Game Proposal : **Watch Out!**
CPSC 427 - Video Game Programming

Team: **Electric Boogaloo**

Team members:

1) Carlo Villaceran (56557465)
2) Katie Louie (81317901)
3) Linus Moreau (12438479)
4) Tarun Narayan (12071601)
5) Yan Naing Win (97097349)

Story

In Watch Out!, players will assume the role of a lone survivor trapped in the dark, medieval world of Draugamoor, forced to face Aldric's brutal forces. Obsessed with power and superstition, Aldric forces prisoners into deadly outdoor arenas scattered across his empire. These are sprawling, open battlefields, surrounded by dense forests, jagged cliffs, and treacherous streams, where the remains of the fallen serve as grim reminders of the ruthless nature of the hunt. Under dark, oppressive skies, the eerie sounds of wildlife are mixed with the distant beat of war drums as enemies close in, determined to take you down. With no weapons at hand, survival relies solely on your agility and wits.

Gameplay

Watch Out is an arcade-style survival game focused on enemy mechanics and progression, set in a single, inescapable canyon that is procedurally generated. The playable area is roughly five times the size of the screen and there will be stationary obstacles that the player will have to move around/jump over. Since it is a valley, the basic enemies will come down from the cliffs, with more dangerous foes appearing as time passes. The player starts at the center of the and can run around to evade attacks, but there's a twist: the player cannot attack. Instead, enemies can harm each other through their own attacks. Survival depends on the player's ability to outmaneuver foes while luring them into damaging one another. The game ends when the player is defeated, and their high score, based on the time survived, will be recorded.

Perspective:

- top-down perspective to give players a comprehensive view of the arena.

Battleground:

- Open battlefield enclosed by high cliffs, forming a border that prevents players from escaping
- Entry points placed around the perimeter serve as locations where waves of enemies enter the battlefield

- Features diverse terrain that create obstacles and limit movement for characters

Main Character:

- A playable humanoid character with no weapons and can’t attack enemies directly
    - Movement and controls go into more detail in the devices section
- The player has a health bar that deteriorates with enemy damage
- The player has a stamina bar that lowers when rolling or sprinting is used
- The player can hold **unlimited** traps and items.
- The player has the ability to sprint, roll, jump and dash to avoid enemy damage.

Enemies and Types:

All enemies have a health bar that depletes when they take damage.
Can be harmed either by other enemies or by traps.

Boar

- A non-humanoid enemy that constantly charges at the player.
- If the boar collides with the player or an enemy, the player or enemy takes damage.

Barbarian

- A humanoid enemy that uses a sword.
- Relentlessly stalks the player, attempting to get close enough to perform a melee attack.
- If the player or any other enemy enters the barbarian's attack hitbox, they will take damage.

Archer

- A humanoid enemy that uses a bow and arrow.
- It is slow moving and periodically shoots arrows at the player from a distance.
- Both the player and other enemies hit by the arrows will take damage.

Magician

- A humanoid enemy that can throw fireballs or ice bolts.
- Can rain down shards from the sky that damage everyone within the targeted area
- Moves faster than an archer but has lower Health.
- Both the player and other enemies hit will take damage.

Bomber

- A humanoid enemy that throws bombs towards the player.
- A bomb has a radius in which characters within that radius will take damage.

Troll

- A large enemy with high health
- Turns slowly and may incidentally crush smaller characters underfoot
- Wields a big club

Dragon

- A non-humanoid enemy that spews red hot flame towards the player.

Items/Collectibles:
Items are collected by the player running over them
Traps

- Traps spawn as collectibles on the canvas.
- Player can set them along the way.
- The player or enemies lose health when they step on the placed trap.
- The trap gets destroyed when the enemy/player steps on it.

Heart

- Heart spawn as collectible on the canvas.
- Player collects Hearts to increase their health bar.
- Exclusive to the player; enemies cannot consume this.

Inventory:

- Player has an inventory of items that can be invoked at any time

Obstacles:

- Rock formations that block the way
- Shallow river that slows entity movement
- Deep river which is uncrossable
- Trees whose trunks block movement
- Cliffs (the border map will consist of unscalable cliffs)

Enemy spawning:

- Enemies spawn in a staggered/randomized manner until the maximum number of enemies is reached.
- The type of enemy spawned is dependent on the number of enemies present of each type (eg., if there are more barbarians than archers, the game will spawn an archer next to maintain enemy equilibrium).

