Game Proposal: Eternal Gauntlet (Group 24)

CPSC 427 – Video Game Programming

Team: <West Coast Kevs>

Kevin Yi, 16389124

Roy Lee, 36934206

Skye Cheng, 94740792

Jaiveer Tiwana, 54932769

Jasper Huang, 78305794

Story:

Briefly describe the overall game structure with a possible background story or motivation. Focus on the gameplay elements of the game over the background story.

It's been 10 years since the world was lost to a monster invasion. What remains of humanity lives in fear in a small city at the edge of the world. Like many before you, you set out to defeat the demons and reclaim the world for mankind once again. But before you can face the leaders of the monsters you must first defeat their subordinates.

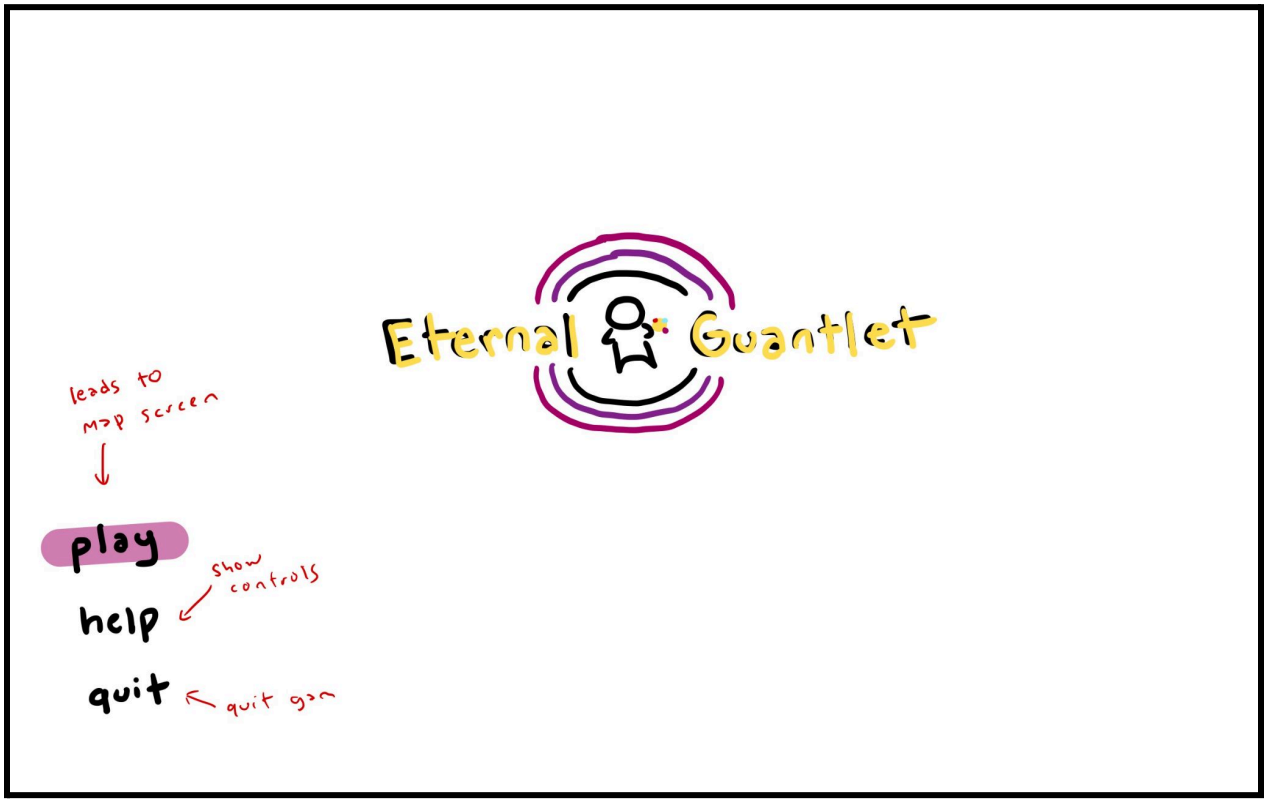
Gameplay Progression:

- Travel through the map by battling enemies/bosses
- Enemies drop 3 random minor upgrades (You choose one from 3 options)
- Minibosses drop 3 random powerful upgrades (You choose one from 3 options)
- Visit loot locations or shop locations on the map for upgrades
- Defeat the final boss to beat the game
- Losing to any enemy restarts the game (since this is a rogue like)

Scenes:

Produce basic, yet descriptive, sketches of the major game states (screens or scenes). These should be consistent with the game design elements, and help you assess the amount of work to be done. These should clearly show how players will interact with the game and what the outcomes of their interactions will be. For example, jumping onto platforms, shooting projectiles, enemy pathfinding or 'seeing' the player. This section is meant to demonstrate how the game will play and feeds into the technical and advanced technical element sections below. If taking inspiration from other games, you can include annotated screenshots that capture the game play elements you are planning to copy.

