CS 442 Project Report
Zombie Dash

Document detailing the deliverables and development produced for the project

Prepared by

Brian Yu, Michal Fitzryk, Tim Jang, Tim Villaraza

***for use in* CS 442**

at the

University of Illinois Chicago

Spring 2022

List of Figures

List of Tables

No tables in this document.

I Project Description

1 Project Overview

Our project is focused on the creation of a 2D shooting based game called Zombie Dash. The game involves a player that has to escape from hordes of zombies that follow the player around within a map. It will be a survival/strategy game which has the player explore the map to get resources to complete levels.

The player will be able to do some basic game mechanics that is to be expected from a shooting game. This include movement, shooting, and picking up items for the player to be able to use. The player will also have a health bar and stamina meter that will be used to enhance the survival aspect, making it more realistic.

2 Project Domain

Our game will be distributed on PC platforms initially, as outlined in the document by the preceding group. The PC platform is one of the most flexible, as many people have some sort of desktop or laptop computer, and our game would be not as performance dependent as other games, making it able to run on a variety of different PC specs.

However, we would also like to distribute the game on tablets and mobile devices. As mobile applications are becoming a more rapidly growing sector of gaming due to everyone having mobile phones or other devices, putting out the game on these platforms will be a great asset for the future. Our game will be able to be ported by these platforms by putting it on Steam, Epic Games, and Google’s Play Store.

3 Relationship to Other Documents

This document was created with references to other documents detailing the process of software engineering, including Robertson and Robertson's *Mastering the Requirements Process*, Silberschatz, Galvin, and Gagne's *Operating System Concepts*, and J. Bell's *Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports*. The UML diagram system that we used for this documentation follows M. Fowler's *UML Distilled, Third Edition*. The original project design for the coding project was developed and designed in the document entailed here by CS 440 Spring 2019, Group 14: Danielle Hatten, Manuel Torres, Meaghan Ryan, Nerijus Gelezinis' *Zombie Dash*.

4 Naming Conventions and Definitions

4a Definitions of Key Terms

- 2D - A physical space defined by an area with height and width but no depth.
- Tilemap - A 2D grid made up of rectangular tiles of equal size to display an image.
- Collision - Contact between two objects to result in some interaction.
- Sprite - A computer graphic that can be moved on-screen and manipulated as a single entity.
- Pathfinding - The plotting of the shortest route between two points, generally relying upon Dijkstra's Algorithm for finding the shortest path.

4b UML and Other Notation Used in This Document

This document will generally follow the Eclipse Papyrus Version 3.1.0 of UML, which is the current up to date version of UML. The arrows show the relation of the entities in the UML diagram. A use case is found inside the box and is named with the function. Single lines are referred to as association lines which represent relations and are named with the function.

4c Data Dictionary for Any Included Models

No data dictionary models are included for this project as it is not within the scope of our application.

II Project Deliverables

5 First Release

Figure 1 - Screenshot of First Release of Demo

For the first release of our project, which was on Febuary 25th 2022, The after player sprite and map textures was chosen the functionality of the player sprite was added. The player sprite didn't match the map very well as despite being a top down game, the player sprite was a completely top down rather than partly top down. The player's movement controls were usable as the player was able to use WASD to move around the map. The player's aiming systems was also implemented as the player sprite was able to face the direction in which the mouse was pointing at. The camera was also locked onto the player sprite so that when the player moved, the camera would follow the player sprite. This allowed the player to see where they were going and also enjoy the game. The player also had to left click in order to shoot and repeatly in order rapid fire.

3 files containing a map in each one were created for the first release. However, due to functionality problems with the collisions of the map and player map 3 was the only map/file deemed to be usable for reveiw. Even then, the player was still able to move through the buildings and the border. The starting menu was also created where if the player pressed play they were able to play the game.

