Game Proposal: Seekers

CPSC 427 – Video Game Programming

Team: Los Pollos Hermanos Gaming (LPHG)

Samuel Yiu (32494940)

Ahmed Al Busaidy (29200722)

Jakob Khalil (86176674)

Amir Goodarzvard Chegini (56851199)

Brendon Son (87271490)

Story:

Theme: In a world where ancient magic collides with forgotten technology, survival is everything. Explore vast, mutated landscapes, battle twisted creatures, and unlock the secrets of a land lost to time. Wield powerful swords, master arcane spells, and uncover the mysteries of a world where magic and machines are forever entwined.

Setting: The game is set in the lands of *Eldoria*, a once-vibrant world now shrouded in darkness. Villages are in ruins, forests are twisted by dark magic, and the skies are clouded with an ever-present gloom. Ancient dungeons and crumbling spires dot the landscape, hiding powerful relics and secrets of the past. The wilderness is overrun with twisted creatures, and Morgrath's fortress looms ominously on the horizon, casting its shadow over the land. The journey will take players through cursed villages, treacherous mountains, and forgotten ruins in their quest to restore light to *Eldoria*.

Background Story: In the once-peaceful lands of *Eldoria*, the monstrous tyrant *Morgrath* has seized power, casting a shadow of darkness over the villagers. Under his brutal reign, fear and suffering spread like a plague. A group of brave souls, known as the *Seekers*, have risen to challenge *Morgrath*, but each has fallen to his overwhelming power. Yet, with every failed attempt, more *Seekers* emerge, determined to free their people from oppression. The villagers, fed up with cruelty and suppression, now await the one true leader who will rise, defeat *Morgrath*, and restore peace to the lands in between.

Player Motivation: You (the player) are a spirit of vengeance, traveling the realm by possessing brave explorers. Through possession, an explorer becomes a seeker, infused with the power to slow time. You drive the seeker to uncover the secrets of the past and find a way to restore the world. In each life, a new Seeker serves as the vessel for the spirit of vengeance, destined to carry on the legacy of those who came before.

Game Overview: "*Seekers*" is a top-down 2D action RPG that melds dungeon crawling with open-world exploration. Players assume the role of a *Seeker*, an explorer striving to uncover ancient secrets and restore balance to the lands of *Eldoria*.

The gameplay emphasizes:

- **Exploration**: Navigate a procedurally generated world with diverse biomes like ruined cities, mutated forests, and desert wastelands.
- **Combat**: Engage in skill-based, real-time battles against mutated creatures and autonomous machines using a variety of weapons and abilities.
- **Progression**: Delve into procedurally generated dungeons for randomized loot and face hand-crafted spires with unique bosses to acquire powerful artifacts, buffs, and abilities that enhance the Seeker's capabilities, and advance through the game.
- **Dynamic Environment**: Experience a world affected by dynamic weather and day/night cycles that impact gameplay (e.g., rain slows movement, night enhances enemy speed).
- **Replayability**: Upon death, players control a new Seeker, retaining global buffs but losing items and levels, encouraging continuous improvement and strategy adaptation.

Scenes and Gameplay:

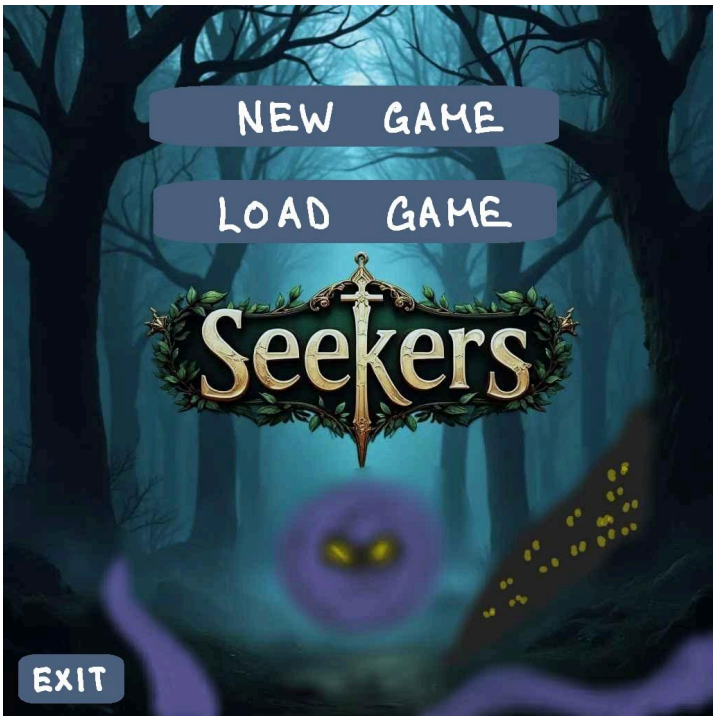
The gameplay and scenes of *Seekers* follow a structured progression of exploration, combat, and strategic interaction within the dark fantasy world of *Eldoria*. The game aims to immerse the player in a dynamic, procedurally generated world, filled with ancient secrets, hostile environments, and progressively challenging enemies. Here's a detailed narrative of how the gameplay unfolds alongside descriptions of the corresponding scenes:

Scene 1: Main Menu

The game begins with a vivid, dark-fantasy background, establishing the theme of a shattered world. The title "Seekers" appears over the backdrop of ruined cities and wilderness. Players are presented with three options: "New Game," "Load Game," and "Exit." The menu design integrates elements of the game story emphasizing the survival theme.

Player Interaction:

- **Navigation**: The player navigates the menu with the mouse, selecting options by left-clicking.
- **Visual Feedback**: Hovering over buttons highlights them with a faint glow, while clicking causes a metallic sound, giving players sensory feedback.

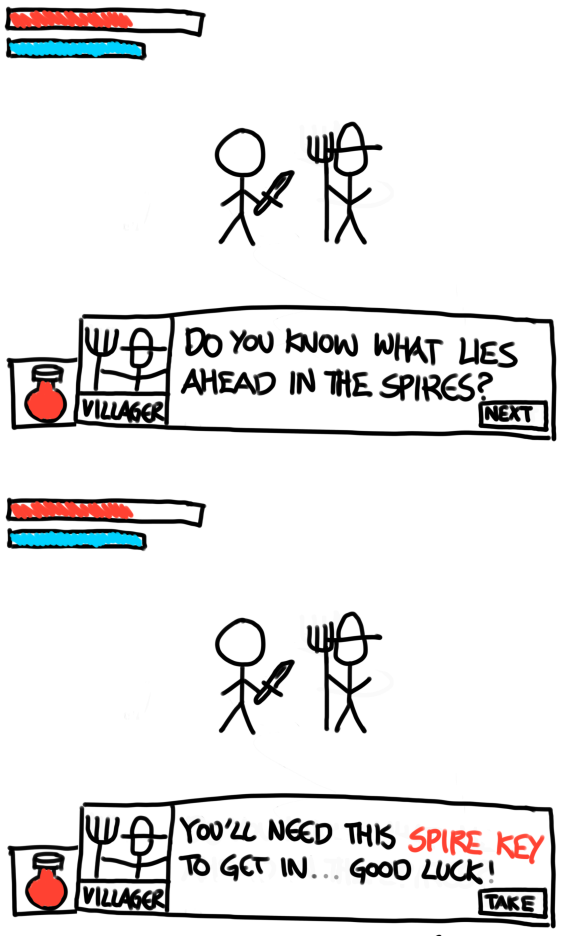


Scene 2: Story Introduction

The story introduction scene serves as a vital part of setting up the player's mission. The scene opens in a small, dimly lit village built from remnants of old technology and magic. The protagonist, known as the *Seeker*, approaches an old villager. The villager recounts the history of *Eldoria*, describing how advanced technology and magic caused a catastrophic event. This sets up the game's lore, explaining the role of the "Ancients" and introducing the Spire Bosses, each a guardian of the secrets of the *Ancients*.

Player Interaction:

- **Dialogue Navigation**: The player can advance dialogue by pressing a key or click to skip through it. Text boxes with subtitles guide the player through this segment.
- **Objective Introduction**: At the end of the conversation, the villager gives the player their first Spire Key. A visual cue shows the item being added to the player's inventory with a sound effect to signify its importance.

