Assignment 06: In-Game-Systems

Team Number: 02

Team Name: Team 02

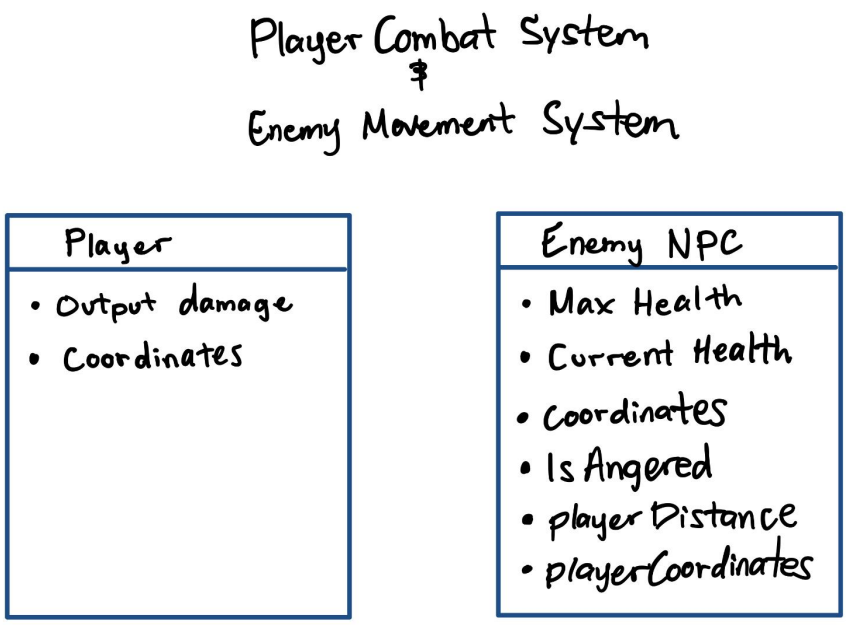
Team Members: Winston Pham
Norberto Gomez Rosales
Daniel Gonzalez

Game Name: Odyssey.

System Description: The in-game system being worked on is the player combat system and enemy movement system. The purpose of the player combat system is for the player to be able to attack the enemy NPCs by inflicting damage. In addition, the enemy NPC will have set health that will decrease as they are attacked before finally being taken out. The purpose of the enemy movement system is to allow the enemy NPCs to follow and chase the player when the player is in its radius. Initially, the combat system was basic. When the player moved within the enemy's collider and left-clicked, the enemy object was destroyed. The new combat system revolves around both the player and enemy having set health. This set health is depleted as the player and enemy attack each other. Both will have damage values that will later be randomized to limit the player and character from dying from one or two attacks. As stated, the enemy movement system will have the enemy NPCs chasing the player to attack them. Our enemy movement system was created using unity's AI engine library. We implemented specific conditions to check whether the player is within the enemy's radius. The enemy would chase the player as long as that condition is met.

Entities: The entities involved in the player combat system are the player and the enemy NPCs. Those NPCs are red rectangular objects serving as testing dummies and skeleton models we use for combat. The skeleton models have idle and walking animations that are used in-game. The player has a script attached that controls the attack behavior. The player and NPCs have scripts attached to them that control and monitor their health. The enemy movement system's entities include the player, the enemy NPCs, and the script attached to the NPCs that controls the movement for the enemy.

Attributes/Properties/Data:



Algorithms: Beginning with our player combat system, the algorithms needed will be enemy detection and an attack algorithm. The enemy detection algorithm is always invoked as the player is walking around the map. When the player is close enough to trigger the enemy's collider, the attack algorithm can be invoked. The attack algorithm is only invoked when the player is close enough to trigger the enemy. At this point, using the left-mouse click, the player inflicts damage onto the enemy NPC. The attack algorithm uses the player's output damage and the enemy's current health. When the NPC is attacked, its current health is decreased by the player's output damage. Next, the algorithm checks whether the enemy still has health. When the enemy runs out of health, it is destroyed. When the enemy is destroyed, the enemy detection algorithm detects that the player is no longer triggering an NPC's collider.

Next, we have the enemy movement system. The algorithms required are player detection and enemy movement. First, the player detection algorithm will check the distance between the player and the enemy's coordinates. If the player is within a certain distance, the enemy's state is set to 'angered' using a boolean. When the enemy is angered, the enemy movement algorithm will check the player's coordinates and begin to follow those coordinates to chase the player. The player detection algorithm is always invoked while the enemy movement algorithm is invoked when the player is detected. Upon detection, 'isAngered' to 'true'. Then we set an 'isStopped' variable that is an extension from NavMeshAgent function to 'false' which invokes the enemy movement algorithm to make the enemy movable to target the player by setting the destination towards him. Once the player falls outside the radius then it calls another statement that sets 'isAngered' to 'false' and 'isStopped' to 'true' having the enemy stop its chase towards the player.

Video: https://www.youtube.com/watch?v=quaV1mIFjiY