CS467 - Capstone Project
Winter 2018

Team Centaurus Final Report
“Not In Kansas Anymore” Unity Game

“Centaurus” Group members
Alejandro Romero
James Whiteley IV
Chewie Lin

Introduction:

Over the course of this term our teams goal was to create a unity based 3D wave survival shooter game. We wanted to create a unique game with our own whacky charm to it. Prior to getting started we had all kinds of ideas and when we saw the number of resources out there we were excited to create an awesome game. Along the way we ran in to some unexpected challenges and realized we would have to curve our expectations a bit, but it was still a great learning experience and we were still able to create a fun and unique game.

One of the first challenges we came across was finding the best way to collaborate on the same game files, we decided to not go with Github but instead use unity's built in collaboration tool. We also ran into difficulty with consistency of assets and models and decided go with a low poly look for all of our assets. Aside from those issues, most of the other issues encountered were code and bug related which we eventually overcame.

From a user perspective, the final version of our project is essentially a computer based video game which requires some level of strategic thinking. The user plays the role of a war veteran who finds himself mysteriously transported to some strange land with hostile creatures. Each levels environment is randomly generated so that there is a unique challenge each playthrough. The user must defeat a certain number of enemies within a wave. Each wave gets progressively harder and has a chance of introducing some new enemies. After defeating a certain number of waves the user progresses to a new level which offers a unique environment. Throughout the game there is also a chance for a unique perk or power up to appear which gives the user helpful new abilities to defeat his foes. If 3 levels can successfully be beaten the user will win the game, otherwise the game will end when the player runs out of health and dies, or when he chooses to quit from the menu. Upon death the user has a chance to restart the level. We assume the target user already has experience with 3D video games and do

not expect them to have any major issues, however setup and gameplay instructions are provided in the following section.

Fig. 1-1 **Player surviving a hungry pack of wolves.**

Fig. 1-2 **This is going to hurt.**

Setup & Usage:

Starting the game

To begin the game, open the NotInKansasAnymore.exe file provided in the .zip file submission. All files used by the .exe are in the same folder and need to remain in the same folder to be used properly by the game. This will bring up the configuration menu. Simply choose your desired resolution and graphics quality (We recommend Ultra) and hit play!

Additionally, If you have Unity installed, you can play it through Unity with the provided projectfolder. To do this, open Unity and then select the provided NotInKansasAnymoreGame project folder. If you are not on the start menu, then go to the Scenes folder within Assets folder and select the Level0 scene. Once you are on the start menu scene, click on the play button to begin.

Controls

Movement (up, down, left, right): ‘w’, ‘s’, ‘a’, ‘d’ keys
Movement (Turning Player): moving mouse over screen
Fire: left mouse click
Toggle first/third person view: ‘t’ key
Pause menu: ‘p’ key
Menu navigation: mouse movement and left mouse click

Step by step guide

Simple Step-by-Step How to play Guide.
1. Run the NotInKansasAnymore.exe file. 2. Select “Play!” on the configuration menu. 3. On the main menu click “Play”. 4. After reading the intro click “Start”. 5. Survive waves by running around (**‘**w’, ‘s’, ‘a’, ‘d’ keys) avoiding enemies and shooting (left mouse button) at them from a distance. Press “T” to toggle first person mode if you prefer. 6. If you see any perks (Blue Stars) go run over them for a nice bonus. 7. After surviving 5 waves you will be taken to intro screen of next level. Click “Start” to begin next level. 8. Survive 5 more harder waves to go to next level. 9. After defeating all enemies in all waves you will be taken to the Win Screen. Congratulations you won the game.

Navigating game

Main menu: The main menu is very simple and provides an option to quit (which exits the program) and play (which leads to the intro of the game).It also allows the player to select which level they wish to start at in case they left off at a later point and wish to continue. See fig. 2-1 below for image.