6 Second Release

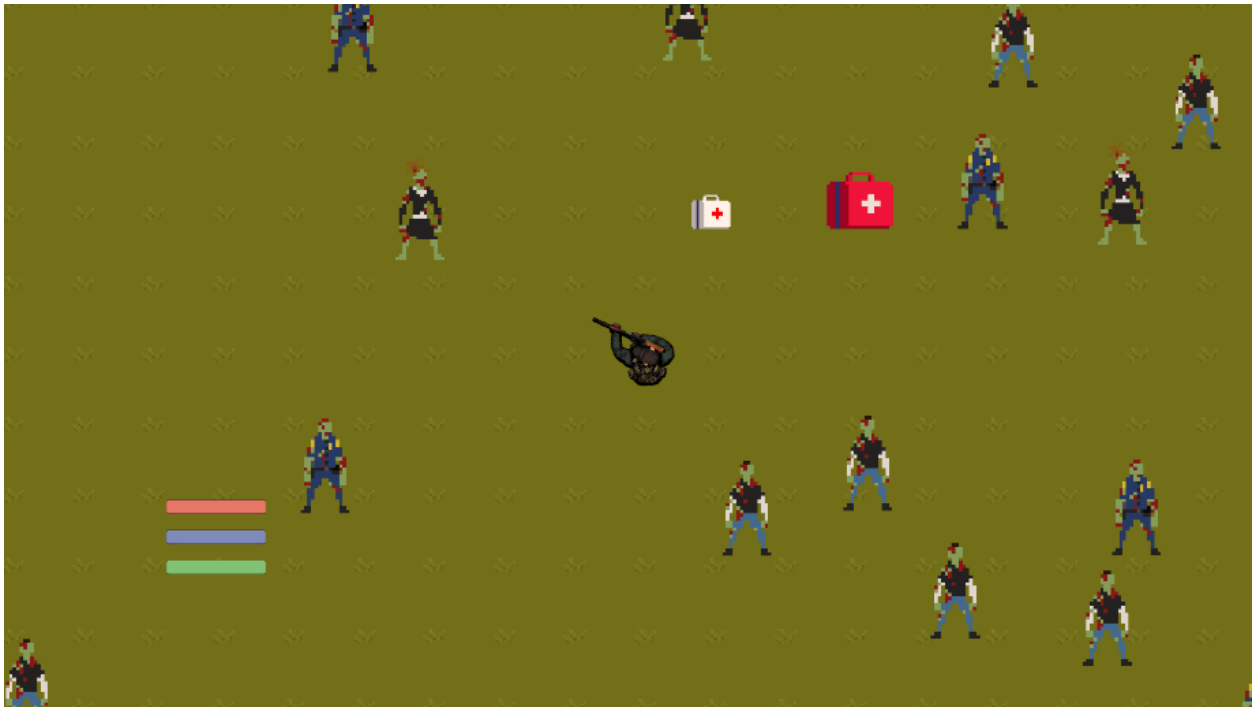

Figure 2 - Screenshot of Second Release Test Scene

When reaching the second release, at April 4th 2022, There was more to the gameplay. The map was able to spawn zombies in the open areas (where there was no collision) and were able to move towards the player no matter where the player went. When the zombies were killed by the player, there was a chance that they were able to drop a med kit. If the player was able to get to the med kit then they were able to health themselves. The zombies were also able to move around the in game objects and parts of the environment where there was collision in order to attack the player. There was an issue with collison where the zombies not only overlapped eachother, but if the player fired at them at that state they would all be destroyed at the same time.

The player at this point now had auto fire where the player just has to hold the mouse button to fire repeatedly. The game overlay also showed the player stats suchs as health, etc. If the player were to be attacked by the zombie then the health bar decreases in response to this. If the player health bar were to be empty, then the screen would display a game over massage and allow the player a chance to restart the game if they pressed the restart button.

Not only was music added to the game, but if the player was in the main menu and pressed the sound button, they would be able to adjust a slider to change the volume of the game. Sound effects were also added to the game depending on how the game went. This included zombie

melee attacks and when the player was shooting. Also the collision were fixed so that the player cannot leave the map or walk through the buildings. Though there we still some texture issues where only parts of the game environments (mainly trees and bushes) showed up in display with their top half cut off. Also more detail and in game objects were added to the game for greater in game experience.

