Sergey Evlannikov

Development of Game Prototype

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

10 May 2014

Helsinki Metropolia University of Applied Sciences

Author	Sergey Evlannikov
Title	Development of Game Prototype
Number of Pages	45 pages
Date	10 May 2014
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructors	Niko Leskinen, Platform Operations Manager Miikka Mäki-Uuro, Senior Lecturer

The study is dedicated to game prototype development. The development process is considered from the standpoint of a programmer, game designer and artist. The goal of the study was to prepare a ready game prototype. The paper describes the process of creation of a game prototype using the Unity 3D engine. Also, a lot of attention is paid to the game design part.

The study was carried out by using an iterative development approach. First, the concept of the game is described in detail. Then, the subsequent iteration of development is described. Also the parts not included in the prototype are covered.

The aim of the study is to be of use to those who study the development of games and novice developers in general.

Keywords	Game development, Unity3D, game design, Photoshop, turn based games, C#, 2D games

Tekijä Otsikko Sivumäärä Aika	Sergey Evlannikov Peliprototyypin kehittäminen 45 sivua 10.5.2014
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmointi
Ohjaajat	Niko Leskinen, Platform Operations Manager Miikka Mäki-Uuro, Lehtori

Tutkielmassa käsitellään peliprototyypin kehitystä. Kehitysprosessia tarkastellaan ohjelmoijan, pelisuunnittelijan ja taiteilijan näkökulmasta. Tutkielmassa kuvataan peliprototyypin toteutus Unity 3D -työkalulla, mutta myös pelisuunnitteluun kiinnitetään huomiota.

Peliprototyyppi toteutettiin iteratiivisesti. Tutkielmassa kuvataan jokaisen iteraation sisältö. Tarkoituksena on antaa yleiskuva peliprototyypin kehittämisen vaiheista.

Tutkimuksen tavoitteena on antaa hyötyä niille, jotka opiskelevat pelien kehittämistä ja haluavat kehittyä alalla.

Avainsanat	Pelin kehitys, Unity3D, pelisuunnittelu, Photoshop, vuoropohjainen pelit, C#, 2D pelejä

Contents

Abbreviations

AI	Artificial Intelligence.
GUI	Graphical User Interface is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators.
IAPs	In-Application Purchases.
JPEG	Joint Photographic Experts Group
PNG	Portable Network Graphics
IP	Intellectual Property.
Ads	Advertisement.
API	Application Programming Interface.
SDK	Software Development Kit.
FSM	Finite State Machine.
Indie	Independent games are video games created by individuals.
KISS	Keep It Short and Simple. The KISS principle states that most systems work best if they are kept simple rather than made complex.

1 Introduction

The study is dedicated to game prototype development. A prototype is an incomplete version of an idea. A prototype contains only a part of the designed functionality. A prototype may be completely different from the final product. Using a prototype a designer can get early feedback from potential users. It allows estimating the validity of the initial idea.

Game development is a software development process which results in a game. A computer game is a game that involves human interaction with a user interface to generate visual feedback on a video device and profit for developer. Development is performed by one developer or by group of developers. A game development team can range in size from 20 to 100 members depending on the project. [1]

A development team includes the following roles:

A game designer is a person who creates the game design. Game design includes game play, the rules and structure of a game. A game designer has to have a clear understanding of the game concept and provide the game with narrative content for development team. In larger projects, there are often separate designers for various parts of the game, such as game mechanics, user interface, characters, dialogue, etc. The designer should have good imagination, logical thinking and understanding of the software part of the project.

A game artist is a person who creates game art. Art may be 2D oriented or 3D oriented. 2D artists may produce concept art, sprites, textures, environmental backdrops or terrain images, and user interface. 3D artists may produce 3D models, animation and 3D environment. Artists sometimes occupy both roles.

A game programmer is a software engineer who implements game codebases and related tools. An individual programming task may include physics, AI, managing of graphical content, integration of sounds, implementation of various games rules, UI, input processing, network communications, etc.

Creating a game can be done either by a large studio or a small group of independent developers (indie). The minimum game team must contain at least one artist and one programmer. With the increasing number of people the complexity of their interactions increases accordingly Brooks [2] discusses several causes of scheduling failures. The most enduring is his discussion of Brooks' law: Adding manpower to a late software project makes it last longer. A man-month is a concept of a unit of work proportional to the number of people working multiplied by the time that they work; Brooks' law says that this relation is a myth.

Complex programming projects cannot be perfectly partitioned into discrete tasks that can be worked on without communication between the workers and without establishing a set of complex interrelationships between tasks and the workers performing them.

Therefore, assigning more programmers to a project running behind schedule will make it last even longer.

The present study was implemented by one developer and two supervisors that allow minimizing communication channels. The disadvantage of this approach is the lack of time.

The study was performed for Rovio Entertainment Ltd video game developer and Entertainment Company. According to the company policy every employee can spend 10% on innovations. Such motivation technique was introduced at Google Inc. and was called Innovation Time Off. Part of assets (some images and sound effects) was taken from other projects.

Nowadays game market is very wide in scope and rapidly grows with billions of dollars turnover. Current market leaders have large potential for grow. Game companies focus on differentiating their portfolios and look for new opportunities to gain new customers. Technology progress has a huge impact on the video game industry by enforcing constant growth. Video games are a large and growing market, but many sectors have already matured. An increasing demand for fun and unique games on the customer side forces companies to search for new game ideas. The opportunity lies in completely new products and services. It is becoming increasingly important to prototype new game first, because they are becoming more complex. Prototyping is very important for collecting information about systems before they are actually built.

A lot of tools can be used to help in game development. The creation of a computer game is a complex task and game engines are built to make it easier. There are several available engines with different features. It is important to select the best engine for a specific game project. Unity3d engine was selected as the best suited for this study. Here the attention is focused on the scope of the game presented in this paper. Detailed design of the game is discussed later in section three. In game development, scope relates to game designing. Scope is a way to evaluate if a game is achievable within a certain timeframe. Scope is something to be decided before implementation starts, not something to be discovered post-facto. First of all, the game in question is a 2D game, because making 2D drawings is fast and easy. The project can be completed over one year. Modules of the game include a single player game, multiplayer mode and simple shop.

The report is written in five sections. The first section introduces the relevance of the study. Section 2 deals with the background of game development and the currently used development methods. Section 3 introduces the methods and tools selected for the implementation process. Section 4 includes the implementation results for solving the problems. Section 5 concludes the results. The main emphasis of this project is placed on prototype development (Section 4), as the most challenging part of the project.

2 Theoretical Background

Design is about making decisions in many fields design require engineering skill, prototypes help designers select the best solution. This section describes prototyping used for game design.

2.1 Software Prototyping

Prototype refers to incomplete versions of the software program. A prototype typically simulates only a few aspects of, and may be completely different from, the final product. Prototyping is a process of creating model for future applications in order to determine the validity of the application, its functionality and, in general, the concept of the application. If an application is developed by a third-party request, the client can also

show a prototype in order that he could monitor and make adjustments to application [3].

The prototype has an ability to get rid of misunderstanding between different specialists (manager, producer, designer, programmer, and customer) involved in the project, organize thoughts and avoid errors and unnecessary work still in the early stages of development. Application can be tested using the focus group; it helps to get useful information from potential users.

Prototypes can be analyzed along four dimensions [3]:

- Representation describes the form of the prototype, e.g., computer simulation.
- Precision describes the level of detail at which the prototype is to be evaluated; e.g., informal or highly polished;
- Interactivity describes the extent to which the user can actually interact with the prototype; e.g., watch-only or fully interactive;
- Evolution describes the expected life-cycle of the prototype, e.g. iterative.

Prototyping is a very important part of game development. Well-made prototypes help the producer assess the design and uncover practical issues and problems that a 2D design on paper cannot. Prototyping aids extensive testing and evaluation of consumer products. A fully functional prototype that closely represents the final manufactured solution can be sent for user trials. All possible testing is conducted on the prototype, and if the results are satisfactory, the design is finalized and the product goes in for mass production.

2.1 Development Methodology

There are two main approaches to software development. The goal of this section is to compare the differences.

The waterfall model is a sequential design process that emphasize completion of one task before proceeding to the next, progress is seen as flowing steadily downwards

(like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation, and Maintenance. These stages are shown in Figure 1.

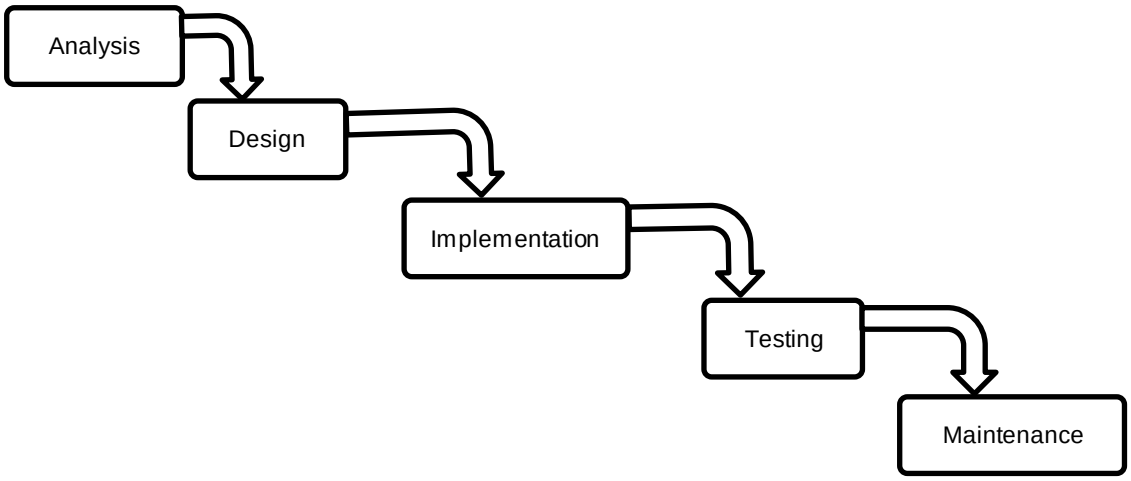

Figure 1. Waterfall model. Progress flows from the top to the bottom, like a cascading waterfall.

This approach has some advantages. The development team has to collect all of their requirements before production starts and that helps to prioritize the effort. The waterfall model can also help for planning and budgeting. This approach is highly risky, often more costly and generally less efficient than more Agile approaches.

The basic idea behind the iterative method is to develop through repeated iterations (see Figure 2), allowing software developers to take advantage of what was learned during development of earlier versions of the system. Design is modified at each iteration. Each iteration is typically 1-4 week long and while developers are implementing the design for the current iteration, designers are creating the requirements for the next iteration, to be completed at the same time as implementation of the current. When the current implementation is completed, it goes to QA to test, while development codes the next iteration, and so forth. This allows finding flaws in implementation and correcting any flawed assumptions [4].

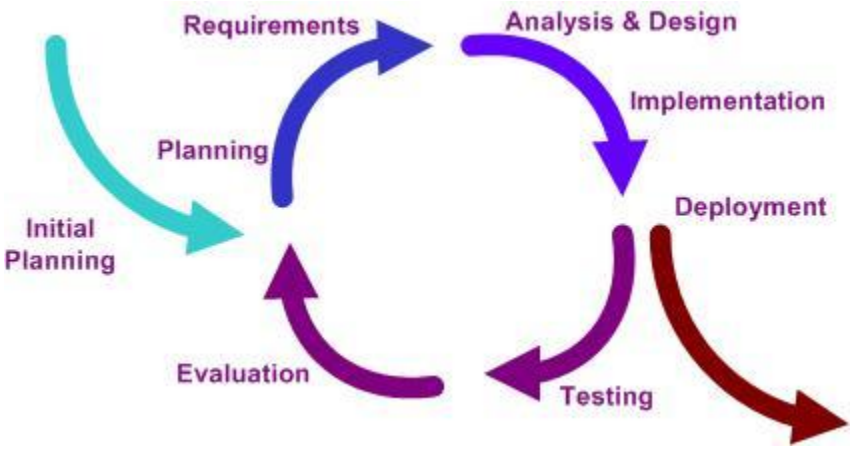

Figure 2. Iterative development.

