15/06/2015

Final Report

Games Context Group 1

Danilo Dumeljić	ddumeljic	4282442
Stephan Dumasy	sdumasy	4286723
Dennis van Peer	dvanpeer	4321138
Olivier Dikken	odikken	4223209
Jonathan Raes	jmraes	4300343

Abstract

This deliverable is the main document about the developed, implemented, and validated software product.

It will present the main functionalities of the product and discuss to which extent they satisfy the needs of the user. For this purpose, an evaluation of the functionalities performed using a well-justified method needs to be presented, as well as a failure analysis – where the product does not perform as needed.

Delft University of Technology

Table of Contents

This deliverable is the main document about the developed, implemented, and validated software product.

It will present the main functionalities of the product and discuss to which extent they satisfy the needs of the user. For this purpose, an evaluation of the functionalities performed using a well-justified method needs to be presented, as well as a failure analysis – where the product does not perform as needed.

Furthermore, the Final Report will also contain a section describing the HCI module that was realized for the user interaction with the developed solution. This section will reveal what the students learned in the Interaction Design course and will be evaluated by the corresponding lecturer. The grade for Interaction Design will be assigned based on the content of this section (see Section 8 for assessment process and criteria).

Finally, an outlook will be given regarding the possible improvements in the future and the strategy to achieve these improvements.

Note that this report should not repeat the material from Product Vision, but should complement it by providing results as response to expectations and strategy described in the Product Vision document.

Introduction

Introduction, including a brief problem description and end-user's requirements.

Queues are a fundamental component of reaching any service. New customers enter at the end of the queue and progressively make their way to front as preceding customers are served in order.
But what are all these customers doing in that queue if they are not being served? Nothing. Standing in line often is a boring and lonely experience. While as participant you are surrounded by people! Our project aims to take care all this boredom and social awkwardness in queues.
How? by having the participants of these queues play a game! A game that requires people in a row to actively engage and socially interact and communicate with each other in order to be successful.
Because the players of the game are simultaneously participants in a queue, our game must respect the dynamics of queues. People must still be able to be served at any moment and new people must be able to enter the queue at any time.

Overview

Overview of the developed and implemented software product.

We created a 2D oriented game where up to six players use physical movements to control an in-game wave. The players' movements will be captured by a camera and an camera detection algorithm. The camera looks orthogonally at the row of people that turn to face the camera/screen.
The camera detection algorithm detects the people standing before the camera and uses their height outline to generate a wave on the screen.

Penguins.
Penguins spawn on the left side of the wave and move to the right slowly if they touch the wave. The players have to physically cooperate to move the wave in such a way that the penguins on the wave catch the (tuna) fish that randomly spawn somewhere on the wave.

Obstacles.
While the players are trying their best to reach the fish, enemies spawn in the scene and do a characteristic action to attack the penguins and reset the progress made.
These are the obstacles that can be encountered in the game:
Killer whale: The killer whale jumps up from below the screen in an attempt to kill to destroy the penguins. First a warning block will be displayed for a moment on the location where the whale will attack to warn the players that there's a whale coming for them.

Eskimo: The eskimo will throw a spear at the penguins horizontally from the left side of the screen. A warning line will display where the spear will be flying.

Killer Whale: The killer whale will jump up from the sea below to attack the poor penguins. A red square is shown for a short time before the whale attacks to warn the players of the incoming danger.

Polar Bear: The polar bear will arrive from either the left or right side. While moving the polar bear is harmless but once it has picked its spot is will kill any penguin that crosses its path. No warning will be displayed when the polar bear arrives, the players can only guess where the bear is going to stop.

Lightning Strike: Occasionally a lightning strike will strike down on the unsuspecting penguins. The lightning, as lightning tends to do, will strike on the highest point of the wave. This obstacle will show a red warning line to show where the lightning is going to strike.

Yeti: The yeti (though it won't show itself) will throw a snowball from far behind the wave. This snowball travels towards the wave and will kill any penguins that it hits as it flies over the wave. No warning will be displayed where exactly this will happen, the players can only see the ball coming at them from behind the wave.

Grey Ball: The grey ball will drop down from above and crush any penguins it touches. The players will have to use the wave to throw these balls out of the scene as fast as possible. This powerup will show no warning except for the audible warning saying there is incoming danger.

Combo.

The in-game progress is measured by a so-called combo count. It is incremented each time a fish is eaten and reset when a penguin is killed.

