NTNU
Norwegian University of
Science and Technology

Play-Create-Share Gameplatform for the iPad

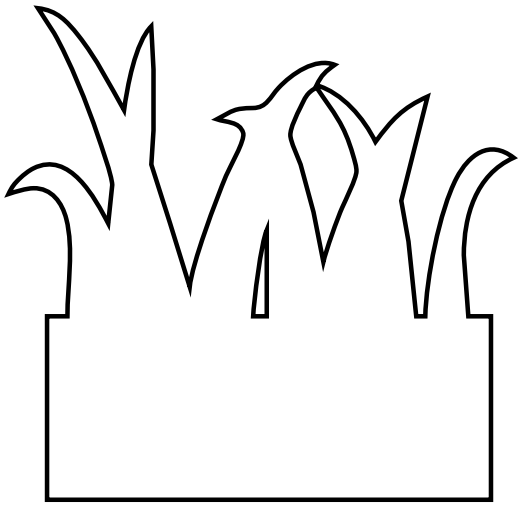
Exploring Game Creation Technique on a Touch Screen Platform

Fredrik Ludvigsen

Master of Science in Computer Science
Submission date: June 2011
Supervisor: Alf Inge Wang, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

TDT4900 - Computer and Information Science, Master Thesis Spring 2011



Play-create-share gameplatform for the iPad

by

Fredrik Ludvigsen
Advisor: Alf Inge Wang

NTNU – Trondheim
Norwegian University of
Science and Technology

Abstract

This report presents the implementation and analysis of a simple create-and-play game for the *iPad*. The game is a platformer with a level editor, utilizing the *iPad*'s touchscreen and acceleration sensor. We believe the results are also applicable to games, that have level editors, for other types of post-PC tablet devices (*Samsung Galaxy Tab*, *Motorola Xoom* etc.).

The analysis comprise a usability test with a questionnaire. 33 university students and 4 children attended the test. The attendants were given several opportunities to provide feedback as free text. The free text answers have also been categorized and summarized.

The level editor provided a low entropy building scheme which was popular among the test population. Multiple mechanisms were provided for scrolling inside levels and controlling the game character. The population was divided in which mechanisms they learned to use, and which they preferred. We believe that this indicates that they did all contribute to a good game and editor experience.

We also provide some general advice about GUI appearance on touch based devices. Many of our test attendants were dissatisfied with how well they could understand the meaning of the menu icons. Some attendants suggested that on-screen text should be used more extensively, but we do not conclude whether an improved set of icons, more text, a demonstration or information mode or some other, yet unknown strategy is the best way to handle the issue.

Preface

This report is one of the main products of my master thesis in computer science at NTNU during spring 2011. I chose this project because I wanted to get some real experience developing games, and the *iPad* seemed like an appropriate platform for a small game. The *iPad* also provided me with an opportunity to learn a new programming language and an API that is much wanted in the software development business.

In the end I had not learned a lot about the *iOS* API's, but a lot about casual games, indie games and game editors in general. The usability tests and the following analysis was an eye-opener to me. I learned much about the design and creation of self-explaining interfaces. The truth is that there is no magic bullet solving this problem, but there is hard work, valuable experience and a never ending improvement process. If software were buildings, games would be the kindergartens; you never know what to expect and there is no recipe for fun.

I would like to thank Alf Inge Wang for being my advisor during this project. He helped with directing the project, searching for relevant technical material, providing frequent feedback underway and reviewing the report, among other things. He even conducted a few of the user tests, voluntarily, during his own spare time. I would also like to thank all my test subjects and everyone else who helped me during the project.

Fredrik Ludvigsen

Contents

Chapter 1

Introduction

1.1 Motivation

Since *Android* and *iPhone* smartphones started hitting the market in 2008 and 2007 there has been a growing interest in games and other software for touchscreen based devices. These devices cannot use traditional PC-based software for many reasons.

- They typically run on battery power.
- They have small screens.
- They typically have no keyboard or mouse.

Tablet computers has been available since long before *Android* and *iPhone* were introduced, but recently a new category has been introduced. Devices in the new category uses smartphone operative systems like *Android* and *iOS*. They are distinctively different from the previous tablets because they lack a pen-like input device. Older operative systems relied on the precise input of a mouse and were relatively hard to navigate only by touching the screen. Throughout this report we will refer to the new tablet category as post-PC tablets.

The goal of this project is to explore techniques that are suitable for implementing a game with an editor on a post-PC tablet.

1.2 Software for mobile operative systems

Dependency on a keyboard and mouse is typical for almost all software that has been developed for PCs. Some software also uses the CPU extensively. Application candidates that could benefit from a transition to smartphones and post-PC tablets should:

- use little CPU power
- function properly without large screen interfaces
- function properly without a keyboard
- function properly without right-click and mouse scrolling

Benefits from a transition to mobile devices includes:

- long battery life
- small size
- positioning technology
- cameras
- microphones
- multi-touch interfaces
- simple widespread market platforms

The post-PC tablet game market is dominated by casual games. *Apple* recognized the state of their *App Store* and appropriately described it as a gold rush. But gold rushes end, and as they do, only the most adaptive gold diggers will remain—adapting appropriately to the platform will be key to the post-PC tablet application market.

High quality applications like *Garageband* and *iMovie* demonstrates the convenience of creating music and movies on *iPad*. This project will explore options directed towards creating games on *iPad*, but with the limited resources of a single person for implementation and testing. Games like *Second Life*, *Little Big Planet* and *ModRacer* for *Playstation 3* has shown that harvesting the creative power of users can result in games that renew themselves, prolonging their lives and giving them the time needed to become rich on content and loved by their users (Sanjeev et al., 2008).

Chapter 2

Research questions and method

2.1 Task description

> Play-create-share is a new trend for video games, seen in games such as Little Big Planet and ModRacer for Playstation 3. In this project, the goal is to develop a game with a game level editor for the *iPad*. The project should utilize the *iPad*'s touch screen to provide a new type of editor, encouraging the user to create levels. The project aim for the development of a concrete game that has been designed by the supervisor.

2.2 Level sharing platforms

Jonathan Blow has shown examples of just how complex game development is (Blow, 2004). His article acted as a warning that high levels of complexity might be one of the biggest challenges of any game creation project. By keeping the number of components low, we might have reduced the complexity exponentially. This is true since complexity emerges between the components. On average, for each component added, there is an increased chance that new interactions between components are needed.

Since there was only a single person working on the project, no sharing platform was developed for the game, but we made the game with such a platform in mind, so that it would be easy to add in the future. We believe that sharing platforms are important for viral marketing simply because they

create an incentive to spread the word.

2.3 Research questions

The research questions in this project are tied to the combination of play-create-share game editors and touchscreen interfaces. In the following list we will abbreviate the concept of game level editors on post-PC tablets to *ml-editors*, as in mobile level editors.

RQ1 How should developers address the lack of pointing devices like mice and pens in ml-editors?

RQ2 How should developers address the lack of traditional game controllers in ml-editors?

RQ3 How should developers address the need to edit levels that are bigger than the screen in ml-editors?

RQ4 To what extent can ml-editors support creative processes?

For RQ1-3 a few different proposals have been implemented and tested. The test results could provide a few implications based on well the test attendants handled simple tasks.

Through the test process, some examples of creative content (RQ4) have been collected. These do not give any definite answers, but they might suggest that the concept is worth looking into for other creative applications.

2.4 Scope of the project

This project was conducted as a part of video game research programme at NTNU, which started in 2008 (nor, 2007-2008). The only contributor to the implementation code, except the libraries mentioned in Section 4.4 and a few hundred lines of sample code, were the members of this project.

The scope included the development of a new type of game and an assessment of its properties. It was important that the game would be fun to play, so we focused on creating a small, but high quality game. By keeping the game

small, we left room for adjustments that could be needed in order to create an entertaining experience.

The following issues were relevant to the development:

- Game and editor mechanics in post-PC tablet games and editors

Relevant issues that did not receive substantial attention were:

- Aesthetics in post-PC tablet games
- Sound usage in post-PC tablet games
- Performance in post-PC tablet games
- Direct comparison of PC and post-PC tablet games

The game was tested by 37 people in order to assess the implications of the choices that had been made during development. The test attendants were given questions about their gaming habits, their test performance and any advice they could be able to provide.

2.5 Focus of the project

RQ1 and RQ3 were core questions that could be relevant for any type of level editor on post-PC tablet platforms, while RQ2 was merely addressing games that would have been played with joysticks and similar controllers on other platforms.

The creation of new game levels is a creative process. By assessing what kind of creative processes can take place between the level editor and the user, we would also assess why anyone would use it in the first place (RQ4). The only non-creative processes that could take place are the copying of other levels (completely directed) and entirely random productions (no direction). In both cases, the user will not have made anything that came from his or her own imagination.

The properties of mobile platforms, listed in Section 1.2, suggest that casual games is a suitable genre for mobile platforms. This is also confirmed by the market. Casual games like *Angry Birds*, *Tiny Wings* and *Cut the Rope* have

been among the most successful games at the *Android Market* and *Apple App Store*.

(Taefay, 2010) lists four common elements that inform the design of casual games:

- Rules and goal must be clear.
- Player need to be able to quickly reach proficiency.
- Casual game play adapts to a player's life and schedule.
- Game concepts borrow familiar content and themes from life.

Reaching proficiency stand out because it is in direct conflict with the need to provide the player with powerful tools that enable creation of beautiful and challenging game levels.

Out of necessity, a lot of effort also went into learning the Objective-c programming language, the *cocos2d* game API and the *Box2D* physics API.

2.6 Research method

Basili has identified three common research approaches for software engineering experiments (Basili, 1993):

The engineering method : By using the engineering experimental method, engineers build and test a system according to a hypothesis. Based upon the result of the test, they improve the solution until it requires no further improvement. The engineering method is typically used to find better methods for structuring large systems, and software engineering is here viewed as a creative task not to be controlled by anything else than necessary restrictions on the resulting product.

The empirical method : A statistical method is proposed as a means to validate a given hypothesis. Unlike the analytical method, there may not be a formal model or theory describing the hypothesis. Data is collected to verify or falsify the hypothesis. The empirical method can be applied on new technology to determine if this new technology is better or worse than the existing for producing software effectively.

The mathematical method : The mathematical method is based on mathematical and formal methods for doing experiments. A formal theory is developed and results derived from that theory can be compared with empirical observations. The mathematical method is usually used to find better formal methods and languages, where software development is viewed as a mathematical transformation process.

The engineering method was used, but only for a single iteration. Here is what was done:

1. Literature study
2. Implementation of a platform game with a level editor for the *iPad*
3. A usability test with test subjects from the university. This included a modified version of the System Usability Scale (Lewis and Sauro, 2009).
4. Analysis of test results

Success criteria for this project relate to the creation of what has been described in Section 2.1 as "a new type of editor" for the *iPad*. Success in this project would call for an application that fits within any game definition and provides an editor that could be used to create central parts of the game content. Egenfeldt Nielsen, Heide Smith and Pajares Tosca have provided a discussion of what is required for something to be classified as a game, but no absolute answer is given, despite extensive efforts carried through by experts in the field (Nielsen et al., 2008). Whether or not the game is successful in its mission, that is entertaining whoever uses it, is not a criteria, as anyone could argue that a game is not entertaining.

Chapter 3

Prestudy - Games with editors

3.1 Related Research

In the specific area of games with editors on post-PC platforms we were unable to find any research material. There is, however, lots of material available about create-play-share games, fan based game modifications, game creation theory and other related concepts. In this chapter we will present some of them.

Second life is a popular MMO that provides what is called a metaverse, which are "fully immersive virtual spaces that significantly differ from online games in several ways". In *Second Life*, sharing is a part of the performance issue (Sanjeev et al., 2008). The amount of user created content provides a challenge for the implementation of the server. All of this content is collected on the server and appears to players that are nearby. The content can also remain for a long time, which means that large amounts of content can accumulate.

It has previously been shown how preference models can be used to create personalized game levels (Shaker et al.). These levels aim to optimize the experience based on questionnaires administered to players after playing different levels. While these kinds of levels are automatically created, the same strategy might relate to user guidance in more classic editors.

It has previously been suggested that platform games are harder to produce by randomization strategies than RPG and strategy games (Compton and Mateas, 2006). An algorithm for building platform patterns has been built and tested. The algorithm takes platform jump distance and other kinds of

difficulties into consideration, utilizing the rhythmic properties of platformer games. Rhythmic actions help the player reach states of higher concentration, called "flow" (Sweetser and Wyeth, 2005).

The legal issues of intellectual property rights to user created content have been looked into in (Humphreys et al., 2005). A case study of *Auran* and their 3D train and railroad simulator *Trainz* was carried through. The article emphasized the importance of the question of which extent to which commercial entities can acknowledge fan based ownership as a port of the 'drift of value' from producer to consumer. There is a balance between how much freedom is given to the users of an editor and how contractual restructuring is needed by the developer.

David Buckingham and Andrew Burn have discussed the need for a better theoretical game literacy framework in (Buckingham and Burn, 2007). One argument is the need for a better public understanding of games in order to create and use them for educational purposes. It is emphasized that in order to teach through games, we should first teach about games, "If you want to use television to teach somebody, you must first teach them how to use television". This relates to games with editors because non-creative playing might relate to reading the same way game level creation relate to writing.

Magy Seif El-Nasr and Brian K. Smith have presented experiences from attempts to teach students programming, mathematics, physics and aesthetic principles by modifying existing games (El-Nasr and Smith, 2006). The authors observed that different game engines implicitly stress the use and development of certain skills, making the choosing of game engines an important issue for the educational values. This represents a small step towards content creation, and not only game "consumption". More importantly for this project, it demonstrates the great skill requirements of some powerful game content editors.

3.2 The *iPad* platform

The *iPad* post-PC tablet was released on April 3, 2010. It is a 0.7 kg heavy computer with a 242.8 mm by 189.7 mm touchscreen with multitouch support. It is designed to be used with bared fingers, so non-conductive gloves or stylus pens does not work. It uses light sensors to automatically adjust the screen brightness to its surroundings. Figure 3.1 shows Steve Jobs holding an *iPad*.

Figure 3.1: Steve Jobs holding an *iPad*

There is no intrinsic native orientation of the *iPad*, it responds to an acceleration sensor by turning the interface so that it is in an upright position, unless overridden by the current application.

Apple claims that the *iPad* has 10 hours of battery when playing video. This is one of the properties that makes it more mobile than laptops.

Without modification it will only run software that either is approved by *Apple* and distributed through the *Apple App Store* or runs in a web browser.

It supports 802.11a/b/g/n and *Bluetooth 2.1* connectivity. The processor is a 1GHz *Apple A4* custom-designed, high-performance, low-power system-on-a-chip with 256 MB DDR RAM (App, 2009).

3.3 Real-time Strategy Games

Many Real-time Strategy (RTS) games have level editors. Some examples are *Anno*, *Age of Empires*, *Warcraft*, *Starcraft* and even *Settlers*.

Many RTS editors are powerful tools, including features like event control and the ability to create whole series of levels (El-Nasr and Smith, 2006). *Starcraft* and *Age of Empires II* have level editors that are independent programs, derived from the tools that were used by the publishers to create some of the original game content. Figure 3.2 shows the interface of the level editor which was included in the RTS game *Starcraft*.

The games work well on their own, but the editors extend their area of application. For instance, a creative player can prepare a map to share with his friends on a LAN party.

3.4 Simulators

The genre of simulator games includes a wide range of games. Some include level editors, but others do not. Some simulators even have level creation or other kinds of content creation as their main objectives. The first game in the *The Sims* series could go on as long as the user wanted to. The user decided when the creation had been completed or when it had been lost.

SimCity and *The Sims* allows the player to pause the game at any moment to edit the game world. Figure 3.3 shows *The Sims* when the simulation is

Figure 3.2: The level editor in *Starcraft* (*Blizzard*)

Figure 3.3: *The Sims*

paused and building mode is active. Building material costs money, so the player always have to make sure there is a stable source of income available. This way the editing becomes more of an economic sustainability challenge, than a purely artistic challenge.

The Sims and *SimCity* are examples of games that focus on the creation of non-reusable content. The content they create does not have any use other than to continue building from a previous point in time or just displaying the creation. Strict definitions of level editors might even exclude this kind of games because there is no difference between playing and level editing.

3.5 Platformers

Platformers that have level editors is a small, but diverse game category. The diversity might be linked to high level of entropy and a small set of possibly successful levels that the traditional platformers provide (Compton and Mateas, 2006).

Figure 3.4: *Little Big Planet*

Limbo is a game that is based on traditional 2D platformer concepts. Its mind puzzling levels has brought the genre further, even without making any big changes to it. This is an indication that platformer editors are too hard for amateurs to use. Another indication is presented in (Compton and Mateas, 2006). *Limbo* came out in 2010, 25 years after *Super Mario Bros.*, but could still innovate, using the same concept.

Little Big Planet (*LBP*) and *Minecraft* demonstrate the diversity of the genre. *LBP* has a separate edit mode where the player is able to reverse time and undo creations freely. *Minecraft* has only one mode, where the player makes the creations while playing the game. *LBP* is a 2D-physics based game while *Minecraft* is based on discretized, approximate 3D-physics. Both games, however, rely on moving the character around while editing. Figure 3.4 shows an image of the quasi-2d world of *Little Big Planet*.

Platformer editors are not as common as the real-time strategy level editors and simulator editors, but a few games have become very popular. The *Minecraft* project was started by a single developer, but became so popular that he could expand the project even before the beta testing started. The game had grown large and profitable even with no money spent on advertising (Froholt, 2010). Figure 3.5 shows the blocky, discretized world of *Minecraft*.

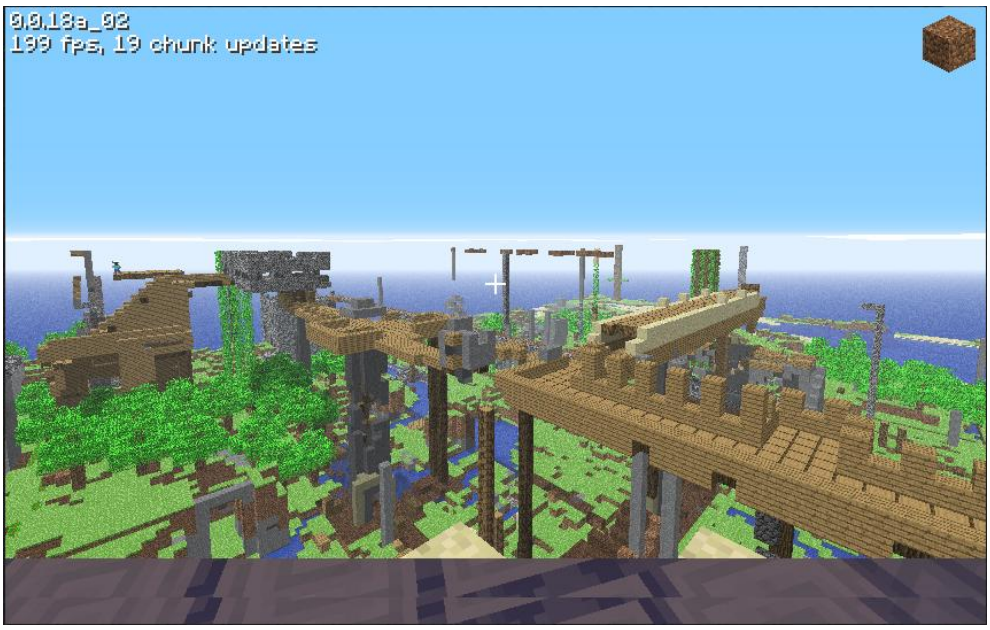


Figure 3.5: *Minecraft*

3.6 Puzzle Games

Puzzle games is a wide category for games that require careful thinking.

Crayon Physics is a puzzle game that allows the player to create shapes that will act physically within the game. This is the mechanism used both to solve puzzles and to create them. The editor lets the user create puzzles while physics and switches are turned off.

To create a shape in *Crayon Physics*, the user must draw the shape with the mouse cursor. This mechanism requires very little effort per created shape, and enable the player to build quickly. This is important because the player sometimes have to build while the game is running.

The Incredible Machine (*TIM*) is similar to crayon physics, but is based on a discrete grid and a fixed set of objects that have complex interactions. Puzzles in *TIM* are solved while physics and other game object interactions are halted, but with a fixed set of objects, set by the level creator. The Figures 3.6-3.7 show *TIM* and *Crayon Physics*.