2.2 Game Development

This section describes the basic game development concepts used in the present study. Game development is a very broad discipline and the majority of terms cannot be covered in a single paper.

2.2.1 Concept Document

Game development starts from a concept document. The concept document serves the purpose as a way to present a game concept. The concept document includes all the information produced about the game like the high concept, the genre of the game, gameplay description, features, setting, story, target audience, hardware platforms, estimated schedule, marketing analysis, team requirements, and risk analysis. High concept is a few sentences long description of a game.

When the concept of the game is ready, programmers begin to develop a prototype of the game.

2.3 Finite State Machine

The Finite State Machine is used by game developers to implement any behavior, for example the behavior of some units or complex objects.

FSM consists of four main elements:

- States that determine the behaviour
- The state transitions which define the transition from one state to another
- Rules and conditions that must be met for the transition
- Input states obtained externally or internally, according to the rules that may lead to a transition state

Advantages of employing finite automata are most notably evident in the case of applications for mobile devices requiring avoiding wasting screen space, memory, processing power and other resources.

2.4 Game Resources

Any game consists of many parts, but among them there are two main ones. First is game resources, and the second is the code.

Game resources are the graphic, music and other assets that are used to decoration of the game. Graphic files are used to store images. Typically, this is graphical representations of game objects, which are used in two-dimensional games.

JPEG is a standard of image file storage, designed specifically for the storage of digital images [5]. JPEG feature is the ability to compress image with loss of quality. There are 10 JPEG compression levels (1 to 10). PNG has different from the JPG format compression algorithms to compress an image without losing quality. PNG format good fit for images with clear color transitions. If the image has smooth color transitions - the best choice for its compression is JPG, but PNG is quite possible to use.

Texture - is a bitmap image that is superimposed on the surface of the landfill, which is part of a three-dimensional model. Today, three-dimensional models are created with a high level of detail - the modern computers have enough power for a realistic rendering of such models. In the games of yesteryear models usually look schematically, but even so allow texturing models look realistic enough. [9]

2.5 Genre and Audience

Information about the game genre and target audience is very important, as it allows understanding the features of the game, especially the future promotion of the finished product. Target group represented by users who may be interested in the game. Following are the types of players who meet in the virtual world:

Casual players

Casual players are regular players who have been in the game by accident, advertising, or at the invitation of a friend, or just stumbled on the link. They have not yet realized who they are within the game and are looking at whether there linger. Further, they either leave or move into the next type.

Non-paying players

These are players who for various reasons do not pay, but play. There are about 80-85 % in any game (up to 90-95 % in games with poorly implemented monetization). The reasons that they do not pay are different: from the strange principles such as ”I will not pay, because I want to unleash the money” to causes such as lack of ability to pay electronically. Most simply mind spending money on virtual goods. That some of these players, mainly spread negativity about micropayments, outraged at forums and blamed developers.

Donators

These are players who have been paying. They pay in different ways, but the average payment to paying for well-made and customized monetization is approximately $ 50 per month. Yes, many of them will pay $ 5, others $ 100, here shows himself perfectly Pareto principle.

Hard-core players

Hard-core in terms of monetization - is the most interesting type of players that creates maximum problems: they are constantly in the game, they played well and that get paid. There is a concept: a triangle time - skill - money, which says that a player or spend time or money, or the tops of the leaves due to skills. Hard-core spread this tri-

angle to shreds simply because they are often very time consuming, they are masters of the game and they buy goods and services in the game. It is through this type of player shareware MMO often receive negative fame and f2p model is considered as the essence of evil and greed of developers [8].

2.6 Monetization

Most of modern mobile games are very much focused on the monetization. There are ways to present the game for a wide audience:

1) Premium pricing model. Its paid games, user has to buy all content. All internal content usually available there. There is also Subscription model, where user has to pay for time. But they have almost equal sense.

2) Paid version (premium) where users first buy the game, then play it indefinitely. All internal content usually available there;

3) Subscription version, (pay-to-play). Are free, but for the games require a monthly payment (average $ 15). Example - Eve Online, WoW;

4) Fremium products that are free, but to access the full functionality that you need to buy an extended version or access to premium content (basically, this model is used for mobile products). Examples - any mobile game or application that is put in a stripped-down demo mode and has the " Buy full version ";

5) Free to play game. They are completely free, but inside they can buy different goods or services, is the most profitable payback model for games.

According to a report by an app-store analytics company, Distimo [7] freemium now accounts for 71% of Apple AppStore revenues in the US, up from somewhere around 50% last year, and rising. In Asia, freemium is 90% of App Store revenues.

3 Description of Method and Tools

This section describes the methods and tools used in the study, as well as the reasons for having selected the given methods and tools.

3.1 Methodology

Iterative methodology was selected as design methodology. Iterative development is an approach to building software (or anything) in which the overall lifecycle is composed of several iterations in sequence. Each iteration is a self-contained mini-project composed of activities such as requirements analysis, design, programming, and test. The goal for the end of iteration is an iteration release, a stable, integrated and tested partially complete system. To be clear: All the software across all the teams is integrated into a release of each iteration. Most iteration releases are internal, a baseline primarily for the benefit of the development team—they are not released externally. The final iteration release is the complete product, released to the market or clients [4].

This paper presents a simulated iterative process. The whole development process is described in Figure 3.

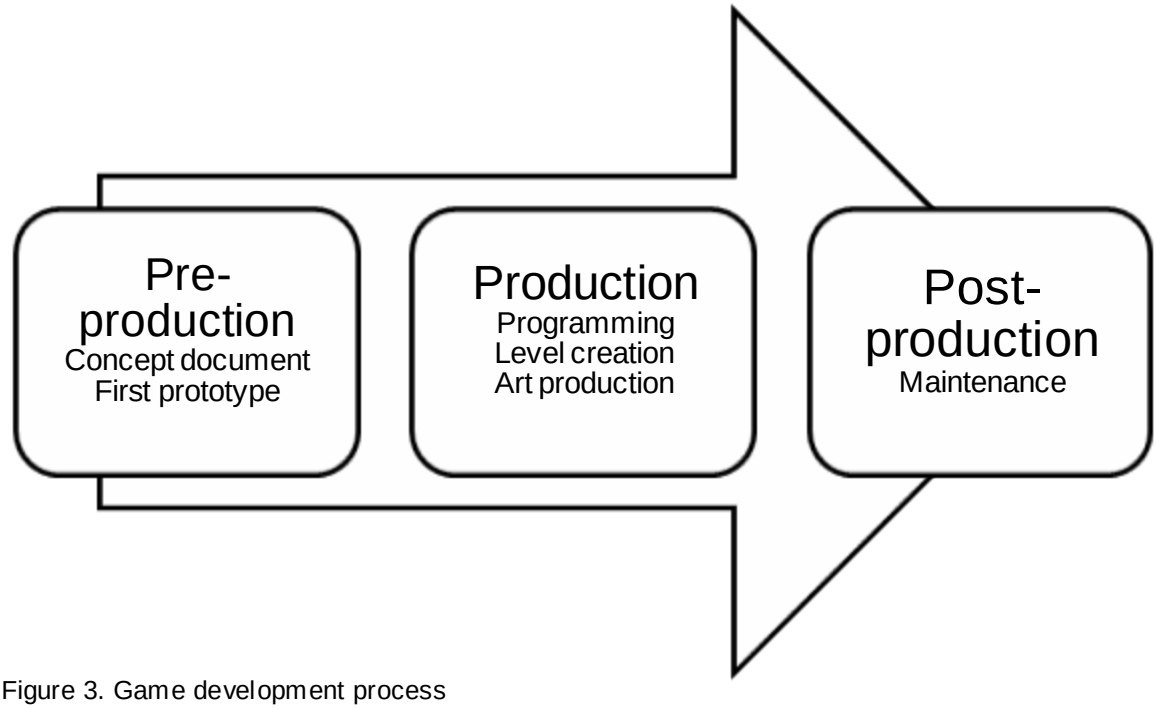

Figure 3. Game development process

In the next section the creation of the playable prototype is described as well as the execution of each sequential iteration, bearing in mind in particular what was added to the game, what was changed and which techniques were used.

3.2 Tools

In this subsection is described the software used in the game creating process. The selection of the working tools was done according to the KISS principle, which means that a high-level abstraction of the program that ensures the best results in shortest time was used. The KISS principle states that most systems work best if they are kept simple rather than made complex; therefore simplicity should be a key goal in design and unnecessary complexity should be avoided.

Unity3D

Without a doubt, as the development environment was chosen Unity3D. At the start of the study the latest release was Unity 4.2. In Unity3D context, KISS principle means that any problem can be solved either simply or not be solved at all, because Unity source code is black box and cannot be changed. Since the study was carried out by one person alone and in a short time, it was inadmissible to be distracted by the technical problems of the engine. At the start of the project two third-party plug-ins: NGUI and 2DToolKit were used. They were necessary for comfortable work with two-dimensional graphics and graphical user interface. Later, it was planned to completely transfer all the work to Unity3d. This is due to the fact that in the next Unity version 4.3 it is planned to introduce the native tools to work with 2d.

There are many game engines, e.g. Cocos2d, Unity3d, Box2D, Chipmunk etc. Despite the fact that Unity3d is mainly focused on the three-dimensional games, a significant advantage in favour of this engine was the multiplatform support.

Photoshop

Photoshop was selected as the graphic editor. This is quite a powerful development environment for creating beautiful sprites in a short time. The advantage with using Photoshop as the graphic editor program is creating simple vector images that can easily be painted by hand with a wide choice of brushes.

Google drive

During work on large projects is very wise to use a version control system such as SVN or GIT. On one hand, it allows saving work in time and provides powerful tools for de-

velopment in team. On the other hand, it nevertheless introduces more complexity and administrative costs of setting up these products. Since this study describes a fairly simple project with only one developer, it was enough to use virtual data storage, which prevents data loss and gives access to the same data in several locations. Google Drive can easily meet these modest needs.

Other tools

As additional software some simple utility for converting audio files was needed.

Also for writing the thesis Microsoft Word was needed and Microsoft Visio to create diagrams. Gimp was used as a free Photoshop analogue for image processing on work computer. Tools to decompile java libraries (for the study of Amazon plug-ins) and .apk archives (to check which section contains a manifest file) were applied. Also tools to prepare audio for Unity were necessary.

4 Results and Analysis

This section describes game design on a more detailed level and presents the results of each iteration.

4.1 Game Design

Games are born out of ideas. But in order to formalize the basic features it is necessary to describe the future of the game in a design document. In this section the design of the game is described. At this stage it is too early to start programming, but it must be clear what the game is about.

4.1.1 High Concept

High concept is a few sentences long description of a game. [8]

- Two opponents own towers and a set of random cards. They have to destroy each other, or rebuild tower to the heavens.

- Each turn players choose a card. Depending on the card, the player can launch creatures into the enemy's tower or strengthen their own positions.

- The bird's launch is completely similar as in the Angry Birds game. To win, the player has to destroy the enemy tower or rebuild their own to a certain height. For the protection, the player needs to upbuild the wall. Each card can cost in resources.

