The Momoks

HauntU: A Swarm Attack Survival Game

Project Documentation

TEB1043: Object Oriented Programming

No.	Student Name	Student ID	Program
1	Mawar Fasha binti Roslan	22010115	IT
2	Sarah Yasmin binti Azrul Asraf	22010114	IT
3	Ainur Batrisyia binti Muhamad Zairul	22010449	IT
4	Luqman bin Azman	22010284	CS
5	Nur Edleena Erniesa Binti Mohd Erman	22010329	IT
6	Aina Aireena Binti Anuar	22010366	IT

Contents

1.0 Project Overview

Project Name: HauntU
Platform: Unity
Programming Language: C#

Overview:
HauntU is a 2D action game developed using Unity, where players control a character who must fight supernatural enemies and navigate through dangerous environments. The game is set in a dark and eerie world, where players encounter various waves of enemies and powerful bosses. The primary goal is to survive against enemy waves, manage resources like health and stamina, and defeat bosses using a variety of weapons.

The game focuses on action-packed combat, with the player engaging in melee and ranged combat using a variety of weapons, including brooms, pistols, and shotguns. Players also deal with environmental challenges, such as knockback effects, screen shakes, and damaging hazards.

Gameplay Focus:

- Combat: Player uses primary weapons, like a broom, and secondary weapons like a pistol or shotgun to battle enemies.
- Survival: The player must manage health, stamina, and weapon durability while fighting waves of enemies and bosses.
- Exploration: Players traverse different maps, encountering various environmental hazards and challenges.

2.0 Key Features

- **Enemy Waves & Boss Fights:**
  The game is designed around progressively difficult waves of enemies, with boss battles at the end of certain waves. Players must strategize their attacks and manage resources to defeat tougher enemies.

- **Health & Stamina Management:**
  Players must keep track of their health and stamina throughout the game. The player's health is impacted by enemies' attacks, while stamina is required for dashing and other actions.

- **Weapon System:**
  A variety of weapons are available, including brooms, pistols, shotguns. Each weapon has its own attributes like damage, range, and attack speed.

- **Interactive Environment:**
  Environments are destructible, with objects that can be destroyed to trigger visual effects or open paths. There are also pickups that players can collect to restore health or gain power-ups.

- **Wave Spawner & AI:**
  The game has a wave-based spawning system, with enemies that follow and attack the player. Bosses appear at specific wave intervals, with stronger AI behaviors and special abilities.

- **Upgrades and Power-ups:**
  Players can collect upgrades and power-ups that enhance their abilities, such as boosting health, increasing damage, or reducing weapon cooldowns.

- **Pause Menu & Inventory:**
  The player can pause the game at any time to access the inventory or upgrade menu, where they can switch weapons, check stats, and upgrade abilities.

Ghost Characters (Enemies):

- **Hantu Pocong:** Slow but tough, absorbs damage, and can summon more pocongs.
- **Toyol:** Small and fast.
- **Pontianak:** Moves quickly, attacks with sharp claws.

3.0 Maps

HauntU features three distinct environments where players will encounter different challenges and enemies. Each map is designed to provide a unique gameplay experience with environmental hazards, enemy placements, and strategic locations for combat.

1. **The Barn:**
    - A spooky, abandoned barn located on the outskirts of a ghost town. The Barn is filled with hay bales, old tools, and creepy sounds. It features tight spaces and enemies that can surprise players by emerging from the shadows.
2. **Gas Station:**
    - An eerie gas station in the middle of nowhere. It's a desolate place with leaking fuel tanks, abandoned vehicles, and broken-down machinery. Players must be careful not to damage the environment too much, as explosive hazards are present.
3. **Abandoned Hotel:**
    - A large, creepy hotel that's seen better days. This multi-floor hotel offers a combination of long hallways, abandoned rooms, and hidden passages. Players will need to explore each floor while dealing with supernatural enemies and the ever-present sense of dread.

