**1. Hitboxes and Collision Detection (2pts)**

* **a)** The player should be able to fire projectiles and damage/kill enemies and bosses using the keyboard, and the player should be able to be killed by the enemies’ and bosses’ projectiles.
  + **Current Status**: You have implemented player controls and projectile firing, but you need to implement collision detection logic to handle interactions between player projectiles and enemies/bosses, as well as between enemy projectiles and the player.
* **b)** For the player’s projectiles, your implementation can be as simple or complicated as you want.
  + **Current Status**: You have a basic implementation for player projectiles. Ensure that they have distinct textures to avoid confusion with enemy projectiles.
* **c)** For enemy projectiles, hitboxes vary depending on the sizes and shapes of the bullets.
  + **Current Status**: You need to define hitboxes for enemy projectiles. This can be done by adding a hitbox component or properties to the projectile classes.
* **d)** For enemies and bosses, the hitboxes are much larger and mostly cover the entire sprite.
  + **Current Status**: You need to implement hitboxes for enemies and bosses that cover their sprites. This will involve defining the hitbox size and checking for collisions.
* **e)** The player’s hitbox is much smaller than the sprite.
  + **Current Status**: You need to implement a hitbox for the player that is smaller than the player's sprite and check for collisions with enemy projectiles.

**2. Life Systems for the Player (3pts)**

* **a)** The player has an initial number of lives that shall be displayed properly.
  + **Current Status**: You need to implement a life system for the player, including a way to display the number of lives.
* **b)** Every time the player is hit by a projectile, it loses one life immediately.
  + **Current Status**: This functionality is not implemented. You need to add logic to handle player damage and life reduction.
* **c)** If there are still lives left, the player respawns from the center bottom screen.
  + **Current Status**: This functionality is not implemented. You need to implement respawning logic and invincibility after respawning.
* **d)** If there are no more lives left, the player loses the game and the game should show a proper prompt for the player to end or exit the game.
  + **Current Status**: This functionality is not implemented. You need to add game over logic and prompts.
* **e)** If the player survives till the Final Boss is beaten, the player wins the game.
  + **Current Status**: This functionality is not implemented. You need to implement win conditions and prompts.

**3. Game Details (5pts)**

* **a)** Entry/exit of waves of different types of enemies, bosses (timeout should be implemented as well if the bosses do not die in a certain amount of time).
  + **Current Status**: You have a **WaveManager** that handles enemy spawning, which is a good start. However, you need to implement timeout logic for bosses if they are not defeated within a certain timeframe.
* **b)** Movement of enemies, bosses, bullets should be fleshed out to make it feel like a real game.
  + **Current Status**: You have basic movement for the player and bosses, but you need to ensure that enemies have varied movement patterns (at least three different types).
* **c)** Quantities of enemies, bullets, and their patterns.
  + **Current Status**: You need to define the quantities of enemies that spawn and their attack patterns.
* **d)** Win/loss condition should be implemented correctly.
  + **Current Status**: This functionality is not implemented. You need to add logic to determine when the player wins or loses the game.

**Summary of What You Are Lacking**

1. **Collision Detection**: Implement hitboxes for the player, enemies, and projectiles, and ensure collision detection is functional.
2. **Player Life System**: Create a system to manage player lives, handle damage from enemy projectiles, and manage respawning.
3. **Timeout Logic for Bosses**: Extend the **WaveManager** to include logic for handling timeouts for bosses if they are not defeated within a certain timeframe.
4. **Varied Enemy Movement**: Implement varied movement patterns for enemies to enhance gameplay.
5. **Define Enemy Quantities and Patterns**: Specify how many enemies spawn and their attack patterns.
6. **Win/Loss Conditions**: Implement logic to determine when the player wins or loses the game based on the state of the game objects.