NTNU – Trondheim
Norwegian University of
Science and Technology

Pyception

Teaching Python with a Serious Game

Kristian Laskemoen

Master of Science in Informatics
Submission date: June 2013
Supervisor: Alf Inge Wang, IDI

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

Abstract

This thesis set out to study how an online serious game could affect users' motivation on learning Python. One of the projects core goals is to find out whether learning Python is more motivating when having an effortless start through a web based game. A second goal is to find out if Python as a programming language are well suited for a serious game.

After the development and implementation of the game, it was performed a user experiment in order to receive feedback. Data from this user experiment were collected in a questionnaire and the results regarding usability were collected in a System Usability Scale.

The results were positive, and indicates that most users were satisfied with the game. Feedback from users indicate that the game was both fun and educational, and more motivating than traditional task solving. However, some changes need to be applied in order to publish this serious game.

II

Sammendrag

Denne masteroppgaven har gjort en studie på hvordan et nettbasert "serious game" kan påvirke brukerenes motivasjon, hva gjelder å lære Python. Et av prosjektet hovedmål, er å undersøke hvorvidt det er mer motiverende å lære Python med et nettbasert spill, som er enkelt å komme i gang med. Et annet mål har vært å finne ut om Python som programeringsspråk, er egnet for bruk i et "serious game".

Etter utvikling og implementasjon av spillet, ble det utført en brukertest for å motta tilbakemeldinger. Data fra brukertesten ble samlet i et spørreskjema, og resultatene hva gjaldt brukervennlighet ble samlet i et "System Usability Scale"-skjema.

Resultatene var positive, og indikerer at de fleste brukeren var fornøyd med systemet. Tilbakemeldingene fra brukerene tyder på at spillet var både morsomt og lærerikt, og også mer motiverende enn tradisjonell oppgaveløsing. På tross av dette er det fortsatt enkelte aspekter ved spillet som må forbedres før spillet kan publishers for allmennheten.

Acknowledgements

This thesis is the result of the course *IT3901 - Informatics Postgraduate Thesis: Software* at the Department of Computer and Information Science, under Faculty of Information Technology, Mathematics and Electrical Engineering, at the Norwegian University of Science and Technology.

I would like to thank my supervisor Alf Inge Wang for guidance and feedback during the past year. His motivation has been an inspiration throughout the project.

I would like to thank Anne Harnæs Sivesind for continued support, suggestions and proofreading.

I would like to thank Magnus Blütecher Dysthe for creating the graphics of Pyception.

And finally I would like to thank everyone who participated in my prototype testing, and othervise aided me in the making of this thesis.

Trondheim, 1st of June, 2013.

Kristian Laskemoen

Contents

I Introduction **1**

1 Introduction **3**
1.1 Project Context . . . 3
1.2 Motivation . . . 3
1.3 Problem definition . . . 3

II Research Design **5**

2 Research Questions **7**

3 Research Method **9**
3.1 Literature Review . . . 9
3.2 Developing a Prototype . . . 9
3.2.1 Chosen Research Method . . . 10
3.3 Testing . . . 10
3.3.1 Testing in the Development Cycle . . . 11
3.3.2 User Experiment . . . 11
3.3.3 System Usability Scale (SUS) . . . 11

III Prestudy **13**

4 Chosen Technology **15**
4.1 Python . . . 15
4.2 Django . . . 15
4.2.1 Model Template View . . . 16
4.3 MySQL . . . 16
4.4 JavaScript . . . 16
4.5 JSON . . . 16
4.6 EaselJS . . . 17
4.7 HTML . . . 17
4.8 CSS . . . 17
4.9 Technology Summary . . . 17

5 Previous Work **19**
5.1 Educational Games . . . 19
5.2 State of the Art . . . 19
5.2.1 Khan Academy . . . 19

5.2.2 Codecademy . . . 20
5.2.3 Code Hero . . . 21
5.2.4 Robocode . . . 21
5.2.5 hACKME game . . . 22
5.2.6 Key Features . . . 22
5.3 Game genres . . . 22
5.3.1 Real Time Strategy . . . 22
5.3.2 Point and Click Adventure . . . 23
5.3.3 Role Playing Games . . . 23
5.4 Syllabus of TDT4110 . . . 23

IV Own Contributions 25

6 Concept 27
6.1 Sketches . . . 27
6.1.1 Real time strategy . . . 27
6.1.2 Point and click adventure . . . 28
6.2 Sketch Evolution [notes] . . . 29
6.3 The Final Concept . . . 30
6.3.1 Game Story . . . 30
6.3.2 Playing Field Elements . . . 31
6.3.3 Code Interface . . . 32
6.3.4 Game objective . . . 33
6.3.5 Key Features . . . 33
6.4 Syllabus . . . 34

7 Requirements 35
7.1 Functional Requirements . . . 35
7.2 Non-functional Requirements . . . 37

8 Design Choices 39
8.1 Architecture . . . 39
8.1.1 Server Side Execution . . . 39
8.1.2 Client Side Execution . . . 39
8.1.3 Summary . . . 40
8.2 Game Engine . . . 40
8.2.1 Crafty . . . 40
8.2.2 EaselJS . . . 40
8.2.3 Unity . . . 41
8.2.4 Summary . . . 41

9 Architecture 43
9.1 Physical View . . . 44
9.2 Logical view . . . 44
9.2.1 Game Server . . . 44
9.2.2 Game client . . . 45
9.3 Development View . . . 47
9.4 Process view . . . 47
9.4.1 Server . . . 47

9.4.2 Game client 48
9.5 Scenarios 49

10 Implementation **53**
10.1 Milestones 53
10.2 Graphics 54
10.3 Exercises 54
10.4 Security 56
10.5 Challenges 56

11 User Experiment **59**
11.1 Test Framework 59
11.1.1 Users 59
11.1.2 Goal 59
11.1.3 Environment 60
11.2 Execution 60
11.2.1 Eventualities 61
11.3 Questionnaire 61
11.3.1 Calculation of Score 62
11.4 Success Critera 63

V Evaluation **65**

12 User Experiment Results and Analysis **67**
12.1 Execution 67
12.1.1 Users 67
12.1.2 Environment 68
12.1.3 Challenges 68
12.2 Results 68
12.2.1 G1 Is the game fun ? 69
12.2.2 G2 Is the Game Educational? 71
12.2.3 G3 Do Pyception feel more like a game than programming tasks? 73
12.2.4 G4 Is it easy to get started with the game? 75
12.2.5 G5 Is the game motivating? 76
12.3 System Usability Scale 78
12.4 Success Criteria 78
12.4.1 SC1 - Fun and Educational game 78
12.4.2 SC2- Motivation 79
12.4.3 SC3 - Usability 79

13 Difference between a novice and an experienced coder **81**

14 Technical Analysis **83**
14.1 Architectural issues 83
14.1.1 Security 83
14.1.2 Performance 83
14.2 Python Issues 84
14.3 Existing Solutions 84

14.4 Summary 84

15 Requirement fulfillment **85**
15.1 Functional Requirements 85
15.1.1 Game 85
15.1.2 Playing Field 86
15.1.3 Code Interface 87
15.1.4 Lecturer 88
15.2 Non-functional Requirements 88

VI Conclusion **91**

16 conclusion **93**
16.1 Research Questions 93
16.1.1 Research Question 1 93
16.1.2 Research Question 2 93
16.1.3 Research Question 3 94
16.1.4 Research Question 4 94
16.2 Project Evaluation 95
16.2.1 Pre-study 95
16.2.2 The Engineering Method 95
16.2.3 The User Experiment 95

17 Future Work **97**
17.1 Finalizing Pyception 97
17.1.1 Security 97
17.1.2 Settings 97
17.1.3 Full Browser Support 98
17.1.4 Exercises 98
17.2 Improving Pyception 98
17.3 Research Improvements 98

Appendix A Acronyms **101**

Appendix B Installation Guide **103**
B.1 Installation 103
B.2 Uninstallation 104

Appendix C Questionnaire **105**

List of Figures

4.1 Visualisation of the relationship between the technologies. 18

5.1 Screenshot of the Knowledge map. December 2012 20
5.2 Screenshot from Codecademy of the coding interface. The task at hand is described in the column to the left, and the right column is where the user code. 21

6.1 Mockup example of coding in a real time strategy game 28
6.2 Mockup of an adventure where the player explores a library, and has found a computer terminal 29
6.3 Inspiration for a top down Role Playing Game (RPG), from gamefroot 30
6.4 The game sketch of Magnus Blütecher Dysthe 31
6.5 Screenshot of the game as it is today. The code exercises are accessible by docking at any of the satellites 32
6.6 From left to right: The player spaceship, an unsolved satellite, a solved satellite, the power station and Dr.Thereon 32
6.7 A screenshot of the code interface 33

9.1 Illustration of the 4+1 view model. 43
9.2 All of the software in Pyception is deployed on the game server, however it provides web access to players and lecturers 44
9.3 Class diagram alt1. 45
9.4 Class diagram game client. 46
9.5 Pyception's two packages and their relationship 47
9.6 insert caption 48
9.7 insert caption 48
9.8 Third person view of the gameplay. 51

10.1 Screenshot of the coding interface showing an easy task from level 1 55
10.2 Screenshot of the coding interface showing a difficult task from level 4 56

12.1 Chart displaying the skills possessed by the users. 67
12.2 Chart displaying the motivation of the users. 67
12.3 G1 69
12.4 G2 71
12.5 G3 73
12.6 G4 75

12.7 G5 76

List of Tables

7.1 Functional requirements Game 35
7.2 Functional requirements Playing field 36
7.3 Functional requirements Player Coding 36
7.4 Functional requirements Lecturer 36

11.1 The research questions and their derived goals. 60
11.2 Goals and their relation to the questions of the questionnaire listed in Appendix C. 61

12.1 Questions related to G1 69
12.2 Goal ID 2 and the related questions 71
12.3 Goal ID 3 and the related questions 73
12.4 Goal ID 4 and the related questions 75
12.5 Goal ID 1 and the related questions 76
12.6 Pyception SUS score 78
12.7 SC1 summary 79
12.8 SC2 - summary 79

13.1 The average score of experienced and novice coders, and the differences 82

Part I

Introduction

Chapter 1

Introduction

1.1 Project Context

Over the course of the last 30 years, computers have evolved from being a work station available only for business situations, via home and personal computers, to laptops and mobile devices. The one techology that is in center of this evolution is the internet, which connects computers together in a world wide web. Today, it is almost impossible to live without access to one or more computers as more and more services, such as banks and maps, are available through internet. The evolution of the service infrastructure is driven forward by persons who know the languages that computers understand, among them Python. The demand for coders capable of producing quality code is formidable and the first step towards becoming a coder, is learning how to progam.

1.2 Motivation

Prior to selecting a thesis, my motivation was "to develop something cool". I wanted to my thesis project to include development of some software, preferably something that would not end up on a shelf and never be looked at again.

The supervisors of my institute publish thesis proposals online on a system called DAIM[1]. While browsing the thesis proposals, one in particular caught my interest. Alf Inge Wang, proposed to create a serious game where the player would learn how to code Python by playing the game. I immediately like the proposal, and spent some time reflecting over it.

Inspiered by the success of Khan Academys[2] online math course, I proposed to create the serious game as a web service. In addition I proposed that the game should require no installation process, in order to create an effortless start.

1.3 Problem definition

Learning Python today requires a serious effort from the user to set up the environment in which the learning takes place. Different versions of python are

[1] http://daim.idi.ntnu.no/

[2] https://www.khanacademy.org/exercisedashboard

available, making the process of finding and downloading the correct interpreter software more complex. On the windows platform, in order to use the interpreter and check if the installation was a success, the user is required to manually set the path variable. In order to change the path variable it is necessary to navigate through 6 menus in the control panel. With the path variable is set the user need a tool to develop in, preferably an Integrated Development Environment (IDE). A quick Google search on "Python IDE" gives an abundant amount of suggested tools. All the steps and choices are irrelevant in regard to the goal, Learning Python, and only serves to increase the difficulty of the first step towards learning. Finding a way to reduce the initial effort, and let the user focus on the important part, learning to program, is an considerable part of this project.

During the last decade, serious gaming and Gamification has received lots of attention, and is making its way to the consumers. Researchers point to several advantages compared to traditional education, among them motivation and individual learning pace. Bringing these positive traits along in the process of learning the programming language Python is important in this project.

Part II

Research Design

Chapter 2

Research Questions

This chapter intends to list the research questions of this thesis, and briefly elaborate. Based on an analysis of my problem description the following research questions are deducted.

RQ1 Which systems related to Pyception exists today, and do they exhibit any features that should be included in Pyception?
Serious gaming is not a new concept, and educational gaming is even older. It is important to learn from the projects that already exist in that field, and bring positive traits along in Pyception.

RQ2 Is it possible to create a serious game based on programming that is both fun and educational?
All game types may be considered educational, but is it possible to learn a player how to code without removing the fun aspect of a game?

RQ3 Will more people be motivated to learn programming if the learning process has an effortless start through a web based game?
The traditional approach of learning a programming language is to download and install software related to the preferred language. Then the user needs a code environment suited to the programming language, and usually some settings are required before the programming can start. Elimination of this threshold, along with the added factor of fun, may be a combination that gets more users through the declarative learning phase.

RQ4 Is the Python language a suitable language for a web based serious game?
The Python language first appeared in 1991 and has since been released in several vesions, the newest being 3.3.1. Has the language matured enough to support the features necessary for a secure and functional web based serious game?

Chapter 3

Research Method

Answering the research questions defined in Chapter 2, require various approaches. A literature review is requiered to answer RQ1, while RQ2 and RQ3 require user testing of a prototype. RQ4 will be answered through a combination of the literature review and an analysis of the prototype,

3.1 Literature Review

The Pyception project goal is to develop a prototype of a serious game teaching Python. In order to create a basis for the development of a prototype, it is necessary to review existing solutions that are related to the project. Games and gamified services that teach mathematics, programming or computer science will be the basis for the literature review.

3.2 Developing a Prototype

A resonable amount of research have been conducted in software engineering. People are developing tools, methods, technologies etc. However, numerous projects claiming to be research are simply development projects as they fail to thoroughly test and validate the research topic. Basili defines resarch as follows: "Research involves gaining understanding about how and why a certain type of tool might be useful and by validating that a tool has certain effects by carefully designing an experiment to measure the properties or to compare it with alternatives." [3] In order to qualify as a research activity and not development, Basili presents several research paradigmes to which an activity must cohere.

1. **The Scientific Method:** This approach is based on observations of the software leading to a proposal of a theory of improvement. The proposed theory is tested by measuring and analyzing, and if possible repeat the procedure. [3]

   The scientific method is an inductive paradigme best used in order to gain an understanding of the situation. It thoroughly analyse a complex situation, where many factors may contribute to strengths and weaknesses. With focus on one tool or process, it is possible to asses its role and effects in the situation.

1.1 **The Engineering Method:** Start by examining existing solutions and propose an improved solution. Develop or implement the suggested solution and perform an analysis of the new solution. Repeat the development process until no improvements appear to be feasable. [3]
The engineering method differs from the scientific method by developing a product. The product development is an iterative process where the product is subject for testing and analysis in each iteration, and then improved accordingly. The iterative cycle continues until the product satisfy all the proposed requirements.

1.2 **The Empirical Method:** Propose a model, develop statistical and qualitative methods and apply to case studies. Measure, analyze and validate the model and repeat the process. [3]
The empirical method aims to find new solutions by suggesting a new model not necessarily based on a previously existing model. This allows the method of finding revolutionary models. In order to validate the suggested model, it is subject to analysis of effects on the process or tool. It is not enough to simply suggest a new model without proof that exceeds existing models or tools.

2. **The Mathematical Method:** This approach is started by proposing a formal theory or a set of axioms. From the formal theory or axioms, a theory is developed and results are derived. If possible, compare results with empirical observations. [3]

   The mathematical method is a deductive analytical model, and does not require experimental design in the statistical sense. Developing models and understanding boundaries is based on manipulations of the model through an analytical framework that the method provides.

3.2.1 Chosen Research Method

Based on the research processes suggested by Basili, this project will follow the engineering method. After the basis for a prototype is established through the literature review, the prototype development will start in a cycle consisting of development, demonstration and analyis. As the project is limited in time to one school year, Basilis goal of reaching a state where no improvements appear feasable, is not likely to occur. In the allocated time frame, the project will strive to produce a best possible serious game and leave a working prototype with a record of suggested improvements. As the time span approach the end, the prototype will be completed and tested on a sizable crowd. Their reception and feedback will be subject to analysis for the report detailing possible future work.

3.3 Testing

Experiments are necessary to gain knowledge related to the project, but experimentation alone is of no value if there is no underlying framework where experimantal results can be interpreted. [3] During the development cycles, quick feedback is important in order to spend most of the time developing the

prototype. At the end of the project, a more thorough analysis of the functional prototype is requirerd as to determine the success of the project.

3.3.1 Testing in the Development Cycle

The development cycle consists of three parts, development, demonstration and analysis. Each iteration will start by producing some new software features or modifying existing ones. Then the features will be presented to my supervisor, and in cooperation, we perform an analysis of what is good, what needs more work and what to focus on in the next iteration of software development.

3.3.2 User Experiment

The user experiment will be designed to let a greater audience test the prototype and assess the usability. Their feedback will be measured with the help of questionnaires and personal feedback. The voluntary test users will have access to the prototype for a limited time. By using their existing knowledge in gaming and programming, they are free to play the game as they wish. After the test users have finished the game or played for a while and feel done, a questionnaire is provided for them to fill out.

The questionnaire consists of two parts were the first part is a custom designed and the second part is a SUS form. The first questionnaire will gather information regarding the technical aspects, game experience and mastery. It is important that the questionnaire enlighten the educational aspect of Pyception, as it is what separates serious games from games.

3.3.3 System Usability Scale (SUS)

The System Usability Scale form is a likert scale with 10 questions used to assess usability. The usability of a system, as defined by the ISO standard ISO 9241 Part 11, can be measured only by taking into account the context of use of the system [23]. Despite this, there is a need for broad general measures which can be used to compare usability across a range of contexts. In addition, there is a need for "quick and dirty" methods to allow low cost assessments of usability in industrial systems evaluation [24]. From a completed SUS form, a SUS score in the range 0 - 100, is calculated and gives a general overview of the usability of the system.

