# *SE2250B - Phase 3*

## Software Feature 1: Enemy Shooting

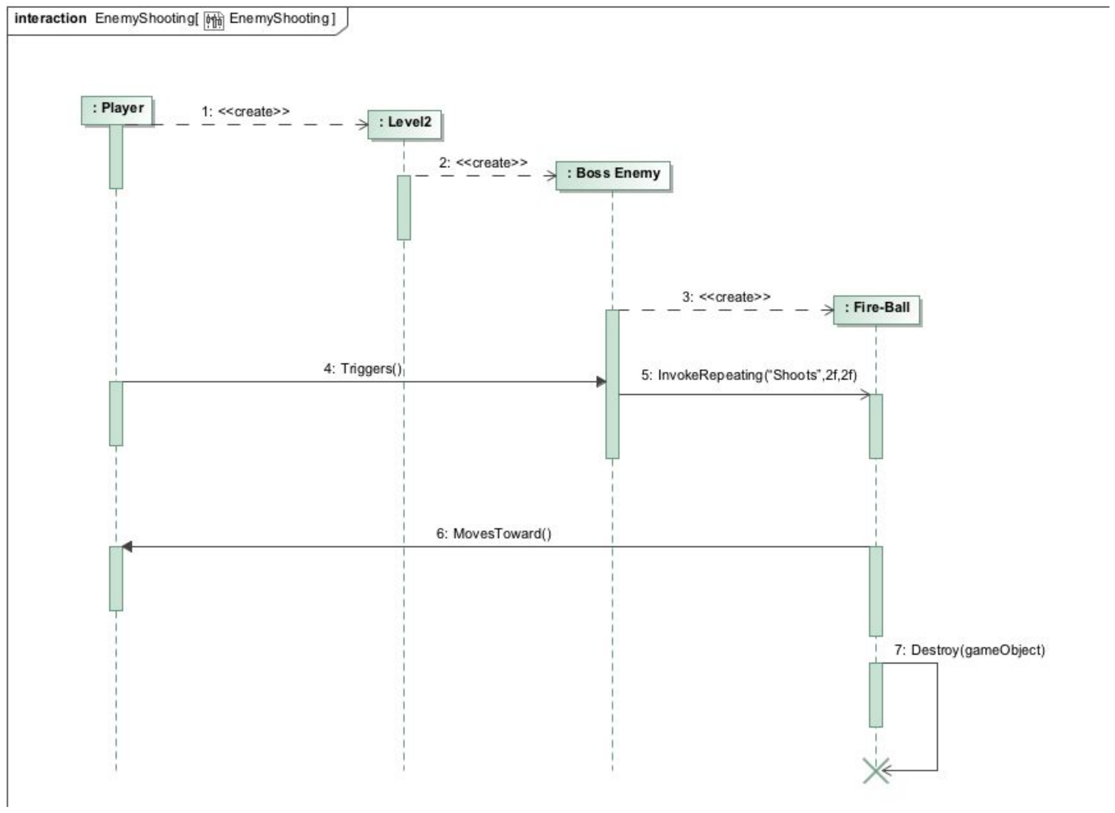
1. *Identify a new software feature. Describe how your feature will behave (requirements):*

- This Software Feature enables the boss enemy to shoot projectiles at the player.
- The boss enemy shoots **fire-ball projectiles** at the player. These projectiles deal more damage than regular enemy attacks.
- Once the player spawns into the 2nd level, the boss enemy locates the player and begins firing using AI pathfinding.
- AI pathfinding works by observing the players x and y positions. Once the enemy detects a change in the player's movement, it rotates itself towards that position and begins firing.

2. *Define the key components (objects) of the new software feature using the object-oriented approach:*

| *Class* | *Description* |
|---|---|
| *FireBallMovement* | FireBallMovement is the class responsible for directing the FireBall projectile towards the player and destroying it upon impact. The "Fireball" projectile is directed towards the player using GameObject.FindWithTag"Player". |
| *EnemyTwo* | This class is responsible for the boss enemy's shooting and movement. EnemyTwo uses AI Pathfinding to detect the player's movement and flips the boss enemy's sprite in that direction. Furthermore, it has a shoot method which instantiates the projectile it shoots, a vector3 constructor, and Quaternion.identity. |