4.0 Classes and Description

Here is a key classes in the game, describing their roles and functionality:

Player Classes

- **PlayerHealth:**
  Handles the player's health, including current health, maximum health, damage resistance, and the effects of knockback. Also manages the death state and health recovery after taking damage.
- **PlayerController:**
  Manages player movement, controls for interacting with objects, and combat actions. This class processes inputs for walking, dashing, and attacking. It also determines the player's facing direction and movement speed.
- **ActiveWeapon:**
  Controls the player's currently equipped weapon, including attributes like damage, cooldown, and attack mechanics. The class also manages weapon swapping, firing, and attacks, including melee and ranged options.
- **DamageSource:**
  Defines the damage amount and type for different sources, including projectile weapons, melee attacks, and environmental damage. It calculates damage dealt to enemies and the player.
- **Stamina:**
  Manages the player's stamina pool. Stamina is required for actions like dashing or using special abilities. The class tracks current stamina, maximum stamina, and stamina regeneration over time.
- **Projectiles & SlashAnim:**
  Manages projectile-based attacks (e.g., bullets, energy blasts) and animations for melee attacks. This class handles the movement, collision, and damage of projectiles, as well as animations for attacking.
- **PlayerControls:**
  Handles the input system for the player, defining movement actions, combat actions (attacks, dashes), and inventory usage. It creates seamless interaction between the player and the game world.

Enemy Classes

- **EnemyHealth & BossEnemyHealth:**
  Tracks enemy and boss health, handling health depletion, death effects, and spawning death animations or special effects. Boss enemies have additional attributes and complex behavior.

- **EnemyFollow:**
  AI that allows enemies to follow the player based on proximity and attack when in range. It also determines the speed at which enemies move and their behavior when encountering obstacles.

- **Knockback:**
  Defines the knockback effect for both enemies and the player, determining how far and in what direction they are pushed when hit by an attack.

- **Flash:**
  Implements flash effects used to indicate temporary invincibility or other status effects, allowing players or enemies to avoid damage for a brief period.

- **WaveSpawner:**
  Controls the spawning of enemies in waves, including normal enemies and bosses. The class also adjusts spawn rates and enemy counts for increasing difficulty as the game progresses.

Game Management Classes

- **Destructable:**
  Handles destructible environmental objects, triggering visual effects like explosions or broken objects. It allows objects in the game world to interact with players and enemies dynamically.
- **Parallax:**
  Implements a parallax scrolling effect to give the game's 2D world depth. It creates the illusion of layers in the background that move at different speeds, adding a sense of immersion.
- **Singleton:**
  Ensures that certain classes (e.g., game managers, systems) have only one instance throughout the game, improving performance and reducing memory usage.
- **ScreenShakeManager:**
  Creates screen shake effects to enhance the intensity of events like explosions or boss attacks. This effect adds an extra layer of drama and impact to critical moments in the game.
- **Timer, PauseGame, Menu:**
  Manages the in-game timer, pause functionality, and menu systems. This includes the game-over screen, settings menu, and inventory management. Players can pause the game, check stats, and make decisions.

Weapon & UI Classes

- **Pistol & Shotgun:**
  Controls weapon behavior for ranged combat, including firing mechanics, ammo count, and damage. These classes provide the player with versatile weaponry for combat against enemies.
- **WeaponInfo:**
  Contains detailed information about each weapon, including damage, cooldown, and range. This class is used by various weapon types to define their behavior and attributes.
- **ActiveInventory & GameOverScreen:**
  Manages the player's inventory system, allowing them to equip, upgrade, and switch weapons. The game-over screen displays player stats and progress at the end of the game.

5.0 Development Log

Phase 1: Core Player Mechanics (10 Sept - 15 Sept)

Date	Assigned Developer	Classes Worked On	Details
10 -15 Sept	**Luqman**	PlayerController, PlayerHealth, PlayerControls	Started work on **PlayerController** to handle player movement and actions. Implemented **PlayerHealth** system to track player health. Set up **PlayerControls** to handle input actions.
	Luqman	ActiveWeapon, WeaponInfo, DamageSource	Implemented **ActiveWeapon** to manage the player's current weapon, and **WeaponInfo** to hold weapon data (e.g., damage, range). Developed **DamageSource** to handle damage calculation.
	Luqman	PlayerController, PlayerHealth	Polished **PlayerController** for smooth movement and added death state to **PlayerHealth** (with kill count and health management).
	Luqman	PlayerControls	Finished input handling for player actions (movement, attacking, dashing).
	Luqman	PlayerController, PlayerHealth, DamageSource	Final integration of **PlayerController** with health and damage systems, ensuring damage and death effects are correctly handled.
	Luqman	ActiveWeapon, WeaponInfo	Polished weapon mechanics and damage calculations to make weapons fully functional.