Each of the ten questions will by the user be rated between 1 and 5, where 1 represents "Strongly disagree" and 5 represents "Strongly agree". The computation of the total score is done by subtracting 1 from the score of every odd question, and the even questions are calculated by taking 5 and subtracting the user score. The final score is achieved by adding the calculated odd and even numbered results, and multiplying the sum by 2.5.

The SUS score ranges from 0-100, where a higher score represents better usability. However, most systems score within 10 points from 70. [35]

Part III

Prestudy

Chapter 4

Chosen Technology

The development of Pyception relies on several 3rd party technologies. This section describes the necessary technologies, and gives a short reasoning to why that particular technology is chosen.

4.1 Python

Python is an interpreted, high-level, object-oriented programming language with support for dynamic typing. The kernel of Python is very small, but has the ability to import extension modules. This enables a wide range of opportunities, such as creating graphical user interface or web-related utilities[11]. The design philosophy emphasizes on code readability with a clear and clutter free syntax and it is currently taught as the first programming language to students at NTNU through the course TDT4110 [10]. The Python website[17] is a good resource for Python, providing access to interpreter, documentation, and communities.

Python has the ability to execute code from a string, a feature that enables it to run code submitted during runtime. This provides an opportunity to execute and analyze code provided from games, an ideal trait for the server in client-server architecture. Python has web related utilities, but it is not suitable to create the client side of a game running in a web browser.

4.2 Django

Django is an open source high-level Python web framework [15]. A web framework intends to ease the creation of standard features, such as database management and URL parsing of a web site [16]. These features are common for most web pages, and necessary for Pyception as well. Django is based on the principle of "don't repeat yourselves" and applies the Model Template View (MTV) pattern.

Being written in Python, Django enables the project to create a web service, which executes a user's Python code. The Django application will serve as the server side of Pyception, and among other things deal with storage and administration of tasks and solutions.

It is imporant to note that Django applies MTV pattern in addition to the Model View Controller (MVC) pattern. This is important because when referring to the view in this project, it is referred to the MTV view, as described in Section 4.2.1.

Bitnami Djangostack The installation of Pyception use the Bitnami Djangostack as an installer of Django. The Djangostack combines Django with MySQL database, and generally makes an easier installation process.

4.2.1 Model Template View

MTV is a software architechture pattern that focuses on code reusability and separation of concerns, one of Djangos core philosophies. Data storage is modeled in the model, templates handles how the data is represented and the view handles what data is presented. The nameing of the pattern is unfortunate as the view mentioned here must not be confused with the view in MVC.

4.3 MySQL

MySQL is the world's most popular open source database software, and is a relational database management system. It features high speed, great reliability, ease of usage and is the preferred choice for web applications [13]. MySQL supports a wide range of platforms, among them Linux, Windows and Mac OS X, and have an Application Programming Interface (API) for several programming languages.

The Django web framework requires a database to store information of the web service it supplies. Out of the box, PostgreSQL, MySQL, Oracle and SQLite is supported, and other databases are available through 3rd party software [14]. MySQL is chosen as it satisfies the needs of Django in addition to me having prior experience using it.

4.4 JavaScript

JavaScript is a high-level, dynamic, interpreted language that is supported in all modern web browsers. The design philosophy behind strives to maximize flexibility, rather than focusing on structure or encapsulation [12]. In web applications, JavaScript is executed on the client side and often interacts with the document object model of the page.

The extensibility and flexibility of JavaScript makes it well suited to create the game client. However, the most important aspect of choosing JavaScript is that it runs on the client's computer, thus demanding far less computational power from the server.

4.5 JSON

JavaScript Object Notation is a simple text-based standard for a data-interchange format. It is designed to be readable by humans, and easily interpreted by ma-

chines. JSON is derived from the JavaScript language, but is language independent, and supported by many programming languages[18].

Communication between the JavaScript frontend and the Django backend is necessary, and JSON is supported in both JavaScript and Django. An added positive feature is that I have previous experience working with JSON.

4.6 EaselJS

The EaselJS is part of the CreateJS suite[20] and designed to handle working with the HTML 5 canvas. The CreateJs suite consists of modular JavaScript libraries and tools which act together to enable rich interactive content on open web technologies via HTML5. The libraries are designed to be independently functional, and EaselJS provides an easy interface to the HTML5 canvas.

EaselJS will serve as the graphic engine in Pyception, making player and world animate and move according to programmed behavior.

4.7 HTML

The HyperText Markup Language (HTML) is the publishing language of the World Wide Web. It is one of the main components of the Open Web Platform. The first version of HTML was described by Tim Berners-Lee in late 1991. [21] The language is written with HTML elements consisting of tags with parameters such as <h1>for header text and <img>for images.

Pyception needs HTML to create a webpage and a canvas element which the JavaScript game can use.

4.8 CSS

Cascading Style Sheets (CSS) is a simple mechanism for adding style (e.g., fonts, colors, spacing) to Web documents.[22] Primarily CSS is used to separate the content of a webpage from the style.

In Pyception, CSS will style all HTML elements visible to the user.

4.9 Technology Summary

The backend server solution for Pyception is developed on the Django webframework. Django again is developed in Python, but has no good means of storing user supplied data on its own. This is where MySQL is handy. As a database it stores all data necessary for Pyception.

On the frontend, where the game meets the player, two technologies comprise the experience. JavaScript is supported by all major browsers, and has along with the graphic library EaselJS, enabled me to create a game.

Connection between the game server and client is provided by JSON.

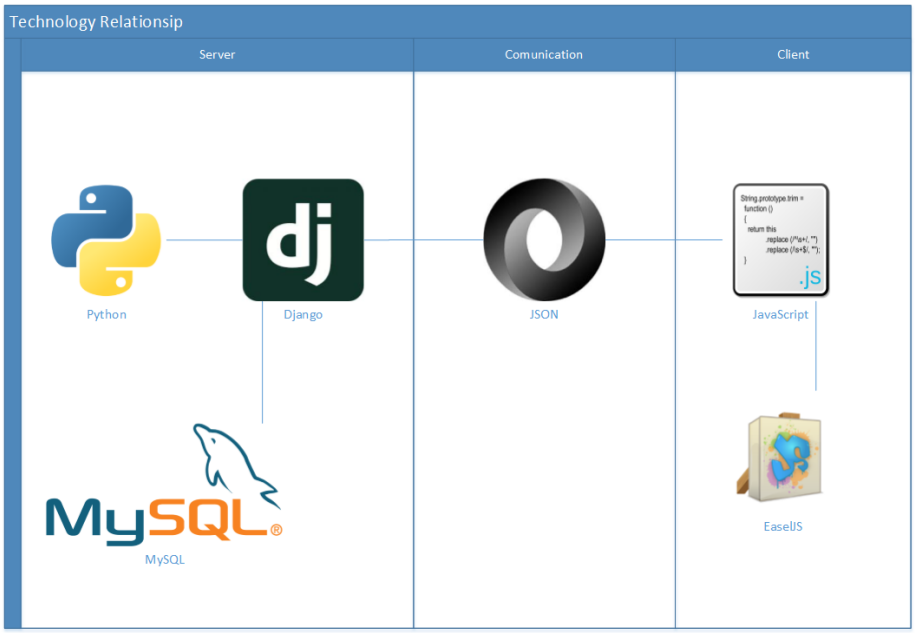

Figure 4.1: Visualisation of the relationship between the technologies.

Chapter 5

Previous Work

5.1 Educational Games

Moreno-Ger et al.[5] describe their vision of future educational games, and their integration with online learning platforms. First they address three approaches to designing educational games; edutainment, repurposing existing games and specifically designed games. The first two are rejected based on lack of entertainment or educational considerations, leaving specifically designed games as the desired approach.

In [2], Malone finds three categories of situations that improves the interest of a student: challenge, fantasy and curiosity. The challenge makes the student reach for a goal and do a little more, or better. A good fantasy evokes feelings in the student, and lets the student care more about the task at hand than at studying some topic. Curiosity is a driver that keeps the student going for more. When discussing the fantasy, Malone identify an important difference between intrinsic and extrinsic fantasies. Extrinsic fantasy is a fantasy the gamer can evoke, but the fantasy has no effect on the task at hand. The intrinsic fantasy makes the player use elements of the fantasy to play the game.

5.2 State of the Art

This section intends to present services that are meant to be educational and fun, and are available online. In order to create a best possible prototype, I have chosen to check into several serious games with various curriculum in the field of science. The marked for educational services online is emerging, and opportunities unlimited. A great challenge is to create a service that are both educational and fun, nevertheless successful services exist and hopefully Pyception will manage to adapt their key features. In order to get a thorough insight in the varied field of online education, I choose to review the following services.

5.2.1 Khan Academy

The Kahn Academy is an nonprofit educational organization with a mission statment of "providing highquality education to anyone, anywhere"[25]. The

organization is built around a webservice which provide some 2500 micro lectures in subjects ranging from arithmetic to physics and history. Microlectures are videos that are freely available from the webpage. The serives also provides some computer science exercises, and more than 300 exercises in the field of mathematics. The mathematics exercises are structured as a tree in what they name a knowledge map, and as the exercises are completed they change colors. Various achievements are available to attain from solving a lot of problems or gaining proficiency in a subject.

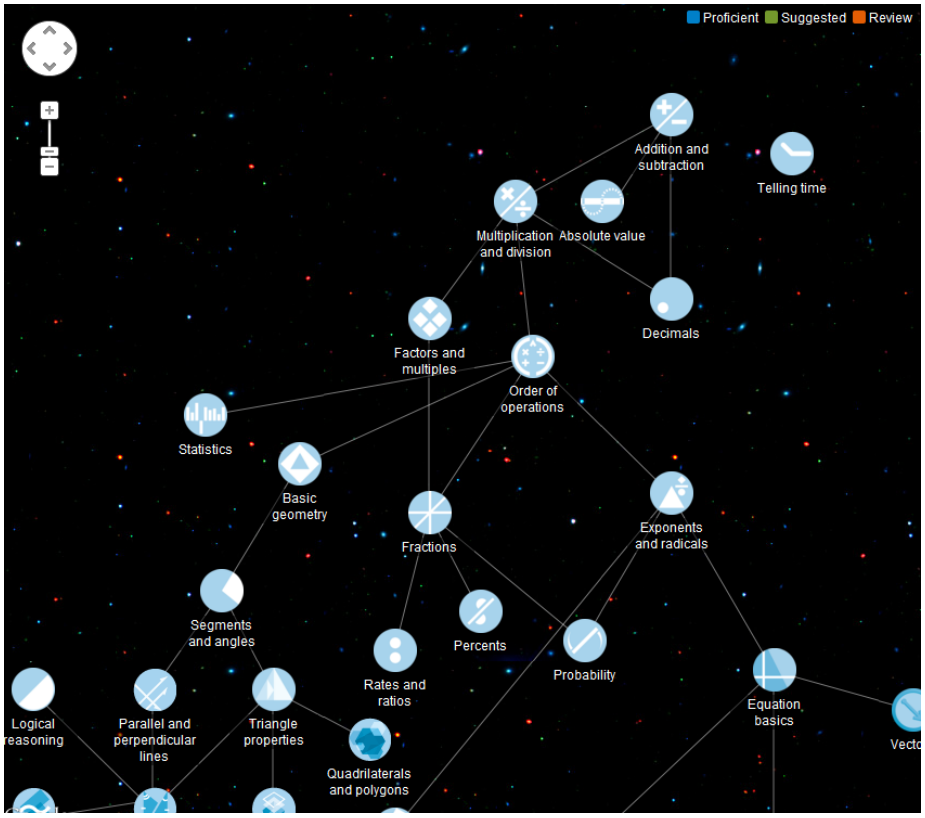

Figure 5.1: Screenshot of the Knowledge map. December 2012

Among Malones' three chategories, Khan Academy mainly focuses on the challenge aspect.

5.2.2 Codecademy

Codecademy is a webservice that is designed to help users learn how to program[26]. The service was launched in 2011, and was originally based on coding in JavaScript. Interactive lessons are offered in the form of courses, and the courses are structured in sections of a varying number of exercises and difficulty. The service is gamified, which means that when a user completes an exercice, section or course, a reward is assigned. In addition to the lectures provided in JavaScript, Codecademy also features lessons in Python and Web fundamentals, among others.

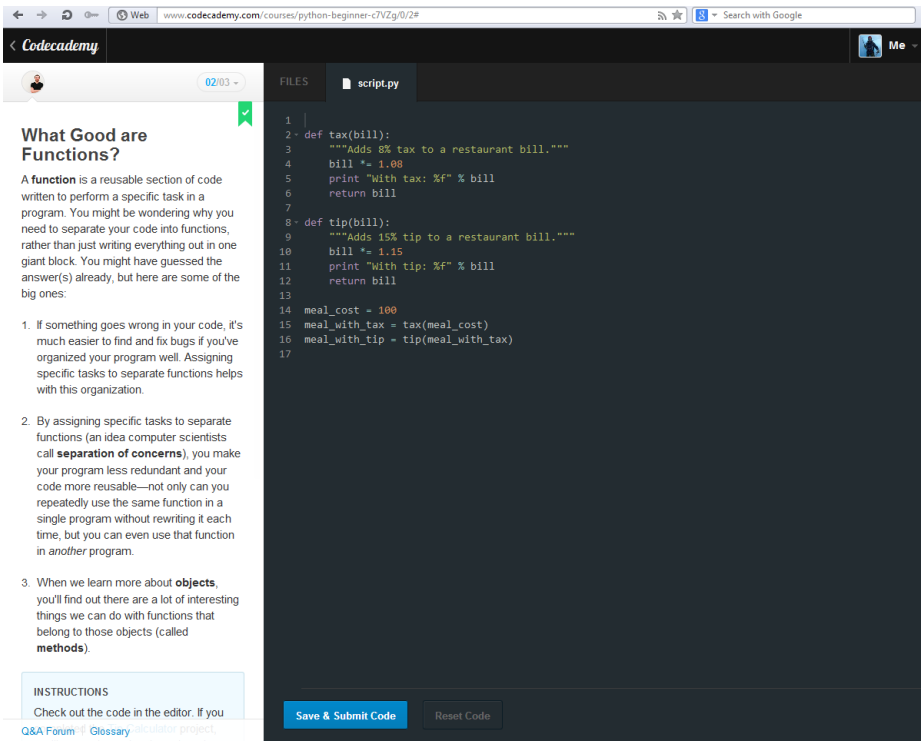

Figure 5.2: Screenshot from Codecademy of the coding interface. The task at hand is described in the column to the left, and the right column is where the user code.

Codecademy mainly focuses on Malones' challenge category.

5.2.3 Code Hero

Code Hero is a 3D first person shooter, where the player is armed with a gun fireing JavaScript code[27]. The player starts out "inside" a computer, and makes his way through the game by shooting JavaScript code at objects to change their behavior. This enables the player to add, remove and change code in existing objects. The learning process of the game is based on hints and copy pasting from earlier sections of the game. As the player works his way through the game, he learns how to code JavaScript with the Unity engine. Thus making Code Hero "the game that teaches you how to make games"

Unlike the two previous services, Code Hero focuses on both fantasy and challenge.

5.2.4 Robocode

Robocode is a programming game where the gamers develop own robotic tanks to fight other robotic tanks[28]. The game was first released back in 2000, and was built for tanks coded in Java. In later years, the game has been adapted to other languages, such as .Net or Scala. The game is played by a gamer

developing his own robot in a built-in-editor, or in an IDE of his own preference. The main objective of the game is coding an Artificial Intelegence (AI) for the robot which purpose is to kill the other robots and outlive them. This requires the player to know how to code before he starts playing. When the player is satisfied with his robot, he can pit it against other people's robots through the robocode game engine.

Robocode is kind of a jack of all trades by appealing to all three og Malone's categories.

5.2.5 hACKME game

hACKME game is a game designed for learning about software security by breaking web pages. The game is a series of challenges grouped in levels of increasing difficulty. Challenges are typically to find a password, or to trick the service into believing that you are authorized. The game does not make the player code, but rather find bugs placed in the code made by the developers. hACKME is a game developed at NTNU, and it has resemblance to other hacking games, like try2hack[34] or notpron[33].

hACKME game tingles the players curiosity, although the main focus lies within the challenge category.

5.2.6 Key Features

The common denominator of all the reviewed services are that their goals is to be educational. However, not all of the services have the same approach on the element of fun. According to Malone it is possible to categorize fun games in three categories, fantasy, challenge and curiosity[8]. While Khan Academy, Codecademy and hACKME mainly focus on the challenge aspect, Code Hero focuses more on both fantasy and challenge. With the freedom given to the player in Robocode, the game stimulates the curiosity, in addition to applying challenge and fantasy.

Common for these services is the challenge aspect. Code hero, which is the most game like of the examined services, also incorporate the fantasy aspect. In order for Pyception to become a fun and educational game, including the aspects of both challenge and fantasy will be an important factor.

5.3 Game genres

This section presents a few appropriate game genres in relation to Pyception.

5.3.1 Real Time Strategy

Real Time Strategy (RTS) is a game genre where the player controls an array of units, with similar or different capabilities. The player controls all his units from a third person view, and as units are selected there is a menu available which displays the units' capabilities. Often most of the units available are fighters, with the addition some units are able to build structures. The structures enables the player to produce new units or upgrade the existing ones. For a player to be able to train units or build buildings, there is usually one or more resources

required. The resources can either be acquired by gathering them from the playing field, or they accumulate by time. A player is able to play against artificial intelligence opponents, or other human players.

5.3.2 Point and Click Adventure

A point and click adventure is a game with an extrinsic fantasy, where the player has to search through the 2D or 3D game world for objects or clues of what to do next. The game is played in first or third person view and the player is the main character, navigating by clicking at objects or other characters. The story of a point and click adventure is driven by finding objects, talking to characters or combining objects in creative ways. This helps the player solve some riddles that have to be sorted out in order to progress through the game, and may unlock new features, worlds or complete the adventure.

