ECG_std	2ECG_Var	2ECG_Med	2ECG_RR n	2ECG_RR s	2ECG_HR n
	200.74	3.607037	13.01072	200		1.800811	0.173905	0.030243	1.77908	883	95	69
	214.3649	34.69471	1203.723	189		1.778239	0.344744	0.118849	1.79374	730	51	83
	199.0855	2.514406	6.322237	199		1.818187	0.309935	0.09606	1.77908	749	40	81
	195.3312	2.447294	5.989247	195		1.802019	0.456594	0.208478	1.74487	789	33	77
	185.9764	9.743795	94.94153	183		1.807828	0.212424	0.045124	1.77908	683	61	89
	222.7249	2.347721	5.511793	223		1.81813	0.227263	0.051648	1.77908	642	27	94
	213.1185	2.640751	6.973564	213		1.815167	0.319359	0.10199	1.73998	744	77	83
	203.1776	3.734047	13.94311	203		1.820826	0.45735	0.209169	1.73021	701	23	86
	218.3271	3.07125	9.432578	218		1.814919	0.423949	0.179733	1.73021	676	24	89
	211.7536	7.74396	59.96892	211		1.851585	0.307383	0.094485	1.80352	669	51	91
	207.5446	4.9601	24.60259	207		1.817896	0.319042	0.101788	1.77419	692	47	88
	202.8386	3.723001	13.86074	203		1.817237	0.287376	0.082585	1.78886	873	43	69
	218.3734	2.919529	8.523648	218		1.804388	0.435792	0.189915	1.82307	732	47	83
	198.6072	2.888291	8.342227	198		1.81382	0.394422	0.155569	1.76931	952	64	64
	199.2256	2.323796	5.400026	199		1.814079	0.45576	0.207717	1.74487	806	37	75
	214.0188	2.348353	5.514761	214		1.823188	0.420876	0.177137	1.79374	852	37	71
	247.8886	2.380136	5.665047	248		1.802162	0.257462	0.066286	1.78397	908	140	69
	248.2429	3.703666	13.71714	247		1.868854	0.609425	0.371398	1.79863	1000	160	65
	186.8809	3.005263	9.031605	186		1.812868	0.480893	0.231258	1.79863	653	21	92
	192.923	3.299519	10.88683	192		4.994429	0.01	0.0001	5			
	191.9274	3.145649	9.895109	191		1.817585	0.285574	0.081552	1.76931	647	120	97
	168.2691	6.6873	44.71998	168		1.823753	0.29193	0.085223	1.77908	903	48	67
	204.7113	3.764773	14.17351	204		1.816188	0.1857	0.034484	1.79863	825	34	73
	200.6236	3.321386	11.03161	200		1.83963	0.180119	0.032443	1.80841	783	43	77
	201.1748	3.634207	13.20746	201		1.803924	0.188938	0.035697	1.78886	963	50	63
	210.1144	9.823168	96.49462	207		1.823876	0.231025	0.053373	1.79374	653	29	94
	209.8332	4.622267	21.36535	209		1.812651	0.245068	0.060059	1.80352	732	38	83
	205.1535	3.155795	9.959042	205		1.781258	0.573391	0.328777	1.77908			
	194.8965	6.075883	36.91635	194		1.826053	0.163033	0.02658	1.82796	695	54	87
	189.5815	2.549925	6.502119	189		1.820267	0.242181	0.058652	1.79863	826	39	73
	189.2421	2.508697	6.29356	189		1.818366	0.165727	0.027465	1.77908	748	72	82
	201.2246	2.677877	7.171027	201		1.818414	0.314719	0.099048	1.77908	582	25	100
	189.0726	2.432685	5.917958	189		1.799065	0.572871	0.328181	1.80352			
	184.0355	2.726807	7.435476	184		1.808256	0.198336	0.039337	1.77908	636	37	95

2ECG_HR s	2ECG_RMS	2ECG_NN5	2ECG_PNN	2ECG_RR t	2ECG_TINN	2ECG_VLF	2ECG_LF p	2ECG_HF p	2ECG_VLF	2ECG_LF	2ECG_HF	2ECG_LF n
8.9	91	148	45	22	374.8	200	2000	1350	6.1	62	32	60.7
6	38	42	10	10.3	147.1	270	920	133	20	70	10	83.6
6.2	28	23	6.1	9.9	144.6	290	820	187	23	63	14	79.5
9.3	33	45	12	8.8	134.4	38	180	190	9.4	44	47	44.7
12	52	48	11	10.6	156.2	490	450	251	41	38	21	57
6.1	24	2	0.43	6	93.6	32	190	20.3	13	78	8.5	89.1
15	120	34	8.5	7.5	87.1	100	880	912	5.3	46	48	23.8
3.6	18	4	0.92	5.6	84.4	42	170	109	13	53	34	58
5.6	19	6	1.3	5.7	85.1	61	190	45.1	21	64	15	76.4
15	43	25	5.6	10.8	165.1	210	580	410	17	48	34	43.8
10	33	37	8.6	9.3	127.6	79	660	332	7.4	62	31	65.6
7	45	91	26	9.4	142.5	66	390	257	9.3	55	36	51.6
10	43	53	13	11.5	178.6	140	380	250	18	49	33	57.1
9.8	61	120	39	13.5	229.5	240	1300	641	11	60	30	63.5
5.2	36	75	20	9.6	153.2	160	340	223	22	47	31	56
3.7	32	38	11	9.3	139.3	110	330	103	20	61	19	67.9
15	86	151	46	23.3	410.1	5200	1600	587	70	22	8	61.3
26	200	147	50	20.9	309.3	3100	1700	1260	52	27	21	45.2
3.6	12	1	0.22	6.2	94.6	13	120	20.2	8.4	79	13	82.7
21	190	266	58	11.7	58.7	440	970	5170	6.7	15	79	9.96
4.9	41	42	13	12.7	198	210	740	128	20	69	12	83.5
7	30	35	9.7	9.4	147.4	120	420	151	17	61	22	70.9
11	42	28	7.3	10.1	149.2	120	290	158	21	51	28	61.1
8	48	89	29	11.8	183.3	110	500	409	11	49	40	52.8
14	17	7	1.5	7.2	101.4	29	310	73.2	7	75	18	79.2
9.6	30	12	2.9	7.4	109.3	28	260	128	6.6	63	31	62.2
9.1	29	30	7	13.9	215.6	470	1400	157	23	70	7.6	88.2
3.9	42	81	23	10.8	166.6	87	410	363	10	48	42	50.7
15	58	72	18	15.5	244.9	520	630	319	35	43	22	60.4
7.1	17	8	1.6	7.8	113.2	33	130	88.3	13	52	35	57.5
9.8	30	48	10	10.3	162.2	53	360	304	7.4	50	43	50.9

2ECG_HF n	2ECG_LF/H	2ECG_(Seri	2Lean_mea	2Lean_std.	2Lean_Var	2Lean_Me	2Lean_(Ser	2Chair_lea	2Chair_std	2Chair_Var	2Chair_me	2Chair_NO
31.5	1.9		21.06198	9.392937	88.22726	20.3		3.997245				3
12.2	6.9		19.60483	2.370361	5.618609	19.4		4				1
18.2	4.4		23.2114	1.956161	3.826566	23.1		3.866212				62
47.6	0.94		14.81842	2.144246	4.597791	14.8		3				1
31.5	1.8		17.94457	2.319273	5.379025	17.8		2.960355				17
9.73	9.2		14.16441	1.759214	3.094833	14.1		3.969945				36
24.6	0.96		15.66623	2.405884	5.788279	15.7		2.59877				43
37.4	1.6		13.3071	2.157305	4.653966	13.3		3.290984				96
18.1	4.2		14.31914	1.728425	2.987453	14.2		2.90499				43
30.9	1.4		24.84593	4.00075	16.006	24.6		2.262474				100
32.8	2		18.74856	4.19624	17.60843	18.7		2.754121				20
33.5	1.5		29.01701	3.137886	9.846329	29.15		2.999311				3
37.9	1.5		14.27031	3.968775	15.75117	14.2		3.755326				18
31.6	2		19.42315	2.31372	5.3533	19.4		2.131507				113
36.7	1.5		18.84177	2.489284	6.196534	18.8		2.010974				112
21.4	3.2		13.08584	2.006443	4.025812	12.95		1.943228				41
22.2	2.8		27.80389	2.665821	7.106604	28		2.368852				87
34.1	1.3		15.70549	19.07123	363.7118	13.1		2.921125				9
13.8	6		19.76277	1.973624	3.895191	19.7		2.066393				57
			19.54577	2.174871	4.730064	19.5		2.002732				3
53	0.19		20.06671	2.156788	4.651735	20.3		3.995899				38
14.4	5.8		17.86961	2.177209	4.74024	17.9		3.085558				51
25.4	2.8		11.91128	2.338972	5.470789	11.8		4				1
33.8	1.8		6.500616	1.730792	2.995643	6.5		3.705882				56
43.4	1.2		20.82726	2.525225	6.376764	20.8		3				1
19	4.2		13.74403	3.64097	13.25666	13.5		3.877816				95
30.3	2.1		22.3376	2.5281	6.391292	22.4		2.441781				12
			13.3286	3.099106	9.604458	13.4		2.28942				49
9.62	9.2		26.87529	5.044371	25.44568	27.1		2.661191				150
45.1	1.1		22.07447	3.011143	9.06698	22		2				1
30.4	2		19.90144	3.200959	10.24614	19.8		2.843943				61
38.7	1.5		13.62005	2.102839	4.421932	13.7		2.125257				36
			24.24668	2.279499	5.196115	24.3		2				186
43.4	1.2		17.75299	2.459736	6.050302	17.8		2.934111				178

2Chair_(Se	2Grid_x	2Grid_y	2PANAS_P	2PANAS_N	2PANAS_(S	2ADACL_A	2ADACL_A	2ADACL_B	2ADACL_B	2ADACL_(S	3EMGn_M	3EMGn_st
	5	1	34	34		15	5	17	7		195.2706	8.492629
	6	2	35	24		16	7	12	7		227.0746	2.375027
	3	1	41	29		20	5	19	5		185.0379	2.915785
	4	3	25	25		11	8	12	7		178.7595	2.369681
	3	3	26	26		13	13	13	8		178.9797	2.403896
	2	4	29	32		12	7	13	7		180.7468	4.031017
	4	2	37	17		14	7	11	5		173.9458	2.403479
	3	2	35	20		15	5	13	5		206.0375	2.363066
	3	3	25	28		15	7	15	7		218.1449	3.19131
	5	2	37	37		16	5	18	5		186.3913	3.985794
	1	2	27	29		15	5	17	7		175.124	2.38272
	3	3	26	26		15	6	15	6		167.7196	2.399986
	4	3	41	17		16	12	14	8		199.6938	2.545065
	6	3	33	10		14	13	6	14		179.2725	2.463371
	2	4	18	36		11	6	17	9		171.9926	2.783399
	5	1	33	35		17	5	17	6		178.157	2.440382
	4	3	28	23		13	6	15	7		224.0507	2.366809
	2	5	23	27		8	10	17	6		224.0448	2.406682
	2	3	32	28		14	6	17	6		212.926	3.392408
	2	1	19	18		12	6	13	10		171.8427	2.404402
	3	1	37	31		13	9	15	7		184.7937	2.49719
	4	1	48	31		19	6	18	5		667.935	9.810385
	2	3	33	25		15	11	17	7		190.0054	2.886415
	3	1	28	36		12	7	18	7		215.8209	3.426717
	3	3	26	29		10	7	15	5		176.0158	2.441747
	5	2	33	25		16	6	13	7		187.6794	3.115237
	8	2	35	31		17	6	15	8		182.0833	3.087749
	3	3	24	26		16	9	14	9		179.7302	10.19142
	3	2	32	16		15	5	12	5		184.8487	2.758289
	2	6	34	28		14	16	15	11		177.6022	2.379929
	3	3	21	26		12	10	17	5		174.1984	2.398496
	3	3	23	21		10	7	14	6		208.953	2.521688
	4	1	33	20		16	6	13	11		173.4142	2.38908
	4	4	33	27		16	13	16	13		196.02	2.757283

3EMGn_Va	3EMGn_M	3EMGn_(S	3EMGa_M	3EMGa_std	3EMGa_Va	3EMGa_M	3EMGa_(S	3ECG_Mea	3ECG_std	3ECG_Var	3ECG_Med	3ECG_RR m
72.12474	197		194.4515	3.247338	10.5452	194		1.801947	0.187734	0.035244	1.77908	923
5.640752	227		237.6041	2.498106	6.240532	237		1.80173	0.237569	0.056439	1.77908	719
8.501804	185		196.9247	2.296518	5.273997	197		1.805272	0.248746	0.061875	1.76931	825
5.615389	179		192.5975	2.229417	4.970299	192		1.803575	0.463038	0.214404	1.73998	752
5.778717	179		233.8733	2.973344	8.840775	234		1.809227	0.20492	0.041992	1.77908	753
16.24909	180		221.9962	2.38476	5.687079	222		1.814307	0.223374	0.049896	1.77419	677
5.77671	174		214.8659	2.614158	6.833823	215		1.814358	0.322324	0.103893	1.74487	800
5.584081	206		205.2285	2.642002	6.980176	205		1.822765	0.451249	0.203626	1.72532	729
10.18446	218		223.7797	3.323112	11.04307	223		1.813143	0.425533	0.181079	1.73021	702
15.88655	186		212.0122	4.383452	19.21465	211		1.828805	0.298736	0.089243	1.76931	641
5.677354	175		204.4544	2.806603	7.877021	204		1.814467	0.312309	0.097537	1.76931	651
5.759931	168		200.3423	4.60357	21.19286	199		1.81783	0.295549	0.087349	1.78397	913
6.477357	200		218.1087	2.543337	6.468563	218		1.827521	0.283369	0.080298	1.80352	770
6.068195	179		197.1227	2.716175	7.377606	197		1.815025	0.393523	0.15486	1.77419	972
7.747308	172		198.0022	2.315492	5.361505	198		1.810932	0.463704	0.215022	1.74487	815
5.955467	178		213.4894	2.422185	5.866978	213		1.815989	0.400663	0.160531	1.77908	842
5.601785	224		245.7692	2.333843	5.446821	246		1.819609	0.224622	0.050455	1.81329	1080
5.79212	224		247.9817	3.623975	13.1332	247		1.811256	0.135496	0.018359	1.80841	1000
11.50843	213		188.116	3.134419	9.824584	187		1.813932	0.458951	0.210636	1.77908	643
5.78115	172		225.056	3.786023	14.33397	225		4.627688	1.147998	1.317899	5	
6.235958	185		192.0769	3.157457	9.969534	191		1.81528	0.275667	0.075992	1.75953	740
96.24366	668		203.3563	6.589503	43.42155	203		1.816421	0.276871	0.076658	1.77419	888
8.331389	190		201.2553	3.312831	10.97485	200		1.821177	0.184589	0.034073	1.80841	828
11.74239	216		201.6696	3.297754	10.87518	201		1.827342	0.169988	0.028896	1.79863	727
5.962128	176		202.6757	3.328411	11.07832	202		1.80319	0.175423	0.030773	1.78397	939
9.704705	187		204.562	3.757526	14.119	204		1.811419	0.213381	0.045531	1.78397	738
9.534193	182		209.2887	4.175677	17.43628	208		1.814898	0.231272	0.053487	1.79863	752
103.865	178		203.5511	2.657569	7.062674	203		1.826843	0.160831	0.025867	1.80352	725
7.608158	185		194.2967	4.121494	16.98671	194		1.829065	0.135791	0.018439	1.81329	683
5.66406	177		189.8013	2.488676	6.193509	190		1.818139	0.243782	0.05943	1.79374	794
5.752785	174		192.3178	2.553156	6.518605	192		1.812652	0.170502	0.029071	1.77419	786
6.358913	209		199.5341	2.774668	7.698785	199		1.819859	0.318989	0.101754	1.77908	603
5.707705	173		188.624	2.438269	5.945156	188		1.817374	0.213096	0.04541	1.78886	893
7.602609	196		184.1234	2.486958	6.184962	184		1.807834	0.183123	0.033534	1.76931	741