Phase 2: Combat Mechanics (16 Sept - 25 Sept)

Date	Assigned Developer	Classes Worked On	Details
16-25 Sept	**Sarah & Luqman**	Pistol, Shotgun	Started work on the **Pistol** and **Shotgun** weapons, integrating animations and firing mechanics for both.
	Sarah & Luqman	MagicLazer	Implemented **MagicLazer** for the **Shotgun**, adding a new weapon attack mechanic with a laser beam effect.
	Sarah	Pistol, Shotgun	Polished shooting mechanics for **Pistol** and **Shotgun**, including cooldown, damage, and ammo management.
	Sarah & Luqman	Pistol, Shotgun	Debugged weapon functionality, ensuring smooth transitions between weapons and proper visual feedback (muzzle flash, etc.).
	All Team Members	Pistol, Shotgun, MagicLazer	Final adjustments for weapon mechanics, ensuring proper firing behavior and effects for all weapons.
	Mawar, Nina, Sarah, Aina, TIsya	Graphics	Start sketching and rendering ideas for characters.

Phase 3: Camera and Enemy AI (21 Sept - 26 Sept)

Date	Assigned Developer	Classes Worked On	Details
21 – 26 Sept	**Nina & Luqman**	CameraFollow, MouseFollow	Implemented **CameraFollow** for smooth camera movement tracking the player. Started work on **MouseFollow** to align the player character with the mouse position.
	Nina	EnemyFollow	Developed **EnemyFollow** AI, allowing enemies to track and move towards the player.
	Nina	CameraFollow, MouseFollow, EnemyFollow	Integrated **CameraFollow** with the player character's movements, while refining the **EnemyFollow** logic.
	Nina	CameraFollow, EnemyFollow, MouseFollow	Ensured smooth interaction between the camera, player, and enemies in the game world.
	Nina	EnemyFollow	Polished the **EnemyFollow** AI to handle different enemy behaviors based on player location and movement.

Phase 4: Health and Visual Effects (26 Sept – 31 Sept)

Date	Assigned Developer	Classes Worked On	Details
26-31 Sept	**Mawar & Luqman**	EnemyHealth, BossEnemyHealth, PlayerHealth	Started implementing **EnemyHealth** for enemies, ensuring health tracking and damage handling. Added **BossEnemyHealth** for boss enemies.
	Mawar	PlayerHealth	Refined **PlayerHealth** to manage health regeneration and death effects.
	Mawar	EnemyHealth, BossEnemyHealth	Integrated **EnemyHealth** with damage systems for both regular enemies and bosses.
	Mawar & Luqman	EnemyHealth, BossEnemyHealth	Continued polishing enemy health systems, including death animations and effects.
	All Team Members	PlayerHealth, BossEnemyHealth	Final testing and debugging of health systems for player and boss enemies, ensuring proper behavior during gameplay.
	Mawar, Nina, Sarah, Aina, TIsya	Graphics	Finalizing graphics, final sketches and integrating in game.
	Mawar, Nina, Sarah, Aina, TIsya	SFX	Finding suitable SFX for the game.

Phase 5: Visual Feedback (5 Oct - 9 Oct)

Date	Assigned Developer	Classes Worked On	Details
5 – 9 Oct	**Aina & Luqman**	Flash, ScreenShakeManager, Knockback	Began work on **Flash** for visual feedback on damage, and **ScreenShakeManager** for camera shakes during impacts.
	Aina	Flash, ScreenShakeManager	Integrated **ScreenShakeManager** with combat events like enemy hits.
	Aina	Knockback	Implemented **Knockback** mechanics, allowing enemies and the player to be knocked back on impact.
	Aina	Flash, Knockback	Polished **Flash** for effective visual feedback and synchronized **Knockback** effects.
	All Team Members	ScreenShakeManager, Knockback	Final testing for **ScreenShakeManager** and **Knockback** to ensure smooth and responsive visual effects.
	Mawar, Nina, Sarah, Aina, TIsya	SFX	Finalizing SFX for the game.