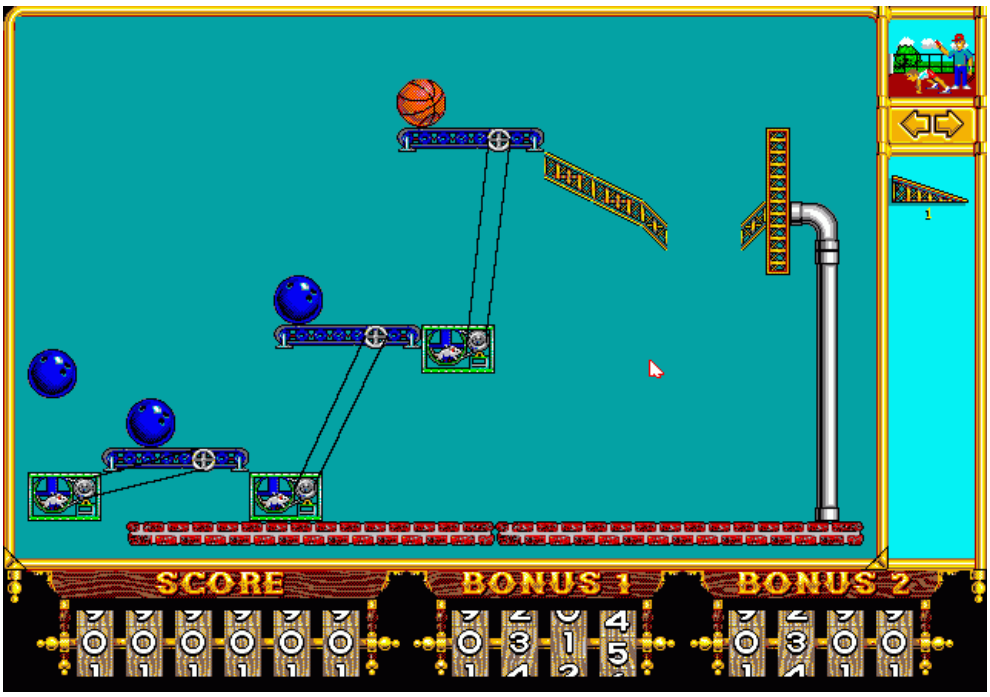


Figure 3.6: *The Incredible Machine*

3.7 Social

The genre of social games is, not surprisingly, the dominating game genre in the social network of *Facebook*. Social games often rely on viral marketing. Intrinsic high quality is a property that can help the viral marketing effect of any game (Sauro, 2011). For games with editors, this also applies to the creations of the user. A player who is excited about his own creation might be one of the best ambassadors for the game.

For viral marketing to work well, the players should be able to share their creations with other players (Taefay, 2010). Games like *Farmville*, *Café World* and *What To Wear* use sharing intentionally to increase the amount of players. The Figures 3.8-3.9 show images from *What To Wear* and *Farmville*.

Some of these games are strictly in the casual genre; they have more in common with trophy shelves, chess clocks and paper based role-playing than with *SimCity*, *Starcraft* and *The Incredible Machine*, but they are very popular games based on the creativity of players.

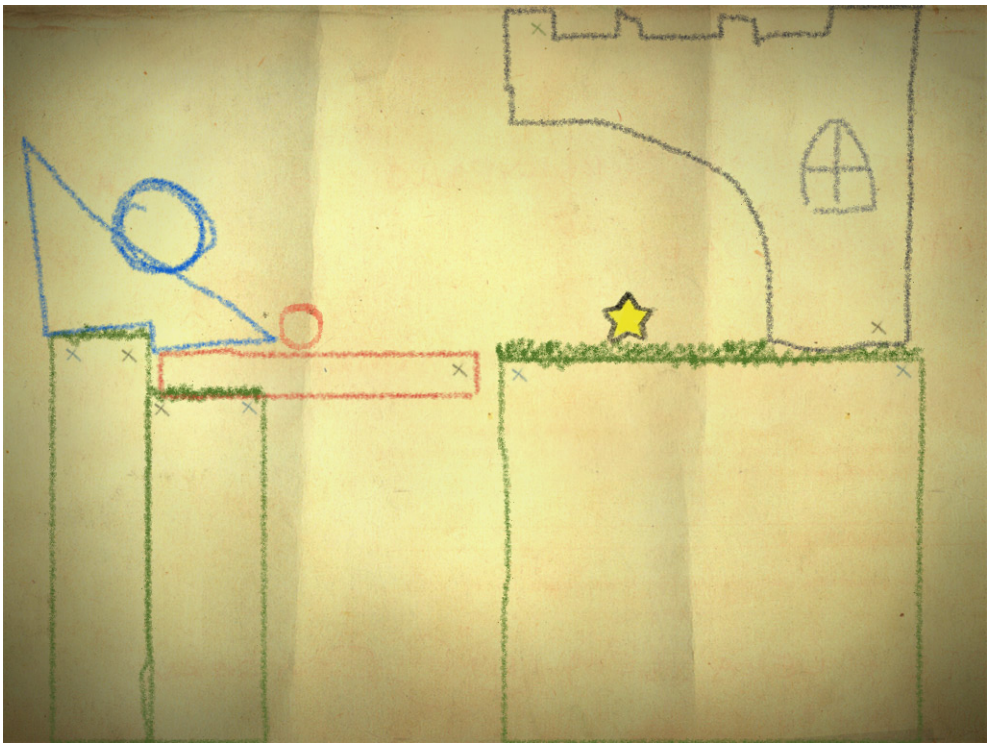


Figure 3.7: *Crayon Physics*

Figure 3.8: *What To Wear*

Figure 3.9: *Farmville*

3.8 Racing

Few racing games include track editors, but there are some notable exceptions. *Trackmania* might be the best known racing game with a track building system, but it was not the first. *Stunts*, which came out in 1990, included loops, jumps and a track editor.

ModNation Racers represented a shift toward consoles. The editor had changed a lot, in order to be used with game controllers, but the ground principles were the same. The Figures 3.10-3.11 shows *Stunts* and *Mondation Racers*.

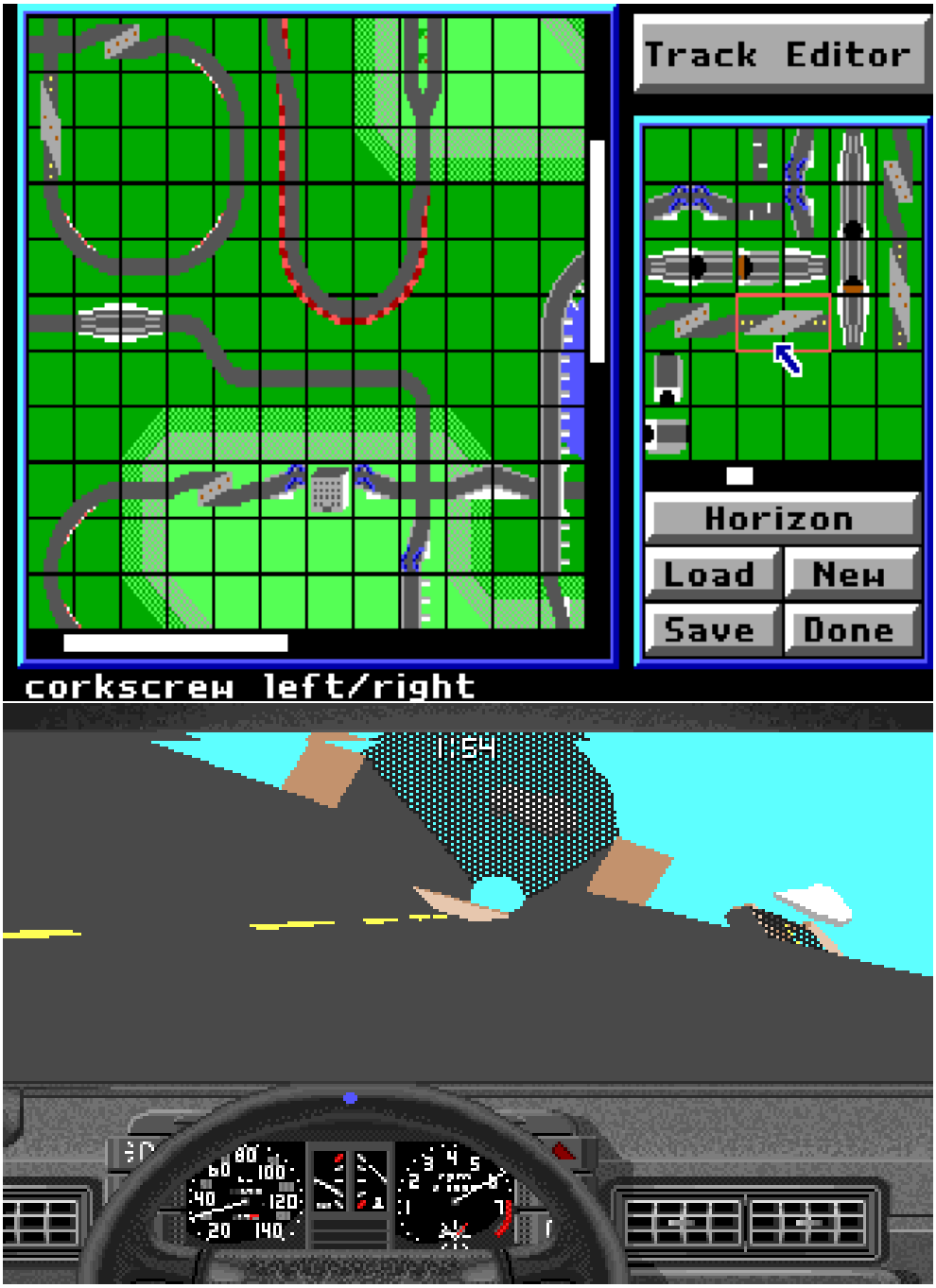


Figure 3.10: *Stunts*

Figure 3.11: *ModNation Racers*

Chapter 4

Own contribution

4.1 Initial System requirements

The initial system requirements of the system were derived from the task description in Section 2.1 as following:

Pre-R1 It should be a game.

Pre-R2 It should be a platformer.

Pre-R3 It should include a level editor.

Pre-R4 It should produce content suitable for a sharing platform.

The concept of a game is hard to define. Some scientists say games should be fun, some say they should be challenging and the list goes on with empirical and subjective properties (Nielsen et al., 2008).

Instead of facing any definition of a game directly, we let the test subjects decide. They were told that they were going to test a game. Whether or not the software was a game could, by our definition, be decided by the users. Although this requirement is vague it is important, because it has implications on many of the properties of software.

We focused on making the game nice to look at, nice to listen to, fun and easy to use (Pre-R1).

One proposed definition of platformer games is, "A 3D or side-scrolling 2D game in which a player must jump over/into/on objects/platforms in order

to complete the game", (Answerbag, 2003). In other words, the game was required to have platforms that the player could jump on and fall off of (Pre-R2). This implicated the existence of a character that the user control.

We chose to implement the level editor as a separate game mode (Pre-R3). It allowed the user to save the level at any time, and play it in play mode.

To enable easy sharing, we chose to save the levels in separate files (Pre-R4). There were no sharing platform created, but if one was built after the project, it would only have to handle one level per file.

Following is a summary of the refined system requirements:

R1 The system should satisfy the users' definitions of a game.

R2-a The game should have platforms that the player can jump on and fall off.

R2-b The game should have a character that the user controls.

R3 The game should have at least two separate modes. At least one mode should allow level modifications and at least one should not.

R4 Every game level should be stored in a separate file.

4.2 Mid Project System Requirements

We used the 2D-physics library *Box2D* for crash detection and gravity rendering. With this library, it was be easy to implement obstacles that behaved in a natural way. We decided to add some game objects that the character could interact with, as if crashing with them or pushing them.

We also wanted to use some classic platformer game mechanics, like points, a time constraint and start and end positions.

The following requirements were added during the project:

R5 The editor should allow the user to add objects that behave naturally and which the user can interact with.

R6 The editor should allow the user to add objects that the character can pick up.

R7 There should be a timing mechanism that is active while playing the levels.

R8 There should be a start point and an end point that apply when playing the levels.

4.3 Design & architecture

Implementation strategy

The combination of the *cocos2d* and *Box2D* libraries is common for *iPhone* game development and it is easy to see why (Zyn, 2011), (Eri, 2011). *cocos2d* is a 2D-graphics library. It does not contain any specific game backbone, but a lot of tools common to 2D games. Menus, background music, sprite rendering and display rotation were concepts that were easy to implement using *cocos2d*. *Box2D* complemented this with a 2D physics simulation. It did not render anything on the display, but it provided interfaces that were great for implementing it.

Box2D supported 2D shape queries. This means it could be used to find out whether there was any object present at any given point in the 2D space. *Box2D* would answer any space query with references to *Box2D* objects, which we would need to identify and associate with any texture that we wanted to use. As a convenience, *Box2D* objects contained an untyped reference that could be used for any purpose. We used this reference to refer objects of our own class, that bound *cocos2d* objects, *Box2D* objects and application specific data together. Listing 4.1 shows the base class that was inherited by all game objects.

Listing 4.1: Base class of game objects

```
@interface GameObject : NSObject
{
        b2Body* body;       // Box2D specific
        CCSprite* sprite;   // cocos2d specific
        CGSize bodySize;    // Application specific
}
```

Grid based blocks

The grid based blocks were used in order to reduce the entropy of user created content. This was one way of reducing, the set of choices that the user had to make.

When two grid based blocks lined up, they were supposed to behave no different from a single block covering the same area. By replacing all lined up blocks with an equivalent set of larger blocks, both the rendering algorithms and the physics simulations would have fewer bodies to work with.

The problem of merging many blocks into as few bigger blocks as possible is an NP-hard problem (Lingas, 1982). Solving this problem was not necessary, but doing so could have increased the performance of the application after the merging was complete.

Pausing the application while building could degrade the experience of the user, for instance by not responding appropriately to finger swipes. Swiping with one or more fingers should produce series of blocks.

In order to increase the minimum performance of the editor we applied two block transformation restrictions.

- Blocks should only split when parts are being removed explicitly by the user.
- Blocks should only merge after new blocks have been created explicitly by the user.

With this restriction, the computational cost off adding a block would never exceed O(n), where n is the number of existing blocks. Also the computational cost of filling the interior of some empty area, would not exceed O(n), where n is the number of added blocks.

GUI design ideas

Scrolling

In the first implementation of the game, the levels had unlimited space in three directions. The fourth was a limited by an infinitely wide floor object. It became apparent that the user would have to view a little bit at a time,

and we wanted this to feel natural. We chose to test three different mechanisms for sliding the viewing perspective horizontally, vertically or diagonally. Figure 4.13 shows what the mechanisms looked like.

Character control

We chose to use classic control mechanisms, known from *Super Mario Bros.*, to control the game character, but the *iPad* did not have built-in game controllers or general purpose hardware buttons. We needed to control the horizontal movement and jumping for the character. We decided to use an emulated joystick for horizontal movements and the rest of the screen surface, except for the menu area, for jumping. We also decided to allow the user to control the horizontal movement by tilting the *iPad*. This was convenient, since an acceleration sensor is built into every *iPad*. Figure 4.12 shows the game with the device tilting control enabled.

Menu appearance

The menu was built to be independent of language. It was based on glyphs that the users were supposed to recognize the meaning of. There was, however, a single line with information in English at the top of the screen that could help the user understand how to use every mode. For example the information bar would display "Delete/Load/Save/Exit?" when the file menu was active; Figure 4.7 illustrates this.

4.4 Implementation

Development process

The implementation of the game was a 4 month continuous development process with the goal of producing a game as described in Section 2.1 with a few modifications. The development method was incremental and based on a minimal sample implementation of an *iPhone* game. The sample was a demonstration of how to create *cocos2d + Box2D* based project for *XCode*, the official *iOS* development environment, provided by *Apple*.

The system was tested and demonstrated every week and new goals were defined based on the demonstration.

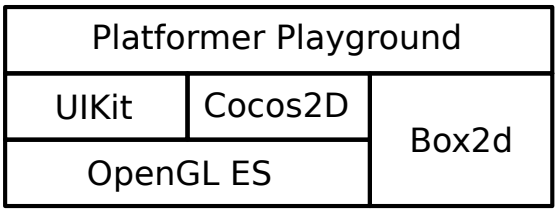


Figure 4.1: Library stack

Objective-C++ implementation

Figure 4.1 shows a simplified overview of the libraries used in the project. *Platformer Playground* was the name of our game. *Box2D* is an open source physics simulation library for 2D physics. It is written in C++, but there also exists ports for other languages. The difference in the implementation of *Box2D* and *cocos2d* meant that the project would have to include both C++ code and Objective-C code.

UIKit is the official UI library for *iOS* that has been provided by *Apple*.

cocos2d and *UIKit* contained some features that were similar. The *UIKit* provided a lot of widgets that were familiar to *iOS* users, like the UIScrollView and UIKeyboard, but it was not comprehensive enough for performance-intensive games. *cocos2d* and *UIKit* used different coordinate system handling routines, which forced us to provide our own implementation, just for synchronizing them. For instance, the UIScrollView had to be handled carefully by our code when the *cocos2d* library was performing interface rotation in response to rotation of the *iPad* device.

Figure 4.2 shows an overview our scene classes. The *CCScene* class, provided by *cocos2d*, was used to implement the top level, mutually exclusive game modes.

MenuScene was a small class that handled the menu layout on its own. It contained a text field and inherited the *UITextFieldDelegate* in order to communicate with the *UIKeyboard*. The *PlayScene* and *BuildScene* classes left the details to other, more specialized classes, like *GameWorld* and *Hud*.

The *Hud* and *PlayModeHud* classes implemented the custom GUI widgets in the editor and in the play mode. These are shown in Figure 4.3.

Figure 4.4 presents the *GameWorld* class. The *GameWorld* singleton was the owner of the game objects, which are presented in Figure 4.5. The

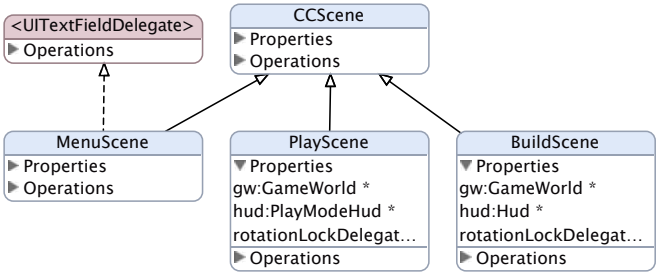


Figure 4.2: Scenes

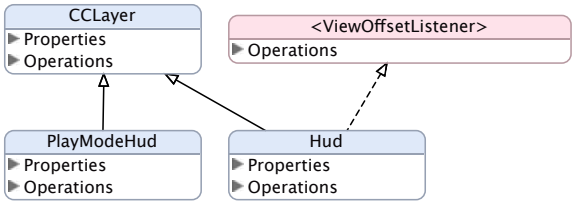


Figure 4.3: Widget overlay classes

GameWorld object handled adding and removing objects using functions and objects from *cocos2d*, *Box2D* and a few custom classes, which have not been described in detail in this report.

Platformer Playground - The game

Figure 4.6 shows the main menu of *Platformer Playground*. Selecting the hammer symbol would cause a list of stored levels and a text field with the placeholder *'<New playground>'* to appear. Entering any level name followed by *'return'* would add a new level to the list. Tapping one of the level names in the list would open the editor and start editing the selected level.