3ECG_RR s	3ECG_HR n	3ECG_HR s	3ECG_RMS	3ECG_NN5	3ECG_PNN	3ECG_RR t	3ECG_TINN	3ECG_VLF		3ECG_LF p	3ECG_HF p	3ECG_VLF	3ECG_LF
110	66	11	140	153	48	17.7	264.5	660	2500	1750	13	51
40	84	5.2	26	26	6.3	9.6	143.3	70	670	118	8.1	78
34	73	3.5	31	37	10	8.2	122	110	310	259	16	45
32	80	6.5	36	61	16	8.7	137.9	21	210	126	5.9	59
44	80	5.8	38	67	17	10.7	157.7	310	370	136	38	45
33	89	4.2	41	2	0.45	5.5	81.4	44	100	40.6	23	55
100	77	15	160	52	14	9.9	145	190	1000	1710	6.5	35
19	82	2.5	17	0	0	4.8	74.9	22	120	41.4	12	66
31	86	6.4	23	11	2.6	8.1	120.4	54	360	141	9.7	65
28	94	7	15	1	0.22	9	132.8	68	200	51.1	22	62
24	93	6.1	17	3	0.66	5.8	86	9.1	240	84.7	2.7	72
46	66	5.5	47	100	31	11.7	184.9	410	310	242	43	32
32	78	5	28	27	7	9.6	135.1	120	190	153	27	40
60	63	8.4	58	94	30	15.5	228.7	670	1000	355	33	49
33	74	3.3	33	39	11	7.9	119.9	40	360	172	7	63
35	71	4.1	31	21	6	10.1	153.9	540	220	51.9	66	27
72	56	5.9	62	108	39	12.5	179.1	1300	1900	294	37	55
64	60	5.3	46	78	27	14	205.4	280	1300	287	15	70
15	94	3.9	10	0	0	4.3	65.4	8.9	100	21	6.8	77
43	82	8.3	35	43	11	9.1	141.7	160	600	225	16	61
50	68	7	32	35	11	11.4	164.8	460	1000	120	29	64
49	73	8.4	48	59	17	11.9	186.2	98	490	242	12	59
31	83	7.4	25	15	3.7	9.4	139.9	160	180	114	35	40
58	64	4.5	50	106	34	13.2	203.9	370	880	397	22	54
40	82	8	25	13	3.3	12.1	181.4	170	540	92.8	22	67
49	80	7.4	63	23	5.7	7.7	113.3	84	270	95.7	18	60
20	83	4.4	15	0	0	6	89.2	140	140	12.7	47	48
24	88	3.9	18	3	0.69	6.2	91.4	35	170	19.7	16	76
35	76	3.5	31	41	11	9.3	147	51	400	176	8.2	64
38	77	5.5	33	41	11	10.9	164.3	30	390	145	5.4	69
28	100	8.2	17	4	0.82	6.5	95.2	36	270	89.6	9.1	68
44	67	3.8	56	151	46	9.7	162.4	100	230	489	13	28
50	82	7.3	51	131	32	13.5	212.8	280	370	993	17	23

3ECG_HF	3ECG_LF n	3ECG_HF n	3ECG_LF/H	3ECG_(Seri	3Lean_mea	3Lean_std.	3Lean_Var	3Lean_Me	3Lean_(Ser	3Chair_lea	3Chair_std	3Chair_Var
36	48.1	34	1.4		26.66535	8.949685	80.09687	31.2		3.999321		
14	82.3	14.5	5.7		20.46549	1.932119	3.733084	20.5		4		
38	52.3	44.3	1.2		21.21522	1.744071	3.041782	21.3		3.448464		
35	52.8	31.2	1.7		14.43814	2.010372	4.041594	14.3		3		
17	61.4	22.7	2.7		16.88257	2.321489	5.389313	16.8		2.989233		
21	70.5	27.4	2.6		12.53667	1.979994	3.920377	12.5		3.983673		
59	19.8	33.6	0.59		15.09097	1.955907	3.825572	15.2		2.361396		
22	69.8	23.8	2.9		13.48141	2.028042	4.112956	13.5		2.77512		
25	69.6	27	2.6		16.28463	1.87482	3.514948	16.3		2.290301		
16	78.5	20.5	3.8		32.0565	7.068966	49.97028	28.4		2.696306		
25	72.1	25.2	2.9		19.62826	4.133053	17.08212	19.7		2.100823		
25	49	37.8	1.3		26.15821	3.349064	11.21623	26.1		3		
33	52.8	43.1	1.2		18.0345	3.651846	13.33598	17.8		3.317558		
17	68.8	24.3	2.8		18.49966	2.229234	4.969482	18.5		3.377688		
30	63.6	30.7	2.1		23.72454	3.888145	15.11767	23.7		4		
6.4	68	16.1	4.2		14.91394	1.963741	3.856279	14.7		2.064252		
8.5	81.3	12.6	6.4		27.76658	2.509568	6.297933	28.1		2.005468		
15	81.5	17.5	4.7		17.49774	3.61761	13.0871	17.5		2.455731		
16	80.1	16.6	4.8		19.48089	2.117002	4.481696	19.5		2		
					19.4224	2.22517	4.951382	19.45		2		
23	67.9	25.6	2.6		21.19381	2.05858	4.237752	21.4		3.806934		
7.5	88.5	10.4	8.5		19.69932	2.31556	5.361818	19.6		1.695147		
29	63.7	31.3	2		12.08317	2.333393	5.444724	12.1		4		
25	58.5	37.2	1.6		12.17591	2.585413	6.684361	12.1		3		
24	66.9	30.1	2.2		22.1987	2.362571	5.581741	22.1		3		
12	83.8	14.4	5.8		10.55915	3.00338	9.020292	10.6		3.391364		
21	58	20.3	2.9		23.80958	2.532123	6.411646	23.7		2.890561		
4.3	79.8	7.04	11		16.36048	2.986659	8.92013	16.4		2.325342		
8.7	79	9.13	8.7		29.9132	6.050189	36.60478	30.1		1.995896		
28	67.4	29.6	2.3		22.66123	2.225045	4.950826	22.7		2.00137		
26	67	25	2.7		20.16587	2.727492	7.439211	20.1		2.999316		
23	72.5	24.2	3		13.12428	2.018117	4.072796	13.3		1.088235		
59	29.8	63	0.47		23.66932	2.61321	6.828867	23.8		2		
60	26.3	69.9	0.38		17.49849	3.202185	10.25399	17.4		2.995885		

3Chair_me	3Chair_NO	3Chair_(Se	3Grid_x	3Grid_y	3PANAS_P	3PANAS_N	3PANAS_(S	3ADACL_A	3ADACL_A	3ADACL_B	3ADACL_B	3ADACL_(S
	1		8	1	41	16		14	7	12	8	
	1		6	3	31	15		13	7	11	13	
	22		7	2	37	15		16	7	12	12	
	1		6	3	28	14		12	8	8	10	
	3		6	4	30	24		13	11	13	9	
	5		6	4	35	26		13	6	11	12	
	61		6	4	24	10		12	7	7	8	
	187		7	3	36	12		16	6	11	5	
	53		6	3	36	19		17	6	15	7	
	106		8	2	47	21		18	5	15	11	
	27		7	3	34	14		14	6	14	10	
	25		7	3	39	19		16	5	15	10	
	1		7	3	47	16		17	7	14	9	
	3		5	3	35	12		13	14	6	15	
	3		6	3	38	13		13	7	7	14	
	88		8	3	46	18		19	5	8	9	
	22		7	3	39	16		15	7	15	8	
	7		7	3	35	13		12	7	12	13	
	129		8	2	41	19		16	5	15	6	
	1		9	1	42	11		17	5	13	6	
	69		7	2	40	22		16	8	14	10	
	36		9	2	48	10		19	6	11	7	
	3		5	3	37	16		15	9	14	9	
	4		8	2	40	20		16	7	16	8	
	1		6	4	30	11		14	7	5	9	
	34		8	2	34	15		17	6	11	6	
	7		8	4	40	24		18	8	14	9	
	37		5	4	25	16		15	9	12	12	
	3		7	3	35	12		16	5	11	7	
	3		7	3	40	25		14	13	14	10	
	23		6	3	33	21		14	10	13	6	
	19		4	2	31	22		15	7	15	5	
	28		8	1	39	16		19	7	15	8	
	5		8	4	39	13		16	7	11	16	

EMGnMax	EMGnMin	1EMGnNor	2EMGnNor	3EMGnNor	EMGaMax	EMGaMin	1EMGaNor	2EMGaNor	3EMGaNor	1Grid_xNo	2Grid_xNo	3Grid_xNo
267	163	0.119799	0.144936	0.310294	239	185	0.422707	0.291481	0.175028	0.875	0.5	0.875
246	214	0.316427	0.585181	0.408581	275	172	0.554668	0.411309	0.636933	0.375	0.625	0.625
212	175	0.555643	0.385681	0.271293	217	189	0.281788	0.360196	0.283026	0.875	0.25	0.75
193	167	0.350341	0.43918	0.452288	211	183	0.318305	0.440398	0.342768	0.625	0.375	0.625
192	163	0.346858	0.364591	0.551024	251	175	0.207781	0.144426	0.774648	0.875	0.25	0.625
239	167	0.225541	0.180276	0.190928	243	214	0.497365	0.30086	0.275732	0.75	0.125	0.625
257	165	0.095317	0.107727	0.097237	231	203	0.355726	0.361373	0.423782	0.875	0.375	0.625
230	199	0.28397	0.239497	0.227015	244	191	0.240037	0.229767	0.268463	0.75	0.25	0.75
239	203	0.322622	0.394306	0.42069	257	207	0.168483	0.226543	0.335593	0.875	0.25	0.625
219	171	0.228538	0.240025	0.320652	230	198	0.437463	0.4298	0.437881	0.875	0.5	0.875
198	167	0.333526	0.383347	0.262065	241	190	0.288429	0.344011	0.283419	0.25	0	0.75
185	159	0.353606	0.360852	0.335369	240	189	0.194338	0.271346	0.222398	0.75	0.25	0.75
224	190	0.327121	0.295434	0.285113	236	206	0.258089	0.412447	0.403624	0.75	0.375	0.75
202	167	0.405501	0.310138	0.350644	231	187	0.169681	0.2638	0.230061	0.75	0.625	0.5
199	159	0.220217	0.250489	0.324814	222	189	0.210478	0.309866	0.272793	0.875	0.125	0.625
230	167	0.158396	0.172454	0.177095	239	202	0.279503	0.324833	0.310525	0.625	0.5	0.875
242	215	0.345777	0.342332	0.335212	265	239	0.367812	0.341871	0.260356	0.75	0.375	0.75
239	215	0.384487	0.378015	0.376868	267	238	0.37591	0.353205	0.344196	0.75	0.125	0.75
242	198	0.361408	0.322974	0.339226	207	178	0.373074	0.306238	0.348828	1	0.125	0.875
195	163	0.311693	0.254098	0.276335	243	183	0.192614	0.165383	0.700933	0.875	0.125	1
213	175	0.22458	0.216975	0.25773	213	180	0.33918	0.361437	0.365967	0.875	0.25	0.75
708	5.6911	0.81249	0.869698	0.942952	577	102	0.085556	0.139514	0.213382	0.75	0.375	1
206	179	0.391231	0.43878	0.407607	224	191	0.346116	0.415495	0.310767	0.75	0.125	0.5
233	191	0.276949	0.415066	0.590974	220	191	0.333306	0.331849	0.367917	0.875	0.25	0.875
214	166	0.212388	0.235544	0.208662	240	189	0.271104	0.238721	0.26815	0.75	0.25	0.625
220	175	0.230296	0.299125	0.281764	247	194	0.245673	0.304045	0.199283	0.75	0.5	0.875
204	171	0.25153	0.341872	0.335857	261	197	0.159282	0.200519	0.192011	0.75	0.875	0.875
233	163	0.368355	0.345988	0.239003	223	193	0.339476	0.405117	0.351703	0.625	0.25	0.5
231	172	0.26621	0.266784	0.217774	278	183	0.10754	0.125226	0.118913	0.625	0.25	0.75
193	167	0.378296	0.392988	0.407777	208	182	0.336529	0.291596	0.300049	0.625	0.125	0.75
230	163	0.122756	0.186834	0.16714	208	181	0.344185	0.305263	0.419179	0.625	0.25	0.625
225	199	0.398032	0.391631	0.382809	216	190	0.304418	0.431715	0.366697	0.75	0.25	0.375
191	165	0.356339	0.320844	0.323624	207	181	0.387709	0.310483	0.293231	0.75	0.375	0.875
213	183	0.40885	0.383736	0.433999	203	175	0.412274	0.322697	0.325837	0.75	0.375	0.875

