Product planning

Team Kroket

May 4 2016

Irene van der Blij	ivanderblij	4385691
Jochem Heijltjes	jheijltjes	1534041
Mayke Kloppenburg	mlkloppenburg	4383265
Alan van Rossum	alanvanrossum	4293932
Harvey van Veltom	hvanveltom	4350073

Contents

1 Introduction **2**

2 Product **3**
- 2.1 High-level product backlog 3
- 2.2 Roadmap 5

3 Product backlog **6**
- 3.1 User stories of features 6
- 3.2 User stories of know-how acquisition 7
- 3.3 Initial release plan 8

4 Definition of Done **9**
- 4.1 Feature 9
- 4.2 Sprint 9
- 4.3 Weekly release 9

A Glossarium **11**

1 Introduction

In recent years, escape rooms have been getting very popular. In an escape room a team is locked up in a room. The goal is to escape the room. The team can escape by cooperating and solving puzzles. But what if you want to take it further?

This game is played by three players; one player has an Oculus Rift, and the other two play with smartphones. The three players are a team of agents of the Central Intelligence Agency. They are investigating the disappearance of their fellow agent. The two smartphone players are in the CIA headquarters. The Oculus player is in the woods following a lead, when he is knocked unconscious. When the Oculus player wakes up, he is locked up in a dark, scary room. Deadly gas slowly starts filling the room. The gas will fill the room completely within a certain amount of minutes; a timer starts. Luckily, the person locked in the room still has his earpiece and is able to communicate with the two players in the headquarters. The Oculus player has to search the room for clues and puzzles; he can look and move around in the room and interact with objects. The two agents in the headquarters will support and help solve the puzzles. When a puzzle is solved, information (e.g. a key, a secret room, or a new puzzle) is released which will help the Oculus player escape the room.

The experience this game will give is unlike a real escape room. To make things difficult for the Oculus user, and to give a real immersed experience, anything can happen. From loss of gravity to hallucinations, everything is possible in the virtual world...

The structure of this document is as follows. Chapter 2 contains the high-level product backlog and the roadmap. In the high-level product backlog the features are prioritised with the MoSCoW method. The roadmap provides an illustration of the main tasks per week. Chapter 3 discusses the product backlog. This chapter contains user stories of features and know-how acquisition and also the initial release plan. Chapter 4 is the definition of done; it is discussed when the product is considered finished. Appendix A is the glossary.

2 Product

In this section, the product will further be elaborated by discussing the high-level product backlog and the roadmap. The high-level product backlog will be represented as a set of epics aligned with the product vision. The roadmap will contain the major release schedule and the release goals.

2.1 High-level product backlog

In this subsection the possible features of our product are separated into four categories according to the MoSCoW method.

The categories are defined as follows:

- **Must haves**
  These are the essential features of the game, that guarantee that it works correctly. Without these, the game is unplayable.
- **Should haves**
  These are the features that make this game worth playing. The game is playable without these, but it will not be too enjoyable.
- **Could haves**
  These are extra features that will only be implemented if there is enough time.
- **Won't haves**
  These are features that will not be implemented in our product, but will be taken into consideration for an extending project in the future.

Must haves

- The players must be able to connect to the game with their devices.
- After the players are connected, the game must start by telling the players the storyline of the game and the timer must start.
- The Oculus Rift user must be able to look around the virtual world by turning his head while wearing the Oculus Rift.
- The Oculus Rift User must be able to interact with the virtual world through moving the joystick and pressing buttons of a wireless controller.
- Non Oculus Rift players must be able to perform actions on their smartphones that directly or indirectly affect the Virtual Reality.
- There has to be a link between the actions of the Rift User and the information represented on the smartphones.
- Through solving puzzles the players must be able to 'beat' the game and emerge victorious.

Should haves

- When starting up the game, a screen should appear that shows a connect and start button.
- When the connect button is clicked, the player should be connected to the server so he can join the game. (The start button is not clickable yet).
- When a player is connected the screen shows which and how many players have connected to the game.
- When enough players are connected, the screen should give the option to start the game; the start button becomes clickable.
- A scoring system that takes into account the amount of puzzles solved correctly, time taken and bonuses found.
- Hints must be available in case players get stuck.
- Narrative or puzzles that immensely affect the Oculus Rift user's perception of reality, such as blurry vision or inverted gravity.
- A soundtrack that plays during the game for both the Rift User and the smartphone users.
- Players should be able to see the remaining time.