main menu:

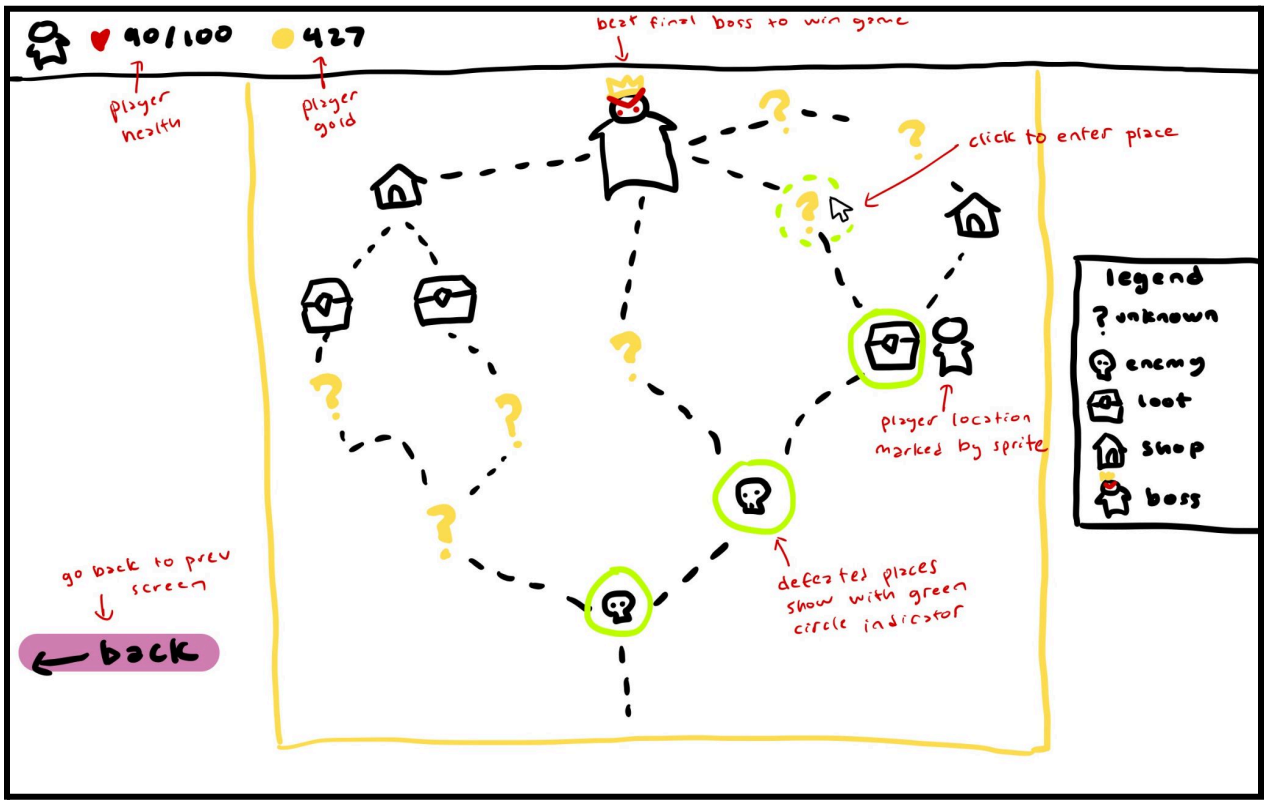


Clicking **play** leads to the **map screen**

Clicking **help** leads to the **controls screen**

Clicking **quit** will quit the game

map screen:



This is the screen that the player will use to navigate/progress through the game. The map is randomly generated at the start of the game. The player can only enter a level that is 1 step removed from a place they have already been to/defeated. Once the player defeats a level it is no longer marked as unknown on the map. Players can also not re-enter levels they have beaten already.

Clicking on a reachable **?unknown** will lead to a new fight with an enemy and the **combat ui** screen