Stamina:

- The player loses stamina while sprinting
- Rolling, dashing, jumping each consume a fixed amount of stamina
- Stamina recovers gradually over time
- The player cannot perform actions that require more stamina than they have

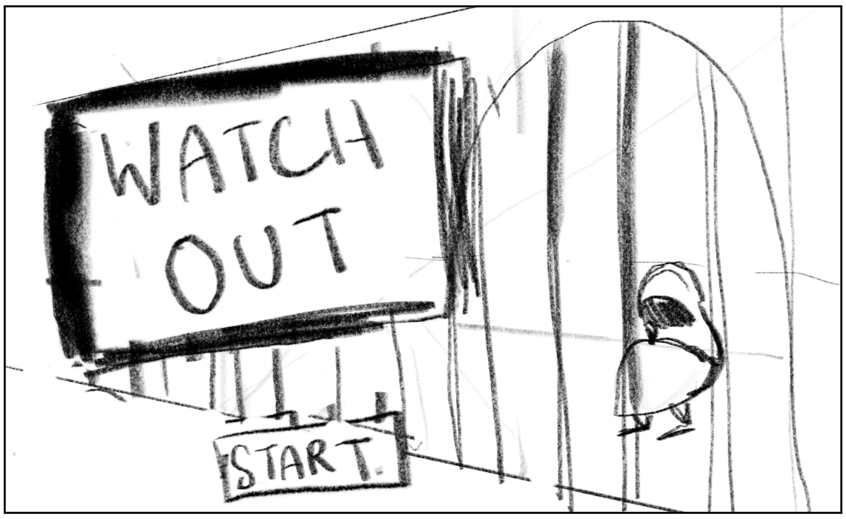
Goal:

- Use enemy aggression to get enemies to kill each other by utilizing their attacking patterns, and traps.
- Survive as long as possible.

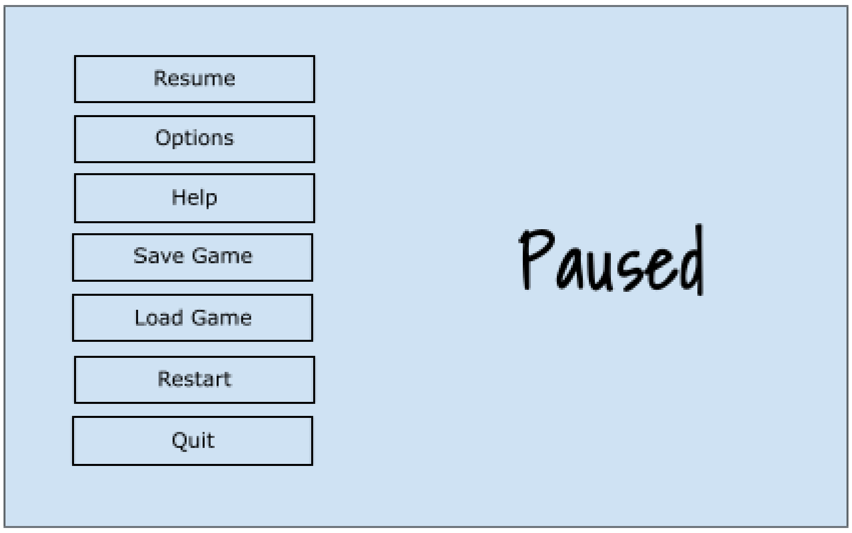
Scenes (Pictures)

The following illustrations showcase the game's setting and the core gameplay mechanics.