- Three types of resources will add to the game elements of a strategy to make it more interesting, on the other hand the simplicity and intuitiveness of the game will leave it accessible and very easy to play. It is as well a sequel to the hit Angry Birds game with a social component (multiplayer feature, or at least play against computer), but leaving it still the same fun and cool.

Figure 4 illustrates first idea of the game scene.

Figure 4. First game concept

4.1.2 Concept Paper

Concept papers are summaries of projects or issues that reflect the interests, experience and expertise of the writer or organization. Concept papers generally serve the purpose of providing in-depth discussion of a topic that the writer has a strong position on, usually with the intent of obtaining funding for that project from donors. [9]

Audience

The audience is really wide. First of all, it is targeted to Angry Birds fans, as the game uses the characters and the mechanics of the original game. Secondly, the game appeals to fans of the style of fantasy, as it contains the basic elements of medieval tales.

The gameplay is as simple as possible, so the game will be simple and understandable for casual players. The elements of the card and strategy game make the game interesting for the fans of various genres.

Gameplay

The episode consists of a large mountain, the top of which is necessary to achieve. Each sequence contains several levels. Choosing this level depends on the height of the lower of the tower.

Each level starts with terrain overview, and then the turn goes to player. As it is a turn-based game, players will take turns when playing. The player is given a choice of five cards. Cards are randomly chosen from the current deck. The current deck is initialized at the level start; it contains the card according to the level and bonus cards. There are two types of cards: the cards to charge the catapult/sling-shot and cards to rebuild the tower/wall (Later can be added cards with resources, cards allow player to change weak cards, some bonus and special cards).

If a player has chosen a construction card, his tower/wall will rebuild, and the turn goes to the second player. If a player chose a card with a creature/spell, the aiming procedure starts. The player selects the tension force and the launch angle and releases the projectile.

The level ends when one of the towers is destroyed or the height of one of the towers has reached a certain level. The episode ends when one of the towers is comparable with the height of the mountains, the player is shown the final episode of the comics and marked as completed one of the parties. The player can play the episode again.

The first version of the user interface is shown in Figure 5:

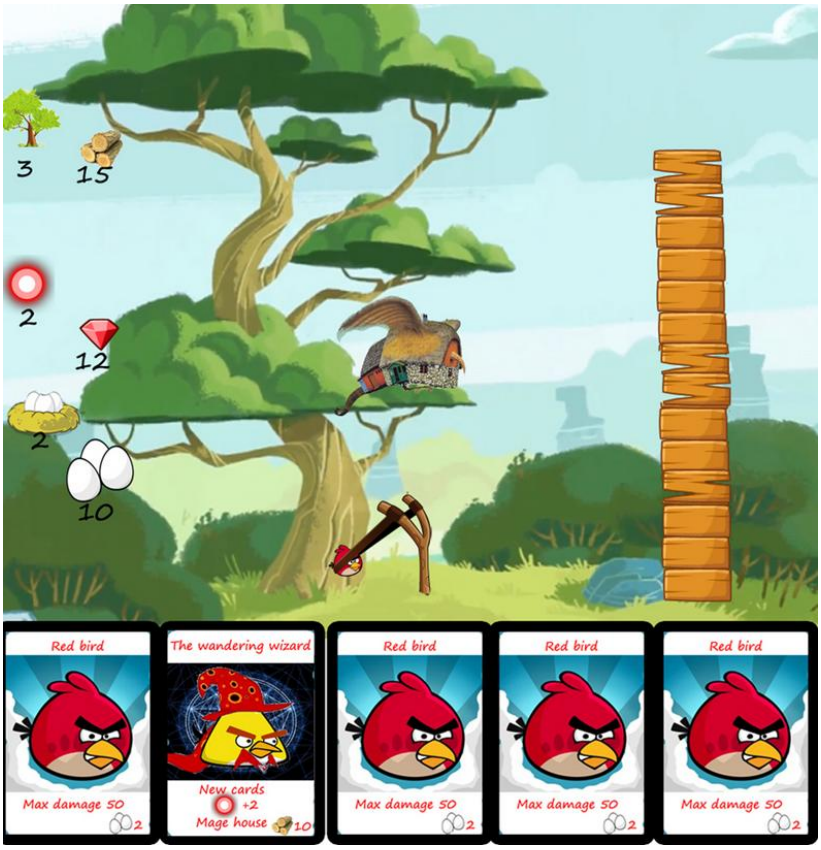

Figure 5. User interface first proto

Story

The story is about the new bad piggies character Pigzard who wants to capture the power of King Pig. For this, he proves that the mouth of the volcano island is a big pot of food. Being eternally hungry and foolish, the king pig orders to build a huge tower to get the food.

In the case of the Bad Piggies victory, King Pig jumps into the mouth of a volcano and Pigzard becomes the new king of the pigs. In the case of the victory of Angry Birds, they pull Pigzard into the mouth of a volcano.

The game has the following advantages:

1. Unique mechanics

The first advantage of this game is the original mechanics. This game is not a direct clone of any other game, but only combines the successful elements. Diversification may attract fans of different genres.

2. Angry birds' fans

At the beginning the game was planned in fantasy style with medieval castles, magicians and elves, then it was decided that the Angry Birds characters fit much better; mainly due to the fact that they already have a huge fan base.

3. Easy to maintain

The great advantage is achieved by reusing the same material. Instead of a large number of levels, new updates and increasing the size of the game, the game will contain some scenes. In the multiplayer mode, players will meet with other players on these scenes.

4. Monetization

There are

- Special consumable cards
- Stacks of resources
- Power Ups (like in Angry birds)

As the monetization model was selected free to play. It can be even more interesting in the case of multiplayer game: everyone wants to be the first.

4.2 Iteration 1. Playable and Simple

In this section the finite state machine for the game is identified. Also during the first iteration a simple playable version of the game was implemented.

4.2.1 State Machine

This game, as most of the other games, is a state machine. States are based on two scripts: Game Manager and Level Manager. While the game manager will describe the more common states such as level selection or main menu, the level manager describes the states of the game.

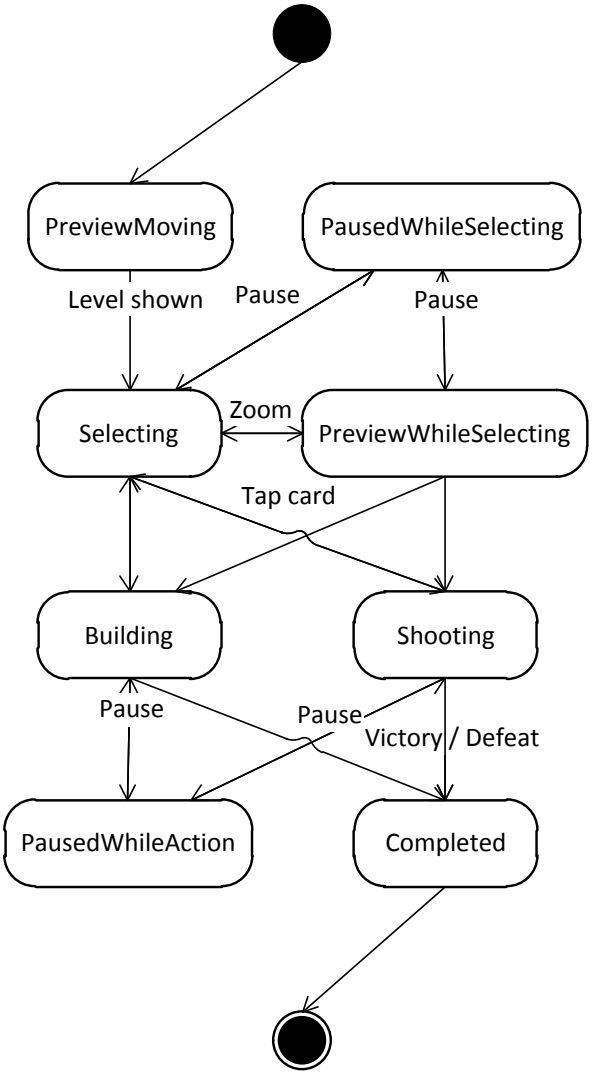

Figure 6. Level state machine

Here is presented the rough state machine [10] diagram, likely to change in the future (Figure 6).

4.2.2 Implementation

The purpose of the first iteration is a working prototype of the game with no frills. From the above-described states was implemented only the level manage. Also, the script to manage the catapult and a data structure that stores information about the status of the tower were created.

4.2.3 Slingshot Script

The most important script that defines the gameplay is a slingshot script. This is a simple script that describes what happens when the player clicks next to the charged slingshot (or catapult) and the process of aiming and object launching by giving impetus in the right direction. Also, this script is responsible for activation of special abilities and auto-destruction of the object when it stops. Resource usage of the original game was difficult because the Angry Birds uses box2d physic engine and port code from one engine to another could be more difficult than writing new code from scratch.

4.2.4 Result Screenshot

Figure 7 demonstrates the result of first iteration.

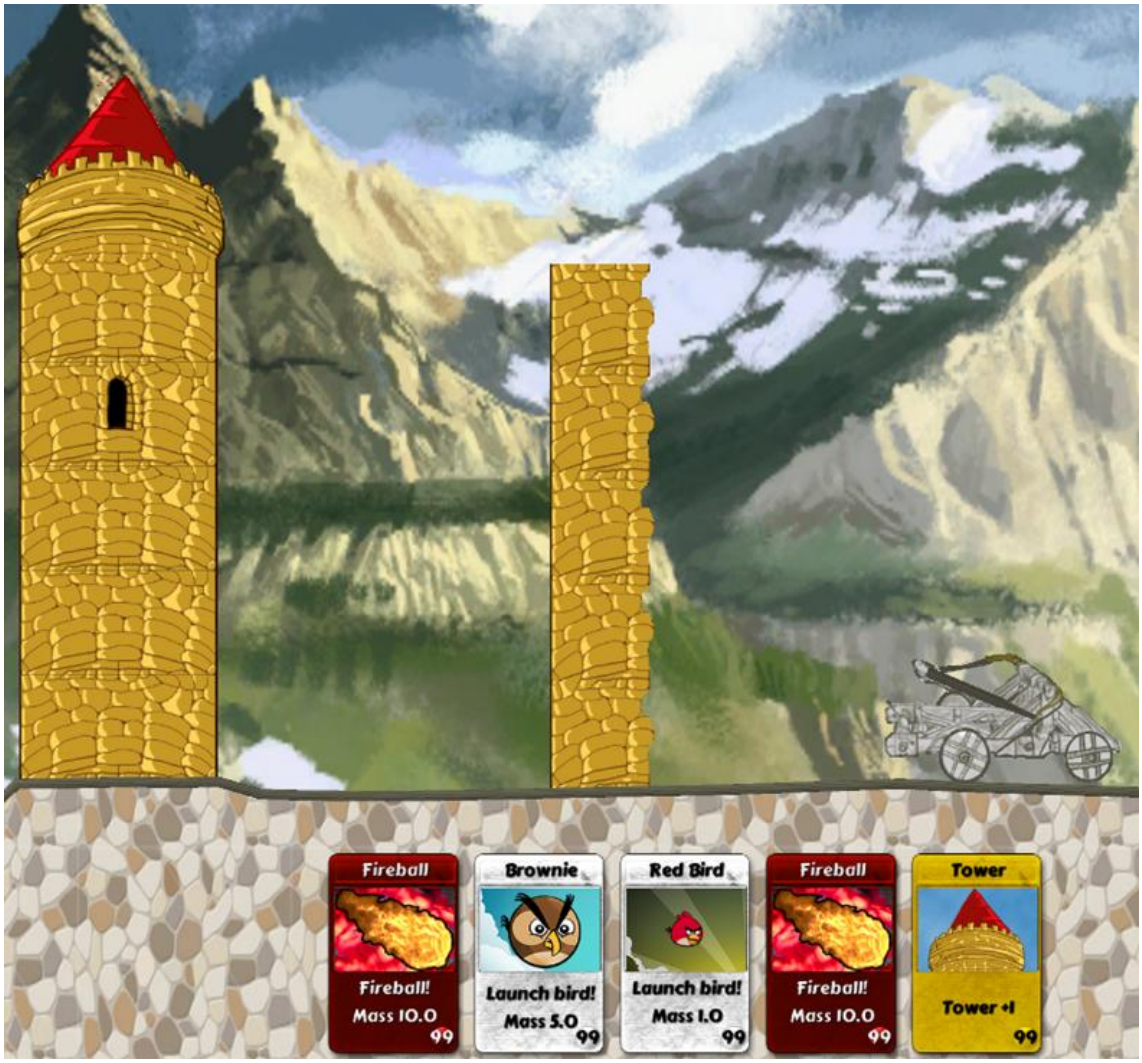

Figure 7. First iteration game result

Most of sprites were redrawn at next iterations. User Interface scaled incorrectly, all elements are misplaced, but already at this stage the basic idea of the game is demonstrated.

