# Feature Documentation - SE2250B Final Project

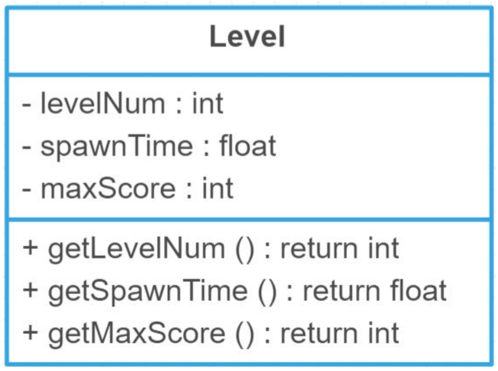
Gabriel St-Germain (250896056), Gary Wu (250908484), Michael Zhou (250921585)

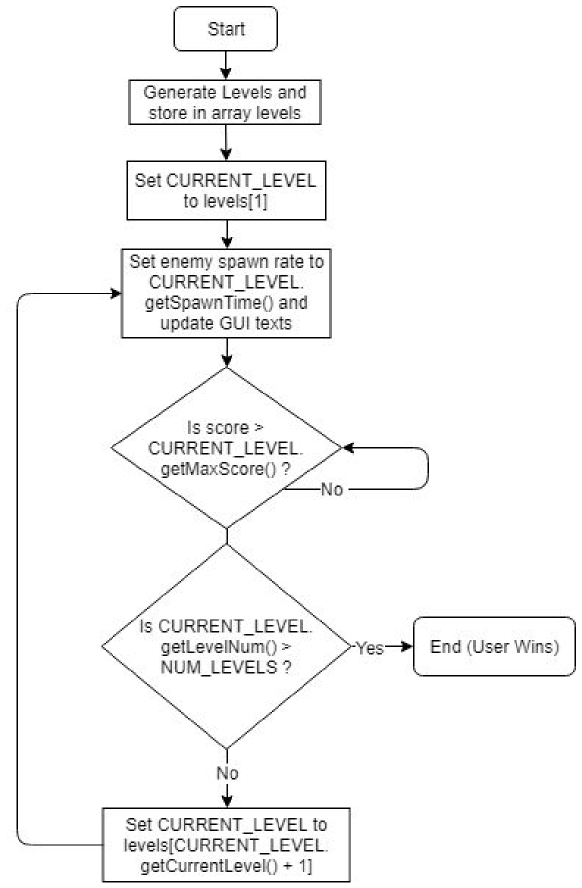
## 1) Level Progression

Description: This feature will allow for a level progression in the game, making it more difficult each level. Levels are based on score, meaning that the player moves on to the next level after reaching a certain score. Each level lowers the time between enemy spawns, meaning that more enemies will come at once. The amount of points to move on to the next level also increases, making it harder to pass higher levels. New features will also be unlocked depending on the level. At level 2, a third type of enemy that shoots will start spawning. At level 3, a new weapon called the double blaster will be available. At level 4, new weapon will also be available called “the destroyer”, which destroys all enemies on the map with a 10-second charging time.

Key Components: A new class will be created to store all the information pertinent to each level. This includes the level number, the score needed to move on to the next level, and the enemy spawn time.
At the start of the game, an algorithm will generate all the levels up to a specified number (100 levels by default). The algorithm will calculate the level parameters using simple relationships:
spawnTime = 4 / (Level Number)
maxScore = 20 * (Level Number)^2
A static variable named CURRENT_LEVEL in the Main script will keep track of the current level, and the Update function will continuously check if the total score is high enough to move on to the next level. Once the score is high enough, CURRENT_LEVEL will be updated and the enemy spawn time will be updated.

Diagrams:


Level
- levelNum : int
- spawnTime : float
- maxScore : int
+ getLevelNum () : return int
+ getSpawnTime () : return float
+ getMaxScore () : return int



Start
Generate Levels and store in array levels
Set CURRENT_LEVEL to levels[1]
Set enemy spawn rate to CURRENT_LEVEL. getSpawnTime() and update GUI texts
Is score > CURRENT_LEVEL. getMaxScore() ?
No
Is CURRENT_LEVEL. getLevelNum() > NUM_LEVELS ?
Yes
End (User Wins)
No
Set CURRENT_LEVEL to levels[CURRENT_LEVEL. getCurrentLevel() + 1]