5.3.3 Role Playing Games

Role playing game is a genre where the player controls an avatar, which usually have different skills and abilities than the player has in the real world. The game is commonly set in a fantasy world, and the avatars are able to move freely around. Typically the player controls an avatar in third person, but may also play in first person or as a party with control of several avatars. The main objective of a RPG is to solve one or more quests, either by yourself or more commonly as part of a group.

5.4 Syllabus of TDT4110

TDT4110 is a beginner's course in IT, and teaches basic programming in the Python language, both for further studies in informatics and for the use of programming as a tool in other courses. The course covers the following concepts: variables, functions, control structures, data structures (lists, dictionarys), methods with parameter passing and basic algorithms.

Part IV

Own Contributions

Chapter 6

Concept

In the thesis description it is stated that the product should be a prototype of a game where the player learns to code in the language Python. The game shoud be online and easily accessible, meaning the player should not have to install anything local on the computer in order to play the game. This left me with quite the freedom to create a game as I like, so a few sketches were produced along with a short analysis of their fun and education potensial.

6.1 Sketches

This section presents the sketches by genre. Each sketch is first outlined, followed by an analysis of fun and educational potensial.

6.1.1 Real time strategy

Python as a part of a RTS RTS games are heavily reliant on mouse usage for selection, placement and moving about. There are usually some keyboard shortcuts, and production buildings can queue some units for later production, but most actions are completed by a mouse click. By allowing a player to code/script the selection, placement and moving of units and buildings, the need for a mouse can be eliminated. Further the player should be able to code own functions of buildings and units. For instance, the player could code an attack unit or a group to have a patrol function with two points to patrol as parameter. Many of the actions a player does, is repetitive. For instance, production of several copies of the same unit is done by a function applying the "for loop" on one of the building's own base functions. Or have worker units gather resources as part a "while loop".

Would this game be fun and educational? The idea of being able to code the behavior and add new features to existing units and buildings in a game, is compelling. If the command console were given the right feel, a player could get the feeling of hacking through the game. However, the need to code every command, and repeat the commands in every game, would make it highly repetitive. Players unfamiliar with coding would probably have a hard time learning how to play, as it is required to both grasp the game mechanics and

Figure 6.1: Mockup example of coding in a real time strategy game

how to code at the same time. The educational aspect would be limited as the programming is tied to game features. Composing additional curriculum would demand making changes in the game, not simply adding tasks.

6.1.2 Point and click adventure

Python as part of a point and click adventure As the game itself is centered around solving one or more riddles, one can easily use coding tasks as the riddles a player have to solve in order to advance through the game. For instance, use a game based on a brilliant student at NTNU, who has been framed for cheating at an exam. In order to clear her name she has to hack her way through NTNU's IT systems in search of the material used to frame her. In that way the player could move around on a virtual campus, and use code in various terminals to gain access to other parts of the campus or IT-systems and files. Once a player finds and clicks on a computer terminal, the player gets a simple coding interface and access to run code. In order to guide the player, the player could have a smart phone with current tasks and available achievements.

Would the game be fun and educational? According to Malone[8] there are three categories that makes a game interesting; goal, curiosity and fantasy. The suggested point and click adventure feature all three categories. There is a clear goal of the player to wash her hands of the alleged exam cheating. In addition, the smart phone achievements guiding system will help the player keep track through the game. The fantasy of the game is based on solving the

Figure 6.2: Mockup of an adventure where the player explores a library, and has found a computer terminal

primary goal, proving that the player did not cheat on the exam. The fantasy and the graphics of the game are both factors that may please or displease a gamer.

6.2 Sketch Evolution [notes]

This section will outline the process of composing the game's sketch, from idea to the final concept. Both sketches had compelling arguments for choosing one over the other. The RTS has a very tightly coupled intrinsic fantasy, where everything you do in the game has to be coded, and give both immediate and delayed feedback. However, it has a very high initial step in the learning process and the code curriculum would be narrow. Point and click adventure, on the other hand, has the possibility for a diverse curriculum with task difficulty ranging from "hello world!" to complex algorithms. Implementing an intrinsic fantasy in a point and click adventure would be feasible, but not have the same direct feedback as in a RTS.

After brief discussion with my supervisor I decided to go ahead with the second sketch, Point and click adventure, as it provides the best opportunity for a broad curriculum. This sketch also has an intrinsic fantasy, and appears to have the least complex implementation.

After the decision of going with the point and click adventure, I spent more time refining the game concept. Even though this was the best option for my concept, I decided to make some modfications. When it comes to game freedom, point and click adventure is not very flexible. The player selects objects or other characters, and the avatar moves by itself to the specified area, if possible. In the genre RPG, the player controls the avatar freely in the world, but apart from that RPG has a lot of common ground with point and click adventure. As in point and click adventure, the RPG gameplay is still focused around solving riddles. However, point and click might be a little dated, as it was more common in games 15 years ago, so I felt that adapting the RPG movement style would create a better and more current game experience. In order to create the room needed for the player freedom in RPG, I decided to abandon the head on perspective seen in 6.2 in favor of a top down, bird's eye, perspective.

Figure 6.3: Inspiration for a top down RPG, from gamefroot

For a long time I kept developing the system according to the top down view sketch, until my friend Magnus Blütecher Dysthe presented me with a sketch for a game he was thinking about developing. I instantly fell in love with his concept of a game set in space with nodes connected in a grid. In order to adapt my game to Magnus' sketch, only minor changes were required, and the game would be far more appealing. I presented my thesis and asked if it was ok to model the game according to his sketch, see figure 6.4. Magnus agreed to this, and has later contributed to the game by making the game graphics. Most of the changes necessary to adapt the game to the space concept, were simplifying the development by reducing features such as collidable objects and unavailable areas. It also presented an easier way to create an enemy that act logical, and not silly or out of place. This were the final iteration of the sketch, and is the outlines of what the game is today, see fig 6.5.

6.3 The Final Concept

This section gives an outline of what the game is today. When the game starts, the player is presented with a story that gives the player a motive to play and an introduction to what to expect.

6.3.1 Game Story

"This is planet Pythonia. The planet is located in quadrant 2d5, section 12. It's inhabitants are peaceful and does not believe in weapons. The planet is powered by a swarm of satellites orbiting the planet and converting the sun rays to usable electric power. Lately Dr.Thereon has developed a death ray, and

Figure 6.4: The game sketch of Magnus Blütecher Dysthe

uses his spacecraft 'Radix' to travel around the planet and disable all satellites, leaving Pythonia without power. When Dr.Thereon started his corrupt plot of havoc, the Pythonian launched their maintenance crafts, but to no avail. They were shot down while correcting the satellites, every last one. Except one, yours. You are the pilot of the last spacecraft in Pythonian orbit, and the Pythonians have fitted your ship with time warping capabilities. This means that you need not to fear Dr.Thereon when you are docked with a satellite. Your task is to repair all the satellites in the quadrant, and dock with the power dish, to expel Dr.Thereon and proceed to next quadrant. The faith of Pythonia lies in your hands. Good Luck!"

6.3.2 Playing Field Elements

The objects displayed in figure 6.6 make up the interaction on the playing field.

The spaceship This is the avatar representing the player in the game. It is flown using the arrow keys on the keyboard, and can dock with satellites and the power station when the speed is sufficiently low.

Satellites The satellites distributed on the playing field are linked together in a grid of power connections. The coding exercises are placed in the satellites and is accessible for a player if he successfully docks with a satellite. The satellites are displayed in two colors, red and green, and distinguish between respectively solved and unsolved exercises. Upon beeing solved, the satellite changes color from red to green, and the power connections of the satellite are changed to a light blue color.

Power station The power station is the level exit. When all satellites are powered on by solving their exercises, the player is allowed to dock with the

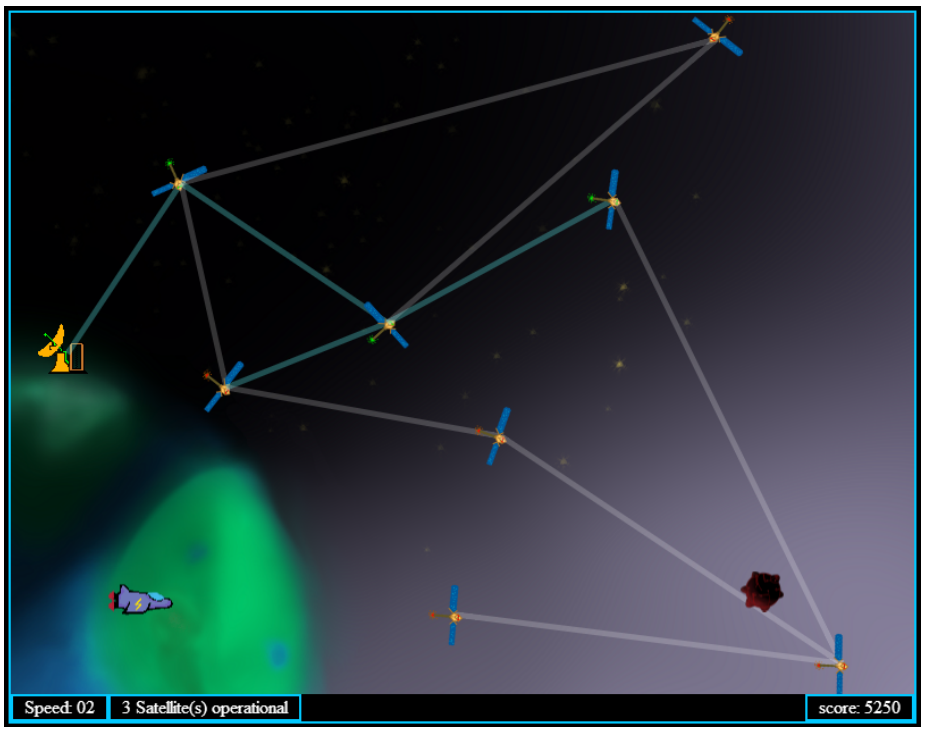

Figure 6.5: Screenshot of the game as it is today. The code exercises are accessible by docking at any of the satellites

Figure 6.6: From left to right: The player spaceship, an unsolved satellite, a solved satellite, the power station and Dr.Thereon

power station and proceed to the next level.

Dr.Thereon Dr.Thereon is represented by his spaceship 'Radix', and has the only weapon in orbit around Pythonia. His mission is to wreak havoc on the people of Pythonia by disabling their power sources. He completes this by three means. First he has disabled all the satellites, second he shoot down other ship within his range and third, as the player fixes the satellites, he flies around and mess with the code in some of the solved exercices, disabling the satellite again.

6.3.3 Code Interface

The code interface is displayed when the player has docked with a satellite. Four text areas are displayed in the code interface, where the largest is reserved for the player to code in. The other text boxes display the task, show the output of the player's code, and present the assessment of the player's code.

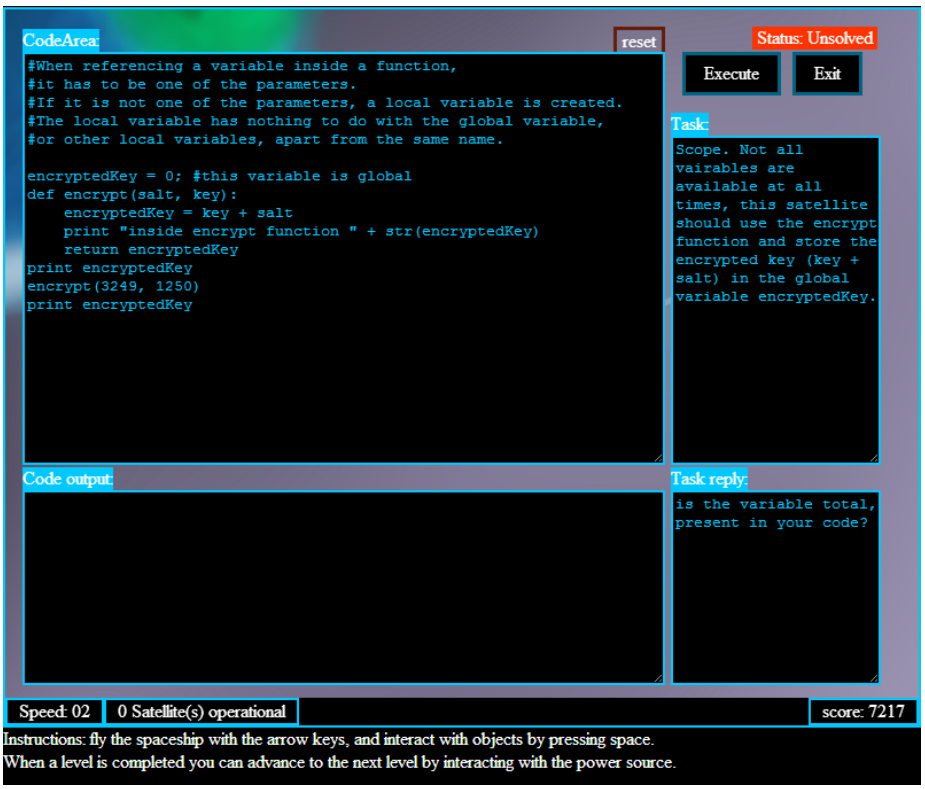

Figure 6.7: A screenshot of the code interface

In addition to the text area, the interface has three buttons available; execute, reset and exit. The execute button is used to send the player's code for interpretation and evaluation, and the result will be presented in the two bottom text areas. If the player is stuck and want to start over, the reset button will remove the code and restore the exercise to its initial state. The exit button will leave the coding interface, and return the player to the playing field.

6.3.4 Game objective

In order to win the game, a player has to complete all the levels. The levels are completed by solving the exercises, and powering on every satellite the particular level and dock with the power station.

6.3.5 Key Features

According to section 5.2.6 it is important that the game adhere to one or more of Malone's fun categories[8]. Covering all three categories is a tremendous challenge for all games, and if not implemented correctly may lower the fun value of the game entirely. Based on this Pyception will mainly focus on two of the aspects, challenge and fantasy.

The sketch detailed in the previous section, would mainly appeal to the fantasy and challenge categories defined by Malone [8]. Through the game story

and the graphics displaying a space environment, the fantasy aspect is catered. In addition, the exercises will provide the player with a set of challenges.

6.4 Syllabus

A target audience for Pyception are students attempting to learn programming, such as NTNU students attending the course TDT4110. It is natural for the game to follow the syllabus of the mentioned courses, as stated in section 5.4, and if the Pyception is successful, it may be deployed as supplementary educational tool or even as a part of the compulsory exercises.

Chapter 7

Requirements

Requirements are important in order to provide a clear set of features that must be present in the project, and also make sure all stakeholders needs are satisfied. Requirements are divided in two groups, functional and non-functional requirements.

7.1 Functional Requirements

Functional requirements are the baselane for the game set according to the thesis description, and will later be the basis of the evaluation. FR1 group are requirements regarding the game in general. The requirements related to actions that take place on the playing field, are listed in FR2. In FR3 the requirements are tied to the players interaction with the coding interface, as well as the response generated by the interface. Finally the FR4 lists requirements of features that a lecturer would require in order to adapt the game syllabus to his or her course.

FR 1 Game

Id	Description	Priority
FR 1.1	It should be possible to start playing without an installation process	High
FR 1.2	The game should have an intrinsic fantasy, where the player's action affects the playing field, and the playing field affects the play.	High
FR 1.3	The game should proceed to the next level when the player interacts with the power source after all the tasks are correctly solved.	High
FR 1.4	The game should store tasks in a database.	High
FR 1.5	The game should retrieve tasks from the database.	High

Table 7.1: Functional requirements Game

FR 2 Playing field

Id	Description	Priority
FR 2.1	As a player I want to be able to navigate the space ship freely on the playing field	High
FR 2.2	As a player I should not be able to navigate the space ship outside the playing field	Middle
FR 2.3	As a player I want to be able to dock with all satellites	High
FR 2.4	As a player I want to not be able to dock with the powerstation when some tasks are unsolved	Middle
FR 2.5	As a player I want to be able to dock with the powerstation when all tasks are solved	High
FR 2.6	There will be an enemy on the playing field at all times.	Middle
FR 2.7	If the player is in range of the enemy, a penalty will be applied.	Middle
FR 2.8	The enemy will sporadically ruin the satellites and mess up the code	Low
FR 2.9	As a player I want to be able to earn points and keep track of the score	Middle
FR 2.10	As a player I want to be able to see a measure of completion of the current level	Low

Table 7.2: Functional requirements Playing field

FR 3 Code Interface

Id	Description	Priority
FR 3.1	As a player I want to enter my own code	High
FR 3.2	As a player I want to have my code evaluated	High
FR 3.3	As a player I want feedback regarding what is wrong in my code	High
FR 3.4	As a player I want feedback when my code is correct	High
FR 3.5	As a player I want to be able to reset my code	Middle
FR 3.6	As a player I want to have my code stored so I can go back and continue later	Middle
FR 3.7	As a player I want to have my finished tasks stored so I can go back and review it later	Middle
FR 3.8	As a player I want to be able to debug code	Middle

Table 7.3: Functional requirements Player Coding

FR 4 Lecturer

Id	Description	Priority
FR 4.1	As a lecturer I want to have a password protected administration account with access to the exercises.	Middle
FR 4.2	As a lecturer I want to be able to create new tasks	High
FR 4.3	As a lecturer I want to be able to edit tasks	Middle
FR 4.4	As a lecturer I want to be able to delete tasks	Middle

Table 7.4: Functional requirements Lecturer

7.2 Non-functional Requirements

Non-functional requirements describe necessities not directly related to features of the game, like security and extendability. Definition of the non-functional requirements are listed alphabetically in this section.

Accessibility A high priority of Pyception is the accessibility. In the thesis description it is listed that it should be easily accessible online. From a users perspective, it should not be necessary to perform any installation process prior to playing the game. Installation guide should be available to lecturers who want to establish a gameserver.

