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

* Dennis has been hospitalized since week 4

Table of Contents

1. Introduction

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.
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 of the boredom and social awkwardness present 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.
The players of the game are simultaneously participants in a queue so 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. This means that our game should allow players to drop in at any time without interrupting the gameplay, but also that an engaged player may leave at any time without experiencing the feeling of loss.
For a business to install this project at one of their queuing points it must not be an obstacle in any way. The setup of the game must not interfere with the proper functioning of the queue which also means that this project can not be applied to every shape or type of queue - we chose to make it specifically for a straight segment of a one person wide queue. Furthermore since we are using camera detection we decided to limit the controls so that every participant only has to move up and down (and not make any particular motions that could annoy other queue participants) and we scrapped the possibility of having audio controls early in the process. When playing, users will need to focus their full attention on the screen and if one of the members does not want to participate it is best he/she stands outside the detected zone otherwise he/she will ‘block’ the part of the terrain corresponding to the player’s position in the detection zone which might cause the rest of the team to lose their combo count. Therefore this game might not be the most appropriate in very quiet queues or where many participants need to focus on something else. On the other hand this product can be installed in less obvious places such as a waiting zone for a slow elevator. People waiting could quickly play the game and have smiles painted on their faces when taking that elevator and reminiscing about how good it felt to work together moving clumsy penguins along a slope!

2. Overview

We created a 2D oriented game where up to eight players use physical movements to control an in-game wave. The players' movements will be captured by a camera and a 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:

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 increases, higher tiers will be activated. When a higher tier activates, more difficult obstacles will be spawned. Also the rate at which penguins are spawned is increased after the third tier. When an obstacle is hit, the combo counter is reset and so is the tier. The ball spawn speed is set 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.

3. Reflection

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 bugs that we know of and looks pretty good graphically.

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 meaningless names. Also we didn't think about the architecture a lot and did not implement design patterns much. 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 with an abstract factory pattern 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 inefficiency 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 75%.

We have tested our code during the development using unit tests. We tried to keep our line and branch coverage up as high as possible. Sometimes due to time pressure for a certain feature tests were not immediately implemented and our coverage lagged behind. Also some methods that called jME functions could not be tested. We have separated these parts into separate methods so that the amount of untested lines remained minimal.

We had two parts of our code that were particularly hard to test, which were the cam detect package and the sound state. This was because these 2 components use many assets and library calls. The sound player for example make use of an audio renderer that could not be mocked, and neither be created in the test environment (without recreating the entire game - which could make the test fail if another game component does not perform as expected). The camera detection relies on openCV that also introduced some difficulties in how to make a test that runs the code, does not crash, and still tests something useful.

4. Description

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. The camera looks at the pixel colors to see whether there are changes. We found that light on the players often gives interference in the form of flickering dots. Therefore we included a memory in the camera detection where these flickers are not picked up by the detection.

Still some interference was there so we made the detection even less prone to pick up wrong points, by ignoring differences to the background that are too small and inconsistent to be produced by a present human body. This way even if the lighting in the environment changes the people will still be the only thing controlling the wave. The algorithm divides 32 points with equal distance (on the x-axis) 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.

Wave formation

The camera detection passes the points on to the curve generation algorithm through an interface. This algorithm generates a chordal catmull-rom spline through the control points to create a custom mesh that is smooth and does not contain any sharp turns or too steep slopes. This is the wave on which the penguins slide in the game.

5. Interaction Design

Playing a computer game with a group of people in a public environment such as a queue requires smooth human-computer interaction. If input in unclear players will very easily get confused and/or lose interest in the game. This specific game requires players to be physically positioned in a zone where the camera detection can view their bodies next to each other. Because the camera looks at the highest points on the screen that is not the background, anything that is in front of or behind a player is not picked up. The players will use their own bodies to control the game in an intuitive way, i.e. the left part of the row of players will control the left part of the wave. Also raising your hands to raise the wave is an intuitive action to do when controlling the wave that is picked up instantly and automatically by the players.

Since our game is played in a public area people won't be going there with the intent to play our game. They are just getting in line for a service or walking somewhere when they come across the game. Therefore our game needs to draw these people their attention to convince them to try the game out.

In order to draw these people their attention we use a happy scene on which penguins slide over ice on their bellies. This sight is happy, colorful and cute to instantly capture the people's interest. The largest parts of the screen are the wave and the water, these have soft, mellow colors so that it will not strain on peoples eyes and they can look at the screen as much as they want.

To draw the player's attention to certain important in-game elements we also use a lot of color. When an obstacle is going to spawn, like a killer-whale or spear a red, slightly transparent block is shown on the spot where it will spawn. This way the players will instantly notice that something will happen on that area and know to get the penguins out of there. This warning block draws attention but is still soft because of its transparency. This block is the looks same for all obstacles, except for the shape which is linked to the obstacle. Only the polar bear, which swims in through the water and the snowball, that is thrown from the far back of the scene, do not have this warning block, as these are already very clear when they show up and are not deadly from the moment they are visible.

The targets in the game a sea creatures that the penguins have to touch to get their bonus. These models have bright, remarkable colors that draw the player's attention which do not give the impression that getting them will have a bad effect. The most important target, the fish that increases the combo count is red and rotates. Because of this the fish is very noticeable and one of the first things the players sees when he looks at the screen. This makes it very obvious that this is the targets that should be collected.

We have some, but not many text elements in the game. Since the players of our game will have a fair distance from the screen any text that is displayed should be big enough for everyone to easily read. The most important text is the combo counter and the highest combo yet display. Since these are the most important they are also the biggest. They are located at the bottom of the screen where they are very clear to see and are in no way in the way of the gameplay. When a higher combo is achieved and the game advances to the next combo tier, a message is displayed to indicate this. The message shows the tier number that is reached and shows a hint on what new obstacle they can expect. Because of

the hint the players will wonder what the obstacle is and engage in gameplay in order to find out what will be coming to them.