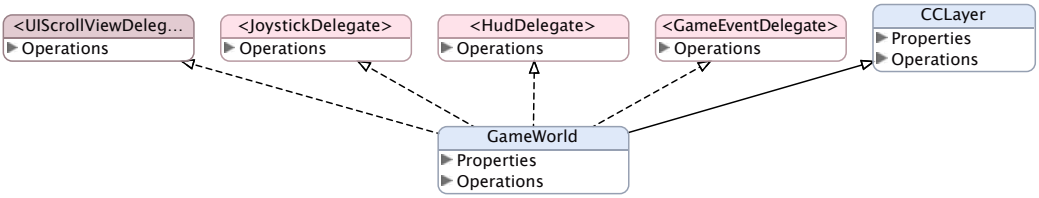


Figure 4.4: Game object controlling class

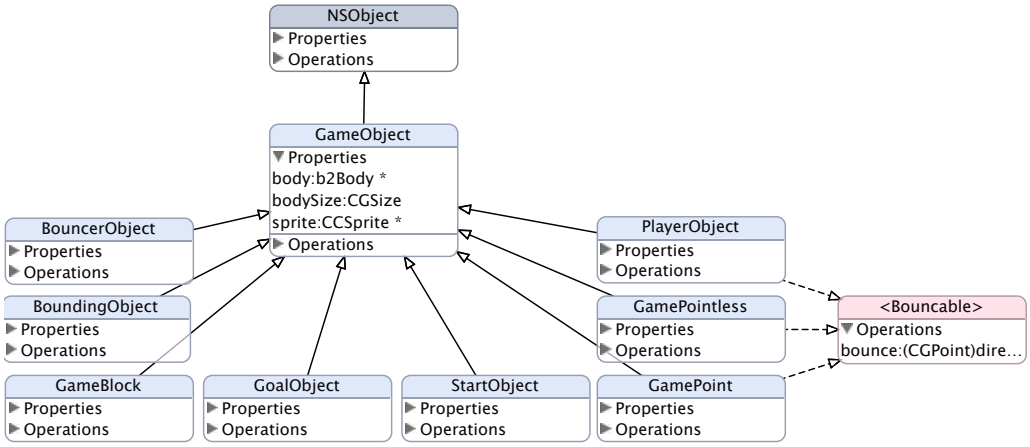


Figure 4.5: Game Object classes

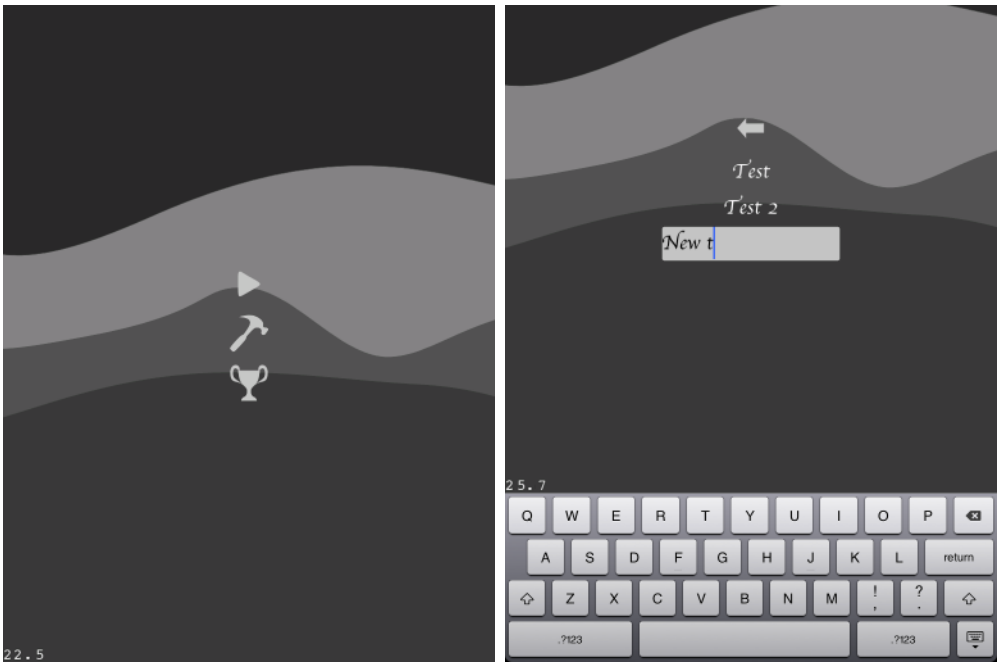


Figure 4.6: The main menu and the build level menu.

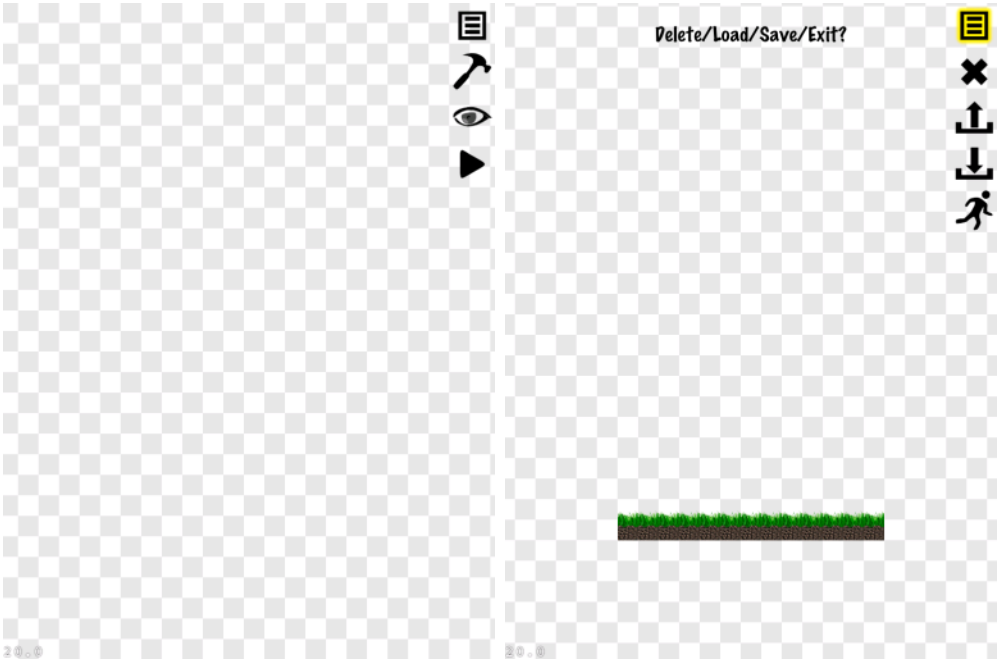


Figure 4.7: The editor when idle and when the file menu is activated.

The editor of *Platformer Playground* had a combined menu and toolbox that was based on states. Figure 4.6 illustrates following states:

- *'edit//'*
- *'edit//file'*

Figure 4.8 shows a complete tree of all states in the game. While using the editor, there was only one menu available, residing in the upper right corner. This menu changed itself depending on what state the game was in. For instance it would only show options relevant to level modification while the *'/edit//build'* mode was active.

In order to add or remove content, the user had to activate the building mode. Each placeable item had its own icon in the menu when build mode was active. The glowing icon at the top of the menu showed which editor mode that was active; tapping it would deactivate that mode and activate *'edit//'*. Figure 4.9 shows the build mode.

The grass blocks were static building blocks. They behaved like infinitely hard, floating objects with infinite mass.

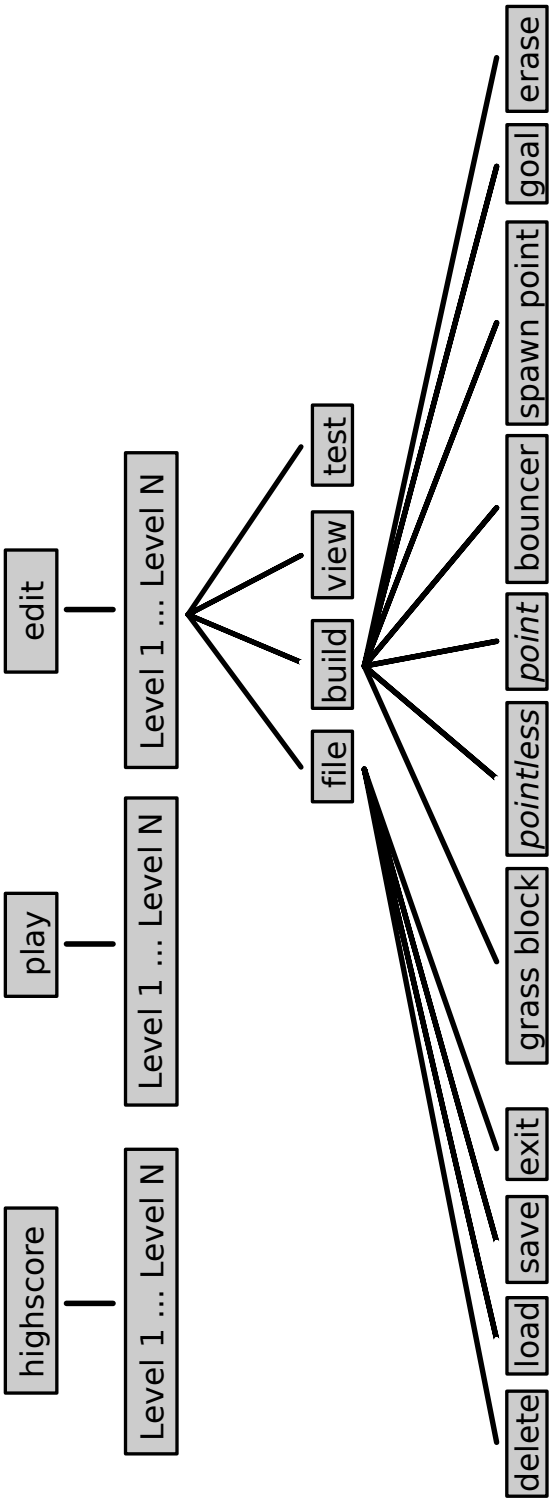


Figure 4.8: The states of *Platformer Playground*

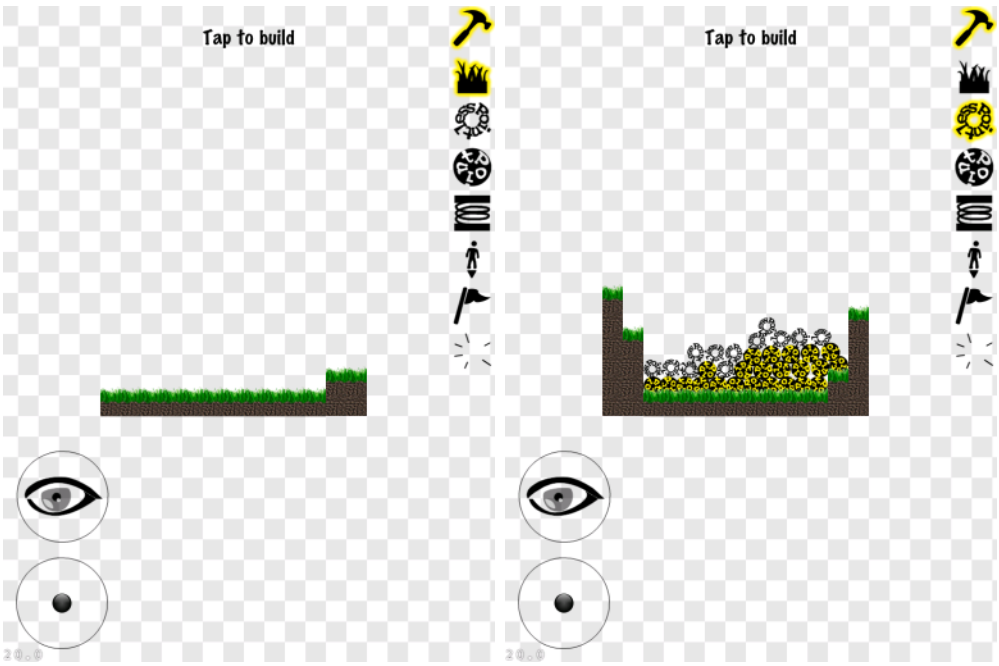


Figure 4.9: Placing grass blocks, *pointless* and *point*

The *point* and *pointless* marbles were affected by physics simulation at all times. Placing a marble in mid-air would cause it to fall directly down once it appeared, even if the player was neither in play mode or test mode.

Point marbles could be picked up by the player, either in test mode or in play mode. *Pointless* marbles looked a little different, and could not be picked up by the player, but were in every other way similar to *point* marbles.

Figure 4.10 shows the springs. The springs were static, grid based objects. Springs affected marbles and the player by giving it an impulse pointing straight up. The character got a greater impulse. The idea was that the object would bounce as much as possible, but not making it hard too for the player to play with. Because the rabbit character should be controlled by the player, giving it a powerful impulse would not confuse the player too much. Different impulse strengths were not tested after the initial implementation.

Figure 4.11 shows a level with a spawning point and a goal flag. These were not active during testing. When entering test mode, there were no character present, but it could be placed directly by a finger tap from the player. The thought is that the player should not have play through the whole level every time a part of it should be tested.

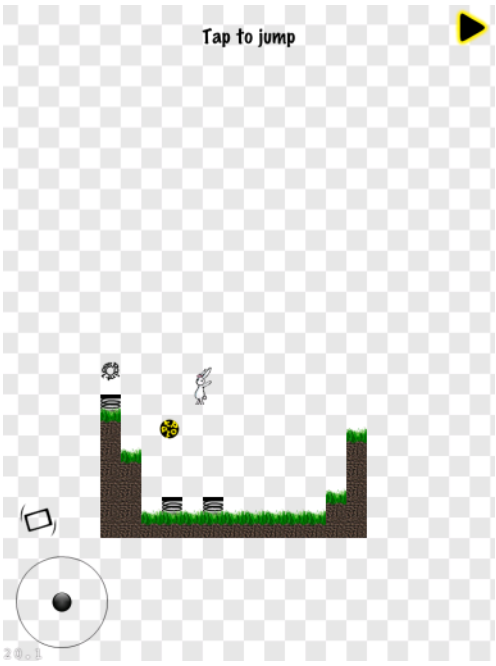


Figure 4.10: Springs behave like trampolines, but with a fixed amount of bouncing force.

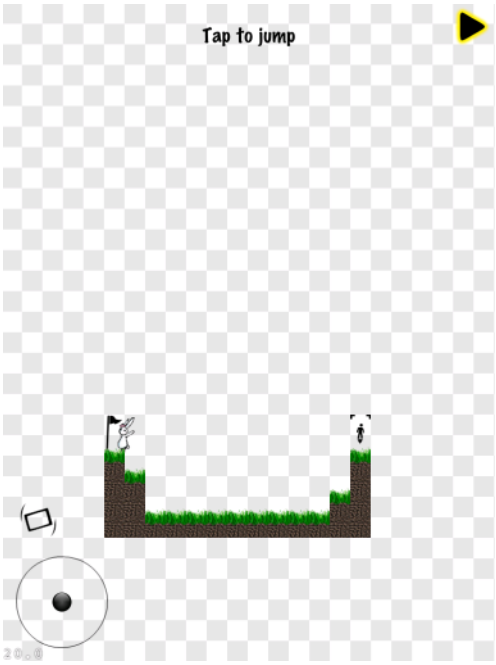


Figure 4.11: Start and end points.

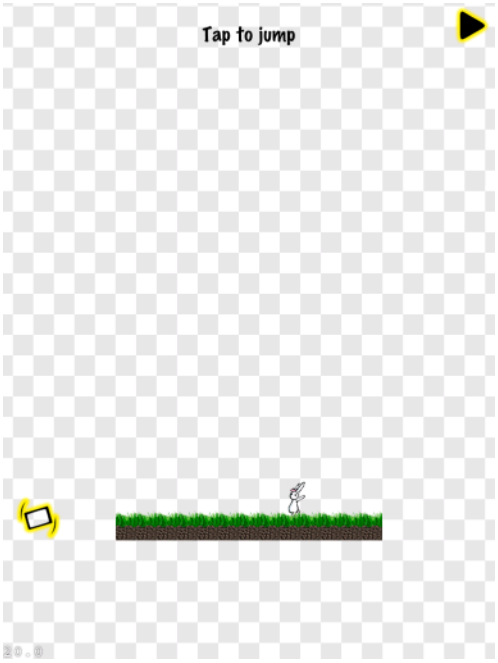


Figure 4.12: Controlling the game character by tilting the *iPad*.

Play mode and test mode allowed the player to control a rabbit character. The available control mechanisms were:

1. unless the character was in free fall, giving the rabbit an impulse pointing straight up, as if jumping
2. pulling the character straight left or right

There were two mutually exclusive mechanisms for pulling the character, joystick or tilting. When tilting was active, as illustrated in Figure 4.12, the vertical component of a vector pointing straight out of from the right edge of the screen would affect the the horizontal movements of the rabbit. To compensate for the difference in effort required to use the two mechanisms, the tilting had a larger zone yielding 100% power than the joystick. An angle of 30°would yield full speed when using the tilting mechanism.

There was also a joystick available when building, like the one available when testing. There was no rabbit character to control while building, but by using the joystick, the player could control the panning of the camera directly.

Figure 4.13 shows the three mechanisms for moving around while editing without actually testing or playing.

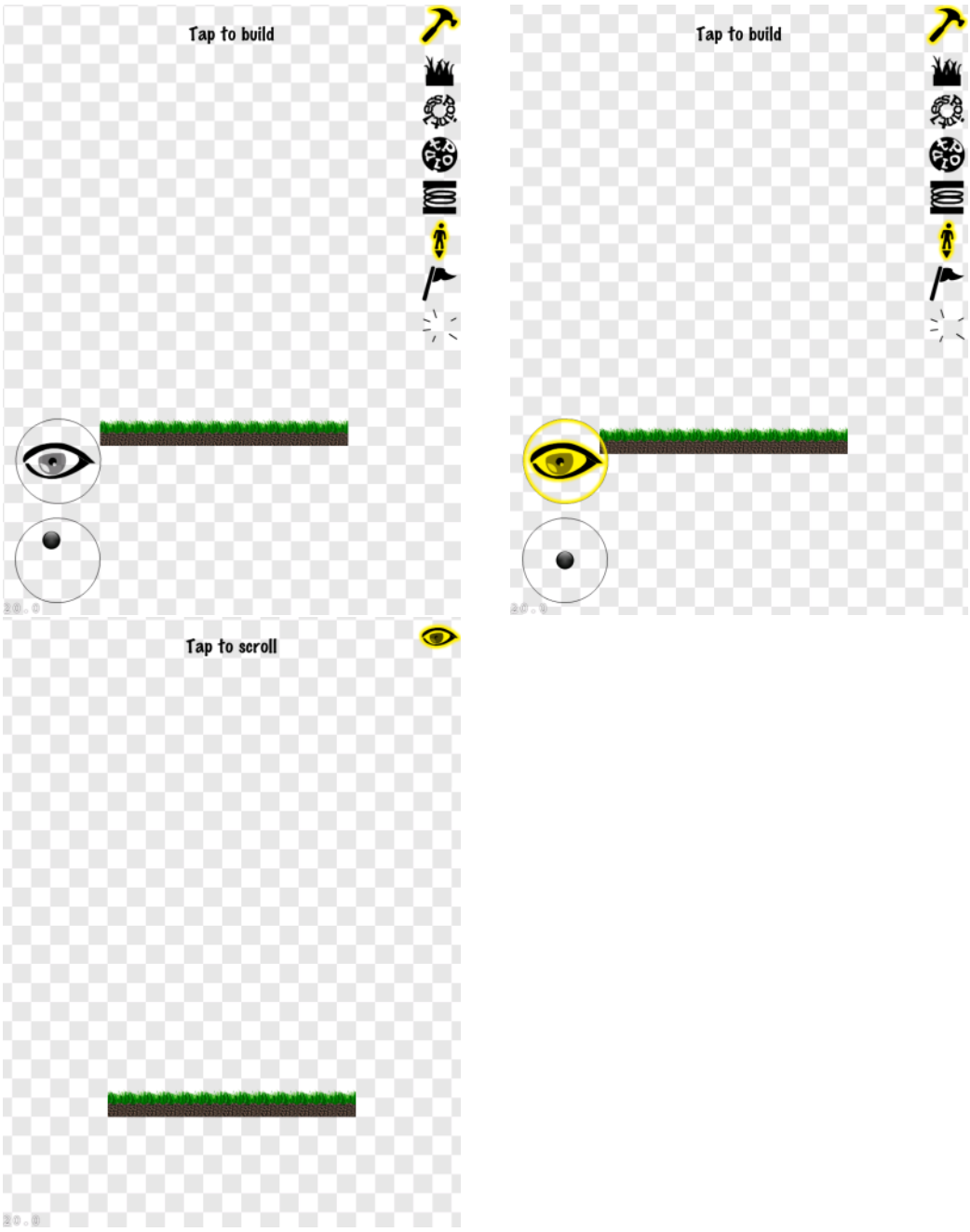


Figure 4.13: Scrolling mechanisms in the game

Figure 4.14: The editor, when holding the *iPad* horizontally.

The joystick controlled the viewpoint directly by translating angle into viewpoint panning speed. Holding the joystick at a constant non-zero angle from its origin would move the viewpoint at a constant speed until the limits of the level were reached.

The tool next to the joystick could temporarily activate a more direct viewing mechanism. While holding down one finger on the tool icon, the user could use finger swipes with another finger to pan the viewpoint. Translating finger swipes into relative position changes is a common mechanism on smartphones and post-PC tablets.

The third viewing mechanism had its own view mode. This was equivalent of constantly holding one finger on the swipe scrolling tool embedded in build mode.

Figure 4.14 demonstrates that the game was agnostic about which way the *iPad* was held. When holding it horizontally the view rectangle would change, rendering a different subset of the level. When the acceleration sensor controll mechanisms was activated this feature was deactivated for obvious reasons.