When the combo meter reaches 3, 6, 9, 12, and 15, tiers 1 until 5 will be activated respectively.

When a higher tier activates, more difficult obstacles will be spawned. Also the rate at which penguins are spawned is increased after tier3. When an obstacle is hit, the combo counter is reset and so is the tier. So ball spawn speed is back to default, and the obstacle spawn too, spawning only the easier to evade obstacles.

Powerups.

Power-ups will be activated as a penguin eats a special fish. A octopus or shrimp can be eaten to activate a powerup. One of these powerups is positive. When the penguins get the octopus the penguins will get stuck in a slowly growing ball of ice. The penguins will roll over the wave in this ball instead of gliding on their stomachs. Penguins stuck in ice-balls will not be able to get killed when hit by obstacles.

When the penguins get a shrimp however the up and down controls of the wave will be inverted. When the players increase their height, the wave will move down and vice versa.

Reflection

Reflection on the product and process from a software engineering perspective.

Overall product

After weeks of hard work we can finally say that we are satisfied with our product. Over the past weeks our product has grown significantly. Every sprint a lot of work has been done so our work was equally divided over the weeks. Our game does not contain any known bugs and looks graphically pretty good.

Since we had a feature lock we made sure that all the required features were implemented before that date. Because of this we had a soft border between our features and graphics / debugging. We didn't put a lot of time in graphics on the early stages of our project. Still we managed to get a lot of graphics done in the last two weeks.

Code Structure

In the beginning our code base was not really organized. Most of our classes were in random packages and packages had weird names. This was mostly because we were unfamiliar with the engine. Along the way we started to know JME3 better and our code got better through many refactors. Now we have a proper structure in our code. Our packages and classes have clear names and similar classes are in the same package.

In our code we implemented quite a few design patterns to make our code even better. For example we have implemented reflections so new tiers or spawnable objects can be added easily.

Testing

Our testing of the system did not always go as smooth as we wanted. We did test every week but since we refactored a lot of code almost every week, we broke many of those tests. This caused a lot of inefficiently since we had to write tests over and over. Although we managed to get most of our code tested and achieved a line coverage of more than 80%.

In a really early stage of our product we encountered a problem for testing methods that created an material. This was because it required an AssetManager to make the material. Many of these methods were refactored to get a clone of an already existing material.

We had two parts of our code that were particularly hard to test, which were the cam detect package and the sound state.

Description

Description of the developed functionalities.

Camera Detection:
Our camera detection system will film the players in the queue. It will be aimed at the queue from the side. The game uses OpenCV to capture and process the images from the camera. The camera detection system detects the people on the screen by comparing each image to the previously set background image. 32 dots are divided horizontally over the screen, the height of the points is determined by the highest point of the foreground.
These points are then used in an algorithm to form the wave.

Sound system:
Sounds in our game are handled by the sound-state. The sound state holds a command queue that is processed every update cycle. Any state or control can create a new sound command and add this to the queue (eq on a collision or spawning of an obstacle or target). The sound-state will process the queue next cycle and execute the commands and so play the required sounds.

Graphics system:
The scene in our game is managed by JMonkeyEngines scene graph. On top is the root node, to this node either another node or a geometry can be attached. When something is attached to the scene graph it will be displayed on the screen.
Controls can be attached to anything in the scene graph in order to control it. When an operation is performed on a node in the scene it will be performed on everything attached to that node.

Interaction Design

Special section on interaction design (development of the HCI module).

Playing a computer game with a group of people in a public environment requires human interaction. This specific game require players to physically be positioned in a zone where the camera detection can view their bodies. They use the camera detection as controls and can view the status of the game on a big screen (all humans look at the same big screen). The group of people present near where the game is set up can interact with it in several ways.

A preset human can keep his distance and try to ignore the game. This would be the case if he/she finds the game to be disturbing (the sounds or the players actively moving could be exhausting) or it could be that the concerned human has no interest at all in which case it would be easy for him/her to ignore. This is important to consider because the setup takes up physical space and it would be inappropriate making it in such a way that all those that do not want to participate are either mentally annoyed of physically encumbered due to the set up. Therefore it should be set up in a corner / along a wall in such a way that the sound is not directed into an open room and that people can easily walk around it.