Could haves

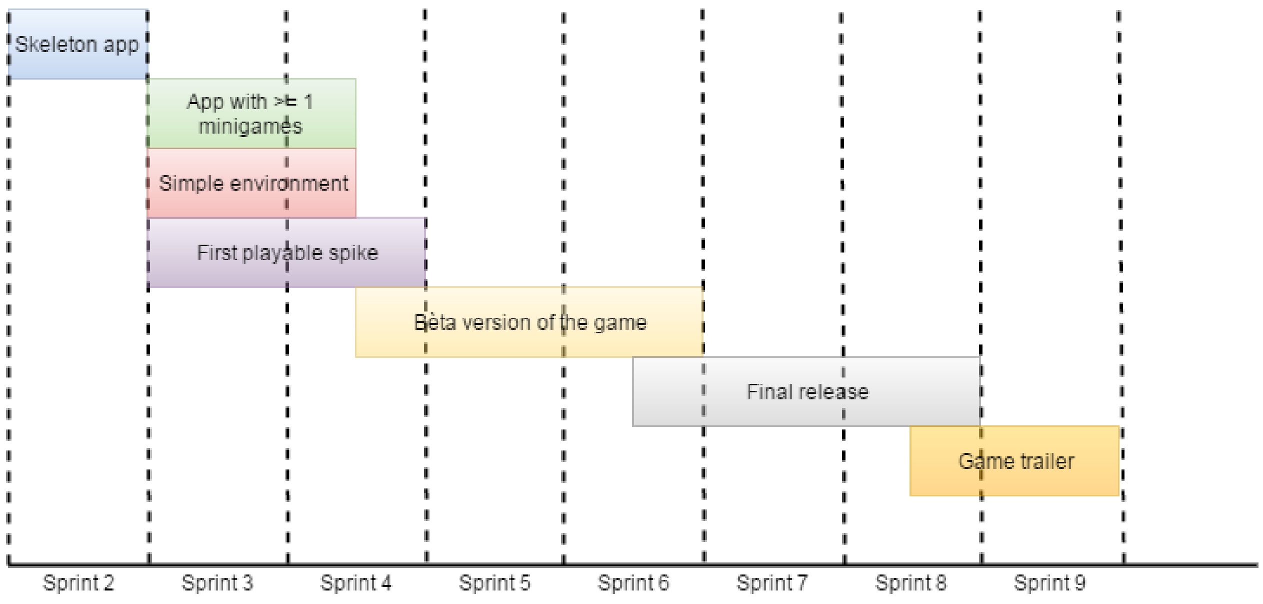
- Specific sound effects that are played when certain actions occur in the virtual reality.
- A scoreboard that tracks the highscores of multiple teams, across game sessions.
- Additional settings, like a crashing airplane or a defect submarine.
- Compatibility for just two players, so one smartphone and one Oculus Rift.

Won't haves

- Compatibility for multiple Oculus Rift devices
- Compatibility for three or more Smartphone users
- Support for microphones to allow players to communicate online rather than locally
- Support for iOS and Windows phone
- Multiple storylines

2.2 Roadmap

The following image shows all the different releases and when they should be finished. The exact content of all releases can be found in chapter 3.3, the initial release plan.



3 Product backlog

In this sections the user stories of features and know-how acquisition will be described. Lastly, the initial release plan will be discussed.

3.1 User stories of features

As an Oculus Rift player
I want to be able to look around
So I can see all of the room

As an Oculus Rift player
I want to be able to interact with the object in the room
So I can find hidden clues and puzzles

As an Oculus Rift player
I want to be able to move around
So I can explore the room

As a smartphone player
I want to be able connect my phone to the server
So I can join the game

As a smartphone player
I want to be able to see the puzzles
So I can solve them and help the Oculus Rift player

As a smartphone player
I want to be able to interact with my phone
So I can solve the puzzles

As a player
I want to be able to solve all the puzzles
So I can win the game

As a Oculus Rift player
I want to able to see the time left
So I can adjust my playstyle accordingly

As an Oculus Rift player
After receiving a hint from the smartphone player
I want to be able to use the information to progress in the game

3.2 User stories of know-how acquisition

As a programmer
I want to be able to make an Android app
So I can create the app for the game

As a programmer
I want to able to establish a connection between several different platforms
So the smartphones and Oculus Rift players can communicate

As a programmer
I want to familiarise myself with the Oculus Rift development kit
So I can use the Oculus Rift in the game

As a programmer
I want to be familiar with the Java programming language
So I can create a game written in Java

As a programmer
I want to be able to work with the JMonkey game engine
So I can use it as a platform to create the game

3.3 Initial release plan

- **Skeleton app**
  The skeleton app is the most basic app. It will only have a small menu with 'start' and 'connect' buttons.