Phase 6: Game Flow and UI (7 Oct - 10 Oct)

Date	Assigned Developer	Classes Worked On	Details
7 -10 Oct	**Tisya & Luqman**	Timer, PauseGame, Menu	Started work on **Timer** to track game time, and **PauseGame** to manage pausing and resuming gameplay.
	Tisya	PauseGame, Menu	Developed **Menu** functionality, enabling navigation between pause, main menu, and settings.
	Tisya	Timer, PauseGame	Refined **PauseGame** and **Menu** systems, ensuring smooth transitions during pauses.
	Tisya	Timer, PauseGame	Polished UI interactions and fixed bugs in **PauseGame** and **Timer** mechanics.
	All Team Members	PauseGame, Menu	Final tests on game flow features, ensuring proper UI and pause functionality.

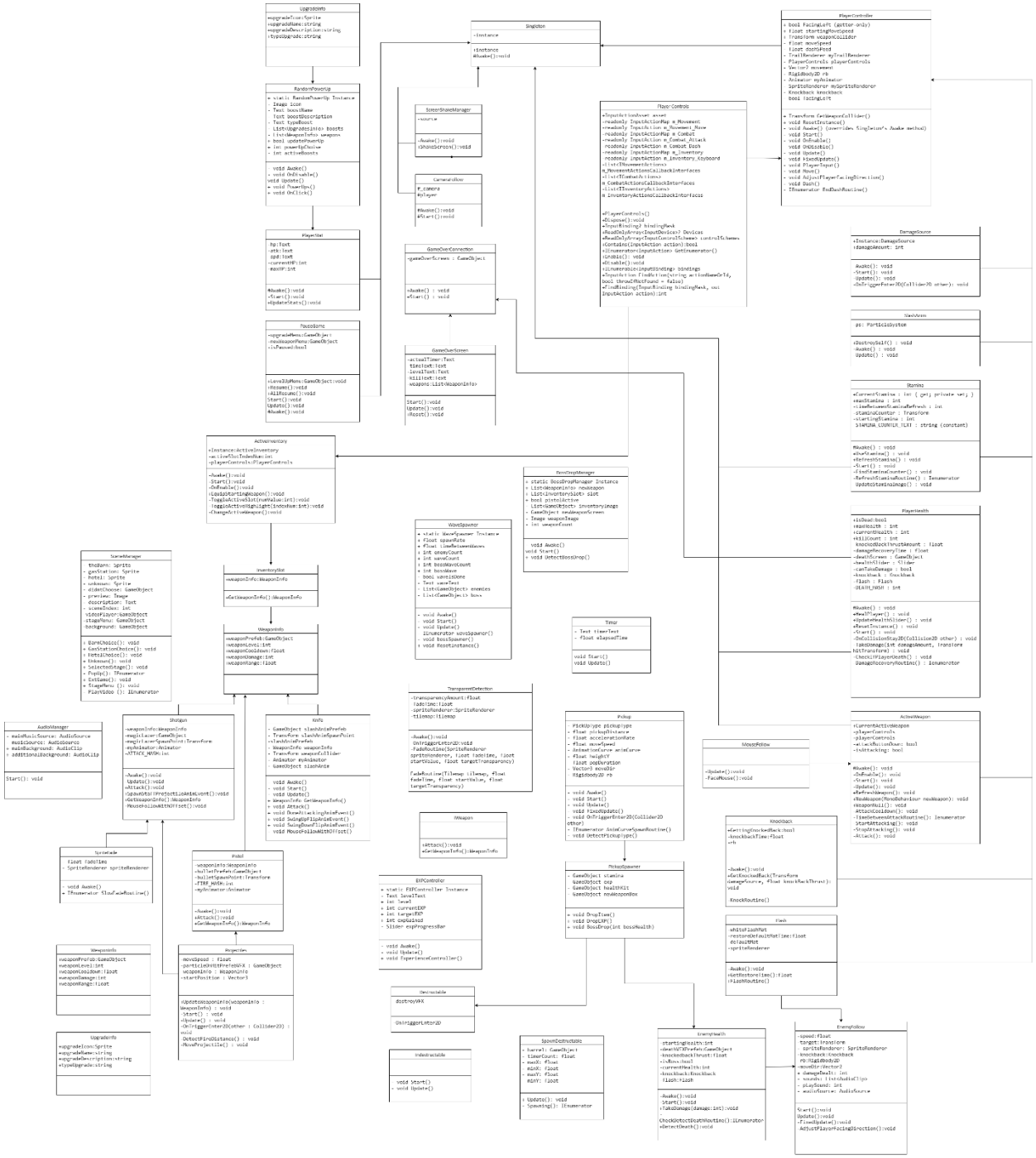
Phase 7: World and Environmental Mechanics (11 Oct - 15 Oct)

