CPSC 427 (Fall 2025) - Game Proposal

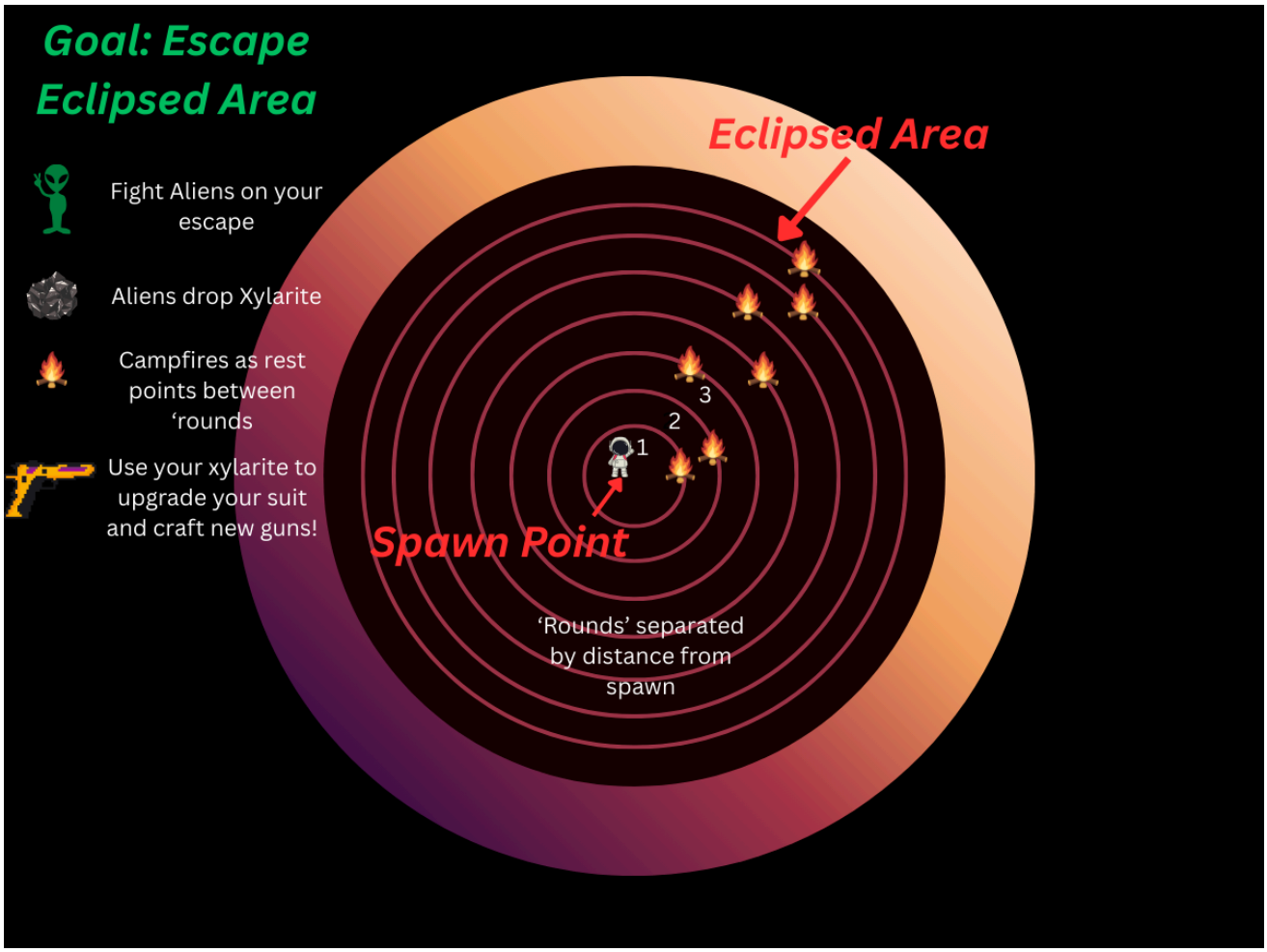
Team Name	Team Saturday (Canvas Team #14)
Game Name	Eclipse
Team Members	Durukan Butun (23857824) Nanjou Olsen (59393686) Alan Zheng (18887208) Chowdhury Zayn Ud-Din Shams (46176756) Han Kim (45508413) Will Beaulieu (24994386) Nishant Molleti (21147343)

Game Concept

Eclipse is a top-down shooter game inspired by horde survival games like Zombie Estate. The story revolves around our main character who has crash-landed on an enemy planet where there is a permanent eclipse caused by a celestial body between the planet and the star. The main goal is to escape the shadow of the eclipse as hostile aliens roam inside it.