Figure 4.15 shows the play mode and the high score list mode. When playing the levels, the GUI was similar to the one found in the editor, but without any editing or saving features. The character would start at the spawning point, which was invisible in play mode. When the character touched the flag, the elapsed time and the number of points collected would be recorded

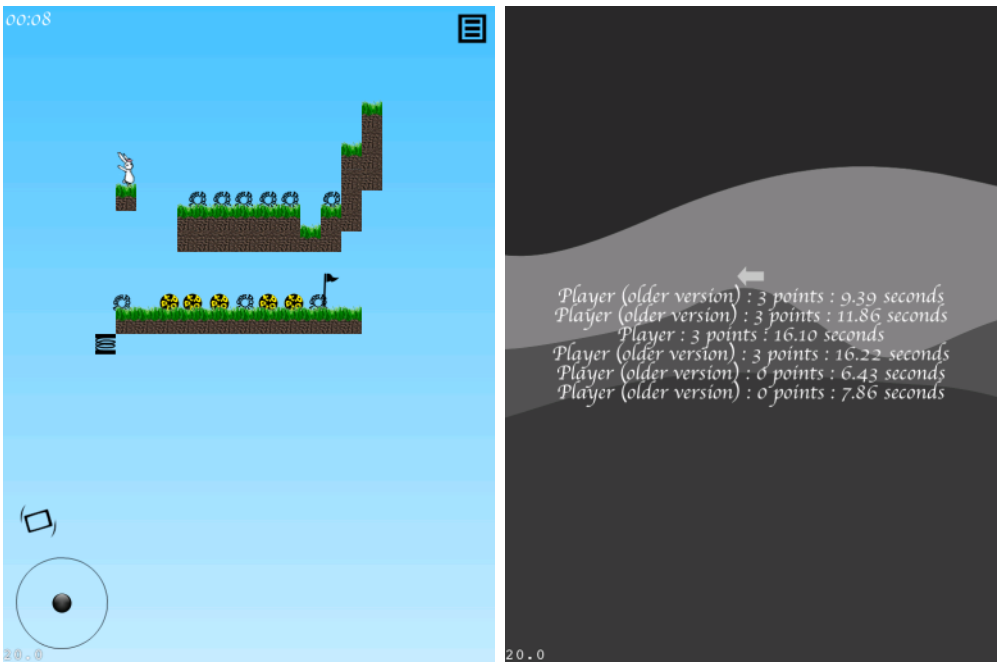


Figure 4.15: Play mode and a high score list.

in the high score list of the level.

Chapter 5

Results and evaluation

5.1 Experience with development tools

Using the Objective-C language and the *iOS*, *cocos2d* and *Box2D* libraries cost a lot more time than we had expected. We believe that it would be time well spent to search for alternatives in future projects. A good game library should provide all the most basic features of the UIKit, *cocos2d* and *Box2D*, eliminating the need to combine different libraries.

5.2 User tests

Procedure

The usability test was threefold; see Appendix A for the full details of the test document.

1. Preliminary questions about the attendant's game playing habits
2. Level building, level testing and free play as following
   (a) The attendant was asked to create a simplistic level with a given minimal amount of content. No help was given during this task.
   (b) The attendant was asked to create a level containing a puzzle or other kind of challenge. No help was given during this task.

(c) The attendant was given the opportunity to do whatever he or she wanted with the software, knowing that they were soon going to be asked questions about it. During this part of the test the attendants were explicitly allowed to ask any kind of questions.

3. Final questions about the game and the editor - These questions included the opportunity for the attendants to give suggestions about the software, written in free text.

System usability scale

The *System Usability Scale* can provide an assessment of the usability of a system (Lewis and Sauro, 2009). In this project a modified version of *SUS* were used. The modification was mainly to change the word 'system' for the word 'game'. The test is shown in Appendix A and the results in Figure 5.5.

Test subjects

Test subjects were chosen from students found at the *NTNU* campus between 9th and 11th of May. There was no explicit incentive for the students to attend the test other than the opportunity to test an *iPad* game for 10-15 minutes and to gain a little bit of insight into the project.

42 students were asked and 33 attended the test. The majority were in their 10th and final semester of a 5-year master's programme in Computer Science or Communications Technology. 4 children of ages 11, 11, 14 and 15 also attended. The average age of the attendants was between 22 and 23 years.

5.3 Analysis

The results in Figure 5.1-5.3 are provided for reference. They reveal that the test population consist mostly of people that play games regularly.

Figure 5.4 indicate that cellphones with keypads are unusual devices to play games on. It also shows that post-PC tablets is far less common than cellphones with touchscreen.

If we compare the numbers in Figure 5.4 we see post-PC tablets is the only platform where more people play regularly than actually own the device.

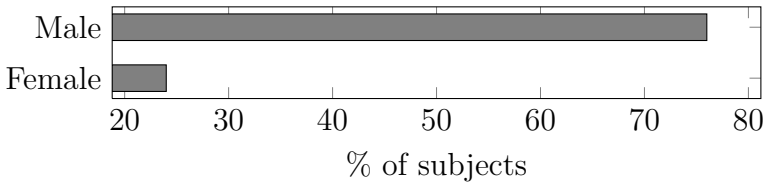


Figure 5.1: Male and female test subjects

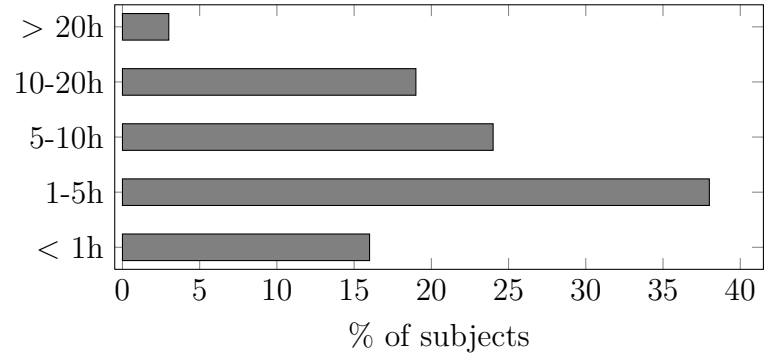


Figure 5.2: Weekly time spent playing games

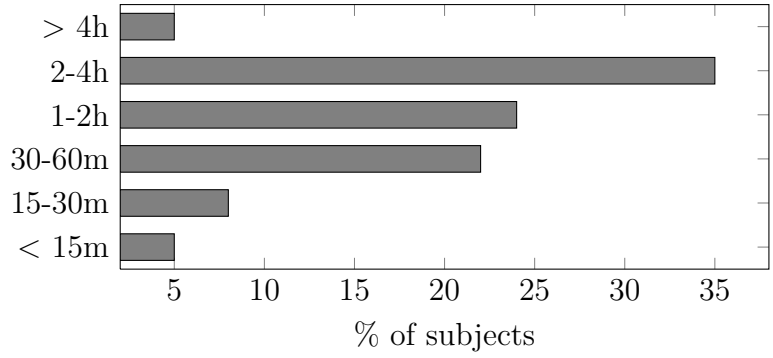


Figure 5.3: Time spent per playing session

Keypad cellphones have the exact opposite trend, as 5 out of 6 don't use their phones for playing. Among handheld and TV console owners, one third don't play.

The attendants were asked which games they had played the most the last 5 years. Only 4 games were mentioned by more than 10% of the attendants, Starcraft II, World of Warcraft, Team Fortress 2 and Morrowind: Oblivion.

The results from the system usability scale gives the game a score of 74.1, this will roughly place it among the 30% most usable programs (Sauro, 2011). A score of 80.4 would place the game among the top 10% of the programs. These top score programs are more likely to be recommended by the users, which is a critical property for viral marketing.

Figure 5.6 imply that the test population feel confident about their own creativity. It also shows, even more evidently, that the population feel comfortable when using touch-based editors.

Figure 5.7 suggests that the test population like to be creative and share their content in games, but that this is not necessarily the most important feature when they choose their games.

Figure 5.8 presents a population that would like to play games on the *iPad*, but not all agree that games with level editors are preferable.

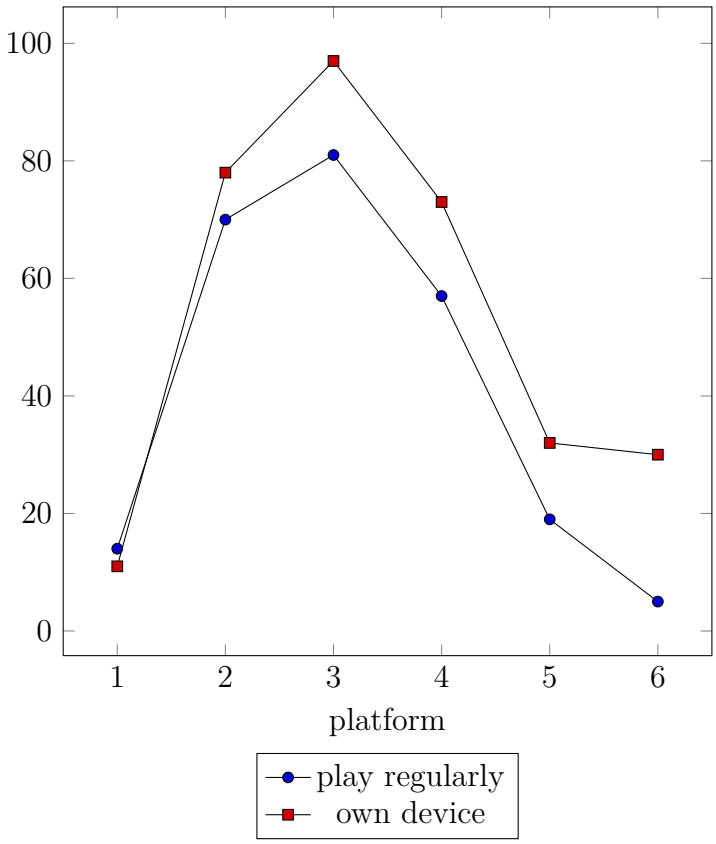
Particularly popular and unpopular properties

The attendants were given the opportunity to give answer whether they liked or disliked anything in particular about the editor. More than 10% of the attendants gave the following, or equivalent, comments.

Good
- The editor was easy to use.

Bad
- I needed some more advanced objects.
- I needed some explaining text in the GUI.
- I (could easily have) deleted my level by accident.
- I found the GUI unfamiliar.

The figures 5.9-5.12 show simplified lists of further suggestions from the test attendants. These were given as answers to the following four questions:



1. Post-PC tablet
2. Cell w/ touchscreen
3. Computer
4. TV console
5. Handheld console
6. Cell w/ keypad

Figure 5.4: Test subjects who own and use devices

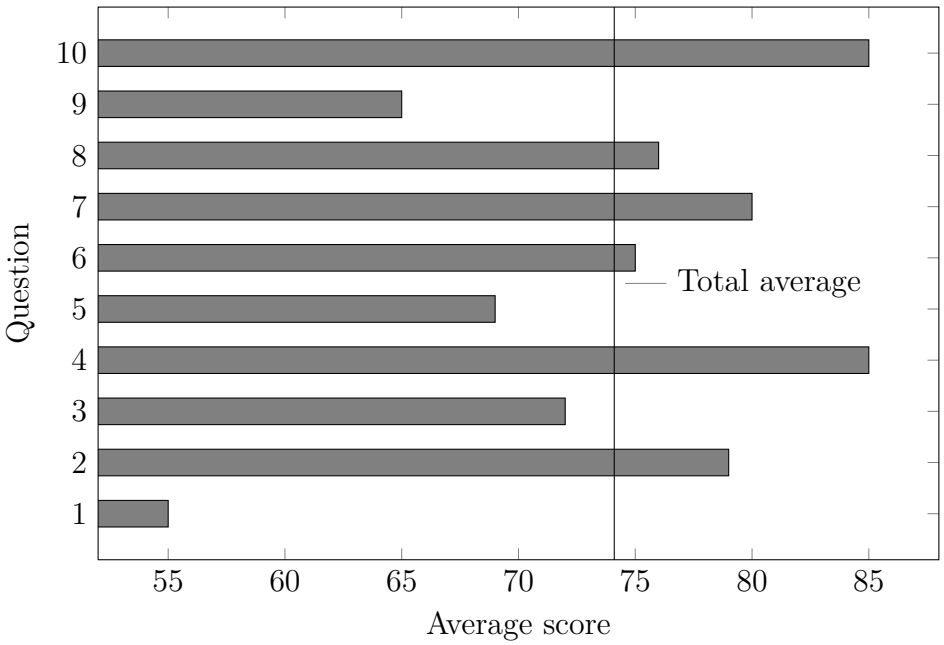


Figure 5.5: System usability score (SUS)

- Were you missing anything the editor?
- Do you have any suggestions about the editor?
- Were you missing anything in the game as a whole?
- Do you have any suggestions about the game as a whole?

The results are in agreement with the first listing in Section 5.3. They suggest three areas of improvement potential:

- There should have been more advanced game objects.
- The GUI was not intuitive enough.
- Better progress handling should have been available. This could have been automatic saving, undo options etc.

This suggests that the applied menu appearance strategy was a failure. The users were confused and did only occasionally understand the meaning of a glyph before they had tried it out. They did, however, usually recall the

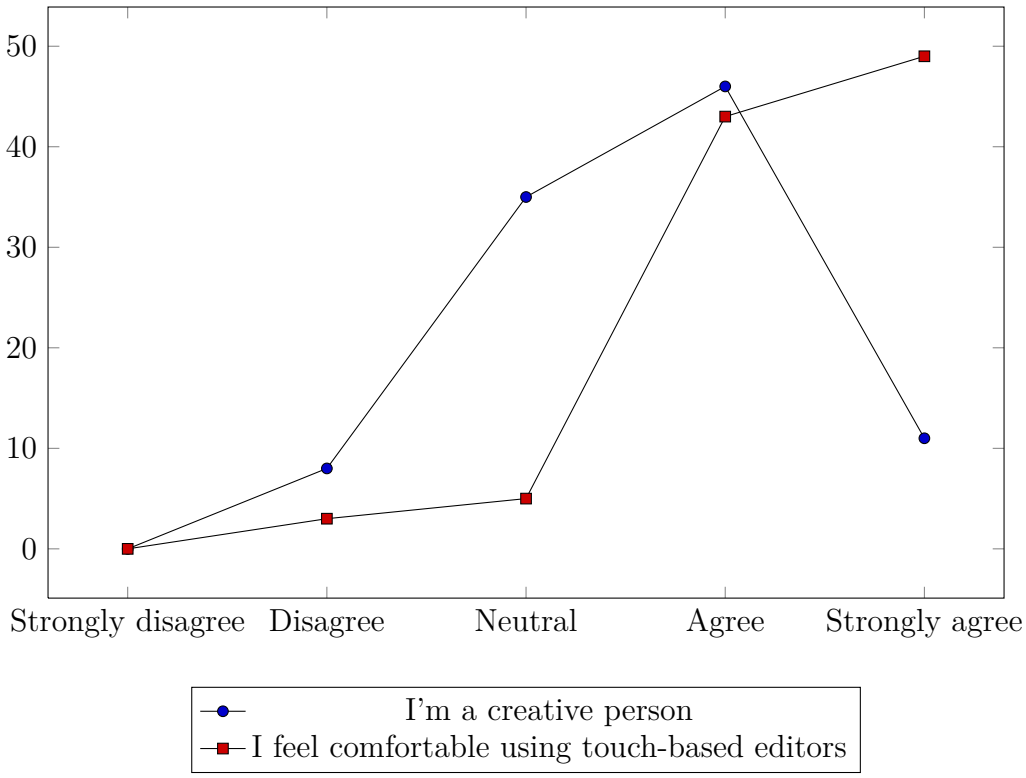


Figure 5.6: Subjects about creativity (1/2)

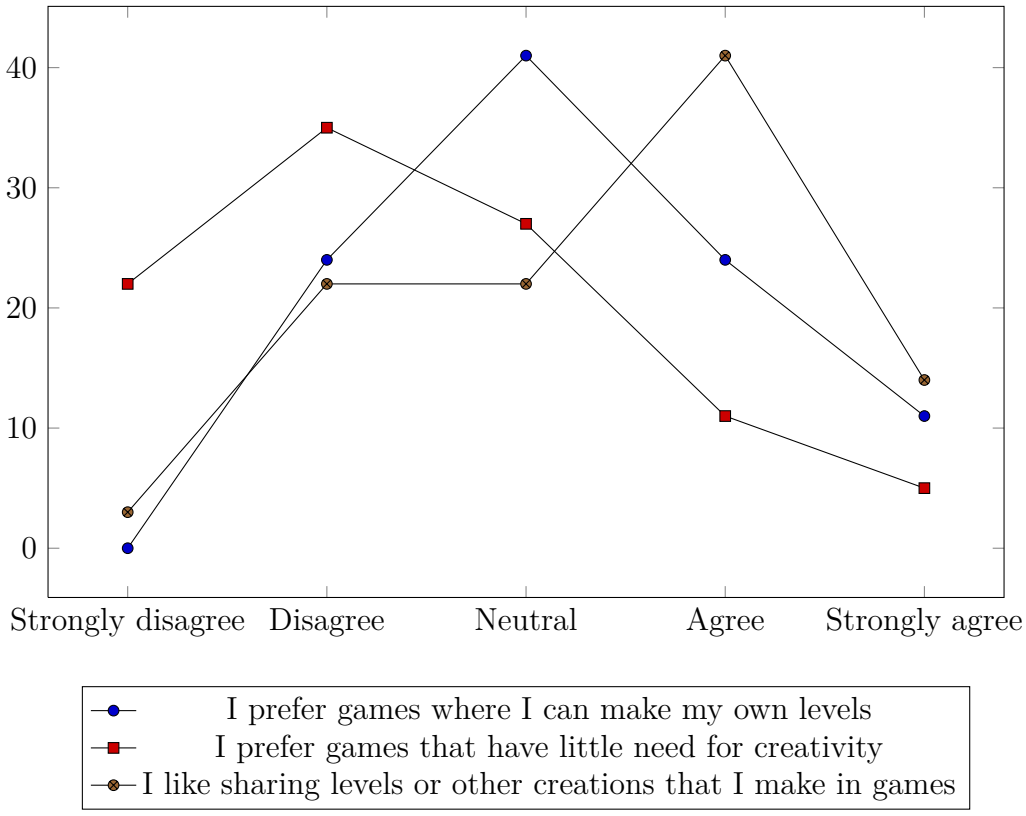


Figure 5.7: Subjects about creativity (2/2)

