Final Report

Computer Games Contextproject 2015-2016
Course TI2806, Delft University of Technology

Group PixelPerfect

June 16, 2016

Supervisor
Dr. Ir. Rafael Bidarra

Teaching Assistents
Sander van den Oever
Jurgen van Schagen

Team

David Alderliesten	4368703	*J.W.D.Alderliesten@student.tudelft.nl*
Floris Doolaard	4362748	*F.P.Doolaard@student.tudelft.nl*
Dmitry Malarev	4345274	*D.R.Malarev@student.tudelft.nl*
Jesse Tilro	4368142	*J.Tilro@student.tudelft.nl*
Wouter Zirkzee	4398858	*W.J.Zirkzee@student.tudelft.nl*

Contents

1 Introduction

PixelPerfect is a game company founded by David Alderliesten, Dmitry Malarev, Floris Doolaard, Jesse Tilro, and Wouter Zirkzee for the ContextProject course at the Computer Science degree program of Delft University of Technology. The company was requested to create a virtual reality game that could be played in a party situation, allowing for one player to wear an Oculus Rift virtual reality headset, and up to four other people playing on their phones. The situation that the game would have to be played in was one in which all players were able to verbally communicate without issues. No further requirements were given.

The product developed was a video game called "Space Crew." In Space Crew, the team of five players must work together to successfully bring their spaceship to different locations in order to survive for as long as possible. The player wearing the virtual reality headset is able to see the cockpit and all of the spaceship's tracking systems. Emergencies can occurs, such as a fire outbreak, an encounter of a hostile ship, or leakage in the essential plasma system. The captain is able to communicate the problems that arise, along with important information such as locations and types of problems, to the other players, who have been split into roles that can each fix certain issues that could arise on the ship. These players must also communicate with the captain to verify that issues have been solved. Players must also ensure that multiple people are not trying to complete an identical issues, as this is not efficient and could cause fixed to be performed when they are not needed.

Players earn points for completing events, and lose health for failing them. The game continues until the health either falls under the accepted parameters, or when the team decide to stop playing. The aim of other teams is to be each others' scores. The events that can occur are generated randomly, and multiple events can occur at an identical moment, meaning that players will never have an identical experience when playing the game twice. Role switching between games is also encouraged, including the altering of the player that fulfills the role of captain in the Oculus Rift virtual reality headset.

Within this report, you will find a simplified development outline and product overview. The report contains a full product overview, a reflection on the development process and final product, an overview of the developed functionalities, the results of PixelPerfect's interaction design study, an evaluation of the product, and the outlook for the product on the market and possible future development.

2 Product Overview

The Product Overview contains a simple overview of the components of the final product, including the Oculus Rift virtual reality headset client, and the associated mobile phone client. The interdependency of these clients is also explained.

2.1 Oculus Rift Client

The Oculus Rift client is meant to run on a capable desktop computer, acting as both the center connecting server for all mobile phone clients and as the host for the Oculus Rift device. These elements are handled by the JavaMonkeyEngine 3.0[LINK NAAR JME], a Java-based engine meant for the creation of three dimensional applications. The virtual reality integration is handled by LibreVR[LINK NAAR LIB]. The client runs a server to allow other devices to connect and runs the center game engine, which is responsible for the game logic and handling of the graphical display of the Oculus Rift.

The game logic consists of every piece of game code that has to do with the game design elements. This includes the random creation of events, the maintaining of currently active events, the handling of messages received and their validation, and other elements essential to the running of the game.

The graphical display consists of the display of a spaceship cockpit, which can be looked through by the Oculus Rift player by moving their head. This cockpit contains contextual clues for the events that are active in the spaceship. The graphical display also consists of two indicators that consist of a heads-up display which show the current team score and spaceship health. The heads-up display also consists of ”debug mode” which can be activated which shows game logic information that is mainly meant for debugging problems during gameplay.