- **App with one or more minigames**
  When pressing start in the app, the smartphone player can play one or more minigames. The minigames do not have to be triggered by the Oculus Rift player yet.

- **Simple environment**
  As an environment for the Oculus Rift player, a simple room is built. This room is not decorated. Since ultimately in this game the Oculus Rift player has to interact with objects to unlock minigames, the room will contain one or more coloured cubes.

- **First playable spike**
  In this release the most important thing is that the smartphone players and the Oculus Rift player are connected. When the Oculus Rift player interacts with a cube in the room, it should trigger a minigame in the smartphone app. When this minigame is finished, it should be possible for the Oculus Rift player to interact with another block.

- **Beta version of the game**
  In this release there will be more minigames present, ideally all of them. Ideally the cubes are replaced by appropriate objects. There should be a timer present. Also there should be some interaction from the room with the Oculus Rift player, e.g. loss of gravity. If there is time left, the room will be somewhat decorated.

- **Final release**
  This release should contain all minigames. The room should be fully decorated, giving the appropriate creepy atmosphere. Also, there should be an animation of the deadly gas slowly filling the room. Bugs in the beta version are fixed.

- **Game trailer**
  This is not a real release of the game, but a trailer for the game. It should make clear what the game is and make people excited to play.

4 Definition of Done

The definition of done describes for three different aspects of the project what it means when it is considered to be done. When a certain aspect is done it can be assured the aspects will hold up to the standards described in this section. The three aspects are features, sprints and weekly releases.

4.1 Feature

The requirements for when a feature implemented in the game is labeled "done" are as follows:

- The passage of unit tests. These unit tests are designed specifically to ensure that the new feature doesn't contain any bugs or errors that will cause errors in the application.
- The passage of integration tests. The integration tests are there to ensure that adding the new feature to the existing system won't cause any errors or loss of functionality. This is different from unit testing because unit tests are designed specifically to test the feature itself and integration testing are there to ensure that addition of the feature to the system won't cause any errors.
- Javadoc documentation. This documentation is so that the developers can quickly determine what every part of the new feature does.
- The assurance of code quality. To ensure the code quality the feature will be tested using multiple static analysis tools. these analysis tools are Checkstyle, PMD and Findbugs.

4.2 Sprint

The definition for when a sprint is considered done are:

- An update of the system documentation. This documentation includes new documentation for any new features as well as updated documentation for where the system is updated.
- Sprint reflection. In the sprint reflection the team will reflect back on what planned tasks were and weren't accomplished as well as reflect back on any problems that occurred during the sprint.
- Sprint backlog. The knowledge gained during the sprint reflection will be used to make the sprint backlog for the next sprint. The sprint backlog will contain the tasks that weren't completed during the previous sprint, new tasks that solve problems encountered in the the previous sprint and new tasks that will ensure new features that the game needs to have.

4.3 Weekly release

Finally, the definition for when a weekly release is considered done:

- The application passes a certain percentage of tests. these tests include both JUnit and integration tests. Due to certain limitations of testing 100% test coverage isn't achievable. The goal is that every weekly release is tested for at least 75%. test coverage will be measured via the line coverage method of the program Cobertura.

- The game will need to be able to run. After every week the stakeholders need to be able to see the game via a demo so that they can give their feedback.
- A certain level of code quality assurance. Errors in the code quality will be detected using the static analyses tools PMD, Checkstyle and Findbugs. Any errors found will either be corrected or given an explanation for as to why removing said error is unnecessary.
- The implementation of certain features. These features include all the features described in section 2.1: Must haves as well as a certain percentage of the features described in the section 2.1: Should haves.

A Glossarium

Acronyms

CIA Central Intelligence Agency. 2

List of terms

Android Operating system for Mobile Phones and tablets. Makes use of the Linux Kernel and the Java programming platform. 7

Checkstyle static code analysis tool that checks whether Java source code complies with the Java coding rules. 9, 10

Findbugs Static code analysis tool for Java bytecode. 9, 10

iOS Apple mobile operating system. 4

Javadoc A documentation generator created by Sun Microsystems for the Java language for generating API documentation. 9

Oculus Rift A pair of Virtual Reality goggles with a stereoscopic field of view, designed for gamers. The device measures the movements of its wearer. 2–4, 6–8

PMD Static code analysis tool for programming mistakes. 9, 10

Smartphone Phone with many features running an advanced operating system, similar to a desktop computer. 2, 4, 6, 8

Virtual Reality An artificial world, generated by a computer. 3