1ADACL_tc	2ADACL_tc	3ADACL_tot
2	20	11
0	14	4
-17	29	9
-9	8	2
-8	5	6
6	11	6
-14	13	4
-11	18	16
-8	16	19
6	24	17
9	20	12
7	18	16
6	10	15
-16	-7	-10
-7	13	-1
17	23	13
-1	15	15
-7	9	4
-1	19	20
13	9	19
5	12	12
15	26	17
-19	14	11
5	16	17
-19	13	3
-15	16	16
3	18	15
-8	12	6
-9	17	15
-11	2	5
8	14	11
-3	11	18
-11	12	19
-24	6	4

Appendix H - TETRIS CODE

TETRIS SCENARIO 1

```
#!/usr/bin/env python

# PyTris (c) Lukasz Grzegorz Maciak
# Licensed under GNU General Public License Version 3

import sys, pygame, random
from pygame.locals import *
from pygame import K_q
import time

img = pygame.image.load('C:/Users/erikm/Google Drive/My
   Documents/Skoleting/Master/Scenarios/1st Prototype 27
   apr/Experiment/Pytris-master/cat.png')
pygame.mixer.pre_init(44100,-16,2,2048)
pygame.mixer.init()
pygame.init()

pygame.mixer.music.load('C:/Users/erikm/Google Drive/My
   Documents/Skoleting/Master/Scenarios/1st Prototype 27
   apr/Experiment/Sound/Relaxing.mp3')
positivesound = pygame.mixer.Sound('C:/Users/erikm/Google
   Drive/My Documents/Skoleting/Master/Scenarios/1st
   Prototype 27 apr/Experiment/Sound/levelup.wav')
positivesound.set_volume(0.05)
pygame.mixer.music.play()
pygame.mixer.music.set_volume(0.3)

# useful constants
size = width, height = 800, 600
lines_per_level = 200           #increas level after x
   lines
delay = 800                     #speed
mode = False                            #for switching
   directions
counter = 0             #feedback bar counter
st = 1
bitnumber = 0
lastlinecount = 0

# define colors
black = 0, 0, 0
red = 255, 0, 0
green = 0, 255, 0
```

```
blue = 0, 0, 255
white = 255, 255, 255
yellow = 255, 255, 0
purple = 160, 32, 240
cyan = 0, 255, 255
orange = 255, 165, 0
gray =  45, 45, 45

currentcolor = red      #for feedback bar
myfont = pygame.font.Font(None, 24)
timeFont = pygame.font.Font(None, 24)
perf = myfont.render("PERFORMANCE:", 1, white)
myfont2 = pygame.font.Font(None, 14)
timeshow = timeFont.render("TIME:", 1, white)
startTime = time.time()
lastTime = time.time()
countdown = 300

block_size = 15         # each piece is 4 blocks
block_gap = 1           # distance between blocks

offset = block_size + block_gap

twidth = 10 * offset
right_edge = 125+twidth - block_size

theight = 24 * offset
bottom_edge = 100+theight - block_size

start_point = 125+twidth/2 - block_size, 100
next_point = 125+twidth+125, 100+100

# Block object exists mostly to allow us to break a piece
   into individual
# components once it is locked in place. This is why the
   color information
# is redundant
class Block(object):
        """ Represents a rectangular Tetris block """

        def __init__(self, x, y, color):

                self.x = x
                self.y = y
                self.color = color
        def draw(self):

                pygame.draw.rect(screen, self.color, self.
                   rect)

        @property
        def rect(self):
                return (self.x, self.y, block_size,
                   block_size)

class Piece(object):
        """ A Tetris Piece - composed of 4 blocks """

        def __init__(self,x,y):

                self.x = x
                self.y = y
                self.mobile = True

                self.rotation = 0 # defines which of the
                   members of self.positions to use for
                   this object

                # set of offsets that is applied to
                   coordinates of each block in this piece
                self.positions = None


        def get_blocks(self):
                """ Returns an array of 4 blocks which make
                    up this piece; each block has it’s own
                   coordinates and draw function """

                blocks = []

                for i in range(4):
                        blocks.append( Block(self.x + self.
                           positions[self.rotation][i][0],
                           self.y + self.positions[self.
                           rotation][i][1], self.color) )

                return blocks

        def draw(self):
                """ Draw all the blocks of this piece to
                   the screen """

                blocks = self.get_blocks()

                for b in blocks:
                        pygame.draw.rect(screen, self.color
                           , b.rect)

        # calling flip repeatedly cycles through all
            available positions and goes back to the first
            one
        def flip(self):
                if(self.rotation < len(self.positions) -1):
                        self.rotation += 1
                else:
                        self.rotation = 0

        def set_point(self, x, y):
                self.x = x
                self.y = y


# Define different piece types: Z, S, O, T, I, L, J

class ZPiece(Piece):

        def __init__(self,x,y):

                super(ZPiece, self).__init__(x, y)

                self.color = red

                self.positions =        [
                                                ( (0,0), (
                                                   offset,
                                                   0), (
                                                   offset,
                                                   offset),
                                                    (2*
                                                   offset,
                                                   offset)
                                                        ),
                                                ( (0,0),
                                                    (0,
                                                   offset),
                                                    (-
                                                   offset,
                                                   offset),
                                                    (-
                                                   offset,
                                                   2*offset
                                                   )    )
                                       ]
class SPiece(Piece):

        def __init__(self,x,y):

                super(SPiece, self).__init__(x, y)

                self.color = green

                self.positions =        [
                                                ( (0,0), (-
                                                   offset,
                                                   0), (-
                                                   offset,
                                                   offset),
                                                    (-2*
                                                   offset,
                                                   offset)
                                                     ),
                                                ( (0,0),
                                                    (0,
                                                   offset),
                                                    (offset
                                                   , offset
                                                   ), (
                                                   offset,
                                                   2*offset
                                                   )      )
                                       ]


class OPiece(Piece):

        def __init__(self, x, y):

                super(OPiece, self).__init__(x,y)
                self.color = yellow

                self.positions =        [
                                                ( (0,0), (
                                                   offset,
                                                   0), (
                                                   offset,
                                                   offset),
                                                    (0,
                                                   offset)
                                                       )
                                        ]


class TPiece(Piece):

        def __init__(self,x,y):

                super(TPiece, self).__init__(x, y)

                self.color = purple

                self.positions =        [
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (-
                                                   offset,
                                                   offset),
                                                    (offset
                                                   , offset
                                                   )
                                                   ),
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (offset
                                                   , offset
                                                   ), (0,
                                                   2*offset
                                                   )   ),
                                                ( (0,0), (-
                                                   offset,
                                                   0), (
                                                   offset,
                                                   0), (0,
                                                   offset)

                                                   ),
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (-
                                                   offset,
                                                   offset),
                                                    (0, 2*
                                                   offset)
                                                    ),
                                        ]


class IPiece(Piece):

        def __init__(self,x,y):

                super(IPiece, self).__init__(x, y)

                self.color = cyan

                self.positions =        [
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (0, 2*
                                                   offset),
                                                    (0, 3*
                                                   offset)
                                                       ),
                                                ( (0,0), (
                                                   offset,
                                                   0), (2*
                                                   offset,
                                                   0), (3*
                                                   offset,
                                                   0)
                                                   )
                                        ]


class LPiece(Piece):
       def __init__(self,x,y):

        super(LPiece, self).__init__(x, y)

        self.color = orange

        self.positions =        [
                                        ( (0,0),
                                          (0,
                                          offset),
                                           (0, 2*
                                          offset),
                                           (offset
                                          , 2*
                                          offset)

                                          ),
                                        ( (0,0),
                                          (0,
                                          offset),
                                           (offset
                                          , 0),
                                          (2*
                                          offset,
                                          0)
                                                   )
                                          ,
                                        ( (0,0), (-
                                          offset,
                                          0), (0,
                                          offset),
                                           (0, 2*
                                          offset)
                                                   ),
                                        ( (0,0),
                                          (0,
                                          offset),
                                           (-
                                          offset,
                                          offset),
                                           (-2*
                                          offset,
                                          offset)
                                                ),
                                        ]

class JPiece(Piece):

        def __init__(self,x,y):

                super(JPiece, self).__init__(x, y)

                self.color = blue

                self.positions =        [
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (0, 2*
                                                   offset),
                                                    (-
                                                   offset,
                                                   2*offset
                                                   )
                                                         )
                                                   ,
                                                ( (0,0), (
                                                   offset,
                                                   0), (2*
                                                   offset,
                                                   0), (2*
                                                   offset,
                                                   offset)

                                                   ),
                                                ( (0,0), (
                                                   offset,
                                                   0), (0,
                                                   offset),
                                                    (0, 2*
                                                   offset)
                                                         )
                                                   ,
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (offset
                                                   , offset
                                                   ), (2*
```

```
                                                        offset,
                                                        offset)
                                                             ),
                                        ]



class Grid(object):

        def __init__(self):

                self.current = None
                self.next = None
                self.blocks = []

                self.next_piece()

                self.total_cleared_lines = 0
                self.cleared_lines = 0
                self.level = 1
                self.score = 0

                self.lines_til_next_level = lines_per_level

                self.delay = delay

                self.game_over = False

                self.next_rect = pygame.Rect(150+125,
                   90+100, 300, 300)

                # this is for multiplying scores
                self.multiplier =       {
                                                0 : 0,

                                                   # no
                                                   lines
                                                   cleared
                                                1 : 40,

                                                   # single
                                                    line
                                                   cleared
                                                2 : 100,
                                                    # two
                                                    lines
                                                    cleared
                                                3 : 300,

                                                    # three
                                                    lines
                                                    cleared
                                                4 : 1200

                                                    # TETRIS
                                        }

                self.nfont = pygame.font.Font(None, 24)
                self.largefont = pygame.font.Font(None, 40)
                self.smallfont = pygame.font.Font(None, 14)


        def random_piece(self):
                rekkefolge
                   =[0,3,1,2,0,0,1,1,1,0,0,3,0,0,2,3,0,0,2,1,1,2,3,1,2,

                pcs =   {
                                #0 : ZPiece(*next_point),