6.5 Third Release

Figure 3 - Screenshot of Second Release of Demo

For the third and final release, which happened on April 28th, 2022. All collisions issues at this point of development were resolved where the zombies were no longer overlapping each other/properly colliding with each other. Also any texture issue regarding how only half of the game object was showing were also resolved as well.

The other maps/levels were also created in the same file as a way to both save time and make sure the game properly functions as intended. Also the game at this point now had the ability to allow the player to teleport to the next map/level when they are able to reach a yellow dot located on the map that the player is present at. When on a different map/level the zombies functioned the same way they did in the previous map/level and proceed to attack the player. Zombie animations were also implemented into the game, though they only move a little. If the zombies were to be killed they have a chance of dropping 2 different medkits: one small one that restores

25% of the health bar and one big one that restores around 75% of the health bar (only fills to the max). The 2 medkits are also given their own skins and sizes.

The player is now able to dash in the game whenever the player presses the shift button. But, there no animations for it so it looks like the player is teleporting from one place to another based on the direction that the player is facing. Whenever the player dashes the blue bar on the UI decreases from each use. Then when the player stops dashing or the bar is fully empty the bar slowly refills itself.

7 Comparison with Original Project Design Document

In comparison to the original project design document, we did not port the game to mobile devices. The original document wanted a mobile first approach whereas we went with desktop first one. Our application can be ported as a mobile one, however we do not know how to manage the controls in this aspect, nor did we have time to try implementing it in this fashion.

Another difference between our game and the design document is multiplayer. The document describes having multiple gameplay modes including single and multiplayer.

III Testing

8 Items to be Tested

Player Movement Script

ID# - PlayerMovement

Collision

ID# - Collision

Player Gun

ID# - PlayerGun

Zombie Movement

ID# - ZombieMovement

Damage/Health

ID# - Damage/Health

9 Test Specifications

ID# - PlayerMovement

Description: Player movement script allowing for player movement and interaction.

Items covered by this test: Movement with WASD, Dashing with SHIFT, and looking where the MOUSE points at.

Requirements addressed by this test: Functional Requirements.

Environmental needs: Unity.

Intercase Dependencies: NA.

Test Procedures: Start the game and have the user control the player.

Input Specification: The WASD button, SHIFT button, and the MOUSE.

Output Specifications: The position of the sprite changes and the direction the sprite is looking at changes.

Pass/Fail Criteria: The player sprite moves on the screen based on the player's input when playing the game. The sprite faces the same direction where the mouse is located at. The player changes position faster if SHIFT is pressed.

ID# - Collision

Description: Collision makes it so that the the game characters are unable to go eachother and the enviroment.

Items covered by this test: Environment interaction, character interaction, object interaction.

Requirements addressed by this test: Functional Requirements.

Environmental needs: Unity

Intercase Dependencies: PlayerMovement, Collision

Test Procedures: Start the game and have the player and zombies interact with the environment and each other.

Input Specification: WASD buttons

Output Specifications: Player and zombies are moved on screen into eachother or the enviroment/objects.

Pass/Fail Criteria: The test passes if the zombies and the player are unable to pass through the objects with collision, unable to pass through the environment with collisions, unable to pass through eachother, and unable to pass through other zombies.

ID# - PlayerGun

Description: Player shooting script allowing for player to shoot bullets.

Items covered by this test: MOUSE and MOUSE 1 (left click).

Requirements addressed by this test: Functional Requirements.

Environmental needs: Unity

Intercase Dependencies: PlayerMovement, Collision

Test Procedures: Start the game and have the player to spawn on the map.

Input Specification: Mouse position and MOUSE 1 (left click).

Output Specifications: Bullet objects are shown to be both spawned and moving on screen.

Pass/Fail Criteria: The test passes if the player sprite is able to shoot the bullet based on the command of the player, the bullet is launched based on the direction that the player/sprite is looking/pointing at with the mouse.