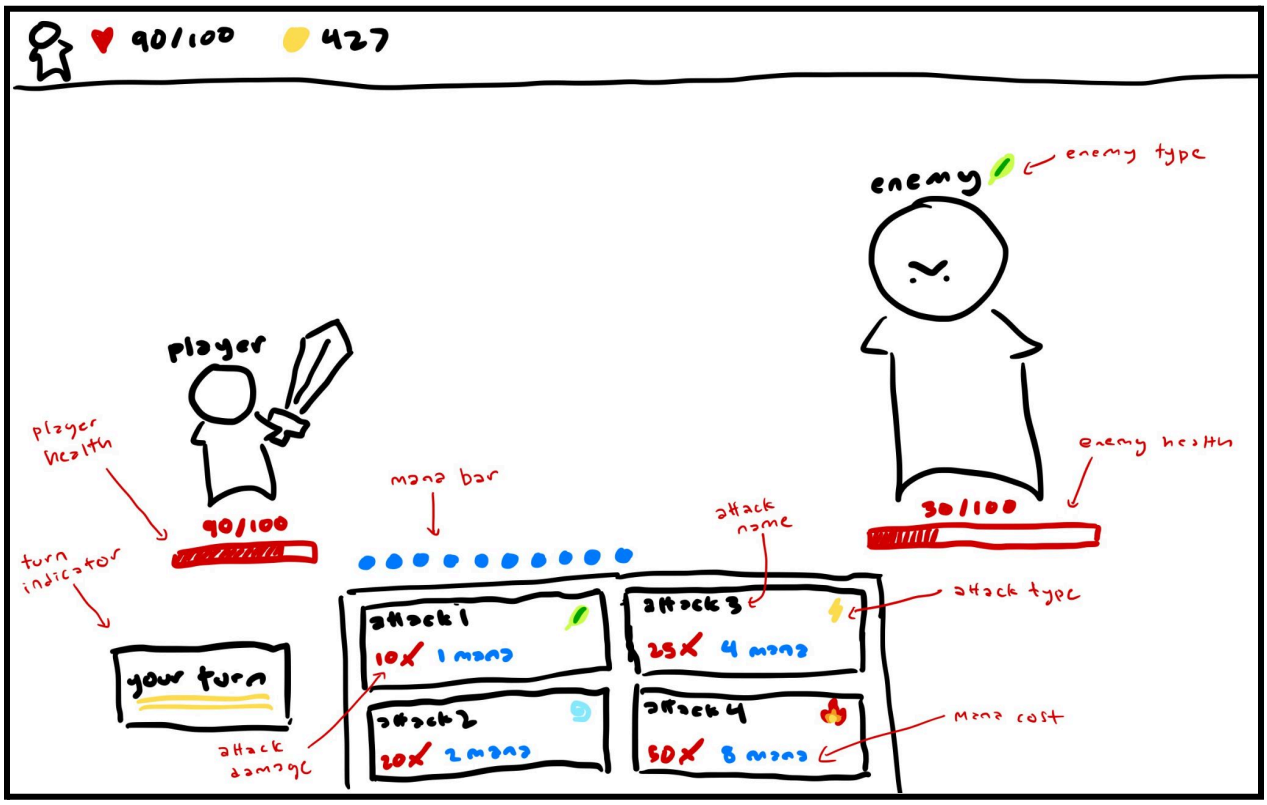
Clicking on a reachable **loot** icon will lead to the **loot screen**

Clicking on a reachable **shop** icon will to the **shop screen**

Clicking on a reachable **boss/final boss** will also lead to a new fight and the **combat ui** screen

Clicking on **back** will lead to whatever previous screen the player was on

combat ui:



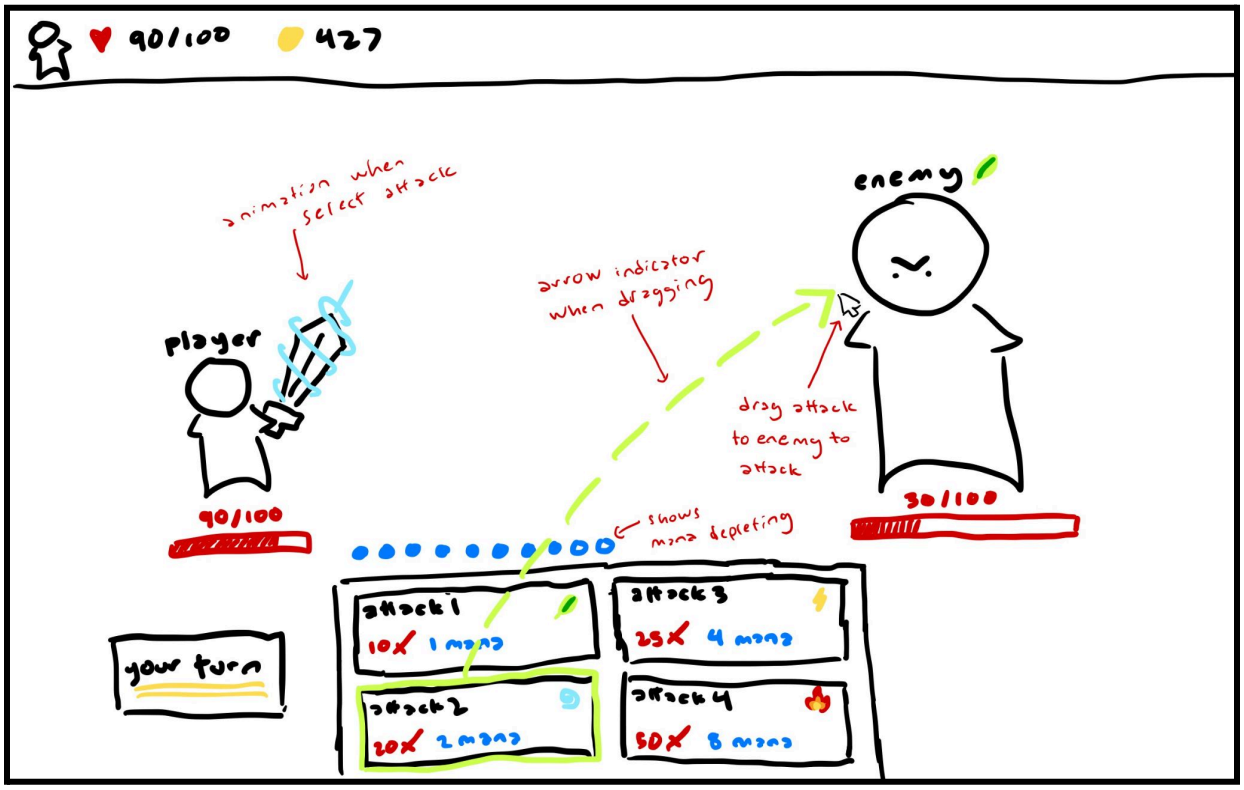
This is the combat ui for whenever a player is in combat with an enemy. When it is the player's turn, they can do as many attacks as long as they have mana for it.