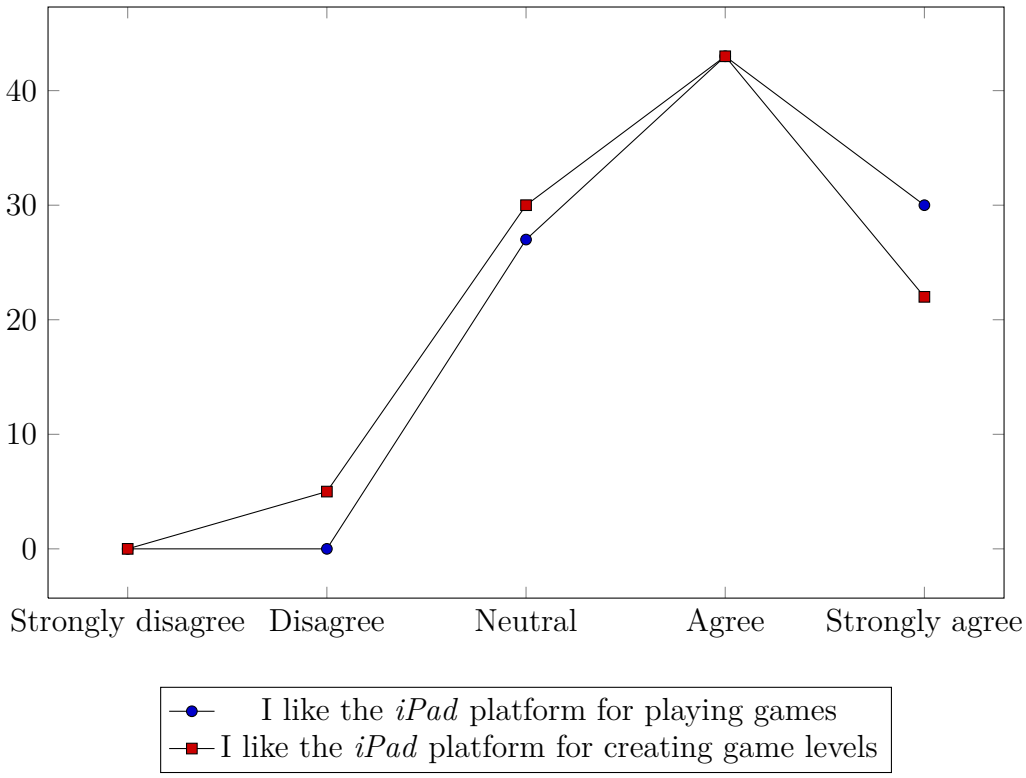


Figure 5.8: Subjects about the *iPad*

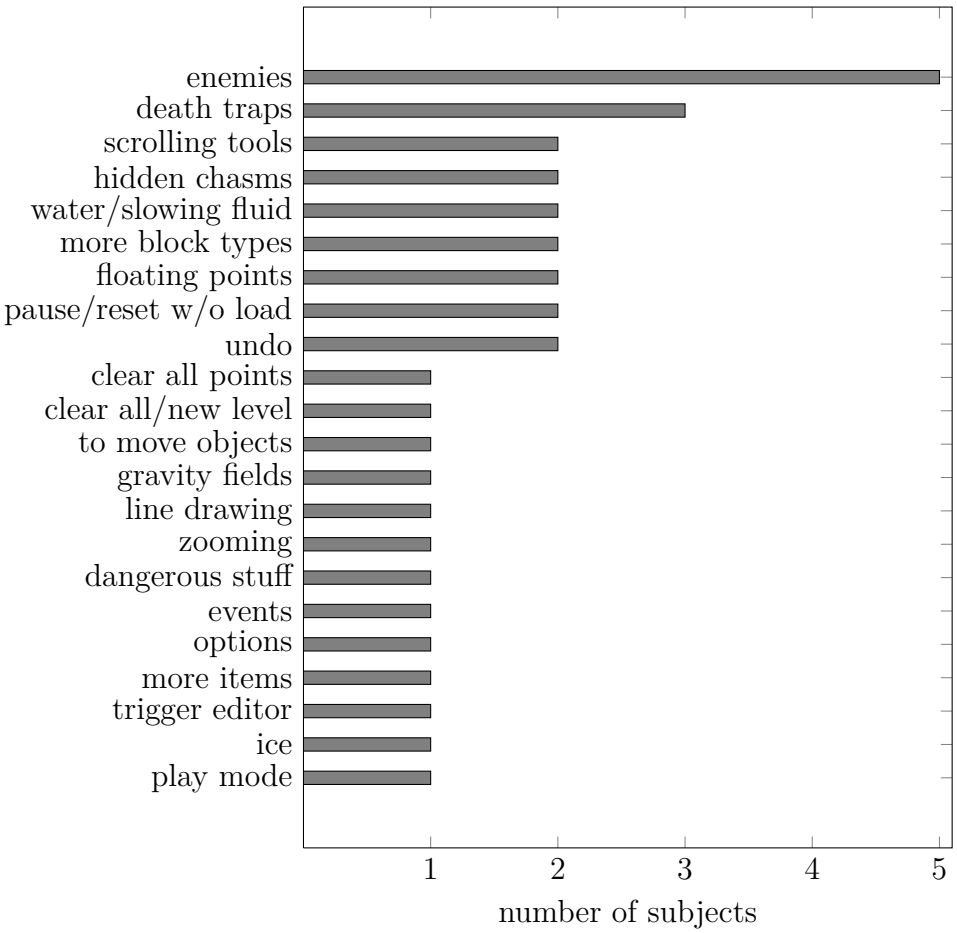


Figure 5.9: Missing tools or options in the editor

meaning of every glyph, once they had tried them at least once. Figure 5.5 did not render the problem as critically bad, but there is room for improvement.

The figures 5.13-5.15 shows how large percentage of the population who used some of the different features in *Platformer Playground*. The attendants were not explicitly taught how to use these features and could not ask for help during the first two parts of the test. The third and final part was not mandatory, but included the opportunity to ask questions and get help.

There was one attendant who did not use any of the controller-mechanisms for the game character, but all claimed to have tested their levels and used at least one of the scrolling mechanisms.

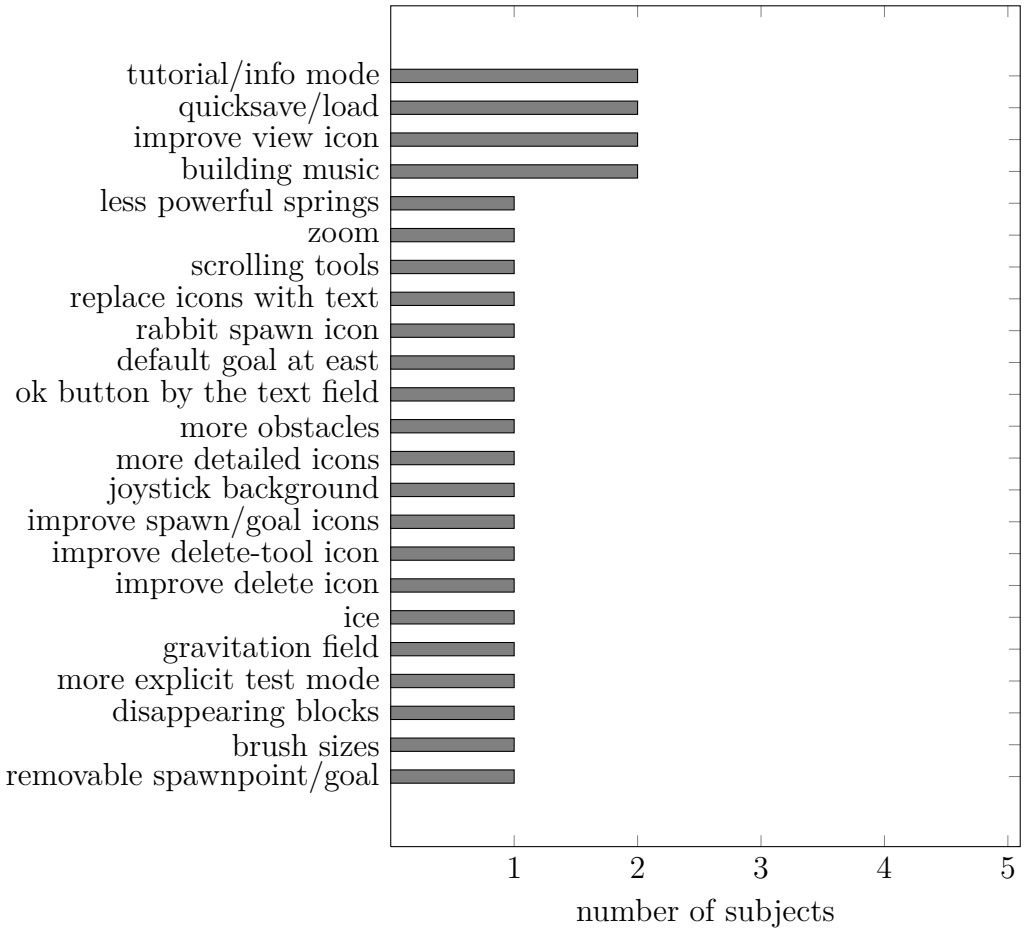


Figure 5.10: Suggestions about the editor

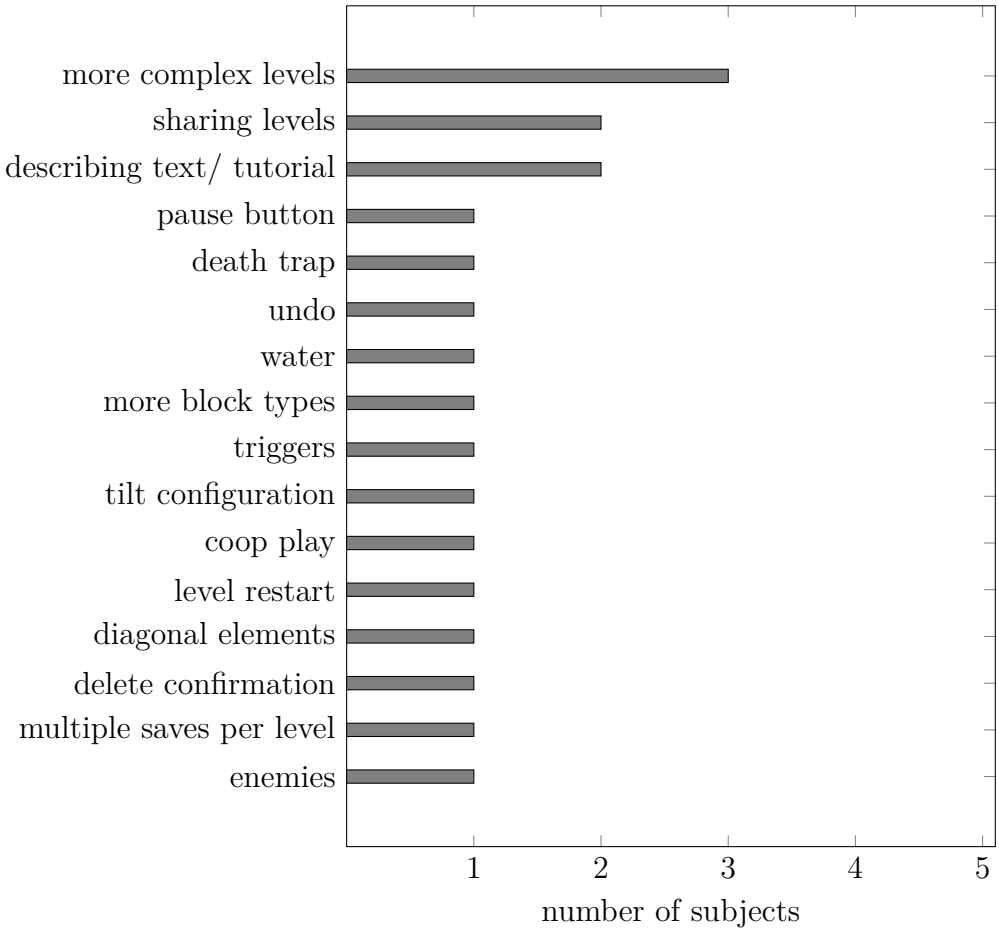


Figure 5.11: Missing options/tools in the game as a whole

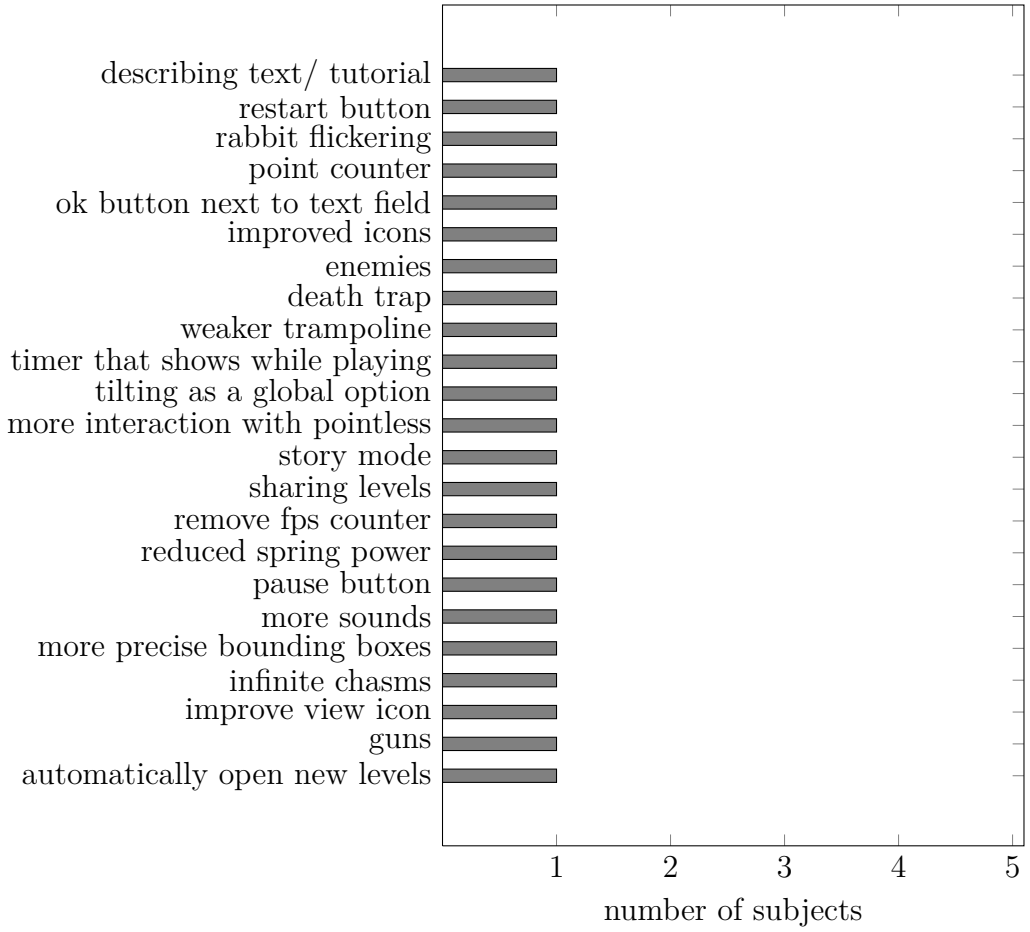


Figure 5.12: Suggestions about the game as a whole

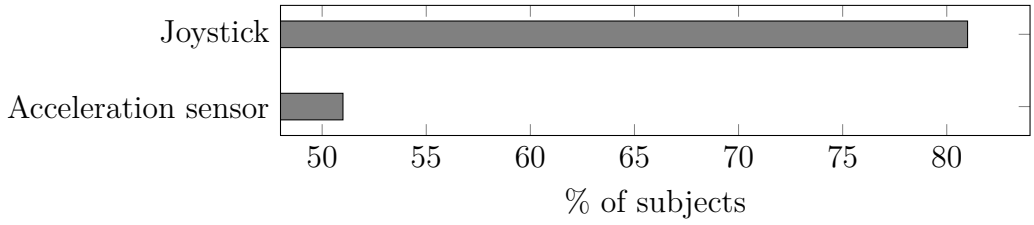


Figure 5.13: Test subjects who used controller-mechanism

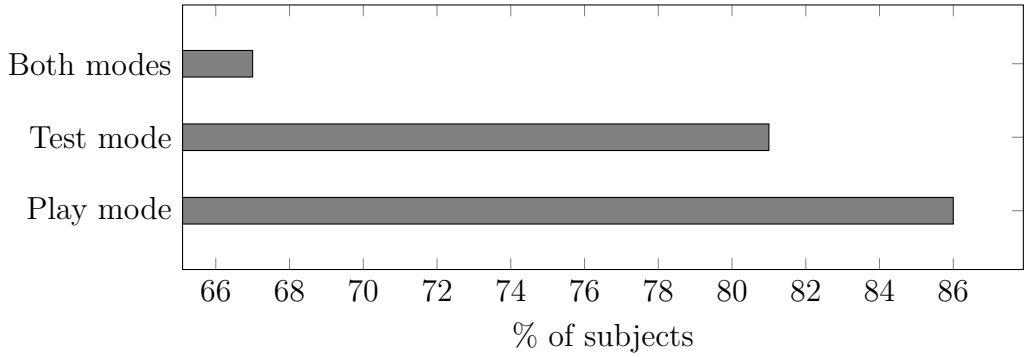


Figure 5.14: Test subjects who tested their levels

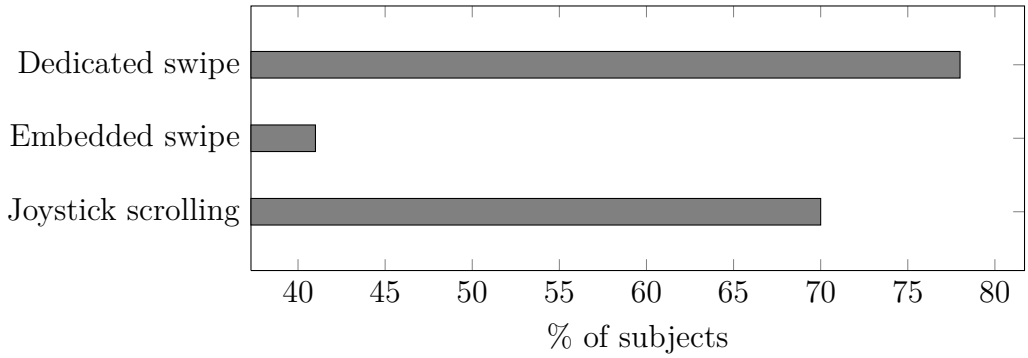


Figure 5.15: Test subjects who used scrolling mechanism

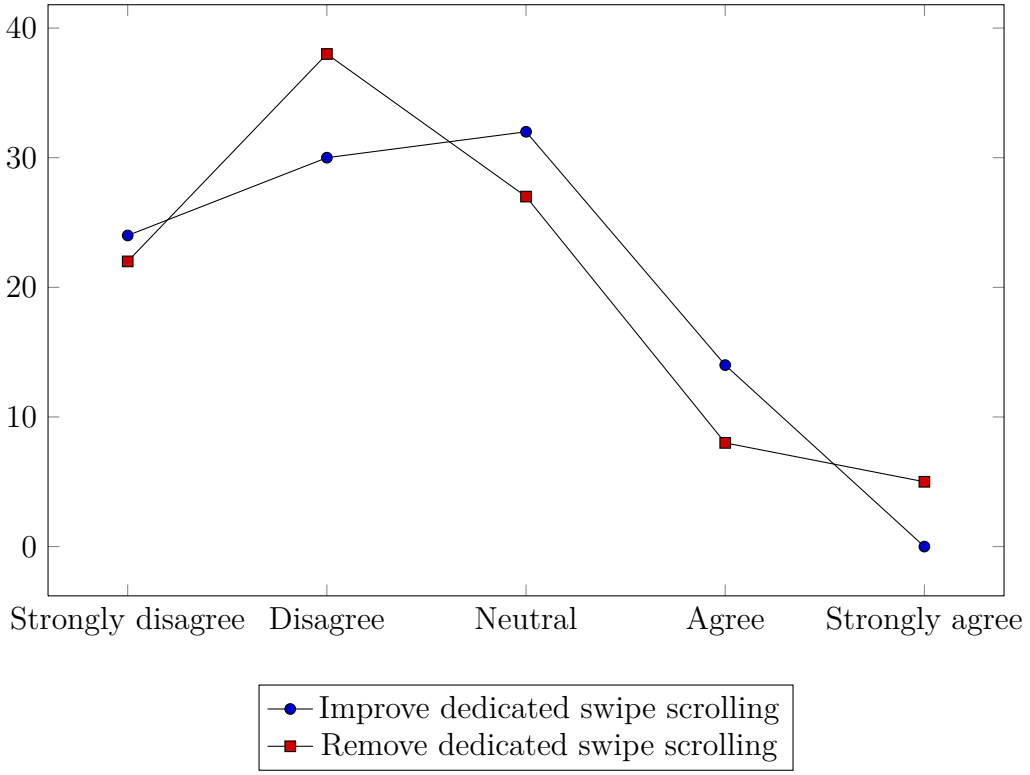


Figure 5.16: Test subjects who agree to changes

When the test attendants answered whether they would suggest removing or improving the features mentioned in the figures 5.16-5.18 the *iPad* and the game was still available, and they could still ask questions.

Figure 5.16 suggests that the test attendants were happy about the dedicated swipe scrolling mode. They do not want it improved and do not want it removed.

Figure 5.17 is in agreement with Figure 5.15 that the swipe scrolling tool, embedded in the building mode ('*/edit//build*'), needed some improvement. It does, however, look like the attendants liked the tool when they finally found out how to use it.

It is evident from Figure 5.18 that the joystick scrolling tool was important to the test attendants as almost 50% were in strong disagreement that the tool should be removed.