2.2 Mobile Client

The mobile client was developed entirely by PixelPerfect and is aimed at the Android operating system [ANDROID LINK], which as the largest market share of mobile phones right now [SOURCE]. The client allows a person to connect their mobile phone to an Oculus Rift client that is running Space Crew, and then allows them to choose a role and play the game. Each event can be fixed by a specific set of roles (all roles can douse a fire, but only the scientist may fix a plasma leak), and each role can choose to play a mini-game associated to their selected event. Successful completion of the event sends a message to the Oculus Rift client that the event was completed, idem for failed events. Each event has a sequence of actions and a subset of choices that must be selected and completed before such a message is sent.

The mobile client makes use of multiple system calls to the Android operating system, but was written entirely from scratch for the purpose of this project. The game logic of the mobile client can be easily ported to another mobile operating system (such as iOS or Windows phone) with minimal effort, as only the display of the mini-games would have to be altered.

3 Reflection

In this section, we will describe the reflection of the process during the assignment from a software engineer's standpoint.

The process of the assignment's implementation did not start smoothly. This was mostly due to the fact that the documentation for the engine we were using went offline, rendering it unusable and causing us to attempt functions on a trial and error method until we received a passable alternative. The alternative itself was honestly sub-par: it did not contain all the documented functions that the engine had.

Another issue we had involved the programming language used to make the product. As explained in the Product Overview, we had to use JavaMonkeyEngine, a Java-based engine. The issue was that the language is sub-optimal for this specific context, causing us to limit possible features that we could implement because the language could not support it. A recommendation for any future projects like this would be to use a language that is more compatible with this context. An example of such language would be C-sharp.

Besides those issues, the project went pretty well. The coordination between group members was good and the tasks could be easily divided, so there were no problems with group members recieving fewer tasks and spending less time in the implementation and development.

4 Developed Functionalities

For a game to be good, it should not only be intriguing to the player(s) but also evoke a feeling of replability within a player, sometimes known as the "one-more-turn" effect [CITATION]. In order to achieve that, the game needs to have high quality content and functionalities that can keep players entertained. In this section, we will describe the specific functionalities that "Space Crew" contains and why these were added. These functionalities are divided between the ones present in the Oculus Rift server application and the ones present in the mobile client.

4.1 Oculus Rift Client Functionalities

As explained in the product overview, the Oculus Rift client acts as the server of the game as well as the component responsible for the rendering of the ship for the captain. The ship is capable of generating events using a Poisson distribution, which dictates when events are spawned at which random intervals. These events contain parameters that cause variation among them, such as different locations or "fixing values" needed. When these events are generated, the player using the Oculus Rift is notified by receiving visual- and audio-based feedback on what event has spawned, as well as the parameters that it contains.

In the case that the server receives a completion message of an event from the mobile client, the server will verify that the event is present in the generated event log, which consists of a collection of all currently active events. If the parameters match, the log discards the completed event and points are assigned to the team as a result. In the case of an incorrect completion or an event running out of time, the ship takes damage and lowers the team's score. When the ship's health is equal to or less than zero, the game is over.

4.2 Mobile Client Functionalities

The mobile clients are required to connect to the server in order to play. The connection is done by writing the IPv4-address in the input field of the mobile client when the client is started up. The player has the ability to select one of four roles, each capable of resolving different issues in the ship. The game features a role lock system for these roles, which prevents two players from having the same role. This is done by communicating the roles chosen and saving their allocation within the server.

Communication between server and client is done using messages that the client sends to the server. Whenever a player completes a mini-game, the client sends a message containing the event that was completed and the parameters selected on the mobile client during the mini-game. These are then validated server-side. Players also have the ability to repair the ship with two health points by completing the appropriate mini-game. This means players will have something to do while they wait for new events to arise, meaning no player is ever left inactive.