A present human can also observe the game on the screen - without partaking in the game play. The most logical case in which this would happen is when someone is queuing and the section of the queue in front of the person is playing the game. In this case the human is trying to figure out what the controls are and what the objective is. The active movements of the people playing can increase the observing person's interest as he/she realized that people are prepared to physically move to influence the game. Also if the observing person is in the queue and will soon be in front of the camera he/she might feel pressure to perform and not let the group that is currently playing down. Therefore the observing person will try to understand the key elements of the gameplay and be prepared before hopping in. This would not happen if the queue is too short in which case everyone willing to play can play. People present in the environment can also observe the game to see how well the players perform (e.g. a mother can stand outside the detection zone and watch her daughter interact with the game) and the fact that the game is displayed on one central screen would give the observing people a feeling that they are part of what is happening (instead of watching a screencast from a distance the person is actually present next to the players and can feel the general mood etc…). Finally one can also be in an observer state if one has finished playing the game and has moved along the queue outside of the detected zone. Then one would be observing out of curiosity to see if the group still performs as well since he/she has left (notice this is also why we based the game play achievements on a combo count system instead of reaching higher levels).

Last but not least a present human can be in the detection zone actually playing the game. In this case he/she would be facing the big screen and adjusting his/her highest body points to alter the structure of the wave. The player will have to constantly pay attention to spawning obstacles and penguins and firstly try to make the penguins avoid incoming danger and secondly try to make penguins reach powerups and targets to increment the combo count and reach new tiers and eventually encounter more difficult enemies - playing the game as intended.

There is no need for the players to interact with the setup of the game or even game menus. This is done by the responsible person before the game is played and once launched nothing needs to be adjusted - unless the background changes (in which case the camera detection frame needs to be selected and the background can be updated). In case the setup makes the camera detect too much along a vertical axe (which would mean that players can not fill up the detected zone and therefore can't extend the height of the wave to its maximum) the detected zone can be changed in the software to be the effective zone in which the players move.

An important aspect of game development is user testing. This involves getting a set of users to play the game and give feedback. We have tested the game during several phases of its development on small test groups. Unfortunately the test groups only consisted of people we know well (i.e. roommates and classmates). Even though this causes the feedback to be less diverse and the users most likely put in more effort to try the game (out of respect) we believe that the fact we have good relations with the test groups offered an advantage in that they tried their best to give us honest criticism on which we could improve. We set up the game in a 'full' living room (full as in that there was little space for movement without the chance of knocking over objects - not the ideal environment for this type of game) with a very inconsistent background meaning that if the game could be played in these conditions it 'should' work well when set up properly.

During the first test the game was still in an early phase and all graphics were blocks and balls with no textures and no sound. Also the gameplay was not balanced at all - it was in a state useful for implementation but not testing. There was no difficulty scale and penguins / enemies appeared at random (so that we could test if enemies did as we intended them to without having to reach a high combo count before those enemies have a chance of being spawned). Overall the feedback was more positive than expected. Test subjects took some time getting used to the game and needed some explanation from our side, but once they started to get the feel they did their best to structure the wave creating slopes for the balls to roll down and jump over and they managed to get higher scores that we initially expected. The major flaw we discovered was that a big part of losing the combo count (this happens when a penguin is hit by an enemy) was left up to luck. Enemies spawn at random but sometimes they spawn too close to the penguins in a way in which the collision is unavoidable. This obviously frustrated the test subjects up to a point where they gave up after losing in this fashion several times in a row.

We used this information to focus on the spawning system and incremental difficulty system of our game. We split the difficulty up into 'tiers' making it clearer for the players when which type of enemies will spawn and made it so that the enemies warn the player where they are going to strike - to leave the players enough time to plan a strategy to react.

As mentioned our game was in an early stage and therefore lacked visual and audio feedback. However many testers made clear it was not intuitive for them to play the game as we intended, not knowing which objects were enemies and which were powerups / targets and after playing several times one person even mentioned he did not realize one powerup would invert the controls (and just thought the game was 'bugging'. To compensate for this we chose to use simple models to represent the enemies (i.e. polar bear) and add simple audio feedback that indicates whether an event was positive (i.e. a success beep sound) or negative (when the combo count is lost an annoying 'game over' type sound is triggered).

Also adding a scenario to the game helps players understand what is going on and why they are playing. After implementing these audio and visual features we ran another test on a small group of students and received feedback confirming what we expected the new features would add to our project.

Evaluation

Evaluation of the functional modules and the product in its entirety, including the failure analysis.

Outlook