3. *UML Sequence Diagram:*



## Software Feature 2: Guns Switching

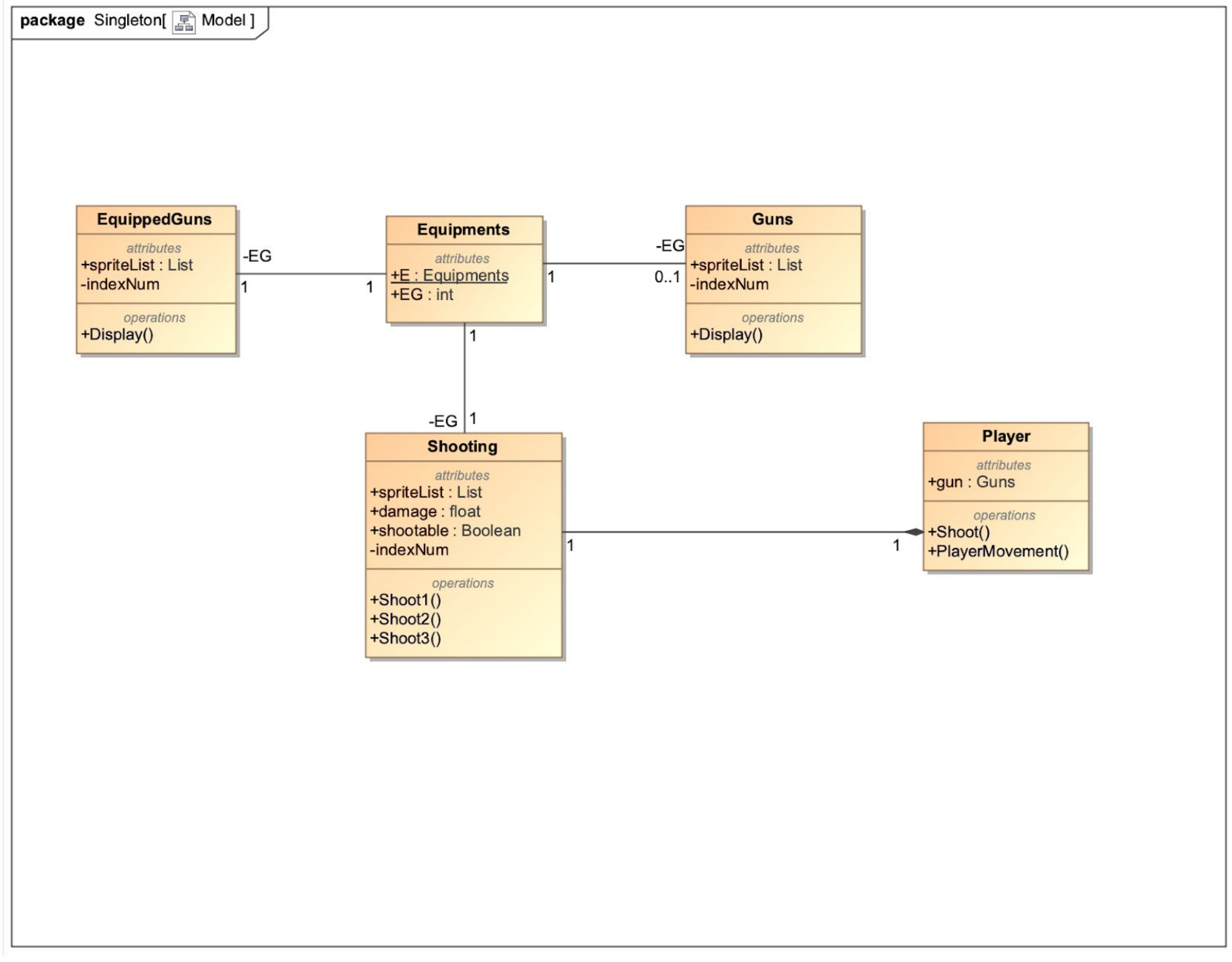
1. *Identify a new software feature. Describe how your feature will behave (requirements):*
    - The guns choice that the player chose can be kept throughout the game.
    - There are three guns for the player to choose from.

- The guns that the player chode have different functionalities(such as shooting multiple bullets at once or generating more damage).
- The sprites have to match correctly in the inventory, equipped item slot and the gun that the player holds.

2. *Define the key components (objects) of the new software feature using the object-oriented approach:*

| *Class* | *Description* |
|---|---|
| *Equipment* | This is a class that applies the principle of Singleton so that all of the classes listed can access the stored that represents the player's choice. (The "EG" in this class is the index of the equipped guns in the list) |
| *EquippedGuns* | This class stores the sprite list of the guns that will be instantiated according to the singleton value. |
| *Guns* | This will be the class that handles the sprite image that represents the gun that the player is holding. It is also changed according to the singleton index value. |
| *Shooting* | The class that will vary the shooting type (single bullet or multiple bullets) according to the singleton index. |

