Challenge 1: Enemy PVector Detection

1. Introduction:

Enemies move toward the player until the collision box detects them and stops their movement. It is necessary to determine when the enemies move and stop, and when the collision box becomes active.

1. Code

When the enemy is to the right of the player, set the collision point as the rightmost point of the player and the leftmost point of the enemy.

if (monsterCurrentCenterXI > playerCollisionCenterXI)

monsterCollisionPlayerDetectionXI = monsterCurrentCenterXI - monsterWidthSetting/2;

playerCollisionDetectionXI = playerCollisionCenterXI + playerWidthSetting/2;

When the left point of the enemy is to the right of the right point of the player, and there are no other enemies blocking the way, the enemy will move to the left.

if (monsterCollisionPlayerDetectionXI > playerCollisionDetectionXI && !monsterBlockedX)

monsterDirectionXI = -1;

Each direction needs to be checked once.

1. Player Experience

It is crucial for enemies to move toward the player, and a well-functioning collision system is essential for the game. This allows for the logic loop of enemy spawning, seeking, and attacking the player. Only then do enemies possess basic playability.

Challenge 2: Enemy-to-Enemy Collision Detection

1. Introduction

If there are no other enemies blocking the way, the blocking boolean value for the enemy is set to false, allowing the PVector to be assigned. This challenge requires solving how to determine whether other enemies are blocking the path.

1. Code

need to loop through the entire monster array

for (Monster m : monsterArray) {

These two judgment logic can refer to the image "Monster Monster Collision Detection" for better clarity.

if (m.monsterCurrentCenterYI > monsterCurrentCenterYI - monsterHeightSetting && m.monsterCurrentCenterYI < monsterCurrentCenterYI + monsterHeightSetting)

if (monsterCurrentCenterXI > playerCollisionDetectionXI && m.monsterCurrentCenterXI >= monsterCurrentCenterXI - monsterWidthSetting && m.monsterCurrentCenterXI <= monsterCurrentCenterXI)

If the conditions for the detection are met, set the boolean value to true, allowing for the PVector assignment to be determined.

monsterBlockedX = true;

1. Player Experience

Enemy collision is very important. It is key to the smooth functioning of the combat system. Otherwise, if the enemies overlap, they won't occupy space properly, which significantly diminishes the visual effect.

Challenge 3: Shoot

1. Introduction

The weapon can rotate 360 degrees around the player. I need to obtain the values of mouseX and mouseY to calculate the position of the weapon's endpoint and bullet’s PVector.

1. Code

Just as shown in the image "Weapon End Point."

∵a^2 + b^2 = r^2

x^2 + y^2 = R^2

a/x = b/y = r/R

R = sqrt (x^2 + y^2)

∴a/x = r / sqrt (x^2 + y^2)

a = x \* r / sqrt (x^2 + y^2)

∴weaponEndPointXF = mouseXToPlayerI \* weaponLength / sqrt(mouseXToPlayerI\*mouseXToPlayerI + mouseYToPlayerI\*mouseYToPlayerI);

weaponEndPointYF = mouseYToPlayerI \* weaponLength / sqrt(mouseXToPlayerI\*mouseXToPlayerI + mouseYToPlayerI\*mouseYToPlayerI);

The calculation logic for bulletSpeedPVector is similar to the above, and the same image can be used to understand the logic.

R = bulletSpeed (which is int)

The values of a and b have already been calculated above. x and y are no longer the mouse position but rather the corresponding values of the PVector.

∵x / a = bulletSpeed / r

X = a \* bulletSpeed / r

∴bulletSpeedXF = weaponEndPointXF \* bulletSpeed /player.weaponLength;

bulletSpeedYF = weaponEndPointYF \* bulletSpeed / player.weaponLength;

1. Player Expeirience

By calculating the bullet's PVector in this way and combining it with the player's PVector, a realistic bullet effect can be achieved. Additionally, these parameters are adjustable, allowing for further possibilities without limitation.