Date	Assigned Developer	Classes Worked On	Details
11 -15 Oct	**Luqman**	Destructable, Parallax, TransparentDetection	Developed **Destructable** for breakable objects, and **Parallax** for layered background effects.
	Luqman	TransparentDetection	Worked on **TransparentDetection**, ensuring correct fade-in/fade-out effects for transparency in certain objects.
	Luqman	Parallax, Destructable	Final adjustments to **Destructable** objects and added complex parallax effects to the background.
	Luqman	Parallax, TransparentDetection	Integrated **Parallax** and **TransparentDetection** into gameplay environments.
	Luqman	Destructable, Parallax	Polished and fine-tuned visual effects for environmental interactions.

Phase 8: Final Integration and Balancing (16 Nov - 20 Nov)

Date	Assigned Developer	Classes Worked On	Details
16 - 20 Nov	**All Team Members**	Final integration of all classes, balancing game difficulty, testing enemy waves	Final adjustments and integration of all systems, balancing difficulty and ensuring smooth transitions between phases.
	Luqman & Tisya	BossDropManager, WaveSpawner	Integrated and balanced **WaveSpawner** and **BossDropManager** for final wave mechanics and loot drops.
	All Team Members	Final game flow checks, bug fixes	Performed final checks and debugging to ensure everything works as intended.
	Luqman & Tisya	Polished game flow, UI, and effects	Final polishing of UI, animations, and overall gameplay mechanics.
	All Team Members	Testing and integration	Comprehensive testing of the game.

6.0 UML



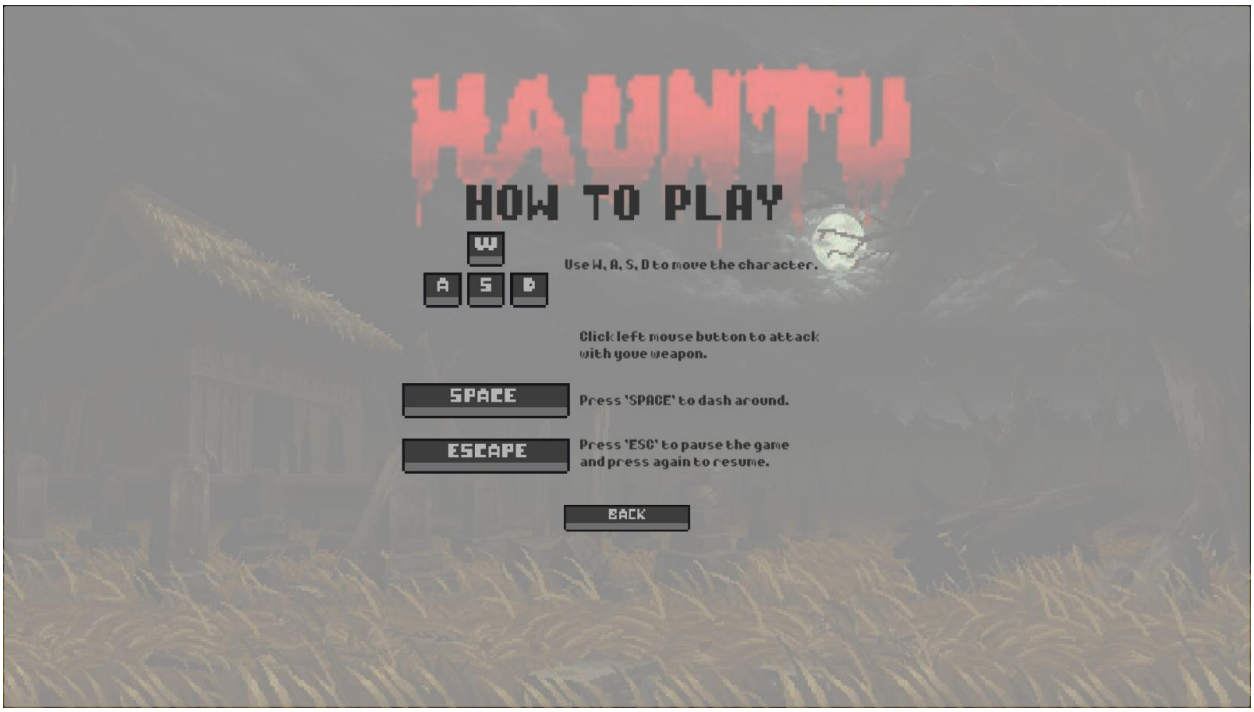
7.0 Screenshot

1. Main Menu

Welcome to the main menu of *HauntU*! From here, you can access all key features:

- **Start**: Dive right into the thrilling world of *HauntU* and face the supernatural challenges ahead.
- **How to Play**: New to the game? Learn the controls, gameplay mechanics, and strategies to survive.
- **Customize** (Coming Soon): Personalize your game experience with exciting options (currently unavailable).
- **Settings**: Adjust audio and view credits.
- **Quit**: Exit the game when you're ready to take a break.

2. How to Play

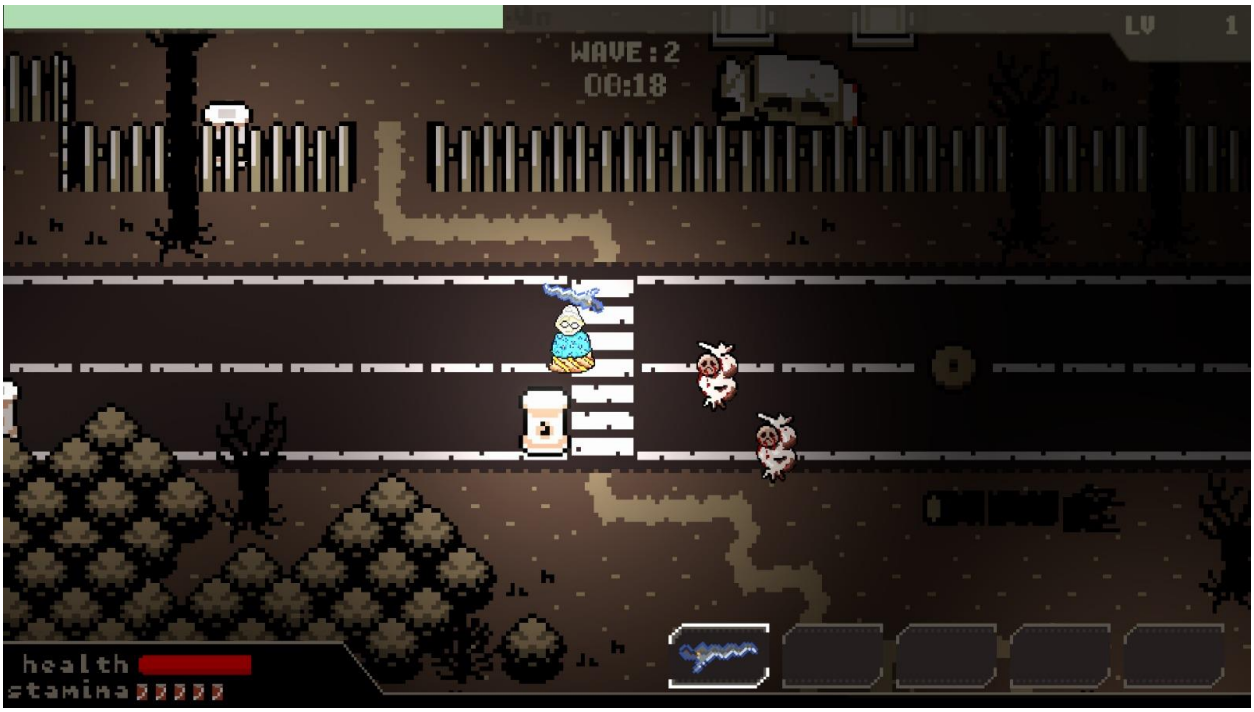


How to Play: A manual for the game.

3. Credits

Credits: Acknowledges the creators and contributors behind *HauntU*.