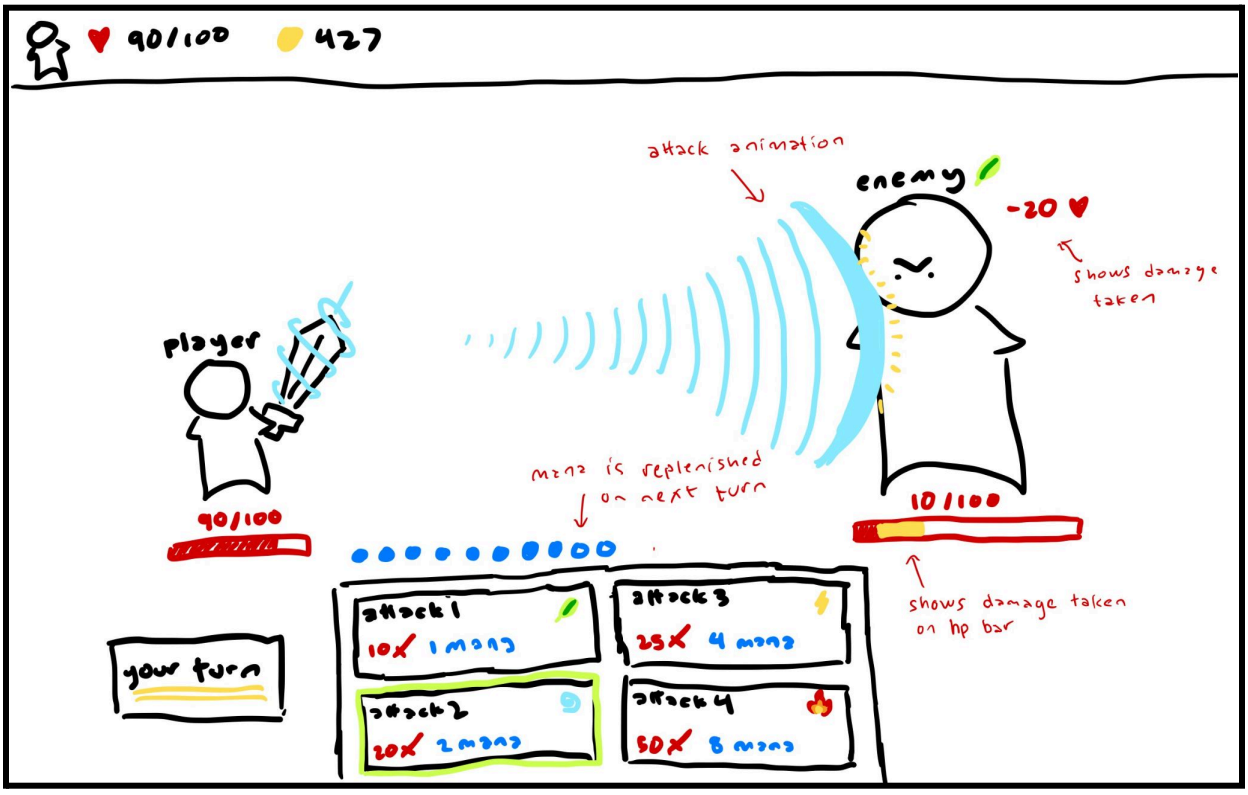
Click and drag an attack over on the enemy to use it (see scenes below for example)

Defeating an enemy will lead to the **enemy defeated** screen.