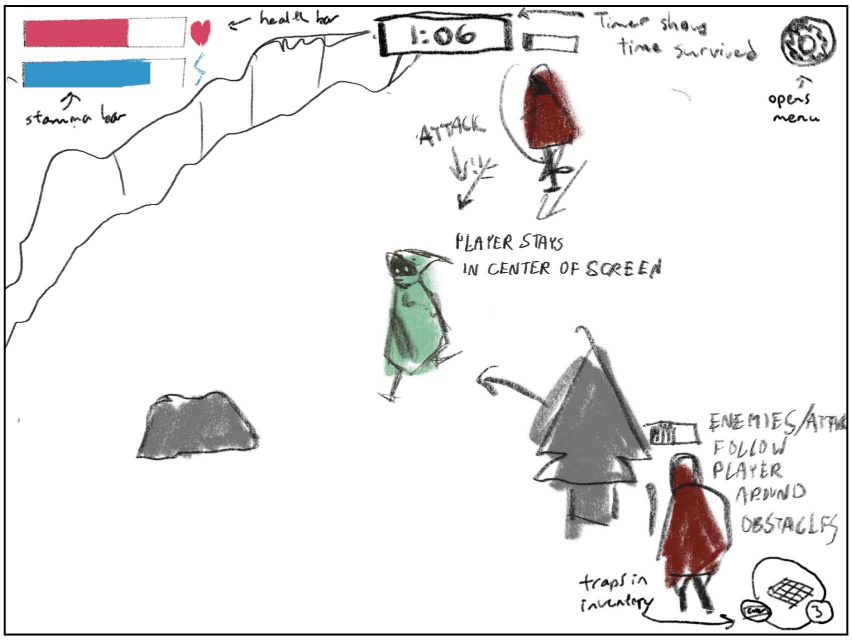
Start Screen



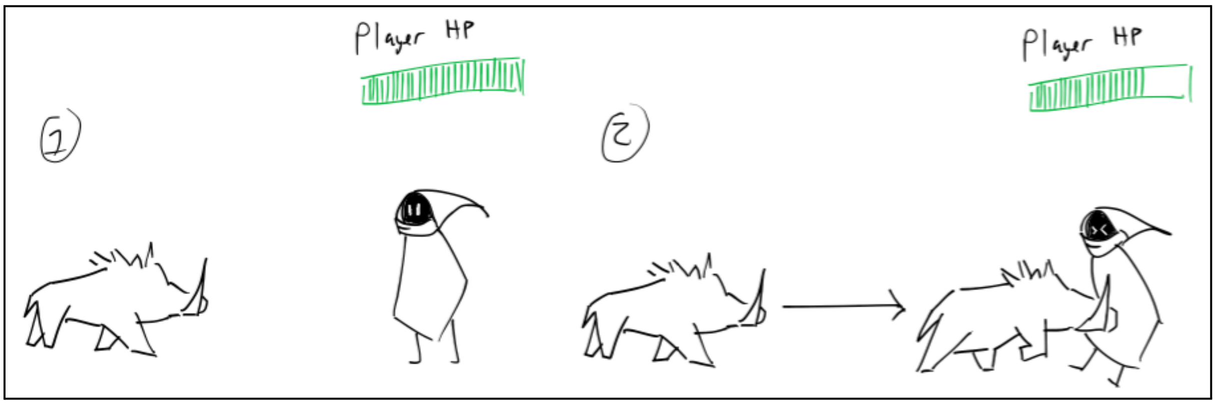
Pause Menu



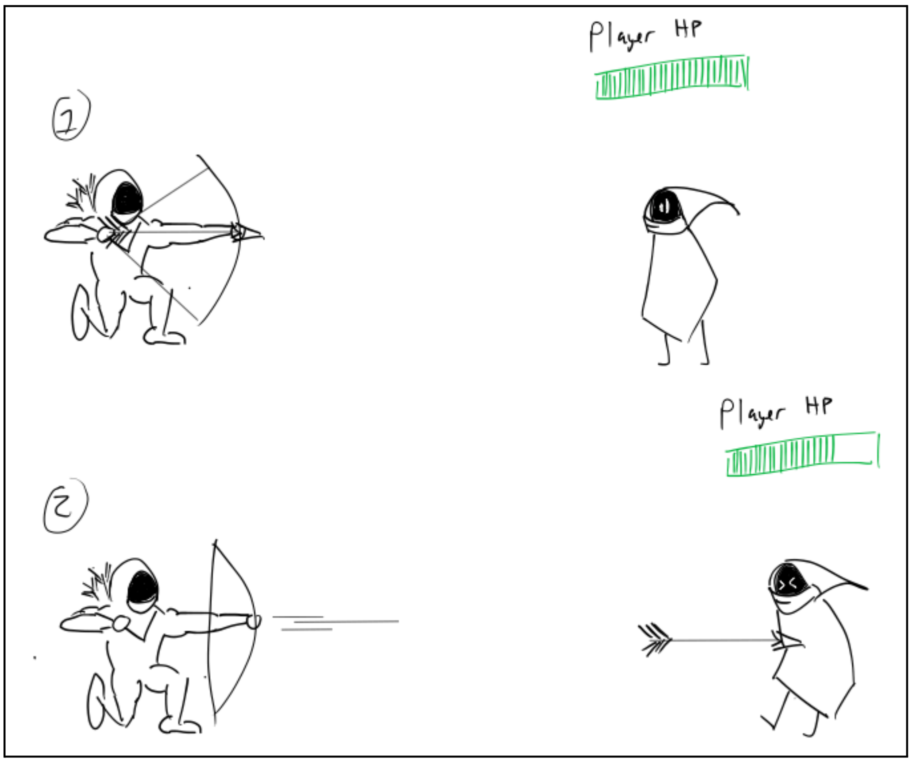
Gameplay Screen


health bar
1:06
Timer shows time survived
stamina bar
opens menu
ATTACK
PLAYER STAYS IN CENTER OF SCREEN
ENEMIES/ATTA
FOLLOW PLAYER AROUND OBSTACLES
traps in inventory
3

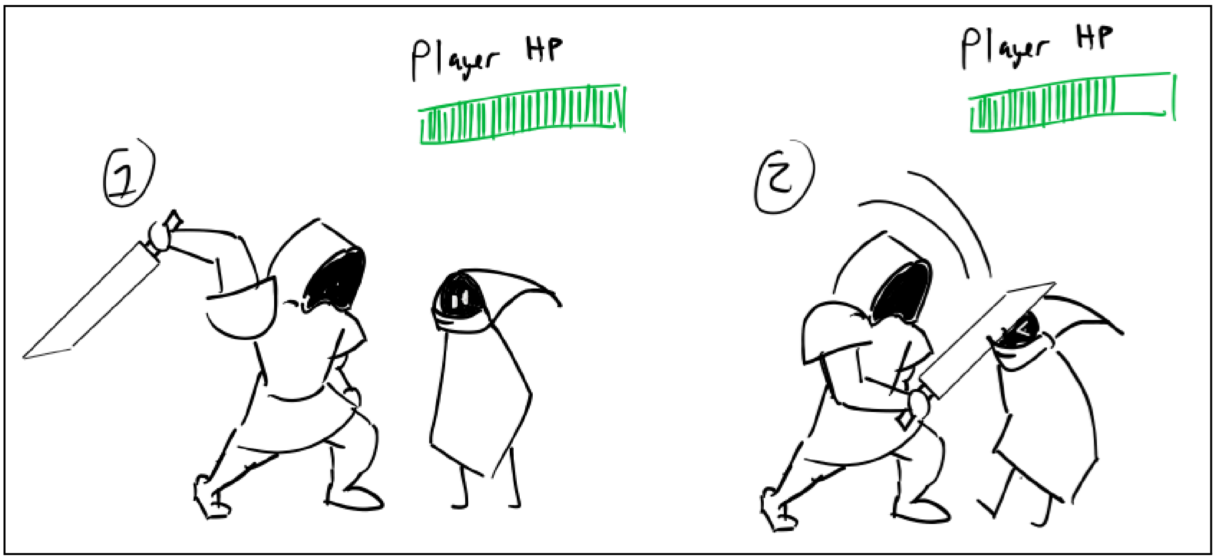
Enemy Attacking Pattern + **Player** taking damage