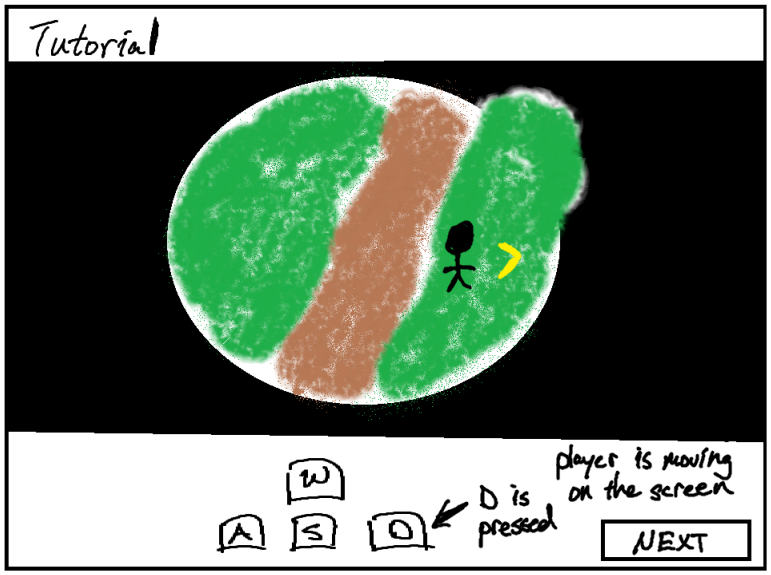


Scene 3: Tutorial

The *Tutorial Scene* introduces the player to the basic mechanics and controls in a safe environment. The player is guided through movement, combat, dodging, looting, and inventory management. This scene ensures the player is familiar with core gameplay elements before entering the more challenging overworld and dungeons.

Player Interaction:

- **Movement:** The player is instructed to use the **WASD** keys to move the Seeker. A prompt appears on-screen, "Use WASD to move."
- **Camera Rotation:** The player is instructed to use the **Q** and **E** keys to rotate the camera.
- **Looting and Inventory Management:** The player is prompted to "Press ‘F’" to pick up items such as sword, bow, armor, and health potion from the ground. The tutorial then explains how to open the inventory using the **‘I’ key**, where players can equip the items.
- **Dodging and Time dilation:** The player is shown how to dodge using the **spacebar** and slow down the time by holding the **shift** key. This is demonstrated by dodging or slowing down projectiles that are shot from a machine in the tutorial area.
- **Combat:** After moving around, an enemy appears, and the player is prompted to use **left-click** for light attacks and **right-click** for heavy attacks. The tutorial explains the mechanics of the sword, bow, and ward. The player will be prompted to switch the weapon by using **tab** to try a ranged weapon as well. There will be a text reminding the player to use consumables, dodge, and time dilation during the combat as needed.
- **Completing the Tutorial:** Once the player has completed these tasks, they are given the option to continue into the overworld or replay the tutorial.



Scene 4: Overworld Exploration (“Surface”)

After the introductory cutscene, the player is placed in *Eldoria*, a procedurally generated world. This vast open world is split into several biomes, including ruined cities, mutated forests, and desert wastelands. Dynamic environmental elements such as weather and the day/night cycle influence gameplay. During exploration, the player encounters hostile creatures and scavenges for resources. Players move from one settlement to the next, finding dungeon entrances, interacting with NPCs, and uncovering secrets hidden in the land.

Player Interaction:

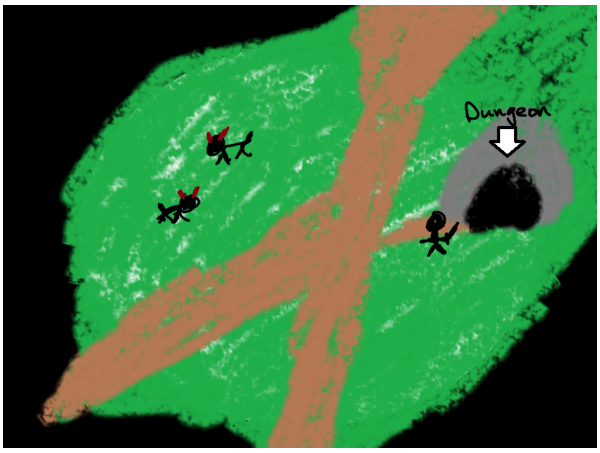
- **Movement**: Players use the WASD keys or arrow keys to move the Seeker and Q and E for camera rotation, with smooth camera tracking that stays centered on the player.
- **Combat**: When enemies approach, combat is initiated. The player uses left-click for light attacks, right-click for heavy attacks, and can use special abilities like time dilation by holding down the shift key.
- **Looting**: After combat, defeated enemies drop loot. A prompt ("Press ‘F’ to collect") appears when the player is near the dropped items, such as consumables or weapons.
- **Environmental Effects**: Weather impacts movement (e.g., rain slows players) and enemy behavior (e.g., nighttime boosts enemy speed and aggression). Visual indicators, like dark clouds or shifting winds, signal upcoming weather changes.

Scene 5: Dungeon Exploration

The player finds dungeon entrances scattered across *Eldoria*. Upon entering, the environment changes dramatically. Dungeons are procedurally generated, meaning no two runs will be identical. Dark corridors, enemy-filled chambers, and traps await the player. Each dungeon culminates in a mini-boss fight, where the player must use both combat prowess and strategy to overcome more powerful enemies. Defeating mini-bosses provides significant loot, such as weapons, armor, and consumables, to prepare for future spire encounters.

Player Interaction:

- **Entering a Dungeon**: As the player approaches a dungeon entrance, a prompt ("Press ‘F’ to enter") appears. Upon entering, the player transitions to an underground environment with distinct visual and audio shifts (e.g., the echoing of footsteps).
- **Exploration**: Players navigate the dungeon using WASD, encountering environmental hazards such as pressure plate traps or hidden spike pits.
- **Combat**: Mini-bosses appear at key points. Players must adapt to their attack patterns and could use time dilation abilities to slow the action, evading heavy attacks and launching counter-strikes.
- **Looting**: After defeating enemies or discovering treasure rooms, players press ‘F’ to open chests and collect loot. The inventory is updated in real-time with visual feedback.

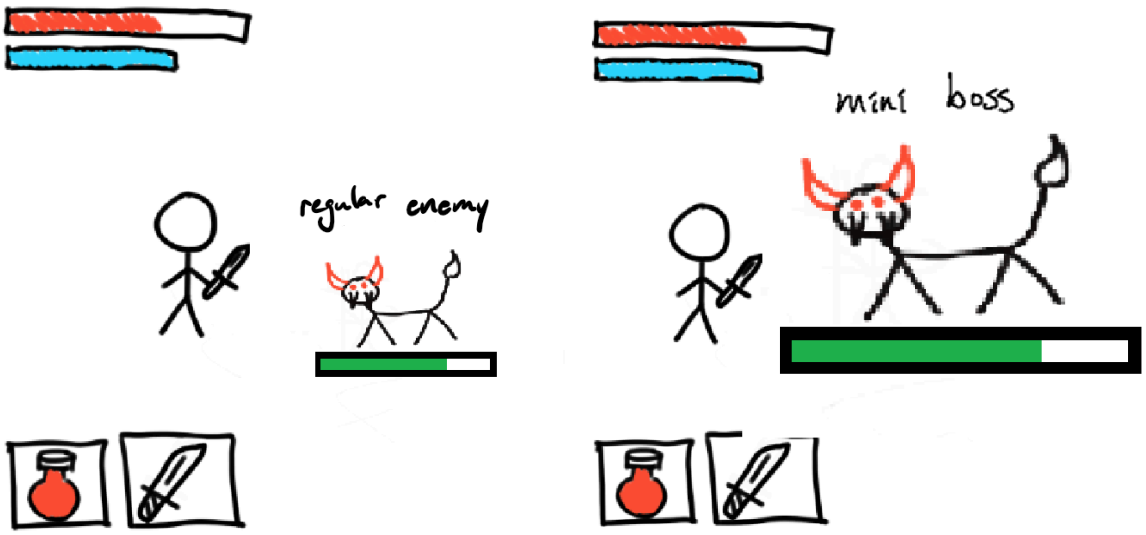


Scene 6: Mini-Boss Encounter Scene

The *Mini-Boss Encounter Scene* showcases battles with mini-bosses that appear in dungeons. These bosses scale in difficulty based on the dungeon's distance from the player's starting point.

Player Interaction:

- **Entering a Mini-Boss Room:** As the player progresses deeper into a dungeon, they encounter a mini-boss room. Visual cues such as increased darkness and a change in music signal the upcoming battle.
- **Combat Mechanics:** The mini-boss has higher health and damage than regular enemies, and its abilities scale based on dungeon proximity (e.g., further dungeons have more powerful mini-bosses).
- **Using Abilities:** Players could use dodging, time dilation, and ranged/melee attacks strategically to defeat the mini-boss. Players are encouraged to exploit patterns in the boss's attacks.
- **Victory:** Upon defeating the mini-boss, the player receives a large amount of loot, including weapon upgrades and consumables. A prompt indicates the rewards.

