Wondercraft: Final Report

Mitchell Blanchard: 101000294, Nathan Marshall: 101010064, Megan Perera: 100998562

Section I: Overview

We wanted to create a game in which the player could play as a wizard, explore an interesting world, and discover different spells by crafting items they found within different levels, which they can use to defeat the enemies which are plaguing the land. The player controls their character with the arrow keys. Left and right move the character in their respective direction, up to jump, and down to pick up an item (when standing by it). The space bar casts a spell, which will cause damage to enemies. The player's inventory can be opened and closed with "I". Navigating within the inventory is done with arrow keys, and use space to select an item and space again to move it to the desired location, for example, the crafting or equipment menu. Aesthetically, we aimed to create a vibrant world full of color, and make interesting and aesthetically pleasing protagonist, enemies, and areas to traverse.

Section II: Post-Mortem

Our original plans included more worlds, more enemies, and more content. We wanted to create six platforming levels for the player to explore, the first being a simple tutorial level, teaching the basic mechanics, the rest allowing the player the experiment with spells and crafting as well as explore each level. We aimed to design and implement five enemies, and we wanted to not just have different spells, but different spell patterns and differing spell types (i.e. not all would be offensive, some would be for easier travel or defensive). Finally, we wanted to add small additional changes of pace to the game; we wanted to include mini puzzles for the player to solve and a boss battle at the end for a change of pace and challenge in the gameplay.

We successfully created a platforming game with sophisticated collision detection. We also created a crafting system that works as following: a player finds stones which can be combined to create gems. Gems can be combined with equipment to create new hats, staffs, and robes. Staffs will change the player's current spell, while hats and robes will impact their stats, such as health, speed, etc. We created three unique enemies, and four levels for the player to explore. However, we were not able to implement all the ideas we had planned. Our miniature puzzles and boss battle did not make it into the game. There were two levels and two enemies that were cut from our game.

Ultimately, the majority of the content we did not implement we did not due because time restraints. Creating the collision detection alone took weeks of planning, time, and testing. Having to balance the creation of a game with four other intensive classes proves incredibly difficult, leading us to cut content we felt was not entirely necessary to the core experience.

If we were to do the project again under the same working conditions, we feel as though we would stick more strictly to the schedule we set out for ourselves. We also feel as though we would scale down our expectations to be more realistic given the circumstances.

Section III: Bugfixes and Efficiency Upgrades

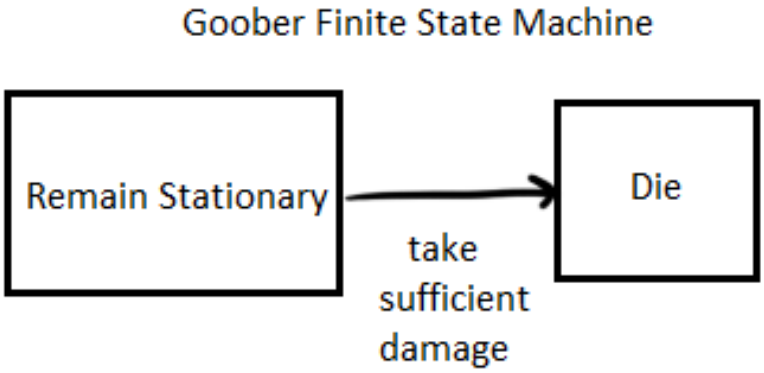
One major bug fix was fixing our inventory system. While we were working on our game for feature complete, we tested our crafting to see if it works correctly, and it did. However, to properly access the crafting menu in game, the inventory needed to be fixed, as it was incredibly buggy.

Attempting to move any items around the menu would cause them to multiply and spawn in random locations within the menu. To fix this, we just needed to review and clean our code for the inventory system a little.

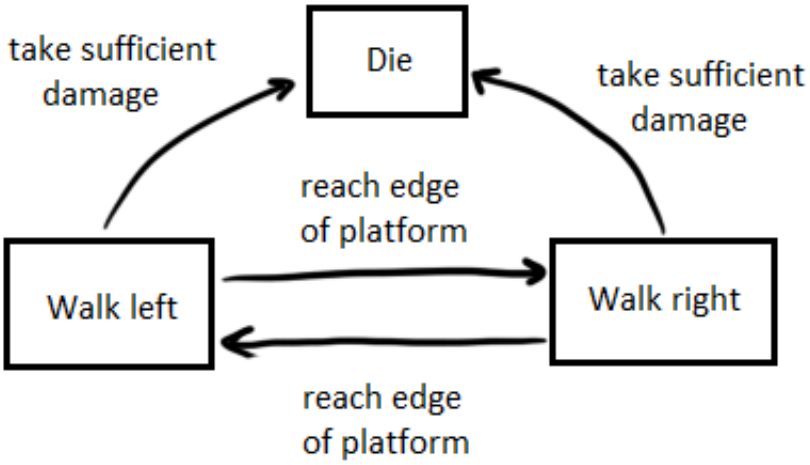
An upgrade we made was the implementation of a vertex array for our tiles. Initially, we stored the tiles within a 2D array, which we knew we would need to change later. Of course, we switched it for efficiency's sake.