4.3 Iteration 2. Camera and Terrain

In this section work on such important concepts as terrain and camera control are described. It is also described how to create a seamless texture, and furthermore the process of the creation of the stone wall texture is explained.

4.3.1 Terrain

Despite the fact that the terrain plug-in was already imported during the first iteration, serious work on the terrain was begun at the second iteration.

For this simple game it would have been enough to use rectangles as terrain. But luckily in the open source there is a powerful tool for Unity [11]. It is a quite powerful 2D terrain generator that uses the principles of Unity 3D terrain generation and it can be used as it is. In the code only the maximum tile texture sizes were changed.

A seamless tile texture can be created in Photoshop. Next, is described how it was achieved. First, blank stones were drawn by hand. In fact, they are quite simple curves (see Figure 8).

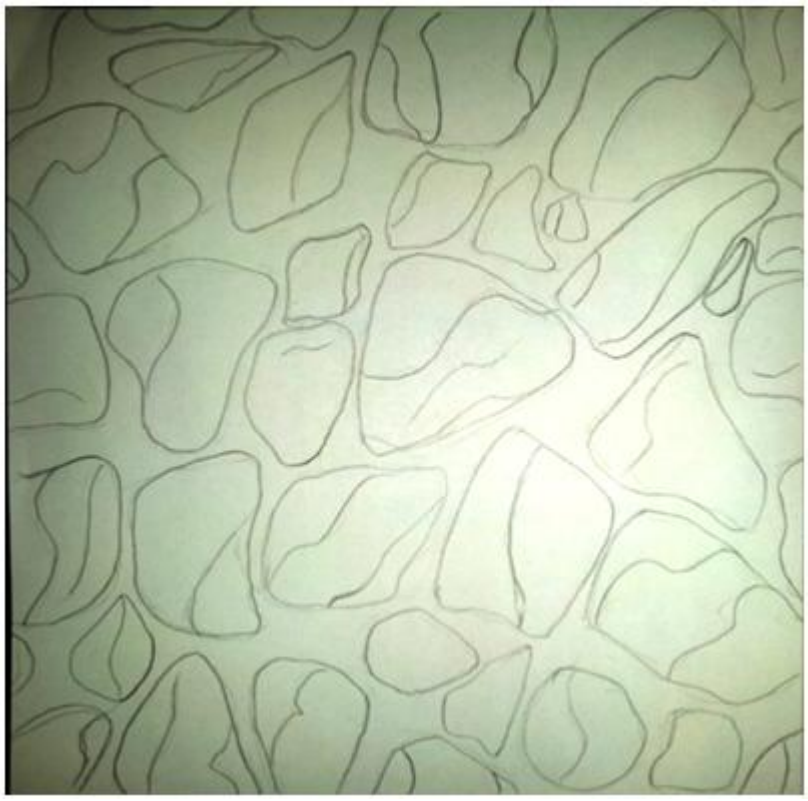

Figure 8. Sketch

Then the image was vectorized. In fact, it can be simply lead around by a brush. In general, the vectorized form is straighter and the picture is resizable.

To make it seamless, Photoshop offset filter was used to move the image to a half of its length and then all defects were corrected. The result is the next texture (see Figure 9):

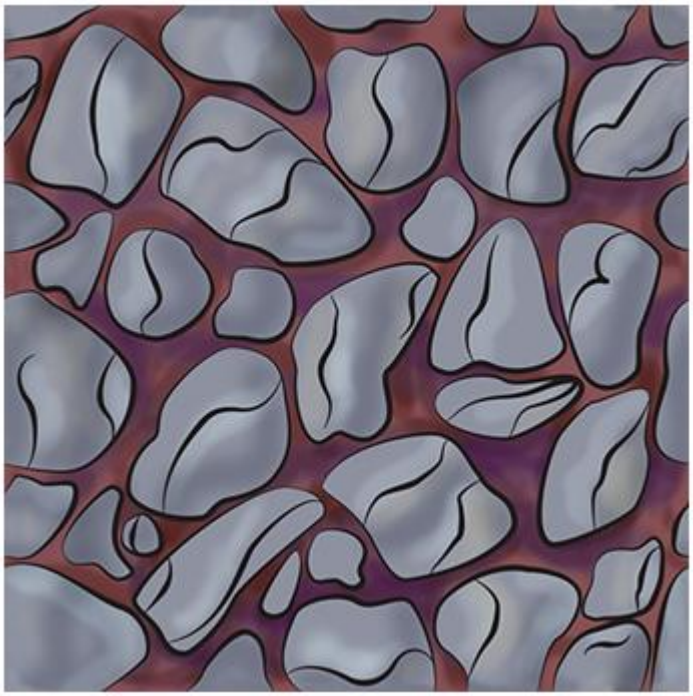

Figure 9. Terrain rock texture

Next the black stroke was removed, since zooming out black lines turned into separated black pixels spoiling the image. The same method was used to create a texture of grass.

4.3.2 Camera

The basic idea for a 2D game is that camera is in orthogonal projection. Orthogonal camera can show near and distant objects in the original size. Indeed, why would one need a third dimension in the 2D game? Unity3D is an engine for creating three-dimensional games, but it does not stop from creating beautiful two-dimensional scenes, and the Z axis can serve as a convenient solution for problems with overlapping objects.

So, in the camera script there are two main parameters: the cameras position itself and the orthographic size. In Unity the orthographic size is half of the vertical size of the viewing volume. The horizontal viewing size varies depending on the viewport's aspect

ratio. But this game has landscape orientation, the idea is to see the whole level (for example 100 units in width), so it was necessary to use the following orthographic size:

Outsize = Level Width*2*Screen Height/Screen. Width

Camera position should be in the centre of the level.

Preview -> Selecting -> Preview While Selecting ->Building / Aiming and shoot -> Enemy selecting -> Enemy Building/Aiming and shoot

Preview. This state should review whole battlefield at level start, camera should stay on some special level points (enemy position, bonuses locations and terrain traits).

Selecting. At this state, the camera is pointed at the tower of the current player, cards are shown. It is the default state after turn is changed.

Preview While Selecting. At this stage cards are shown; player can observe whole level, zoom in and zoom out.

After one card is selected, the camera goes into one of the following states:

Building. Camera focused on built object (tower or wall) center.

Shooting. A big disadvantage and problem of the prototypes from the previous iteration was too small a launched object, especially on small screens. The reason of it was to show the enemy's tower to make the aiming process simpler. Complex shapes were distorted. To fix it in the next iteration, the plan is to redraw the launched objects. During the shooting state the camera is focused on the slingshot while the player aims. Then camera follows the launched object as long as it does not disappear.

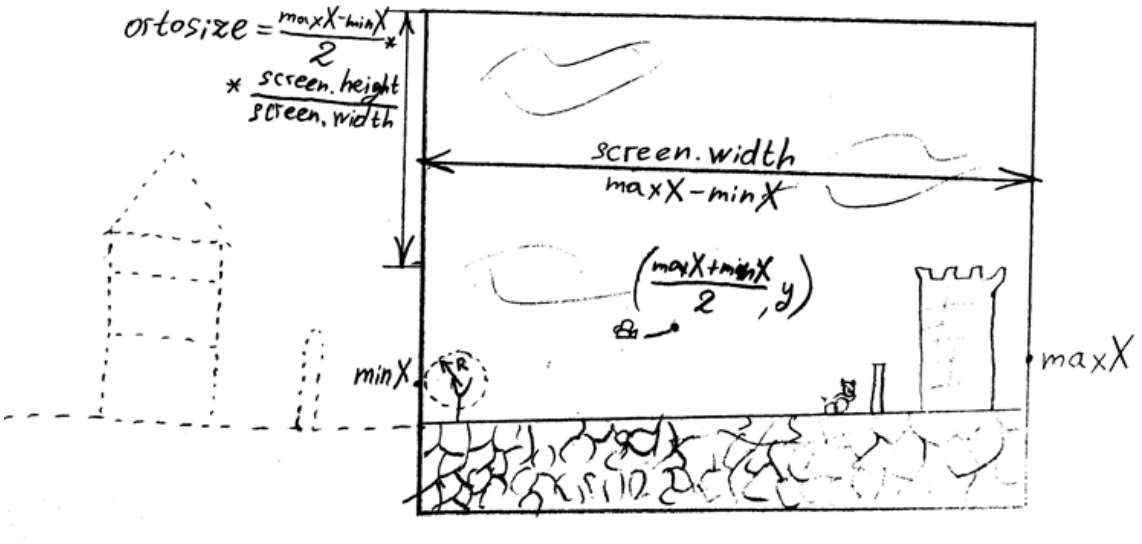

Figure 10. Schema of screen size.

Special attention should be paid to the point when the player aims. This situation requires that in the screen there are the slingshot and the enemy's tower visible. This point imposes some limits on the level: too large a level has no place on a small screen. The main idea of the screen size is demonstrated in Figure 10.

4.3.3 Images

In this subsection, the process of images creating is described. Most of the images were created in the following way. A small part, for example, backgrounds, skies and buttons was just filled with standard Photoshop tools.

The basic idea about the graphics creation is to make it as simple as possible. Since no experienced artist was involved in this project, some challenging monsters, and the human characters can look unprofessional, to say the least. Besides, the complex composition is difficult to represent on a small screen of a mobile device. Roughly the same logic regards the animation. Small scripts to process frame sequences can be created, as it does not take up too much memory; the problem is that in the frame drawing itself. Therefore, there is extremely simple animation of blinks and emotions expressions of characters. In any case, animation is out of scope of the current project.

Step 1. Hand drawing (see Figure 11). Of course drawing directly on the tablet can be faster, but the real pencil is much more convenient and easier. In addition, this step adds a sense of scale, due to which one would want to work thoroughly on a drawing.