Compatibility Pyception is a proof of concept, making a wide compatibiliy unneccessary in order to prove the value of the project. Compatibility is a medium priority. The game should be playable in the newest version of all major browsers. The game server should be compatible with both Linux and Windows platforms, but as noted in appendix B, the installation guide will only be available for the Windows platform.

Performance Pyception is a proof of concept, making performance a low priority issue. The game should be playable, but not scalable to usage for many players.

Security As this is a proof of concept, the security aspect is low on the priority list. The topic should recieve some attention in the report, listing potential security issues that are present in Pyception.

Usability The usability of the system has a high priority in the project. The game should be easy to use and understand, but hard to master. Gamers should be able to understand the game mechanics within few minutes, and be able to understand how to submit code for execution and evaluation. Solving tasks may require the gamer to learn something new, so the gamers are expected to spend time solving the tasks. There should be an understandable interface for lecturers to create and edit tasks.

Chapter 8

Design Choices

The basis of Pyception has always been to create an educational game, where the player will learn Python by playing a game in his web browser. The loose boundaries has given me the freedom to create the game as it suited me, but in the process I had to make a lot of choices. In this section I will discuss notable choices made regarding Pyception.

8.1 Architecture

An entire chapter is devoted to describing the chosen architecture of Pyception, however the implemented architecture was not the single option that was considered. The core challenge of Pyception is to create a web service that execute and evaluate user produced code. Essentially two options are available, execute code locally on the client or on the web service.

8.1.1 Server Side Execution

Prior to the project start, to my knowledge there existed web frameworks, such as Django. This would allow me to create a Python web app, and by using the Python statement 'exec' I could execute Python code from a string. Getting a string containing user code from a web browser to a web server I considered trivial, and similarly returning the result of evaluated code. Another advantage is that there is no need for locally installed software apart from a modern web browser. This will enable a user to access the game and start playing, without going through the hassle of an installation process. The server execution approach would necessarily create a somewhat larger work load on the server, due to the webserver not only serving the game, but also executing the player code. Executing code on the server would present a major risk to security, as stated by [30] "run native machine code on the client machine—an ultimate hacker goal and the definition of disaster". However, if the executed code is properly sandboxed on the server, the disaster may be avoided.

8.1.2 Client Side Execution

Locally executed code would have the advantage of performance by creating less load on the server and distributing the processing requirements to the clients

machines. Security of the system would benefit from client side execution by not creating a vulnerable server.

The drawback of client executed code is that it complicate the solution. Either the player would have to install Python locally or a Python interpreter in JavaScript would have to be written.

8.1.3 Summary

Having the user install the Python interpreter would not create an effortless introduction to programming and thus be in violation with RQ3, which states that the start should be effortless. In fact, the installation process is part of what the project seeks to remove from the learning process, in order to let the student focus on coding and not the facilities required to produce code. Pyception is a project limited in time and the complexity and uncertainty of implementing a Python interpreter in JavaScript were estimated a risky venture.

When comparing the security risk of server side execution with the complexity and uncertainty of client side execution, I decided in favor of server side execution. Pyception is a proof of concept developed in a short time frame, requiring some sacrifices. If the prototype proves to be success, security can be prioritized in the future.

8.2 Game Engine

In order to find a suitable game engine for Pyception, I searched the web for recommended game engines. A post on the Stack overflow forum[29], were decisive in my inquiry for a suitable game engine. The post recommended Crafty and EaselJS, and with a later update, gives extra support in favor of EaselJS. In addition to the game engines recommended by stack overflow, I looked into the Unity game engine before I chose game engine.

The criteria for decision will be based on the game engines ease of usage, documentation and available community support.

8.2.1 Crafty

The Crafty engine is a JavaScript library designed to work with the HTML5 canvas. Their website provides several basic demo games that are playable, and giving indication on features of a game created with Crafty. The demos shows input from mouse, keyboard and playing of animations and sound. The site features a thorough documentation of Crafty's features, a thin getting started section, and also an active forum. Crafty boast of being cross browser compatible, however basic features like mouse and keyboard input, failed to function in the Opera web browser on some of the demos and tutorials.

8.2.2 EaselJS

The EaselJS is part of the CreateJS suite and designed to handle working with the HTML5 canvas. The rest of the suite handles audio, animation and preloading. Similar to the Crafty website, the EaselJS website has several demos showing the engine in use in several different applications, thorough documentation, an active discussion forum and a getting started section.

8.2.3 Unity

The Unity game engine is designed to create 3D games as web plugins. It supports several programming languages and has its own integrated development environment. A prerequisite of playing a game with the unity engine, is downloading and installing their browser plugin. The Unity website features, loads of impressive demos, extensive documentation and a large community.

8.2.4 Summary

The Unity game engine is far superior to the two HTML5 game engines when it comes to documentation and community support. However, it is also far more complex when it comes to development, and requires the user to install a browser plugin. Of the two HTML5 game engines, Crafty failed to read keyboard input from some web browsers during testing. This inconvenience left me with the EaselJS as the prefered game engine.

Chapter 9

Architecture

This chapter is devoted to create an understanding of the architectural structures of Pyception. Usually there are several stakeholders in a project and they will have different opinions on what is relevant to them. In order to present the system to all stakeholders, many projects have crammed one architectural view with as much information as possible and ultimately failed to present a clear view to any one of the stakeholders. To avoid such missfortune, the documentation of Pyception follows the method presented in Kruchtens "4+1 View model"[6]. The model presents the architecture in five concurrent views, with

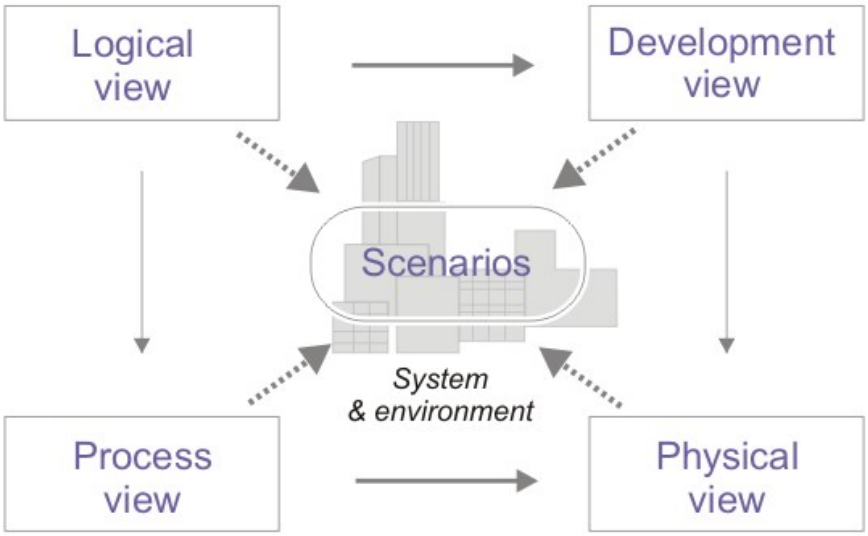

Figure 9.1: Illustration of the 4+1 view model.

the purpose of creating a clear understanding of the system as a whole for all stakeholders. The five views are; logical view, development view, process view, physical view and scenarios, and are related as seen in figure9.1

9.1 Physical View

The physical view describes the mapping of software onto physical objects such as server and clients. The view is intended for administrators, providing an overview of the system and its communication.

Pyception is deployed on one server, to which players and administrators can connect via HTTP connections. The server is responsible for serving the game, storing code exercises, executing player code and evaluating the result of executed code. In addition it provides an admin interface for lecturers to add, edit and delete tasks. The game package is loaded directly into the players browser from the server, and most of the game execute there. When the server is running, all that is required of a player, is to enter the correct web page.

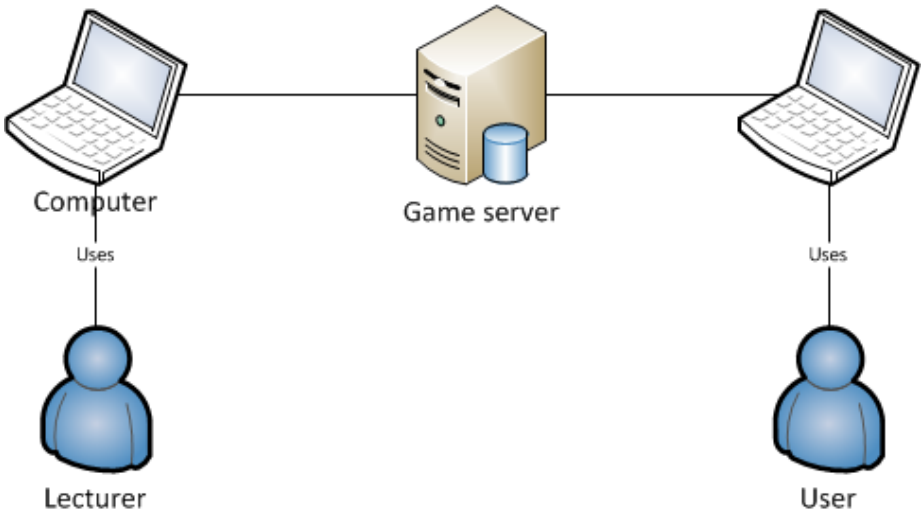

Figure 9.2: All of the software in Pyception is deployed on the game server, however it provides web access to players and lecturers

9.2 Logical view

The logical view is primarily concerned with the functional requirements of the system, detailing how the system is coded. By the means of class diagrams, the logical view convey how the classes and objects are related and utilized. We will start by examining the game server and later explore the more extensive game client.

9.2.1 Game Server

The Django game server is a Python web framework based on the MTV pattern. The game server is a small service based on only one model which is the mission tasks. The model is backed by the four views detail(), evaluate(), evaluateJSON() and task_JSON(). The first two are linked with the templates detail.html and evaluate.html. detail.html presents the missions stored in the

model, and supplies a way to test tasks without doing it in the game. The evaluate.html presents the results of a task submittet by detail.html.

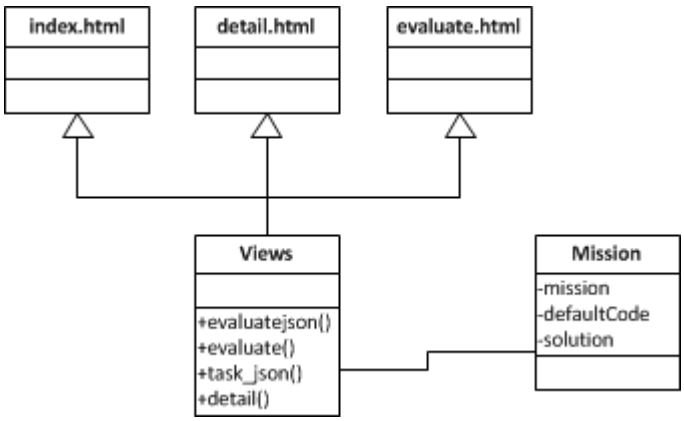

Figure 9.3: Class diagram alt1.

The evaluateJSON() and task_JSON() views are used by the game client where the latter provides details regarding the task the player is about to solve. The former execute the player code, evaluate according to the mission, and presents the result to the game.

The Django framework provides an administrative interface to the model, thus enabling lecturers to add, edit and remove missions tasks. Access to the administrative interface is only granted to users with the correct privileges, in most cases limited to the lecturer.

9.2.2 Game client

The game client can roughly be divided in two parts, playing field and coding interface. The two parts are connected, but do not run simultaneously and correspond to two different functional requirement groups, FR2 Playing Field and FR3 Code Interface. The playing field is based on a game loop that runs continuously until a player docks with a satellite. The game loop is then paused and the coding interface is displayed.

Playing Field

The playing field is developed with a game loop based on the pipe and filter pattern, where a stack of elements is processed one element at a time. When the entire stack is processed it starts over again. In Pyception the game elements are queried for actions, updated and painted on every tick. The game is limited to a maximum of 30 ticks per second. However, if the game is played in an environment with limited resources, the amount of ticks per second will be lower. The four stages of the game loop are player, level, enemy and Graphical User Interface (GUI).

The player class reads the input stored by event handlers, and uses the input to move the player freely on the playing field. There is code implemented in the player class to prohibit the player from leaving the playing field, and if the

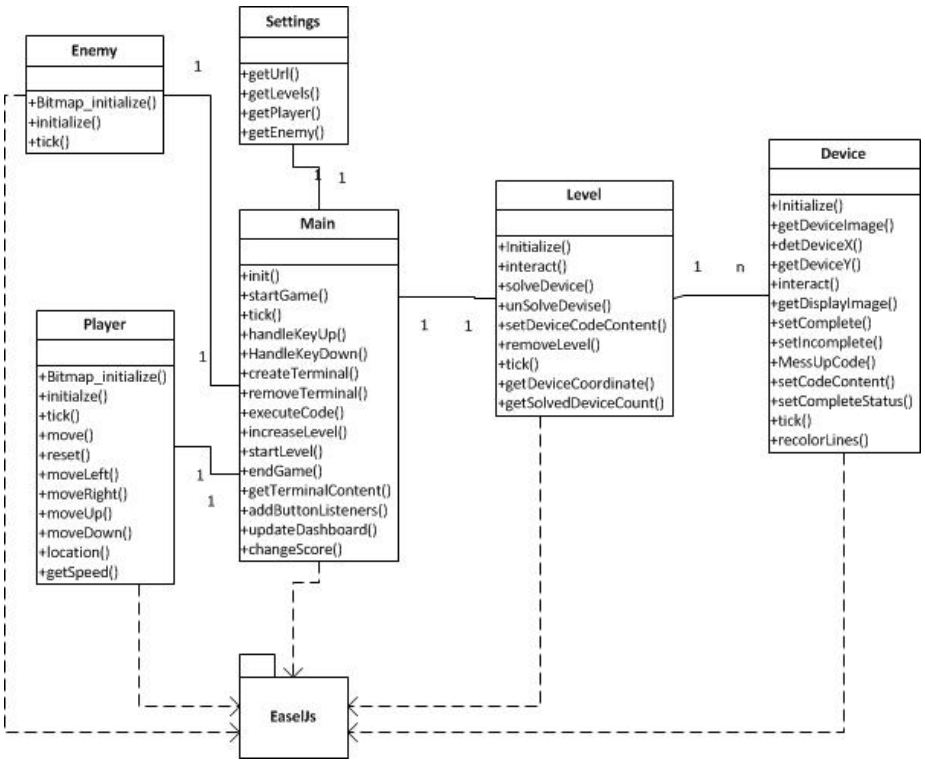

Figure 9.4: Class diagram game client.

player press the interact button, the player class calls the interact function of the level class to check if interactions are possible.

The level class handles player interaction with satellites and the power station. It checks for conditions like speed and number for solved satellites, before it allows a docking or rejects docking and display a warning message.

The enemy is programmed to fly from satellite to satellite, thus keeping the enemy on the playing field. If the player comes within range of the enemy, the player is fired upon, resetting all satellites of the current level and subtracting points from the players score. Occasionally, when the enemy arrives at a solved satellite, the satellite is disabled by randomly changing some letters of the code that the player has used to solve the satellite.

After the three game logic stages have been processed, the GUI is updated. The game elements are redrawn, and the amount of solved satellites and score is updated.

Coding Interface

The player is granted access to the coding interface if a successful docking has occured. The coding interface has four text boxes and three buttons, where the largest textbox is for write the code. The other three boxes are used to present the player with the task that must be solved, output from the player's code and output from the code evaluation. The three buttons of the interface are execute, reset and exit. The first button sends the player's code to the

server, execute and evaluate it, and display the result in the code output box and mission output. The reset button fetches the task from the server, and enables the player to retry solving the task as the lecturer created it. The exit button leaves the coding interface and return the player to the playing field. If the player later returns to the satellite, all code is stored enabling the player to continue coding where he left off or use the solution as a reference.

The feedback provided to the player, is reliant on how the tasks are created. It is important that the lecturer creating the tasks have in mind that some output may be lost, if not specifically output in the solution code.

9.3 Development View

The development view illustrates organization of the actual software modules in the project [6].

Pyception can be divided in two packages, and the packages are both hosted on the server. However, the game client is downloaded to a player's browser, and for the most part executed locally on the player's computer. The game client uses the server to perform two activities, fetch task metadata and execute task.

The server package is responsible for storing code exercises, executing player code and evaluating the result of executed code. In addition it provides an admin interface for lecturers to add, edit and delete tasks. When the server is running, all that is required of a player is to enter the correct web page. As the player is loading the webpage, the game is included.

Connecting the server and client are ordinary HTTP requests for loading the game to the player's web browser, and JSON for loading and executing exercises while playing the game.

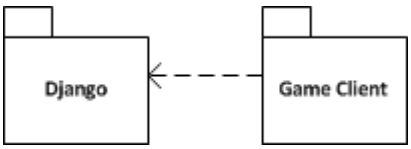

Figure 9.5: Pyception's two packages and their relationship

9.4 Process view

The process view is designed to provide an understanding of how the processes relate to eachother with regards to concurrency and distribution. The view is particularly useful for developers, seeking to mend bugs or expand the project.

9.4.1 Server

The game server is developed on the Django framework. A feature of the Django framework is that it handles all the processes necessary for a webserver, making that a non issue in this project. The only processes in the gameserver that need attention, are those coded in the view. The evaluateJSON function, beeing the

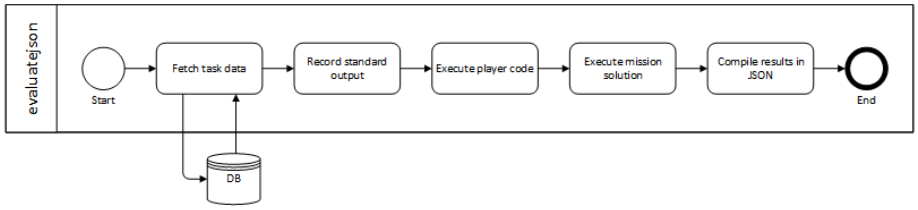

Figure 9.6: insert caption

longest and most complex of the four functions of the view, will now be detailed. The function is called when a player hits the 'execute' button in the game client, sending a post request with player code to an address with specified task id that is unique for each task. The function proceeds by fetching the mission object containing the solution code from the database. It carries on by starting to keep record of the standard output, and then executing the playersubmitted code. The output record is made available for usage in the solution and then the solution code is executed. After execution and evaluation, the results are compiled in a JSON object, and returned to the game client. There are no write operations or exclusive reads, so there should not be any concurrency issues if several users hit the execute button simultaneously.