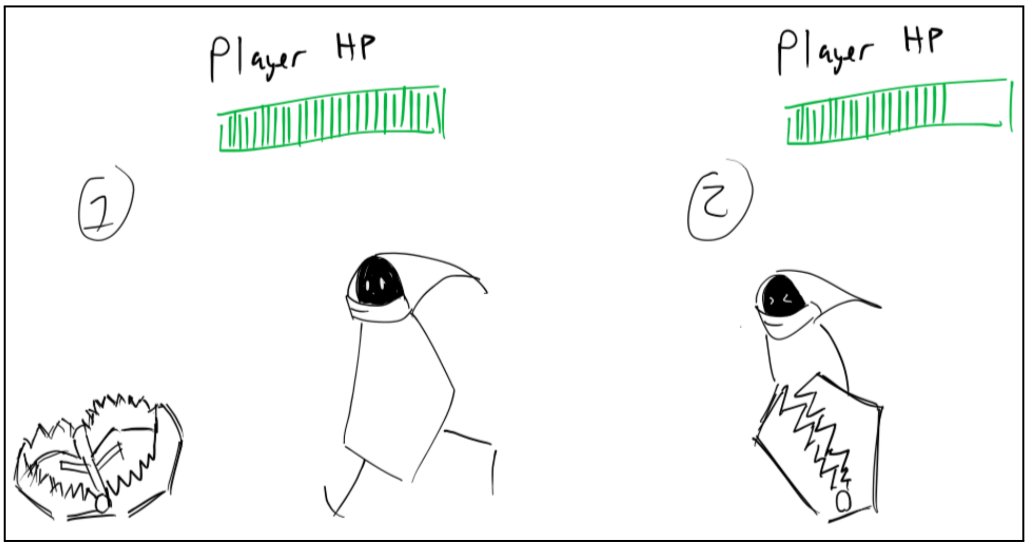
Player takes damage from a Boar's charge