ID# - ZombieMovement

Description: Zombie Movement script allowing for zombie to move towards the player.

Items covered by this test: Sprite movement

Requirements addressed by this test: Functional Requirements.

Environmental needs: Unity, Pathfinding program.

Intercase Dependencies: PlayerMovement, Collision

Test Procedures: Start the game and have the zombies move to the player position.

Input Specification: WASD button

Output Specifications: Zombies are shown moving on the screen to the player.

Pass/Fail Criteria: The test passes if the zombies not only goes to the player's position (even if the player moves), but it is able to move around obstacles to reach the player.

ID# - Damage/Health

Description: Zombies and players are able to damage eachother

Items covered by this test: Zombie damage, Player Damage, Player Heal

Requirements addressed by this test: Functional Requirements.

Environmental needs: Unity

Intercase Dependencies: PlayerMovement, ZonbieMovement, PlayerGun

Test Procedures: Start the game and have the zombies move to the player position and attack him.

Input Specification: MOUSE, MOUSE 1.

Output Specifications: Zombies are removed when killed by the player, player dies when killed by zombies, player heals from health drops.

Pass/Fail Criteria: The test passes if the zombies are removed from the game after being hit 3 times, the player dies from 3 hits by a zombies up close, player gains 25% health back when interactiong with the small mnedkit, and player gains 75% health back from interactiong with the big medkit.

10 Test Results

ID# - PlayerMovement

Date(s) of Execution: 2/24/22

Staff conducting tests: Tim V. and Brian Yu

Expected Results: Player is able to move accordingly to the input provided by the user.

Actual Results: The user is able to successfully move the player sprite according to the input provided by the user.

Test Status:

Pass.

ID# - PlayerGun

Date(s) of Execution: 2/24/22

Staff conducting tests: Tim V. and Brian Yu

Expected Results: Player is able to shoot accordingly to the input provided by the user.

Actual Results: The user is able to successfully shoot based on the direction that the mouse is positioned at.

Test Status:

Pass.

ID# - Collision

Date(s) of Execution: 2/23/22

Staff conducting tests: Tim J., Michal Fitzryk and Brian Yu

Expected Results: Player and zombies are unable to move through eachother and obstcles/enviroment that have collisions.

Actual Results: The player was able to phase through all of the obstcles/environment that has collisions activated.

Test Status:

Fail.

ID# - ZombieMovement

Date(s) of Execution: 3/14/22

Staff conducting tests: Tim J. and Michal Fitzryk

Expected Results: Zombies are able to reach the player and avoid obstacles.

Actual Results: The zombies didn't move after being spawned into the map and errors were being displayed in the unity console.

Test Status:

Fail.

ID# - Damage/Health

Date(s) of Execution: 3/14/22

Staff conducting tests: Tim V., Biran Yu, and Michal Fitzryk

Expected Results: Player is able to get back health after interacting with the health pack dropped by the zombie after being damaged.

Actual Results: The player gains back its lost health.

Test Status:

Pass.

11 Regression Testing

The ZombieMovement test had to be repeated multiple times as more levels were added to the game later on in the development process. This was due to the fact that the pathfinder program wasn't allowed to have duplicated grids in the same file. So the pathfinder grid had to be expanded to reach the other level. Then it had to be checked whether or not the zombies were still able to move properly.

The Collision and PlayerTestMovement tests also had to be repeated together as more maps were added to the git repository. This was due to the fact that as a new map file was created all collison setting had to be reapplied to it. Therefore it had to be tested whether or not the collisions were functioning properly with player movement. Collision testing was also repeated to make sure the bullets fired by player were proplery interacting with the enviroment and in game objects.