The twist in our gameplay compared to others in the genre is the realistic lighting system and focus on immersion. Unlike most horde games that play like an arcade shooter, eclipse will use its realistic lighting and sound system to immerse the player into the world.

When the player first enters the world, they are slow and weak, facing weak enemies. The enemies drop a rare mineral called Xylarite which can be used to upgrade the player's weapons and space-suit, allowing the player to get stronger over time. Every time the player reaches a certain distance from the start point, they will find a campfire (a rest point), marking the end of that ‘round’. While resting at a campfire, players can use their Xylarite to upgrade their character, guns, and acquire utility items. As time goes on and as the player gets further from their spawn, the difficulty and quantities of the enemies will increase.



Game Mechanics

Player Movement

The player can freely move around the world using the WASD keys. Their flashlight can be aimed in any direction using the mouse. Weapons are fired in the same direction that the flashlight is aimed.

Players can also unlock a dash movement ability that provides a temporary speed boost on a cooldown.

Combat

Our game's combat is ranged and horde-based: the player uses their gun to shoot down groups of enemies coming towards them. Bullets from the player's gun are collision based and deal different damage amounts based on the weapon type and upgrades. If the player collides with an enemy or enemy bullet, they lose health and have a few invincibility frames. If the player's health reaches 0, they die and are sent back to the beginning of the game (in a rogue-like fashion).

Character Interaction

The player can interact with campfires to heal, obtain new weapons, and upgrade their suit. They can also open treasure chests (containing Xylarite), and collect the Xylarite dropped from enemies by getting close to it.

Resource Management

The player will need to manage the following resources:

1. **Xylarite:** An alien mineral dropped by all enemies and found within treasure chests. Players can use their Xylarite at safe points to acquire new weapons, equipment upgrades, and abilities.
    a. **Suit upgrades**: Movement speed, armour (damage reduction), max health, health regeneration, Xylarite pickup range, Xylarite recovery (bonus % dropped on kills).
    b. **New Weapons**: Sci-fi variants of a pistol, submachine gun, assault rifle, and sniper rifle.
    c. **Weapon Upgrades**: Damage, fire rate, ammo capacity.
    d. **Weapon Ammo**: Each gun type consumes its own ammo type. Ammo can be crafted at the campfire using Xylarite.
    e. **Flashlight Upgrades**: Brightness, cone width.
    f. **Dash Ability**: dash duration, dash speed boost, faster dash cooldown
2. **Utility Items**: Single-use items that can be purchased with Xylarite and used at any time during combat.
    a. **Med Kit**: Heals the player
3. **Player Health**: The player will take damage if they come into contact with an enemy or a projectile. Health can be restored by resting at campfires or using healing items.
4. **Weapon Ammo**: Weapons can only be fired a limited number of times before needing to be reloaded. Once a weapon runs out of ammo, it will automatically be reloaded after a short cooldown. The player can also manually reload their weapon by pressing the R key.

Game Levels

The game will have a set number of procedurally-generated levels (10-25), each with a similar objective for the player: travel a certain distance away from the level's starting point. This distance effectively creates a "combat zone" that the player must escape in order to complete each level.

Procedurally-generated levels will contain obstacles that the player will need to navigate around, such as trees and abandoned buildings, as well as hordes of enemies that the player will come across while exploring. Treasure chests containing Xylarite will also be scattered around each level.

Once the player nears the edge of their current level's "combat zone", a campfire will be generated at a nearby point outside of the zone, and the player will be guided towards it.

After reaching a campfire, the player will have an opportunity to spend their Xylarite to upgrade their equipment, and then take a short rest to recover their health. After resting, they will wake up and see a horde of enemies approaching, thus creating a new combat zone and restarting the game loop.

This game loop will continue until the player reaches their ultimate goal of escaping the eclipse, with levels getting progressively more difficult as the player makes more progress towards their goal.