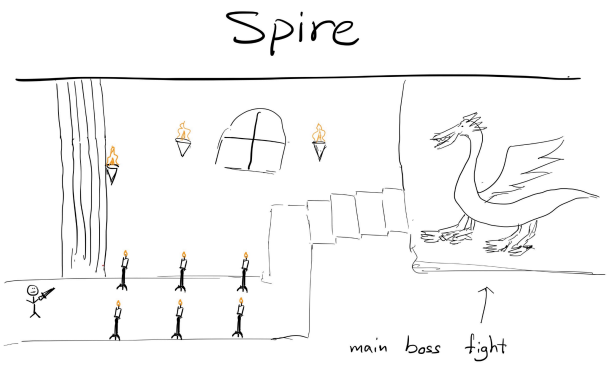


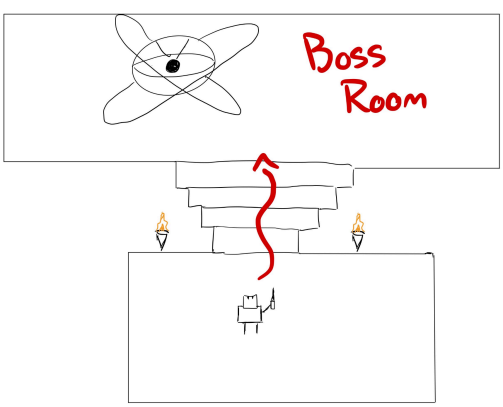
Scene 7: Spire Encounters

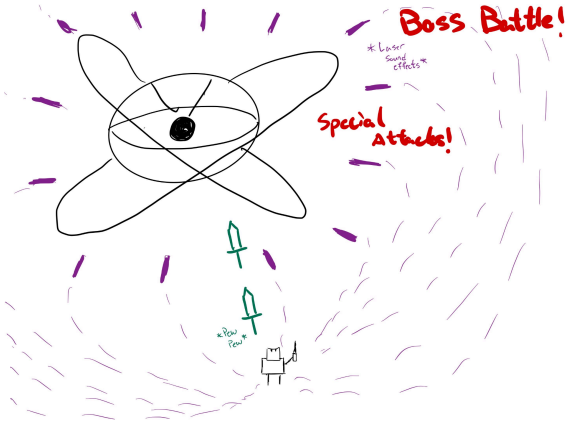
Spires represent the pinnacle of challenge within *Seekers*. These towering structures house the main bosses of the game, who guard the keys to the next spire. Unlike dungeons, spires are not procedurally generated. Instead, each is meticulously designed, with unique environments and boss mechanics that test the player's abilities. There are five spires in total, each one housing progressively harder bosses with more complex mechanics and environmental challenges.

Player Interaction:

- **Entering a Spire**: To enter a spire, the player must possess a Spire Key. As the player approaches the spire door, a prompt ("Press 'F' to unlock with Spire Key") appears.
- **Combat**: The Spire Boss fights are multi-phase encounters that require the player to learn unique attack patterns. For example, one Spire Boss might summon minions while another manipulates the environment (e.g., collapsing parts of the floor or triggering lightning strikes).
- **Player Abilities**: The player uses time dilation, dodging, and a variety of attacks to survive each encounter. Timing is crucial, and dodging at the right moment can make or break the fight.
- **Rewards**: Upon victory, the boss drops rare loot and a key to the next spire. The scene transitions with a celebratory sound and visual effect as the player unlocks the next level of progression.





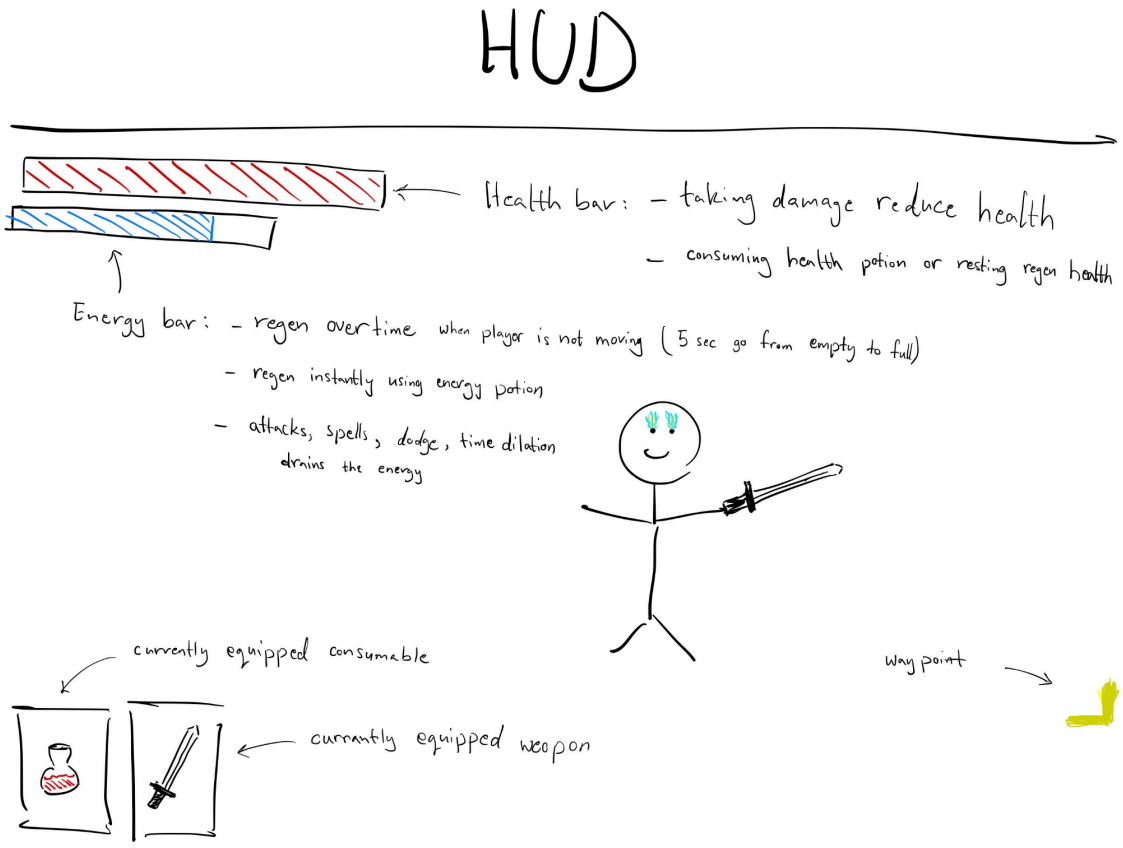


Scene 8: HUD (Heads-Up Display)

The HUD is minimalist but functional, giving players vital information without cluttering the screen. It displays the player's health, energy, equipped weapon, consumables, and a compass waypoint pointing toward key locations marked by the player (e.g., dungeons, spires).

Player Interaction:

- **Health and Energy Management:** Players monitor their health and energy bars, which deplete as they take damage or use special abilities.
- I**nventory and Equipment**: The HUD shows quick access to consumables, which players can activate using the number keys. Weapon and armor icons are displayed, with visual cues for durability or enhancements.
- **Navigation**: A waypoint marker is displayed as an arrow to help the player navigate the map.