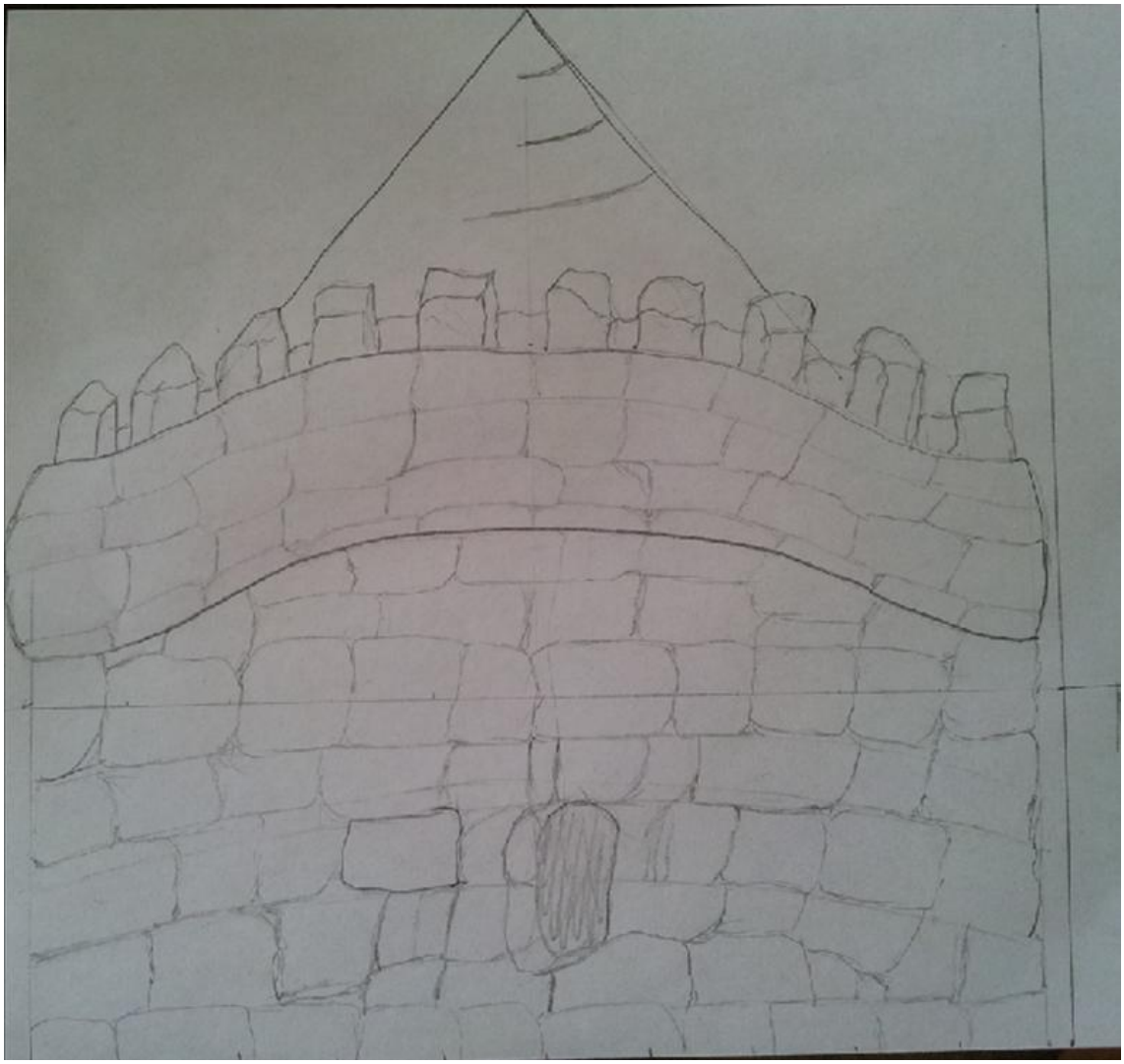

Figure 11. Hand drawing

Step 2. Image vectorizing. It was done manually using the pen tool in Photoshop. It is possible just to cut around the picture, but the vector shape is perfectly flat, which makes it easy to change the image size. Raster images are made of pixels. A pixel is a single point or the smallest single component in a display device. Vector images are mathematical calculations from one point to another that form geometrical shapes. The result of the vectorization is shown in Figure 12.

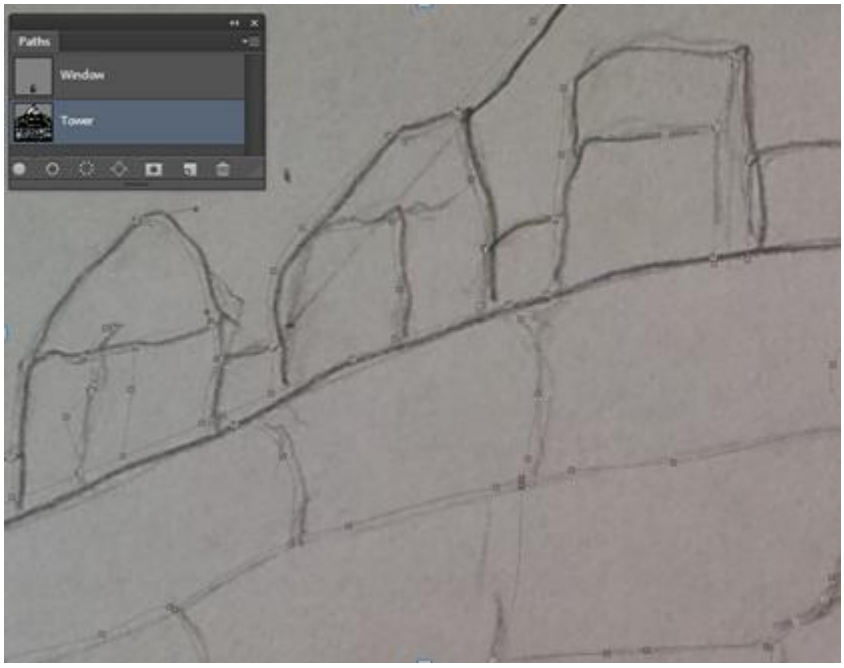

Figure 12. Vectorisation

The third step is a path stroking. Photoshop allows simulating the pressure of the brush, which gives good results, see Figure 13.

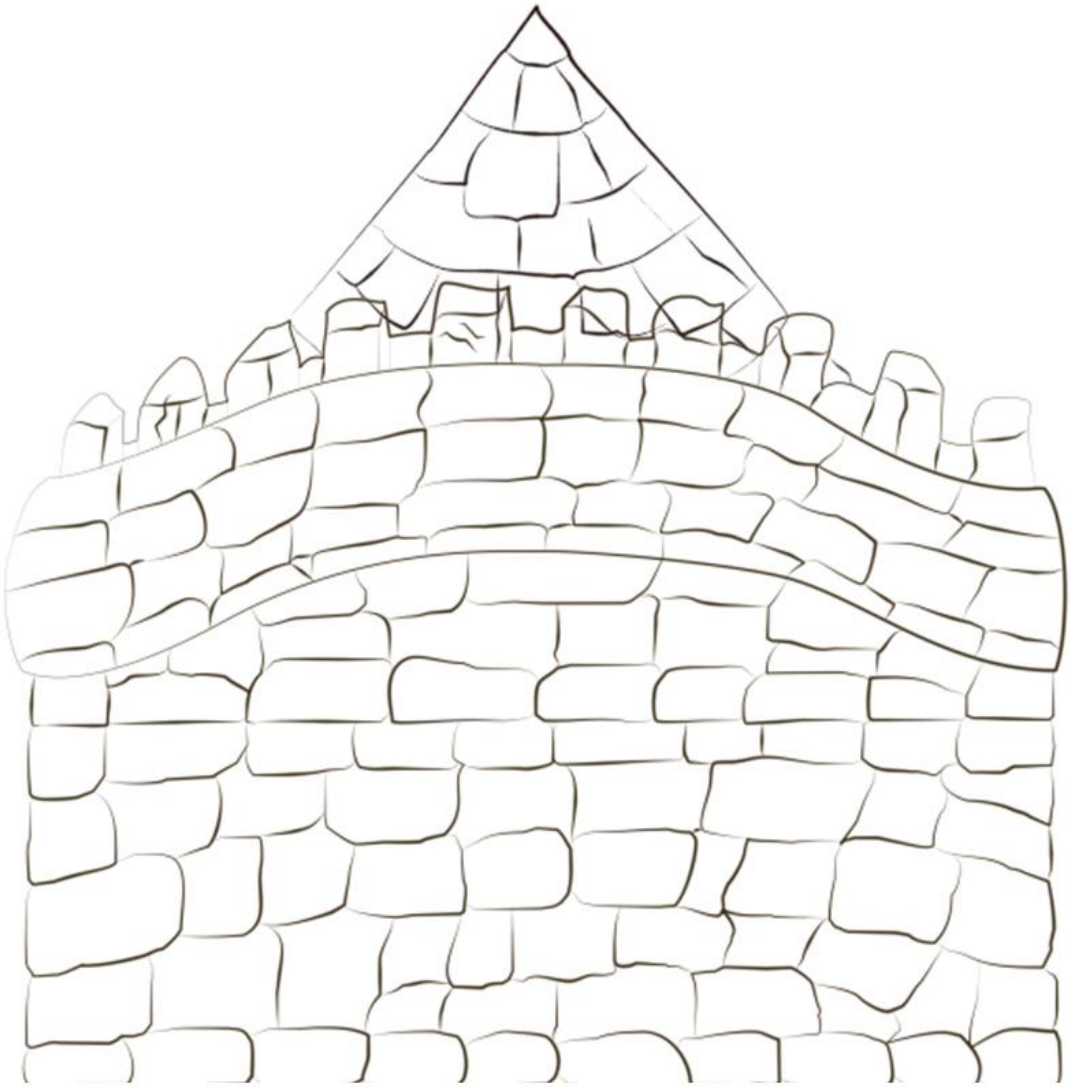

Figure 13. Stroked picture

The next step is filling. At this stage a wide variety of tools, was used. If a figure is stroked from the inside, the Paint Bucket tool in Photoshop gives the following result (even with maximum 255 tolerance):

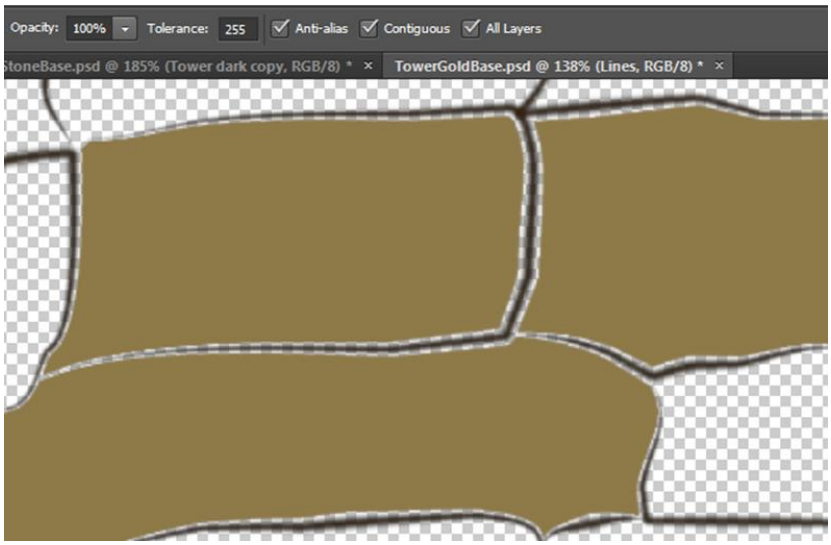

Figure 14. Painted tower (incorrect)

Therefore the entire area of the shape was selected and filled into a new layer:

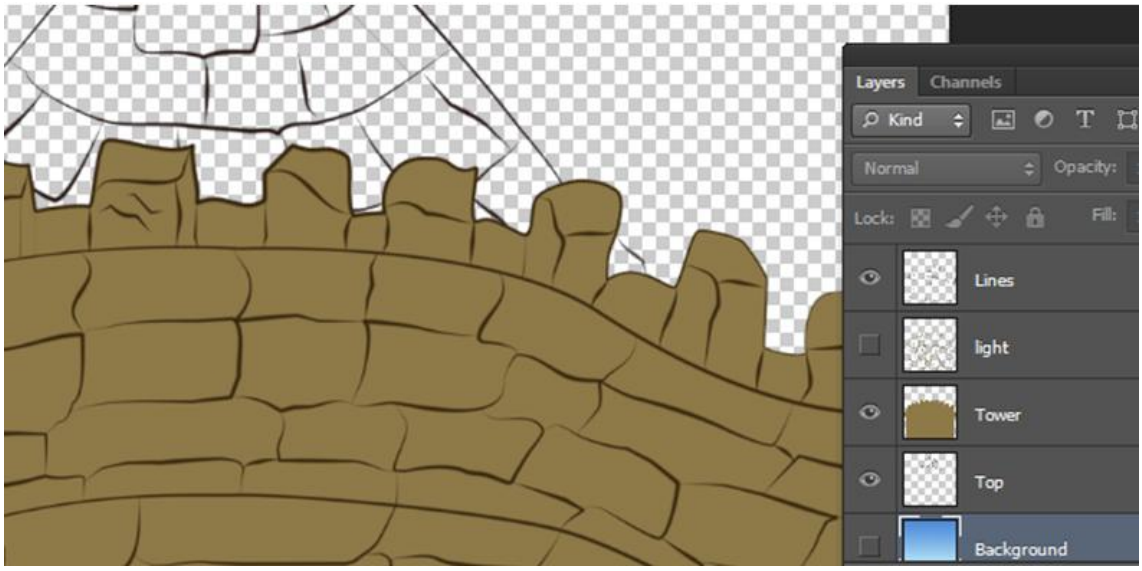

Figure 15. Painted tower (correct)

After that lights and shadows were added, see figure 16.