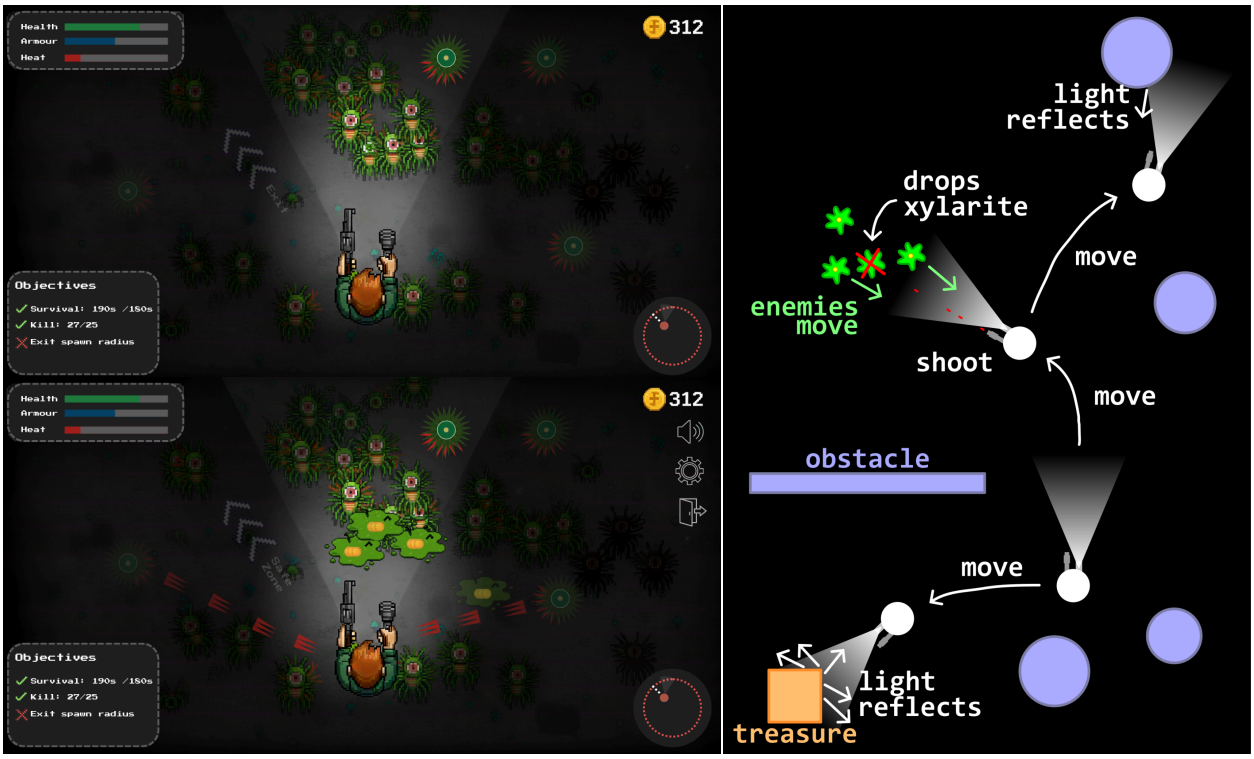
Difficulty Scaling

As shown in the diagram below, enemies gradually scale in difficulty across levels. To prevent players from farming a single level for too long, enemy difficulty also increases more rapidly within each round based on the time spent there. Enemy difficulty is a multiplier applied to the enemies health and spawnrates.