9.4.2 Game client

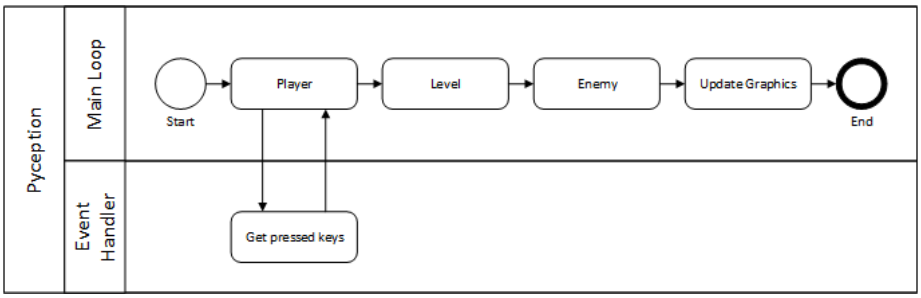

Figure 9.7: insert caption

The game client features more complex code with an actual concurrency issue.The pipe and filter main loop, which handles, among other things, movement of player, is reliant on keyboard input. JavaScript has no feature allowing to query if keys are pressed, so key handling needs to be controlled outside of the main loop with an event handler. The key handling is composed by 2 listeners, one for key down and one for key up. When a key is pressed, the key id is stored in a list, and removed when released.

The main loop shown in fig 9.7 shows the pipe and filter pattern applied in Pyception. The first processed action in the game, is that of the player. It reads the list of stored keypress, and moves the player possision accordingly. If the interact key is pressed, it attempts to interact with the environment. The interaction can have five outcomes. Nothing happens if the player is not close to any other game objects. If the player is in the proximity of the power station

or a satellite, a warning message is displayed if the speed is too great. If the player is within the speed limit of 0-3, and in proximity of a satellite, the playing environment is paused, and the coding terminal is brought up. If the player is in range of the powerstation and within speed limits, but has not repaired all satellites, a warning is displayed. The final outcome occurs if the player is in range of the powerstation, within the speed limit and has solved all satellites. This completes the level and either advances the player to the next level, or completes the game.

After the player action is complete, the level action starts. The level process all devices(satellites) and rotate them according to their random rotation and handle the display of warnings, if necessary. Next process is the enemy, which moves its possision closer to its target, or if it is at its target, chooses a new target. The enemy also check if the player is within its range, and if so, fires upon the player. The firing action causes the level to reset, and deduct points from the score. After the three previous actions has been completed, the final thing that happens in the main loop, is that the graphics are updated to correspond with the objects new possitions.

9.5 Scenarios

The scenario view is an addition to the four views and serve as an illustration of the architecture from a third party view. The view is necessary to provide a sense of wholeness to the architectural views, and is useful when creating test cases.

Figure 9.8 illustrates the game from a player's view. Pyception will always start by showing the story screen to the player, and giving the player the option to start the game. When the game is started, the player is in control of a spaceship and can maneuver freely around the playing field. From the playing field there are three possible events that may occur, come within range of Dr. Thereon, dock with a satellite or dock with the powerstation.

Dr. Thereon Dr.Thereon is the enemy who in the story has disabled all the satellites providing power to the planet Pythonia. The enemy is flying around the playing field between satellites, and if the player flies within range, Dr.Thereon fires his laser and kills the player. The player looses a large portion of his points, and all satellites on the level are reset to not fixed.

Satellite The satellites contains the coding exercices. If the player is within range of a satellite and flying with a speed of 3 or lower, he is able to dock with a satellite. When the player docks with a satellite, the code interface is brought up and the time warp is powered up and freezes Dr.Thereon in order to let the player code in his own pace. The code interface has three options; execute, reset and exit. The execute button sends code to the server for evaluation, and if correct, the satellite is marked as solved. The reset button fetches the initial code from the server, and is useful in cases where the player has deleted too much code, or just want to start over. The exit button terminate the code interface and leave the satellite in the state it is, solved or not. The player is returned to the playing field, and the time starts running again.

Power station The power station is used to complete a level. Similar to the satellites, the player has to be in range and fly slow to dock with the power station. An additional requirement is added in order to successfully dock with the power station. The player has to solve all satellites on the playing field, or a warning message is displayed if the player attempts to dock with the power station, and the player continues to play on the playing field. If a player has solved all satellites on the level and attempts to dock with the powerstation, the spaceship is moved to the next level. When the player solves the final level, a game completed screen is showed and the player has won. If the player wishes, it is possible to start the game over again while keeping the current score, thus enabling the player to earn more points.

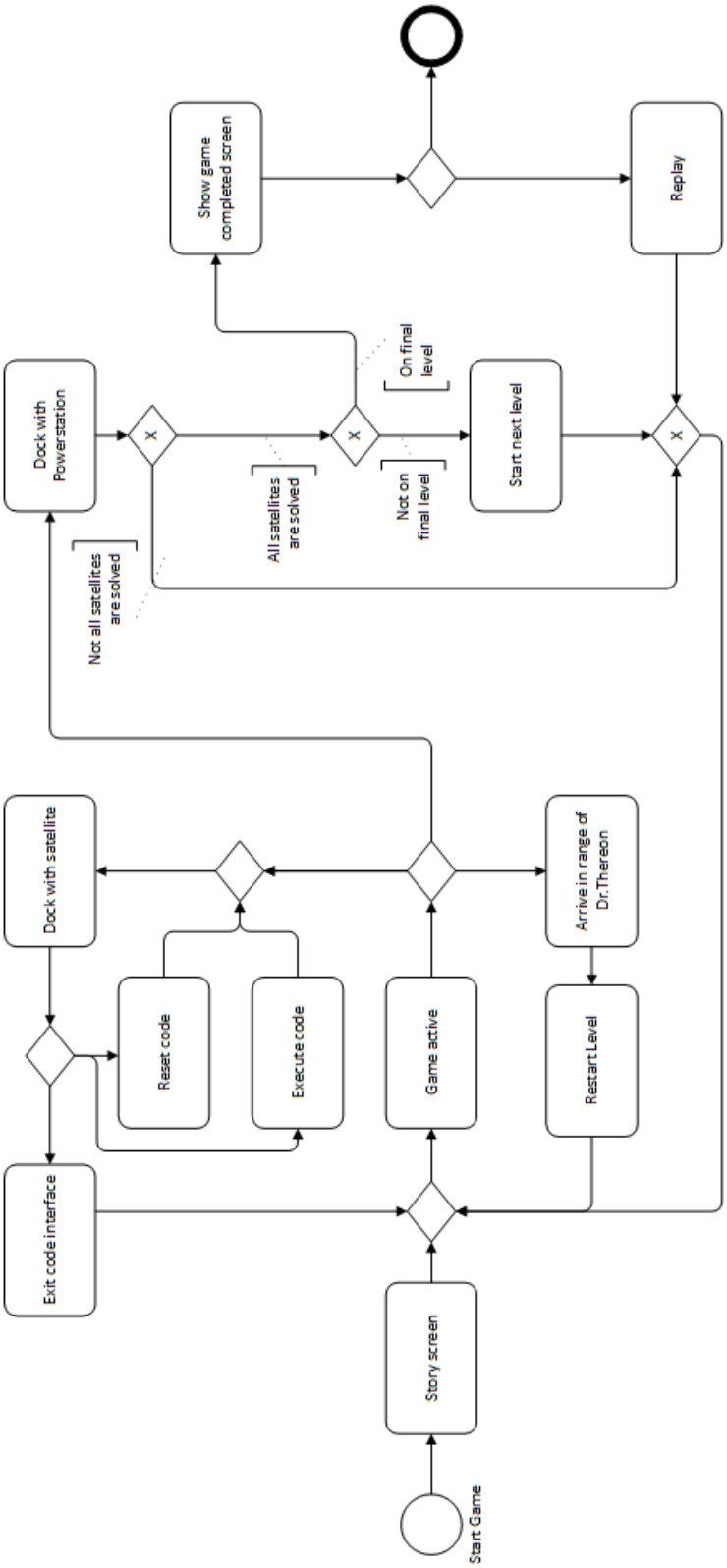

Figure 9.8: Third person view of the gameplay.

Chapter 10

Implementation

10.1 Milestones

Development of Pyception has been divided in milestones, where each milestone has a key feature that needs to be implemented. By implementing the milestones in a chronological order, they will give an indication of the progress in the development. The milestones will cover the core features of the game. However, other aspects such as graphics and development of curriculum, is not accounted for and will be developed independent of the milestones.

M1 Django "Hello world!"
Use the Django web framework to create a web page, and familiarize with the features available in the framework.

M2 Execute submitted code
Test interesting game engines and determine which to implement in the game client. Use the chosen game engine to create a demo capable of handling keyboard input to move a player shape around the screen.

M3 Web client "Hello world!"
Test interesting game engines and determine which to implement in the game client. Use the chosen game engine to create a demo capable of handling keyboard input to move a player shape around the screen.

M4 Interaction
Expand the demo to include an object that the player shape can interact with by pressing a key.

M5 Submit code
Create a code interface where the player is able to type code and submit the code for evaluation. Tie the code interface to the object created in M4.

M6 Client - Server - Client
Modify the server to execute submitted code and reply using JSON to create a good communication link between the server and the client.

M7 Task status
Utilize the JSON reply from the server, and display results to the user. Store solved exercises as completed

M8 Level
Create a level with several tasks.

M9 More Levels
Create more levels and a way to complete a level and advance to the next level.

M10 Enemy
Add an enemy to the game and create structured behavior for him.

M11 User test the game
Apply the game to a set of subjects. Record their responses.

10.2 Graphics

Malone describes three categories that are important in fun games; fantasy, challenge and curiosity [8]. In Pyception, graphics plays an important role in the creation of a fun fantasy, along with the story of the game. Three options were considered for the process of obtaining graphics, creating it myselves, finding it on the internet or have someone create it for me. The first option were discarded on the grounds that I do not possess adequate skills in graphic design. Several sites on the internet, such as Open game art[32], offer a myriad of graphics with various licences. Finding suitable and good graphics among the multitude of available graphics may be a considerable task, still it is a better alternative than the first option. The final option of obtaining graphics, relies on favors from my friends. As the budget of Pyception is virtually non existent and hiring a graphic designer is expensive, it is not an option to pay a professional. Fortunately, Magnus Blütecher Dysthe presented me win a game idea and was willing to create the graphics for it. I am very happy with the graphics he created and Magnus' graphics are the core contributor to the fantasy of Pyception .

10.3 Exercises

The creation of exercises for Pyception was based on the syllabus of TDT4110 as discussed in section 6.4.

Paras and Bizzocchi note that the flow is a considerable aspect in education. "Flow explains a phenomenon that many people find themselves experiencing when they reach a state where there becomes a perfect balance between challenge and frustration, and where the end goal becomes so clear that hindrances fall out of view." [4]

Pyception must strive to create a syllabus that is easy enough to immerse the player in a flow while still challenging enough to educate the player. It is also important that the exercises follow the TDT4110 curriculum, as outlined

Figure 10.1: Screenshot of the coding interface showing an easy task from level 1

in Section 5.4, if it is to be utilized in the course. The exercises should reflect the fantasy outlined in the concept, in order to achieve a wholeness of the game.

Listed below is the syllabus of the respective levels. The levels are designed to try and create a good educational flow for the player.

Level 1 has a total of 8 satellites with individual exercises. In order to create good flow the difficulty range from the most basic, like printing “hello world” to the standard output. The difficulty increases with exercises teaching how to use variables and how to apply basic arithmetics in Python.

The second level introduces the player to functions, which are widely used throughout the rest of the game. The exercises start off by teaching how to write functions, by using variables and arithmetics learned on the previous level. The syllabus of level two cover functions and more arithmetics, such as the modulo operation.

When arriving at level 3, the player is expected to be familiar with functions and basic arithmetics. The third level utilizes functions in all tasks to convey scope, and introduce control structures to the player. The control structures applied in exercises at this level are limited to if and else. The final level consists of 4 exercises. Control structures such as for loops and if statements with ‘elif’ statements, are utilized in these exercises. Challenging the player with exercises of the most difficult type, the final level involves skills attained on all the previous levels.

Figure 10.2: Screenshot of the coding interface showing a difficult task from level 4

10.4 Security

The non-functional requirement security is listed with a low priority due to the fact that Pyception is a prototype. This is reflected in the implementation by executing player code on the server. "Run native machine code on the client machine, an ultimate hacker goal and the definition of disaster" [30]. The quote states the fact that letting other people execute code on your server, is a bad idea. However the nature of this project is to prove the possibilities that lies within online education, so some sacrifices must be acceptable in order to produce a prototype within the allocated time frame. The reasoning behind this decision is discussed in section 8.1.1 and will also be included in suggestions for future work in Chapter 17.

10.5 Challenges

In every project, at some time, there is bound to arise challenges that affect the participants. How these challenges are handled varies from project to project, but common to all are that they must be solved. When it comes to software development, common challenges can be new technologies, compatibility with legacy systems or unstructured existing code.

Pyception is a project that starts from scratch, minimizing the risk of challenges concerned with existing systems. However, the implementation has not

been without challenges as I am a young and inexperienced developer.

One of the main challenges of Pyception was the technologies necessary in the implementation. Among other, the main technologies JavaScript and Django were unknown to me at the start of the project. This required me to familiarize myself with the technologies before I was able to use them properly, and this consumed significant amount of the time available to the project.

Initially the project planned to use Unity as a game engine, but due to the complexity of the framework, I decided to examine other options. An additional feature of Unity that is in disfavour to this project, is the fact that Unity requires the installation of a web browser plugin and would be in conflict with the effortless start stated in RQ3 . Eventually I overcame this challenge by using the EaselJS framework in place of Unity.

Graphics posed another challenge to me, as I do not possess skills in that field. This was resolved by having a friend create the graphics, and is discussed in section 10.2

Creating relevant and understandable tasks for a player that has not programmed before posed a challenge to me as I have five years experience with programming. To overcome this challenge I researched the curriculum of the course TDT4110, as well as TDT4100 object oriented programming, and other introductory programming courses at NTNU.

Chapter 11

User Experiment

This chapter will present the most important data generating activity for the evaluation of the project, the user experiment.

11.1 Test Framework

According to ISO 9241-11, usability is defined as follows: "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfacton in a specified context of use."[23] where "contex of use" is defined as: "User, tasks, equipment(hardware, software and materials), and the physical and social environment in which a product is used"

This means that usability depends on the context of use, and to measure the usability we need to focus on the right users with the right goals in the right environment. In order to start the usability testing for Pyception these parts need to be defined

11.1.1 Users

Pyception is a serious game for people who want to learn programming in the language Python. In order to get a most accurate user testing, it is important to find test persons who are motivated to learn coding, or motivated to learn more. Testing on experienced user may also supply valuable test results, in the form of suggestions or eradication of bugs in the game or exercises.

11.1.2 Goal

The goal of this experiment was to get an overall picture of how Pyception worked in a real life setting. In the following paragraphs, thesis RQ2 and RQ3 will be outlined and analyzed based on data from the user experiment, while RQ1 and RQ4 are of a technical nature and will be evaluated on other datasets

RQ2 Is it possible to create a serious game based on programming that is both fun and educational?

From this question we need to figure out if players enjoy playing the game, and if they learn something from it. Pyception is a custom made serious game,

and it would be interesting to know if the game feels different from just solving tasks.

RQ3 Will more people be motivated to learn programming if the learning process has an effortless start through a web based game?

By providing a different approach to teaching programming, and lowering the initial effort required, it is possible to reach a new audience.

Related RQ	Goal id	Goal
RQ2	G1	Is the game fun ?
RQ2	G2	Is the game educational ?
RQ2	G3	Do pyception feel more like a game than programming tasks?
RQ3	G4	Is it easy to get started with the game?
RQ3	G5	Is the game motivating?

Table 11.1: The research questions and their derived goals.

The releationship between goal and questions asked by the questionnaire, is listed in Table 11.2.

11.1.3 Environment

The setting in which the test is conducted, may affect the users, their motivation and how they perform. It is imperative that they feel comfortable during the test, as a stressed out person would have a hard time having fun and will not perform at their best. The target environment for Pyception is an educational setting. Due to the educational nature of Pyception, it is important that I as a test leader meet the users at their local education environment. In this test case it is natural to utilize NTNU as a basis for a good test environment.

11.2 Execution

When a suitable user group and environment has been acquired, it is time to conduct the testing. In order to conduct the tests as smoothly as possible, a structured plan has been deviced.

The testing will be carried out by the users on their computer. Pyception will be deployed locally on the computers used in the test, in order to avoid interference between testers or errors caused by network connection. The deployment will be taken care of by me, so that when the test starts, all the users have to do in order to start the game, is to enter the specific web address. The test will be conducted with 1-3 persons at a time, so there will be no lack of help if something goes wrong.

At the start of the test, the users will be informed of the address of the game, and that they are free to use any aids they would like to use such as Google or ask me a question. I will be present during the testing to help the test users if they get stuck, uncover a bug and observe for anything else that might arise.

The test session will end either by the user completing the game, or spending more than an hour. Following the test, the users will be asked to fill out a

Goal id	Questionnaire question
G1	Q15, The game felt like more than just task solving Q16, The game was captivating Q17, The points motivated me to play better
G2	Q16, The game was captivating Q21, I received sufficient feedback when things did not go according to my plan Q22, I received sufficient feedback when things went according to my plan Q23, The game motivated me to code more Q27, I would successfully learn to code Python by playing Pyception Q28, I feel that my coding skills were improved by playing Pyception
G3	Q15, The game felt like more than just task solving Q16, The game was captivating Q18, I find the game visually appealing. Q20, The game was more motivating than solving traditional tasks
G4	Q5, It was easy to get started with the game and exercises. Q6, Understanding how to control the game was easy. Q7, Navigating between satellites and Dr. Thereon was easy
G5	Q9, Earning points were hard. Q15, The game felt like more than just task solving. Q16, The game was captivating Q17, The points motivated me to play better. Q20, The game was more motivating than solving traditional tasks. Q23, The game motivated me to code more Q29, My motivation towards learning to code is increased.