5 Interaction Design

5.1 Method

In order improve the usability of the software system under development, some evaluation on the Human Computer Interaction Design was performed. The main purpose of the software product was to collectively entertain a group of people. Therefore the goal of the evaluation was to find out how much the game was enjoyed by players. In addition, the opportunity was seized to gather feedback on how the gameplay experience might be enhanced, in the opinion of players.

In order to perform the evaluation, the team organized an experiment with real users. Over the course of four hours, different people were invited to come play the game. The people were selected from the team members' personal connections, i.e. friends and colleagues. Most of them reported to occasionally play video games, so they modelled the target group of casual gamers quite well. Different subsequent play sessions were organized and the participants were grouped as much as possible. Given the fact the game must be played with five players and there where some constraints on the sample group, there was a lack of participants for some sessions. In this case some of the developers would fill in and play along. All participants would be briefed on what the game was about, so they got an idea of what they were up to. Then, the the participants would play the game for about ten to fifteen minutes. Finally, during the debriefing, the participants would be briefly interviewed / provided with a questionnaire.

During the experiments, the team would monitor the play session and some data would be recorded. These measurements for each session concerned:

1. The time until the first event was successfully solved. The entire game is about solving events. It takes the player wearing the head-mounted display to understand the virtual environment they're in. It then requires them to verbally communicate with the rest of the players. Finally at least one player using a mobile device must be able to react to this correspondingly. If the group succeeds in performing this task, we believe this means they understand the game well enough explain it to each other, play it in its entirety and to enjoy it. We expected this time to be at most two minutes.

2. The number of times the player with the head-mounted display had to report data on the same event on average. If this number is larger than two, it means the information might not be comprehensible enough.

3. The emotional state of the players, measured by means of a SAM evaluation form. Given the purpose of the software, we have specific requirements on the emotional effects it has on its users. First and foremost the game must be a fun endeavor, so it should yield happy, satisfied and excited users. Additionally, it should bring users into a mental state of operation called flow. For this to be achieved, users should not experience too much anxiety, worry, apathy, boredom or relaxation. Instead, they should experience a mix of arousal and control. (The means by which this is effectuated is explained in the section on game design.)

5.2 Reflection

The data of the process can be found in Appendix A. The experiment allowed participants to take turns playing as the captain and playing as a role on the mobile client. One session included a full group (meaning five players all taking turns and playing the game without any developer interference), and four sessions included partial groups (between two and four players taking turns, with developers taking over on the mobile client to fill the missing participant slots). Participants were briefed before and after the experiment, and were requested to fill-in a questionnaire after the rounds were completed in order to collect feedback. Members of the development team kept track of the relevant variables mentioned above, without information the participants that these data points were being tracked.

5.3 Conclusion

The Conclusion section contains information about the data and feedback collected during user experience testing, and contains an evaluation of the limitations and adjustments required for the experiment during future user test pages.

5.3.1 Feedback

On an experience level, players mentioned they enjoyed the idea and many found the Oculus Rift virtual reality interface to be very intuitive. The concept of the game was approved of by almost all players.

On an implementation level, our experiment clearly showed that the participants found it difficult to distinguish the buttons on the mobile client, and found it difficult to figure out what they meant. Another major issue that was found after the reflection was that participants were confused by the lack of on-screen instructions and notifications, including a lack of health and score indicators. Some players also noted events were not completing successfully despite correct input of parameters, and that it was difficult to spot events and their parameters in the virtual reality cockpit environment. A major issue encountered during playtesting was that the event validation system were not functioning, either.

Based on the feedback of the testers, numerous alterations were made. The virtual reality interface was changed to include more information about the parameters of events, and certain text elements had their font hexcode made lighter and size increased to ensure that they could be seen easily. The mini-games were also altered based on feedback, to ensure that all instructions were displayed correctly, and to ensure that they functioned correctly and that the purpose of each event was clear. Major issues with the event validation and feedback were fixed, including sounds that would inform the captain that an event was completed successfully (for example, the completion of a hostile ship event would sound a major explosion sound).