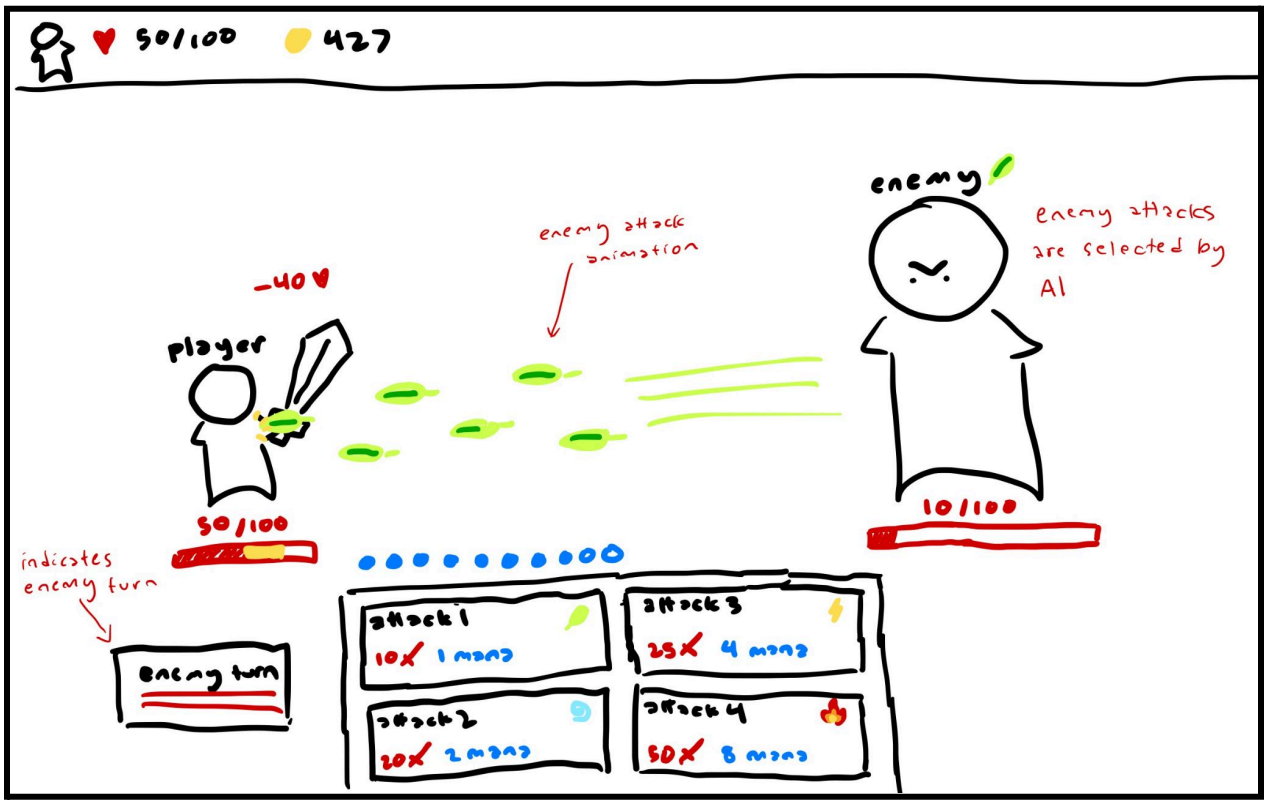
Losing to an enemy will lead to the **game over** screen (since this is a rogue-like).

combat ui - player turn:





combat ui - enemy turn:



enemy defeated screen:

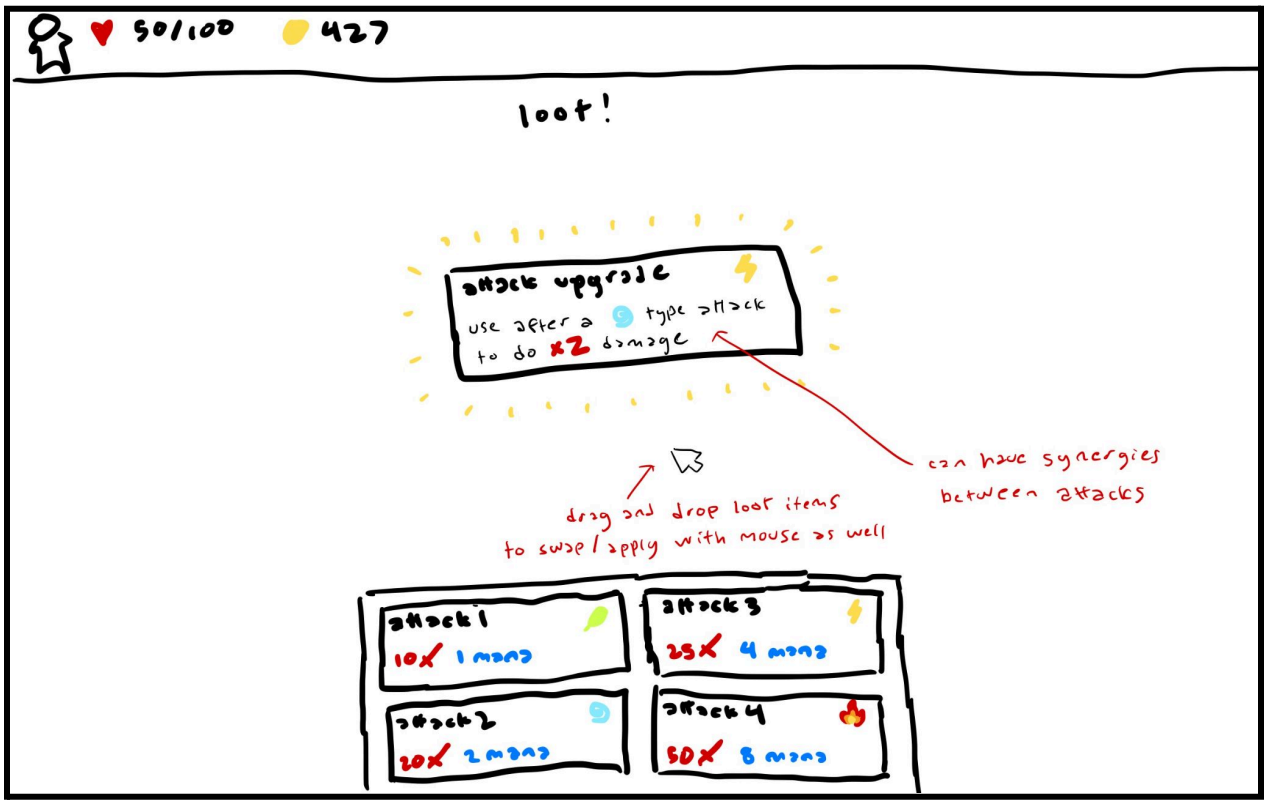
This is the screen for when a player defeats an enemy. The player will gain some hp back and some gold, as well as be rewarded with 3 randomly generated new attacks/attack upgrades, where they have to choose one.

If they choose a new attack, they can drag and drop to swap it with an existing attack.

If they choose an attack upgrade, they can drag and drop it on an existing attack to apply it.

After choosing one, this will lead back to the **map** screen.

loot screen:

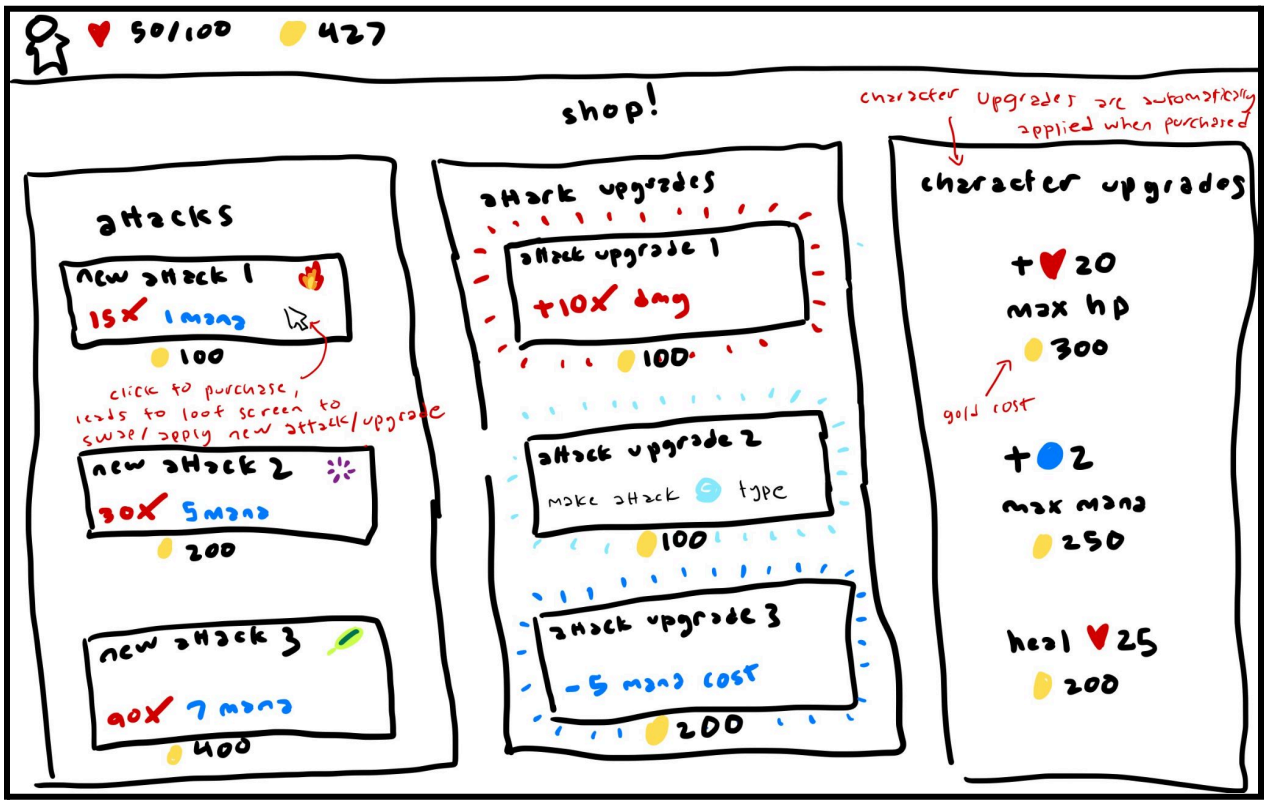


This is the loot screen which is for when a player enters a **loot** location on the **map**, or after they purchase an item(s) from the **shop**. Similar to after defeating an enemy, the player can drag and drop to swap an attack or apply an upgrade. After doing so, this will lead to either:

Back to the **map screen**, if the player entered a **loot** location.

Back to the **shop screen**, if the player purchased an item from the shop.

shop screen:



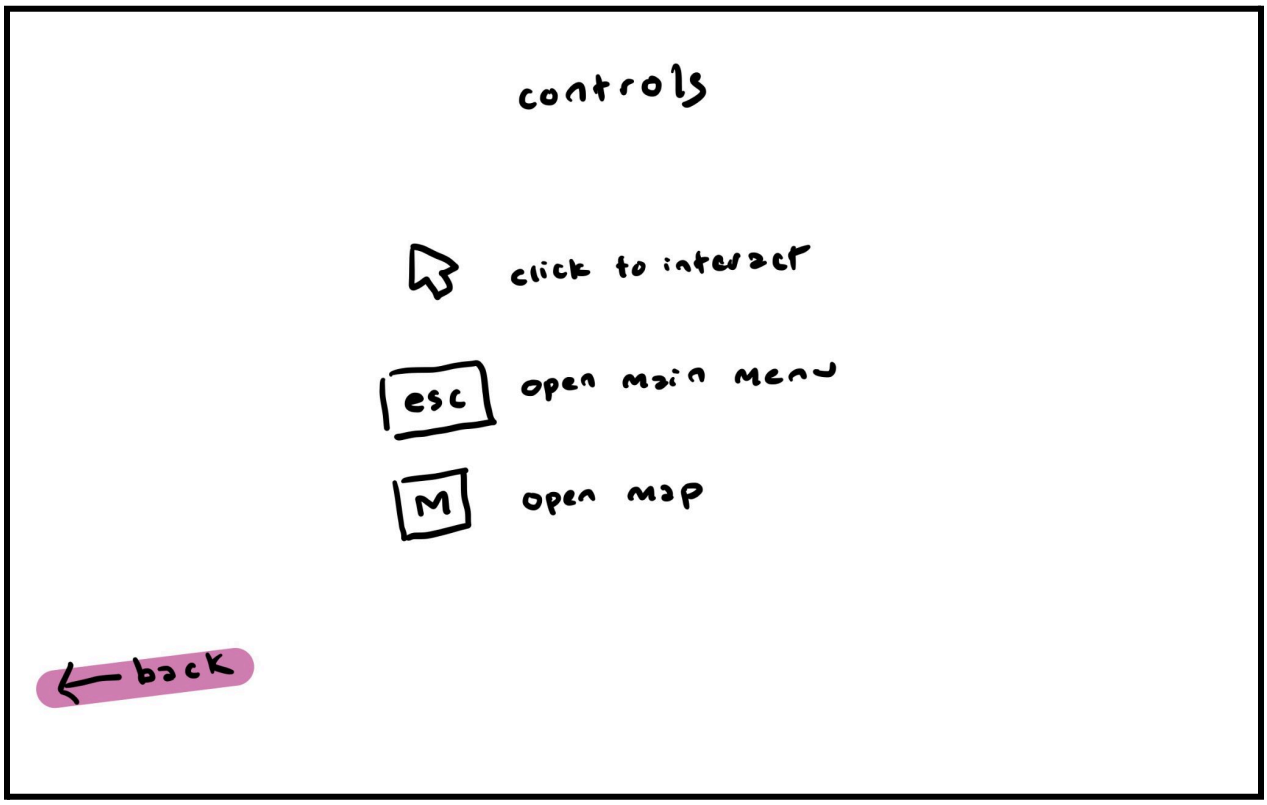
This is the shop screen where the player can spend the gold they have earned to purchase new attacks, attack upgrades, and character upgrades.