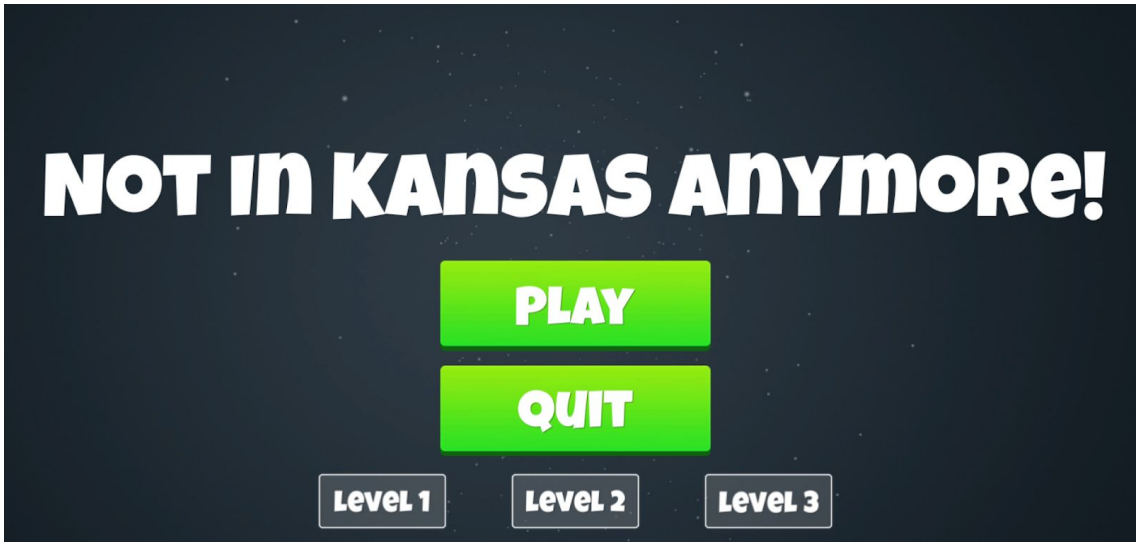

Fig. 2-1 Main Menu

Pause menu: This provides three options: Resume, Restart, and Quit. Clicking Resume or pressing the 'p' key will unpause the game. Restart will restart the current level and reset the wave back to 1. Quit will take you to the main menu and you will lose all current progress.

Game over menu: If your health reaches 0 from the enemies harming the player character, you will die and a game over menu will appear with two options: Restart Level and Quit. Restart Level will reset the wave back to 1 and start you again on the current level. Quit will take you to the main menu and you will lose all current progress.

Game Objective: There are a total of 3 levels, each level containing 5 waves of enemies. Your objective is to kill all the enemies in each wave of each level. There are perks that randomly appear to help you complete your objective that appear as a giant blue star (see fig 2-2). These perks include instant kill, rapid fire, full health, and increased movement speed. When a perk star appears you only have 15 seconds to grab it before it disappears, so you better keep your eyes peeled!

Fig 2-2 Perk Star, First Person View

Fig. 2-3 Third Person Gameplay

Win Screen: (*Spoiler Alert*) Upon beating the game the player will reach the win screen, which will give them the option to play again or quit the game.

