## Component Design and Integration Plan

1. **Come up with a game pitch using the components we have created and their functionality.**

*Outline the game pitch for the game you are going to design and create using the components made in class/with tutorials.*

Zombie Holdout is a classic arena shooter type game with a top down perspective and focuses on fast paced combat. It has an infinite number of spawning enemies, with the time between each spawn decreasing after each spawn until it reaches 0 and goes into a sort of horde mode where the enemies never stop spawning and your only goal is to survive the longest and kill the most. The character controller made in class can be modified for use in a top down environment, removing the jump functionality and having the player rotate with the cursor. To give the player an incentive to take risks in order reduce the pressure from the large number of enemies, there will be doors that can block off the spawn point of the enemies for a length of time, reduce the pressure in the long run, but posing an increased risk to the player as they have to get closer to where the enemies come from. This fulfils the player to level interaction by having a way for the player to directly affect the game environment. The player will have 3 lives, losing one whenever they come in contact with a enemy, ending the game if all 3 lives are lost and saving the points that you have earned by killing enemies as your high score and how long you survived as your best time, so the player has the incentive to beat their high score.

1. **Analyse and evaluate the required functionality of the components in order to achieve the gameplay and functionality required by the design of your game pitch.**

*Document the required functionality of your components. Identify whether they already contain appropriate functionality or whether they will need to have their functionality expanded. If you need to expand the functionality of the components beyond their current functionality, outline how and why you need to do this. Outline how you will use/configure the components in order to achieve the gameplay and functionality of your game pitch’s design.*

The character controller already has most of the functionality that is required for it to fit with the game pitch, but it will need to have the jumping functionality removed to keep with the top down perspective and keeping the player more confined, and add a system that rotates the player to follow the mouse position. I can simply remove the code that references jumping and add some code that uses the co-ordinates of the mouse within the cameras view to translate into a rotation vector for the player object. I will need to add a health system and a score system to the player object that will be affected by objects other than the player. Health is simply two float values of current health and max health and will be influenced with an “OnCollisionEnter” function with an if statement to check if the collider the player has entered has the tag of enemy, taking from the current health value if it is an enemy. A score system would be float value within the player objects script that gets accessed and influenced by the enemy’s script, where a value for how many points should be given after the player successfully kills an enemy. Bullet components would consist of a game object prefab with a script that contains the damage that should be dealt upon hitting an enemy and a declaration that destroys the game object upon hitting any collider within the game scene. A gate interaction and nav mesh obstacle would be required to fulfil the requirements for the game, the gate can simply be two cubes animated to scale up and down upon the players interaction with it, meanwhile, the interaction would have be included in the players scripting, conducting a ray cast upon the press of a key, with the ray cast returning whether it had hit the doors trigger collider or not, toggling the animation for the gate on a successful true returned value. This gives the player a game object that is directly influenced by their actions. Its essential that the enemies are always a constant threat, which means that they have to actively hunt down the player, using what we learned about Nav Meshes and Nav Mesh Agents in class, I can place a nav mesh agent onto a game object it chase the player while avoiding walls, making an enemy that takes into account their environment while going after the player. Using what I learned about HUD elements during class, I can adapt the pre-existing HUD we made as a tutorial, to be used in this project. The HUD will need to have a panel with a visual representation of stamina added, the players current score, high score, current time and best time also need to be displayed with the ability to update their values when needed. The pre-existing components used where, player movement, HUD, player health, bullets, enemy pathfinding, text file manager, log reader, gate trigger, most of the player functionalities and the timer trigger. Of those, player movement, player health, HUD, Camera setup, enemy pathfinding and the gate trigger where originally created during TAFE session with the movement’s functionality being extended to include a sprinting and stamina function while dropping the jumping functionality, health functionality was changed slightly by setting the max health to 3 and having the enemies take a very specific amount of health to reduce the amount of work I had to put into the heart UI, the HUD being extended greatly by adding score elements and stamina representation, enemy pathfinding and nav meshing was briefly taught during a session and from there I added nav mesh obstacles that the enemies try to path through but are unable to, the gate trigger started off as a simple door that only opened if you had a key, that was changed to no longer need the key.

On the other hand, the bullet was taken from Sykoo’s top down shooter tutorial and adjusted so that the bullet game object would be destroyed no matter what collider it enters, the text file manager, log reader and timer remained greatly unchanged from the one that was used in Immergo Media’s game making tutorial aside from the timer being changed so that the longest time was recorded instead of the shortest, the Camera was made mostly from the Sykoo’s tutorial with very little changes to it. the functionalities of the player were taken from Sykoo’s, Brackey’s and Immergo’s tutorials with much of the movement and dying functionality being extended by adding sprinting, stamina and the addition of a ‘death screen’ but other parts like the Shooting and getting the player object to face the mouse stayed completely the same.

1. **Determine which components require dynamic functionality.**

*Explain which components need to be created in a way that they can be dynamically used or applied in various contexts to achieve a range of functionality.*

The health system will need to be designed in a way that it can be used for both the player and the enemy, allowing a unified system to handle health of all of the entities within the scene. Although they will essentially be exact duplicates, the entirety of the enemies and their attached components will need to be designed so that they will be able to individually and accurately find a path towards the player and have their own individual health values. The same sort of setup is required for the gates, having independent intractability but acting functionally the same as the others.

1. **Consider the technical impact of the component’s functionality/design on the overall project.**

*Explain how the components integrate with other components in order to achieve the functionality required by the overall design of the project. How have you assembled game objects in the game engine to ensure that each component has access to the other components and game objects it requires access to?*

The player script is the most integrated script within my solo game, taking functions, variables and components from nearly every other script, the health, stamina and score related HUD UI elements from the UI script, the mouse position in relation to the camera and scene, uses functions from the timer script to control the timer within the scene, the enemies script uses the players collider to reduce the players health and the position of the connected bullet spawn point and its rotation for the place to instantiate the bullet prefab and the direction it should travel in. The gate components use the player components to function and operate, the enemies get the player component upon being instantiated to set a target that the enemy will head towards following the shortest path over the nav-mesh. Variables and functions within the player script that are accessed by other scripts are given a public security level to prevent issues occurring when trying to access these variables and functions from within a different class.