Figure 16. Finalized result

According to the game design the tower should have a certain level of hit points. It is logical to assume that the tower with fewer hit points should look more ruined. On the other hand the size of the game is limited and floor images cannot take a lot of memory. To solve this problem, the following method was invented.

4.3.4 Post-processing of Images or Destructible Tower

The levels of destruction and rattles were denoted using transparency in one image.

At first, the part of the image was selected. The selected part became the new layer. Fortunately, Photoshop provides a great tool for this task:

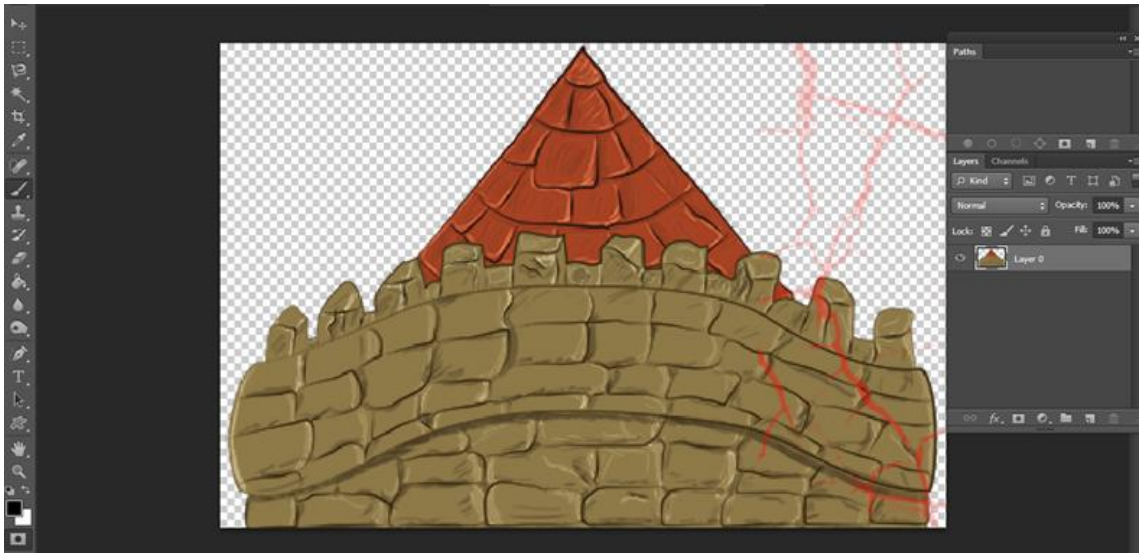

Figure 17. Mask selection in Photoshop

Then the opacity of the selected area was changed:

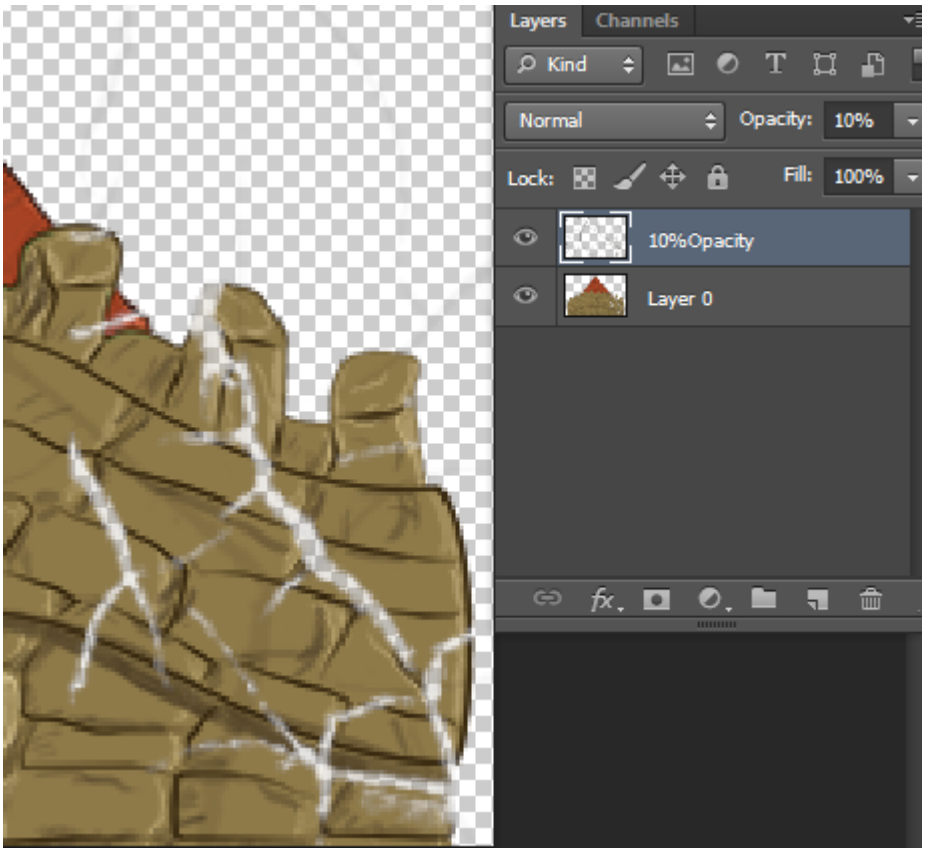

Figure 18. Changing layer opacity

In the end, all the layers were combined in one and the image was saved as PNG:

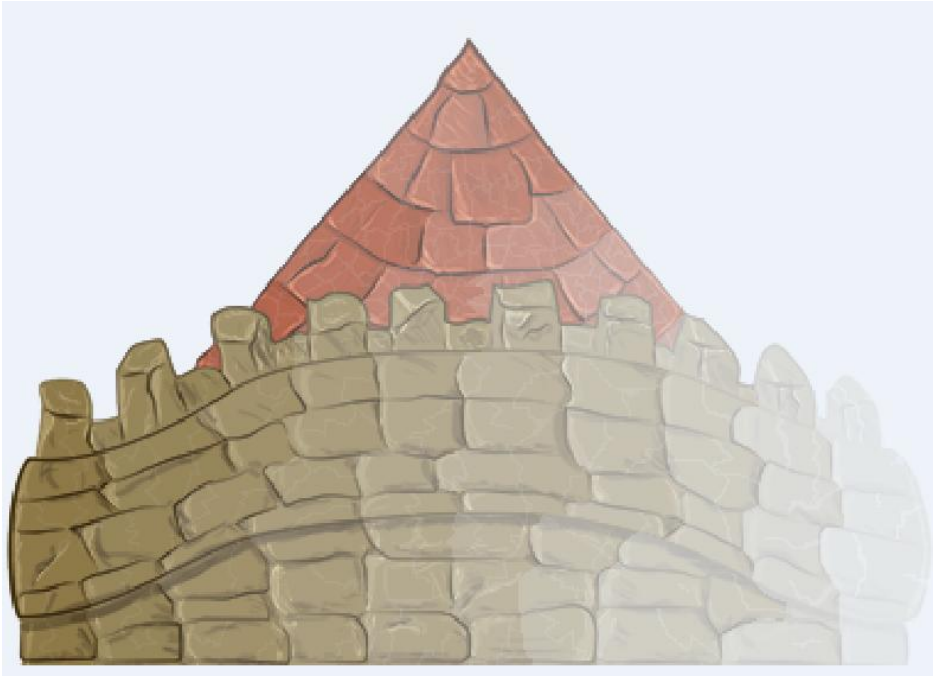

Figure 19. Result image

For these purposes, a small shader "LitCutoutVertexColor" was written. This generally repeats the standard Transparent/Cutout Unity shader, but discards the information about light. As a result, twenty different stages of destruction represented only by one sprite:

Figure 20. Result in the game

The disadvantage of this approach is that the tower can be destroyed only from one side. But this is not contrary to the game design.

4.3.5 Result Screenshot

Figure 21 describes the results of the second iteration.

Figure 21. Iteration two game results

During the second iteration new cards were added, also the basic elements of the user interface were implemented: buttons allowing restarting the fight and quitting the game, and the current score.

4.4 Iteration 3. Design Simplification

In this section, the details of the user interface, in particular the card is explained. Also certain simplifications carried out in the design of the prototype are introduced.

4.4.1 Changes in Game Design

At this stage, the iterative approach had shown the benefits. Since the creation of the second version of the prototype has exposed some of the weaknesses of the original concept. Whereas before this moment most the important idea of the study was the simplicity [12] of the project, but at this stages it is time to think about the player.

4.4.2 Resources

The main goal of the changes in the design is to make the game more cognitively accessible, understandable, simple, vivid and conceptual. The system of resources was not introduced yet, but it would be excessively difficult for the casual user. The player wants to get into the game and play and not think of why some cards are unavailable and what the rubies are. For at the beginning was decided to completely eliminate the concept of resources, even the card colour differentiation.

In addition, it will simplify the project, reduce the work on balance, make it easier to work on GUI and AI and greatly simplifies the creation of tutorials. And also as a result the following simplification is obtained:

4.4.3 Card Layout

The card layout was changed; all texts and explanations were deleted. The card effect should be obvious only from the picture:

Figure 22. Old card layout

On the previous version a lot of space allocated for the description, it can be uncom-

Figure 23. New card layout

In addition, it greatly reduces the work for localization.

4.4.4 New Character

Figure 23 shows the new character Piggish archer.

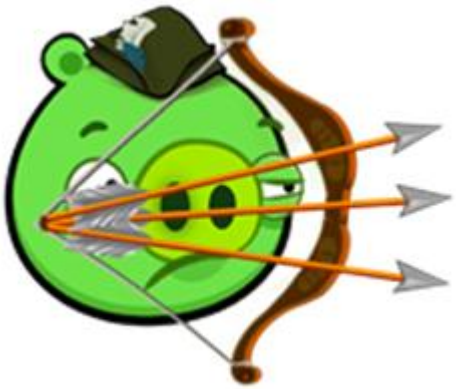

Figure 24. Piggish archer

A new character is needed as a twin for the blue birds. Since pigs are big enough, the expanding trio looks unrealistic. An archer fits very well into the medieval style of the game

4.4.5 Result Screenshot

Figure 25 demonstrates the results of the third iteration.

