**Ideation**

1. Exploration of Alternative Ideas

When I started using tools like arrays and classes, the first types of games that came to mind which seemed fun to create were pixel-style action-adventure games like Minecraft and Terraria, as well as real-time strategy city-building games like the Anno series. To narrow down the final concept, I categorized these game types by their perspectives—side-scrolling and top-down—and summarized their core content as combat, building, exploration, and character progression. I also reflected on other similar games I know, such as Crashlands, Starbound, Stardew Valley, and Clash of Clans.

However, given the limited time I have, it's clear I won't be able to implement all the features I envision. One significant reason I decided to drop the side-scrolling perspective is that it requires generating maps and simulating gravity and collisions. These mechanics do not directly contribute to the fun of the game but require additional features to support the gameplay experience, thus significantly increasing development complexity without adding immediate value. This reflects a design thinking approach, where I critically assess the implications of my choices.

Another factor supporting this decision is my experience playing Riftbreaker, a game that focuses on fast-paced shooting and combat, with base-building and tower defense elements seamlessly integrated. This combination enhances the gameplay experience, which I want to replicate in my design.

Therefore, I have settled on a top-down perspective for my final concept, focusing on large-scale on-screen enemy battles. This choice aligns with my goal of quickly providing a stimulating experience. If I have time remaining, I will also add base-building elements to enhance the experience.

1. Considerations of Interaction and Implementation

My game components will include a map, player, enemies, and combat. Without the building elements, I do not need to generate the map using arrays or randomization, I can directly use a large square map, which significantly reduces the workload related to map creation. I decided to use green ground as my map, but after testing, it appeared very monotonous, and players had difficulty perceiving their movement since the entire screen was filled with a solid color. Therefore, I added grass to enhance the visual variety. However, the grass still felt dull, so I implemented horizontal movement for the grass to simulate wind effects and added a slight shadow effect beneath it. Since the grass appears quite small on the map, it does not require high precision, but these changes greatly improve the visual experience.

I decided to place the player in the center of the screen and recognized keyboard input to add momentum to the player's PVector. This momentum is inversely applied to the map's coordinates, for instance, when the player moves to the right, the player's PVector has a positive value on the x-axis, but the map's coordinates need to subtract this positive value, allowing the map to move left, creating the illusion that the player is moving to the right. The same logic will apply to all objects, ensuring that when each object is stationary, it appears still relative to the ground.

Enemies will spawn randomly at the edges of the map and the number will gradually increase, permanently moving toward the player, which will provide a tense and thrilling experience for the player. This setup introduces several technical challenges: 1. Collision detection between enemies and the player—determining when an enemy stops relative to the player's position. 2. Collision detection between enemies, without this, enemies may overlap each other.

In terms of combat, the player can shoot bullets using the left mouse button. When a bullet hits an enemy, it inflicts damage, and both the bullet and the enemy are removed. Melee combat requires consideration of weapon trajectories and collision detection, which is more complex than bullets and does not significantly enhance the game's excitement, as it adds a more efficient attack option when enemies get close. If melee combat is to be implemented, many other elements will need to be balanced, so I will prioritize it lower in my work.