Player takes damage from an Archer's arrow

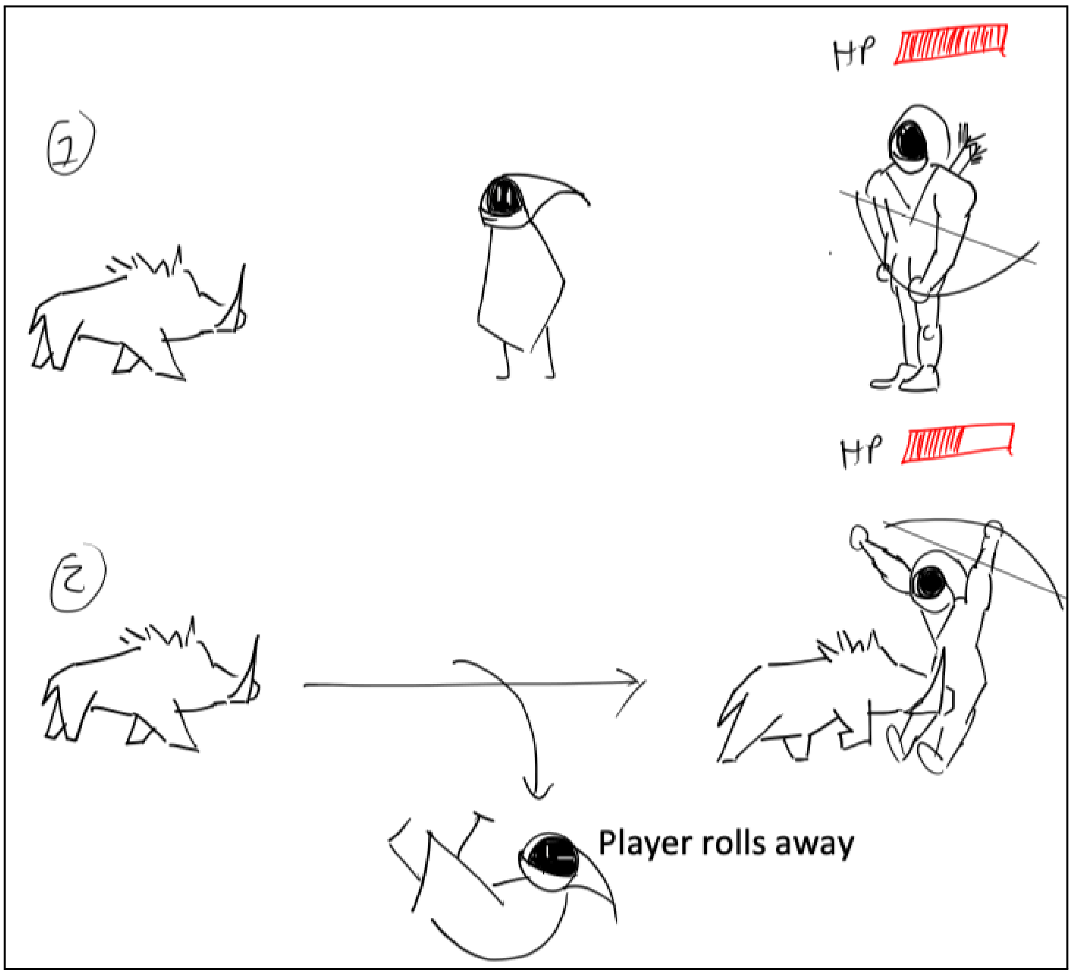


Player takes damage from Barbarian melee attack

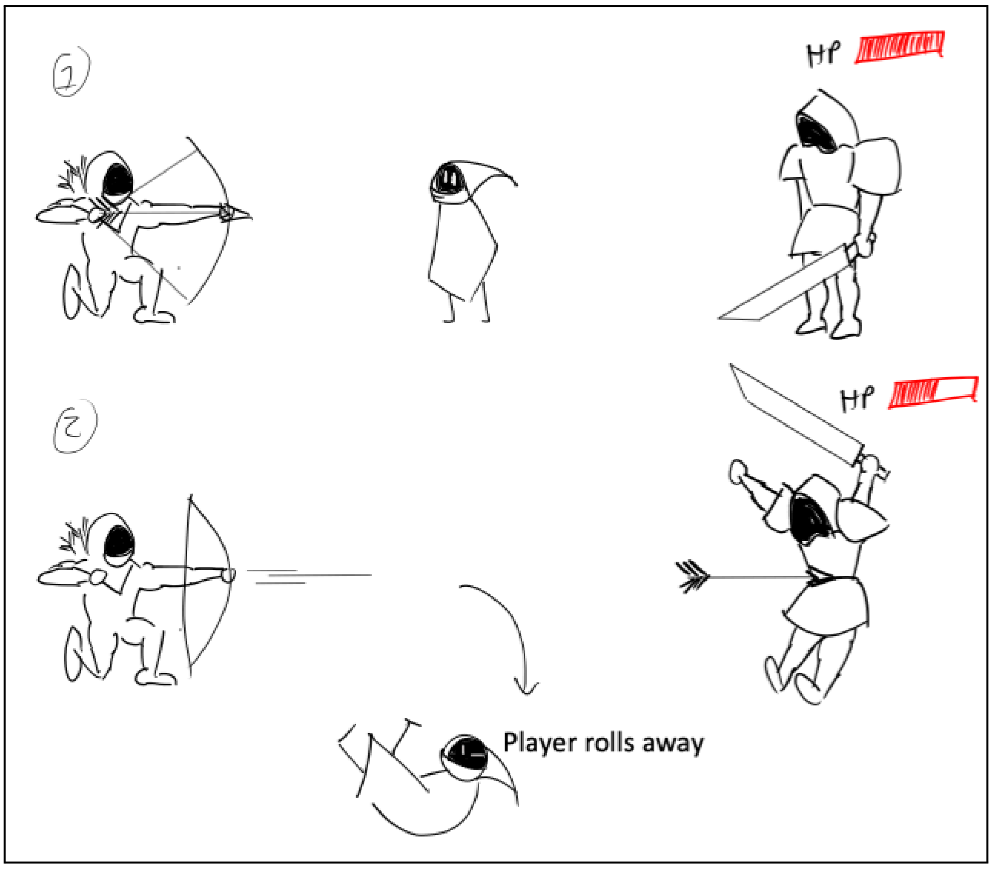


Player takes damage walking onto a trap

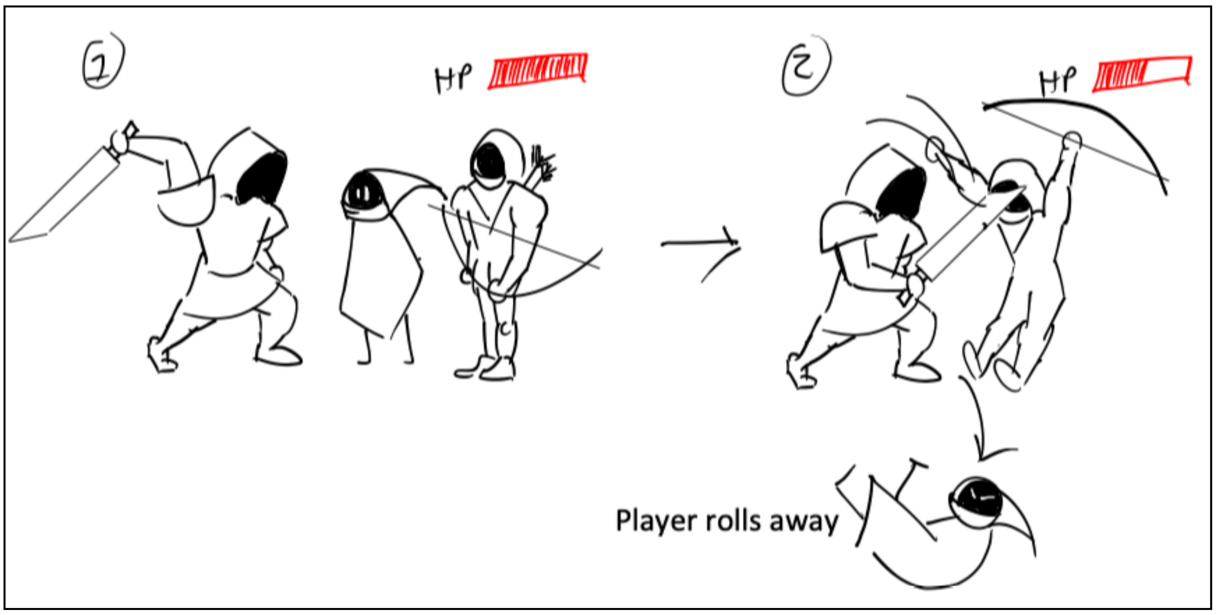
Enemies taking damage



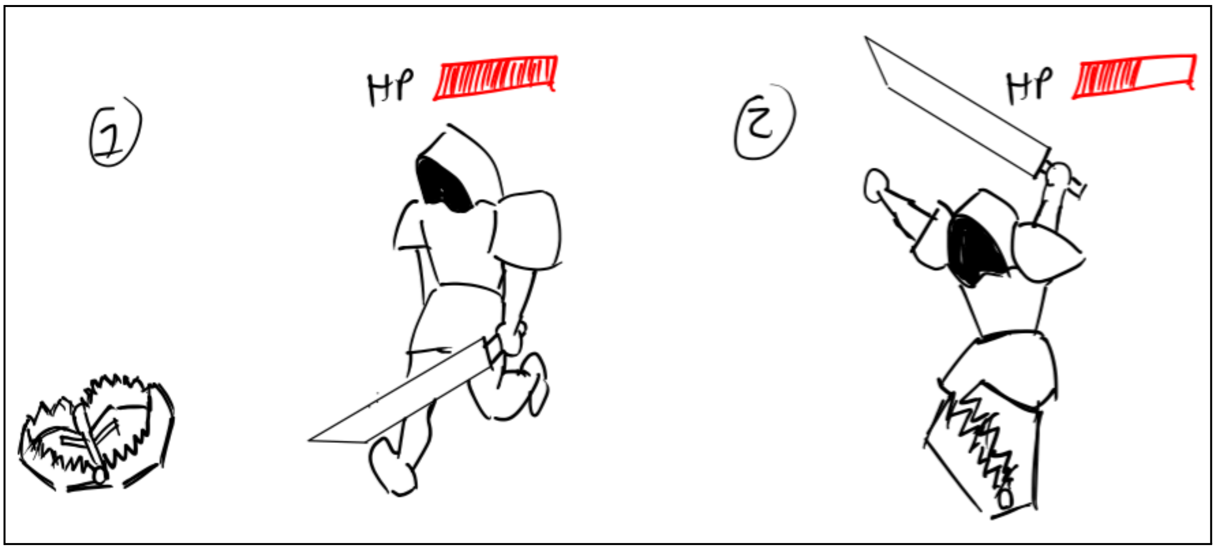
Enemy takes damage from a Boar's charge



Enemy taking damage from an Archer’s arrow



Enemy takes damage from Barbarian’s melee attack



Enemy takes damage from trap

Technical Elements

1) Rendering
    a) **Sprite Rendering** - Render Canvas/Background & 2D Characters (Player & Enemies)
    b) **Character Animations** - Display animations when a player or character takes an action
    c) **Layer Management** - A system to manage rendering order of game objects on the canvas, such as characters and UI elements to ensure gameplay elements are displayed appropriately.
    d) **Effects Rendering** - Render effects upon actions (eg. Character's color changes upon getting attacked/colliding)
    e) **Procedural Generated Map** - New map generated every game to make the game more interesting, and to eliminate repetitiveness.
2) Assets
    a) **Characters** - Player and Enemies with varying attributes.
    b) **Environment** - Canvas that matches the theme of medieval setting.
    c) **UI Elements** - Elements like health bars, and timer.
    d) **Obstacles** - Objects that impede the movement of player and attacks of enemies (eg. Rocks, Trees)
    e) **Collectibles** - Objects that give the player a strategy to take out the enemies (eg. Traps)
3) 2D Geometry Manipulation
    a) **Transformation & Translation** - Smooth movements (walk, sprint, jump, roll & dash), scaling and rotation of characters based on input.
    b) **Collision** - Collision detection algorithms to detect interactions between player, characters and environment
    c) **Hitboxes** - Hitboxes for interactive objects for accurate interactions such as enemy attacks.
4) Gameplay Logic