Figure 25. Iteration three game results

Cards' layout are changed, basic level structure are shaped.

4.5 Iteration 4. AI

In this section described process of creation basic computer response, also during forth iteration was improved game control and simple level system was created.

4.5.1 AI Definition

In this iteration was implemented a simple artificial intelligence script. Basically, this script chooses a random enemy tower's floor and calculates factors to launch the object in this floor. It is also necessary to introduce a miss factor, the probability that the opponent will miss.

At this stage, AI will ignore any terrain features. Depending on the height of the walls, AI will try to aim into unprotected floors. Depending on the game situation AI must choose the appropriate card. If a player is close to win (tower too high) or nearly de-

stroyed, the AI must try to attack. Otherwise AI must to build, of course if AI have needed card.

4.5.2 Graphics

First of all an ammo image for Pigzard was repainted:

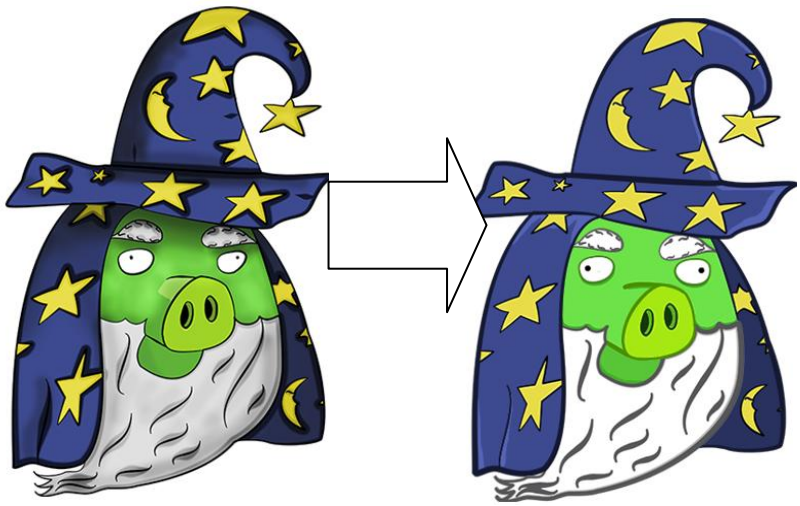

Figure 26. New painting

Now it looks more in accordance with the Angry bird style and has far better scaling.

4.5.3 Level Selection Design

Figure 27 describes a scheme of level selection.

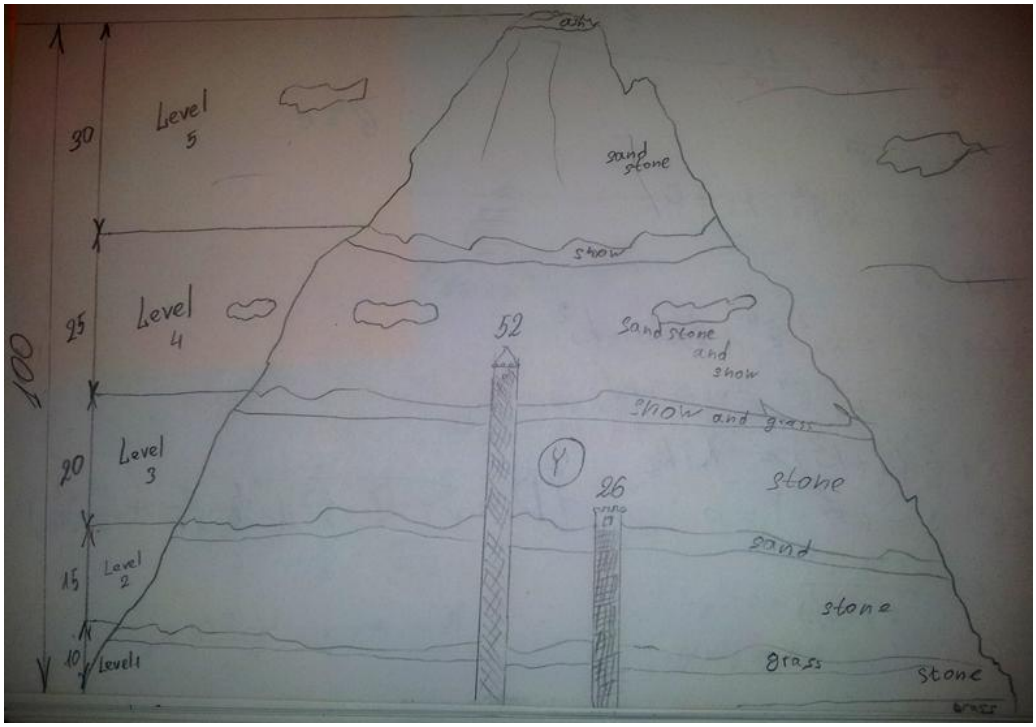

Figure 27. Level selection plan

Mountain is divided into horizontal layers; each layer defines one level of the game.

At first was created a mountain and filled with grunge texture. The result is shown in Figure 28:

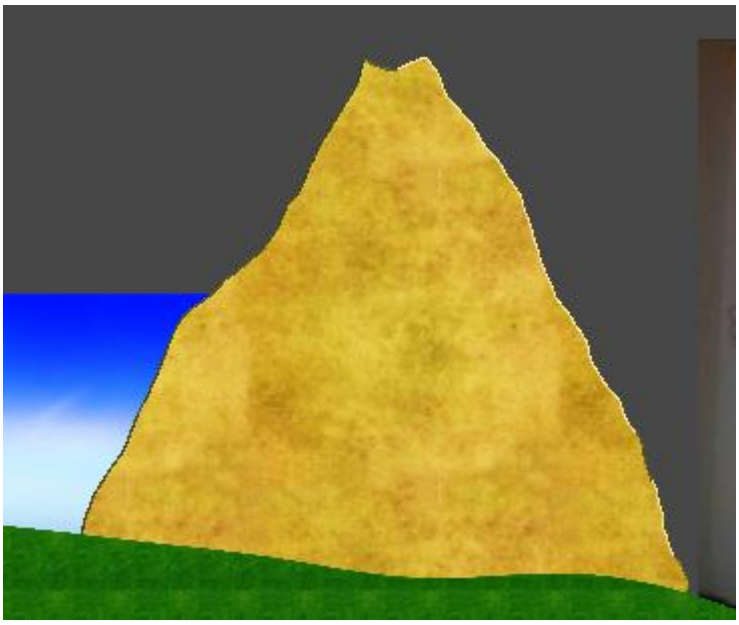

Figure 28. First attempt with grunge texture

Since this creation did not fit the style, it was decided to paint the mountain by hand. At the end, this is a very important part of the game.

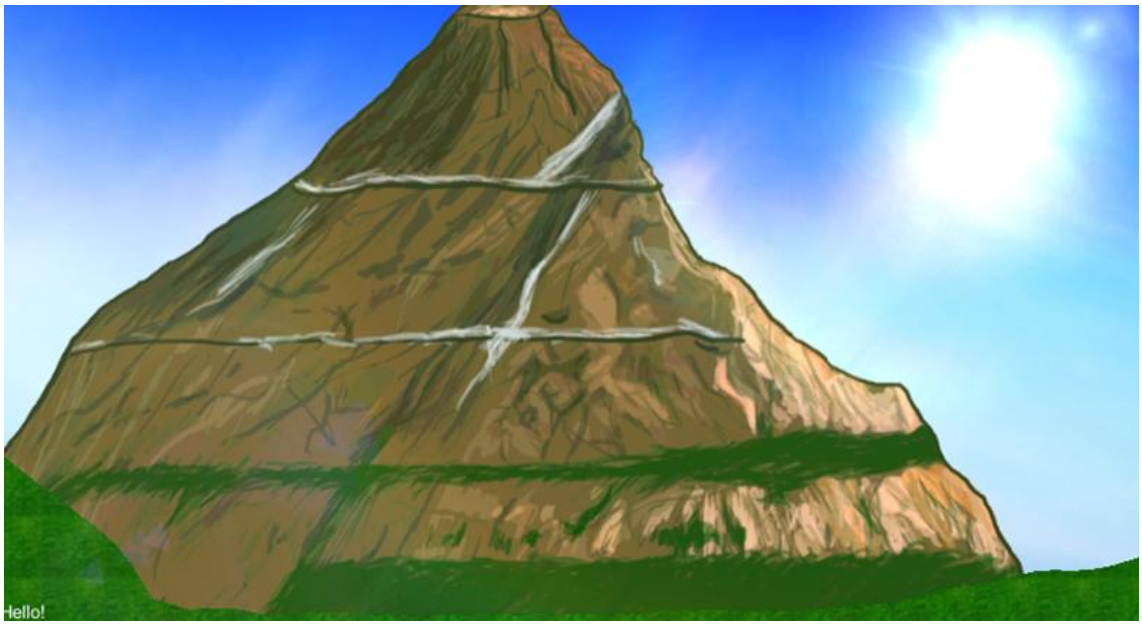

Figure 29. Level selection scene

The remaining episodes should be developed by the same scheme. Design also provides episode selection menu with small replicas of the mountains.

4.5.4 Particles

Particle effects are an integral component of a polished video game. They add that extra flash, fizzle, and pop that makes a good game a great game. When used correctly, particle effects can add that subtle bit of realism that draws the player into the game allowing them to feel, see, and maybe even smell your gameplay. [13]

In Unity the terms Particle System and Particle Effect are used interchangeably. However, these two terms have different meanings.

- Particle System: This is a single Particle System component that is attached to a Game Object.

- Particle Effect: This refers to a hierarchical composition of Game Objects each with their own Particle System component. In other words, a combination of Particle Systems which together compose a Particle Effect.

The component that provides the Shuriken Particle System functionality in Unity is the Particle System component. It is possible to add a particle system to scene in multiple ways. [14]

Particle is very important in this game. There are a lot of special effects used in games. Authentic-looking water, mixed 3d animation, images post-processing, lights and shadows make three-dimensional games super attractive. The choice of special effects for two-dimensional games is very limited; therefore a thorough approach to the creation of particles is necessary.

The first particle system was the smoke of the volcano. In Unity the is a default particle included that is very well suited for smoke. Volcano smoke is described on Figure 29.

Figure 30. Volcano smoke

Another important application of particles is the process of tower destruction.

4.5.5 Result Screenshot

Figure 31 describes results of fourth iteration.

Figure 32. Iteration four game results

At this stage all graphics are finalized, background and effects are added.

4.6 Look into Future

This chapter introduces some conclusions based on the study. Originally the plan was to have five iterations. The main idea of the fifth iteration was to add a simple shop in the game. But it was realized continuing the work would have been a waste of time. And it is not a matter of laziness; according to the Pareto principle roughly 80% of the effects come from 20% of the causes [15].

First, the design of the in-game purchases is explained. Then the final version for multiplayer is described, mostly from the game designer's perspective.

4.6.1 IAP Process Design

This game should support in-app purchases and advertisement. Unfortunately, if one wishes the application to survive in the market it has to generate income. The two main

methods of monetization are advertising and in-game purchases. In this section the design of in-game purchases is explained.