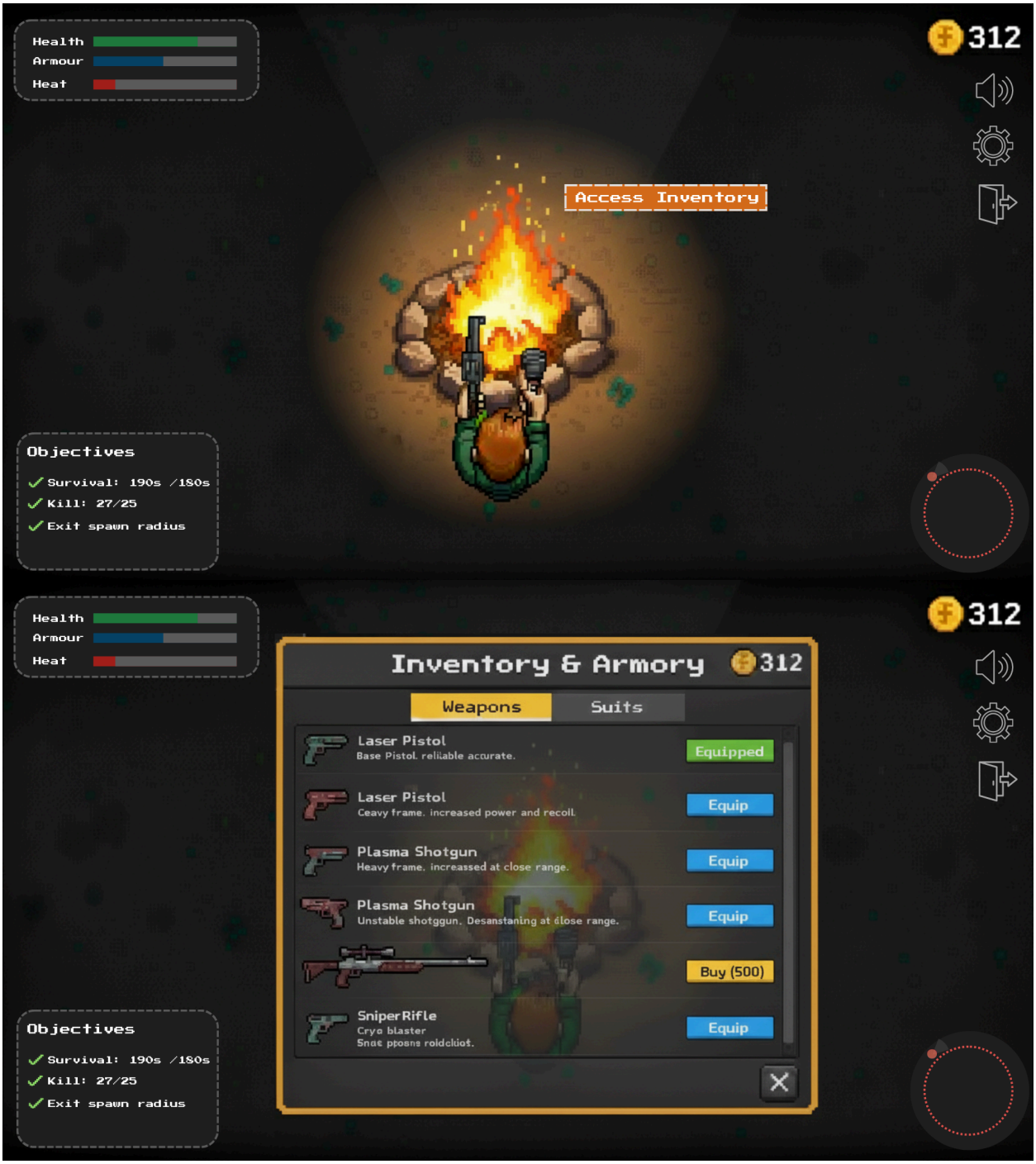
Scenes

Main Menu: selecting “Continue Game” launches the player directly into the game

Combat: player can move, use weapons and items, and collect Xylarite



Rest Area: player can recover health, purchase upgrades, and change equipped items



Controls

Our game will use a combination of keyboard and mouse controls:

1. **WASD Keys**: Move player (8-directional movement)
2. **Mouse**: Aim the flashlight and weapons (independent of player movement)
3. **Left Mouse Button**: Use item (fire weapon, use utility item, etc.)
4. **Mouse Scroll**: Switch to previous/next item slot
5. **Number Keys (1-4)**: Switch to specific item slot
6. **F Key**: Toggle flashlight on/off
7. **E Key**: Interact with the world (open chest, use campfire to upgrade equipment, etc.)
8. **R Key**: Reload weapon ammo
9. **SHIFT Key**: Activate dash ability

Technical Elements

Rendering, Collision, and Geometry

We will render all characters, enemies, and objects using sprite sheets for animations. We will use these sprite sheets to create animations such as walking, shooting, and taking damage.