Besides colors we also use sounds to draw the people's attention and to give them feedback on what happens in the game. Every time a penguin gets a fish and the combo is increased a high pitched, positive pling sounds. Suggesting the players that they have collected something positive. When a penguin dies so the players lose their combo progress, a short tune plays that is louder and decreases in pitch while it plays. This sounds very negative and suggests that something went wrong.

Because of all the techniques we have used, the game is very simple and straightforward, and the the controls are natural, there is almost no learning curve. When a person jumps in within a few seconds he or she will learn that the red planes indicate enemies and that the goal is to collect the fish.

Because we need to be sure of the working of all these techniques, we need to test them with user tests. This involves getting a set of users to play the game and give us feedback. We have tested the game during several phases of its development on small test groups. Unfortunately these 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. Also we were able to pick some people that are naturally critical and search for the downsides of our game. We set up the game in a living room without a unicolor background meaning that if the game could be played in these conditions it should work well when set up properly.

During most tests the game was still in an early phase and all graphics were just placeholder blocks and balls without textures nor sound. Also the gameplay was far from finished like some obstacles were still in production and there was no increase in difficulty. Besides these limitations we got an overall the feedback that 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. Regardless we got some useful feedback and ideas for both improvement of gameplay and new features. Some unfortunate situations were discovered were an obstacle could not be avoided at all. 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 also found that the game was not as simple and easy to pick up as we hoped, so we added audible feedback, updated the colors palette and added rotation to important targets. We also received feedback that players had a hard time manipulating the wave because they did not feel like they had much effect. The solution we found for this problem was to add borders to the upper and lower bounds to the camera detection that could be modified so that a smaller change in the player's height would have a larger effect on the wave.

6. Evaluation

The physics of our game are not as we want them to be yet. When the wave rises too quickly there is a chance that a penguin can get stuck in the wave and shake around until it shoots out.
As our project makes use of camera detection software this needs to be setup correctly and if not, there is a high chance it will not work as expected. The camera detection can fail for some reasons.

Firstly the auto focus functions of the camera can cause false positives in the camera detection. When the camera brings the background out of focus it will seem to the detection algorithm as if it has changed, causing a lot of jitter.. The camera can also auto correct colors or brightness, having a similar effect.

The camera must have a good lateral view of the wave. If one portion of the queue is more close to the camera then the rest, then that part will look bigger to the camera. The wave will then be higher there and the controls will not feel natural anymore to the players.

People walking in front of the players will be very game breaking and annoying for the players. Because our game is aimed at public areas this can be a problem. Luckily because the game is also aimed at queues we think this problem will not arise quickly. Also, due to the same effect as described in the section above people walking behind the players will look small. If someone was to walk behind the players their effect on the wave will be smaller than that of the players and sometimes neglectable.

Sunlight can also be an issue for the camera detection. If the light changes (e.g. a cloud passing in front of the sun, or the weather changing from a sunny day to a dark grey storm) the image changes giving false positives. If this happens the background will have to be reset. Also the shadows of the players should not fall on the background as the algorithm will detect this as foreground. Luckily as long as the height of the player shadows does not exceed the player heights and the offset of the shadows per player is not too large then the game controls will continue to function properly.

This is mainly because we decided to implement our own foreground detection algorithm instead of using the adaptive one already built into opencv (which takes into account if the background slowly changes). We find that in the current setup our algorithm runs faster (because it is simpler and involves less computations) but also slightly more accurate (the built in opencv algorithm does not always accurately segment background and foreground and when a player stays on screen for too long the results are not consistent). Furthermore the camera must be placed on an immobile structure and may not be moved when in working. Even tapping the camera might cause it to shift and give false positives, but only if the background is not a single color.

7. Outlook

We feel like our game has come a long way, but we imagine that it can be a lot more than this. We picture several improvements that we could make in the future in order to greatly improve our game.

The first improvement we want to make to our game is even further improve the the clarity on which part of the wave each player has an influence at a specific point in time. For this we imagine a display behind the wave that shows the players at the location in the wave where they have an an influence. The display would be semi transparent and possibly have a color filter to fit into the theme of the game. The display must also take the shape of the wave for aesthetic reasons: so that there is not a big square display sticking out behind the wave. We have actually already have a start for this improvement, but it does not work well yet and is unfinished.

The second improvement we want is a tutorial for the game that displays if the players are not performing well. If the players do not collect fish for example, a hint will be displayed on the screen.

The next improvement involves creating more visual feedback. We want to add particle effects on dying penguins, smoke from the lightning, water splashed from the bear and whale and some wind-like effect from the spear soaring through the air. The spear and snowball should also follow a more natural curve as a real one would under the effect of gravity. Animations on the models would also be a nice improvement. Penguin movement would be made more fluent, and penguin ‘success’ animations would be triggered when they reach a target causing them to enter the next tier. The bear would have a claw striking animation and the whale would open its mouth when it reaches the top of its attack zone (and maybe spray water out of its back when capturing a penguin). The snowballs thrown by a yeti would actually have a yeti far in the background that throws them. Also the dangerous ‘spiky’ balls should have actual spikes on them.

A fourth improvement we want to make is creating multiple themes, so the game can be applied to for example theme park rides, musical (theater) queues with similar themes or any other service that would like some other theme than the polar theme with penguins.

As an optional fifth improvement, we might make the environment a horizontally side scrolling game to give the feeling of progression. Though were not sure how this would work out and we have not discussed this thoroughly in the group.