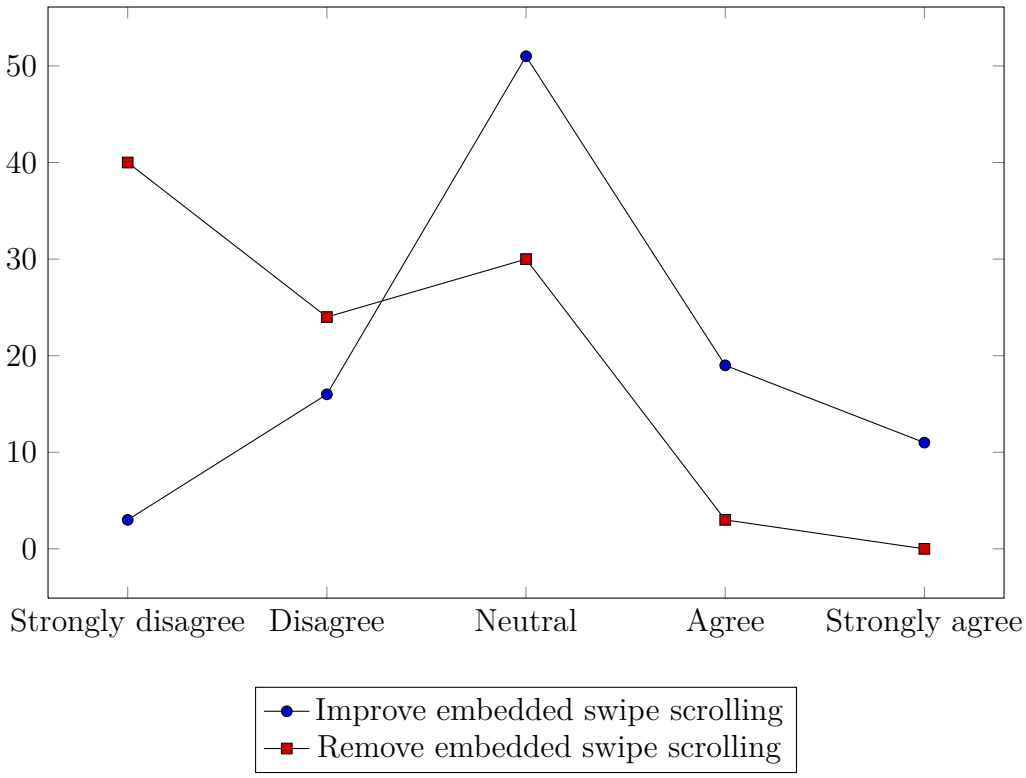


Figure 5.17: Test subjects who agreed to changes

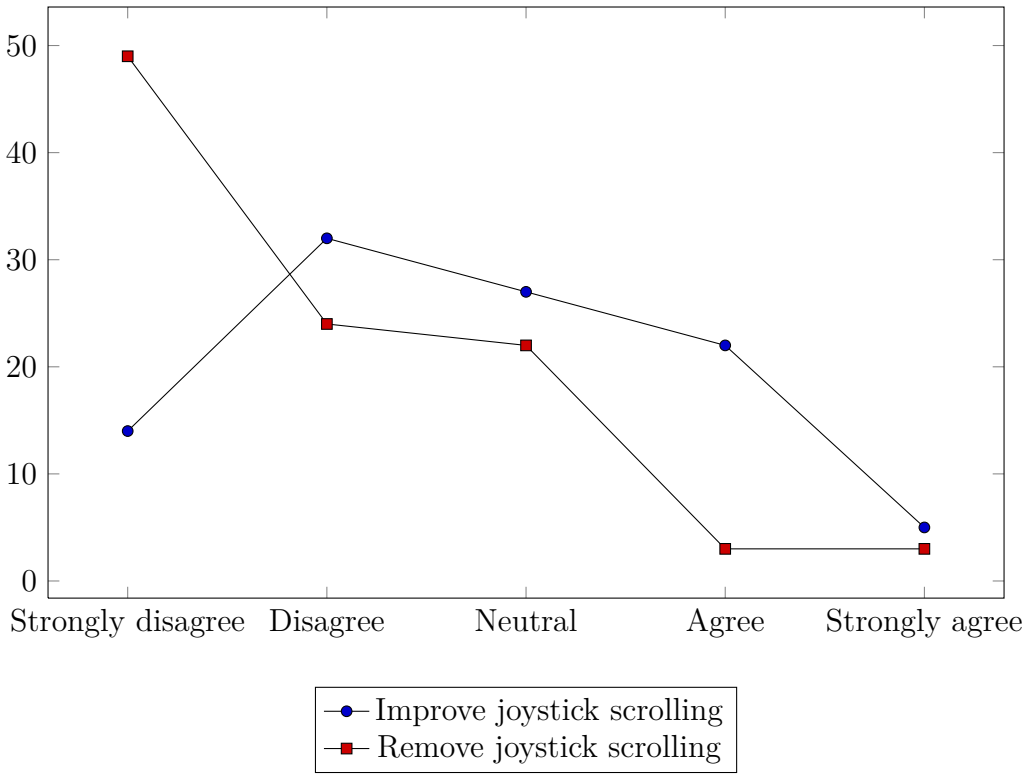


Figure 5.18: Test subjects who agreed to changes

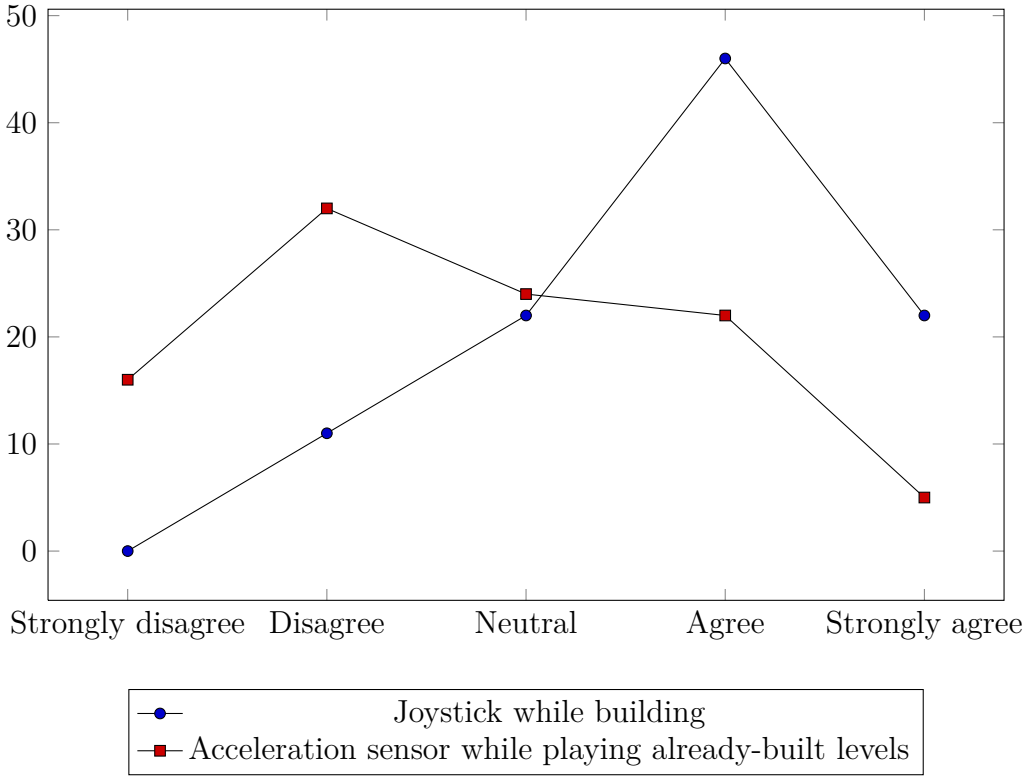


Figure 5.19: Test subjects who prefer control mechanisms

As Figure 5.19 shows, the joystick was the preferred tool for controlling the game character. The tilting feature appeared more like a curiosity, but was welcomed among some users.

While the behavior of the '*point*' and '*pointless*' game objects could potentially increase the entropy of the game levels they were well received by the test population. Some users mentioned that they were hard to control, so this might mean that the entropy is not available to the users anyway. For example it was virtually impossible to stack more than two marbles on top of each other.

As Figure 5.20 shows, the users recognize the simplicity of the grid based blocks better than of '*point*' and '*pointless*', but both concepts seem well received.

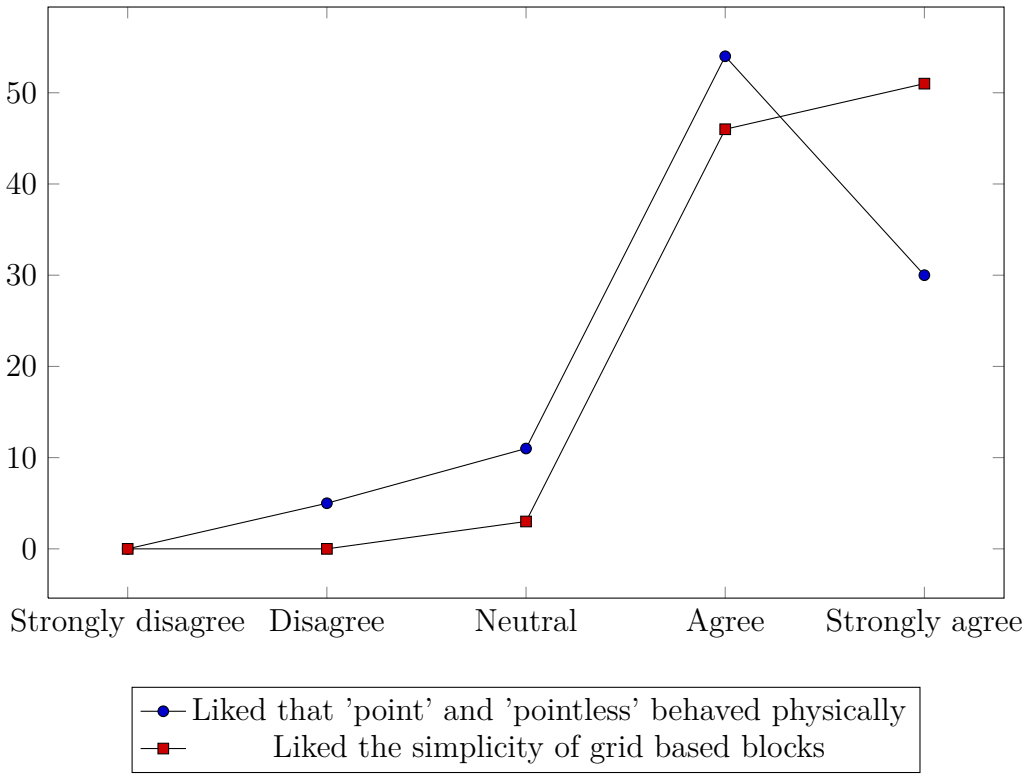


Figure 5.20: Test subjects who prefer control mechanisms

Creative processes

We observed a lot of creativity during the tests. Appendix B shows sketches from some of the game levels that the test attendants produced. The range of use cases included:

- simple drawings, using the grass blocks
- simple marble and spring based systems
- classic, easy platform levels
- platformer puzzles
- spring based jumping arenas

No two levels looked the same, even though they would have been easy to reproduce. All the attendants got the same tasks that they were supposed to solve, but few solved it in the same way. Most of the tests were conducted with two attendants at the same time, and many wanted to test each others game levels, especially those attendants who knew each other.

Especially kids found strong motivation in the support for creative processes. A four year old girl, who did not attend the test, used the game primarily for drawing. She called it the "house drawing game".

Chapter 6

Conclusion

In this project we worked on the research questions in Section 2.3. These relate game level editors and the differences between game consoles, PCs and post-PC tablets.

6.1 Post-PC tablet game level editors

General

Post-PC tablet game level editors, are much like PC game level editors and console game level editors. A powerful editor will create levels with much entropy and be complex to use (El-Nasr and Smith, 2006). Users seem to want both simplicity and power, whether they are on post-PC platforms or elsewhere.

Scrolling

Scrolling is a mechanism for working with levels that are larger than a single screen. Three mechanisms were tested and they were all used by the users. Some users only learned how to use a single mechanism, while other enjoyed having multiple alternatives available. The most complex alternative required two fingers to use and was easily misinterpreted by the users because of its appearance. However, even the most complex alternative was appreciated by the users who learned how to use it.

H1 There is no single scrolling mechanism that will fit the majority of users in general.

This hypothesis is in agreement with the food industry's view of consumers. There are many good recipes, and the best solution is to offer them a choice between the most popular ones (Gladwell, 2004).

Controlling the character in platformer games

The majority of test subjects that were given the choice between controlling the game character with an emulated joystick or tilting the device did prefer the joystick. This result, however, is not very general, because the joystick provided rougher precision and also much less effort than tilting the device. No experimenting with the effort and precision of the mechanisms was carried through, so these differences might have had the most significant impact.

It might be best to offer the users a choice, but we would not like to make this claim without further evidence.

Low entropy building schemes

The grid based block scheme that was used in the game was one of the most popular features. More than 95% of the test population reported that they were in agreement or strong agreement that they enjoyed the simplicity of grid based blocks.

H2 Users generally enjoy the simplicity of low entropy building schemes.

Square buttons

(App, 2010) and (Goo, 2011) advise *iOS* and *Android* developers about the design of square buttons that invite users to tap them. We would further like to emphasize that buttons and other widgets should be as intuitive to understand as possible. This advice is applicable for most platforms where usability is a desired quality, however it requires extra attention because there is a convention that finger-tip-sized square buttons should be used on touchscreen platforms.

While some of our menu items had a satisfactory level of affordance, others did not. We did not investigate whether or not text should be used. Since our game had both buttons that were well and poorly understood, by the test attendants, we believe that designing a comprehensive set of intuitive buttons is possible. User tests can help assure that the designs work well for the majority of the users.

Creativity

We believe, based on the diversity of the game levels that were produced during our tests, that post-PC tablet game level editors can support creative processes very well. We have not provided any strong evidence, but we have not made any observations that suggest otherwise. Since creative processes are central to game level creation, this support is a prerequisite.

6.2 Summary

This project was conducted in order to answer questions about the compatibility between *iPad*, post-PC tablets, and play-create-share games.

One possible contribution towards aswering RQ1 is: approximately square icons, that invite the user to tap them should also be self explanatory. This is hard to achieve and the design process should be aided with user tests and lots of continuous effort.

RQ2 has only been partly answered in this report. An emulated joystick was the favorite choice among the test population, but the two mechanisms provided in the game were not properly aligned for comparison. To assess which mechanism is the best, they should first be aligned for effort required and achievable precision. In the end, the users might even require both mechanisms.

One possible answer to RQ3 is provided as H1. Editors that handle levels that are bigger than the screen should provide a carefully selected set of scrolling mechanisms. Three examples of such mechanisms have been provided.

RQ4 has been answered only by the diversity of the games that the test attendants produced. Further investigation might reveal that better support for creative processes is possible. In particular, more powerful editors might trigger better or more complex ideas of the minds of the users.

6.3 Further work

I see a bright future for play-create-share games because of all the people that enjoyed playing the game. The implementation was small and simplistic, yet it was fun to use.

In the future I would recommend game designers to provide multiple scrolling mechanisms whenever scrolling is needed. I would also recommend putting some serious effort into the menu design. Buttons and similar widgets on Post-PC tablets should be more square than their PC and console counterparts. Text litter small screens easily, so if you choose to have text, make it worthwhile.

Further research efforts could reveal better uses of the acceleration sensor or cameras. The fist model of the *iPad* did not have a camera, but other post-PC tablets and the *iPad 2* do. This could open up new possibilities for editor interfaces. For instance head tracking could be used as a mechanism for scrolling (Bérard, 1999).

Another area that needs work is the sharing platform. I believe that no single, central server should be required in order to share levels with your friends. Central servers represent single points of failure and usually require maintenance (Lee, 2011). P2P technologies or local area wireless communication might help remove this requirement. The XMMS or IRC protocols or even email are communication media that could simplify the design of a server that users would be able to set up on their own.

Appendix A

Test document

Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.

Age: _____

Sex: _____

About your game playing habits:

1. On an average week, how much time do you spend playing games?
   - A. more than 20 hours
   - B. between 10 and 20 hours
   - C. between 5 and 10 hours
   - D. between 1 and 5 hours
   - E. less than 1 hour

1. _____

2. On average, how long do you play one game during one game playing session?
   - A. more than 4 hours
   - B. between 2 and 4 hours
   - C. between 1 and 2 hours
   - D. between 30 and 60 minutes
   - E. between 15 and 30 minutes
   - F. less than 15 minutes

2. _____

Answer the following questions by putting a cross in the most appropriate box. For example, if you mean 'yes', then put a cross in the box next to 'yes'.

3. Do you play the following platform at least a few times a month?
   (a) Computer

   Yes ☐ No ☐

   (b) TV based console

   Yes ☐ No ☐

   (c) Mobile phone with touchscreen (limited or no keypad)

   Yes ☐ No ☐

   (d) Mobile phone with keypad

   Yes ☐ No ☐

   (e) Post-PC tablet (iPad, Samsung Galaxy Tab, Motorola Xoom etc.)

   Yes ☐ No ☐

(f) Handheld console (PSP, DS etc.)

Yes ☐ No ☐

4. Do have your own?

(a) Computer

Yes ☐ No ☐

(b) TV based console

Yes ☐ No ☐

(c) Mobile phone with touchscreen (limited or no keypad)

Yes ☐ No ☐

(d) Mobile phone with keypad

Yes ☐ No ☐

(e) Post-PC tablet (iPad, Samsung Galaxy Tab, Motorola Xoom etc.)

Yes ☐ No ☐

(f) Handheld console (PSP, DS etc.)

Yes ☐ No ☐

5. Which is your favorite platform at the moment?

5. _____

6. What your favourite game genre? If you have many you can list them all.

6. _____

7. Which platform have you played the most the last 5 years?

7. _____

8. Which games have you played the most the last 5 years?

You are going to test a create-and-play gameplatform for touchscreen based handheld devices.

For us it is important to understand how well you can use the system with little or no help from us. If you have any questions for us, you should ask them before you start testing. If the software is malfunctioning however, we might step in to help during the test.

You're going to use the gameplatform to create two game levels.

- The first should be a minimalistic, but playable level containing a few grass blocks, 'pointless' marbles and 'point' marbles. Make sure to tell us when you think you're done with this first level.
- The second level can be however hard you want it to be, but it should have at least one obstacle, puzzle or other kind of difficulty.
- If you complete your second level before the time limit, you may play around with the platform if you wish. Make whatever you want to make.

Answer how much you agree to the following statements

9. I think that I would like to play this game frequently

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

10. I found the game unnecessarily complex

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

11. I thought the game was easy to use

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

12. I think that I would need the support of a technical person to be able to use this game

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

13. I found the various functions in this game were well integrated

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

14. I thought there was too much inconsistency in this game

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

15. I would imagine that most people would learn to use this game very quickly

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

16. I found the game very cumbersome to use

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

17. I felt very confident using the game

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

18. I needed to learn a lot of things before I could get going with this game

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

19. I'm a creative person

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

20. I feel comfortable using touch-based editors

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

21. I prefer games where I can make my own levels

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

22. I prefer games that have little need for creativity

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

23. I like sharing levels or other creations that I make in games

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

24. I like the iPad platform for playing games

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

25. I like the iPad platform for creating game levels

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

26. What did you like or dislike about this particular level editor?

27. There are three different scrolling tools in the level editor.

    1. A dedicated view mode that use finger swipes
    2. A similar finger swipe tool embeded in the building mode. To use this the player has to hold one finger on the tool icon and make finger swipes with another.
    3. A camera panning tool that uses an on-screen joystick.

    (a) Did you use scrolling tool 1?

    Yes ☐ No ☐

    (b) Did you use scrolling tool 2?

    Yes ☐ No ☐

    (c) Did you use scrolling tool 3?