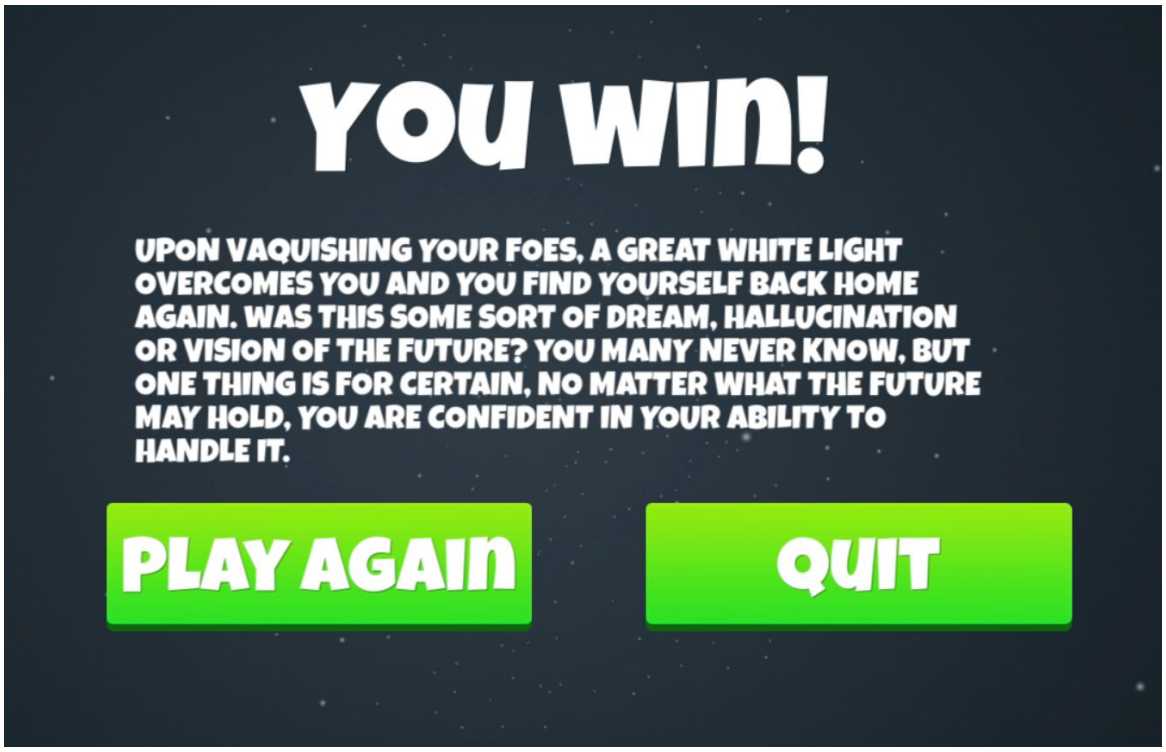

Fig. 2-4 Win Screen

Software System - Unity:

This project used Unity exclusively as its software system. Unity is a game development engine that allows for the creation of both 2D and 3D games. Frequently used game objects in the form of prefabs contain all the necessary resources for that object. For example the Hobgoblin enemy prefab for our game consists of a model and textures, transform for positioning, rigidbody for physics, colliders to give it a presence and interaction, audio source, animator, nav mesh agent for navigating the terrain and various scripts. Each of these has individual values that come together to create the enemy. A scene editor window with its hierarchy allows you to add components like canvases for menus and prefabs while a game window allows you to test play. Unity comes with a handy build feature that allows you to add your games scenes/levels and build a game for multiple systems. In our case PC. We also used some of Unitys more complex features such as the animator window in order to control the various animations of our prefabs, such as walking, attacking and dying.