Table 11.2: Goals and their relation to the questions of the questionnaire listed in Appendix C.

questionnaire regarding Pyception . After the test, the user is allowed to keep Pyception on their computer, and play it later if they choose to do so.

11.2.1 Eventualities

In a user test, it is likely that some unplanned situations will arise. The test leader must be prepared for imaginable situations, and if they arise, adapt the test in a manner that do not significantly impact the user or test result. Should the situation be too difficult to handle, the test leader must consider to abort the test.

11.3 Questionnaire

A valuable part of the user test is the information gathered for later use. A way to structurally gather results from a user test, is to have the users fill out a questionnaire after the test has been performed. A questionnaire will gather

subjective meanings from the users, which is important to have in mind when analyzing.

Most of the questions listed in the questionnaire, are multiple choice questions with five alternatives. The questions are written as statements with five alternatives ranging from strongly disagree via neutral to strongly agrees. The user must select one option that fits the best.

The questionnaire applied on Pyception, is attatched in Appendix C. The questionnaire is divided in six parts, and covers everything from metadata about the user to their experience with the user interface. The six parts are outlined in the following paragraphs.

Metadata The Metadata part captures general information concerning the user's history, previous experience and motivation.

Game Control This part identifies how the user perceive the game concept. Information regarding movement, game flow and difficulty are derived from these questions.

Game experience The experience the user receives from the game is captured by the questions in this part. Motivation and fun are important factors determined in this part.

Proficiency The proficiency part covers the user's overall experience in relation to the educational aspect of Pyception. The questions address how the user perceive the feedback provided, the appropriateness of the syllabus and future aspirations in the field of programming.

Other This part contains a single question where the users are encouraged to state their feelings of the game. This allows the user to supply valuable suggestions, that are othervise not covered by the multiple choice questions.

System Usability Scale (SUS) The SUS is a "quick and dirty" method, that allows a low cost assessments of usability in industrial systems [24]. From a completed SUS form, a SUSscore in the range 0 - 100, is calculated and gives a general overview of the usability of the system.

11.3.1 Calculation of Score

In order to calculate a score based on each questions, the alternatives are converted into a scale with a range of 1 to 5, where 1 represents strongly disagree and 5 represents strongly agree. A score of 3 represents a neutral response.

The average score of each question belonging to a goal is calculated, then the sum of all questions are totaled, and an average of that is the goal's final score. In order to get a significant reply, the average score must be 3.5 or higher, if the is goal to be considered a success. 3.5 is reckoned as significantly better score than the neutral reply.

11.4 Success Critera

This section list success criterias for the user experiment. The success criterias are derived from the research questions.

SC1 Fun and Educational Game

This success criteria is based on Research Question 2 (RQ2). RQ2 tries to determine if it is possible to create a serious game that is both fun and educational. Based on Section 11.1.2, RQ2 is broken down in three goals, G1, G2, G3. If the average of the three goals' score is higher than 3.5, this criteria is considered a success.

SC2 Game Motivation

This success criteria is based on Research Question 3 (RQ3). RQ3 tries to determine if more people will be motivated, if the learning process has an effortless start through a web based game. Based on Section 11.1.2, RQ2 is broken down in two goals, G4 and G6. If the average of the two goals' score is higher than 3.5, this criteria is considered a success.

SC3 Usability

A System Usability Scale (SUS) test is applied, in order to determine if the system is easy to use. According to [37], the average SUS score lies between 52 and 68. If this success criteria is to be met, Pyception has to score within this range or better.

Part V

Evaluation

Chapter 12

User Experiment Results and Analysis

12.1 Execution

The experiment were conducted according to the test framework discussed in chapter 11 and spanned several test sessions. Prior to starting the experiment, the users were informed that they could abort the test whenever they wanted, for whatever reason, and they would not have to state the reason why they chose to quit. When all practical issues were handled, the users were free to test the game. The experiment mainly ran smoothly, however, some changes to the plan were required in order to adapt to situations that arose. The situation and solutions will be discussed in section 12.1.3.

12.1.1 Users

The users that participated in the user experiment were recruited from two groups. Some testers were friends of mine who study Informatics at NTNU, while the other users study other subjects with with little or no programming courses.

In total 14 users with different coding skills participated in the user experiment. The test user's skills and motivation are displayed in figures

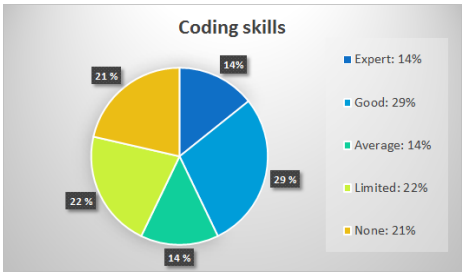

Figure 12.1: Chart displaying the skills possessed by the users.

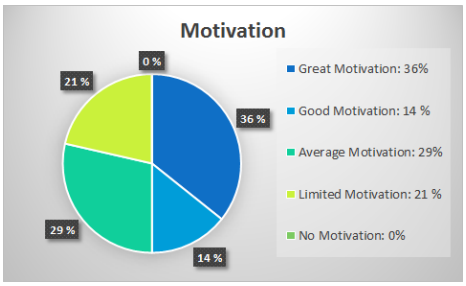

Figure 12.2: Chart displaying the motivation of the users.

12.1.2 Environment

The experiments were conducted at three locations, NTNU campus Gløshaugen, NTNU campus Dragvoll and Berg dormitory. The chosen environments were selected due to being where the participating users usually reside, creating a relaxing environment for the users. The users with the best coding skills were found on NTNU campus Gløshaugen, while the users with the least coding skills resided at NTNU campus Dragvoll.

12.1.3 Challenges

While conducting the first experiment sessions, one slightly significant problem occurred. The server installation process were not rigid enough, causing the setup on the user's computer to fail. As a test leader, I was prepared for such a situation. Two spare laptops with Pyception were brought to the experiment location, so when the installation failed, the users were able to execute the test on the laptops supplied. The later experiment sessions were mainly conducted on the same laptops and the installation process were improved.

During the experiment, some users suffered from buggy or poorly written exercises. To overcome these situations, I provided help to the users on the spot. In cases of ambiguous tasks, an explanation were given to clarify. In cases of buggy and unsolvable exercises, a cheat code were provided.

Apart from the mentioned situations, the experiment went smoothly.

12.2 Results

This section will address each goals on individual pages. The goals are introduced by stating the relevant questions to the particular goal, followed by the results from the user test. The questions are identified in section 11.1.2, table 11.2

An average score for each question is computed and visualized in the charts.

12.2.1 G1 Is the game fun ?

Results

The related questions regarding G1 are listed in table 12.1 below, and the results are presented in fig 12.3

#	Question
Q15	The game felt like more than just task solving.
Q16	The game was captivating
Q17	The points motivated me to play better

Table 12.1: Questions related to G1

Chart 12.3 shows the average score of each question. The red column to the far right in the chart shows the goal's average score.

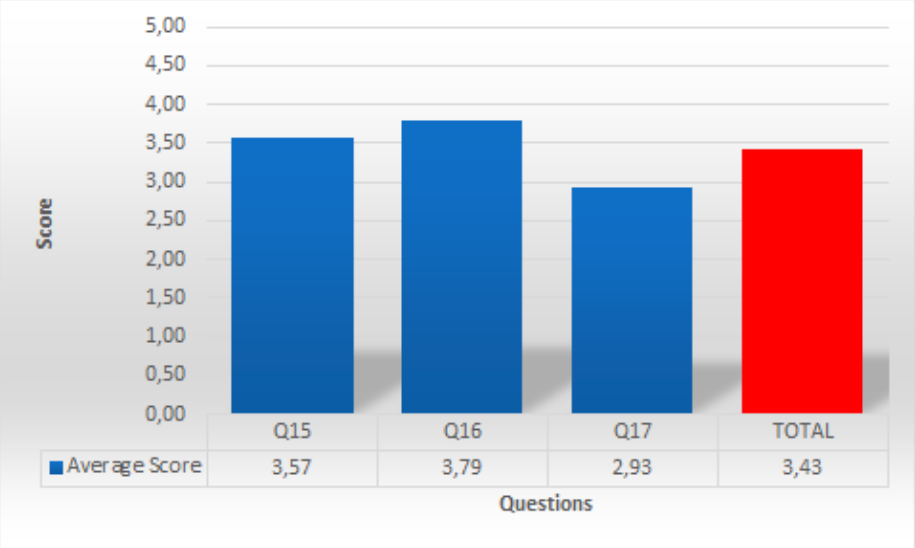

Figure 12.3: G1

Analysis

Based on the findings in section 12.2.1, the users feel the game is more than just task solving, with an average score of 3.79. An average score of 3.79 indicates that the need to be a skilled space pilot, in addition to a knowledgeable Python coder, is an attractive combination.

The points were meant as an incentive for the players to perform better. However, with an average score of 2.93, the incentive has not been clear or beneficial enough for the players to care about. A score of 2.93 is close to neutral, which means the incentive did not impact the game in a negative manner.

With an overall score of 3.43, the goal have room for improvements, and does not meet the criteria for a successful goal. A low average does not count for everything, some users found the game enjoyable and stated "Det var gøy,

vil ha fleire baner :)", which translates to "the game was fun, I'd like to play more levels :)".

12.2.2 G2 Is the Game Educational?

Results

The related questions regarding G2 are listed in table 12.2 below, and the results are presented in fig 12.4

#	Question
Q16	The game was captivating
Q21	I received sufficient feedback when things did not go according to my plan
Q22	I received sufficient feedback when things went according to my plan
Q23	The game motivated me to code more
Q27	I would successfully learn to code Python by playing Pyception
Q28	I feel that my coding skills were improved by playing Pyception

Table 12.2: Goal ID 2 and the related questions

Chart 12.4 shows the average score of each question. The red column to the far right in the chart shows the goal's average score.

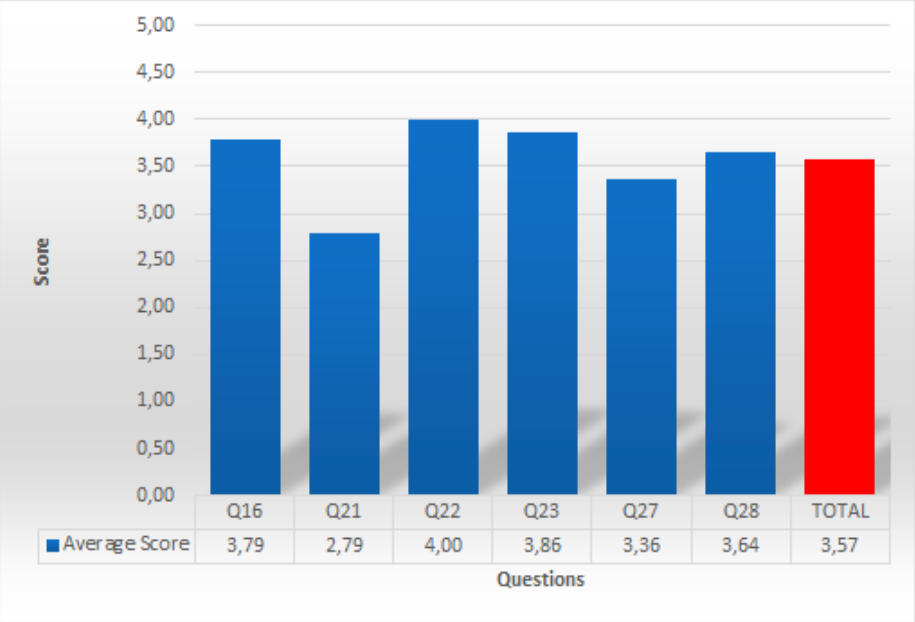

Figure 12.4: G2

Analysis

Presenting feedback on exercises that the user has trouble solving, has been a severe challenge in the project. The freedom of coding gives an exercise a numerous different ways to perform the same task. Predicting the approach chosen by the user, and presenting clues to what is wrong and how to mend it,

was not easy. This is reflected in the feedback from users, with an average score of 2.79.

The feedback of tasks where the users solve the task as intended, is better appreciated. When a task is solved, the user feels that satisfactory feedback is provided, which is reflected with a score of 4.00.

The educational aspect of the game scores an average of 3.57, this proves that users find the game educational. The exercises poses a challenge for the users who does not know how to code, and for experienced coders.

12.2.3 G3 Do Pyception feel more like a game than programming tasks?

Results

The related questions regarding G3 are listed in table 12.3 below, and the results are presented in fig 12.5

#	Question
Q15	The game felt like more than just task solving.
Q16	The game was captivating
Q18	I find the game visually appealing
Q20	The game was more motivating than solving traditional tasks

Table 12.3: Goal ID 3 and the related questions

Chart 12.5 shows the average score of each question. The red column to the far right in the chart shows the goal's average score.

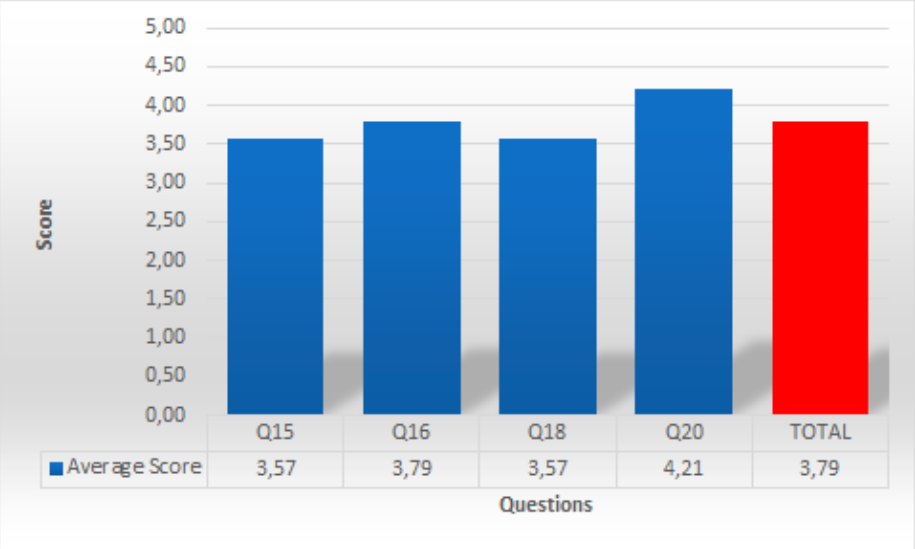

Figure 12.5: G3

Analysis

The user experiments proves that the users find more motivation through the game, compared to traditional task solving. In Figure 12.5, Q20 "The game was more motivating than solving traditional tasks" scores an average of 4.21 where 5.00 represents "strongly agree". This result can be seen as clear indicator that Pyception brings more to the table than traditional task solving.

Upon being asked if the game is something more than traditional task solving, the users opinions were more divided. The question may have been poorly formulated, as "something more" may be ambiguous. However, with an average

score of 3.57, Pyception gives the users a new experience when trying to learn Python.

With a total average of 3.79 users agree that Pyception feel more like a game than programming tasks.

12.2.4 G4 Is it easy to get started with the game?

Results

The related questions regarding G4 are listed in table 12.4 below, and the results are presented in fig 12.6

#	Question
Q5	It was easy to get started with the game and exercises
Q6	Understanding how to control the game was easy
Q7	Navigating between satellites and Dr. Thereon was easy

Table 12.4: Goal ID 4 and the related questions

Chart 12.6 shows the average score of each question. The red column to the far right in the chart shows the goal's average score.

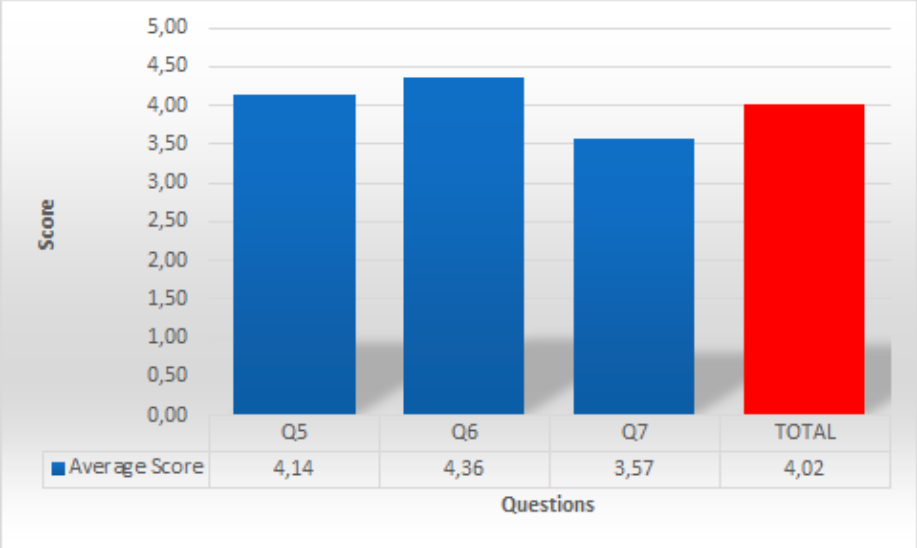

Figure 12.6: G4

Analysis

The users agree that understanding how to control the game, was easy. With an average score of 4.36, Q6 is the statement that the users agree with the most. Controlling the spaceship and docking with satellites is easily understandable, however observations during the experiment suggest that the mastery of controlling the spaceship was harder, as some players had a hard time avoiding Dr.Thereon. This is reflected in Q7, "navigating between satellites and Dr.Thereon was easy", which receives an average score of 3.57

The total average of G4 is 4.02. A core value of the project has been to create an easily accessible game, making this finding positive reading.