Click on any item to purchase it.

Purchasing an attack or attack upgrade will lead to the **loot screen**.

Purchasing a character upgrade will automatically apply it and stay in the **shop screen**.

controls screen:



This is the controls screen which will display the controls for the game. (perhaps it will be more in-depth than shown)

Clicking **back** will lead back to the main menu.

game over screen:

This is the game over screen for when a player loses to an enemy. This leads to the **main menu** screen where a new game will be started if the player chooses to continue to play.

Technical Elements:

Identify how the game satisfies the core technical requirements: rendering; geometric/sprite/other assets; 2D geometry manipulation (transformation, collisions, etc.); gameplay logic/AI, physics.

Rendering (OpenGL):

- For shaders we will use vertex and fragment shaders for sprites/game assets. Utilize OpenGL for effects like animations or special visual effects.
- possibly optimize performance by batching similar sprites together.
- upgrading abilities shows the transformation through visual changes

Geometric/sprite/other assets:

- Acquire some assets from https://kenney.nl/
- Add our own hand drawn assets

2D Geometry Manipulation:

- Use OpenGL's transformation matrices for handling 2D sprites in the world.
- Players can manipulate a sprite to traverse the map to decide which path to take

Gameplay Logic/AI:

- Use a state machine or turn manager that controls which player/enemy takes action can also use an events log to process turns
- Enemy AI, decision-making, some enemy moves will be dynamic based on the player's build/moves, while others will be static (predetermined move set).
- AI to randomly generate the map

Physics:

- attacks like projectiles using velocity and acceleration, particle effects, and explosions.

Advanced Technical Elements:

List the more advanced and additional technical elements you intend to include in the game prioritized on likelihood of inclusion. Describe the impact on the gameplay in the event of skipping each of the features and propose an alternative.

Advanced AI:

- add multiple enemies the player can fight at the same time, the enemies work together to combo their abilities. Skipping this would make the game "easier".

Gameplay:

- add a "quick time event" in which players can "dodge" or "amplify their attack". For example, during an enemies attack, the player might have a chance to dodge the attack. Skipping this would make the game simpler.

Devices:

Explain which input devices you plan on supporting and how they map to in-game controls.

Our game will support keyboard and mouse. There will be some keyboard shortcuts (ie. <esc> for opening the main menu, <m> for opening the map), but the game will mainly be played by using the mouse and clicking on UI elements.

Tools:

Specify and motivate the libraries and tools that you plan on using except for C/C++ and OpenGL.

We aren't sure of what other libraries and tools we plan to use to implement our game yet.

Team management:

Identify how you will assign and track tasks and describe the internal deadlines and policies you will use to meet the goals of each milestone.

We will do our best to break down tasks into stories and record them on a trello board. We will create a different trello board for each milestone and encapsulate the tasks into stories. Assigning tasks at the start of each week and then meeting weekly to track progress and see how overall progress is being made. For milestone 1:

Kevin Yi: implement damage, hp components

Roy Lee: ECS structure setup, basic UI

Skye Cheng: Render player and enemies

Jaiveer Tiwana: Implement combat, turn based system

Jasper Huang: Implement player, enemy and attack entity

Development Plan:

Provide a list of tasks that your team will work on for each of the weekly deadlines. Account for some testing time and potential delays, as well as describing alternative options (plan B). Include all the major features you plan on implementing (no code).

Milestone 1: Skeleton Game task breakdown

Tasks:

- ECS structure setup (registry)
- implement player entity
- implement enemy entity
- implement attack entity
- implement health component
- implement dmg component
- implement combat turn-based system
- rendering player
- rendering enemy

- rendering a single attack
- basic UI, show player's resources (health, mana/# of attack uses), show enemy health, show player abilities, attack button, exit game button
- input controls

General breakdown of milestones:

Milestone 1:

Week 1:

- ECS
    - Entities
        - Player
        - Enemy
        - Attack
    - Components
        - Health
        - Damage
- Gather visual elements for the UI

Week 2:

- Combat system logic
- Minimal rendering for combat (player, 1 enemy, attacks)
- Basic UI for combat
- Input controls

Milestone 2: Minimal Playability

Week 1:

- Have map set up
- Main menu
- Have tutorial
- Add "relics"

Week 2:

- AI for enemy attacks
- Implement block/parry mechanic
- FPS counter
- Enemies drop upgrades

Milestone 3:

Week 1:

- Multiple maps
- Shop setup
- Wire up audio

Week 2:

- Mini bosses and bosses
- More refined rendering
- Random encounters
- Add some enemies, item and attacks

Milestone 4:

Week 1:

- Have more complicated combat
    - Multiple enemies
- Add music and sound effects for enemies

Week 2:

- Add more non combat and combat encounters
- Add more relics/enemies and attacks
- AI generates enemies for the player to fight