                                0 : OPiece(*next_point),
                                #2 : SPiece(*next_point),
                                #1 : TPiece(*next_point),
                                1 : IPiece(*next_point),
                                2 : LPiece(*next_point),
                                3 : JPiece(*next_point)
                        }
                global bitnumber
                c = rekkefolge[bitnumber]
                bitnumber+=1
                return pcs[c]


        def next_piece(self):

                if not self.next:
                        self.next = self.random_piece()
                        self.current = self.random_piece()
                else:
                        self.current = self.next
                        self.next = self.random_piece()
        self.current.set_point(*start_point)

def move_down(self):

        if(self.current.y < theight+100):
                self.current.y += offset

                if self.has_overlap():
                        self.current.y -= offset
                        self.current.mobile = False
        else:
                self.current.mobile = False

        if self.current.mobile == False and self.
           current.y == 100:
                self.game_over = True

# this is a hard drop - just go all the way down
   until you hit something
def drop_down(self):

        while(self.current.mobile): self.move_down
            ()

def move_right(self):
        if(self.current.mobile): self.current.x +=
           offset

        if self.has_overlap(): self.current.x -=
           offset

def move_left(self):
        if(self.current.mobile): self.current.x -=
           offset
        if self.has_overlap(): self.current.x +=
           offset

# TODO: wall kick logic needed
def rotate(self):
        self.current.flip()

        while self.has_overlap(): self.current.flip
            ()
```

```
# Saves individual blocks of the current piece -
   they become part of the grid
# Automatically check for lines
def remember_block_positions(self):

        self.blocks.extend(self.current.get_blocks
           ())
        self.next_piece()

        self.blocks = sorted(self.blocks, key=
           lambda block: block.y)

        self.check_for_lines()

        #for b in self.blocks: print str(b.y)+", "

def draw_blocks(self):

        for b in self.blocks:
                b.draw()

# check if current piece overlaps with walls or
   with other pieces
def has_overlap(self):

        blocks = self.current.get_blocks()

        for b in blocks:
                if b.x < 125 or b.x > right_edge
                   or b.y < 100 or b.y >
                   bottom_edge: return True

        # TODO: optimize the shit out of this
        for b in self.blocks:
                for c in self.current.get_blocks():
                        if b.x == c.x and b.y == c.
                           y: return True

        return False

# TODO: there ought to be a better way to do this
def check_for_lines(self):

        lines = {}
```

```
                for b in self.blocks:

                        if b.y in lines:
                                lines[b.y] +=1
                        else:
                                lines[b.y] = 1

                lines_to_be_destroyed = []

                for ln in lines:
                        if lines[ln] == 10:
                                self.total_cleared_lines +=
                                    1
                                self.cleared_lines += 1
                                lines_to_be_destroyed.
                                   append(ln)

                for l in lines_to_be_destroyed:
                        self.destroy_line(l)

                copy_of_blocks = self.blocks[:]

                if len(lines_to_be_destroyed) > 0:
                        self.collapse_hovering_blocks(min(
                           lines_to_be_destroyed), len(
                           lines_to_be_destroyed))

                        #for ln in lines_to_be_destroyed:
                                #self.
                                   collapse_hovering_blocks
                                   (ln, copy_of_blocks)

                self.calculate_score()


        def destroy_line(self, ln):

                # note the slice notation - I'm iterating
                   over a copy of self.blocks but removing
                # from the original
                for b in self.blocks[:]:
                        if b.y == ln: self.blocks.remove(b)
```

```
    def collapse_hovering_blocks(self, ln, total):
            """ drop down all the blocks that are
                hovering """

            for i,b in enumerate(self.blocks):
                    if b.y < ln:
                            self.blocks[i].y += offset
                                * total



    def block_overlaps(self, block, block_list):


            for b in block_list:
                    if block.y == b.y and block.x == b.
                        x:
                            return True

            return False



    def calculate_score(self):

            # score is calculated like so: M * ( N + 1
                ) where:
            # M is multiplier (see self.multiplier)
            # N is level (self.level)

            self.score += self.multiplier[self.
                cleared_lines] * (self.level + 1)

            tmp = self.lines_til_next_level - self.
                cleared_lines

            if tmp <= 0:
                    self.level +=1
                    self.lines_til_next_level =
                        lines_per_level + tmp

                    self.delay -= self.level * 30

                    if self.delay < 10: self.delay = 10
```

```
        else:
                self.lines_til_next_level = tmp

        self.cleared_lines = 0
        self.draw_text()


# redraws the whole UI - we should really be
   bliting this shit
def draw_ui(self):

        pygame.draw.rect(screen, gray, (125,100,
           right_edge+offset-125, bottom_edge+
           offset-100))

        for i in xrange(10): pygame.draw.line(
           screen, black, (i*offset+125, 100),
           (125+i*offset, theight+offset+100))
        for i in xrange(24): pygame.draw.line(
           screen, black, (125, i*offset+100), (
           twidth+125, i*offset+100))

        self.draw_text()


# I really dislike the default font in pygame
def draw_text(self):

        msg = self.nfont.render("NEXT PIECE:", 1,
           white)
        screen.blit(msg, (250+125, 50+100))

        pygame.draw.rect(screen, black, (250+125,
           100+200, 200, 100))

        #sc = self.nfont.render("SCORE: " + str(
           self.score), 1, white)
        #screen.blit(sc, (250+125, 200+100))

        #sc = self.nfont.render("LINES: " + str(
           self.total_cleared_lines), 1, white)
        #screen.blit(sc, (250+125, 230+100))

        #sc = self.nfont.render("LEVEL: " + str(
           self.level), 1, white)
```

```
                #screen.blit(sc, (250, 260))



                #msg = self.smallfont.render("Left, Right
                   Arrow to move", 1, white)
                #screen.blit(msg, (230, 290))

                #msg = self.smallfont.render("Up Arrow to
                   flip", 1, white)
                #screen.blit(msg, (230, 300))

                #msg = self.smallfont.render("Down Arrow to
                    move downw", 1, white)
                #screen.blit(msg, (230, 310))

                #msg = self.smallfont.render("Space or
                   Enter to drop down", 1, white)
                #screen.blit(msg, (230, 320))

                #msg = self.smallfont.render("Esc to pause,
                    F1 for new game", 1, white)
                #screen.blit(msg, (230, 330))

                #screen.blit(self.smallfont.render("ver
                   0.2", 1, white), (230, 350))

        def draw_game_over(self):

                msg = self.largefont.render("GAME OVER", 1,
                    red)
                screen.blit(msg, (195+125,95+100))

                #msg2 = self.nfont.render("PRESS F1 TO PLAY
                    AGAIN", 1, red)
                #screen.blit(msg2, (175, 130))




# magic
screen = pygame.display.set_mode(size)
pygame.key.set_repeat(100, 150)

#pygame.time.set_timer(USEREVENT+1, delay)
```

```
grid = Grid()
clock = pygame.time.Clock()

paused = False

time_elapsed = 0

while st == 1:
        currentTime = time.time()
        if grid.total_cleared_lines>lastlinecount:
                positivesound.play()
                lastlinecount=grid.total_cleared_lines
                counter+=1
        if not paused and not grid.game_over:

                time_elapsed += clock.tick()

                #print str(grid.delay)

                # timed block drop
                if time_elapsed > grid.delay:
                        time_elapsed = 0
                        grid.move_down()

                # check if a line was created and remove it
                grid.check_for_lines()

                # clear the next piece area
                pygame.draw.rect(screen, black, grid.
                   next_rect)

                # draw the gridlines
                grid.draw_ui()

                # if the current piece is locked in place
                   memorize it’s position
                if not grid.current.mobile:
                        grid.remember_block_positions()

                # draw the memorized blocks
                grid.draw_blocks()


        for event in pygame.event.get():
```

```
        if event.type == pygame.QUIT: sys.exit()


        if event.type == KEYDOWN:

                if event.key == K_ESCAPE: paused =
                   not paused

                if event.key == K_F1:
                        grid = Grid()
                        clock = pygame.time.Clock()
                        counter = 0
                        paused = False
                        time_elapsed = 0

                if mode == True:
                        if not paused and not grid.
                           game_over:
                                if event.key == K_d
                                   : grid.move_left
                                   ()
                                if event.key == K_a
                                   : grid.
                                   move_right()
                                if event.key == K_s
                                   : grid.move_down
                                   ()
                                if event.key == K_w
                                   : grid.rotate()
                                if event.key ==
                                   K_SPACE: grid.
                                   drop_down()
                                if event.key ==
                                   K_RETURN: grid.
                                   drop_down()
                                if event.key == K_q
                                   : mode = False
                                if event.key == K_1
                                   : grid.delay =
                                   100
                                if event.key == K_2
                                   : grid.delay =
                                   200
                                if event.key == K_3
```

```
                                    : grid.delay =
                                    300
                                if event.key == K_4
                                    : grid.delay =
                                    400
                                if event.key == K_5
                                    : grid.delay =
                                    500
                                if event.key == K_6
                                    : grid.delay =
                                    600
                                if event.key == K_7
                                    : grid.delay =
                                    700
                                if event.key == K_8
                                    : grid.delay =
                                    800
                                if event.key == K_9
                                    : grid.delay =
                                    900
                                if event.key == K_0
                                    : grid.delay =
                                    1000
                                if event.key ==
                                    K_UP: counter
                                    +=1;
                                    positivesound.
                                    play()
                                if event.key ==
                                    K_DOWN: counter
                                    -=1.5
                                if event.key == K_m
                                    : counter-=4
                                if event.key == K_p
                                    : st=0
                if mode == False:
                        if not paused and not grid.
                            game_over:
                                if event.key == K_d
                                    : grid.
                                    move_right()
                                if event.key == K_a
                                    : grid.move_left
                                    ()
                                if event.key == K_s
```

```
    : grid.move_down
    ()
if event.key == K_w
    : grid.rotate()
if event.key ==
    K_SPACE: grid.
    drop_down()
if event.key ==
    K_RETURN: grid.
    drop_down()
if event.key == K_e
    : mode = True
if event.key == K_1
    : grid.delay =
    100
if event.key == K_2
    : grid.delay =
    200
if event.key == K_3
    : grid.delay =
    300
if event.key == K_4
    : grid.delay =
    400
if event.key == K_5
    : grid.delay =
    500
if event.key == K_6
    : grid.delay =
    600
if event.key == K_7
    : grid.delay =
    700
if event.key == K_8
    : grid.delay =
    800
if event.key == K_9
    : grid.delay =
    900
if event.key == K_0
    : grid.delay =
    1000
if event.key ==
    K_UP: counter
    +=1;
```

```
                                             positivesound.
                                                play()
                                             if event.key ==
                                                K_DOWN: counter
                                                -=1.5
                                             if event.key == K_m
                                                : counter-=4
                                             if event.key == K_p
                                                : st=0


        if not paused and not grid.game_over:

                grid.current.draw()      # draw current
                   piece
                grid.next.draw()         # draw the next
                   piece
                if counter < 0:
                        currentcolor = red
                else:
                        currentcolor = green
                if counter >=11: counter = 11
                if counter <=-10: counter = -10
                pygame.draw.rect(screen, black,
                   (370+125,100,200,1000), 0)
                screen.blit(perf, (405+125, 100+50))
                #screen.blit(timeshow, (250, 300))
                if currentTime - lastTime > 1:
                        countdown-=1
                        if countdown <= 0:
                                countdown = 0
                        lastTime = currentTime
                m, s = divmod(countdown, 60)
                #if m >= 1:
                #       screen.blit(timeFont.render(str(m)
                   ,1,white), (305, 300))
                #       screen.blit(timeFont.render(":",1,
                   white), (313, 300))
                #       screen.blit(timeFont.render(str(s)
                   ,1,white), (320, 300))
                #else:
                #       screen.blit(timeFont.render(str(s)
                   ,1,white), (320, 300))
                pygame.draw.rect(screen, currentcolor,
                   (370+125,100+230,200,-counter*15), 0)
                screen.blit(img,(180+125,100+250))
                pygame.display.flip()

        # draw game over message
        if grid.game_over:
                #grid.draw_game_over()
                counter = 0
                if currentTime - lastTime > 1:
                        countdown-=1
                        if countdown <= 0:
                                countdown = 0
                        lastTime = currentTime
                #pygame.display.flip()
                grid = Grid()
                clock = pygame.time.Clock()
                counter = 0
                paused = False
                time_elapsed = 0
```