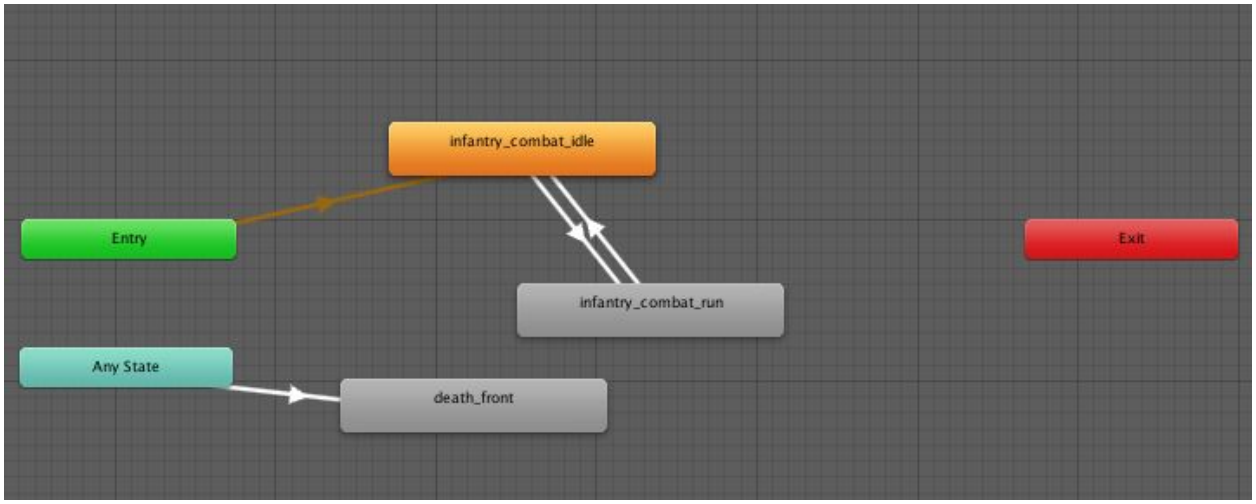

Fig. 3-1 Player Animation Example

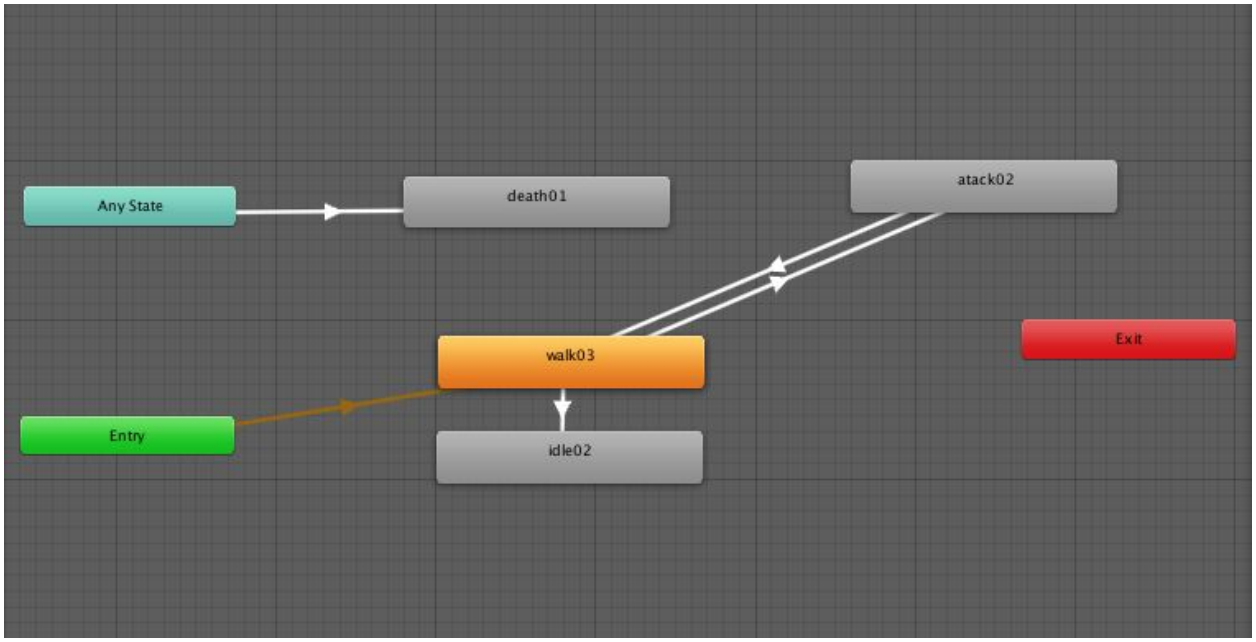

Fig. 3-2 Enemy Animation Example

Additional Tools, APIs, Libraries, etc:

As mentioned above, we used Unity 3D to accomplish the entire project. By default unity uses MonoDevelop IDE, which in turn uses C# language. All our scripts were coded using MonoDevelop. Since most of us had no experience with Unity before this project we greatly relied on the Unity Documentation. We also do not have graphical

design experience so we used free assets from the Unity Asset Store in order to build the level terrain, create enemies, and create our player character.

Teammate Contributions:

James Whiteley IV

James was responsible for creating the main game manager script which sets waves/levels and tracks various aspects of the game, the playable character in the game including all animations (shooting, walking, and dying) and scripts (movement, health, shooting), the pause and game over menus with scripts to control them and toggle pausing the game, perk implementations with necessary scripts (instant kill, increased speed, full health, rapid fire), and the first person view controller script.

Alejandro Romero

Alejandro was responsible for creating the various type of enemy prefabs (finding model assets,creating and setting up animations, creating the various stats such as speed,health and attack, managing sounds and setting up movement via nav agent writing enemy scripts etc..) coming up with and creating a story and music theme via various in game menus, and managing enemy waves which included the code for enemies left counter and timer between wave spawns.