Section IV: Artificial Intelligence

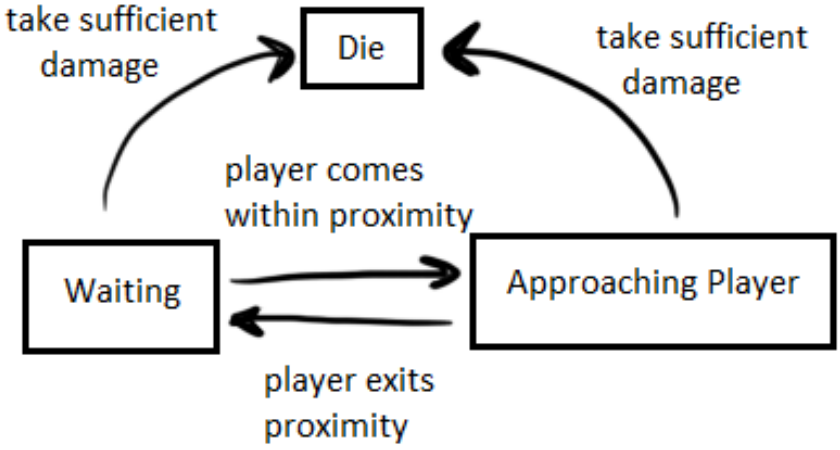
Our game includes three different enemies: a Goober, which is a stationary enemy, a Gremlin, which scouts out the platform it stands on, and a Ghostie, which is an enemy which will chase the player when they come within a certain proximity of them. Below are the finite state machines detailing their behaviours.



Gremlin Finite State Machine



Ghostie Finite State Machine



Appendix A: Milestone Calendar

Milestone Calendar

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Week 2 Tutorial 1 of 8 (Visual Studio, SFML)	8	9	10	11	12	13	14
Week 3 Tutorial 2 of 8 (Introduction to C++)	15	16	17	18 Group formed	19	20	21 Met to crafting and different spell ideas. Game idea submitted
Week 4 Tutorial 3 of 8	22	23	24	25	26	27	28
Week 5 Tutorial 4 of 8	29	30	31	1	2	3 Met to Discuss Game and Make Design Doc	4 Design Document finished and Submitted
Week 6 Tutorial 5 of 8	5 Backgrounds and gremlin sprite completed	6 Protagonist sprite done	7 Ghostie sprite done	8	9	10	11 Structure for Game Set up
Week 7 Tutorial 6 of 8	12 More enemy sprites completed	13	14	15	16	17	18 Projectiles implemented
Week 8 Winter Break	19	20	21	22	23	24	25
Week 9 Tutorial 7 of 8	26	27	28	1	2 Tile Parser for levels done	3 Goobers, Collision Detection	4 Vertical Slice

Week 10 Tutorial 8 of 8	5	6 More levels designed	7 Collision Detection improved greatly	8	9	10	11
Week 11 Project Help	12	13 Sprites Resized (for collision detection purposes)	14 Player sprite broken up into 3 pieces (for drawing them with different sets of clothing)	15 New levels added to game	16 Menu Base done, crafting implemented	17 Menu Implemented (Buggy)	18 Feature Complete (Ghosties and Gremlins implemented, other spells implemented)
Week 12 Project Demos	19	20	21	22	23 Menu Fixed and Functional	24	25 General Code Cleanup
Week 13 Project Demos	26	27 Presentation in Tutorial	28	29	30	31 VertexArray implemented for tiles	1 Final Product (and Report)

Appendix B: Self/Peer/Group Assessment

All our group members contributed to the creation of Wondercraft. Megan did all the art assets except for two which Nathan made. Nathan lead the work on collision detection, and ultimately, all members of the group contributed significantly to all parts of the game. We all put in a tremendous amount of time and effort into the creation of this project. We were most effective when we met up and coded together. There were two kinds of ways we would join up, either in via Skype call or sitting and coding together at someone's house. Without a doubt, when we were physically together we were incredibly efficient and diligent. This was the most effective approach to working on this project. The Skype calls were also quite productive, but not as productive.