SUMMATION OF FINDINGS FOR PORTAL 2D

Mathew Blair (mblair), James Fitzsimmons (fjr), Joana Lopez, (joanal), Pongsakorn Champ Kanjanabutr (pkanj) SWEN30006 - Software Modeling and Design

In this report we aim to summarize the findings made during the implementation process of Portal 2D. Specifically we aim to highlight the noteworthy advancements made in background services for the game, as well as problems encountered and resolutions we found.

Our implementation process has generally run quite smoothly given the complexity of our game design and required calculations. We have, for the most part, managed to stick quite close to our initial design overview, only changing specific designs when a better solution was sought and found. A key example of this continuous improvement is found in our multithreaded online high score loading system.

The initial design had specified the high score loading system run in step with the other level functions, which worked fine for a local high score system. The problem came when switching to an online score loading system. However, given we don't have a dedicated server to service our game, we were experiencing significant loading time issues whilst submitting scores. This presented a user interface problem as the game would become unresponsive while it was attempting to submit the score. We have gotten around this by having a background high score loader thread, that can asynchronously upload scores to the online server and download the latest score updates at a predetermined interval.

This provides significant improvements in game response to the user as they can still manipulate the game environment and check scores without waiting for the server to be available for update. It also allows us to store a local cache of the high scores so that the user can quickly and reliably access high scores, without having to wait for loading. It has the downside of not always having up to the second high scores loaded but we believe this is a small price to pay for the efficiencies gained.

The second implementation success of note, is the manner in which we achieved seamless portal transitions with correct application of real world physics to our game engine. The use of a powerful physics engine, whilst difficult in its complexity and learning curve, has allowed us to achieve what we consider to be very realistic simulations of a physical world. This is key to our ability to successfully provide a learning experience to the players, as any intuitions they were to learn from an inaccurate physics world would not be applicable to real life and hence would be useless.

We believe that playing the game does provide one with a concrete understanding of the concept of momentum and the conservation of momentum, allowing us to achieve our requisite learning aims for the player.

To further the polish of the game, we have extended the physics engine to account for devices such as switches to allow realistic depression of these objects. This is evident in the small interactions. For example if you drop a cube from great height onto a switch, the door starts to open, then the cube bounces up, and the door closes, before the cube lands and opens the door again. We are strong believers in consistent solid user experiences and it is our view this should be reflected in every detail, no matter how small.

This brings up the third point of note, which is less a Java implementation design but is more something on which the whole game leans quite heavily. We have spent hours crafting an original experience

for our players by creating all of our own animations, level artwork, level tiling, music, sound effects and levels.

We have had a few issues with sound and music playback: namely that when you quit the game, the slick OpenAL libraries crash because of faults with Slick' implementation of sound buffer handling. We have managed to fix one of the errors with their sound control by modifying the slick source code and recompiling the .jar file, however given the current issue only causes problems on quitting the game we did not deem it worth our time to rewrite the slick OpenAL libraries for this particular project.

We also had to slightly modify our design for game object updates to allow implementation of animations, however we deemed the addition of animations to be worth the cost as they provide a significantly more immersive game experience to the player.

Another change made to the initial design was the development of an in-game pause menu. Initially we had planned for a straight forward pause menu that allowed the player to take a break. However given the speed at which we were progressing with the implementation we decided a fuller in game menu could be a useful attribute to have and so developed a state based in-game pause menu that allows players to see scores for the level, change options for input and volume level as well as restart the level if they get stuck. This allows for a more natural control flow and makes the state more useful than a simple pause and exit option.

In discussing the negative aspects of our development process, I feel inclined to mention that we do not have as many levels as we had hoped. This is simply due to timing problems, however is not a difficult problem to fix as it just requires creating levels and adding them to the xml file for loading. This is the final great advantage of our system we wish to demonstrate, the extensibility for future additions.

We have designed the system so that the game will automatically adapt to the addition of future levels and achievements, as well as making it easily extensible for the addition of new game mechanics. The ease of adding new levels and achievements is an important aspect if this were a real product, as it allows continuous improvement to the game long after the product has been shipped. For example if we were to create an expansion pack, it would only need to include the level foregrounds, level xml files and updated loading list, which would be a very simple process to patch as it does not touch any of the compiled java classes. As an example to demonstrate the extensibility of new game mechanics, we only decided yesterday to add the dissipation fields to the game, which was simply a manner of creating the dissipation field game object and adding a call in level to update the objects when required and render them. The game objects handle all of their own interactions and as a result we do not have to modify anything else as they are self contained, as we would expect a good object oriented design to be.

Overall we believe that we have been quite successful in this project, implementing what we believe to be a complex design with only a few problems in creation, with plenty of time to extend the project to add more interesting features. We are very happy with our design and the choices we have made to change it where necessary to ultimately end up with a better result. We welcome any constructive criticism you may have, and hope you enjoy Portal!

The Portal2D Team.