TETRIS SCENARIO 2

```
#!/usr/bin/env python

# PyTris (c) Lukasz Grzegorz Maciak
# Licensed under GNU General Public License Version 3

import sys, pygame, random
from pygame.locals import *
from pygame import K_q
import time
from time import sleep

pygame.mixer.pre_init(44100,-16,2,2048)
pygame.mixer.init()
pygame.init()

pygame.mixer.music.load('C:/Users/erikm/Google Drive/My
   Documents/Skoleting/Master/Scenarios/1st Prototype 27
   apr/Experiment/Sound/War Warrior.mp3')
positivesound = pygame.mixer.Sound('C:/Users/erikm/Google
   Drive/My Documents/Skoleting/Master/Scenarios/1st
   Prototype 27 apr/Experiment/Sound/levelup.wav')
positivesound.set_volume(0.7)
negativesound = pygame.mixer.Sound('C:/Users/erikm/Google
   Drive/My Documents/Skoleting/Master/Scenarios/1st
   Prototype 27 apr/Experiment/Sound/Wrongbuzz.wav')
negativesound.set_volume(0.6)
pygame.mixer.music.play()

# useful constants
size = width, height = 800, 600
lines_per_level = 200          #increas level after x
   lines
delay = 200                    #speed
mode = True                            #for switching
   directions
counter = 0            #feedback bar counter
st = 1
bitnumber = 0

# define colors
black = 0, 0, 0
red = 255, 0, 0
green = 0, 255, 0
```

```
blue = 0, 0, 255
white = 255, 255, 255
yellow = 255, 255, 0
purple = 160, 32, 240
cyan = 0, 255, 255
orange = 255, 165, 0
gray =  45, 45, 45

currentcolor = red      #for feedback bar
myfont = pygame.font.Font(None, 24)
timeFont = pygame.font.Font(None, 50)
perf = myfont.render("PERFORMANCE:", 1, white)
myfont2 = pygame.font.Font(None, 14)
timeshow = timeFont.render("TIME:", 1, white)
startTime = time.time()
lastTime = time.time()
lastTime1 = time.time()
countdown = 270

block_size = 15         # each piece is 4 blocks
block_gap = 1           # distance between blocks

offset = block_size + block_gap

twidth = 10 * offset
right_edge = 125+twidth - block_size

theight = 24 * offset
bottom_edge = 100+theight - block_size

start_point = 125 + twidth/2 - block_size, 100
next_point = 125 + twidth+125, 100+100

# Block object exists mostly to allow us to break a piece
   into individual
# components once it is locked in place. This is why the
   color information
# is redundant
class Block(object):
        """ Represents a rectangular Tetris block """

        def __init__(self, x, y, color):

                self.x = x
                self.y = y
        self.color = color

    def draw(self):

        pygame.draw.rect(screen, self.color, self.
           rect)

    @property
    def rect(self):
        return (self.x, self.y, block_size,
           block_size)

class Piece(object):
    """ A Tetris Piece - composed of 4 blocks """

    def __init__(self,x,y):

        self.x = x
        self.y = y
        self.mobile = True

        self.rotation = 0 # defines which of the
           members of self.positions to use for
           this object

        # set of offsets that is applied to
           coordinates of each block in this piece
        self.positions = None


    def get_blocks(self):
        """ Returns an array of 4 blocks which make
            up this piece; each block has it's own
           coordinates and draw function """

        blocks = []

        for i in range(4):
            blocks.append( Block(self.x + self.
               positions[self.rotation][i][0],
               self.y + self.positions[self.
               rotation][i][1], self.color) )

        return blocks
        def draw(self):
                """ Draw all the blocks of this piece to
                   the screen """

                blocks = self.get_blocks()

                for b in blocks:
                        pygame.draw.rect(screen, self.color
                           , b.rect)

        # calling flip repeatedly cycles through all
           available positions and goes back to the first
           one
        def flip(self):
                if(self.rotation < len(self.positions) -1):
                        self.rotation += 1
                else:
                        self.rotation = 0

        def set_point(self, x, y):
                self.x = x
                self.y = y


# Define different piece types: Z, S, O, T, I, L, J

class ZPiece(Piece):

        def __init__(self,x,y):

                super(ZPiece, self).__init__(x, y)

                self.color = red

                self.positions =        [
                                                ( (0,0), (
                                                   offset,
                                                   0), (
                                                   offset,
                                                   offset),
                                                    (2*
                                                   offset,
                                                   offset)
                                                        ),
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (-
                                                   offset,
                                                   offset),
                                                    (-
                                                   offset,
                                                   2*offset
                                                   )    )
                                          ]
class SPiece(Piece):

        def __init__(self,x,y):

                super(SPiece, self).__init__(x, y)

                self.color = green

                self.positions =        [
                                                ( (0,0), (-
                                                   offset,
                                                   0), (-
                                                   offset,
                                                   offset),
                                                    (-2*
                                                   offset,
                                                   offset)
                                                     ),
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (offset
                                                   , offset
                                                   ), (
                                                   offset,
                                                   2*offset
                                                   )      )
                                          ]


class OPiece(Piece):

        def __init__(self, x, y):

                super(OPiece, self).__init__(x,y)
                self.color = yellow

                self.positions =        [
                                                ( (0,0), (
                                                   offset,
                                                   0), (
                                                   offset,
                                                   offset),
                                                    (0,
                                                   offset)
                                                       )
                                        ]


class TPiece(Piece):

        def __init__(self,x,y):

                super(TPiece, self).__init__(x, y)

                self.color = purple

                self.positions =        [
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (-
                                                   offset,
                                                   offset),
                                                    (offset
                                                   , offset
                                                   )
                                                   ),
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (offset
                                                   , offset
                                                   ), (0,
                                                   2*offset
                                                   )   ),
                                                ( (0,0), (-
                                                   offset,
                                                   0), (
                                                   offset,
                                                   0), (0,
                                                   offset)

                                                   ),
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (-
                                                   offset,
                                                   offset),
                                                    (0, 2*
                                                   offset)
                                                    ),
                                        ]


class IPiece(Piece):

        def __init__(self,x,y):

                super(IPiece, self).__init__(x, y)

                self.color = cyan

                self.positions =        [
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (0, 2*
                                                   offset),
                                                    (0, 3*
                                                   offset)
                                                        ),
                                                ( (0,0), (
                                                   offset,
                                                   0), (2*
                                                   offset,
                                                   0), (3*
                                                   offset,
                                                   0)
                                                   )
                                        ]
```

```
class LPiece(Piece):

        def __init__(self,x,y):

                super(LPiece, self).__init__(x, y)

                self.color = orange

                self.positions =        [
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (0, 2*
                                                   offset),
                                                    (offset
                                                   , 2*
                                                   offset)

                                                   ),
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (offset
                                                   , 0),
                                                   (2*
                                                   offset,
                                                   0)
                                                         )
                                                   ,
                                                ( (0,0), (-
                                                   offset,
                                                   0), (0,
                                                   offset),
                                                    (0, 2*
                                                   offset)
                                                        ),
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (-
                                                   offset,
                                                   offset),
                                                    (-2*
                                                   offset,
                                                   offset)
```

```
                                                             ),
                                              ]

class JPiece(Piece):

        def __init__(self,x,y):

                super(JPiece, self).__init__(x, y)

                self.color = blue

                self.positions =        [
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (0, 2*
                                                   offset),
                                                    (-
                                                   offset,
                                                   2*offset
                                                   )
                                                          )
                                                   ,
                                                ( (0,0), (
                                                   offset,
                                                   0), (2*
                                                   offset,
                                                   0), (2*
                                                   offset,
                                                   offset)

                                                   ),
                                                ( (0,0), (
                                                   offset,
                                                   0), (0,
                                                   offset),
                                                    (0, 2*
                                                   offset)
                                                          )
                                                   ,
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (offset
                                                   , offset
```

```
                                                        ), (2*
                                                        offset,
                                                        offset)
                                                             ),
                                            ]



class Grid(object):

        def __init__(self):

                self.current = None
                self.next = None
                self.blocks = []

                self.next_piece()

                self.total_cleared_lines = 0
                self.cleared_lines = 0
                self.level = 1
                self.score = 0

                self.lines_til_next_level = lines_per_level

                self.delay = delay

                self.game_over = False

                self.next_rect = pygame.Rect(150+125,
                   90+100, 300, 300)

                # this is for multiplying scores
                self.multiplier =       {
                                                0 : 0,

                                                   # no
                                                   lines
                                                   cleared
                                                1 : 40,

                                                   # single
                                                    line
                                                   cleared
                                                2 : 100,
                      # two
                      lines
                      cleared
                    3 : 300,

                      # three
                      lines
                      cleared
                    4 : 1200

                      # TETRIS
                }

    self.nfont = pygame.font.Font(None, 24)
    self.largefont = pygame.font.Font(None, 40)
    self.smallfont = pygame.font.Font(None, 14)


def random_piece(self):
    rekkefolge =
      [3,5,3,5,4,0,1,5,5,1,1,2,1,2,5,2,3,0,2,2,0,3,6,0,4,5,

    pcs =   {
                0 : ZPiece(*next_point),
                1 : OPiece(*next_point),
                2 : SPiece(*next_point),
                3 : TPiece(*next_point),
                4 : IPiece(*next_point),
                5 : LPiece(*next_point),
                6 : JPiece(*next_point)
            }

    global bitnumber
    c = rekkefolge[bitnumber]
    bitnumber+=1
    return pcs[c]


def next_piece(self):

    if not self.next:
        self.next = self.random_piece()
        self.current = self.random_piece()
    else:
```

```
                self.current = self.next
                self.next = self.random_piece()

        self.current.set_point(*start_point)

def move_down(self):

        if(self.current.y < theight+100):
                self.current.y += offset

                if self.has_overlap():
                        self.current.y -= offset
                        self.current.mobile = False
        else:
                self.current.mobile = False

        if self.current.mobile == False and self.
            current.y == 100:
                self.game_over = True

# this is a hard drop - just go all the way down
    until you hit something
def drop_down(self):

        while(self.current.mobile): self.move_down
            ()

def move_right(self):
        if(self.current.mobile): self.current.x +=
            offset

        if self.has_overlap(): self.current.x -=
            offset

def move_left(self):
        if(self.current.mobile): self.current.x -=
            offset
        if self.has_overlap(): self.current.x +=
            offset

# TODO: wall kick logic needed
def rotate(self):
        self.current.flip()

        while self.has_overlap(): self.current.flip
```

```
            ()

    # Saves individual blocks of the current piece -
        they become part of the grid
    # Automatically check for lines
    def remember_block_positions(self):

            self.blocks.extend(self.current.get_blocks
                ())
            self.next_piece()

            self.blocks = sorted(self.blocks, key=
                lambda block: block.y)

            self.check_for_lines()

            #for b in self.blocks: print str(b.y)+", "

    def draw_blocks(self):

            for b in self.blocks:
                    b.draw()

    # check if current piece overlaps with walls or
        with other pieces
    def has_overlap(self):

            blocks = self.current.get_blocks()

            for b in blocks:
                    if b.x < 125 or b.x > right_edge
                        or b.y < 100 or b.y >
                        bottom_edge: return True

            # TODO: optimize the shit out of this
            for b in self.blocks:
                    for c in self.current.get_blocks():
                            if b.x == c.x and b.y == c.
                                y: return True

            return False

    # TODO: there ought to be a better way to do this
    def check_for_lines(self):
            lines = {}

            for b in self.blocks:

                    if b.y in lines:
                            lines[b.y] +=1
                    else:
                            lines[b.y] = 1

            lines_to_be_destroyed = []

            for ln in lines:
                    if lines[ln] == 10:
                            self.total_cleared_lines +=
                                 1
                            self.cleared_lines += 1
                            lines_to_be_destroyed.
                                append(ln)

            for l in lines_to_be_destroyed:
                    self.destroy_line(l)

            copy_of_blocks = self.blocks[:]

            if len(lines_to_be_destroyed) > 0:
                    self.collapse_hovering_blocks(min(
                        lines_to_be_destroyed), len(
                        lines_to_be_destroyed))

                    #for ln in lines_to_be_destroyed:
                            #self.
                                collapse_hovering_blocks
                                (ln, copy_of_blocks)

            self.calculate_score()


    def destroy_line(self, ln):

            # note the slice notation - I'm iterating
                over a copy of self.blocks but removing
            # from the original
            for b in self.blocks[:]:
                    if b.y == ln: self.blocks.remove(b)
```

```
    def collapse_hovering_blocks(self, ln, total):
            """ drop down all the blocks that are
                hovering """

            for i,b in enumerate(self.blocks):
                    if b.y < ln:
                            self.blocks[i].y += offset
                                * total



    def block_overlaps(self, block, block_list):


            for b in block_list:
                    if block.y == b.y and block.x == b.
                        x:
                            return True

            return False



    def calculate_score(self):

            # score is calculated like so: M * ( N + 1
                ) where:
            # M is multiplier (see self.multiplier)
            # N is level (self.level)

            self.score += self.multiplier[self.
                cleared_lines] * (self.level + 1)

            tmp = self.lines_til_next_level - self.
                cleared_lines

            if tmp <= 0:
                    self.level +=1
                    self.lines_til_next_level =
                        lines_per_level + tmp

                    self.delay -= self.level * 30
```

```
                        if self.delay < 10: self.delay = 10

                else:
                        self.lines_til_next_level = tmp

                self.cleared_lines = 0
                self.draw_text()


        # redraws the whole UI - we should really be
           bliting this shit
        def draw_ui(self):

                pygame.draw.rect(screen, gray, (125,100,
                   right_edge+offset-125, bottom_edge+
                   offset-100))

                for i in xrange(10): pygame.draw.line(
                   screen, black, (i*offset+125, 100),
                   (125+i*offset, theight+offset+100))
                for i in xrange(24): pygame.draw.line(
                   screen, black, (125, i*offset+100), (
                   twidth+125, i*offset+100))

                self.draw_text()


        # I really dislike the default font in pygame
        def draw_text(self):

                msg = self.nfont.render("NEXT PIECE:", 1,
                   white)
                screen.blit(msg, (250+125, 50+100))

                pygame.draw.rect(screen, black, (250+125,
                   200+100, 200, 100))

                sc = self.nfont.render("YOUR SCORE: ", 1,
                   white)
                screen.blit(sc, (240+125, 220+100))
                sc = self.largefont.render(str(self.score)
                   ,1,white)
                screen.blit(sc, (270+125, 240+100))

                #sc = self.nfont.render("LINES: " + str(
            self.total_cleared_lines), 1, white)
        #screen.blit(sc, (250, 230))

        #sc = self.nfont.render("LEVEL: " + str(
            self.level), 1, white)
        #screen.blit(sc, (250, 260))



        #msg = self.smallfont.render("Left, Right
            Arrow to move", 1, white)
        #screen.blit(msg, (230, 290))

        #msg = self.smallfont.render("Up Arrow to
            flip", 1, white)
        #screen.blit(msg, (230, 300))

        #msg = self.smallfont.render("Down Arrow to
             move downw", 1, white)
        #screen.blit(msg, (230, 310))

        #msg = self.smallfont.render("Space or
            Enter to drop down", 1, white)
        #screen.blit(msg, (230, 320))

        #msg = self.smallfont.render("Esc to pause,
             F1 for new game", 1, white)
        #screen.blit(msg, (230, 330))

        #screen.blit(self.smallfont.render("ver
            0.2", 1, white), (230, 350))

def draw_game_over(self):

        pygame.draw.rect(screen, black,
            (125,0,800,800),0)

        msg = self.largefont.render("GAME OVER", 1,
             red)
        screen.blit(msg, (195+125,95+100))

        #msg2 = self.nfont.render("PRESS F1 TO PLAY
             AGAIN", 1, red)
        #screen.blit(msg2, (175, 130))
```

```
# magic
screen = pygame.display.set_mode(size)
pygame.key.set_repeat(100, 150)

#pygame.time.set_timer(USEREVENT+1, delay)

grid = Grid()
clock = pygame.time.Clock()

paused = False

time_elapsed = 0

while st == 1:
        currentTime = time.time()
        if (currentTime-lastTime1>150):
                mode = False
                grid.delay = 170
        if (currentTime-lastTime1>240):
                mode = True
                grid.delay = 150
        if not paused and not grid.game_over:

                time_elapsed += clock.tick()

                #print str(grid.delay)

                # timed block drop
                if time_elapsed > grid.delay:
                        time_elapsed = 0
                        grid.move_down()

                # check if a line was created and remove it
                grid.check_for_lines()

                # clear the next piece area
                pygame.draw.rect(screen, black, grid.
                   next_rect)

                # draw the gridlines
                grid.draw_ui()
                # if the current piece is locked in place
                    memorize it’s position
                if not grid.current.mobile:
                        grid.remember_block_positions()

                # draw the memorized blocks
                grid.draw_blocks()


        for event in pygame.event.get():

                if event.type == pygame.QUIT: sys.exit()


                if event.type == KEYDOWN:

                        if event.key == K_ESCAPE: paused =
                            not paused

                        if event.key == K_F1:
                                grid = Grid()
                                clock = pygame.time.Clock()
                                counter = 0
                                #countdown = 300
                                paused = False
                                time_elapsed = 0

                        if mode == True:
                                if not paused and not grid.
                                    game_over:
                                        if event.key == K_d
                                            : grid.move_left
                                            ()
                                        if event.key == K_a
                                            : grid.
                                            move_right()
                                        if event.key == K_s
                                            : grid.move_down
                                            ()
                                        if event.key == K_w
                                            : grid.rotate()
                                        if event.key ==
                                            K_SPACE: grid.
                                            drop_down()
                                        if event.key ==
    K_RETURN: grid.
    drop_down()
if event.key == K_q
    : mode = False
if event.key == K_1
    : grid.delay =
    160
if event.key == K_2
    : grid.delay =
    200
if event.key == K_3
    : grid.delay =
    300
if event.key == K_4
    : grid.delay =
    400
if event.key == K_5
    : grid.delay =
    500
if event.key == K_6
    : grid.delay =
    600
if event.key == K_7
    : grid.delay =
    700
if event.key == K_8
    : grid.delay =
    800
if event.key == K_9
    : grid.delay =
    900
if event.key == K_0
    : grid.delay =
    1000
if event.key ==
    K_UP: counter
    +=1;
if event.key ==
    K_DOWN: counter
    -=1.5;
    negativesound.
    play()
if event.key == K_m
    : counter-=4
if event.key == K_p
```

```
                              : st=0
        if mode == False:
                if not paused and not grid.
                    game_over:
                        if event.key == K_d
                            : grid.
                            move_right()
                        if event.key == K_a
                            : grid.move_left
                            ()
                        if event.key == K_s
                            : grid.move_down
                            ()
                        if event.key == K_w
                            : grid.rotate()
                        if event.key ==
                            K_SPACE: grid.
                            drop_down()
                        if event.key ==
                            K_RETURN: grid.
                            drop_down()
                        if event.key == K_e
                            : mode = True
                        if event.key == K_1
                            : grid.delay =
                            160
                        if event.key == K_2
                            : grid.delay =
                            200
                        if event.key == K_3
                            : grid.delay =
                            300
                        if event.key == K_4
                            : grid.delay =
                            400
                        if event.key == K_5
                            : grid.delay =
                            500
                        if event.key == K_6
                            : grid.delay =
                            600
                        if event.key == K_7
                            : grid.delay =
                            700
                        if event.key == K_8
```

```
                                                    : grid.delay =
                                                    800
                                                if event.key == K_9
                                                    : grid.delay =
                                                    900
                                                if event.key == K_0
                                                    : grid.delay =
                                                    1000
                                                if event.key ==
                                                    K_UP: counter
                                                    +=1;
                                                if event.key ==
                                                    K_DOWN: counter
                                                    -=1.5;
                                                    negativesound.
                                                    play()
                                                if event.key == K_m
                                                    : counter-=4
                                                if event.key == K_p
                                                    : st=0


        if not paused and not grid.game_over:

                grid.current.draw()      # draw current
                    piece
                grid.next.draw()         # draw the next
                    piece
                if counter < 0:
                        currentcolor = red
                else:
                        currentcolor = green
                if counter >=11: counter = 11
                if counter <=-10: counter = -10
                pygame.draw.rect(screen, black,
                    (370+125,100,200,1000), 0)
                screen.blit(perf, (405+125, 50+100))
                #screen.blit(timeshow, (250+125, 300+100))
                if currentTime - lastTime > 1:
                        countdown-=1
                        if countdown <= 0:
                                countdown = 30
                        lastTime = currentTime
                m, s = divmod(countdown, 60)
                if m >= 1:
```

```
                screen.blit(timeFont.render(str(m)
                   ,1,white), (305+75, 300+100))
                screen.blit(timeFont.render(":",1,
                   white), (313+85, 300+100))
                screen.blit(timeFont.render(str(s)
                   ,1,white), (320+95, 300+100))
        else:
                screen.blit(timeFont.render(str(s)
                   ,1,white), (320+95, 300+100))
        pygame.draw.rect(screen, currentcolor,
           (370+125,230+100,200,-counter*15), 0)
        pygame.display.flip()


# draw game over message
if grid.game_over:
        grid.draw_game_over()
        pygame.display.flip()
        sleep(1)
        counter = 0
        if currentTime - lastTime > 1:
                countdown-=1
                if countdown <= 0:
                        countdown = 30
                lastTime = currentTime
        grid = Grid()
        clock = pygame.time.Clock()
        counter = 0
        paused = False
        time_elapsed = 0
```