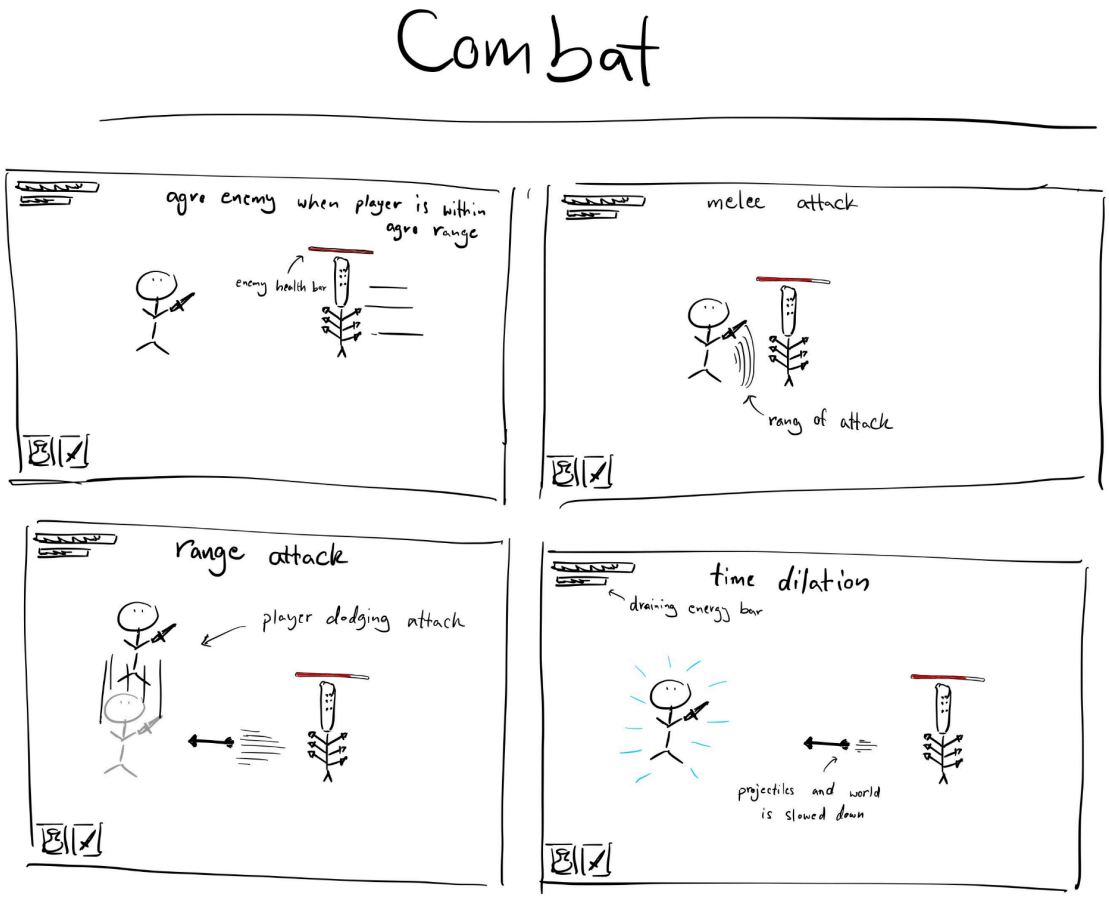


Scene 9: Combat and Abilities

Combat in *Seekers* is fast-paced and skill-based. Players must manage their stamina while attacking enemies using a variety of weapons and abilities. The *Seeker* can switch between melee and ranged weapons on the fly, and the use of time dilation adds a tactical layer to combat. During time dilation, enemies slow down, allowing the player to maneuver around or execute critical attacks.

Player Interaction:

- **Melee and Ranged Combat**: Left-click for light attacks and right-click for heavy attacks. Light attacks are faster, consume less energy, but deal less damage compared to heavy attacks. Each weapon has a unique range and projectile pattern.
- **Abilities**: Time dilation is activated by holding down the shift key, allowing the player to slow down time. During this phase, the player can dodge incoming attacks or land precise hits.
- **Visual and Audio Feedback**: When the player lands a successful hit, enemies stagger, and a sound effect signifies damage. Critical hits show additional particle effects to highlight their importance.
- **Enemy Behavior**: Enemies react dynamically based on proximity and attack style, and bosses introduce more intricate mechanics, requiring timing and strategy.

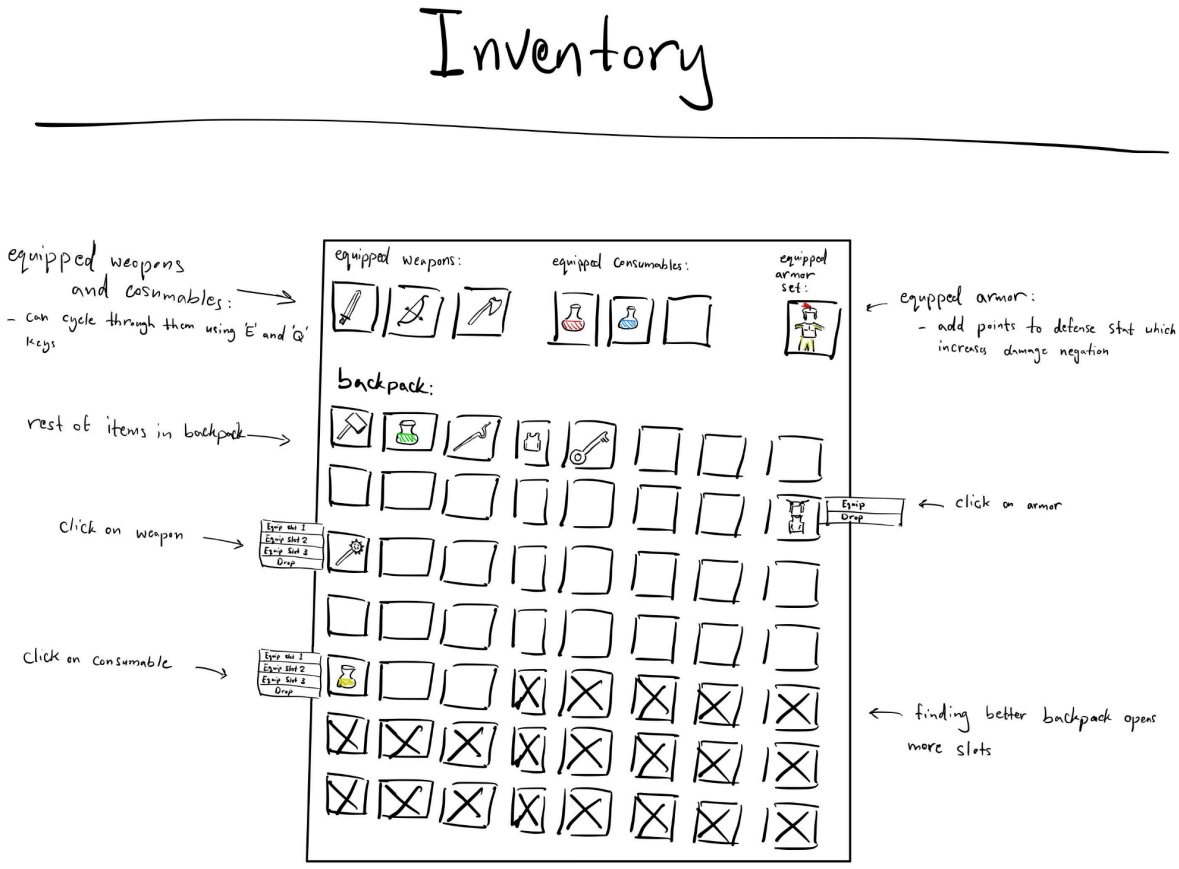


Scene 10: Inventory Management

The inventory is a key component of player progression. Players can view their collected items, equip different weapons or armor, and assign consumables to quick slots.

Player Interaction:

- **Item Management**: Players press ‘I’ to open the inventory. From here, they can equip gear by clicking the item and selecting the slot they want to move it into.
- **Buffs and Stats**: The inventory screen also displays the player’s stats, showing their current health, energy, buffs, and debuffs.

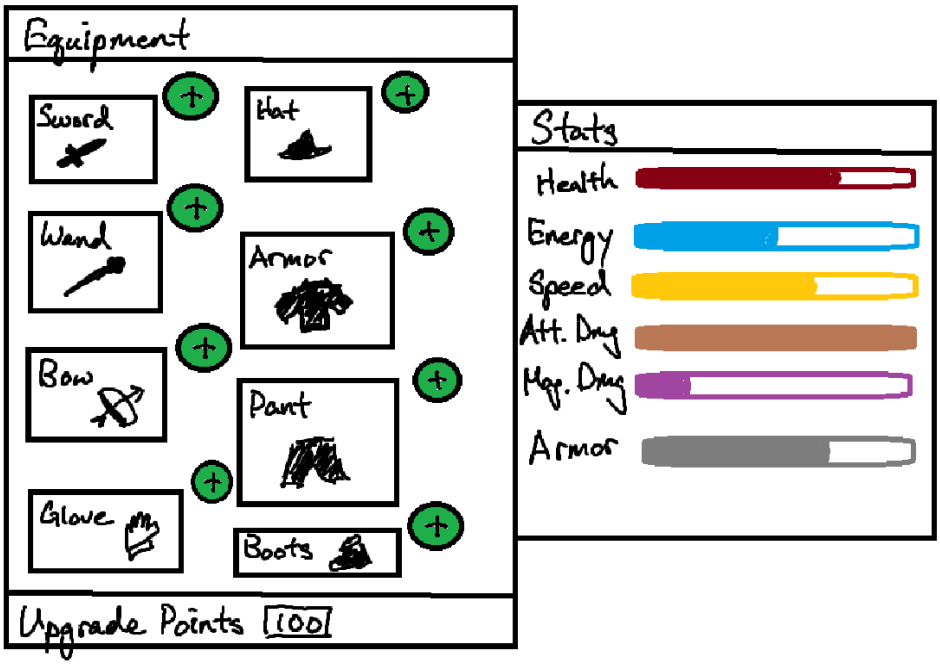


Scene 11: Player Progression and Level-Up Scene

In *Seekers*, player progression is based on equipment upgrades and defeating spire bosses, rather than a traditional experience points system. Players upgrade their weapons and armor through looting in dungeons and across the overworld.

Player Interaction:

- **Weapon Upgrades:** Weapons (sword, bow, and ward) can be upgraded with different stats such as attack damage, speed, range, and size. The player collects these upgrades through loot found in the surface, dungeons, or spires.
- **Armor Upgrades:** Body armor can also be upgraded through loot, increasing the player's defense and reducing damage taken from enemies.
- **Spire Boss Progression:** Defeating a Spire Boss unlocks access to the next spire and tougher dungeons, allowing the player to continue progressing through more difficult content.
- **Elemental Variants:** Weapons and armor may also have elemental effects (fire, snow), providing additional strategic options during combat. For example, fire projectiles would cause less damage initially, but catching a fire would reduce health incrementally over time. Snow projectiles would cause the player or enemy to freeze for some time.