5.3.2 Limitations and adjustments

The limitations of the experiment included the involvement of developers, which meant certain components were testing with pre-existing knowledge, possibly skewing the results of those sessions. Another limitation was that the build of the game had a major bug in it that caused most events to be uncompleteable, skewing the results to a negative bias. The parameters that were checked and measured were not sufficient for the final data checking, since parameters were dependent on the occurrence of events, and since this was randomized through a Poisson process, it was difficult to get a fair sample comparison.

Potential adjustments for future experimentation includes ensuring that all sessions are fully staffed with volunteers to fill all slots of the game, and to ensure that the game client is tested and stable before doing the full playtesting sessions. Additional parameters should also be checked, such as which events can be completed with specific parameters, and which events were far more difficult with certain parameters.

6 Evaluation

The final version of the product is an enjoyable game that is based on communicative tasks by utilizing human-technology interaction to provide the experience. The players can use a virtual reality headset to see around the cockpit or they can connect to the server with their Android mobile device through the provided application. The captain is able to see multiple sources of information in his cockpit in order to allow them to correctly communicate the issues of the spaceship to the crew, including health points, score and active events. These events are scheduled at random intervals through the use of a Poisson process. The ships health points and score are manipulated by completing or failing the events that have arisen. The crew players can complete the events, causing them to not lose health points and gain points, by performing the right task given by his captain. To prevent idle time players can repair the ship at moments that they do not have an active event to attend to, with two health points being added per correctly completed idle events.

While the game has all of the aforementioned features, it was not possible to implement everything that had been initially planned in the product vision and game design documents [CITATION]. One of these features was for the captain to pick a route to a planet, which would alter the events along that route. The game is also not won when a planet is reached, which was the original plan. A deviation occurred from our original plan to create levels which had to be completed and chose to as to emulate a survival style game. To accomplish this the team is allowed to play for as long as they can stay alive and try to get the highest score possible.

7 Outlook

In the past weeks a lot has been learned and our goals to improve productivity and work strategies were attained. There are certain issues that would have to be improved during future work on a project as a team.

One of these issues regarded allocation for each sprint. A few times there were problems with regressive implementations, meaning that certain group members were waiting on another group member due to dependencies of their work on someone else his work. This means that there was an incorrect allocation of tasks, since the tasks were too closely related to each other. If possible, closely related tasks should be allocated to the same developer. This developer can then work on their tasks with has little regressive development and unforeseen issues as possible.

Another important issue was the weekly planning. Especially for tasks that were quite large in man-hours we lacked some willpower to begin early with them. For example, a late start occurred thinking about experiments for the project for the user testing. This can be solved by making a better planning of tasks in the future. This way the tasks can be partitioned into smaller sub-tasks which are easier to solve. This also improves the finishing time of tasks in general, meaning less stressing occurs on the deadline..

Overall the project was successful and the only major problems which occurred were resolvable. Only the documentation of jMonkeyEngine provided us with some problems with doing research. The jMonkeyEngine development team eventually fixed their problems and this allowed the team to complete the project with great success.

8 Conclusion

Space Crew is a space-based virtual reality game developed by the PixelPerfect team for the Contextproject course at Delft University of Technology. The game features two separate client, namely an Oculus Rift server client and an Android based mobile client. The server client displays the virtual cockpit and acts as a central server for the mobile clients. The four mobile clients allow events to be completed and connect to the server to allow for interaction between the two devices. The game features randomly generated events that act as looming disaster, all randomly generated by a Poisson process. The events all feature unique clues and visual/sound effects in the virtual reality client, and each event has a unique mini-game in the mobile client. Experimentation done during user-testing revealed a major technical issue that was fixed on time, and many design related feedback that resulted in many changes to the product design and usability. Although the game is seen as quite enjoyable based on user feedback, it is ready to be enhanced upon and extended, including features that were not able to be developed due to time and technical restrictions.