Interactions such as the player character hitting a wall, collecting loot, activating save points, and hitting enemies with projectiles will rely on 2D collision detection. Geometry manipulation (such as translation and rotation) will be handled using OpenGL transformation matrices to animate and position entities in the world.

Gameplay Logic, AI

Player controls will be responsive, with keyboard inputs being utilized for movement and world interaction, while the cursor will be used to aim the guns and the flashlight.

Enemy AI will control both stationary and moving enemies. In particular, “horde logic” will govern how enemies chase and surround the player, creating immersive gameplay. Enemy AI will also react to player abilities, such as the flashlight or projectile attacks. Grid-based pathfinding will be utilized to allow enemies to navigate around obstacles.

We will also implement physics behaviors, such as enemy knockback when hit.

Advanced Technical Elements

Advanced Lighting System (Raytracing)

Our lighting system is the backbone of our game. We aim to implement raytraced lighting, where light rays are physically simulated so that they realistically bounce off surfaces, produce dynamic soft shadows, and fade away in a gradient after some point. This will make the game, which is set in the dark, feel more ambient and immersive.

- **Plan B**: If we can't get our raytraced lighting system working in time, we can just implement a simple lighting system where we shine some static light in a cone from the player's flashlight, and have it fade away as it moves farther from the flashlight. This won't be nearly as impressive, but it will keep our game playable.
- **Impact if skipped**: Since our game revolves around light and darkness, having a working lighting system is key. Without a proper lighting system, we'll be left with a game set in complete darkness that frustrates players (since they'll be forced to fight enemies in the dark), or one set in daylight that falls very short of our expectations (since dynamic lighting is one of the biggest things that sets our game apart).

Procedural Map Generation

We plan to procedurally generate our game's world to make it more replayable and give our players more of a challenge, since each level will be different each time they play the game.

- **Plan B**: If we can't get procedural generation working in time, we'll manually create a set of levels for our game, so that players have something to play. If we had enough time, we could also create multiple variations of each level to add a little bit of variation.
- **Impact if skipped**: Our game would still be playable without procedurally-generated levels, but it wouldn't have nearly as much replayability, since players would be playing through the exact same levels each time.

Tools

Beyond OpenGL and C++, we'll need an audio library to handle our game's music and sound effects. We're currently looking at using either OpenAL or FMOD for this; we'll need to do some more research before committing to a specific library.

We may also use some open-source pixel art assets in our game.

Team Management

We will use a Kanban board on GitHub Projects to track issues and milestone tasks. We will also meet, at a minimum, every Friday from 4-5 PM to review our tasks and assign them to team members.

Internal deadlines will be noted on each task card. No card's deadline can be set beyond our final internal deadline, which is two days before a milestone is due on Canvas.

Member Duties

While our “member duties” represent our designated areas of focus, these duties are not strictly fixed. This is especially true for M1, as our priority is establishing the core features of our game.

- **Will**: lighting
- **Nishant**: physics, lighting
- **Alan**: player interactions, enemy AI
- **Durukan**: player interactions, enemy AI
- **Zayn**: UI, assets
- **Nanjou**: world design, UI, assets
- **Han**: enemy animations, optimization

Development Plan

Milestone 1: Skeletal Game

Week 1: Basic Engine

- **Zayn**, **Han**, **Nanjou**: ECS design, placeholder sprites
- **Will**, **Nishant**: Start simple lighting system
- **Alan**, **Durukan**: Input handling

Week 2: Core Game Elements

- **Will**, **Nishant**: Finish simple lighting system
- **Nanjou**: Procedural world element placement
- **Alan**, **Durukan**: Player movement and actions
- **Zayn**, **Han**: Start collision system for entities (character, enemies, bullets)

Milestone 2: Minimal Playability

Week 1: Start raytraced lighting system, collision system, tutorial area, enemy AI, basic UI

Week 2: Finish raytraced lighting system, asset integration (texture, sound), full procedural map generation, basic skill/upgrade tree

Milestone 3: Playability

Week 1: Main menu, broader skill/upgrade tree, special abilities, sprite animation

Week 2: Complete UI (animations, splash screens), further asset integration

Milestone 4: Final Game

Week 1: Additional stretch goal features, QoL improvements, difficulty balancing

Week 2: Optimization, final bug fixes