TETRIS SCENARIO 3

```
#!/usr/bin/env python

# PyTris (c) Lukasz Grzegorz Maciak
# Licensed under GNU General Public License Version 3

import sys, pygame, random
from pygame.locals import *
from pygame import K_q
import time
from time import sleep

img = pygame.image.load('cat.png')


pygame.mixer.init()
pygame.init()

pygame.mixer.music.load('C:/Users/erikm/Google Drive/My
   Documents/Skoleting/Master/Scenarios/1st Prototype 27
   apr/Experiment/Sound/Sandstorm.mp3')
positivesound = pygame.mixer.Sound('C:/Users/erikm/Google
   Drive/My Documents/Skoleting/Master/Scenarios/1st
   Prototype 27 apr/Experiment/Sound/levelup.wav')
positivesound.set_volume(0.7)
negativesound = pygame.mixer.Sound('C:/Users/erikm/Google
   Drive/My Documents/Skoleting/Master/Scenarios/1st
   Prototype 27 apr/Experiment/Sound/Wrongbuzz.wav')
negativesound.set_volume(0.3)
levelupsound = pygame.mixer.Sound('C:/Users/erikm/Google
   Drive/My Documents/Skoleting/Master/Scenarios/1st
   Prototype 27 apr/Experiment/Sound/Epic_win2.wav')
pygame.mixer.music.play()

# useful constants
size = width, height = 800, 600
lines_per_level = 200           #increas level after x
   lines
delay = 600                     #speed
mode = False                            #for switching
   directions
counter = 0             #feedback bar counter
highscore = 780
highscorecount = 3
```

```
st = 1
bitnumber = 0
lastlinecount = 0


# define colors
black = 0, 0, 0
red = 255, 0, 0
green = 0, 255, 0
blue = 0, 0, 255
white = 255, 255, 255
yellow = 255, 255, 0
purple = 160, 32, 240
cyan = 0, 255, 255
orange = 255, 165, 0
gray =  45, 45, 45

currentcolor = red      #for feedback bar
myfont = pygame.font.Font(None, 24)
timeFont = pygame.font.Font(None, 24)
perf = myfont.render("PERFORMANCE:", 1, white)
myfont2 = pygame.font.Font(None, 14)
timeshow = timeFont.render("TIME:", 1, white)
highscoreFont = timeFont.render("SCORE TO BEAT:",1,white)
myfont3 = pygame.font.Font(None, 35)
myfont4 = pygame.font.Font(None,50)
inthelead1 = myfont4.render("YOU ARE NOW IN THE LEAD!",1,
   green)
inthelead2 = myfont4.render("YOU ARE NOW IN THE LEAD!",1,
   blue)
thirdplace1 = myfont4.render("YOU ARE NOW IN 3rd PLACE!",1,
   green)
thirdplace2 = myfont4.render("YOU ARE NOW IN 3rd PLACE!",1,
   blue)
secondplace1 = myfont4.render("YOU ARE NOW IN 2rd PLACE
   !",1,green)
secondplace2 = myfont4.render("YOU ARE NOW IN 2rd PLACE
   !",1,blue)
startTime = time.time()
lastTime = time.time()
countdown = 300

block_size = 15         # each piece is 4 blocks
block_gap = 1           # distance between blocks
```

```
offset = block_size + block_gap

twidth = 10 * offset
right_edge = 125+twidth - block_size

theight = 24 * offset
bottom_edge = 100+theight - block_size

start_point = 125+twidth/2 - block_size, 100
next_point = 125+twidth+125, 100+100

# Block object exists mostly to allow us to break a piece
   into individual
# components once it is locked in place. This is why the
   color information
# is redundant

def bitcountfunction(arg):
        arg-=1

class Block(object):
        """ Represents a rectangular Tetris block """

        def __init__(self, x, y, color):

                self.x = x
                self.y = y
                self.color = color

        def draw(self):

                pygame.draw.rect(screen, self.color, self.
                   rect)

        @property
        def rect(self):
                return (self.x, self.y, block_size,
                   block_size)

class Piece(object):
        """ A Tetris Piece - composed of 4 blocks """

        def __init__(self,x,y):

                self.x = x
                self.y = y
                self.mobile = True

                self.rotation = 0 # defines which of the
                   members of self.positions to use for
                   this object

                # set of offsets that is applied to
                   coordinates of each block in this piece
                self.positions = None


        def get_blocks(self):
                """ Returns an array of 4 blocks which make
                    up this piece; each block has it’s own
                   coordinates and draw function """

                blocks = []

                for i in range(4):
                        blocks.append( Block(self.x + self.
                           positions[self.rotation][i][0],
                           self.y + self.positions[self.
                           rotation][i][1], self.color) )

                return blocks

        def draw(self):
                """ Draw all the blocks of this piece to
                   the screen """

                blocks = self.get_blocks()

                for b in blocks:
                        pygame.draw.rect(screen, self.color
                           , b.rect)

        # calling flip repeatedly cycles through all
           available positions and goes back to the first
           one
        def flip(self):
                if(self.rotation < len(self.positions) -1):
                        self.rotation += 1
                else:
                        self.rotation = 0
        def set_point(self, x, y):
                self.x = x
                self.y = y


# Define different piece types: Z, S, O, T, I, L, J

class ZPiece(Piece):

        def __init__(self,x,y):

                super(ZPiece, self).__init__(x, y)

                self.color = red

                self.positions =        [
                                                ( (0,0), (
                                                   offset,
                                                   0), (
                                                   offset,
                                                   offset),
                                                    (2*
                                                   offset,
                                                   offset)
                                                        ),
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (-
                                                   offset,
                                                   offset),
                                                    (-
                                                   offset,
                                                   2*offset
                                                   )    )
                                        ]
class SPiece(Piece):

        def __init__(self,x,y):

                super(SPiece, self).__init__(x, y)

                self.color = green
                self.positions =        [
                                                ( (0,0), (-
                                                   offset,
                                                   0), (-
                                                   offset,
                                                   offset),
                                                    (-2*
                                                   offset,
                                                   offset)
                                                     ),
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (offset
                                                   , offset
                                                   ), (
                                                   offset,
                                                   2*offset
                                                   )      )
                                        ]


class OPiece(Piece):

        def __init__(self, x, y):

                super(OPiece, self).__init__(x,y)

                self.color = yellow

                self.positions =        [
                                                ( (0,0), (
                                                   offset,
                                                   0), (
                                                   offset,
                                                   offset),
                                                    (0,
                                                   offset)
                                                       )
                                        ]


class TPiece(Piece):

        def __init__(self,x,y):
                super(TPiece, self).__init__(x, y)

                self.color = purple

                self.positions =        [
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (-
                                                   offset,
                                                   offset),
                                                    (offset
                                                   , offset
                                                   )
                                                   ),
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (offset
                                                   , offset
                                                   ), (0,
                                                   2*offset
                                                   )   ),
                                                ( (0,0), (-
                                                   offset,
                                                   0), (
                                                   offset,
                                                   0), (0,
                                                   offset)

                                                   ),
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (-
                                                   offset,
                                                   offset),
                                                    (0, 2*
                                                   offset)
                                                    ),
                                        ]


class IPiece(Piece):
        def __init__(self,x,y):

                super(IPiece, self).__init__(x, y)

                self.color = cyan

                self.positions =        [
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (0, 2*
                                                   offset),
                                                    (0, 3*
                                                   offset)
                                                        ),
                                                ( (0,0), (
                                                   offset,
                                                   0), (2*
                                                   offset,
                                                   0), (3*
                                                   offset,
                                                   0)
                                                   )
                                        ]


class LPiece(Piece):

        def __init__(self,x,y):

                super(LPiece, self).__init__(x, y)

                self.color = orange

                self.positions =        [
                                                ( (0,0),
                                                   (0,
                                                   offset),
                                                    (0, 2*
                                                   offset),
                                                    (offset
                                                   , 2*
                                                   offset)
```

```
                                                            ),
                                                        ( (0,0),
                                                           (0,
                                                           offset),
                                                            (offset
                                                           , 0),
                                                           (2*
                                                           offset,
                                                           0)
                                                                  )
                                                           ,
                                                        ( (0,0), (-
                                                           offset,
                                                           0), (0,
                                                           offset),
                                                            (0, 2*
                                                           offset)
                                                                 ),
                                                        ( (0,0),
                                                           (0,
                                                           offset),
                                                            (-
                                                           offset,
                                                           offset),
                                                            (-2*
                                                           offset,
                                                           offset)
                                                              ),
                                                ]

class JPiece(Piece):

        def __init__(self,x,y):

                super(JPiece, self).__init__(x, y)

                self.color = blue

                self.positions =        [
                                                        ( (0,0),
                                                           (0,
                                                           offset),
                                                            (0, 2*
                                                           offset),
                                                            (-
                                                           offset,
                                                           2*offset
                                                           )
                                                                  )
                                                           ,
                                                        ( (0,0), (
                                                           offset,
                                                           0), (2*
                                                           offset,
                                                           0), (2*
                                                           offset,
                                                           offset)

                                                           ),
                                                        ( (0,0), (
                                                           offset,
                                                           0), (0,
                                                           offset),
                                                            (0, 2*
                                                           offset)
                                                                  )
                                                           ,
                                                        ( (0,0),
                                                           (0,
                                                           offset),
                                                            (offset
                                                           , offset
                                                           ), (2*
                                                           offset,
                                                           offset)
                                                               ),
                                                ]



class Grid(object):

        def __init__(self):

                self.current = None
                self.next = None
                self.blocks = []

                self.next_piece()
                self.total_cleared_lines = 0
                self.cleared_lines = 0
                self.level = 1
                self.score = 0

                self.lines_til_next_level = lines_per_level

                self.delay = delay

                #self.bitcount = 0

                self.game_over = False

                self.next_rect = pygame.Rect(150+125,
                   90+100, 300, 300)

                # this is for multiplying scores
                self.multiplier =       {
                                                0 : 0,

                                                           # no
                                                           lines
                                                           cleared
                                                1 : 40,

                                                           # single
                                                            line
                                                           cleared
                                                2 : 100,

                                                           # two
                                                           lines
                                                           cleared
                                                3 : 300,

                                                           # three
                                                           lines
                                                           cleared
                                                4 : 400

                                                           # TETRIS
                                                }

                self.nfont = pygame.font.Font(None, 24)
                self.largefont = pygame.font.Font(None, 40)
```

```
                self.smallfont = pygame.font.Font(None, 14)


        def random_piece(self):
                rekkefolge =
                    [3,5,3,5,4,0,1,5,5,1,1,2,1,2,5,2,3,0,2,2,0,3,6,0,4,5

                pcs =   {
                                0 : ZPiece(*next_point),
                                1 : OPiece(*next_point),
                                2 : SPiece(*next_point),
                                3 : TPiece(*next_point),
                                4 : IPiece(*next_point),
                                5 : LPiece(*next_point),
                                6 : JPiece(*next_point)
                        }

                global bitnumber
                c = rekkefolge[bitnumber]
                bitnumber+=1
                return pcs[c]


        def next_piece(self):

                if not self.next:
                        self.next = self.random_piece()
                        self.current = self.random_piece()
                else:
                        self.current = self.next
                        self.next = self.random_piece()

                self.current.set_point(*start_point)

        def move_down(self):

                if(self.current.y < theight+100):
                        self.current.y += offset

                        if self.has_overlap():
                                self.current.y -= offset
                                self.current.mobile = False
                else:
                        self.current.mobile = False
                if self.current.mobile == False and self.
                   current.y == 100:
                        self.game_over = True

        # this is a hard drop - just go all the way down
           until you hit something
        def drop_down(self):

                while(self.current.mobile): self.move_down
                   ()

        def move_right(self):
                if(self.current.mobile): self.current.x +=
                   offset

                if self.has_overlap(): self.current.x -=
                   offset

        def move_left(self):
                if(self.current.mobile): self.current.x -=
                   offset
                if self.has_overlap(): self.current.x +=
                   offset

        # TODO: wall kick logic needed
        def rotate(self):
                self.current.flip()

                while self.has_overlap(): self.current.flip
                   ()

        # Saves individual blocks of the current piece -
           they become part of the grid
        # Automatically check for lines
        def remember_block_positions(self):

                self.blocks.extend(self.current.get_blocks