IV Inspection

12 Items to be Inspected

Player Movement Script

ID# - PlayerMovement

Collision

ID# - Collision

Player Gun

ID# - PlayerGun

Zombie Movement

ID# - ZombieMovement

In-game Menu

ID# - InGameMenu_Behavior

13 Inspection Procedures

Our group held meetings to inspect and implement each other's codes which we would eventually merge into the final product. We held 14 meetings during the project development cycle, with one meeting per week to discuss our project development and progression. On occassion, we also held an additional meeting on some weeks to discuss other issues or solutions that have came up. 20% of the work on the project was done during project meetings, with this primarily being merging repositories of our branches into master and fixing glaring bugs or glitches that have occurred. The other 80% of the work was done outside of these meetings, either through pair programming or by individual work as we each focused on our specific tasks. The results of our meetings were discussed over voice calls during times we were all available electronically, and some were done in person during brief circumstances. Our group also did not use checklists for the inspection of our code, but focused on making sure the functionality of the game was working properly through the use of playtesting and analyzing the output of the engine itself.

14 Inspection Results

ID# - PlayerMovement

Inspected by: Tim Villaraza, Michal Fitzryk, Timothy Jang

Date of Inspection(s): February 13th, February 20th

Results: Player was able to control their movement properly and worked on the first inspection. When changing the scene, the player moved too quickly had to reduce the movement speed to make it scale better with the rest of the environment. Movement then was seen as completed and operable.

ID# - Collision

Inspected by: Tim Villaraza, Michal Fitzryk, Brian Yu

Date of Inspection(s): February 13th, March 27th, April 3rd

Results: Player was able to collide with objects within the scene during the first inspection. When zombies were added into the game, a reinspection was done, resulting in a necessity to fix collision scripts. We continued on with a third inspection which resulted in a failure for a few cases, primarily by the health packs that were implemented. Currently in need of further inspection and work to fix issues.

ID# - PlayerGun

Inspected by: Michal Fitzryk, Tim Jang, Brian Yu

Date of Inspection(s): February 20th, March 27th, April 3rd

Results: Player was able to fire their weapon properly after the first inspection without issues. After changing the player sprite, a second inspection was done, when we discovered an issue with the player model colliding with the bullet, which we were able to solve quickly. We then did a third inspection after changing the weapon's shooting mechanics, which passed as it worked properly.

ID# - ZombieMovement

Inspected by: Michal Fitzryk, Tim Villaraza, Brian Yu

Date of Inspection(s): March 27th, April 3rd

Results: During the first inspection, we found that zombie movement did work for the most part as intended, however it required further inspection due to zombies being unable to path past certain obstacles. The second inspection did not yield further improvements to the script, primarily due to collision of other objects breaking the pathfinding script.

ID# - InGameMenu_Behavior

Inspected by: Tim Jang, Tim Villaraza, Brian Yu

Date of Inspection(s): February 20th

Results: The in-game menu was able to work with the resume and quit buttons that worked properly during its first inspection for its functionality. The options menu functionality also worked favorably and does not require further reinspection.

V Reccomendations and Conclusions

Some of our items that we have covered in this document have passed both testing and inspection processes, albeit with some minor issues that may have occurred, which may require further testing and inspection. Some of the items have also not passed testing and inspection, and thus require further analysis and problem solving to fix these issues which we were unable to solve due to time constraints and some unforeseen circumstances. Also, we have some additional features that we also wanted to implement but as with what has been said previously, we did not have the opportunity to implement them. With additional time, we can continue with fixing the current issues that we have with further actions to improve upon our product.

VI Project Issues

15 Open Issues

We still have the issue where certain parts of the map make only the top half of the environment objects (trees, etc) appear on the screen. Not all of the collisions are working properly. This is mainly true for the maps made in other files. Player also gets stuck way too easily, even if the gap between objects visually appears to be big enough for players to pass through. It's also an issue where the player can enter the building and not be able to come out of it. The other map 2 levels are also not done being decorated. The maps in other files are having collision issues with the environment and the player.