    a) **World Interaction** - Canvas is a bounded space. Movement is restricted within this area.
    b) **Player Controls** - Directional movements (WASD/Arrows) & extra keys for special movements (Jump, Sprint, Dash, Roll)
    c) **Health System**
        i) Player -> Calculate the damage suffered based on enemy attacks and reduce as necessary.
        ii) Enemy -> Calculate the damage suffered based on friendly fire, collisions with UI elements (obstacles)
    d) **State Management** - Manage game state like **start**, **paused** and **game over** & transition between these states.
    e) **Timer System** - **Start** timer as the game begins, **Pause** when the game is paused, and **Stop** when the game is over.
    f) **Gameplay Rules** - Logic to define rules & mechanics like how enemies spawn, different attacking patterns & health system.
    g) **Camera Controls** - Make the camera follow the player as it moves.

5) Artificial Intelligence
    a) **Path Finding** - Basic pathfinding algorithms to enable enemies to chase the player. Enemies gain better pathfinding ability as time progresses.
    b) **Behavior** - AI for different types of enemies with varying levels of difficulty and attack patterns, such as direct charges or projectile shooting.
    c) **Adaptive Difficulty** - Adjust game difficulty (enemy aggression/enemy type) based on time & player's performance while ensuring that gameplay is challenging but not impossible.
6) Physics
    a) **Movement Physics** - Realistic movements with attributes like acceleration, deceleration and inertia considered.
    b) **Projectile Physics** - Projectile trajectories for elements like arrows (Archer)
7) Sound
    a) **Sound Effects** - Sound effects for player interaction, enemy attacks and interactions with the environment (eg. hit the obstacle).
    b) **Dynamic Background Music** - Changes in background music based on game state, player health, and enemy aggression.
8) Software Engineering
    a) **Reloadability** - Able to save the game at a specific time point, with entities and their positions saved, so users can resume the game.

Advanced Technical Elements

Ordered by priority

1) **Swarm Behaviour** - Enemies display complex, coordinated behaviours to give a realistic, unpredictable tactic (Boids Algorithm)
   Impact from non-inclusion: Enemies' behaviour will be less engaging
   PlanB:
2) **Particle System** - Create multiple particle effects like blood splash and fire.
   Impact from non-inclusion: Will reduce the dynamism of the map and quality of atmosphere.

3) **2D Dynamic Shadow** - Shadowing based on light source that will vary based on characters' actions and positionings.
Impact from non-inclusion: Will reduce the quality of the game atmosphere.

Scope

Critical elements

- Basic player movement (no jump, sprint, or roll)
- Enemies spawn consistently
- Enemies do damage to each other
- Characters must collide with other characters
- Procedural generated map

Tasks to remove (if needed)

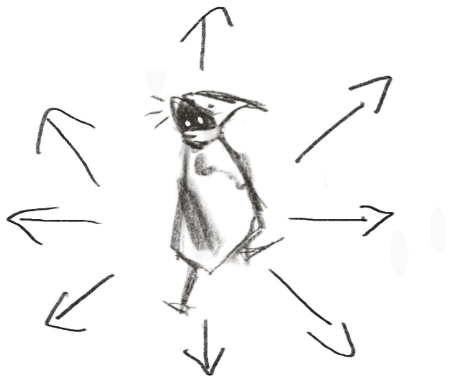
1. Reduce enemy types to 3 (Barbarian, archer, and boar)
2. Focus on having only one type of trap
3. Limited background music
4. Remove player jump, sprint, or roll
5. Get rid of swarm behavior
6. Remove obstacles

Devices

We plan to support standard keyboard input for controlling the player's movements:

Movement: Players will primarily use the keyboard for movement and actions. The default controls will be set to:

1. WASD for basic movement



2. WASD + Shift to Sprint. Uses Stamina

3. WASD + R to Roll. Attacks from enemies while rolling will do no damage. Uses Stamina

4. WASD + Space to Jump

5. WASD + F to Dash (i.e., suddenly move some distance) Attacks from enemies in the distance would do damage. Uses Stamina

Players may fully customize their keys for additional actions such as sprint, jump, roll and dash through the settings/preferences menu. This customization feature ensures that players can tailor the controls to their personal preferences for optimal gameplay experience.

Mouse (optional, if time permits): Menu navigation and UI interaction. This input design ensures that the controls are simple, accessible, and responsive for a fluid gameplay experience.

Tools

Sprites and Asset Management:
Aseprite(Might be paid but there is a free trial), Piskel, or CraftPix will be used for creating 2D sprites for characters and objects (warriors, player, obstacles). These tools offer pixel-art-friendly environments and animations suitable for our game's visual style.

**Optional: Tiled: A popular map editor that we could use for creating and managing the battleground arena layout, including walls and terrain.

Free pixel art asset pack: https://anokolisa.itch.io/dungeon-crawler-pixel-art-asset-pack

Audio:
Audacity for sound editing and creating custom sound effects such as footsteps, warrior collisions, and arena background music.