    Yes ☐ No ☐

28. Did you test your level while you were in editor mode (the mode with a checkerboard background)?

    Yes ☐ No ☐

29. Did you test your level in playmode?

    Yes ☐ No ☐

30. Did you control the character with the on-screen joystick?

    Yes ☐ No ☐

31. Did you control the character by tilting the device?

    Yes ☐ No ☐

32. Were you missing any features in the editor?

    Yes ☐ No ☐

33. If you answered yes in the previous question, what feature were you missing?

34. Do you have other kinds of suggestions about the editor?

35. Were you missing any features in the program as a whole?

Yes ☐ No ☐

36. If you answered yes in the previous question, what feature were you missing?

37. Do you have other kinds of suggestions about the program as a whole?

38. I would suggest that the dedicated view mode should be improved

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

39. I would suggest that the dedicated view mode should be removed

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

40. I would suggest that the finger swipe based scrolling tool in the building mode should be improved

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

41. I would suggest that the finger swipe based scrolling tool in the building mode should be removed

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

42. I would suggest that the on-screen joystick scrolling tool in the building mode should be improved

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

43. I would suggest that the on-screen joystick scrolling tool in the building mode should be removed

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

44. I prefer controlling the character using the on-screen joystick while testing levels that I am building

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

45. I prefer tilting the device when playing already-built levels

Strongly disagree	Disagree	Neutral	Agree	Strongly agree

46. I liked that the round game elements, called '*point*' and '*pointless*', behaved realistically

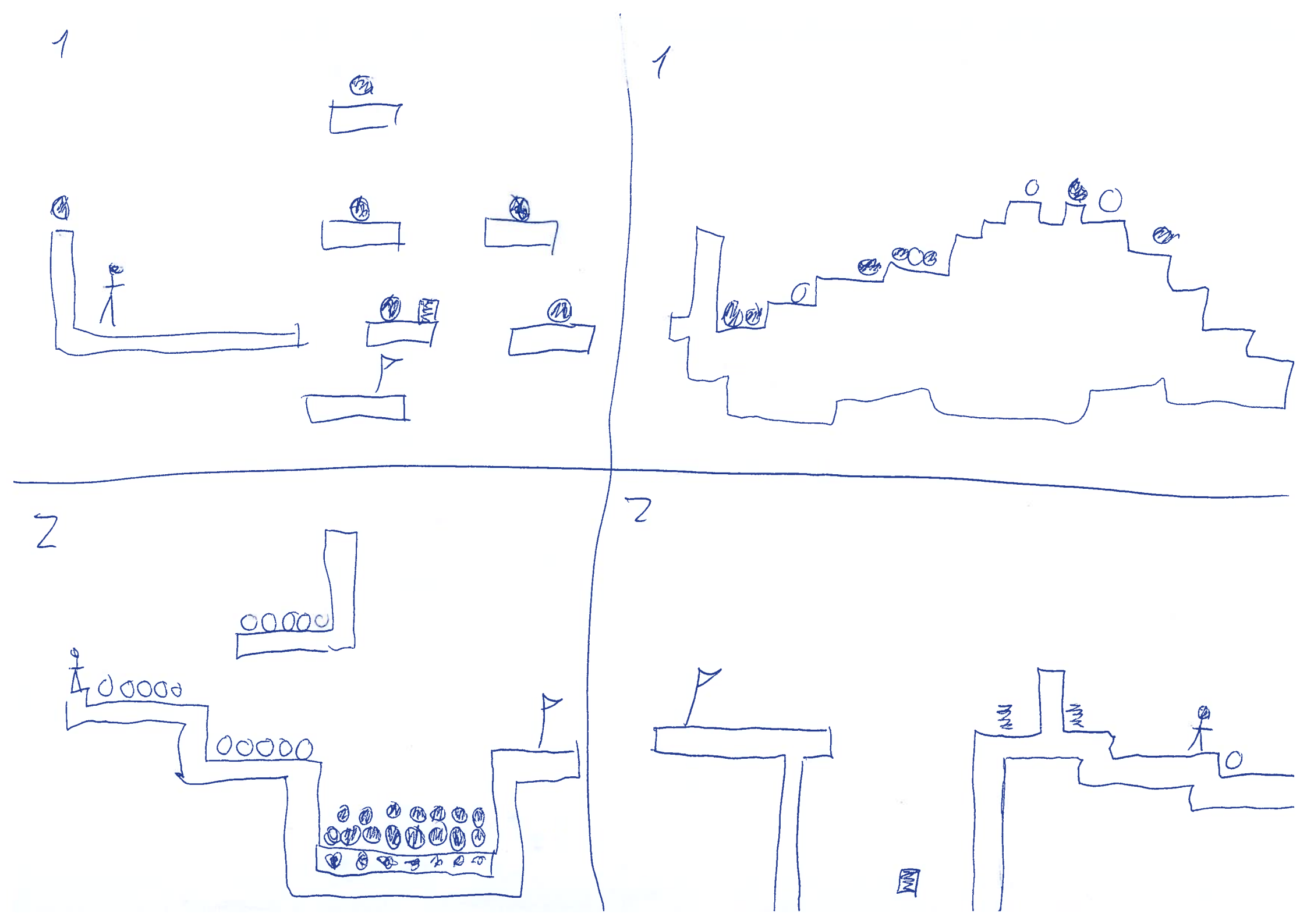
Strongly disagree	Disagree	Neutral	Agree	Strongly agree

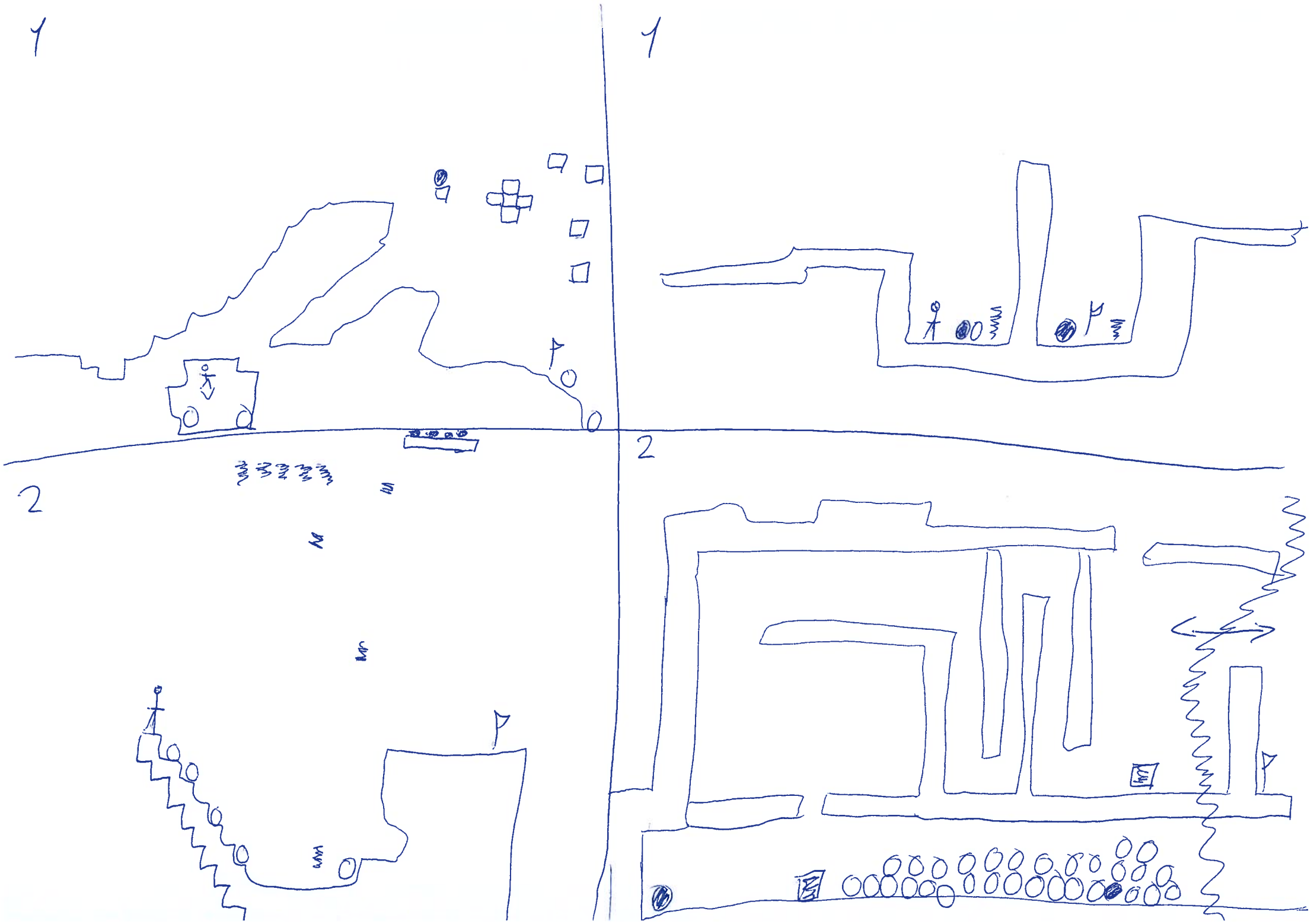
47. I enjoyed the simplicity of grid based grass blocks

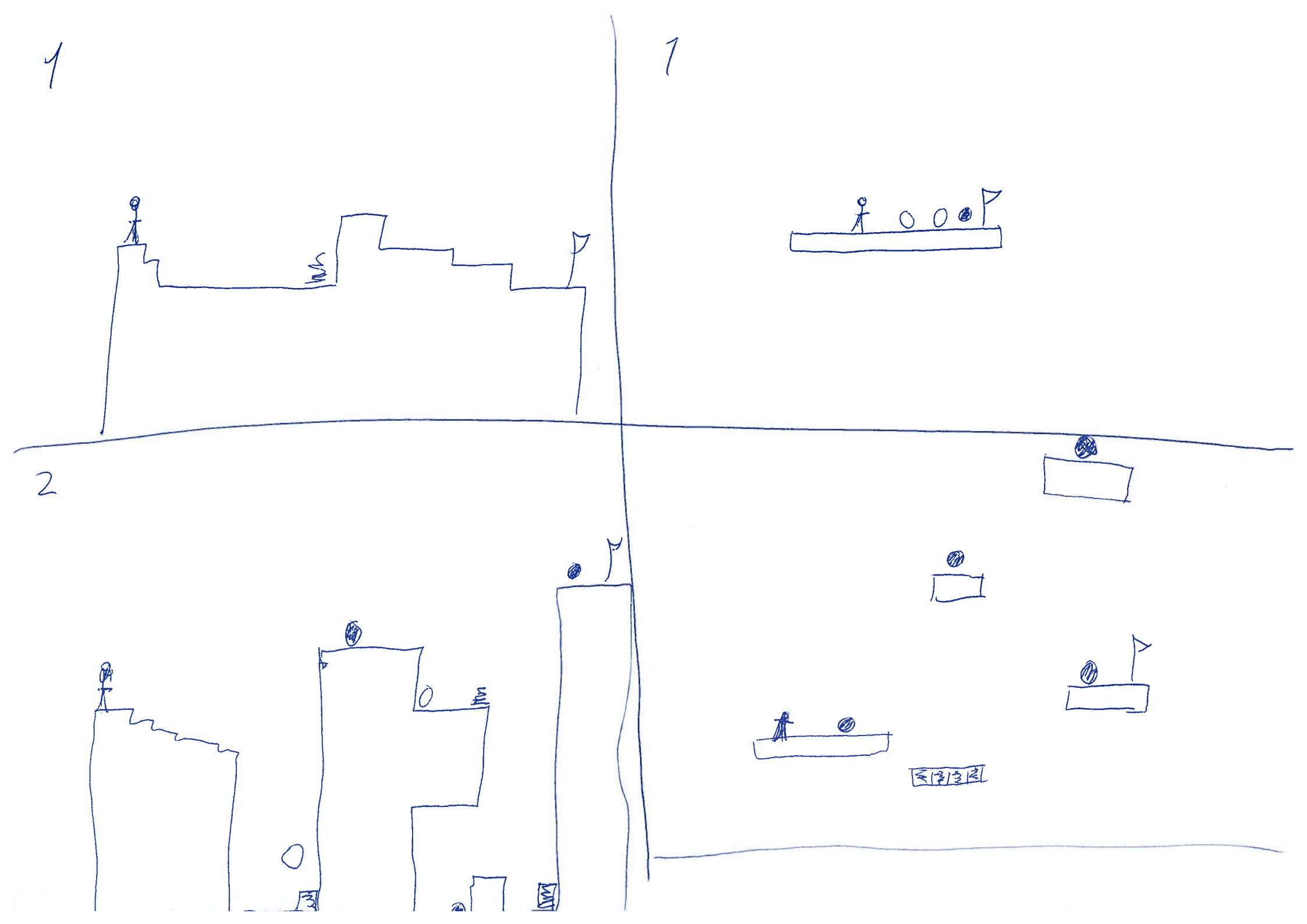
Strongly disagree	Disagree	Neutral	Agree	Strongly agree

Appendix B

Sketches of levels created by the test attendants


1
1
2
2


1
1
2
2


1
1
2

Appendix C

Raw answer data from the questionnaire

#	Age	Sex	1	2	3a	3b	3c	3d	3e	3f	4a	4b	4c	4d	4e	4f
1	20	m	c	c	yes	no	no	no	no	no	yes	no	no	no	no	no
2	21	m	a	a	yes	yes	yes	no	no	no	yes	yes	yes	yes	no	no
3	26	m	e	b	yes	no	yes	no	no	no	yes	no	yes	no	no	yes
4	24	m	c	c	yes	yes	no	no	no	yes	yes	yes	yes	no	no	yes
5	25	m	b	c	yes	yes	no	no	no	no	yes	yes	no	no	no	no
6	23	m	b	b	yes	no	no	no	no	no	yes	no	no	yes	no	no
7	24	m	c	b	no	yes	yes	no	no	no	yes	yes	yes	no	no	no
8	23	f	e	d	yes	no	yes	no	no	no	yes	yes	yes	no	no	no
9	24	m	d	c	yes	no	yes	no	no	no	yes	no	yes	no	no	no
10	24	m	e	f	no	no	yes	no	no	no	yes	no	yes	no	no	
11	23	f	d	e	no	yes	yes	no	yes	no	yes	yes	yes	no	no	no
12	24	m	b	b	yes	yes	yes	no	no	no	yes	yes	yes	no	no	no
13	26	m	b	d	no	no	yes	no	no	no	yes	yes	yes	no	no	yes
14	25	m	c	d	yes	yes	yes	no	no	no	yes	yes	yes	no	no	no
15	27	m	d	d	yes	no	yes	no	no	no	yes	yes	yes	no	no	yes
16	25	m	d	f	no	yes	yes	no	yes	no	yes	yes	yes	no	yes	no
17	24	m	c	b	yes	no	yes	no	yes	no	yes	yes	no	no	no	no
18	23	m	d	a	yes	no	yes	no	no	no	yes	no	yes	no	no	no
19	25	m	c	c	yes	yes	yes	no	no	no	yes	yes	yes	no	no	yes
20	27	m	b	b	yes	yes	yes	no	no	no	yes	yes	yes	no	no	no
21	23	m	e	e	no	no	no	no	no	no	yes	no	yes	no	no	no
22	24	f	d	d	yes	yes	no	no	no	no	yes	yes	yes	no	no	no
23	24	m	c	c	yes	no	yes	no	no	no	yes	no	yes	no	no	no
24	23	m	d	b	yes	yes	no	no	no	no	yes	yes	no	yes	no	no
25	24	m	d	b	yes	no	yes	no	no	no	yes	yes	yes	no	no	no
26	23	m	d	c	yes	yes	yes	no	no	yes	yes	yes	yes	yes	no	yes
27	19	f	b	b	yes	no	yes	no	no	yes	yes	no	no	yes	no	yes
28	23	m	c	b	yes	yes	no	no	no	no	yes	yes	yes	no	no	no
29	28	m	e	e	yes	no	no	no	no	no	yes	no	no	yes	no	no
30	22	m	c	c	yes	yes	yes	no	no	no	yes	yes	yes	no	no	no
31	28	m	e	b	yes	no	no	no	no	no	yes	yes	yes	yes	no	no
32	23	f	d	b	yes	yes	yes	no	no	no	yes	yes	yes	no	no	no
33	23	f	b	b	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
34	11	f	d	d	yes	yes	yes	no	yes	yes	no	yes	yes	yes	yes	yes
35	14	f	d	c	yes	yes	yes	no	no	yes	yes	yes	yes	no	no	yes
36	11	m	d	d	yes	yes	no	yes	no	yes	yes	yes	no	yes	no	yes
37	15	f	d	d	no	yes	yes	no	no	no	yes	yes	yes	yes	yes	yes

#	5	6	7
1	pc	rpg, fps	pc
2	ps3	fps, action, adventure, hack 'n slash	pc
3	pc	adventure, fps, strategy	pc
4	xbox360	strategy/rpg	pc
5	pc	fps/soccer(sport)	pc
6	pc	science-fiction	pc
7	xbox	fps	xbox
8	mobile	casual	computer
9	computer	(R$s and tb-) strategy, fps, casual	computer
10	android	Multiplayer, Rockband	playstation
11	iPad	klikk & pek || multiplayer	mariokart
12	pc	rollespill, strategi, casual	pc
13	iOS	casual, rpg	xbox360
14	ps3	fps, strategy	console
15	ps3	fps, puzzle, adventure	pc
16	iPad	Action, fps	xbox
17	computer	puzzle, rts, rpg	computer
18	pc	puzzles, bygge-spill, hack-n-slash RPG	pc
19	computer	rpg	computer
20	computer	rts, fps	computer
21		fps, strategy	pc
22	computer	portal, cs	ps2
23	pc	puzzle, rpg	pc
24	pc	strategy, mmo, fps, adventure	pc
25	pc	rts, rpg	pc
26	ds	adventure, rpg	pc
27	computer	rpg, platform	computer
28	ps3	Platformers, Action/Adventure	pc
29	TV based Console	sports, fighting, simple as puzzle Bubble	TV console
30	pc	rpg, fps, brain-train, tactical	pc
31	pc	strategy, fps, rpg, adventure	pc
32	mobile w/ touchscreen	strategy	computer
33	pc	adventure	
34	iPad	platformspill, arcadespill, matspill	DS, iPad, iPod, PS3
35	iPad	Action, sport, platform	xbox
36	ps2	platform	ps2
37	iPad, PS3	platform, sport, action	DS

#	8
1	Morrowind, Oblivion
2	world of warcraft, god of war franchise, red dead redemption, counter strike: source, league of legends, battlefield franchise
3	portal 1+2, HL2 + ep.1,2, CoD4, Modern Warfare 2, Monkey Island 1,2,3, Tales of Monkey Island, Psychonauts, New Super Mario Bros.(DS)
4	warcraft 3, lost odyssey
5	counter strike: source, fifa 2006-011, pes 2006-2010, battlefield
6	eve-online, warhammer online, total war, civilizations
7	Call of Duty: black ops(multiplayer og sp), Battlefield, bad company 2 (mp og sp), tiger woods pga tour, left for dead 2
8	-yatzy (computer) -scrabble (mobile) -solitaire(computer)
9	TF2, RockBand
10	Rockband
11	
12	oblivion
13	The Elder Scrolls: Oblivion
14	CoD: Modern warfare 1+2, Fifa: 09-11
15	gta4, wordfeud (scrabble), assassin's creed, warlight, angry birds, team fortress
16	Halo, Gears of War, WoW
17	Diablo II, Starcraft (II)
18	diablo 2, open transport tycoon deluxe, civilization revolution, div. Flash-spill
19	World of Warcraft!, Mass effect, Saints Row 2, Final fantasy tactics A2((12)), Team Fortress 2
20	bad Company 2, Starcraft 2, civ. 5, Uncharted 2
21	starcraft, quake III
22	gta
23	Nurikabe, Evolution Soccer 4, mass effect, …
24	WoW, SC2, CoD, FFXIII
25	fallout 3, Morrowind, Oblivion
26	WoW, Pokemon
27	WoW, SuperMario(DS), Minecraft, Team Fortress 2, Crash Bandicoot(DS)
28	Team Fortress 2
29	EA fifa, Angry birds, counter strike
30	starcraft 1+2, minecraft, half-life 2, gta4, angry birds2, solitaire, slice it!
31	Team Fortress 2, Left for dead 1+2, portal 1+2, starcraft 2, warlight, xmoto, mass effect 1+2
32	World of Warcraft, tower defence
33	CoD, Age of Empires, SC2
34	Sonic Sega, allstar racing, lbp, dougnuth games
35	Little Big Planet, Lego Harry Potter, Lego Star Wars
36	
37	Little Big Planet, Sport Champion(ds move), Sega All Star racing, Lego StarWars, Lego Harry Potter