12.2.5 G5 Is the game motivating?

Results

The related questions regarding G5 are listed in table 12.5 below, and the results are presented in fig 12.7

#	Question
Q9	Earning points were hard
Q15	The game felt like more than just task solving.
Q16	The game was captivating
Q17	The points motivated me to play better
Q20	The game was more motivating than solving traditional tasks
Q23	The game motivated me to code more
Q29	My motivation towards learning to code is increased.

Table 12.5: Goal ID 1 and the related questions

Chart 12.7 shows the average score of each question. The red column to the far right in the chart shows the goal's average score.

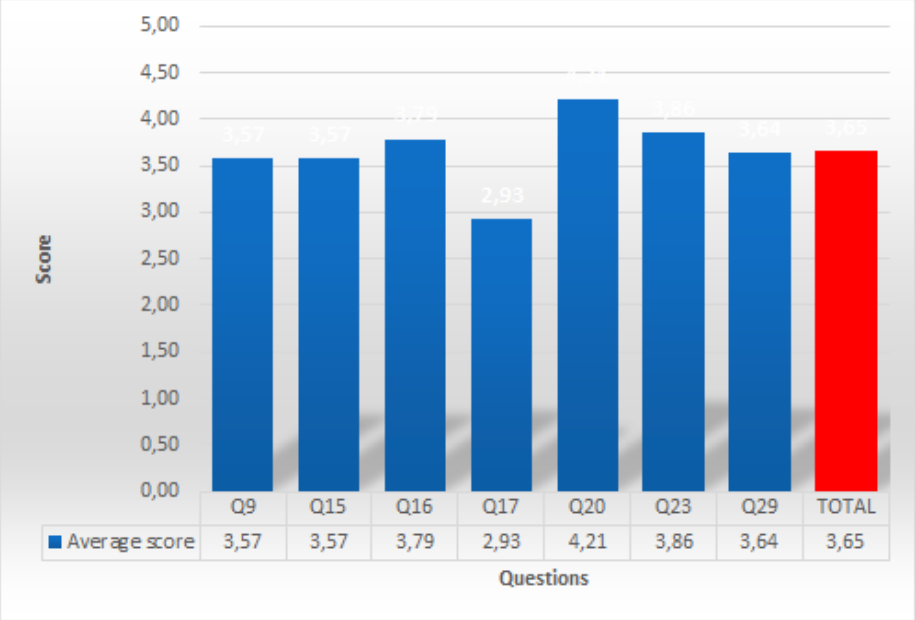

Figure 12.7: G5

Analysis

A game feature meant to be motivational was the points and score system. However as mentioned in section 12.2.1,the appreciation of the points were moderate with an average score of 2.93.

The motivational factor of the game was mentionable, as users rated that the game motivated them to code more. The findings are based on Q23, with an

average score of 3.86. The users also clearly find the game more motivational than traditional tasks, as discussed in section 12.2.3 and illustrated in figure 12.7

With an average score of 3.65, the users agree that the game is motivational.

12.3 System Usability Scale

Table 12.6 presents the results of the SUS schema. The "Average Score" column represents the average point value of what the users voted, while the "Sum Question" column represents the calculated SUS score. At the bottom of the table, the SUS scores are added together in a total of 71.96 points out of 100.

#	Question	Average Score	Sum Question
1	I think that I would like to use this system frequently	3.43	6.07
2	I found this system unnecessarily complex	2.21	6.96
3	I thought the system was easy to use	4.00	7.50
4	I think that I would need the support of a technical person to be able to use this system	1.79	8.04
5	I found the various functions in this system were well integrated	3.64	6.61
6	I thought there was too much inconsistency in this system	2.14	7.14
7	I would imagine that most people would learn to use this system very quickly	4.00	7.50
8	I found the system very cumbersome to use	1.71	8.21
9	I felt very confident using the system	3.71	6.79
10	I needed to learn a lot of things before I could get going with this system	2.14	7.14
	Total score		71.96

Table 12.6: Pyception SUS score

12.4 Success Criteria

This Section presents an anaysis of the success criteria defined in Section 11.4

12.4.1 SC1 - Fun and Educational game

All goals should receive a satisfactory score, of 3.5 or higher.

Results

Goal ID	Goal	Score
G1	Is the game fun?	3.43
G2	Is the game educational?	3.57
G3	Do Pyception feel more like a game than programming tasks?	3.79
Total		3.60

Table 12.7: SC1 summary

Analysis

This success criteria is related to Research Question 2, and tries to determine if it is possible to create a serious game that is both fun and educational. SC1 is measured by goals G1, G2 and G3. In order to fulfill the success criteria, the average of G1, G2 and G3 need to score above 3.5

From the results presented in Table 12.7, we can see that these criterias are met. This means that Pyception can be considered a fun and educational serious game.

12.4.2 SC2- Motivation

All goals should receive a satisfactory score, of 3.5 or higher.

Results

Goal ID	Goal	Score
G4	Is it easy to get started with the game?	4.02
G6	Is the game motivating?	3.65
Total		3.84

Table 12.8: SC2 - summary

Analysis

This success criteria is related to Research Question 3, and tries to determine if more people will be motivated, if the learning process has an effortless start through a web based game. SC2 is measured by goals G4 and G6. In order to fulfill the success criteria, the average of G4 and G6 must score above 3.5

From the results presented in Table 12.8, we can see that these criterias are met. This means that Pyception can be considered motivate users better than traditional tasks.

12.4.3 SC3 - Usability

All goals should receive a satisfactory score.

Results

The results of the SUS schema is presented in detail in Section, and table The calculated score is 71.96.

Analysis

The calclated score is within the parameters established in. This determines that the system is easy to use, and well appreciated.

Chapter 13

Difference between a novice and an experienced coder

In this chapter I will discuss interesting differences and similarities between a novice and an experienced coder, in the data generated by the user experiment.

Based on the questionnaire answers, I have chosen to evaluate the questions with the biggest and smallest leap. The questions and their total average are listed in table 13.1

Q5- It was easy to get started with the game and exercises Even though the users have different coding experience, they agree that starting the game was easy. This was presumed prior to the experiment, and the result is fortunate as one of Pyception's core goals is to have an effortless start.

Q6- Understanding how to control the game was easy Similar to Q5, Understanding the game control was not affected by their previous experience with coding.

Q15- The game felt like more than just task solving The lower score of users who are experienced in programming, indicate that the game feels similar to task solving to them. A reason to this may be that the coding tasks are perceived to be so simple that the challenge element, described by Malone[2], is eliminated. For novice programmers the challenge element of the tasks was prominent, and may have been perceived more as a part of the game.

Q20- The game was more motivating than solving traditional tasks Similar to Q15, the motivational aspect of the game may have been removed when the tasks are not perceived challenging. Pyception does not feature code highlighting, a feature that is well appreciated among experienced coders, and may have a been contributory cause to the lack of motivation.

Q26- I applied pre existing knowledge while playing the game Experienced coders benefiting from existing knowledge, is apparent. However, with an average score of 3.17, the novices claim to have taken advantage of preexisting

#	Question	Experienced	Novice	Difference
Q5	It was easy to get started with the game and exercises	4.17	4.00	0.17
Q6	Understanding how to control the game was easy	4.50	4.33	0.17
Q15	The game felt like more than just task solving	3.17	4.00	0.83
Q20	The game was more motivating than solving traditional tasks	3.83	4.50	0.67
Q26	I applied pre existing knowledge while playing the game.	3.17	4.67	1.50
Q27	I would successfully learn to code Python by playing Pyception	2.83	4	1.17

Table 13.1: The average score of experienced and novice coders, and the differences

knowledge. This is surprising, as I had anticipated a lower score. An explaination to this, may be that users with "limited" code skills are more aware that they apply some pre existing knowledge.

Q27- I would successfully learn to code Python by playing The curriculum displayed in Pyception would definitely be characterized as basic. This may be a turnoff for experienced coders, as they already know their basics and feel the game has nothing to offer them. Creating a game with curriculum that suits both novices and experienced coders is challenging, and the Pyception curriculum is purposely aimed at novices. Due to this fact Q27 has the most divergent answers in the questionnaire.

Chapter 14

Technical Analysis

This chapter is intended to analyse the prototype in regards to research question 4. RQ4 tries to figure out if the Python language is suitable for a web based serious game.

14.1 Architectural issues

In the planning phase of the project, some design choices were made.Thesis choices have been discussed in Chapter 9, and the architecture is discussed in Chapter 8. Among the choices made, was that of the systems architecture. The Pyception project has been developed with a server - client architecture where the game requires a server in order to perform the execution and evaluation of the server. The choice of this architecture has two important impacts on the system in relation to security and performance.

14.1.1 Security

The execution of user submitted code on the server is a major security risk. "Run native machine code on the client machine- an ultimate hacker goal and the definition of disaster"[30]. The security threat could be mitigated by filtering the user submitted code for code that could possible threat the server. However, the threat would still be present, as hackers constantly search for, and occasionally find, errors in deployed systems.

14.1.2 Performance

The tasks written for the Pyception prototype, if solved correctly, require only a tiny amount of computation. However, the nature of Pyception is to learn users how to code, and assuming that users not familiar with Python would create effective code all the time, is downright silly. When a lot of users simultaneously send bad code for evaluation, the servers would require tremendous processing capacity.

14.2 Python Issues

When learning how to code "while loops", all users sooner or later will try out a variant of the "while(True):"-loop, more or less willingly. Endless loops need to be stopped somehow, but this is a real challenge in Python.

A technical issue present in Python is the lack of good ways to implement timeouts[36]. Two options are commonly proposed to solve this; multi-threading or operating system specific signals. By spawning a thread specifically for the function that should timeout, and use the "timeout" argument of "Thread.join" to implement the time limit. Sadly this does not work, and Python has no option to kill a thread.

The other option is to use operating specific signals. The "signal.SIGALRM" function of the "signal" module is an option that may work, but is tricky. A significant drawback of the solution is that "SIGALRM" is a Unix specific, and will not work in a Windows environment.

14.3 Existing Solutions

As mentioned in Section 5.2.2, Codecademy has an online gamified Python service. Their solution is based on a JavaScript implementation of the Python interpreter. There is no documentation available online that specify exactly how the implementation has been performed. However, when executing demanding computations the local utilization of the processor increases. This indicate that the computations is performed at the client side. In addition, if a computation runs for too long, a confirma box is displayed, giving the user the option to stop the process, and reload the JavaScript

14.4 Summary

By examining the Pyception implementation, we can immediately conclude that it is possible to create a web based serious game. However, the implementation chosen has some serious flaws regarding security and performance issues.

The Codecademy implementation has handled the issues present in Pyception. By moving the code exception to the client, the security threat of running user code on the server, is eliminated. Other threats still exist, however that is not the scope of this thesis. The performance issue is handled in much the same manner, by moving the computational load to the client. Codecademy also has implemented a timeout feature, which handles endless loops.

By combining the features of Pyception, a serious game, with the architectural design of Codecademy, client side execution, it is evident that Python as a language is well suited for use in an online serious game.

Chapter 15

Requirement fulfillment

This chapter will review the project requirements as described in Chapter 7. The following sections address the functional requirements and the non functional requirements. Each of the requirements will be listed and evaluated individually.

15.1 Functional Requirements

The four requirement groups identified in Section 7.1 will be the basis for this section. Each of the groups will be listed with their respective requirements. These requirements are important, in order to create a fun and educational game.

15.1.1 Game

FR 1.1 It should be possible to start playing without an installation process

Part of Research Question 3 states that starting the game should be effortless. This is implemented by coding Pyception as a JavaScript web based game, with no installation or registration process.

FR 1.2 The game should have an intrinsic fantasy, where my the player's action affects the playing eld, and the playing field affects the play.

The player affects the game by solving exercises, consequently changing the appearance of the satellites from unsolved red, to solved green. The game affects the play mainly through the enemy, Dr.Therion. Dr. Thereon has the ability to kill the player and reset all exercises as well as corrupt the players solved exercises.

FR 1.3 The game should proceed to the next level when the player interacts with the power source after all the tasks are correctly solved.

FR 1.3 is fulfilled by the player navigating the spaceship to the power source when all satellites are powered on. If the speed is sufficiently low, less than 3, the player is allowed to dock with the power source and proceed to the next level.

FR 1.4 The game should store tasks in a database.

All of the exercises of Pyception, is stored in a MySQL database, and are available through the Django web framework.

FR 1.5 The game should retrieve tasks from the database.

The exercices stored in the MySQL database, are retrieved by the Django framework, and sendt as a JSON object to the gameclient.

15.1.2 Playing Field

FR 2.1 As a player I want to be able to navigate the space ship freely on the playing field

When the playing field is displayed, the player is able to move as he or she wishes. The space ship is controlled by using the arrow keys on the keyboard. There are no objects hindering the players movement on the playing field, however, Dr. Thereon may kill the player.

FR 2.2 As a player I should not be able to navigate the spaceship outside the playing field.

The game client has implemented boundaries along the edges of the playing field. If a player tries to fly out of the playing field, the spaceship will hit the border and have its speed reversed, bouncing the player back into the playing field.

FR 2.3 As a player I want to be able to dock with all satellites

It is possible to dock with both solved and unsolved satellites, provided the speed is low enough. If the speed is greater than the dock limit of 3, a warning message is displayed, informing the player that he or she needs to slow down in order to dock. Docking with a satellite will bring up the coding interface.

FR 2.4 As a player I want to not be able to dock with the powerstation when some tasks are unsolved

If a player tries to dock with the powerstation with some satellites still glowing red and unsolved, the player is denied docking and is informed by a warning message. The warning message states the number of solved satellites, and total number of satellites.

FR 2.5 As a player I want to be able to dock with the power-station when all tasks are solved

In order to dock with the power station and proceed to the next level, all satellites have to be solved and glow green. The docking still requires the player to keep a slow pace, or a warning message is displayed.

FR 2.6 There will be an enemy on the playing field at all times.

The enemy, Dr.Thereon, has a navigation algorithm that is based on the satellites. A satellite is always Dr.Thereon's target, and he moves closer to the satellite with every iteration of the game loop. Upon reaching the satellite, Dr.Thereon selects a new satellite on random. Based on this algorithm Dr.Thereon will never move out of the playing field.

FR 2.7 If the player is in range of the enemy, a penalty will be applied. If a player appears closer to Dr.Thereon than the range of his laser cannon, Dr. Thereon fires on the player. A message stating that the player was killed is displayed close to Dr.Thereon, and after a few seconds, the level is reset to it's initial state with all satellites unsolved. In addition a penalty of 1000 points is applied.

FR 2.8 The enemy will sporadically ruin the satellites and mess up the code

Approximately 20% of the time Dr. Thereon arrives at a solved satellite, he will corrupt the exercise code and disable the satellite. The player is deducted 100 points from the score when a satellite is corrupted.

FR 2.9 As a player I want to be able to earn points and keep track of the score

The player will earn 100 points by solving a satellite. To the far right, below the playing field, the player's score is displayed.

FR 2.10 As a player I want to be able to see a measure of completion of the current level

To the left, below the playing field, the number of operational satellites is displayed.

15.1.3 Code Interface

FR 3.1 As a player I want to enter my own code

By docking with a satellite, the code interface is displayed. The code interface has four text areas, where the largest of them is labeled "CodeArea". In the CodeArea text box, the player is free to enter his or her own code.

FR 3.2 As a player I want to have my code evaluated

The coding interface has three buttons. The "Execute" button will send all code in the "CodeArea" text box to the server for evaluation.

FR 3.3 As a player I want feedback regarding what is wrong in my code

The exercise examining system capture the output of the player code. In order to provide good feedback, the solution code of the exercises must take the output of coding errors into account and display a relevant feedback message. This requirement has received some attention during the user experiment, as some tasks did not correctly anticipate common mistakes, and thus did not provide sufficient help.

FR 3.4 As a player I want feedback when my code is correct

When an exercise is solved, a label in the top right corner of the coding interface changes from "Status: unsolved" on red background to "Status: solved" on green background.

FR 3.5 As a player I want to be able to reset my code

The coding interface has three buttons. The "Reset" button will fetch the exercise from the server, and reset the "CodeArea" text box to the template code.

FR 3.6 As a player I want to have my code stored so I can go back and continue later

Upon leaving a satellite, the content of the "CodeArea" box is stored in the satellite. If a player returns to the satellite later, the stored code is brought back.

FR 3.7 As a player I want to have my finished tasks stored so I can go back and review it later

Upon leaving a solved satellite, the content of the "CodeArea" box is stored in the satellite. If a player returns to the satellite later, the stored code is brought back. It does not matter if the satellite is solved or not, any code is stored for the duration of the game.

FR 3.8 As a player I want to be able to debug code

If a bug is present in the player code when the code is executed, the Python output related to that bug, is presented in the "code output" textbox.

15.1.4 Lecturer

FR 4.1 As a lecturer I want to have a password protected administration account with access to the exercises.

The Django framework takes care of administrator users, enabling lecturers to have a password protected account.

FR 4.2 As a lecturer I want to be able to create new tasks

Through the Django framework administrators are able to create new tasks. In order to add the excercises to a level, the lecturer has to add the task in the "settings.js" file. This requirement is fulfilled, however, the implementation is poor, and improvements will be suggested in Chapter 17, future work.

FR 4.3 As a lecturer I want to be able to edit tasks

Through the Django framework administrators are to edit tasks.

FR 4.4 As a lecturer I want to be able to delete tasks

Through the Django framework administrators are to delete tasks.In order to remove the excercises from a level, the lecturer has to remove the task from the "settings.js" file. This requirement is fulfilled, however, the implementation is poor, and improvements will be suggested in Chapter 17, future work.

15.2 Non-functional Requirements

Accessibility One of Pyception's core goal is to make an easy accessible game. This means that the user should not have to go through an installation, nor a registration process..

The questionnaire users filled out after the user experiment addresses the accessibility in Q5, “It was easy to get started with the game and exercises”. Q5 scored an average of 4.14, indicating the users agree, that the game is easily accessible.

Compatibility The server is implemented in the Django framework and the game client is coded in JavaScript. Django is compatible with both windows and Unix platforms, and JavaScript is implemented in all major web browsers. This makes Pyception compatible with multiple environments, however this has not been tested and verified, as compatibility is a low priority in this project.