The following is the description of setting up Amazon inn-app purchases. To use any of the Amazon In-App Purchasing features the game needs to include the Amazon In-App Purchasing API. On Amazon developer's site there is Unity plug-in, which c consists of:

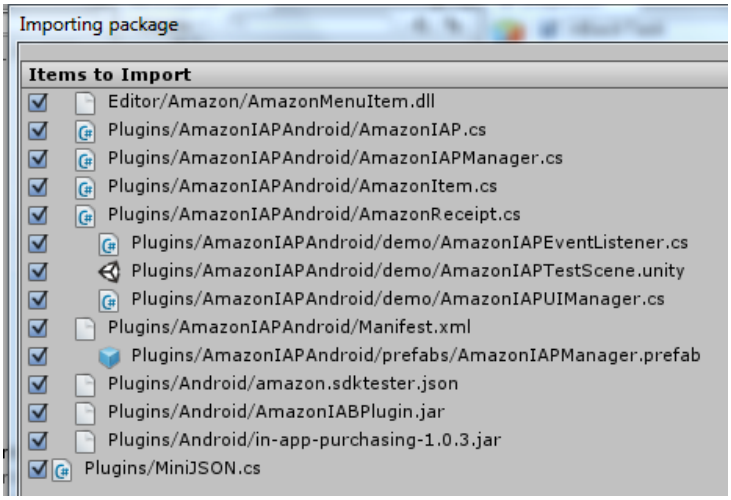

Figure 33. Unity plug-in content

Preliminary design is as follows. In the game there will be the scene "Shop" where the user can make purchases. The initial plan was to add only cards with special spells, but in the future the range of purchases can be expanded. Figure 33 illustrates how the system works:

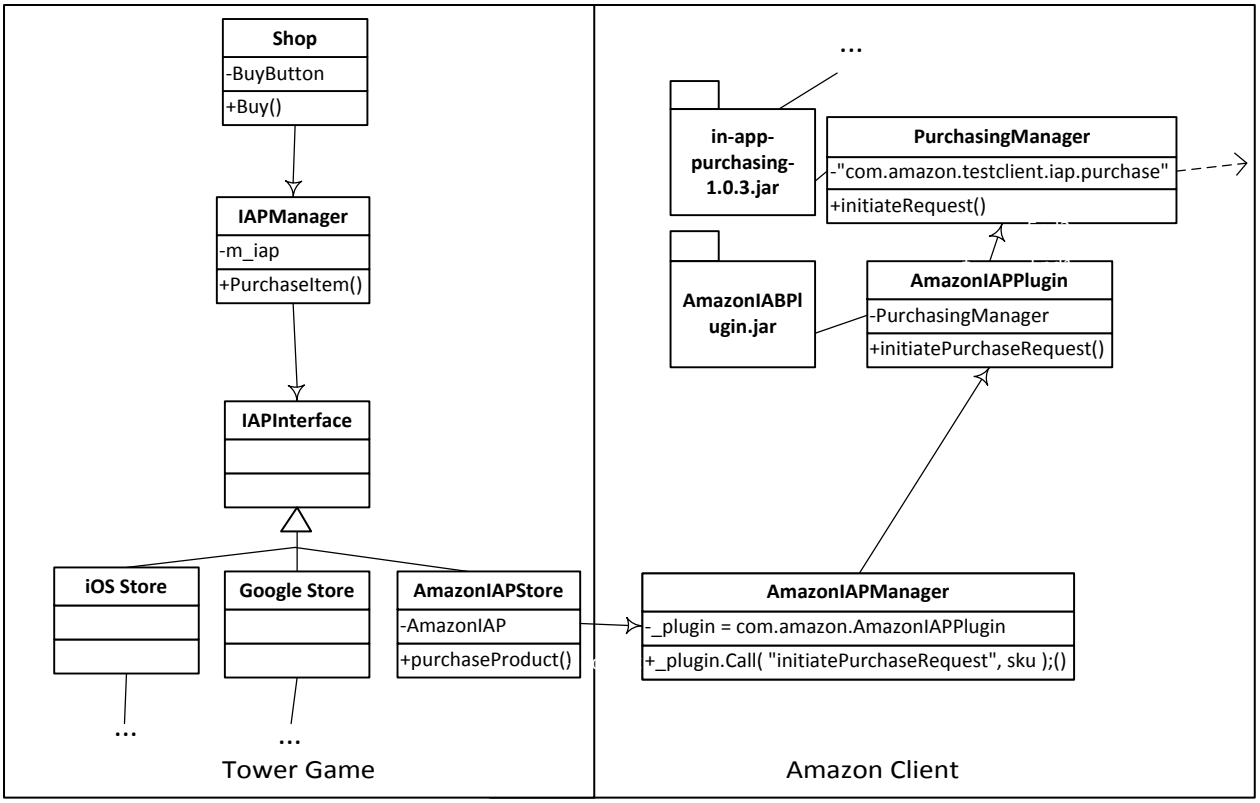

Figure 34. When user clicks Buy button

When a user initiates a purchase, the corresponding function is passed to the Amazon client.

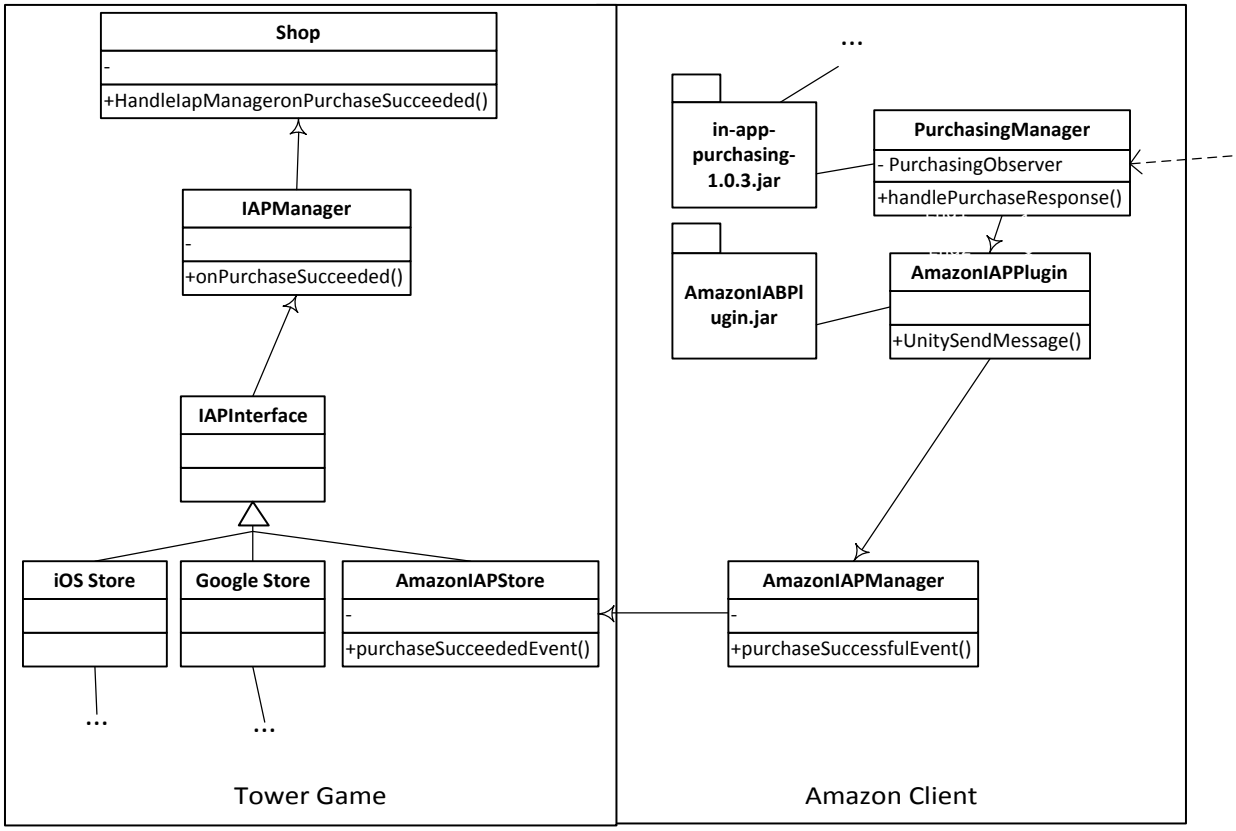

Figure 35. When API receivers response from Amazon Appstore

Such a scheme allows connecting an unlimited number of application stores.

4.6.2 Multiplayer

The game is suitable to play together on one device. This is ready in the prototype; one can select playable sides to play together. This is a nice feature, but the main strength of playing together will be achieved through the multiplayer mode.

Each online player has their own tower. To win in a certain battle, the player must increase their own tower with 15 floors or destroy 15 of the enemy's floors (or the entire tower, if it is lower). To reach the opponent’s location, the magician opens a teleport. The player can observe both locations.

Teleports allow connecting any two players.

Figure 36. Multiplayer battle

In order to diversify the fights, a new game element, a barricade, was designed. See Figure 36

Figure 36. Barricade constructed by player

Different sets of cards, the difference in players' level and unique barricade must make every battle unique.

5 Conclusions

The study consisted of developing a prototype of a game. The effectiveness of an iterative approach to application development was proves during the process. The result also proves that the actual product can be made with only one developer, indicating that in the actual production very much of the time is spent on interpersonal communication.

The outcome of the research reached its purpose. The prototype is playable and ready for future development. As in any project in game creating, motivation is an important part of the development process. When developers stop believing in the success of the project, the project will definitely fail. Individual developers are often faced with the problem of lack of motivation.

The present study also offered a fairly in-depth study of the current situation in the gaming industry. The study provided the author with substantial experience in various

fields. The first being the use of the Unity3d engine, since most of the work consisted of creating a prototype. Also experience in creating game graphics and game design was gained

As for further work, it is recommended to pay more attention to motivation. Also a strong focus on the design can save time in the development.

References

1 Bates, Bob. 2004. Game Design (2nd edition). Cengage Learning PTR.

2 Brooks , Frederick. 1995. The Mythical Man-Month. United States: Addison Wesley.

3 Sears, Andrew & Jacko, Julie. 2009. Human-Computer Interaction: Development Process. CRC Press.

4 Larman, Craig. 2003. Agile and Iterative Development: A Manager's Guide. United States: Addison Wesley.

5 Hamilton, Eric. 1992. JPEG File Interchange Format. C-Cube Microsystems.

6 Wikipedia. KISS principle. Web document. <http://en.wikipedia.org/wiki/KISS_principle>. Accessed on 14.6.2013.

7 In-App Purchase Revenue Hits Record High. Web document. <http://techcrunch.com/2013/03/28/in-app-purchase-revenue-hits-record-high-accounts-for-76-of-u-s-iphone-app-revenue-90-in-asian-markets/> Accessed on 16.1.2014.

8 Salen, Katie & Zimmerman, Eric. 2004. Rules of play. Cambridge: The MIT Press.

9 Moore, Michael & Novak, Jeannie. 2009. Game Development Essentials: Game Industry Career Guide. United States: Addison Wesley.

10 Booch, Grady. 1998. The Unified Modeling Language User Guide. United States: Addison Wesley.

11 Mocný, Ondrej. 2011. Tools for Generating and Editing of 2D Terrain. Web document. <http://e2d.hardwire.cz/Thesis.pdf>. Accessed 29.7.2013.

12 Hunt, Andrew & Thomas, David. 2000. The Pragmatic Programmer: From Journeyman to Master. United States: Addison Wesley.

13 Wittayabundit, Jate. 2011. Unity 3 Game Development. Mumbai: Packt Publishing.

14 Oosten, Jeremiah. 2012. Shuriken Particle Effects in Unity. Web document. Web document. <http://3dgep.com/?p=4313>. Accessed 4.11.2013.

15 What is 80/20 rule? Web document. <http://www.80-20presentationrule.com/whatisrule.html> Accessed on 6.12.2013.