#	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	2	1	3	0	2	2	4	2	3	1	3	4	3	0	2	3	3
2	3	2	3	2	3	2	4	1	3	1	1	3	2	3	4	3	3
3	3	1	4	0	3	1	3	2	3	0	3	4	1	1	3	2	3
4	3	0	3	0	2	1	4	1	3	0	3	3	2	1	3	2	2
5	2	1	2	1	3	2	3	2	2	3	2	3	1	2	0	3	1
6	3	1	3	0	3	1	3	1	2	2	2	3	2	0	3	2	2
7	2	0	3	0	4	0	3	0	3	0	3	4	2	2	3	3	3
8	3	3	2	1	3	1	3	2	2	1	2	4	3	1	3	4	4
9	1	0	3	0	3	1	3	1	1	0	2	3	1	2	1	2	2
10	0	1	3	0	4	1	4	0	3	0	2	3	2	1	3	4	4
11	2	1	3	1	3	0	4	0	3	0	1	4	1	3	3	4	4
12	3	0	4	1	3	0	4	0	4	0	3	3	3	0	2	3	3
13	3	1	3	0	4	1	4	0	3	0	3	4	3	0	3	4	3
14	0	1	2	0	3	2	3	1	2	0	3	3	1	2	1	3	2
15	1	1	2	1	3	1	2	3	2	1	2	4	2	1	2	3	3
16	2	1	2	1	2	1	3	1	2	1	3	3	1	1	1	3	2
17	2	1	3	1	2	1	3	2	2	1	3	4	3	1	4	4	3
18	4	0	4	0	4	0	4	1	4	0	4	4	2	2	1	4	3
19	2	1	3	0	3	1	4	0	1	1	2	3	1	2	1	3	3
20	2	1	3	0	3	0	3	1	4	0	2	2	4	1	2	3	3
21	2	0	3	1	4	1	1	0	3	1	4	4	3	0	3	2	3
22	3	1	1	1	1	2	3	3	1	1	3	1	3	1	1	3	1
23	1	1	3	1	3	2	3	1	3	1	2	3	1	3	1	3	2
24	2	1	3	0	2	1	3	2	3	1	1	2	2	2	2	2	2
25	3	1	3	1	0	0	3	1	3	0	3	3	2	0	3	2	2
26	2	0	3	1	3	1	4	1	2	1	3	4	1	2	1	3	2
27	3	1	3	0	4	0	3		3	0	2	4	2	1	3	2	2
28	2	0	4	0	3	0	4	0	4	0	2	3	2	2	3	2	2
29	2	0	3	1	3	0	2	1	3	1	3	3	2	3	2	3	3
30	3	1	3	1	2	2	4	0	2	0	2	4	4	0	4	4	4
31	0	0	3	0	3	1	4	0	1	0	4	4	2	0	2	2	3
32	3	1	3	1	3	1	4	0	2	1	2	3	3	1	4	4	4
33	3	3	1	2	1	2	1	3	1	2	3	4	2	4	3	3	3
34	2	0	3	1	2	2	4	0	4	0	4	3	4	1	3	4	4
35	2	1	3	0	2	2	2	0	4	0	3	4	2	2	2	4	4
36	4	0	4	0	4	0	4	0	4	0	3	4	4	4	4	4	4
37	2	1	3	1	3	1	2	1	2	1	3	4	3	1	3	3	3

#	26
1	Ikke helt intuitivt hvordan man lagrer og spiller brettet med en gang, kunne vært muligheter for mer avanserte leveler. Men gikk fort å finne ut hvordan alt fungerer.
2	no "move object function". Easy to select objects. Good overview. No zoom buttons
3	Cumbersome: -placing balls/points -deleting Missed textual icons
4	like: easy to use
5	Some levels where not the same when editing as in gameplay. Lower grass was not visible f.eg.
6	like: simplicity, but option to make complex stuff with it; dislike: more fancy features
7	+likte gui-et veldig godt +morsomt konsept -det jeg "gjorde" (f.eks plukka poeng) i testmodusen til editoren ble værende. -savnet pinch-to-zoom når jeg så meg rundt -hadde likt uendelig eller betydelig mer plass til å bygge på. -flere "hindere" -skulle gjerne valgt om pointsa skulle stå stille eller bevege seg i edit mode
8	touch-screen gjorde det enkelt å lage levels; enkel funksjonalitet
9	+easy to use +clean looking +many possibilities -could have been more explanations available. E.g. Difference between point marbles and pointless marbles
10	var noen veldig flotte gni-funksjoner som var behagelig å bruke; var lystbetont spill
11	tok litt tid å finne lagre-funksjonen
12	Veldig lett å teste brett man utvikler, raskt og effektivt.; hadde litt problemer med menyen intill jeg skjønte fargemarkeringen
13	kunne brukt tekst i tilleg til ikoner for å vise hva elementene gjør
14	-pointless is very pointless -not a lot of options +easy to start and test
15	*To easy to mess up (delete/destroy contraption) *Not always intuitive
16	Lite hjelp: informasjon

#	26
17	styre rundt om man skulle utafor en skjerms størrelse
18	Mystery Meat Navigation; "ok"-tast i menyene i stedet for enter; restart-button trengs; musikk i editor trengs
19	Litt upraktisk skjermscrolling
20	Likte: God oversikt over brettet. De fleste ikoner lett å skjønne. Mislikte: Endte opp med å slette et brett da jeg trodde deleteknappen var cancel og gå tilbake.
21	-scrolling med to fingre -blinking av kaninen
22	lite intuitiv, men lett når man skjønte hvordan
23	Does not seem like it can offer enough variety for now
24	Like: easy to use, short amount of time needed to play; dislike: few options, I have to be very creative to make a cool game
25	kanskje litt begrensede artifakter
26	I liked the simplicity. Took some time to understand and recognize some of the tool glyphs. Struggled with delete.
27	I liked the simplicity, …
28	tooltip for the icons
29	the icons could be confusing
30	dislike: preview, a little bit few things
31	flytte tiles
32	Enkel å sette seg inn i. Ikonene ga mening. Litt dumt at man ikke visste f.eks hvor høyt kaninen hoppet og hvordan gjenstandene virket. Men dette skjønner man fort når man spiller spillet.
33	
34	
35	
36	
37	

#	27a	27b	27c	28	29	30	31	32
1	yes	yes	no	yes	yes	no	yes	no
2	no	no	yes	no	yes	yes	yes	yes
3	yes	no	yes	yes	yes	yes	no	yes
4	yes	no	yes	yes	yes	yes	no	yes
5	yes	no	no	yes	no	yes	no	no
6	yes	no	yes	yes	yes	yes	no	yes
7	yes	yes	no	yes	no	no	yes	no
8	yes	no	yes	no	yes	yes	no	no
9	no	no	yes	yes	yes	yes	no	no
10	yes	yes	no	no	yes	yes	yes	no
11	no	no	yes	yes	yes	yes	no	yes
12	yes	yes	yes	yes	yes	no	yes	yes
13	yes	no	yes	yes	yes	yes	no	no
14	no	no	yes	yes	yes	yes	no	yes
15	yes	yes	yes	yes	yes	yes	yes	yes
16	no	no	yes	yes	yes	yes	no	no
17	no	no	yes	yes	yes	yes	no	no
18	yes	yes	yes	yes	yes	yes	yes	no
19	yes	no	no	yes	yes	no	yes	yes
20	yes	no	yes	yes	yes	yes	no	no
21	yes	no	yes	yes	yes	yes	yes	yes
22	yes	no	no	no	yes	no	no	no
23	yes	no	no	yes	no	yes	no	no
24	yes	no	yes	yes	yes	yes	yes	yes
25	yes	yes	yes	yes	yes	yes	yes	yes
26	no	no	yes	no	yes	yes	yes	no
27	yes	yes	no	yes	no	no	yes	no
28	yes	no	no	yes	yes	yes	no	no
29	yes	yes	no	yes	yes	yes	yes	no
30	yes	yes	yes	yes	yes	yes	yes	yes
31	yes	no	yes	yes	no	yes	yes	no
32	yes	yes	no	no	yes	no	yes	no
33	yes	yes	yes	yes	yes	yes	no	yes
34	yes	yes	yes	no	yes	yes	no	no
35	yes	yes	yes	yes	yes	yes	yes	yes
36	no	no	yes	yes	yes	yes	no	no
37	yes	yes	yes	yes	yes	yes	yes	yes

#	33
1	
2	scrolling tools
3	clear all poits, clear all/new level
4	to be able to move obstacles that are already placed
5	
6	gravity fields/ hover fields, death traps, chasms(hidden), water, etc.
7	
8	
9	
10	
11	*flere typer blokker *kunne dra en hel linje (ble litt hullete)
12	fiender
13	
14	enemies or floating points
15	Pause/reset (w/o explicit load)
16	
17	
18	
19	Litt flere platformelementer. Fiender, fallgruver, etc.; Undo-funksjonalitet; Multitouch scrolling/zooming
20	
21	poeng"kuler" som ikke beveger seg; farlige ting
22	
23	
24	more items, events, options
25	trigger editor
26	
27	
28	
29	
30	trap, glas/ice, slowing fluid, enemies
31	
32	
33	play mode
34	
35	Jeg savnet "monster" som du mister liv av.
36	
37	undo/redo button, dødelige hinder, av/på-knapper

#	34
1	
2	zoom and move mode/button as aforementioned
3	some nice building music maybe
4	different "brush"sizes. To enable placing several tiles at the same time
5	
6	quicksave/quickload
7	flere hindere. For eksempel dings som flipper om gravitasjonen osv.
8	nederste ikon var vanskelig å forstå
9	Maybe a "intro" feature for explaining things. Perhaps a "?" mark "mode".
10	
11	
12	is for å få kaninen til å skli, bakker som forsvinner over tid
13	konsistens ved sletting av start/slutt i likhet med andre elementer me slett-knappen
14	
15	*paint bak joystick? *rar startpos for start/mål *confirm delete (yes/no) *kryss-ikon vanskelig ă forstă
16	Noen av ikonene passet ikke den funksjonen de hadde. Litt vel simplistiske ikoner
17	
18	sp.m. 26
19	
20	Under verktøy-fanen, bruk navn i stedet for ikoner. (save, load, delete, osv.)
21	plasserig av flagget helt til høyre
22	
23	
24	
25	
26	some of the glyphs could be a bit more clear. Perhaps a tutorial mode?
27	
28	
29	
30	preview needs to be reset, points are dissappearing from the editor
31	pinch to zoom
32	
33	*en pil eller henvisning til at det er en kanin *flagg=mål *mann=start *mer beskrivende at man er I buildmode og ikke I playmode?
34	
35	Noe annet enn et øye "to scroll"
36	
37	kanskje en hånd I stedet for øye "to scroll" og trampolinen hadde kanskje litt for stor sprett

#	35	36
1	yes	mulighet for mer komplekse leveler
2	yes	text
3	no	
4	no	
5	yes	to add enemies / be able to die; enemies would be both static and moving
6	yes	savegame management, option to save/load to/from a list of savegames for each level.
7	yes	-en "are you sure you want to delete" -mulighet til å dele brettet med venner/omverdenen
8	yes	mulighet for å fjerne elementer ((nederste ikon))
9	yes	An easy way of sharing levels with friends. It's more fun when you can share levels.
10	yes	mulighet for skråplan i gresset
11	yes	mini-tutorial / eller smă piler som sier -> lagring her el.
12	no	
13	no	
14	no	
15	yes	Level restart
16	no	
17	no	
18	yes	coop play, knapper for å åpne dører
19	yes	Konfigurerbar respons på helling. Default som krever ganske stor vinkel for ă fă kaninen opp i god fart
20	no	
21	no	
22	no	
23	no	
24	no	
25	yes	triggers
26	no	
27	no	
28	no	
29	yes	more building blocks to make more complex levels
30	yes	more things to do
31	no	
32	no	
33	no	
34	yes	flere bakketyper, vann?
35	yes	Monster, forskjellige "bakker" (ikke bare gress).
36	no	
37	yes	undo knapp, pigger/dødelige hinder, evt. Av/på-knapp

#	37
1	
2	more text explaining the different buttons and better score system
3	it's a bit simple more features on a story/fixed levels would be nice probably
4	the rabbit's movement was a bit unstable when jumping alongside a wall.
5	add a help menu, f.eg. To explain the view modes in editor
6	
7	
8	*litt vanskelig å forsta forskjell på ikon som delete/load/save/exit. Burde enten hatt tydeligere ikoner eller hatt tekst ved siden av. (spelielt load/save) *Synes ikke øyet var et veldig tydelig symbol, men noe man lett forstår etter å ha testet funksjonaliteten en gang * samle ikonene i toppen av skjermbildet -her er det mye ledig plass. kom ved flere tilfeller borti ikonene når jeg skjulle bygge * help kunne vært fint å ha med i hovedmenyen. Evt. Instruction *Hadde litt problemer med å lage ny playground. Ikke intuitivt å måtte trykke på return
9	
10	tidtaker som vises mens man spiller
11	
12	
13	
14	overpowered trampolin; its possible to get stuck, thats no good; should be able to restart
15	
16	

#	37
17	
18	-Fikse flickering -bounding boxes er litt for store i forhold til sprites, så man kan stå utenfor kantene
19	
20	power-ups; dører+nøkler
21	Huske at man har trykket på ((tilte-symbol))
22	kunne vært enkle instrukser om bruk. Burde også vært lettere å starte spillet.
23	
24	After writing the name of the new level, one should auto step into the game
25	
26	Ability to download maps from other players.
27	
28	
29	
30	easier buttons; hard to understand view and preview; remove fps counter :-p
31	aliens og skytere
32	
33	*mer beskrivende ikoner? *forskjellig lyd til byggetingene *flagg="mål"-lyd *start="start"-lyd ol.
34	du får vite hvor mange poeng du har fått
35	Trampolina skulle kanskje ikke hoppe så høyt. (Ha flere muligheter med høyde) Miste live hvis man detter ned I hull eller blir tatt av monster. Poengsum bør vises. "Start over"-knapp.
36	At vi kunne spille online og multiplayer med andre
37	noe mer å bruke pointless-greiene til, evt. Til å skru av/på en mekanisme som f.eks flyttbare platformer. En "start over"-knapp I spillemodus I tilfelle man blir sittende fast. Litt dramatisk musikk til å være stort sett uten farer? Miste liv/"dø" hvis man detter ned der det ikke er gress. Siden man kan få tak i poeng, bør det vises hvor mye poeng(evt. av x mulige) når banen er fullført.

#	38	39	40	41	42	43	44	45	46	47
1	2	0	2	0	2	0	1	1	4	4
2	2	1	3	2	1	1	3	2	3	4
3	1	1	2	2	1	0	3	2	3	4
4	1	2	2	2	3	1	4	2	4	4
5	3	1	2	0	1	2	3	1	3	4
6	1	0	2	2	2	2	4	1	3	3
7	3	0	3	0	2	3	1	3	4	4
8	3	1	1	0	3	2	2	4	4	4
9	0	1	1	1	1	1	3	1	3	3
10	1	0	2	0	2	0	1	1	3	4
11	2	4	2	1	3	0	4	1	1	4
12	0	2	2	3	0	0	2	4	2	4
13	0	0	3	1	0	0	4	2	3	4
14	1	3	2	1	3	1	3	1	2	3
15	0	1	2	0	3	0	3	0	3	3
16	2	2	2	2	1	1	3	1	3	3
17	0	3	4	0	0	4	4	0	4	4
18	2	4	3	0	4	0	4	0	3	4
19	3	0	4	0	1	1	2	3	3	3
20	0	1	2	0	0	0	2	1	3	3
21	2	2	4	2	1	2	2	3	4	3
22	3	2	2	2	2	2	2	3	3	3
23	2	2	2	2	2	2	3	2	2	3
24	1	0	1	0	1	0	2	2	3	3
25	2	1	2	1	3	0	3	1	3	3
26	2	2	2	1	1	0	3	2	4	4
27	1	1	1	0	3	1	1	3	3	4
28	1	2	1	0	1	0	3	1	3	3
29	2	2	3	2	1	0	3	1	3	3
30	0	1	4	2	0	0	4	0	4	4
31	0	3	2	1	2	0	3	0	4	2
32	1	1	3	0	2	2	2	3	3	4
33	1	1	2	1	2	1	3	3	1	3
34	2	2	2	2	3	2	3	0	2	3
35	2	1	3	1	2	1	3	2	4	3
36	0	1	0	0	4	0	4	2	4	4
37	1	0	1	0	1	0	3	3	3	4

References

Answerbag. Definition of Platform game. http://www.answerbag.com/t_view/269, 2003. URL http://web.archive.org/web/20070211211612/http://www.answerbag.com/t_view/269. [Online; accessed 11-Feb-2007].

iPad - Technical Specifications. Apple, 2009. URL http://liveweb.archive.org/http://support.apple.com/kb/SP580.

Custom Icon and Image Creation Guidelines. Apple, 2010. URL http://web.archive.org/web/20101203210326/http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/mobilehig/IconsImages/IconsImages.html.

Victor Basili. The experimental paradigm in software engineering. In H. Rombach, Victor Basili, and Richard Selby, editors, *Experimental Software Engineering Issues: Critical Assessment and Future Directions*, volume 706 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin / Heidelberg, 1993. URL http://dx.doi.org/10.1007/3-540-57092-6_91.

Jonathan Blow. Game development: Harder than you think. *Queue*, 1:28–37, February 2004. ISSN 1542-7730. doi: http://doi.acm.org/10.1145/971564.971590. URL http://doi.acm.org/10.1145/971564.971590.

François Bérard. The perceptual window: Head motion as a new input stream. In *Proc. Interact99, Edinburgh, A. Sasse &; C. Johnson Eds, IFIP IOS Press Publ*, 1999. 238 pages.

David Buckingham and Andrew Burn. Game literacy in theory and practice, 2007.

Kate Compton and Michael Mateas. Procedural level design for platform games. In *Proceedings of the 2nd Artificial Intelligence and Interactive*

Digital Entertainment Conference (AIIDE), pages 109–111, Marina del Rey, California, June 2006.

Magy Seif El-Nasr and Brian K. Smith. Learning through game modding. *Comput. Entertain.*, 4, January 2006. ISSN 1544-3574. doi: http://doi.acm.org/10.1145/1111293.1111301. URL `http://doi.acm.org/10.1145/1111293.1111301`.

Box2D Physics Engine. Erin Catto and the Box2D community, 2011. URL `http://web.archive.org/web/20090630020943/http://box2d.org/`.

Joachim Froholt. Intervju: Minecraft og fremtiden. `http://www.gamer.no/artikler/73142/intervju-minecraft-og-fremtiden/`, 2010. URL `http://liveweb.archive.org/http://www.gamer.no/artikler/73142/intervju-minecraft-og-fremtiden/`. [Online; accessed 9-June-2011].

Malcom Gladwell. Malcolm Gladwell on spaghetti sauce. `http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/`, 2004. [Online; accessed 6-June-2011].

Icon Design Guidelines, Android 2.0. Google, 2011. URL `http://web.archive.org/web/20101203061340/http://developer.android.com/guide/practices/ui_guidelines/icon_design.html`.

Sal Humphreys, Brian Fitzgerald, John Banks, and Nic Suzor. Fan-based production for computer games: user-led innovation, the "drift of value" and intellectual property rights. *Media international Australia incorporating culture policy*, 114:16–29, 2005. URL `http://search2.scholarsportal.info.myaccess.library.utoronto.ca/ids70/view_record.php?id=1&recnum=22&SID=760717fafbf1a1c5a4539da053640d93`.

Garnett Lee. Sony confirms PlayStation Network hacked. `http://www.shacknews.com/article/68215/sony-confirms-playstation-network-hacked/`, 2011. URL `http://liveweb.archive.org/http://www.shacknews.com/article/68215/sony-confirms-playstation-network-hacked`. [Online; accessed 9-June-2011].

James Lewis and Jeff Sauro. The factor structure of the system usability scale. In Masaaki Kurosu, editor, *Human Centered Design*, volume 5619

of *Lecture Notes in Computer Science*, pages 94–103. Springer Berlin / Heidelberg, 2009. URL http://dx.doi.org/10.1007/978-3-642-02806-9_12.

Andrzej Lingas. The power of non-rectilinear holes. In *Proceedings of the 9th Colloquium on Automata, Languages and Programming*, pages 369–383, London, UK, 1982. Springer-Verlag. ISBN 3-540-11576-5. URL http://portal.acm.org/citation.cfm?id=646236.682869.

Simon Egenfeldt Nielsen, Jonas Heide Smith, and Susana Pajares Tosca. *Understanding Video Games: The Essential Introduction*. Routledge, new edition edition, February 2008. ISBN 0415977215. URL http://www.worldcat.org/isbn/0415977215.

St.meld. nr. 14 (2007-2008), Dataspill, 2007-2008. The norwegian ministry of culture.

Kumar Sanjeev, Chhugani Jatin, Changkyu Kim, Daehyun Kim, Nguyen Anthony, Dubey Pradeep, Bienia Christian, and Youngmin Kim. Second life and the new generation of virtual worlds. *Computer*, 41(9):46 –53, sept. 2008. ISSN 0018-9162. doi: 10.1109/MC.2008.398.

Jeff Sauro. Measuring Usability - Quantitative Usability, Statistics & Six Sigma. 2011. URL http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software. [Online; accessed 6-June-2011].

Noor Shaker, Georgios Yannakakis, and Julian Togelius. Towards automatic personalized content generation for platform games.

Penelope Sweetser and Peta Wyeth. *GameFlow: A Model for Evaluating Player Enjoyment in Games*. The University of Queensland, St Lucia, Australia, 2005. URL http://portal.acm.org/citation.cfm?id=1077253.

Gregory Taefay. *Casual Game Design: Designing Play for the Gamer in All of Us*, 2010.

cocos2d for iPhone. Zynga and the cocos2d community, 2011. URL http://web.archive.org/web/20100621051008/http://www.cocos2d-iphone.org/.