<u>Concept</u>

This is a third person shooter dungeon crawler. The player clears all enemies in a room before proceeding to the next. This continues until the room with the final boss is reached and all enemies in that room are defeated. Instead of a traditional shooter where the player can damage enemies individually, the enemies in each room “share” health. What this means is that when a player shoots and hits an enemy they become “marked” once all enemies in the room are marked then and only then do then die, or disappear.

<u>Algorithmic complexity</u>

There are five categories of scripts for this project which include, scripts for the player, enemies, rooms, camera, and user interface.

<u>Player</u>

The player is moved via rigidbody and the method MovePosition(). The direction the player looks at is controlled by the mouse position. A ray is created between the camera and mouse position by using Camera.main.ScreenPointToRay(Input.mousePosition). A target is then created with the player's original y position and the ray’s x and z position. This is done because the mouse position is only in two dimensions. The player then looks at this target. The dash mechanic is taken care of by the dash() coroutine. The dash can be used every 0.6 seconds. A line renderer is attached to the player to add a visual effect.

The shooting mechanic is taken care of Blaster.cs. This is attached to the blaster object which is a child of the player object and has a child called tip. A ray is created from the tip object’s transform.position to the player’s (parent’s) transform.forward direction. A line renderer is attached to these points to show where it has fired.

<u>Enemies</u>

There are three types of enemies, melee, ranged, and the final boss. Both melee and ranged enemies have their heath taken care of in the Enemies.cs script. This script also tracks whether or not the enemy has been marked. Melee enemies follow the player around the room using RigidBody.MovePosition() in MoveToPlayer.cs. Ranged enemy movement is in MoveRanged.cs. The pointAtPlayer() method makes the enemy slowly rotate towards the player’s position relative to them. maintainHeight() keeps ranged enemies at a constant height in case of collisions. maintainDist() keeps the enemy within its shooting range and moves away if the player comes near them. The boss enemy moves towards the player and shoots them as well. Both the ranged enemy and boss shooting mechanic works similar to Blaster.cs.

Rooms

The enemies for most rooms are spawned with GenMultEnemies.cs. Enemies are spawned at a random location in the room with spawnEnemies(). For the final boss room enemies are spawned with GenBossRoom.cs. This introduces the phase mechanic. There are multiple phases when fighting the boss, this means that the player will have to defeat (mark) all enemies in that room maxPhases number of times.
When the player enters a room for the first time, the enemies spawn, and all the entrances and exits into the room are blocked off, preventing the player from escaping. In order to leave the room, the player must mark all of the enemies with their blaster. Once they do so, the enemies are defeated, and the exits are opened.
As there are multiple pathways the player may follow to reach the end, enemies spawn once a player enters a room, even if they entered from a different entrance than they had before. However, enemies only spawn once per room. If a player enters a room and leaves it upon clearing it, if the player returns to the room, enemies do not spawn again.

UI

The number of health (hearts) the player has is taken care of in HealthManager.cs. The number of hearts is drawn in the bottom left corner of the UI. The maximum health of the player is 5, meaning that they may get hit 5 times before dying, and having to restart the game. The user interface also includes a minimap, to help the player navigate through the narrow hallways that link rooms together.
Also, when the player first starts the game, they are presented with a screen that details the game's controls and the player's objective, as well as the option to begin the game, or exit the game if they do not wish to play.
When the player dies due to losing all their health, or for other reasons, they are greeted with a death screen that asks them if they wish to play again or exit the game. Clicking restart brings the player back to the Start Game user interface, while Quit exits the game.
Lastly, when the player defeats the boss, they are shown a victory screen that congratulates them for beating the boss and winning the game.

Graphics

As we are a group of two, most of the time we spent on this project went into getting the backend logic that controls the player, enemies and their generation, and other mechanics, as well as the level design. As a result, our graphics are all that impressive, consisting of colored cubes that glide across the floor, and lack impressive and fluid animations. However, we still tried to make the game visually appealing, by using free skybox assets and free textures from the Unity Asset Store. All sound effects were also retrieved from the asset store and not created by us.