<u>Test Plan</u>:

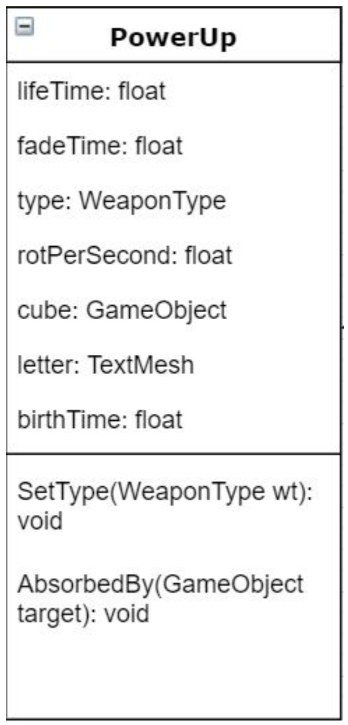
| **Test** | **Passed?** |
|---|---|
| Spawn time decreases each level according to the algorithm | Yes |
| Third enemy type starts spawning after level 3 | Yes |
| When user dies, game restarts to level 1 | Yes |
| Current level is displayed to the screen at all times | Yes |
| A message shows up on the screen when a level is passed | Yes |
| Game shows message when user wins, and restarts | Yes |

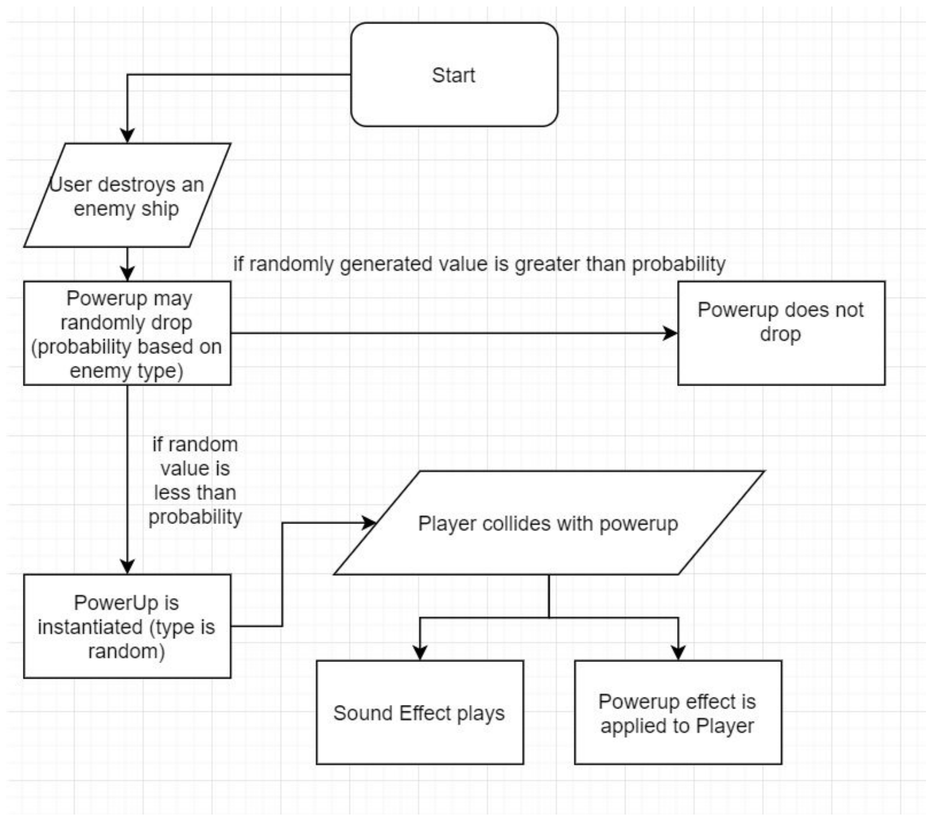
**2) Power-ups**

Description: This feature introduces power-ups to the game, which are upgrades that the player character can acquire after killing enemies. Occasionally, when an enemy ship is destroyed, it will drop an object that is a square tile with a letter in it, which is a powerup. One of the powerups increases the player's movement speed by 10% of it's base movement speed (30). Another powerup increases the player ship's shield level by 1, up to a maximum of 4. The final power up increases the ships rate of fire by 10%.