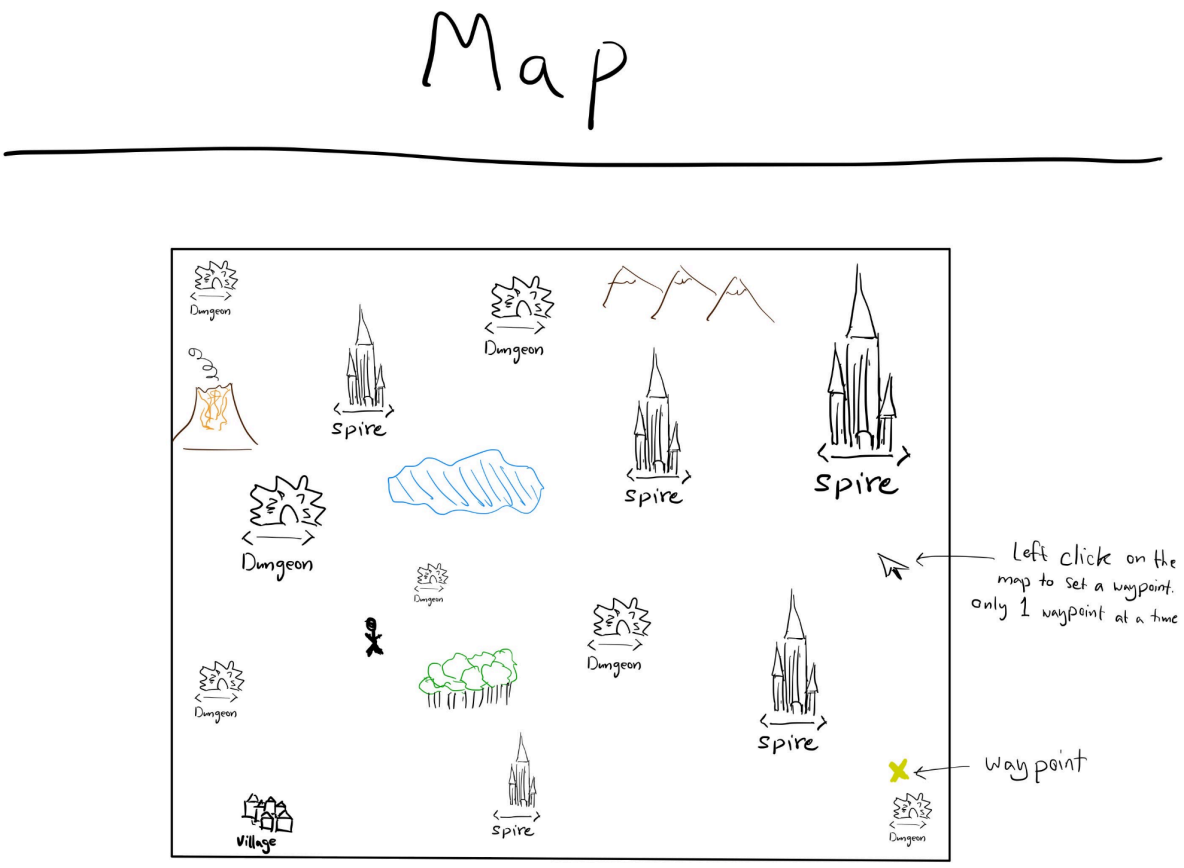


Scene 12: Map

The map provides an overview of the player's surroundings, showing key locations such as settlements, dungeons, and spires. As players explore *Eldoria*, the map dynamically updates to reflect newly discovered areas, making it easier to track progress and plan their next move.

Player Interaction:

- **Map Access**: Players press **'M'** to open the map. The game pauses while the player navigates the map.
- **Waypoints**: Players can set custom waypoints by clicking on the map, guiding them to specific areas of interest.



Scene 13: Dynamic Weather System

Weather in *Seekers* is a dynamic, ever-changing element that directly affects gameplay. Rain, snow, and day/night cycles alter the environment, influencing the player's movement, visibility, and enemy behavior.

Player Interaction:

- **Weather Effects**: The player experiences different weather conditions that impact movement and visibility. Rain may reduce movement speed or make certain areas difficult to traverse, while snow might cause the ground to become slippery.
- **Day/Night Cycle**: The time of day affects gameplay, as nighttime might make enemies faster or affect the player's line of sight, requiring a more cautious approach to exploration.
- **Adaptation:** Players must adapt to changing weather conditions and day/night cycle. For example, during a rainstorm, ranged attacks might be less effective due to reduced visibility, forcing players to rely on melee.

Scene 14: Pause Menu

The pause menu is accessible at any point in the game, providing players with the ability to adjust settings, save progress, or quit. The menu also displays a summary of the player's current objective, ensuring they stay focused on the task at hand.

Player Interaction:

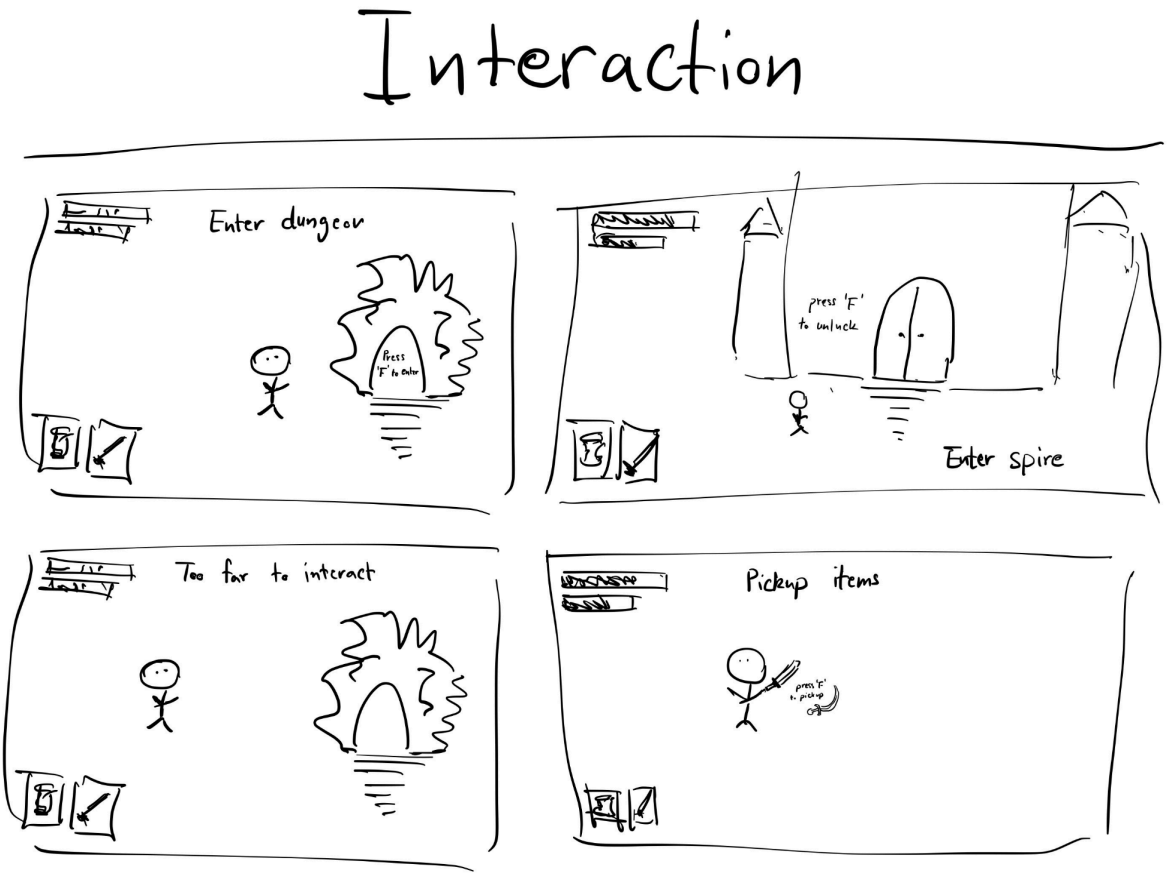
- **Resume**: The player can click the Resume button to immediately return to gameplay. This option is always placed at the top for quick access, ensuring that players can easily get back to action without navigating through complex menus.
- **Settings**: Clicking the Settings button brings up a sub-menu where the player can adjust audio levels, key bindings, and other gameplay options. This allows players to fine-tune their experience without having to quit the game or restart it. Within this section:
    - **Audio Controls**: Audio sliders allow for individual control of music, sound effects, and voiceovers.
    - **Key Bindings**: Customize movement, combat, and interaction keys based on personal preferences.
- **Save and Quit**: Players can choose to save their current progress and exit the game from this option. When selecting this, a confirmation dialogue appears, asking the player to confirm if they wish to save before quitting or continue without saving. This prevents accidental loss of progress. Saving the game updates the player's current state, including location, inventory, and progression through dungeons and spires.