                   ())
                self.next_piece()

                self.blocks = sorted(self.blocks, key=
                   lambda block: block.y)

                self.check_for_lines()
                #for b in self.blocks: print str(b.y)+", "

        def draw_blocks(self):

                for b in self.blocks:
                        b.draw()

        # check if current piece overlaps with walls or
           with other pieces
        def has_overlap(self):

                blocks = self.current.get_blocks()

                for b in blocks:
                        if b.x < 125 or b.x > right_edge
                           or b.y < 100 or b.y >
                           bottom_edge: return True

                # TODO: optimize the shit out of this
                for b in self.blocks:
                        for c in self.current.get_blocks():
                                if b.x == c.x and b.y == c.
                                   y: return True

                return False

        # TODO: there ought to be a better way to do this
        def check_for_lines(self):

                lines = {}

                for b in self.blocks:

                        if b.y in lines:
                                lines[b.y] +=1
                        else:
                                lines[b.y] = 1

                lines_to_be_destroyed = []

                for ln in lines:
                        if lines[ln] == 10:
                                self.total_cleared_lines +=
                                    1
                                self.cleared_lines += 1
```

```
                                lines_to_be_destroyed.
                                    append(ln)

                for l in lines_to_be_destroyed:
                        self.destroy_line(l)

                copy_of_blocks = self.blocks[:]

                if len(lines_to_be_destroyed) > 0:
                        self.collapse_hovering_blocks(min(
                            lines_to_be_destroyed), len(
                            lines_to_be_destroyed))

                        #for ln in lines_to_be_destroyed:
                                #self.
                                    collapse_hovering_blocks
                                    (ln, copy_of_blocks)

                self.calculate_score()


        def destroy_line(self, ln):

                # note the slice notation - I'm iterating
                    over a copy of self.blocks but removing
                # from the original
                for b in self.blocks[:]:
                        if b.y == ln: self.blocks.remove(b)



        def collapse_hovering_blocks(self, ln, total):
                """ drop down all the blocks that are
                    hovering """

                for i,b in enumerate(self.blocks):
                        if b.y < ln:
                                self.blocks[i].y += offset
                                    * total



        def block_overlaps(self, block, block_list):
```

```
        for b in block_list:
                if block.y == b.y and block.x == b.
                   x:
                        return True

        return False



def calculate_score(self):

        # score is calculated like so: M * ( N + 1
           ) where:
        # M is multiplier (see self.multiplier)
        # N is level (self.level)

        self.score += self.multiplier[self.
           cleared_lines] * (self.level + 1)

        tmp = self.lines_til_next_level - self.
           cleared_lines

        if tmp <= 0:
                self.level +=1
                self.lines_til_next_level =
                   lines_per_level + tmp

                self.delay -= self.level * 30

                if self.delay < 10: self.delay = 10

        else:
                self.lines_til_next_level = tmp

        self.cleared_lines = 0
        self.draw_text()


# redraws the whole UI - we should really be
   bliting this shit
def draw_ui(self):

        pygame.draw.rect(screen, gray, (125,100,
           right_edge+offset-125, bottom_edge+
           offset-100))
        for i in xrange(10): pygame.draw.line(
            screen, black, (i*offset+125, 100),
            (125+i*offset, theight+offset+100))
        for i in xrange(24): pygame.draw.line(
            screen, black, (125, i*offset+100), (
            twidth+125, i*offset+100))

        self.draw_text()


# I really dislike the default font in pygame
def draw_text(self):

        msg = self.nfont.render("NEXT PIECE:", 1,
            white)
        screen.blit(msg, (250+125, 50+100))

        pygame.draw.rect(screen, black, (250+125,
            200+100, 200, 100))

        sc = self.nfont.render("YOUR SCORE: ", 1,
            white)
        screen.blit(sc, (240+125, 250+100))
        sc = self.largefont.render(str(self.score)
            ,1,white)
        screen.blit(sc, (270+125, 270+100))

        #sc = self.nfont.render("LINES: " + str(
            self.total_cleared_lines), 1, white)
        #screen.blit(sc, (250, 280))

        #sc = self.nfont.render("LEVEL: " + str(
            self.level), 1, white)
        #screen.blit(sc, (250, 260))



        #msg = self.smallfont.render("Left, Right
            Arrow to move", 1, white)
        #screen.blit(msg, (230, 290))

        #msg = self.smallfont.render("Up Arrow to
            flip", 1, white)
        #screen.blit(msg, (230, 300))
```

```
                #msg = self.smallfont.render("Down Arrow to
                    move downw", 1, white)
                #screen.blit(msg, (230, 310))

                #msg = self.smallfont.render("Space or
                   Enter to drop down", 1, white)
                #screen.blit(msg, (230, 320))

                #msg = self.smallfont.render("Esc to pause,
                    F1 for new game", 1, white)
                #screen.blit(msg, (230, 330))

                #screen.blit(self.smallfont.render("ver
                   0.2", 1, white), (230, 350))

        def draw_game_over(self):

                msg = self.largefont.render("GAME OVER", 1,
                    red)
                screen.blit(msg, (195+125,95+100))

                #msg2 = self.nfont.render("PRESS F1 TO PLAY
                    AGAIN", 1, red)
                #screen.blit(msg2, (175, 130))




# magic
screen = pygame.display.set_mode(size)
pygame.key.set_repeat(100, 150)

#pygame.time.set_timer(USEREVENT+1, delay)

grid = Grid()
clock = pygame.time.Clock()

paused = False

time_elapsed = 0

while st == 1:
        currentTime = time.time()
        if grid.total_cleared_lines>lastlinecount:
```

```
        positivesound.play()
        lastlinecount=grid.total_cleared_lines
        counter+=1
    if not paused and not grid.game_over:

        time_elapsed += clock.tick()

        #print str(grid.delay)

        # timed block drop
        if time_elapsed > grid.delay:
            time_elapsed = 0
            grid.move_down()

        # check if a line was created and remove it
        grid.check_for_lines()

        # clear the next piece area
        pygame.draw.rect(screen, black, grid.
          next_rect)

        # draw the gridlines
        grid.draw_ui()

        # if the current piece is locked in place
          memorize it’s position
        if not grid.current.mobile:
            grid.remember_block_positions()

        # draw the memorized blocks
        grid.draw_blocks()


    for event in pygame.event.get():

        if event.type == pygame.QUIT: sys.exit()


        if event.type == KEYDOWN:

            if event.key == K_ESCAPE: paused =
              not paused

            if event.key == K_F1:
                grid = Grid()
```

```
                    clock = pygame.time.Clock()
                    counter = 0
                    paused = False
                    time_elapsed = 0

                if mode == True:
                    if not paused and not grid.
                        game_over:
                            if event.key == K_d
                                : grid.move_left
                                ()
                            if event.key == K_a
                                : grid.
                                move_right()
                            if event.key == K_s
                                : grid.move_down
                                ()
                            if event.key == K_w
                                : grid.rotate()
                            if event.key ==
                                K_SPACE: grid.
                                drop_down()
                            if event.key ==
                                K_RETURN: grid.
                                drop_down()
                            if event.key == K_q
                                : mode = False
                            if event.key == K_1
                                : grid.delay =
                                100
                            if event.key == K_2
                                : grid.delay =
                                200
                            if event.key == K_3
                                : grid.delay =
                                300
                            if event.key == K_4
                                : grid.delay =
                                400
                            if event.key == K_5
                                : grid.delay =
                                500
                            if event.key == K_6
                                : grid.delay =
                                600
```

```
                                        if event.key == K_7
                                            : grid.delay =
                                            700
                                        if event.key == K_8
                                            : grid.delay =
                                            800
                                        if event.key == K_9
                                            : grid.delay =
                                            900
                                        if event.key == K_0
                                            : grid.delay =
                                            1000
                                        if event.key ==
                                            K_UP: counter
                                            +=1;
                                            positivesound.
                                            play()
                                        if event.key ==
                                            K_DOWN: counter
                                            -=1.5;
                                        if event.key == K_m
                                            : counter-=4
                                        if event.key == K_p
                                            : st=0
                        if mode == False:
                                if not paused and not grid.
                                    game_over:
                                        if event.key == K_d
                                            : grid.
                                            move_right()
                                        if event.key == K_a
                                            : grid.move_left
                                            ()
                                        if event.key == K_s
                                            : grid.move_down
                                            ()
                                        if event.key == K_w
                                            : grid.rotate()
                                        if event.key ==
                                            K_SPACE: grid.
                                            drop_down()
                                        if event.key ==
                                            K_RETURN: grid.
                                            drop_down()
                                        if event.key == K_e
```

```
    : mode = True
if event.key == K_1
    : grid.delay =
    100
if event.key == K_2
    : grid.delay =
    200
if event.key == K_3
    : grid.delay =
    300
if event.key == K_4
    : grid.delay =
    400
if event.key == K_5
    : grid.delay =
    500
if event.key == K_6
    : grid.delay =
    600
if event.key == K_7
    : grid.delay =
    700
if event.key == K_8
    : grid.delay =
    800
if event.key == K_9
    : grid.delay =
    900
if event.key == K_0
    : grid.delay =
    1000
if event.key ==
    K_UP: counter
    +=1;
    positivesound.
    play()
if event.key ==
    K_DOWN: counter
    -=1.5;
if event.key == K_m
    : counter-=4
if event.key == K_p
    : st=0
```

```
        if not paused and not grid.game_over:
                grid.current.draw()      # draw current
                   piece
                grid.next.draw()         # draw the next
                   piece
                if counter < 0:
                        currentcolor = red
                else:
                        currentcolor = green
                if counter >=11: counter = 11
                if counter <=-10: counter = -10
                pygame.draw.rect(screen, black,
                   (370+125,100,200,1000), 0)
                screen.blit(perf, (405+125, 100+50))
                #screen.blit(timeshow, (250, 300))
                if grid.score >= highscore and
                   highscorecount >=0:
                        highscore+=630
                        grid.delay -=100
                        highscorecount-=1
                        paused = not paused
                        pygame.mixer.music.set_volume(0.1)
                        levelupsound.play()
                        if highscorecount==2:
                                pygame.draw.rect(screen,
                                   black, (0,0,800,600), 0)
                                screen.blit(thirdplace1,
                                   (150, 200))
                                pygame.display.flip()
                                sleep(0.2)
                                pygame.draw.rect(screen,
                                   black, (0,0,800,600), 0)
                                screen.blit(thirdplace2,
                                   (150, 200))
                                pygame.display.flip()
                                sleep(0.2)
                                pygame.draw.rect(screen,
                                   black, (0,0,800,600), 0)
                                screen.blit(thirdplace1,
                                   (150, 200))
                                pygame.display.flip()
                                sleep(0.2)
                                pygame.draw.rect(screen,
                                   black, (0,0,800,600), 0)
                                screen.blit(thirdplace2,
                        (150, 200))
                    pygame.display.flip()
                    sleep(0.2)
                    pygame.draw.rect(screen,
                        black, (0,0,800,600), 0)
                    screen.blit(thirdplace1,
                        (150, 200))
                    pygame.display.flip()
                    sleep(1)
                    pygame.draw.rect(screen,
                        black, (0,0,800,600), 0)
                    pygame.display.flip()
                if highscorecount==1:
                    pygame.draw.rect(screen,
                        black, (0,0,800,600), 0)
                    screen.blit(secondplace1,
                        (150, 200))
                    pygame.display.flip()
                    sleep(0.2)
                    pygame.draw.rect(screen,
                        black, (0,0,800,600), 0)
                    screen.blit(secondplace2,
                        (150, 200))
                    pygame.display.flip()
                    sleep(0.2)
                    pygame.draw.rect(screen,
                        black, (0,0,800,600), 0)
                    screen.blit(secondplace1,
                        (150, 200))
                    pygame.display.flip()
                    sleep(0.2)
                    pygame.draw.rect(screen,
                        black, (0,0,800,600), 0)
                    screen.blit(secondplace2,
                        (150, 200))
                    pygame.display.flip()
                    sleep(0.2)
                    pygame.draw.rect(screen,
                        black, (0,0,800,600), 0)
                    screen.blit(secondplace1,
                        (150, 200))
                    pygame.display.flip()
                    sleep(1)
                    pygame.draw.rect(screen,
                        black, (0,0,800,600), 0)
                    pygame.display.flip()
            if highscorecount==0:
                    pygame.draw.rect(screen,
                       black, (0,0,800,600), 0)
                    screen.blit(inthelead1,
                       (150, 200))
                    pygame.display.flip()
                    sleep(0.2)
                    pygame.draw.rect(screen,
                       black, (0,0,800,600), 0)
                    screen.blit(inthelead2,
                       (150, 200))
                    pygame.display.flip()
                    sleep(0.2)
                    pygame.draw.rect(screen,
                       black, (0,0,800,600), 0)
                    screen.blit(inthelead1,
                       (150, 200))
                    pygame.display.flip()
                    sleep(0.2)
                    pygame.draw.rect(screen,
                       black, (0,0,800,600), 0)
                    screen.blit(inthelead2,
                       (150, 200))
                    pygame.display.flip()
                    sleep(0.2)
                    pygame.draw.rect(screen,
                       black, (0,0,800,600), 0)
                    screen.blit(inthelead1,
                       (150, 200))
                    pygame.display.flip()
                    sleep(1)
                    pygame.draw.rect(screen,
                       black, (0,0,800,600), 0)
                    pygame.display.flip()

            pygame.mixer.music.set_volume(1)
            paused = not paused
    if highscorecount == 3:
            screen.blit(timeFont.render("3rd
               place score:",1,white),
               (235+125,100+180))
            screen.blit(myfont3.render(str(
               highscore),1,white),
               (270+125,100+200))
```

```
elif highscorecount == 2:
        screen.blit(timeFont.render("2nd
           place score:",1,white),
           (235+125,100+180))
        screen.blit(myfont3.render(str(
           highscore),1,white),
           (270+125,100+200))
elif highscorecount == 1:
        screen.blit(timeFont.render("1st
           place score:",1,white),
           (235+125,100+180))
        screen.blit(myfont3.render(str(
           highscore),1,white),
           (270+125,100+200))
else:
        screen.blit(myfont3.render("YOU ARE
            NOW",1,white),
           (200+125,100+170))
        screen.blit(myfont3.render("IN THE
           LEAD!",1,white),
           (200+125,200+100))
#else:
#               screen.blit(highscoreFont,
   (220,180))
#               screen.blit(myfont3.render(
   str(highscore),1,white), (270,200))
if currentTime - lastTime > 1:
        countdown-=1
        if countdown <= 0:
                countdown = 0
        lastTime = currentTime
m, s = divmod(countdown, 60)
#if m >= 1:
#       screen.blit(timeFont.render(str(m)
   ,1,white), (305, 300))
#       screen.blit(timeFont.render(":",1,
   white), (313, 300))
#       screen.blit(timeFont.render(str(s)
   ,1,white), (320, 300))
#else:
#       screen.blit(timeFont.render(str(s)
   ,1,white), (320, 300))
pygame.draw.rect(screen, currentcolor,
   (400+125,100+230,150,-counter*15), 0)
#screen.blit(img,(180,250))
```

```
                pygame.display.flip()

        # draw game over message
        if grid.game_over:
                #grid.draw_game_over()
                counter = 0
                highscorecount=3
                highscore = 1410
                if currentTime - lastTime > 1:
                        countdown-=1
                        if countdown <= 0:
                                countdown = 0
                        lastTime = currentTime
                #pygame.display.flip()
                grid = Grid()
                clock = pygame.time.Clock()
                counter = 0
                paused = False
                time_elapsed = 0
```