Key Components: A new class will be created to store all the information and functions necessary for the powerups. This includes:
lifeTime (number of seconds that the powerup will exist after being dropped)
fadeTime (number of seconds it will take to fade away)
type of powerup.
Vectors to determine its rotation and drift
WeaponType type which stores the powerup's type (shield or speed)
rotPerSecond determines how many rotations the cube powerup will make per second
Cube GameObject and letter TextMesh that will comprise the visual element of the powerup during gameplay
SetType(WeaponType wt) randomly determines the type of the powerup upon instantiation
AbsorbedBy(GameObject target) destroys the powerup when it is picked up by the player

Diagrams:




Start
User destroys an enemy ship
if randomly generated value is greater than probability
Powerup may randomly drop (probability based on enemy type)
Powerup does not drop
if random value is less than probability
Player collides with powerup
PowerUp is instantiated (type is random)
Sound Effect plays
Powerup effect is applied to Player

Test Plan:

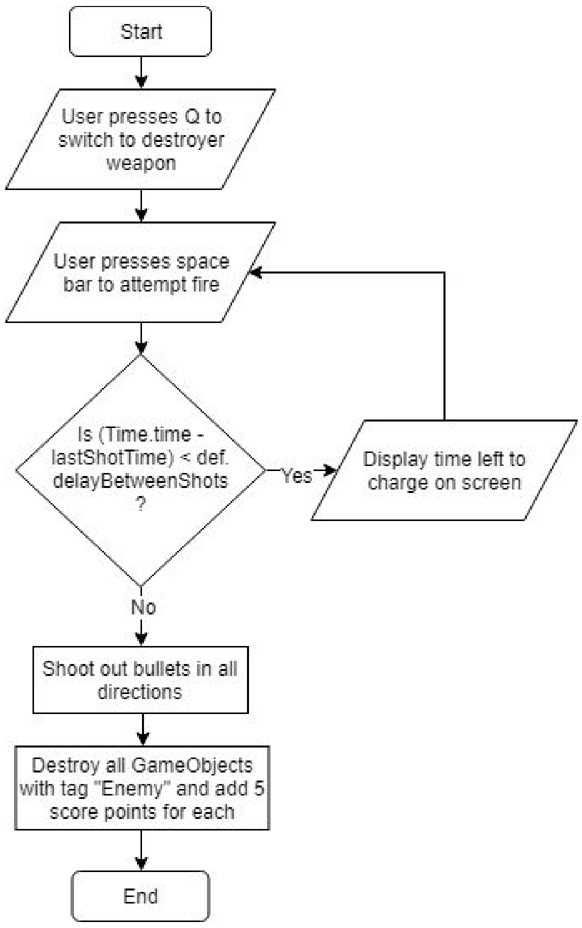
| **Test** | **Passed?** |
|---|---|
| Power-ups spawns occasionally when enemy ships are destroyed | Yes |
| Power-ups are picked up by the Player when Player ship collides with it | Yes |
| Power-ups type varies randomly | Yes |
| Power-ups have an effect when picked up (speed, shield, rate of fire) | Yes |
| Power-ups fade for 4 seconds after 6 seconds being alive | Yes |
| Power-ups drift in random directions | Yes |

**3) Destroyer Weapon**

Description: This feature introduces a new weapon to the game called the destroyer. The destroyer weapon can be shot after a 10-second charge time, and kills all enemies currently on the map.

Key Components: A new Weapon Definition will be created for the destroyer. Damage on hit will be set to zero, and the delay between shots will be set to 10 seconds. When charged and shot, the destroyer will shoot bullets out in all directions. All enemies on the screen currently will be deleted using an array. User will be awarded 5 points per enemy killed using the destroyer.

Diagrams:


Start
User presses Q to switch to destroyer weapon
User presses space bar to attempt fire
Is (Time.time - lastShotTime) < def. delayBetweenShots ?
Yes
Display time left to charge on screen
No
Shoot out bullets in all directions
Destroy all GameObjects with tag "Enemy" and add 5 score points for each
End

Test Plan:

| Test | Passed? |
|---|---|
| Destroyer weapon must be charged 10 seconds before shooting | Yes |
| Destroyer weapon kills all enemies on the screen | Yes |
| Score is updated with 5 points per enemy killed | Yes |
| Bullets shoot out in all directions | Yes |

**4)  Enemy Projectiles**

Description: This feature allows Enemy_2 objects in the game to shoot projectiles downwards in the scene. Each Enemy_2 fires at a default rate of one projectile every 90 frames, and the projectile moves with a speed of 20. The projectiles are instantiated at a slight offset below the barrel of Enemy_2, to prevent any unintentional collisions which would cause the projectile to be stuck in the scene. The projectiles are destroyed when they either exit the bottom of the scene, collide with another Enemy_2 object, or collide with the Hero. In the event that the projectile collides with the Hero, it will remove a shield level, or restart the game if the Hero has no shield left.

