**CS4483**

**Group Game Postmortem - “Buying Time”**

**2020-04-03**

**Group Members:**

**Reid Jackson**

**Morgan O’Brien**

**1.  Project Summary**

*You should provide a brief overview of the project, outlining the main development objectives and tasks that had to be completed along the way.  The original game pitch and your design documentation should provide a suitable foundation for this.*

Our development efforts changed quite a bit over time. We were initially going to pursue Reid’s 2D dungeon crawler, incorporating paper as currency to buy blueprints for gear. This plan changed and changed until we settled on our idea for “Buying Time”.

Buying Time is a survival game where zombie-like enemies spawn in increasingly difficult waves and the player tries to fend them off for as long as possible. We still wanted to incorporate paper as currency, with the ability to gain paper or money from killing enemies and be able to spend the money on blueprints for upgrades like speed, strength, etc. The game will be a 16-bit 2-D

Our development process was planned as follows:

* Create/find the artwork. We wanted to keep a spooky dungeon-crawler feel to this game. Our enemies, hero, and map all needed to have a somewhat coherent theme.
* Create the map. The map should be big enough that the player has lots to explore but not too open so that it is challenging to get through hallways and into to certain areas. The map design ended up being a castle/dungeon style with many rooms.
* Create the player functionality. We wanted our player to be able to run in 8 directions and have a melee sword attack. This would make the game more difficult than having a point-and-click projectile or gun, since the player must engage the zombies up close. The melee range should be slightly larger than the zombie attack range so the player can successfully kill enemies without taking damage.
* Create the wave-spawning system. The enemies should spawn once the last wave of enemies was fully defeated. Each wave should have more enemies than the previous.
* Create the enemy AI. The enemy should pathfind to the player when the player is within eyesight/range of the enemy.
* Create the health system. The player’s health should drop when attacked and the player should regain health by defeating enemies.
* Create the paper money system. The player should gain money when he defeats enemies. The player should be able to fight his way to buying stations and purchase upgrades or health with his money, given that he has enough time to make the purchase without being overwhelmed by enemies.

**2.   Key Accomplishments**

*In this section, you discuss the highlights of the project.  In particular, you should address the following questions:*

* *What went right?*
* *What worked well?*
* *What was found to be particularly useful?*
* *What design decisions contributed to the success of the project?*

*Think about these questions in the context of both your game/technology and the processes/methodologies used in constructing it.*

We accomplished much of what we initially set out to create. The creation of the skeleton framework for the game went particularly well – the map, the artwork, the characters, all were created on schedule and to a good effect.

Creating the map went very smoothly. Reid used Unity tilemap editor, which we found extremely useful, to create the map and the collider boundaries and we were able to create a large, detailed map. The tilemap editor made it much easier to draw large rooms and copy patterns to save time and effort. We used dark castle textures to create a spooky, indoor feel.

Creating the player movement and animation went very well also, since we had much of that implemented in both of our prototypes. We still needed to re-do all the player animation since we had a new hero character. The experience we had from doing it before made it easy when we had to also animate his melee attack, and when we needed to animate the enemies.

Another thing that went smoothly was the arrangement for the background music, which Morgan did in Ableton with royalty-free sounds. We wanted an intense feel with a fast tempo and we accomplished that.

The wave spawning mechanic was another great success for us. Reid tested many different algorithms for the exponential spawn rate – in order to make every wave of enemies more difficult than the rest. We placed specific spawning locations around the map and were able to make effective wave spawners with a relatively simple C# script. The enemies pour out of these locations as the previous wave is killed.

One thing we found extremely useful was the Unity “Collaborate” feature, which allowed us to push changes to each other during development in real time. This feature would commit all project files to a cloud repository which we could both pull from and remain up to date. The hierarchy of files and commits is all handled by Unity and it was a very easy system to learn and use. This feature was especially welcome once the COVID-19 virus forced school to close and we could no longer work together in person. We had brief merge issues once in a while when we both would change something in the same file at the same time, but these issues were simple and few in number.

**3.  Key Problem Areas**

*Things can't go smoothly all the time and every road has a pothole or two along the way.  In this section, you discuss some of the issues encountered in working on your project.  In particular, you should address the following questions:*

* *What went wrong?*
* *What project processes didn't work well?*
* *What technical challenges did you encounter?*
* *What design decisions made it more difficult to succeed in your project?*
* *What were the effects/impact of these problem areas?*
* *What corrective actions did you take to resolve the problems?*

*Again, in answering these questions, consider your game/technology and your processes/methodologies.*

There are several areas that gave us trouble during development, even up to the finished handed in demo.