Chewie Lin

In the first half of the project, most of the work was focused on adding feature and debugging until it is in a usable state. The first challenge was to implement the code that can generate the game mesh boundary and fix all the mesh related bugs[3]. The second was to generate the terrain needed for the game and for Unity AI system[4]. This requires a new set of runtime APIs[5] that can be used to bake the terrain at the start of the level and fix dependencies errors with the AI navigation agents.

Most of the second half was to fix problem related to the script that interferes with the enemy spawn points, UI elements, and any problem that arose from other team members. Such as camera, physics related bugs, and terrain baking. Debugging problems took at away at least a week of time after each large feature upgrade. Once level 1 was completed, level 2 and level 3 were much easier to replicate.

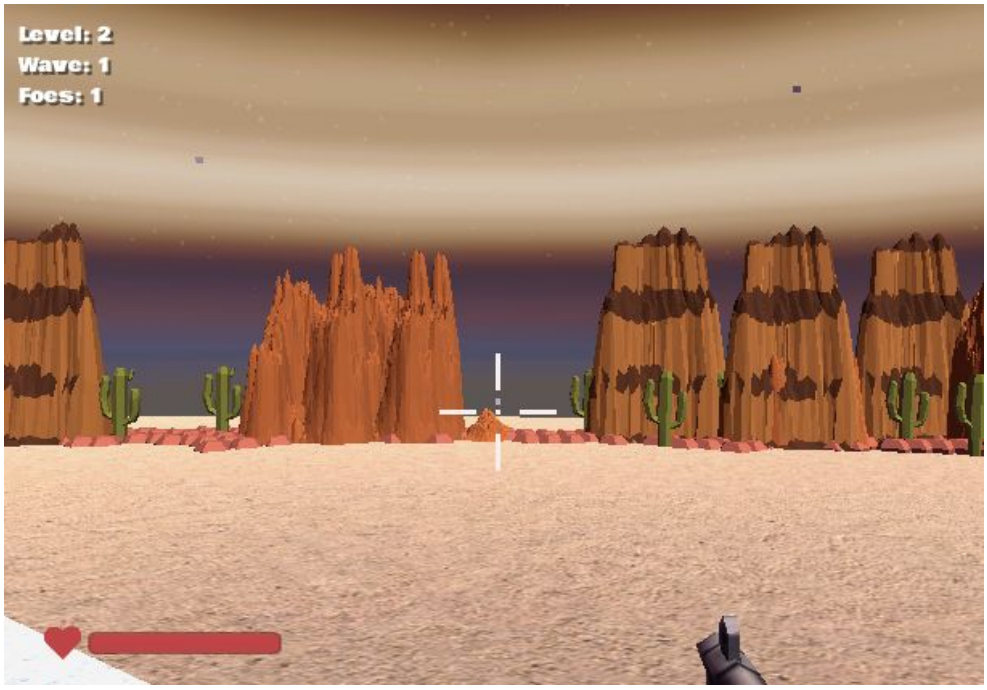

Fig. 4-1 Sample Desert Level

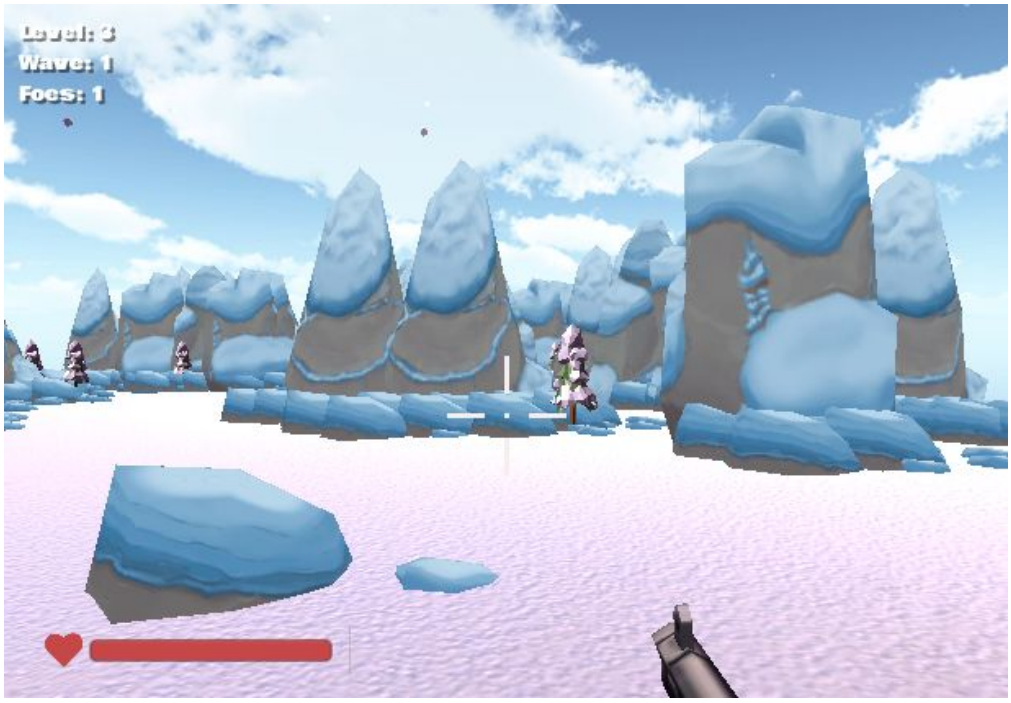

Fig. 4-2 Sample Snow Level

Deviations from Original Plan:

Our original plan included having a boss at the end of each level, but due to our lack of experience with 3D animation, modeling and advanced combat scripting, as well a lack of time we had to nix the idea from the game.

The other problem related to the original plan was the change in the class diagram call sequence. Instead of calling GameManager() at the start of the game, this will be done after the environment has been generated.