Scene 15: Interaction

Interaction within *Seekers* is an essential part of the player's journey, allowing them to engage with the environment, enter dungeons and spires, pick up items, and unlock new areas. This scene captures the mechanics and prompts that appear when the player encounters interactive elements in the world.

Player Interaction:

- **Dungeon Interaction:** As the player nears a dungeon entrance, a prompt appears ("Press ‘F’ to Enter"). The prompt only appears when the player is close enough to the entrance.
- **Spire Interaction:** Players use Spire Keys to unlock spires. A visual prompt ("Press ‘F’ to Unlock") appears when the player has the key, otherwise, a message such as "You need a Spire Key" is shown.
- **Looting:** Loot dropped by enemies is highlighted on the ground, with a "Press ‘F’ to Pick Up" prompt appearing when the player is within range.



Scene 16: Player Death and Revive Scene

The *Player Death and Revive Scene* occurs when the Seeker's health is fully depleted during combat or exploration. *Seekers* employs a permadeath mechanic, where players lose most of their inventory and progress, but retain any global buffs earned from defeating Spire Bosses. The scene helps communicate the stakes of dying while also encouraging players to continue their journey with enhanced abilities in the form of buffs.

Player Interaction:

- When the player's health reaches zero, the screen gradually fades to black. A death summary screen appears, listing the player's stats for the current run, such as total enemies defeated, dungeons completed, and time survived. The player can navigate through this summary with the **Arrow Keys** or **Mouse Scroll**.
- After reviewing their stats, the player is prompted to press **Enter** to "Revive as New Seeker." Upon reviving, the player retains global buffs, such as increased health or attack power, earned by defeating Spire Bosses. All other items and equipment are lost.
- The game transitions to a cutscene where a new Seeker is shown awakening in the starting area. The player can resume exploration using the **WASD** keys for movement.
- The HUD updates to reflect the global buffs that have been retained, indicated by icons showing enhanced health or damage resistance. Players can press **I** to access their inventory and check any starting gear they have been reissued.

This system balances the consequences of death with continued progression, rewarding players for their persistence while encouraging careful planning in future runs.

Scene 17: Endgame Victory Scene

The *Endgame Victory Scene* takes place after the player defeats the final Spire Boss, signifying the culmination of their journey in *Eldoria*. This scene provides closure by showing the narrative consequences of the Seeker's actions and summarizes the player's accomplishments throughout the game.

Player Interaction:

- Upon defeating the final Spire Boss, the player delivers the last blow with their attack. A dramatic cutscene plays immediately after, showcasing the Spire's destruction and the resolution of the game's central conflict. The scene uses visual effects such as the collapse of the Spire and beams of light restoring balance to the land.
- The screen transitions to a "Victory!" message, and the player is shown a detailed summary of their journey. This includes total enemies defeated, dungeons cleared, spires conquered, and loot collected. The player can scroll through this summary using the **Arrow Keys** or **Mouse Scroll**.
- After reviewing the summary, the player is prompted with a **Main Menu** where the player has the option to **Start a New Game** (optionally with higher difficulty) or **Exit** the game.

This final scene provides both narrative closure and a sense of accomplishment for the player. It celebrates their progress while also offering replayability through the option to start a new game.

Essential Technical Elements:

Our game will satisfy the core technical requirements as follows:

Rendering

- **2D Rendering with OpenGL:**
    - Use OpenGL to render sprites and textures in a layered fashion.
    - Implement correct rendering order (background, environment, characters, UI).

Assets

- **Sprites and Textures:**
    - Create 2D sprites for characters, enemies, items, and environments.
    - Use procedural generation to generate maps and levels from assets.

2D Geometry Manipulation

- **Transformations:**
    - Apply translation, rotation, and scaling to game entities.
    - Camera rotation (with Q and E keys)
    - Scale projectiles dynamically (e.g., growing or shrinking over time).
    - Scale player size based on buffs or power-ups.
- **Collision Detection and Resolution:**
    - Implement Axis-Aligned Bounding Box (AABB) collision detection for rectangular sprites.
    - Handle collision responses between the player, enemies, and environment.
- **Particle System:**
    - Generate and manage particle effects for environmental elements (e.g., rain, snow, fog, time dilation).
    - Splashing effect when stepping on puddles.
    - Particle effects for attacks and other visual feedback.
- **Procedural Generation:**
    - Generate and manipulate 2D geometry for procedural level creation (dungeons, surface world).
    - Create algorithms for placing objects, enemies, and obstacles in generated levels.
- **Line of Sight and Vision Radius:**
    - Use raycasting to calculate the enemy's line of sight for aggro.
    - Adjust the player’s vision radius based on weather conditions (such as fog).

Gameplay Logic/AI

- **Enemy AI:**
    - Use state machines to manage enemy behaviors (idle, patrol, chase, attack).
    - Basic decision-making for enemies to react to player actions.
    - Aggro and de-aggro mechanics based on player proximity and actions.

    - Different AI behaviors for various enemy types (melee, ranged, boss).
- **Boss AI:**
    - State machines for boss behaviors with multiple attack patterns.
- **Game Difficulty:**
    - Increase game difficulty after game completion for replayability (easy, normal, hard).
    - Progressive difficulty scaling for dungeons and spires based on distance from the center in the surface world.
    - Adjust enemy stats based on the current difficulty.
- **Procedural Generation:**
    - Algorithm for generating diverse and balanced surface world layouts.
    - Algorithm for populating the world with appropriate enemies and resources.
    - Dungeon generation algorithms with increasing complexity for higher difficulties.
- **Weather and Time System:**
    - Create a day/night cycle that affects gameplay mechanics and enemy behaviors.
    - Implement a weather system with various effects (rain, heat, winter) that impact both player and enemies.
    - Develop logic for transitioning between weather states and time of day.
- **Save/Load System:**
    - Develop a system to save and load game state, including player progress and world state.
- **Permadeath and Character Generation:**
    - Create a system for generating new Seeker characters upon player death.
    - Implement logic for carrying over global buffs and maintaining overall game progress.
- **Inventory and Item Management:**
    - Create an inventory system with slot limitations (backpacks increase inventory by 1 slot).
    - Item pickup, use, and discard mechanics.
- **Environmental Interactions:**
    - Take damage in fires.
    - Walk slowly through puddles.
    - Slip on ice.
    - Interactive object like chest, dungeon entrance, spire entrance

Physics

- **Movement Mechanics:**
    - Implement smooth player movement with acceleration and deceleration.
    - Apply basic physics principles like friction for realistic motion (based on weather).
    - Create sliding effects on icy surfaces.
    - Knockback effects from strong enemy attacks
    - Certain boss has black hole attack with a localized gravitational pull affecting both players and enemies.
- **Projectile Physics:**

    - Implement scaling effects for certain abilities (eg. Time Dilation on entity's motions).
    - Homing projectiles with adjustable turn rates and acceleration for certain boss fights.
- **Weather based:**
    - Snow accumulation effects that gradually slow down movement in affected areas.

Sound

- Add sounds using open-source or free libraries to provide player feedback (movement, weather, projectile, damage, abilities, interacting with chest or doors).
- Audio cues for player health status (low health warning, healing sounds).
- Sound effects for the weather system (rain, wind, thunder, snow).
- day/night cycle audio changes (crickets at night, birds in the morning).
- sounds for different types of attacks (melee, projectile, magic).
- Audio cues for menu navigation and selection.
- Sounds for picking up items, opening inventory, and equipping gear.
- Create audio feedback for successful/failed actions (time-dilation, not enough energy)
- Develop a variety of enemy sound effects (idle noises, aggro sounds, attack grunts).
- Create unique sound sets for boss encounters.
- Add audio cues for resource regeneration (healing, energy recharging).

Advanced Technical Elements:

We plan to include the following advanced technical elements:

1. Procedural Generation of Dungeons

- **Implementation:**
    - Use algorithms like Binary Space Partitioning (BSP) or random walk to generate dungeon layouts.
    - Create a set of room templates and connect them randomly to form unique dungeons.
- **Impact if Skipped:**
    - Without procedural generation, dungeons would be static, reducing replayability.
    - **Alternative:** Design several predefined dungeon layouts to offer some variety.

2. Complex Enemy AI (Priority for Inclusion)

- **Implementation:**
    - Enhance enemy AI with pathfinding algorithms (like A* for navigation).
    - Introduce enemies with varied behaviors, such as ranged attackers or melee.
    - Implement boss AI with multiple phases and attack patterns.
- **Impact if Skipped:**
    - Enemies would have simpler behaviors, possibly making the game less challenging.
    - **Alternative:** Focus on refining basic AI to be more engaging.

