PakBokWai_JaiBokThorn

Created by

Pongpitchaya Jaidee 6632141921

Theerach Sae-Ngow 6630154721

2110215 Programming Methodology

Semester 1 Year 2024

Chulalongkorn University

Rouge Evil

Introduction

Rogue Evil gameplay is inspired by the famous game name “soul knight” and its setting from “Resident Evil”. The objective of the game is to clear all waves of the enemy and beat the last boss as fast as possible.

Rules

The player will be surrounded by a flock of zombies. When a zombie hits the player, the player's HP will decrease. If the player's HP reaches 0, the game will end. The player must shoot waves of zombies to survive until the end of the game. Additionally, after each wave ends, the player can choose a buff to become stronger. Finally, if the player either clears the game or experiences a game over, they can choose to start a new game+ mode, beginning the game with a stronger player.

Example

- When entering the program, the player can choose “Start” to begin the game, “Help” to get more detail about the game and “Exit” to close the program.

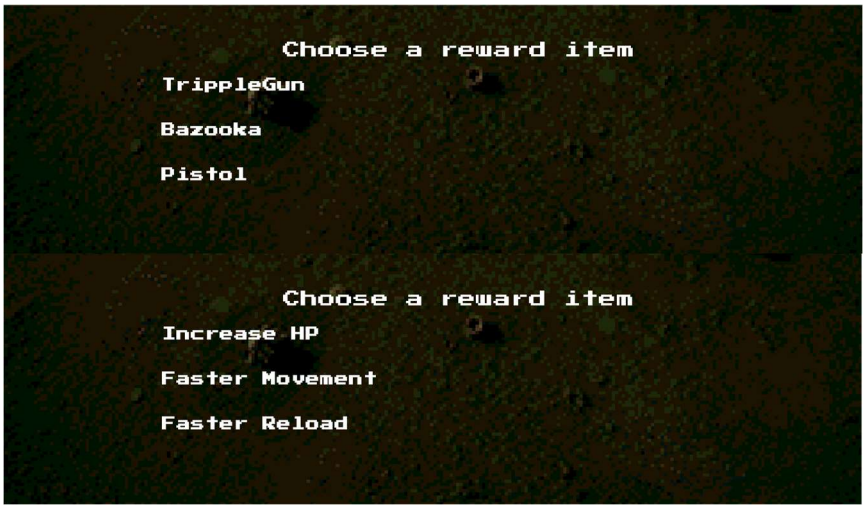
- When the game begins it will show a game as follows.

- A status report will be displayed in the top left corner of the screen. It will show the player's current HP, the current wave, the time played in the current session.

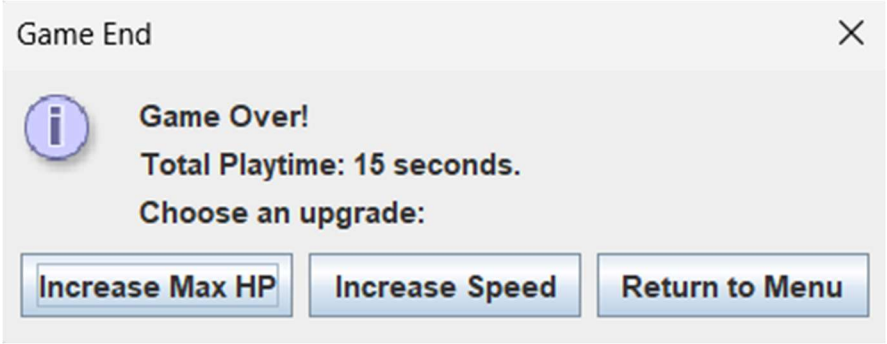
- Above the player, a text will be displayed showing either “READY” or a number. “READY” indicates that the player can fire their gun at any time, while the number represents the cooldown time in seconds before the player can fire their gun again.

- Above each zombie, its HP is displayed. Additionally, different types of zombies have unique HP values, size and features. For example, a Shooter zombie can attack from a distance by shooting at the player, while a Tank zombie is significantly tougher and has more HP compared to a normal zombie.