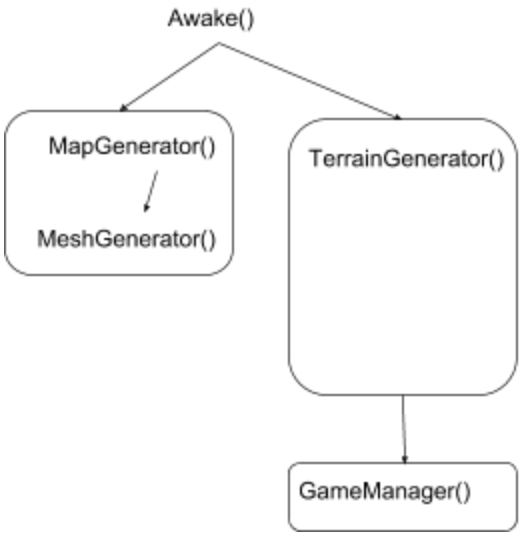

Fig. 5-1

Conclusion:

Overall this project was a great learning experience. Our team gained a lot of knowledge about the different aspects that go into creating a game, as well as a deeper appreciation of game designing as a whole. We were surprised to see actually how crucial it is to have an artist/designers when creating a game. We relied on a lot of Unity's free assets, but that greatly limited what we could achieve. We also would have liked to accomplish much more, but due to our small window of time as well as the factors mentioned above we were unable to fully implement all our original ideas. We are still happy however with what we have achieved. Each of us have acquired a great base level knowledge of 3D game creation as well as using Unity tools. We also enhanced our knowledge of C# which has very similar syntax to C++. The whole process was very enjoyable and we hope you have lots of fun playing our game!

References:

[1]
https://unity3d.com/learn/tutorials/modules/beginner/live-training-archive/creating-a-scene-menu?playlist=17111
[2]
https://unity3d.com/learn/tutorials/topics/user-interface-ui/introduction?playlist=17111
[3]
https://unity3d.com/learn/tutorials/projects/procedural-cave-generation-tutorial/cellular-automata?playlist=17153
[4]
http://www.heparo.com/img/portfolio/terraintlkt/documentation/terraintoolkit_manual.html
[5]
https://github.com/Unity-Technologies/NavMeshComponents