Game Engines Report

by
Alexander Kirk
and Emil Buhl

IT University of Copenhagen
MGAE, F2015
Dan Lessin
December 17, 2015

Contents

1 Introduction

This paper is an technical account of the the design and implementation of the game Redshift in the Unity game engine. This paper will present the initial concept idea and the vision behind the game, followed by a chronological account of the development process and finally an analysis of the implementation and of alternative solutions evaluating the final implementation.

1.1 Scope and Context

The development of Redshift was the subject of two exam projects at the IT-University of Copenhagen during the fall semester of 2015. These two projects belonged to two different course - Game Design and Game Engines. The scope of these two projects are very different, as game Design is focused on the design of mechanics and aesthetics of games, where as Game Engines is focused on working with game engines - coding, scripting, etc. This paper is part of the project on Game Engines, thus the subjects pertaining to the game's design and it's mechanics is not within the scope of this paper. However the focus will be on the implementation of said mechanics. The projects done for Game Design are usually technically simple and often done with 2d-graphics. However merging the Game Design and Game Engines projects allowed for a more technically challenging features and mechanics. The team on the Game Design project included two additional members. These members had only very limited technical experience and was assigned the roles of designers. Working asset creation, ie. sounds, texture, models. They did not work on the scripts, and did only limited work in the Unity editor. Any content created by these members has been gathered in a list(**SE APPENDIX**).

2 Project Description

The goal of this project was create a puzzle game using the Unity 3d-engine. This game should incorporate a fourth spatial dimension, making it a 4d-puzzle game, which require a fourth dimension being implemented in the Unity 3d-engine.

2.1 Vision

2.2 Redshift

3 Week Structure

In this section, we will describe the time table and weekly progress throughout the project. We will go over problems and some design decisions, especially those that had consequences for the overarching work flow.

3.1 Time Table

This time table reflects who took responsibility of what tasks. The table itself was developed without any assignment of responsibilities before beginning the project.

Week number	Task	Owner
31-10-2015	4D-Movement	Both
	Colour switching	Alex
	Pickupable Objects	Emil
7-11-2015	Composite objects, tuning of carrying implementation	Emil
	Interactive objects	Alex
14-11-2015	AI, Hazards and In-game displays	Both
21-11-2015	Doors and locks, Refactoring of code	Alex
	Elevator, level design	Emil
28-11-2015	4D-lighting, sound effects	Alex
05-12-2015	Polish	Both

Table 1: Work schedule separated into weeks

3.2 Breakdown

3.2.1 Week one

Initially, we needed to decide upon what approach to take in regards to the core mechanics of our game. Using the standard asset 'FPSController' shipping with Unity, we only needed a way to explore the fourth dimension. We figured that this movement would take on either of two possibilities; integral or fluid.

To better explore these possibilities, Emil started working with the integral approach, while Alex explored the fluid approach.

Our efforts revealed that a fluid approach would be much more faithful to the mathematical concept of dimensional movement, but would be exceedingly complicated to implement properly in the Unity engine, since the collision detection is handled by Unity behind the scenes. Fluid 4D-movement would require the collision detection system to be extended with a fourth coordinate, which was simply not feasible.

On the other hand, integral movement - being the more 'unrealistic' approach - would be much easier to implement even without compromising the core idea of the game. Additionally, it complimented the collision detection system of Unity through the use of collision layers, which with little interpretation could be transformed into a working 4D-collision engine.

With the movement system in place, Emil implemented the first version of pickupable objects, inspired by unitylessons.com[1]. Meanwhile, Alex implemented the first version of the colour switching algorithms.

3.2.2 Week two

During the second week, we started working with especially composite objects; objects that stretched across multiple collision layers (had a w-width larger than 1). These objects proved to become a problem, given that they seemingly did not follow the laws of physics, detaching from each other seemingly at random.

Investigating, we concluded that this was a problem founded in our PickupObject script[citation needed], where we forced the displacement of carried objects instead of applying forces to it, hence bypassing the inherent physics inside Unity. This problem also meant that objects could displace through walls. Emil began solving these issues, while Alex started working with interactive elements, such as buttons.

Multiple problems were fixed during the week, with both issues related to PickupObject being solved. Button functionality was also added to the code. Regrettably a new problem appeared, where carried objects started floating around the player erratically, increasing in speed as it went. This problem was solved later in the process.

In the end, composite objects achieved a working state. While they did carry some complexity during implementation, their behaviour was reasonable.

3.2.3 Week three

With most of our core functionality implemented, we made an attempt to implement a simple AI and in-game monitors.

The AI was thought to be used as an opponent, but was never fully implemented to fulfil this position. Likewise, the in-game monitors were thought to relay information that was not obviously available to the player.

Due to misunderstandings, we thought the week to be imminently before feature freeze and focused our efforts implementing whatever features we found was still lacking. This reasoning was of course flawed, since we had no need for neither an AI or in-game monitors, and these never appeared in the final product.

The implementation of these minor, irrelevant features took time away from refactoring of our currently implemented code, which could have spared us many troubles later in the development.

3.2.4 Week four

Having gathered all of the features we deemed necessary, we started this week by distributing work between us. Alex became responsible for refactoring all of the currently available code, additionally creating the first implementation of the doors and locks that would later be used. Emil switched focus, and started development of content and designing levels, and was responsible for implementing the first elevator used in the game.

This week was undoubtedly the worst in work flow, and arguably catastrophic for the end result.

Given that the code implementation up until this point had been very specific as to its use, most of the refactoring was concerned with making the code more generic and easily accessible for the designers. Yet unintentionally, the refactoring created a multitude of new problems, breaking all of the previous functionality and thereby rendering most of the code useless. While it did fulfil its purpose regarding core functionality, many tangential features were utterly useless after the refactoring, e.g. the elevator and socketing system were broken, and buttons were volatile.

Upon realizing the situation, we decided to start fixing a multitude of problems, eventually resulting in working, albeit unstable, code. The fixes included correctly rendering trees and bushes, reimplementing pickupables and inventory items, fixing interaction between multiple 4D-objects, adding additional algorithms for animated doors and similar objects, and fixing the floating of carried objects.

3.2.5 Week five

While our time table suggested that we were to add 4D-lighting and sound effects, most of the fifth week went by with content creation and bug fixing.

4 Discussion and Conclusion

5 Bibliography

References

[1] unitylessons.com. 2014. *Unity3D Tutorial - Pickup and Carry Objects.* [Online]. [Accessed 16 December 2015]. Available from: `https://youtu.be/runW-mf1UH0`