There were troubles with spawning enemies, sometimes it appeared as though enemies stopped spawning, although they still should have infinitely as long as the player was still alive. The issue was eventually addressed as an issue with the math behind how many enemies should spawn and the choice of which spawn point to have them spawn at (as they spawn randomly).

A large issue that involved a very simple solution involved the overlay on the player’s screen. The overlay included the player’s health bar and paper amount.

Pathfinding in our implementation didn’t work well, using a pathfinding such as A\* or a map mask to allow for oathing proved troublesome and would not work properly at the end of the implementation. A barebones, straight-line pathing was implemented to at the very least have the enemies ‘follow’ the player. This caused problems however as enemies could get stuck on collision boxes or kited and have their pathing mechanic abused.

The idea of earning paper from killing enemies and being able to spend it to purchase upgrade and items from blueprints was a decision we made to incorporate paper into the game. The issue arose from the blueprint buy stations to access the player from public helper methods to give or increase public variable stats, like speed and max health. Although the same implementation of accessing player methods worked for other scripts run in game, like the combat, this one did not. Several attempts of trying to grab the player from different Unity variables like by tag and GetComponent were tried, but none of which worked. Due to this, the buying stations exist and have their functionality ready, but don’t end up doing anything in the demo functionally.

A technical challenge mentioned previously was the Unity Collab feature, allowing for both of us to work on the same project at the same time and push/pull changes. The issues arose if we both worked on the same file at the same time, this would cause one member’s changes to be lost as the other’s would be replaced. This was resolved by declaring which files were being edited at the time and when they were pushed and free to work on again. Another issue came from saving the Unity workspace before pushing, this allowed for changes the other made, such as moving files to another location to be no push properly. For example, a script file would be moved for one member and only exist in one place, but for the other, they either wouldn’t get the new version or would get duplicate files in the workspace. This was resolved by periodically checking workspaces for files and choosing which member’s workspace to take as the master workspace from that point. Any pushes made at that point to consolidate changes were done without saving the workspace for the ‘slave’ member.

**4.  Lessons Learned**

*Summarize and describe the key lessons and takeaways from the project.  Think about what you would do differently and what you would do the same.  Also, be sure to include new processes or best practices that were developed or discovered as part of the project, and to discuss areas that could be improved and how you would go about doing so.  Finally, discuss if you would like to work on a similar project again in the future, as well as why/why not.*

Creating a 2D world with Tiled program was satisfying and allowed for an easy, streamlined way to create 2D Tile maps from sprite sheets very simply and quickly. It allowed for the creation of presents of terrains like paths, windows and walls, so that you could paint these terrains quickly without having to go tile-by-tile. Creating a large and detailed world was an enjoyable process (though it was a rabbit hole of touching up areas of the map over and over to get them just right, a perfectionist’s worst nightmare). This is a program I would continue to use to develop 2D maps.

What we would do differently in the future would be to have more set roles of development. This would be one us handles the player, one the enemies or the map. Although a lot of work overlaps two roles like the player damaging the enemies, it created problems for development. One of the issues mentioned previously was overwriting the others work because you pushed your changes onto theirs. Also, it would allow for a more in depth understanding of the underlying system instead of parts of for example the player class being developed and understood by different members based on who implemented it. Of course, only having 2 members working the project, overlap of roles was bound to occur eventually.

A process we began using during the development of the game was the use of Unity’s prefabs. These prefabs allowed us to create a type of object, like an enemy and create a prefab of the enemy. What the prefab allowed for was an easy duplication of the object, on the fly, without cluttering the game world with many different copies of the same thing. Prefabs allowed us to create clones of the original, like the enemy, to all spawn with the same properties on the map, without having to manually create and define each one.

The pathfinding of enemies would be top priority of improvement continue development. Replacing the simplistic straight-line pathing with a more sophisticated algorithm such as A\*, and on top of that a form of AI for the enemies to be ‘smarter’. This would improve the game experience and make the enemies feel more alive and interactive. As of right now, they feel very simple and mindless and, in the future, this would be a priority to get working properly earlier on in the development.

Both of us agree that given the opportunity to work on a similar project in the future we would. It was a fun learning experience, getting know developing in Unity and create something of our own, as well as being able to problem solve was worthwhile. This project let us be creative while still having a structure. For both of us personally, we will most likely continue to develop small games on our own time just for fun and to keep learning new types of games and how to develop them.