4. GamePlay

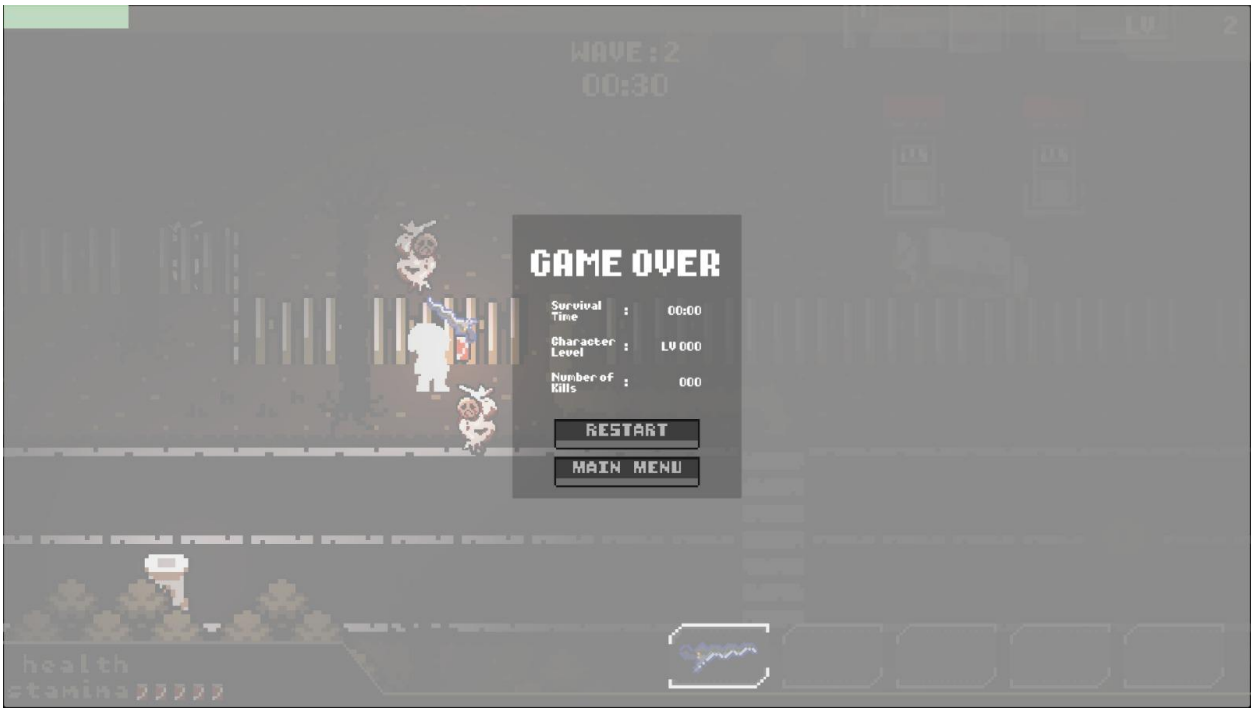


Gameplay: Engage in thrilling battles against supernatural enemies with escalating challenges.

5. Next Level/ Level Up

Next Level/Level Up: Celebrate progress as you advance to more challenging stages or upgrade your abilities.

6. Game Over



Game Over Screen: Displays when the player has lost, summarizing performance and offering options to retry or exit.

8.0 Evaluation of Platform (Unity)

Platform: Unity
Programming Language: C#

Why Unity?

1. **Ease of Use:**
   Unity provides an intuitive interface and a rich set of tools for 2D game development. With its drag-and-drop feature for assets and a visual scene editor, it made development faster and more efficient. The C# scripting language was used to implement the game logic, manage player controls, enemies, weapons, and more.

2. **Performance & Optimization:**
   Unity allows for efficient management of game performance, especially for 2D games, by optimizing sprite rendering, physics calculations, and memory usage. This was important in ensuring smooth gameplay as the game features multiple enemies and complex animations.

3. **Community & Documentation:**
   Unity has a large and active developer community, offering forums, tutorials, and extensive documentation. This was invaluable during development, as it provided quick solutions to any issues encountered.

4. **Scalability:**
   As the game evolves, Unity's structure supports the easy addition of new gameplay elements, enemies, weapons, and systems without significant rework. This scalability made Unity a good choice for future updates and expansion of HauntU.

Challenges & Limitations:

- **Learning Curve for Some Features:** While Unity is user-friendly, there were some challenges when implementing complex game systems like AI behavior and managing complex player inputs. C# scripting required a learning curve for new team members.
- **Asset Management:** Managing large numbers of assets (sprites, animations, sound files) can become cumbersome, especially as the game's scope increased. Proper organization and optimization strategies were needed to keep everything running efficiently.