FreeSound.org for sourcing additional sound effects, such as metallic clashes and ambient sounds.

Team Management

We will utilize **Jira** to create epics, stories, and tasks that align with the project's milestones, ensuring that all necessary activities are completed on time.

Project Management & Task Tracking

Epics and Milestones

Each project milestone will correspond to an epic in Jira. Under each epic, we will break down the tasks into stories that detail the specific requirements and steps needed to achieve the milestone.

Sprints

Tasks will be organized into sprints based on their relevance to the upcoming milestones. This approach will help us stay focused and ensure that all tasks necessary for a milestone are prioritized and completed on time.

Task Assignment

Tasks will be assigned based on each team member's interests, experience, and the specific needs of the task. While **all team members will engage in programming tasks**, all of us will also have a specialization in certain areas of game development.

- **Carlo:** Gameplay Mechanics
- **Katie:** Arts
- **Linus:** Artificial Intelligence
- **Tarun:** Rendering
- **Yan Naing:** Team Management

The core of the tasks are based on rubrics for each milestone. Upon reviewing requirements and the scope of each requirement, tasks will be developed to suit the gameplay we intend to have.

Internal Policies and Deadlines

Sprints will generally be 1-2 weeks long, depending on the milestones and the type of tasks. We will establish a clear 'Definition of Done' for tasks, which may include code reviews and unit/functionality testing. We will also regularly hold 2 in-person meetings after class and 1 online meeting on the weekends to discuss progress, address challenges and plan next steps.

Documentation

Documentation of all phases of the project including technical requirements and meeting notes will be maintained in **Confluence**, to make sure that the resources are readily available for all the team members.

Development Plan

Milestone 1: Skeletal Game

Requirement	Task	Assignee	Due
Textured geometry	Render player entity sprites Render enemy entities sprites Render collectable sprites	Katie	Sep 30 Oct 2 Oct 2
Basic 2D transformations	Entities move according to current velocity	Tarun	Sep 30
Key-frame/state interpolation	Dash linearly interpolates player from current position to current position + direction * distance	Tarun	Oct 2
Keyboard/mouse control	Control the player character - WASD for movement - Shift for sprint - R for roll - F for dash	Tarun	Sep 30
Random/coded action	Spawning of enemies along edge of map Enemies set velocity directly towards player position	Linus	Oct 2
Well-defined game-space boundaries	Stop moving if the player arrives at the edge of map	Carlo	Oct 2
Simple collision detection & resolution (e.g. between square sprites)	Give player and enemy entities hitboxes and reduce player health on collision Player collects collectable when colliding with it Placed trap activates when player/enemy collides with it	Yan	Oct 2
Bug list	Add discovered bugs to list	All + supervised by Yan	
Test plan - a list of player or game actions and their expected outcomes	Add unit tests for every action Manual testing	All + supervised by Yan	
Other	Setup EntityManager	Linus	Sep 28

Camera Controls (basic)	Camera follows the player	Carlo	Oct 2
Simple rendering effects (basic)	Player turns red momentarily upon taking damage	Linus	Oct 2

Milestone 2: Minimal Playability

Requirement	Task
Game logic response to user input	Enemies attack the player if they are in range, otherwise they approach the player
Sprite sheet animation	Animate the player and enemy movements
New integrated assets	Add assets for background (border cliffs, clifftops, grassland) and sprites (different player/enemy stances)
Mesh-based collision detection	Provide a mesh for the player for determining collisions with weapons, enemies, obstacles, and the map border
Base user tutorial/help.	Add a help menu which describes the player controls and tips on eliminating enemies
Frames-per-second (FPS) counter	Add FPS counter in one of the window corners
2-minutes of non-repetitive gameplay	Expand the map and add more enemy types
Basic Physics (Basic)	Add archers that shoot arrows following arcs according to kinematics
Simple Pathfinding (Basic)	Enemies determine paths to take to get into attack range of the player, navigating their way around obstacles
Game stability	Play test and fix any glitches

Milestone 3: Playability

Requirement	Task
5 minutes of non-repetitive gameplay	Add more enemy types and collectables
Handle all user input	Pause game when game window loses focus (when minimized or tabbed out)
Swarm behaviour (Advanced)	Make melee enemy types engage in swarm behaviour
Reloadability (Basic)	Add the option to save the game to file and load it on the

	main menu Autosave when the game is quit Save high score continuously (in case of sudden shutdown)
Basic integrated assets (basic)	Add additional sprites for new enemy types
Game stability	Play test and fix any glitches
Other	Procedural map generation

Milestone 4: Final Game

Requirement	Task
10 minutes of non-repetitive gameplay	Add more enemy types and collectables
User Experience	Allow player to customize controls in settings
Game resolution and aspect ratio are consistent across machines	Play test on different machines and fix if not satisfied
Game stability	Play test and fix any glitches
Audio Feedback (Basic)	Add sound effects for player and enemy actions Add background music that escalates throughout
Particle systems (Advanced)	Add dirt particles for when the ground is hit hard Add smoke/fire particles
2D dynamic shadow (Advanced)	Make fires/lanterns cast dynamic shadows