Appendix G - Risk Assessment

NTNU	Kartlegging av risikofylt aktivitet	Utarbeidet av	Nummer	Dato
HMS		HMS-avd.	HMSRV2601	22.03.2011
		Godkjent av		Erstatter
		Rektor		01.12.2006

Enhet: **Institutt for Maskinteknikk og Produksjon** **Dato:** **05.05.2017**

Linjeleder: **Torgeir Welo**

Deltakere ved kartleggingen (m/ funksjon)**:** Martin Steinert, veileder/Jørgen A. B. Erichsen, Coach/Andreas Wulvik, Coach/ Erik A. Borge, student

Kort beskrivelse av hovedaktivitet/hovedprosess: Masteroppgave Erik A. Borge. Experimentally piloting and testing wether we can measure valence through physiological reactions

Er oppgaven rent teoretisk? (JA/NEI)**:** NEI

Signaturer: *Ansvarlig veileder:* *Martin Steinert* [illegible] *Studenter:* *Erik A. Borge* [illegible]

ID nr.	Aktivitet/prosess	Ansvarlig	Eksisterende dokumentasjon	Eksisterende sikringstiltak	Lov, forskrift o.l.	Kommentar
1	Bruk av Trolllabs workshop.	EB	Romkort	Romkort		
1a	Bruk av roterende maskineri	EB	Maskinens brukermanual	Ukjent	Ukjent	
1b	Bruk av laserkutter	EB	Maskinens brukermanual	Ukjent	Ukjent	
1c	Bruk av 3D printer	EB	Maskinens brukermanual	Ukjent	Ukjent	
1d	Bruk av skjæreverktøy	EB	Ukjent			

1e	Bruk av samenføynigsmidler (lim og lignende.)	EB	Produktets brukermanual og datablad	Datablad	Ukjent	
2	Tilstedeværelse ved arbeid utført av andre.	Andre	Andres HMSRV2601	Andres HMSRV2601	Prosessavhengig	
3	Eksperimentelt arbeid	EB	Risikovurdering		Prosessavhengig	

Enhet: **Dato:**

Linjeleder:

Deltakere ved kartleggingen (m/ funksjon): Martin Steinert, veileder/Jørgen A. B. Erichsen, Coach/Andreas Wulvik, Coach/
Erik A. Borge, student

Risikovurderingen gjelder hovedaktivitet: Masteroppgave Erik A. Borge.
Experimentally piloting and testing wether we can measure valence through physiological reactions

Signaturer: *Ansvarlig veileder:* [illegible] *Student:* [illegible]

ID nr	**Aktivitet fra kartleggings-skjemaet**	**Mulig uønsket hendelse/ belastning**	**Vurdering av sannsyn-lighet**	**Vurdering av konsekvens:**				**Risiko-Verdi (menn-eske)**	**Kommentarer/status Forslag til tiltak**
			(1-5)	Menneske (A-E)	Ytre miljø (A-E)	Øk/ materiell (A-E)	Om-dømme (A-E)		
1	Bruk av Trolllabs workshop.								Vær nøye med opplæring i bruk av maskineri.
1a-i	Bruk av roterende maskineri	Stor kuttskade	2	D	A	A	D	2D	Sørg for at roterende deler er tilstrekkelig sikret/dekket.
1a-ii		Liten kuttskade	3	B	A	A	A	3B	Ikke ha løse klær/tilbehør på kroppen.
1a-iii		Klemskade	2	D	A	A	C	2D	Ikke ha løse klær/tilbehør på kroppen.
1a-iv		Flygende spon/gjenstander	3	C	A	A	B	3C	Bruk øyevern og tildekk hurtig roterende deler (Fres og lignende.)
1a-v		Feil bruk - ødelagt utstyr	3	A	A	C	A	3C	Opplæring.

1b-i	Bruk av laserkutter	Klemskade	2	D	A	A	C	2D	Ikke ha løse klær/tilbehør på kroppen.
1b-ii		Brannskade	3	B	A	A	A	3B	Bruk hansker ved håndtering av varme materialer.
1b-iii		Øyeskade-laser	2	D	A	A	C	2D	Bruk øyevern! Skru av laser når maskinen ved oppsett.
1b-iv		Brann	2	B	A	D	C	2B	Ha slukkeutstur tilgjengelig.
1c-i	Bruk av 3D-printer	Brannskade	3	B	A	A	A	3B	Vær oppmerksom.
1c-ii		Innhalering av plast/ printemateriale	5	A	A	A	A	5A	Bruk åndedrettsvern/ vernebriller.
1c-iii		Feil bruk - ødelagt maskineri	3	A	A	C	A	3A	Opplæring.
1d-i	Bruk av skjæreverktøy	Stor kuttskade	2	D	A	A	D	2D	Bruk skarpe verktøy og riktig skjæreunderlag.
1d-ii		Liten kuttskade	3	B	A	A	A	3B	Bruk skarpe verktøy og riktig skjæreunderlag.
1e-i	Bruk av samenføynigsmidler (lim og lignende.)	Eksponering på øyet	2	D	A	A	B	2D	Bruk øyevern, ha datablad tilgjengelig.

1e-ii		Eksponering hud	4	A	A	A	A	4A	Bruk hansker, ha datablad tilgjengelig.
1e-iii		Eksponering åndedrett	4	A	A	A	A	4A	Bruk åndedretsvært/ god ventilasjon. Ha datablad tilgjengelig.
1e-iv		Søl	4	A	B	A	A	4A	Ha papir/ rengjøringsmateriell tilgjengelig. Ha datablad tilgjengelig.
2	Tilstedeværelse ved arbeid utført av andre.	Se andres risikovurdering om sikkerhet betviles.	3	C	C	C	C	3C	Hold et øye med hva som foregår rundt deg.
3-i	Eksperimentelt arbeid	Skade ved fall e.l.	2	A	A	A	A	2A	Sikre eksperimentelt utstyr. Førstehjelps-kit tilgjengelig.
3-ii		Anfall grunnet mye impuls	2	B	A	A	A	2B	I forkant sikre at testsubjekter er rustet til det som skal gjøres i eksperimentet. Førstehjelps-kit tilgjengelig.
3-iii		Skade ved bruk av sensorikk i nærheten av mennesker	2	B	A	A	A	2B	Sørge for at sensorikk brukes på forsvalig vis og lese datablad på sensorene.

Sannsynlighet vurderes etter følgende kriterier:

Svært liten 1	Liten 2	Middels 3	Stor 4	Svært stor 5
1 gang pr 50 år eller sjeldnere	1 gang pr 10 år eller sjeldnere	1 gang pr år eller sjeldnere	1 gang pr måned eller sjeldnere	Skjer ukentlig

Konsekvens vurderes etter følgende kriterier:

Gradering	Menneske	Ytre miljø Vann, jord og luft	Øk/materiell	Omdømme
E Svært Alvorlig	Død	Svært langvarig og ikke reversibel skade	Drifts- eller aktivitetsstans >1 år.	Troverdighet og respekt betydelig og varig svekket
D Alvorlig	Alvorlig personskade. Mulig uførhet.	Langvarig skade. Lang restitusjonstid	Driftsstans > ½ år Aktivitetsstans i opp til 1 år	Troverdighet og respekt betydelig svekket
C Moderat	Alvorlig personskade.	Mindre skade og lang restitusjonstid	Drifts- eller aktivitetsstans < 1 mnd	Troverdighet og respekt svekket
B Liten	Skade som krever medisinsk behandling	Mindre skade og kort restitusjonstid	Drifts- eller aktivitetsstans < 1uke	Negativ påvirkning på troverdighet og respekt
A Svært liten	Skade som krever førstehjelp	Ubetydelig skade og kort restitusjonstid	Drifts- eller aktivitetsstans < 1dag	Liten påvirkning på troverdighet og respekt

Risikoverdi = Sannsynlighet x Konsekvens
Beregn risikoverdi for Menneske. Enheten vurderer selv om de i tillegg vil beregne risikoverdi for Ytre miljø, Økonomi/materiell og Omdømme. I så fall beregnes disse hver for seg.

Til kolonnen "Kommentarer/status, forslag til forebyggende og korrigerende tiltak":
Tiltak kan påvirke både sannsynlighet og konsekvens. Prioriter tiltak som kan forhindre at hendelsen inntreffer, dvs. sannsynlighetsreduserende tiltak foran skjerpet beredskap, dvs. konsekvensreduserende tiltak.

NTNU	Risikomatrise	utarbeidet av	Nummer	Dato
		HMS-avd.	HMSRV2604	08.03.2010
		godkjent av		Erstatter
HMS/KS		Rektor		09.02.2010

MATRISE FOR RISIKOVURDERINGER ved NTNU

KONSEKVENS	Svært alvorlig	🟨 E1	🟥 E2	🟥 E3	🟥 E4	🟥 E5
	Alvorlig	🟨 D1	🟨 D2	🟥 D3	🟥 D4	🟥 D5
	Moderat	🟨 C1	🟨 C2	🟨 C3	🟥 C4	🟥 C5
	Liten	🟩 B1	🟩 B2	🟨 B3	🟨 B4	🟥 B5
	Svært liten	🟩 A1	🟩 A2	🟩 A3	🟨 A4	🟨 A5
		Svært liten	Liten	Middels	Stor	Svært stor
		SANNSYNLIGHET				

Prinsipp over akseptkriterium. Forklaring av fargene som er brukt i risikomatrisen.

Farge		Beskrivelse
Rød	🟥	Uakseptabel risiko. Tiltak skal gjennomføres for å redusere risikoen.
Gul	🟨	Vurderingsområde. Tiltak skal vurderes.
Grønn	🟩	Akseptabel risiko. Tiltak kan vurderes ut fra andre hensyn.

Appendix J - Participant Consent Form

NTNU

Fakultet for ingeniørvitenskap og teknologi
Institutt for Maskinteknikk og Produksjon

Vår dato	Vår referanse
11.05.17	MS
Deres dato	Deres referanse

Request for Participation in Research Project

Warning
If you are epileptic, please make us aware of this.

Background and Purpose
The purpose of this project is to study the change of physiological reactions due to changes in affect. This experiment is part of a MSc at MTP, Norwegian University of Science and Technology.

What does participation in the project imply?
The participant will be asked to conduct a select number of rounds in a computer game, and data from these will be stored. After being introduced to the task, the participant will be guided through the experiment. The experiment is comprised of three parts; Part I, Part II and Part III. The participant will be asked to fill out several surveys as part of the experiment. A video recording of the participant will be made. The participant will be presented with true and erroneous information.

What will happen to the information about you?
All personal data will be treated anonymously. No name is connected to the gathered data. The only persons having access to the data are the two master students and their supervisor. In case of a publication, participants will therefore not be recognizable by name. The project is scheduled for completion by 01.07.2017. After this date the personal data will be stored encrypted.

Voluntary participation
The participation of this experiment is voluntary, and you can at any time choose to stop and withdraw from the experiment. If you would like to participate or if you have any questions concerning the project, please contact Erik Borge (+47 95222034) or Helge Garsmark (+47 94898375).

Consent for participation in the study
I have received information about the project and am willing to participate. I agree that data is collected, analyzed and published anonymously. I further agree to be confidential about the experiment to provide non-biased conditions for every participant.

Name of the participant (*Please use capital letters*)   e-mail *(for award purposes)*

Signature (*Please include Place & Date*)

Appendix K - Participant Demographics Form

NTNU

Fakultet for ingeniørvitenskap og teknologi
Institutt for produktutvikling og materialer

Vår dato	Vår referanse
11.05.17	HSG
Deres dato	Deres referanse

Background Information Questionnaire

This questionnaire is designed to collect additional background information about you.

Part A. Education

The questions in this section are designed to collect information on your education.

A1. What is your current level of achieved education?

- ☐ High School
- ☐ Bachelor's Degree
- ☐ Master's Degree
- ☐ Ph.D.

A2. When did you graduate?

Month ☐☐ Year ☐☐☐☐

A3. Please record your primary area of specialization.

Primary Area of Specialization: ______________________________

A4. Are you currently studying for a degree? If no, skip to part B. If yes, please specify:

- ☐ High School
- ☐ Bachelor's Degree
- ☐ Master's Degree
- ☐ Ph.D.

A5. When do you plan to graduate?

Month ☐☐ Year ☐☐☐☐

A6. Please record your primary area of specialization.

Primary Area of Specialization: ______________________________

Part B: Demographic Information

The questions in this section are designed to collect some of your demographic information.

B1. Are you:

☐ Male
☐ Female

B2. In what year were you born?

Year of Birth: ☐☐☐☐

B3. What is your nationality (i.e. citizenship)?
Please specify if you have multiple citizenships.

Answer: ____

Part B: Specifics

The questions in this section are designed to collect some of your demographic information.

C1. How many cups of coffee did you drink today?

☐ None
☐ 1
☐ 2 or 3
☐ 4 or more

C2. Are you feeling sick today?

☐ Yes
☐ No

C3a. Do you have any chronic illness? If no skip question C3b.

☐ Yes
☐ No

C3b. Is the illness affecting your emotion?

☐ Yes
☐ No

Helge Soltvedt Garsmark, Master student
Department of Mechanical and Industrial Engineering(MTP), Norwegian University of Science and Technology (NTNU)
Richard Birkelandsvei 2B, NO-7491 Trondheim, Norway.
Phone: +47 94 89 83 75, helgesg@stud.ntnu.no

Part D: Further Participation

D1. Are you willing to receive follow-up questions or surveys of this study via e-mail in the future? If yes, please write your e-mail address below.

E-mail address: ____

Part E: General Information

You have just participated in an experiment on emotional state evaluation, containing elements of stimuli and physiological measurements.

As stimuli, you were presented with three tasks with different difficulty, in addition you were stimulated with different visuals and audio during the three tasks. The audio stimuli was both performance based and in general connected to the different tasks. You provided us with three different types of emotional feedback during this experiment. Also physiological data was collected from you through EMG, ECG, a distance sensor and the chair. The goal of this experiment is to provide qualitative data on the correlation between emotional state and physiological measurements, and further how a possible correlation can be used in affective research.

We wish to remind you to be confidential about the content of this experiment to provide non-biased conditions for every participant, as stated in the consent form. We hope you enjoyed participating, and thank you kindly for your commitment of time to this experiment!

Thank you for your time and participation!