Performance The development of Pyception has not accounted for performance, as this is a low priority non-functional requirement.

Security The security issue present in Pyception, is detailed in Section 14. Apart from that, security has not received any attention, as it is a low priority non-functional requirement.

Usability The game should be easy to use and to understand. As indicated in section 12.3 the total average SUS-score is 71.96. In addidtion, the questionnaire has several questions that address usability, among them Q6 “Understanding how to play the game was easy”, which has a satisfying result, with an average of 4.36.

The usability of Pyception is satisfying,a SUS score of 71.96 is above average[37], but still has room for improvements.

Part VI

Conclusion

Chapter 16

conclusion

16.1 Research Questions

16.1.1 Research Question 1

"Which systems related to Pyception exists today, and do they exhibit any features that should be included in Pyception?"

In Section 5.2, I examined systems that were similar to Pyception. The section were rounded of by identifying important aspects of the examined surfaces, fantasy and challenge. The key aspects have been a factor when developing sketches and eventually the game.

The challenge aspect is featured in Pyception with two features, the enemy and the exercises. The enemy moves about the playing field, and the player has to avoid it or the entire level is reset. Python is thought by a series of tasks with escalating difficulty, creating a challenge for the players. It is possible to edit the tasks in order to adapt the curriculum for a more experienced crowd.

The fantasy aspect is implemented in Pyception through the game set in space. The story provided is intended to motivate the player to help save a fictional world, and all game elements have space themed graphics.

16.1.2 Research Question 2

"Is it possible to create a serious game based on programming that is both fun and educational?"

Most of this thesis is based on the work trying to create a serious game that was both fun and educational. Research question 1 founded the basis for a prototype. In Chapter 6 several sketches were discussed, and a final concept was created. The concept was turned into requirements, described in Chapter 7, design choices are listed in Chapter 8, and an architecture is described in Chapter 9. Finally, the implementation is discussed in Chapter 10. The product developed in the process described, was Pyception, a game I believed is fun and educational.

In order to objectively determine if the game is both fun and educational, a user experiment has been conducted, as described in Chapter 11. Using the results from this experiment, listed in Chapter 12, it is possible to draw some conclusions.

The user experiment gave through observations and a questionnaire answers regarding this Research Question. In order to make a quantitative measure of the success of this Research Question,I formulated the Success Criteria 1, SC1 "Fun and Educational game" 11.4. In Section 12.4.1 the results state that this Success criteria was fulfilled with a score of 3.60, tipping over the 3.50 limit, which in Section 11.3.1 was determined the minimum score for the criteria to be met.

As in every prototype, there is room for improvements in Pyception. However, based on these findings and results, Pyception can be considered a serious game that is both fun and educational

16.1.3 Research Question 3

"Will more people be motivated to learn programming if the learning process has an effortless start through a web based game?"

The implementation of Pyception require no installation or registration process. The only thing required of the user is to enter a web address, which must be considered effortless. The data from the user experiment questionnaire support this claim. Section 12.2.4 lists Q5 "it was easy to get started with the game and exercises", which received a score of 4.14 out of 5, where 5 represents "strongly agree".

The motivational aspect of Pyception was investigated in the user experiment, through Success Criteria 2, "Motivation". In the analysis in Section 12.4.2, SC2 receive 3.84 of 5.00 points, which is above the minimum margin of 3.5 established in Section 11.3.1.

An interesting observation in the data, is that there is a significant difference in the motivation experienced between a novice coder and an experienced coder. Q20 states that "the game was more motivating than solving traditional tasks" receive a score of 4.50 out of 5 from the novice, indicating that Pyception is best at motivating novice coders.

With the success criteria fulfilled in addition to the motivational observation, Pyception gives a clear indication that users are more motivated towards learning Python if the game has an effortless start.

16.1.4 Research Question 4

"Is the Python language a suitable language for a web based serious game?"

The Pyception implementation is a proof that it is possible to implement a serious web game based on teaching Python. Although, Chapter 14 discuss the security and performance issues present in Pyception. Despite these findings, an implementation is suggested where Pyception's issues are handeled.

The Pyception implementation is a proof that Python is a possible, but not suitable language for a web based serious game. The data provided in this thesis is not comprehensive enough to conclude that Python is suitable, as there is no implementation of the system suggested in Chapter 14.

16.2 Project Evaluation

16.2.1 Pre-study

The pre-study is presented in Part III and consists of two main parts; previous work and technology choices.

In the previous work I present a research of already existing educational web services, and analyse them for their focus on one or more of the aspects of challenge, fun or curiosity. I also take a look at different game genres and which of these genres that would suit Pyception.

During this project, a lot of time were invested in coding the game. Due to this, the report mostly reflects all the work done by creating the game.

Finding suitable technologies for the development of the prototype was important. Many technologies were considered, and the ones used are listed and discussed in Part III.

The findings from this part have helped me creating a combination of technology, design and game genre that would suit a serious game.

16.2.2 The Engineering Method

Section 3.2 presents different approaches to performing software research, proposed by Basili et al. [3]. The most fitting approach for this project was the engineering method. The engineering method consists of observing existing solutions, propose improvements, measure and analyse, and repeat the process until no improvements appear possible. Due to the limited time available in this project, only one iteration were performed.

The first step, observing other solution, were performed as a part of the prestudy. Part IV describes the proposed solution, the process of implementing and at the end, a user experiment as an analysis process.

The user experiment were performed, and is summarized in the following subsection. Due to the mentioned time constraint, no improvements as a result of the user experiment, were performed in the prototype. However, proposed changes are discussed in Chapter 17.

16.2.3 The User Experiment

The user experiment was executed with 14 contestants. Even though the number could have been higher, 14 users are enough in order to create a representative result. By executing a user experiment, a lot of feedback was received. However, as the user experiment came late in the process it was more of a user evaluation of the final product. The users were quite satisfied, which is stated in Chapter 12.

Chapter 17

Future Work

This chapter presents some remaining work, and lists features that could improve Pyception.

17.1 Finalizing Pyception

The development of Pyception has consumed a great deal of the time available in this thesis. Some issues still remain in the project due to time constraints, or design flaws.

17.1.1 Security

The chosen architecture with server side execution of code, is a security threat. Unless this threat is properly handled, Pyception can not be deployed online and accessible for all. This is thoroughly discussed in Chapter 14, and the two proposed solutions are detailed below.

Whitelisting By filtering all submitted code for malicious code, it is possible to mitigate the security risk to some degree. By applying this solution, hosting Pyception would still be a risky venture, however, far safer than today's implemented solution.

JavaScript Python Interpreter If Pyception implement a client side Python interpreter, the security threat of running user submitted code on the server, is eliminated.

17.1.2 Settings

The implementation as it is today, store information of levels in a JavaScript file. If a lecturer whish to add or remove a task to a level or add a level, it is necessary to edit the JavaScript file.

By implementing the settings in the Django server, it would be possible for a lecturer to handle level info in the same interface that is used to edit the exercise content.

17.1.3 Full Browser Support

The game is not displayed equally in all browsers, and consequently looks better in Chrome, than in other browsers.

17.1.4 Exercises

There is a total of 22 exercises used in Pyception today. These exercises span from the easiest "hello world"-type of task, to harder exercises with nested control structures. The span in the curriculum is too wide for only 22 exercises, so more exercises should be added.

17.2 Improving Pyception

In this Section, I present some features I believe will improve Pyception.

Audio Add background music and sound effects.

Multiplayer Enale players to cooperate against a common enemy.

Gamification Add achievements to the game.

Social Media Enable users to share their score with friends on social media, such as Facebook or Twitter.

Lenient Penalty When Dr.Thereon kills the player, the players find it annoying to loose all of the solved tasks, and suggest to only disable a few, or half of all solved satellites.

17.3 Research Improvements

The user experiment performed in this thesis featured 14 test users who played as they felt for an hour, then completed a questionnaire. The data were enough to prove the research questions. However, it would be interesting to research if a serious game could teach programming better that traditional task solving.

Bibliography

[1] NTNU *Course information*. http://www.ntnu.edu/studies/courses/IT1103

[2] Thomas W. Malone *What makes things fun to learn? Heuristics for designing instructional computer games* 1980

[3] Victor R. Basili. *The experimental paradigm in software engineering*. In Proceedings of the International Workshop on Experimental Software Engineering Issues: Critical Assessment and Future Directions, pages 3–12, London, UK, 1993

[4] Brad Paras, Jim Bizzocchi *Game, Motivation, and Effective Learning: An Integrated Model for Educational Game Design* 2005

[5] Pablo Moreno-Ger, Daniel Burgos, Iván Martínez-Ortiz, José Luis Sierra, Baltasar Fernández-Manjón *Educational game design for online education* 2008

[6] Philippe B. Kruchten *The 4+1 View Model of Architecture* November 1995

[7] Markus Montola, Timo Nummenmaa, Andrés Lucero, Marion Boberg, Hannu Korhonen *applying game achievementsystems to enhance user experience in a photosharing service* 2009

[8] Thomas W. Malone *Heuristics for Designing Enjoyable User Interfaces" Lessons from Computer Games* 1980

[9] Sebastian Detering, Dan Dixon, Rilla Khaled, Lennart Nacke *From game design elements to gamefulness defining gamification* 2011

[10] NTNU *Course information* http://www.ntnu.edu/studies/courses/TDT4110/2012

[11] M.F.Sanner *Python: A programming language for softwareintegration and development*

[12] Gregor Richards, Sylvain Lebresne, Brian Burg, Jan Vitek *An Analysis of the Dynamic Behavior of JavaScript Programs*

[13] MySQL AB *About MySQL* http://www.mysql.com/about/

[14] Django Software Foundation *Databases* https://docs.djangoproject.com/en/dev/ref/databases/

[15] Django Software Foundation *Django* https://www.djangoproject.com/

[16] Jeff Forcier, Paul Bissex, Wesly Chun *Python Web Development with Django* Addison-Wesley Professional, 2008

[17] Python *Python Programming Language – Official Website* http://www.python.org/

[18] JSON *Introducing JSON* http://json.org/

[19] CreateJS *EaselJS* http://www.createjs.com/EaselJS

[20] CreateJS *CreateJS* http://www.createjs.com/.

[21] World Wide Web Consortium *HTML* http://www.w3.org/community/webed/wiki/index.php?title

[22] World Wide Web Consortium *Cascading Style Sheets home page* http://www.w3.org/Style/CSS/

[23] International Standardation Organisation *ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs), Part 11: Guidance on usability.* March 1998.

[24] John Brooke *SUS - A quick and dirty usability scale* 1996

[25] KhanAcademy *KhanAcademy* https://www.khanacademy.org/

[26] Codecademy *Codecademy* http://www.codecademy.com/

[27] Code Hero *Code Hero* http://primerlabs.com/codehero0

[28] Robocode *Robocode* http://robocode.sourceforge.net/

[29] Stack Overflow http://stackoverflow.com/questions/7376727/html-5-game-development-tools/7379756#7379756

[30] Aviel D. Rubin, Daniel E. Geer Jr *Mobile Code Security* November 1998

[31] Tom S. Chan, Terence C. Ahem *Targeting motivation – adapting flow theory to instructional design* Journal of Educational Computing Research, v21 n2 p151-63 1999

[32] Open Game Art *Open Game Art* http://opengameart.org/

[33] David Münnich *notpron* http://notpron.org/notpron/

[34] raphidae *Try2Hack* http://try2hack.nl

[35] Jeff Sauro *Measuring Usability with the System Usability Scale (SUS)* February 2, 2011 http://www.measuringusability.com/sus.php

[36] Eli Bendersky *How (not) to set a timeout on a computation in Python* http://eli.thegreenplace.net/2011/08/22/how-not-to-set-a-timeout-on-a-computation-in-python/

[37] Jeff Sauro, James R. Lewis *correlations among Prototypical Usability Metrics: Evidence for the Construct of Usability* April 2009

Appendix A

Acronyms

AI Artificial Intelegence

API Application Programming Interface

GUI Graphical User Interface

MTV Model Template View

MVC Model View Controller

SUS System Usability Scale

RTS Real Time Strategy

RPG Role Playing Game

Appendix B

Installation Guide

In this appendix the instructions on how to set up, and remove, the Pyception system are outlined.

B.1 Installation

Step 1 Copy files.
Use "installation step1.bat" to copy files to C:\Pyception

Step 2 Install Django
Navigate to "C:\Pyception\InstallationFiles" and use the Bitnami Django stack to start the installation of Django. During the installer, some options need to be filled.

- Pyception only require the MySQL database, unselect SQLite and PostgreSQL.
- Install bitnami to the standard directory, "C:\BitNami\DjangoStack-1.4.5-0"
- Apache port; use 80 or something else, this setting is not vital.
- MySQL password must be "pythonia".
- Do not change windows setting to associate Python files (.py) to the bundeled Python
- Do not start an inital project
- Uncheck "Learn more about BitNami cloud hosting".
- Do not Launch BitNami DjangoStack

Step 3 Add Python to the path variable

- Run "sysdm.cpl" from command prompt.
- Select the "Advanced" tab

- Press "Environment Variables"
- In the system variables field, select Path, and click Edit.
- Add ";C:\BitNami\DjangoStack-1.4.5-0\python" to the end of the variable value
- Click OK on all dialogs to close them

Step 4 Create the mission database, and add the exercises to the database.
Right click and run as administator "installation step 4.bat". The file is located in "C:\Pyception\InstallationFiles"

Start the game server Right click and run as administator "startServer.bat". The file is located in "C:\Pyception"

Start the game Open your web browser and enter "http://127.0.0.1:8000/executer/". Preferably use the Chrome web browser.

B.2 Uninstallation

Remove Pyception Delete the folder "C:\Pyception" and its content.

Remove BitNami DjangoStack Use the uninstall available from windows control panel, Program and features.

Appendix C

Questionnaire

The questionnaire was originally written and answered in Norwegian, and has later been translated for usage in the thesis.

Metadata

1. My computer skills

   None, Limited, Average, Good, Expert

2. My video game skills

   None, Limited, Average, Good, Expert

3. My coding skills

   None, Limited, Average, Good, Expert

4. My motivation towards learning how to code

   No motivation, Limited motivation, Average motivation, Good motivation, Great motivation

Game Control

5. It was easy to get started with the game and exercises.

   Strongly disagree, disagree, neutral, agree, Strongly agree

6. Understanding how to control the game was easy.

   Strongly disagree, disagree, neutral, agree, Strongly agree

7. Navigating between satellites and Dr. Thereon was easy

   Strongly disagree, disagree, neutral, agree, Strongly agree

8. Repairing satellites were hard.

   Strongly disagree, disagree, neutral, agree, Strongly agree

9. Earning points were hard.

   Strongly disagree, disagree, neutral, agree, Strongly agree

10. The game was slow, lagging or lacked response.
    Strongly disagree, disagree, neutral, agree, Strongly agree
11. Technical issues ruined my game flow
    Strongly disagree, disagree, neutral, agree, Strongly agree
12. I wish i could cooperate with my friends, in order to beat Dr.Thereon.
    Strongly disagree, disagree, neutral, agree, Strongly agree
13. I wish i could play against, and beat, my firends.
    Strongly disagree, disagree, neutral, agree, Strongly agree.

Game Experience

14. The game was suitable in length and duration
    Strongly disagree, disagree, neutral, agree, Strongly agree
15. The game felt like more than just task solving.
    Strongly disagree, disagree, neutral, agree, Strongly agree
16. The game was captivating
    Strongly disagree, disagree, neutral, agree, Strongly agree
17. The points motivated me to play better.
    Strongly disagree, disagree, neutral, agree, Strongly agree
18. I find the game visually appealing.
    Strongly disagree, disagree, neutral, agree, Strongly agree
19. I would like to show my skills on social medias, such as facebook or twitter.
    Strongly disagree, disagree, neutral, agree, Strongly agree
20. The game was more motivating than solving traditional tasks.
    Strongly disagree, disagree, neutral, agree, Strongly agree

Prociency

21. I received sufficient feedback when things did not go according to my plan
    Strongly disagree, disagree, neutral, agree, Strongly agree
22. I received sufficient feedback when things went according to my plan
    Strongly disagree, disagree, neutral, agree, Strongly agree
23. The game motivated me to code more
    Strongly disagree, disagree, neutral, agree, Strongly agree
24. The code exercises were too hard.
    Strongly disagree, disagree, neutral, agree, Strongly agree

25. The learning curve were too steep
    Strongly disagree, disagree, neutral, agree, Strongly agree
26. I applied pre existing knowledge while playing the game.
    Strongly disagree, disagree, neutral, agree, Strongly agree
27. I would successfully learn to code Python by playing Pyception
    Strongly disagree, disagree, neutral, agree, Strongly agree
28. I feel that my coding skills were improved by playing Pyception
    Strongly disagree, disagree, neutral, agree, Strongly agree
29. My motivation towards learning to code is increased.
    Strongly disagree, disagree, neutral, agree, Strongly agree

Other

30. Any other thoughts regarding the game ?

System Usability Scale (SUS)

31. I think that I would like to use this system frequently
    Strongly disagree, disagree, neutral, agree, Strongly agree
32. I found the system unnecessarily complex
    Strongly disagree, disagree, neutral, agree, Strongly agree
33. I thought the system was easy to use
    Strongly disagree, disagree, neutral, agree, Strongly agree
34. I think that I would need the support of a technical person to be able to use this system
    Strongly disagree, disagree, neutral, agree, Strongly agree
35. I found the various functions in this system were well integrated
    Strongly disagree, disagree, neutral, agree, Strongly agree
36. I thought there was too much inconsistency in this system
    Strongly disagree, disagree, neutral, agree, Strongly agree
37. I would imagine that most people would learn to use this system very quickly
    Strongly disagree, disagree, neutral, agree, Strongly agree
38. I found the system very cumbersome to use
    Strongly disagree, disagree, neutral, agree, Strongly agree
39. I felt very confident using the system
    Strongly disagree, disagree, neutral, agree, Strongly agree
40. I needed to learn a lot of things before I could get going with this system
    Strongly disagree, disagree, neutral, agree, Strongly agree