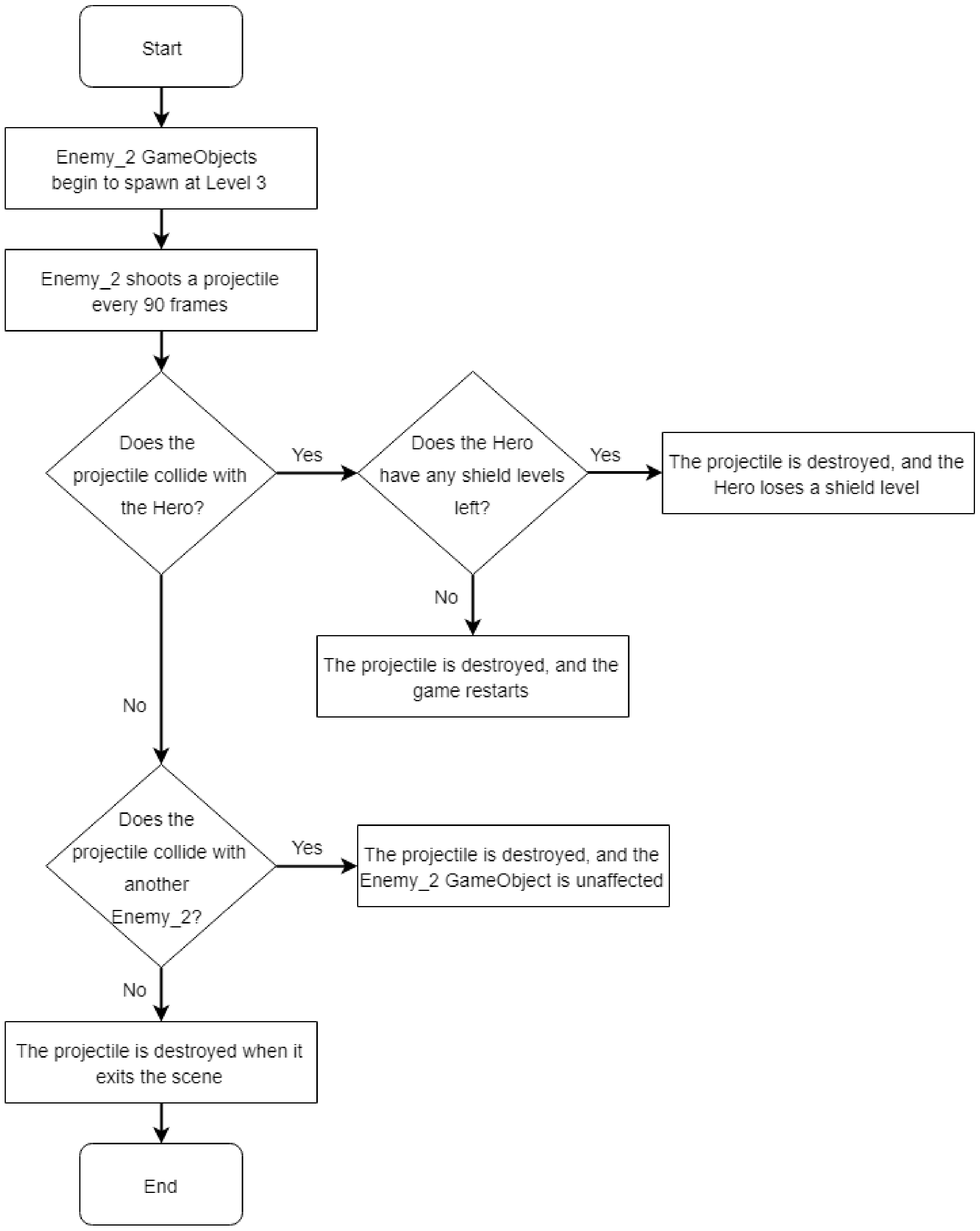
Key Components: A new prefab will be created called ProjectileEnemy, and will be tagged with the same name as well. To instantiate this prefab, a new method called FireProjectile was created in the Enemy_2 script. This method also dictates the velocity that the projectile moves in through: rigidB.velocity = Vector3.down * projectileSpeed;