3. Dynamic Weather System (Optional)

- **Implementation:**
    - Simulate weather changes over time, affecting visuals and gameplay.
    - Implement effects like rain or ice that impact player visibility and movement.
- **Impact if Skipped:**
    - The environment would remain static, but core gameplay remains unaffected.
    - **Alternative:** Use static environmental effects for aesthetic purposes.

Devices and Control Mapping:

Controls are designed for intuitive gameplay, allowing players to move, attack, and interact seamlessly.

Device	**Action**	**Control**
Keyboard	Movement	WASD
	Rotate Camera Clockwise	'E' key
	Rotate Camera Counter-Clockwise	'Q' key
	Interaction	'F' key
	Time Dilation	Hold 'Shift' key
	Dodge/Dash	'Space' bar
	Open Inventory	'I' key
	Open Map	'M' key
	Quick Switch Weapon	'Tab' key
	Use Consumable	'1' through '0' number keys
	Pause/Menu	'Esc' key
Mouse	Aiming	Player's projectiles and attacks go toward the mouse cursor
	Primary Attack (Melee/Range): *Quick attack, consumes less energy, standard damage*	Left mouse button
	Heavy Attack (Melee/Range): *Slow attack, consumes more energy, causes more damage*	Right mouse button
	Menu Navigation/Inventory Selection	Click to select options in menus

Tools:

Since game engines are not permitted, we will use the following libraries and tools compatible with C++ and OpenGL:

Programming Language

- **C++**

Graphics Libraries

- **OpenGL:**
  - For rendering 2D graphics and handling shaders.
- **GLFW (Optional):**
  - For window creation, context management, and input handling.
- **GLEW (Optional):**
  - For managing OpenGL extensions.

Audio Library

- **SDL2_mixer** or **OpenAL:** For playing sound effects and music.

Image Loading

- **stb_image.h:** For loading images into textures.

Development Tools

- **IDE:** Visual Studio or CLion for coding and debugging.
- **Version Control:** Git with GitHub for collaborative development and version tracking.

Asset Creation

- Open-source and free available assets.
- Create our own assets using those possible tools:
  - **GraphicsGale** or **GIMP:**
    - For creating and editing sprites and textures.
  - **Tiled Map Editor:**
    - For designing level layouts and exporting them for use in the game.
  - **Pixelorama**:
    - For asset creation in general.

Project Management

- **GitHub Projects:** For task assignments, tracking progress, and setting deadlines.
- **Discord:** For team communication and coordination.

Team management:

We will collaborate as a unified team, with tasks assigned based on individual strengths and interests. Roles will be flexible to allow team members to assist each other and adapt as needed.

- **Task Tracking:**
    - Use project management tools (GitHub Projects) to assign tasks and monitor progress.
- **Communication:**
    - Hold regular team meetings to discuss updates and address challenges.
    - Use communication platforms (Discord) for ongoing discussions.
- **Policies:**
    - Implement code reviews to maintain code quality and consistency.
    - Document code, assets, and decisions to ensure clarity.
    - Prioritize meeting milestone deadlines and course requirements.

For our project, the team has been strategically assigned to various task categories to leverage their strengths and expertise.

Jakob and Ahmed will be focusing on the **Game and Story** development, focusing on elements such as boss design, spire design, map creation, dungeon and spire placement, dialogue, and the ending cutscene.

Amir and Brendon are responsible for **Movement and Interaction**, ensuring smooth character control, dash/dodge mechanics, and item pickup functionality.

Samuel and Amir will be looking after **Game Mechanics**, which includes collision detection, time dilation, day/night cycle, weather cycle, weather buffs and debuffs like rain, and the level-up system.

Sound Design is managed by Ahmed and Brendon, who are crafting the audio experience.

Visual Assets Design is handled by Samuel and Jakob, who are creating the HUD, inventory, weapons, spire keys, consumables, character models, enemy models, boss models, and both dungeon and spire assets.

Lastly, the entire team will collaborate on **Rendering**, working together on the rendering pipeline and mesh development. This clear division of roles ensures that each aspect of the project is developed with expertise and precision.

Development Plan:

Milestone 1: Skeletal Game

Overview:

Week 1

- **All Members:**
  - Set up the development environment with C++, OpenGL, and necessary libraries.
  - Establish project structure in GitHub.
- **Tasks:**
  - Implement basic window creation and rendering loop
  - Display a simple sprite on the screen
  - Establish the ECS system for the game
  - Implement updateCoordinate system for moving entities
  - Handle basic keyboard input for player movement, dodge, and attack

Week 2

- **All Members:**
  - Implement basic collision detection between the player and the environment.
  - Create placeholder sprites for the player and a basic enemy.
  - Prepare initial game scenes: main menu and gameplay area.

Milestone Description:

For Milestone 1, we will implement the foundational mechanics for our game. First, we will establish the rendering pipeline, ensuring that all in-game textures are rendered in layers, with tiles beneath sprites. We will create simple textures for the default ground and wall tiles, using them to build a small hardcoded sandbox map for testing. Following that, we will add a basic player sprite and enemy sprite to test movement and attack controls. Once everything renders correctly, we will create a default projectile sprite to implement the attack system. At this stage, the player should be able to move using the WASD keys, dodge with the space bar, and attack with the mouse's left click. When the player clicks, a projectile will fire from their position toward the mouse cursor.

Next, we will implement a bounding box and collision system for the player, enemy, projectile, and wall tiles. This will ensure that neither the player nor the enemy can pass through walls or each other, and that projectiles properly collide with the enemy, reducing their health bar (displayed near their sprite). We will then center the camera on the player, ensuring it moves as the player moves. Additionally, we will implement controls allowing the player to rotate their character and camera, ensuring that the player's sprite remains oriented toward the top of the screen, while tile orientation changes with respect to the player's rotation. Finally, we will add scalable entity sprites, providing visual feedback for buffs (e.g., when the player drinks a strength potion, their sprite will increase in size).

Testing checklist:

Rendering Pipeline:

- Verify that the basic rendering pipeline is functional
- Confirm that layers are rendered in the correct order (background, entities, UI)

Textures and Sprites:

- Check that the default ground tile texture is visible
- Confirm that the wall tile texture is visible
- Verify that the player sprite is rendered correctly
- Ensure that the enemy sprite is rendered correctly
- Confirm that the projectile sprite is visible when fired

Map Creation:

- Verify that the hardcoded sandbox map is generated correctly
- Ensure that wall and ground tiles are placed in the intended positions

Player Controls:

- Test WASD keys for player movement in all four directions
- Verify that the left mouse click fires a projectile
- Confirm that projectiles fire toward the mouse cursor position

Collision Detection:

- Test that the player cannot walk through wall tiles
- Verify that the enemy cannot walk through wall tiles
- Ensure that the player and enemy cannot walk through each other
- Confirm that projectiles collide with enemies
- Check that projectiles disappear upon collision with walls or enemies

Combat System:

- Verify that projectiles deal damage to enemies when they collide
- Ensure that the enemy's health bar decreases visually when hit
- Test that enemies are destroyed/disappear when their health reaches zero

Camera System:

- Confirm that the camera is centered on the player's position
- Verify that the camera moves smoothly with the player

Rotation Mechanics:

- Test player rotation controls (e.g., Q and E keys)

- Verify that the player sprite maintains its orientation relative to the camera
- Ensure that the environment (tiles) rotates correctly around the player

Sprite Scaling:

- Implement a test buff (e.g., strength potion)
- Verify that the player sprite increases in size when the buff is applied
- Ensure that the bounding box adjusts to sprite scaling affects, and that collision detection does not break

Performance:

- Check that the game maintains a stable frame rate during normal gameplay
- Verify that there are no memory leaks during extended play sessions

Error Handling:

- Test for graceful handling of missing assets (textures, sprites)
- Verify that the game doesn't crash on common error scenarios

UI Elements:

- Confirm that the enemy health bar is visible and positioned correctly
- Verify that the health bar accurately reflects the enemy's current health

---

Milestone 2: Minimal Playability

Overview:

Week 1

- **All Members:**
    - Implement inventory system
    - Allow the player to interact with simple objects (e.g., pick up an item).
    - Develop basic enemy AI with state machines (idle, chase, attack).
    - Implement procedural generation for simple dungeon layouts.
    - Introduce new assets for environments and enemies.
    - Add sprite sheet animations for the player and enemies.

Week 2

- **All Members:**
    - Implement mesh-based collision detection for irregular shapes.
    - Provide a basic tutorial or help screen accessible from the main menu.
    - Display an FPS counter in the game window title or on-screen.
    - Ensure at least 2 minutes of non-repetitive gameplay.

Milestone Description:

For Milestone 2, the goal is to create a playable demo that showcases our core gameplay loop. First, we will implement a simple AI for enemies using a state machine with four states: idle, patrol, chase, and attack. In the idle state, enemies will wander randomly within a defined circle around their spawn point. In patrol mode, enemies will follow a predefined path marked by waypoint coordinates, moving from one waypoint to the next and retracing their steps once they reach the final waypoint. In chase mode, enemies will pursue the player's coordinates, but with slight randomization in movement to simulate searching for the player after losing sight. If the enemy fails to catch the player after a set time, they will revert to either idle or patrol. In the attack state, enemies will maintain a tactical distance based on both their attack range and the player's, staying close enough to attack while avoiding the player's counterattacks.