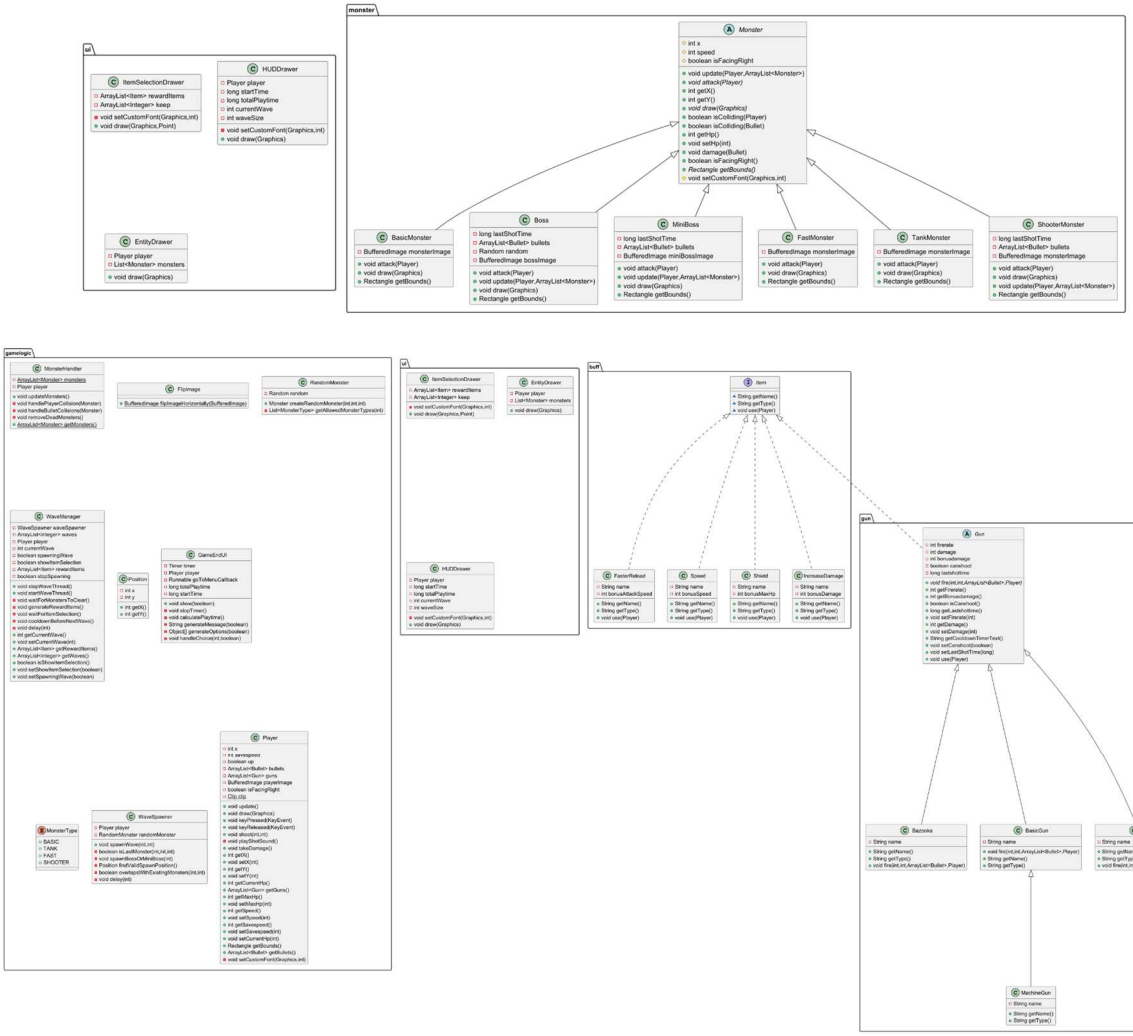
- At the end of each wave, the player chooses one of three buffs. After the first round, new guns unlock with unique play styles, like the fast-reloading "Machine Gun" or the high-damage "Bazooka." Buffs may improve reload speed, fire damage, or other stats.



- In later waves, special monsters will appear, and the number of enemies will increase, making the game progressively harder to complete. New game+ becomes essential for success, as it aligns with the core mechanics of a roguelike game.
- When the game ends, the player can choose to start a new game+ with buffs and begin a new wave or return to the main menu.



Class diagram



1)Package:buff

1.1)Interface Item

1.1.1)Methods

+String getName()	Returns the name of the item.
+String getType()	Returns the type of the item.
+ void use(Player player)	Applies the item's effect to the given Player.

1.2)Class:FasterReload implements Item

1.2.1)Fields

-String name	The name of the item.
-int bonusAttackSpeed	The bonus attack speed to be applied to a weapon's fire rate.

1.2.2)Constructor

+FasterReload(String name, int bonusAttackSpeed)	Initializes the item with a specific name and bonus attack speed value.

1.2.3)Methods

+String getName()	Returns the name of the item.
+String getType()	Returns the type of the item. Always returns "FasterReload".
+void use(Player player)	Applies the item's effects to the specified player: -Retrieves the player's first equipped gun. -If the gun is a MachineGun, set the

	bonusAttackSpeed to 20. -Decreases the gun's fire rate by the bonusAttackSpeed. -Restores the player's health to the maximum.

1.3)Class:IncreaseDamage implements Item

1.3.1)Fields

-String name	The name of the item.
-int bonusDamage	Represents the additional damage bonus granted by this item.