Furthermore, the Enemy_2 Start method will be modified to assign a GameObject variable to the barrel of the Enemy_2 spaceship. This is necessary so that unwanted collisions between the projectile and the spaceship occur, and is implemented through:
projectile.transform.position = barrel.transform.position -  offset;

The OnTriggerEnter and OnCollisionEnter methods were also modified within the Hero and Enemy classes respectively to allow for the appropriate interactions when a colliding with a GameObject with the “ProjectileEnemy” tag. An example of this implementation for Hero is adding an OR condition of  “go.tag == "ProjectileEnemy" to the if statement used to remove shield levels or destroy the Hero and restart the game.

The last component of this feature was changing the Update method in the Enemy class to be a virtual function. This was necessary because Enemy_2 GameObjects required movement AND shooting functionalities throughout each frame of the game.

Diagrams:


Start
Enemy_2 GameObjects begin to spawn at Level 3
Enemy_2 shoots a projectile every 90 frames
Does the projectile collide with the Hero?
Yes
Does the Hero have any shield levels left?
Yes
The projectile is destroyed, and the Hero loses a shield level
No
The projectile is destroyed, and the game restarts
No
Does the projectile collide with another Enemy_2?
Yes
The projectile is destroyed, and the Enemy_2 GameObject is unaffected
No
The projectile is destroyed when it exits the scene
End

Test Plan:

| Test | Passed? |
|---|---|
| Enemy_2 shoots a projectile every 90 frames | Yes |
| Enemy_2’s projectile is destroyed when it collides with Hero or another Enemy_2 | Yes |
| Hero loses shield charges or the game restarts when it collides with an enemy projectile | Yes |
| Enemy_2’s projectile is destroyed when it exits the scene | Yes |

**5) Music and Sound Effects**

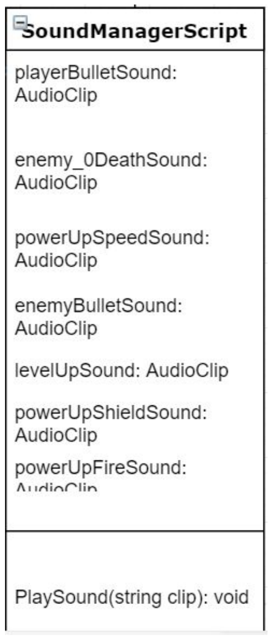
Description: This feature adds audio to the game, making it much more engaging and satisfying to play. There is a simple background instrumental that loops constantly throughout the game. There is a sound effect for: player bullets, enemy bullets, each individual powerup type, leveling up, enemy deaths.

Key Components: A new class will have to be created, called SoundManagerScript. This class will initialize all of the necessary audio clips: playerBulletSound, enemy_0DeathSound, powerUpSpeedSound, powerUpShieldSound, enemyBulletSound, levelUpSound, powerUpFireSound. It will also initialize the AudioSource.

In the start method, this class will load all of the audio files from the resource folder and associated them with the audio clip objects declared at the top of the class.

This class will also have a method called PlaySound (string clip) that uses a switch and has a case for each audio clip. In the appropriate functions in other classes, this PlaySound method in SoundManagerScript will be called to play the appropriate sound a single time. For example, in the Hero class, the PlaySound method will be called when the hero collides with a powerup.

Diagrams:


SoundManagerScript
playerBulletSound: AudioClip
enemy_0DeathSound: AudioClip
powerUpSpeedSound: AudioClip
enemyBulletSound: AudioClip
levelUpSound: AudioClip
powerUpShieldSound: AudioClip
powerUpFireSound:
PlaySound(string clip): void

Test Plan:

| **Test** | **Passed?** |
|---|---|
| Background music plays and loops after finishing | Yes |
| Each power up has a different sound upon pickup | Yes |
| Sound effect plays every time player fires a bullet | Yes |
| Sound effect plays every time enemy fires a bullet | Yes |
| Sound effect plays upon leveling up | Yes |