16 Waiting Room

For future releases, we would like to make a larger variety of zombies, all with their own unique abilities (not just meleeing). Some of these abilities include shooting acid, throw ability, spawning smaller zombies, etc. We would also like to add a boss level where the player has to fight the boss to win the game. We would also plan to have better animations and skins for both the zombies and the player. And, if possible, add a multiplayer feature so that users are able to play with friends (though multiplayer gameplay will be made different from single player gameplay).

17 Ideas for Solutions

Issues involving any hardware devices can not be solved, however other bugs and glitches involving software can be fixed through game updates. This can include any gameplay mechanics that have issues or can be changed to make it more entertaining for the player. Devices that have turned off during gameplay due to shutdowns of any sort can not be solved, but we can implement a method of player progression being saved or having some sort of level selection in a menu. Users that are not entertained by the gameplay are also a potential issue, but we can continue to add features or gameplay elements that may potentially improve their experience.

18 Project Retrospective

What went well:

Our group had great communication with each other, being actively involved in project development for the game. Everyone was present during meetings at all times and we were able to complete our assigned tasks on time. Our group was also able to support each other through managing our values that we learned about in software development, using respect and other important fundamentals that encourage a friendly and productive environment. Our use of Jira was also much better this time around in comparison to last semester by using planning poker which better scaled task completion time.

What didn’t go well:

Towards the end of the semester, one of our members suffered an injury, leading to the third release being slightly underwhelming as much of their work had to be worked around due to this unfortunate event. The initial project description was also slightly vague, resulting in unclear ideas and objectives for the game’s functionality and focus. Furthermore, time constraints due to other classes also did not benefit in terms of the game’s production, leading to us to incorporate more slack than we initially anticipated.

19 Techniques and Methodologies

Our group employed many software development practices that are commonly used nowadays in the industry. We developed through the use of pair programming, often working together on one specific mechanic or during integration to ensure everyone understood what each person has accomplished while learning from one another on their steps to code something that was used within the game.

We planned our releases with sprints, scaling them through the use of planning poker to measure the amount of time require to finish the development process of a certain task. For our planning poker, we scaled it on a Fibonacci scale, with values 1, 2, 3, and 5 being the values we used to calculate the length of a task. For larger tasks, we broke them into smaller ones if possible, making them more manageable.

We also used the Pomodoro Technique, which is a time management system that allowed us to used the time we had to work with in the most efficient and productive way. For us, we worked on tasks in 25 minute increments separated by 5 minute breaks, and after 3 cycles, we took a longer break. We found that doing work in this fashion has helped us be much more efficient and productive in our time management and helped us a great deal overall.

Furthermore, we incorporated the values of XP, Extreme Programming, into our work. The most important values that we used consistently throughout the project workcycle includes communication, respect, and feedback, but we have used all of the others to some degree at one point in time, such as simplicity and courage.

20 Tools Employed

Our project was made using the Unity Game Engine. Unity uses the C# programming language which we coded all of our scripts in. We used the development environment of Visual Studio, which comes bundled in with Unity and contains multiple features that are useful for developing. We used Github as our main tool of version control. Github allowed us to work on multiple parts of the project simultaneously without overwriting each other’s codes. In particular, we did this by using Github Desktop and created separate branches which we wrote to. After committing to a branch, we then reviewed the code in each branch before deciding to pull the branch into master. Finally, we also used Jira as our main tool for planning our sprints and releases.

VII Glossary

Level: Each of a series of stages of increasing difficulty through which a player may progress, completing one stage in order to reach the next.

Health Pack: Items that the player can pick up to restore their health bar. The game contains two different health packs, including a small and a large one that restores differing amounts of health.

Bullet: Projectiles that are fired out of the player's gun that deal damage to enemies in the scene.

Zombie: Enemies that are featured in the game. Our game spawns three different zombie sprites that the player can defeat.

VIII References / Bibliography

[1] Robertson and Robertson, Mastering the Requirements Process.

[2] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.

[3] J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.

[4] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.

[5] D. Hatten, M. Torres, M. Ryan, N. Gelezinis, CS 440: Project Summary Zombie Dash, 2019.

IX Index

No index entries found.