3. *UML Class Diagram:*


package Singleton[ Model ]
EquippedGuns
attributes
+spriteList : List
-indexNum
operations
+Display()
Equipments
attributes
+E : Equipments
+EG : int
Guns
attributes
+spriteList : List
-indexNum
operations
+Display()
Shooting
attributes
+spriteList : List
+damage : float
+shootable : Boolean
-indexNum
operations
+Shoot1()
+Shoot2()
+Shoot3()
Player
attributes
+gun : Guns
operations
+Shoot()
+PlayerMovement()
-EG
1
1
-EG
1
0..1
1
-EG
1
1
1

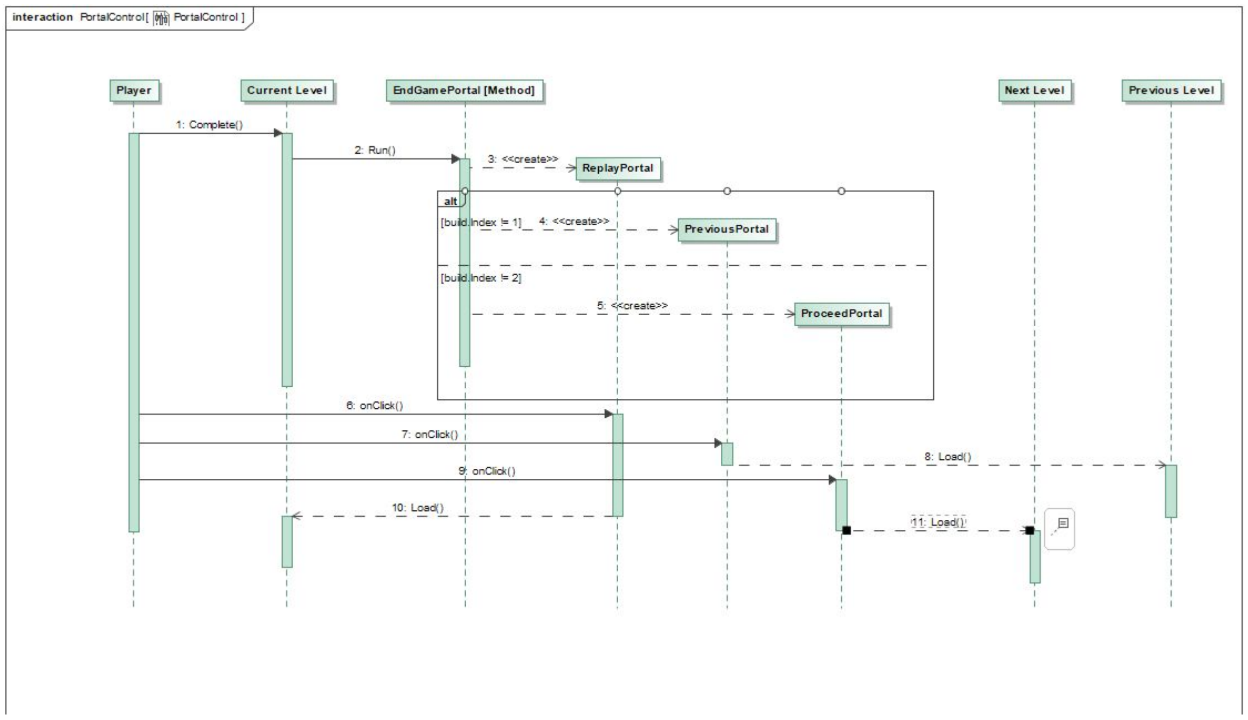
## Software Feature 3: Level Control Portals

1. *Identify a new software feature. Describe how your feature will behave (requirements):*
    - This software feature will control the level selections and scene management.
    - There will be 3 potential portals - Replay, Proceed, and Previous.
    - They will appear at the end of each level, based on which level it is.
    - Replay will replay the level, proceed will load the next level, and previous will load the previous level.
    - The player will decide which portal to use by clicking on it.

2. *Define the key components (objects) of the new software feature using the object-oriented approach:*

| *Class* | *Description* |
|---|---|
| *ReplayPortal* | The ReplayPortal is instantiated at the end of every level. When it is clicked by the player, the ReplayPortal reloads the current scene. |
| *ProceedPortal* | The ProceedPortal is instantiated at the end of level 1. When it is clicked by the player, the ProceedPortal loads the next scene in the build index. |
| *PreviousPortal* | The PreviousPortal is instantiated at the end of level 2. When it is clicked by the player, the PreviousPortal loads the scene that comes before the current scene in the build index. |

## 3. *UML Sequence Diagram:*


interaction PortalControl[ PortalControl ]
Player
Current Level
EndGamePortal [Method]
Next Level
Previous Level
1: Complete()
2: Run()
3: <<create>>
ReplayPortal
alt
[build.Index != 1]
4: <<create>>
PreviousPortal
[build.Index != 2]
5: <<create>>
ProceedPortal
6: onClick()
7: onClick()
8: Load()
9: onClick()
10: Load()
11: Load()