We will then improve the sandbox map by populating it with various enemy types to test the gameplay. Next, we will develop the weapon system, which will include five key stats: attack damage, speed, range, velocity, and width. These stats will add variety by altering the behavior and characteristics of projectiles, since all attacks will use projectiles. Weapons will also be enchantable with either fire or ice, which will apply a burn or slowing effect respectively. The core weapon types will be sword, bow, and magic wand. Swords will fire thick, short-range projectiles, bows will fire thin, long-range projectiles, and wands will shoot multiple medium-sized projectiles in a cone. These weapon implementations will also be used to create variation among enemy types (e.g., warrior, archer, wizard), based on the weapon they use.

At this stage, we will implement the inventory and player/enemy stats systems, which are necessary precursors for the weapon system. Following that, we will debut a weather system that adds dynamic conditions to gameplay, using a randomized weather cycle. Each cycle will have a randomized duration and type, with four weather conditions: sunny, rainy, snowing, and foggy. Sunny weather will apply no modifiers, while rainy weather will slow the player and cause enemies to heal over time. During snow, water tiles will freeze, allowing the player to walk over them but introducing a sliding effect when moving on ice. Additionally, all enemies' weapons will temporarily gain an ice enchantment during snow cycles. Foggy weather will limit player visibility to a circle around their position, with the radius depending on the fog's intensity.

Finally, we will introduce the core dungeon feature by hardcoding a few dungeons for now. If time permits, we will attempt to implement procedural generation, adding variability and new experiences to each dungeon. The dungeon will be filled with regular enemies, and feature a mini-boss at the end. The mini-boss will be a larger, super-charged variation of a regular enemy, with potential to use a special attack later down the line (if time permits).

Testing checklist:

Enemy AI State Machine:

- Verify that enemies can transition between all four states (idle, patrol, chase, attack)
- Test idle state: enemies wander randomly within a defined circle
- Check patrol state: enemies follow pre-assigned waypoints correctly

- Confirm chase state: enemies move towards the player with slight randomization
- Verify attack state: enemies maintain appropriate distance based on attack ranges

Enhanced Sandbox Environment:

- Confirm various enemy types are placed correctly in the sandbox map
- Test interactions between different enemy types and the player

Weapon System:

- Verify all 5 weapon stats are implemented: attack damage, speed, range, velocity, and width
- Test each stats effect on projectile behavior
- Confirm fire and ice enchantments are working (burn and slow effects)
- Verify the three core weapon types (sword, bow, magic wand) function correctly
- Test different projectile behaviors for each weapon type

Enemy Variation:

- Confirm enemies use different weapon types (warrior, archer, wizard)
- Verify enemy behavior changes based on their weapon type

Weather System:

- Test randomized weather cycle generation
- Verify weather duration is within the specified bounds
- Check all four weather types (sunny, rainy, snowing, foggy) occur
- Test sunny weather (no modifiers)
- Verify rainy weather slows player movement and heals enemies over time
- Confirm snowing weather freezes water tiles and allows player to walk on them
- Test sliding effect on ice tiles during snow weather
- Verify enemies gain ice enchantment during snow weather
- Check foggy weather reduces player vision to a circular area
- Test different fog intensities affect vision radius

Dungeon Feature:

- Verify hardcoded dungeons are generated correctly
- If implemented, test procedural dungeon generation for variety and playability

Inventory System:

- Confirm inventory UI is visible and functional
- Test adding and removing items from the inventory
- Verify weapon switching through inventory
- Check inventory capacity limits (if applicable)

Gameplay Loop:

- Test the overall game flow from start to finish of a dungeon
- Verify the balance between player abilities and enemy difficulty
- Check if core gameplay elements (combat, exploration, weather effects) work together cohesively

Performance:

- Test frame rate stability with multiple enemies and weather effects active
- Check for memory leaks during extended play sessions

User Interface:

- Verify all necessary information is displayed (health, inventory, current weapon, etc.)
- Test UI responsiveness to game state changes

Controls:

- Confirm all player actions (movement, attacking, inventory management) are responsive
- Test weapon switching mechanics

Audio:

- Verify sound effects for different weapons and enemy types
- Check for weather-specific audio cues

Visual Effects:

- Test visual feedback for different weapon types and enchantments
- Verify weather effects are visually distinct and clear

Save/Load System (if implemented):

- Test saving game state (player position, inventory, dungeon progress)
- Verify loading saved games restores all relevant information

---

Milestone 3: Playability

Overview:

Week 1

- **All Members:**
    - Optimize game performance and fix bugs identified in previous milestones.
    - Enhance enemy AI with more complex behaviors.

- Add additional assets and refine animations.
- Implement basic physics-based interactions (e.g., projectile motion).

Week 2

- **All Members:**
    - Ensure at least 5 minutes of non-repetitive gameplay.
    - Conduct playtesting sessions to gather feedback.
    - Implement improvements based on player feedback.

Milestone Description:

The goal for Milestone 3 is to deliver a complete, robust, and playable game that sustains at least 5 minutes of non-repetitive gameplay. One of the focuses will be implementing procedural generation to any aspect of Milestone 2 that was hard coded such as the maps and different weather environments in the game. We will look into creative ways of modifying enemies' projectiles that apply physics-based animations to satisfy the advanced feature requirements. The development process will include optimizing game performance, enhancing enemy AI, and implementing save/load features for the game.

At this point, we will start working towards implementing the main storyline of our game. First, hardcode the spires (main story dungeons). We will try to make it deep and engaging, with many rooms so that players have a chance to explore and fight many enemies. Then for each spire, we will design a boss room. To satisfy this requirement, we will create a new set of assets to match the theme of the main story, and use them as the building blocks for this component.

Following from that, we will build our unique bosses for the main story, giving each a unique special ability that facilitates engaging action-packed gameplay. One such boss might be called Mograth, a giant flying eyeball that shoots projectiles in all directions, that uses a pathfinding algorithm that makes them chase the player. If time permits, we will add more special abilities for even more engaging battles.

Finally, we will implement our special game mechanic: the power to slow time. The player will be able to hold a key to slow down all enemies and projectiles, while they move freely at normal speed. This will make use of the energy resource bar, slowing all enemy and projectile speed components by an amount that is determined by the player’s stats. This will add a layer of complexity to gameplay, making it more exciting and interactive for the player.

Extensive playtesting, both internal and external, will be conducted to gather feedback and identify issues. We will prioritize robustness and stability, ensuring proper memory management, graceful handling of user inputs, and consistent performance. Additional testing will be carried out to prevent memory leaks, crashes, or glitches. If time constraints arise, alternative options (Plan B) include prioritizing core gameplay and stability over new features or implementing a selection of basic or advanced features from the Suggested Game Features list. Throughout the milestone, we will maintain flexibility to adjust their approach based on progress and feedback, ensuring the delivery of a high-quality game that meets all mandatory requirements and creative components.

Milestone 4: Final Game

Overview:

Week 1

- **All Members:**
    - Finalize all core features and fix any remaining bugs.
    - Implement optional advanced features if time permits (e.g., dynamic weather).
    - Polish the game with sound effects, music, and visual enhancements.
    - Develop a comprehensive in-game tutorial.

Week 2

- **All Members:**
    - Ensure at least 10 minutes of engaging, non-repetitive gameplay.
    - Conduct extensive playtesting and balance the game difficulty.
    - Prepare final documentation and presentation materials.

Milestone Description:

For Milestone 4, the focus will be on delivering a polished final version of the game that offers at least 10 minutes of continuous, non-repetitive gameplay. As a group, we will finalize all core features, address remaining bugs from previous milestones, and, if time permits, implement advanced features such as new special abilities. A comprehensive in-game tutorial will be developed to make the game self-explanatory, eliminating the need for external instructions. To enhance the user experience, we will add sound effects, music, and visual polish. Extensive playtesting will be conducted, with feedback from external players used to balance difficulty and make final adjustments.

We will also prioritize the game's stability by ensuring consistent resolution and aspect ratio, focusing on memory management, robust user input handling, and optimizing real-time performance. The bug list will be updated, reprioritized, and all critical issues will be resolved before the final submission. Additionally, a 6-minute video report will be created to showcase game features, highlight new creative elements, significant bug fixes, and demonstrate the game's evolution from concept to final release. This report will also include insights from user testing, feedback received, and changes made in response. Lastly, the team will maintain focus on delivering a high-quality, self-contained game that meets all mandatory requirements and incorporates creative components to enhance the overall gaming experience within the time permitted.

To finish the project, we will incorporate backstory through in-game cutscenes to enhance immersion, and then release it on Steam to make a Gazillion dollars.
