MonkeyBusiness

Xing Tao Shi, Marcus Ng, Ishtiaque Mahdi

Period 4

ShootyTD

Concept:

Our concept is a top down shooter tower defense. Our idea is based off the games *Sanctum* and *Orcs Must Die!*. The main point of the game is to defend a nexus from waves of enemies. Enemies will spawn at specific locations and make their way towards the nexus or the player depending on their target. The player will use weapons, towers, and traps to survive and defend the nexus. He or she will also be able to buy weapons, towers, and traps from a shop with money dropped by the enemies.

The user will move with WASD and shoot with the mouse. At the start of each wave, enemies will be stored in a queue and then dequeued when they spawn on the battlefield. We will use an ArrayList to store spawned enemies to detect collisions with bullet objects. Traps will be able to be placed on top of one another, and they will be stored in a stack. The last trap placed will be triggered first. The player's inventory will be stored in an ArrayList allowing he or she to select a weapon with numbers. We want to use a CSV file to save high scores.

Term 2 Concepts:

- *Queue* - We will use a queue to store enemies. Each queue will represent a wave. The enemies will be dequeued from the queue with a reasonable time interval so that we can control the flow of the game.
- *Priority Queue* - We will use a priority queue to keep track of enemies that are in range of each turret. Once enemies walk in range of a turret, they will be added to that turret's enemy priority queue. The turret will target the closest enemies first. When an enemy dies, they will be dequeued.
- *ArrayList* - The queuedEnemies ArrayList will contain a queue of enemies for every wave. An ArrayList will also stores spawned enemies to allow us to detect collision with bullet objects. We will also use an ArrayList to toggle between the character's inventory, and store turrets, traps, and gold in order to display them in the game.
- *Stack* - Stacks will be used to set up multiple traps at the same position in the game. Once an enemy steps on a trap, that trap is popped out of the stack and the next trap in the stack replaces the old trap at the same position.

UML Diagrams

GAME MANAGER	PLAYER
- Player player - Nexus nexus - Shop shop - ArrayList<Queue<Enemy>> queuedEnemies - ArrayList<Enemy> spawnedEnemies - ArrayList<Bullet> bullets - ArrayList<Turret> turrets - ArrayList<Trap> traps - ArrayList<Gold> gold - int waveInterval - bool gameOver - int highscore - void setup() - void draw() - void keyPressed() - void keyReleased() - void saveHighscore() - void gameOver() - void restartGame()	- PVector dir - float speed - int startingHP - int currentHP - int money - ArrayList<Weapon> weapons - bool isDead - color c Constructors, setters, and getters… + void move() + void shoot() + void takeDamage(int) + void dead() + void healthBar() + void display()

ENEMY	NEXUS
- PVector dir - PVector target - float speed - int startingHP - int currentHP - int damage - int goldAmount - bool isDead - color c Constructor, setters, and getters… + void move() + void attack(Object) + void takeDamage(int)	- int startingHP - int currentHP - boolean isDead - color c Constructors, setters, and getters… + void takeDamage(int) + void dead() + void healthBar() + void display()

+ void dropGold() + void dead() + void healthBar() + void display()	

SHOP	WEAPON
- ArrayList<Weapon> weapons - ArrayList<Turret> turrets - ArrayList<Trap> traps Constructors, setters, and getters… + void buy(int) + bool afford(int) + void display()	- int damage - int price - double range Constructors, setters, and getters… + void shoot() + void display()

GOLD	BULLET
- int amount - color c Constructors, setters, and getters… + void pickupGold() + void display()	- PVector loc - color c Costructors + void move() + void display() + bool inRange(double) + bool collidesWithEnemy(Enemy)

TURRET	TRAP
- int startingHP - int currentHP - int damage - int price - double firerate - double range - ArrayList<Enemy> inRange	- int damage - int price Constructors, setters, and getters… + void dealDamage(int) + bool collidesWithEnemy(Object) + void display()

Constructors, setters, and getters… + void takeDamage(int) + bool enemyInRange() + void shoot() + void display()	