1.3.2)Constructor

+IncreaseDamage(String name, int bonusDamage)	Initializes the item's name and bonus damage value.

1.3.3)Methods

+String getName()	Returns the name of the item.
+String getType()	Returns the type of this item. Always returns “IncreaseDamage”.
+void use(Player player)	Applies the item's effects to the specified player: -Retrieves the first equipped gun from the player's inventory. -Checks the gun's type: 1. If it's a MachineGun or ShotGun, set bonusDamage to 2. 2. If it's a BasicGun, set bonusDamage to 10. -Decreases the gun's fire rate by the bonusAttackSpeed. -Restores the player's health to the

	maximum.

1.4)Class:Shield Implements Item

1.4.1)Fields

-String name	The name of the item.
-int bonusMaxHp	The additional maximum HP provided by the shield.

1.4.2)Constructor

+Shield(String name, int bonusMaxHp)	Initializes the shield with a specific name and maximum HP bonus.

1.4.3)Methods

+String getName()	Returns the name of the item.
+String getType()	Returns the type of the item. Always returns "Armor - Shield".
+void use(Player player)	Applies the shield's effect to the specified player by: 1. Increasing the player's maximum HP by bonusMaxHp. 2. Restoring the player's current HP to the new maximum.

1.5)Class:Speed Implements Item

1.5.1)Fields

-String name	The name of the item.
-int bonusSpeed	The speed bonus provided by the item.

1.5.2)Constructor

+Speed(String name, int bonusSpeed)	Initializes the speed boost item with a specific name and speed bonus.

1.5.3)Methods

+String getName()	Returns the name of the item.
+String getType()	Returns the type of the item. Always returns "Boost - Speed".
+void use(Player player)	Applies the speed boost to the player by: 1. Increasing the player's speed by bonusSpeed. 2. Storing the new speed value in the player's saved speed.

2)Package:bullet

2.1)Class:Bullet

2.1.1)Fields

-int x	The name of the item.
-int y	The speed bonus provided by the item.
-int speed	The speed at which the bullet moves, defaulted to 15.
-int directionX	The x-directional component of the bullet's movement.
-int directionY	The y-directional component of the bullet's movement.
-int damage	The damage value of the bullet.
-Image bulletImage	The image used to represent the bullet.

-final Int width	The width of the bullet, set to 10.
-final Int height	The height of the bullet, set to 10.

2.1.2)Constructor

+Bullet(int x, int y, int mouseX, int mouseY, int damage)	-Initializes a bullet with a given starting position (x, y), target position (mouseX, mouseY), and damage value. -Calculates the direction (directionX, directionY) based on the target's position and normalizes the movement speed.

2.1.3)Methods

+void update()	Updates the position of the bullet based on its direction and speed.
+void draw(Graphics g, Color c)	Draws the bullet on the screen using the provided Graphics object and color.
+boolean isOutOfBounds(int width, int height)	Returns true if the bullet is out of the screen bounds.
+int getX()	Returns the x-coordinate of the bullet.
+ int getY()	Returns the y-coordinate of the bullet.
+Rectangle getBounds()	Returns a Rectangle representing the bullet's bounds for collision detection.
+int getDamage()	Returns the bullet's damage value.

2.2)Class:MissileBullet extends Bullet

2.2.1)Fields

-explosionRadius	The radius of the missile's explosion, defaulted to 50.

2.2.2)Constructor

+MissileBullet(int x, int y, int mouseX, int mouseY, int damage)	Inherits from the Bullet class and initializes the missile bullet with the specified position (x, y), target position (mouseX, mouseY), and damage value.

2.2.3)Methods

+void causeAreaDamage(ArrayList<Monster> monsters)	-Checks if any Monster within the provided list is within the missile's explosion radius. If so, it applies damage to that monster. -It calculates the distance between the missile and each monster, and if the distance is less than or equal to explosionRadius, it calls the damage() method of the Monster class to apply damage.
+void draw(Graphics g, Color c)	Draws the missile and its explosion radius on the screen using the provided Graphics object and color. It calls the parent class's draw() method to render the missile and then draws a semi-transparent red circle representing the explosion radius.

3)Package:gamelogic

3.1)Class:FlipImage

3.1.1)Methods

+static BufferedImage flipImageHorizontally (BufferedImage image)	Creates a horizontally flipped version of the input image and returns it.

3.2)Class:GameEndUI

3.2.1)Fields

-final Timer timer	The timer used in the game.
-final Player player	The player instance.
-final Runnable totalPlaytime	Callback to return to the menu.
-long startTime	The start time of the game.
-long totalPlaytime	Total playtime of the game.

3.2.2)Constructor

+GameEndUI(Timer timer, Player player, long totalPlaytime, long startTime, Runnable goToMenuCallback)	Initializes the GameEndUI with required dependencies.

3.2.3)Methods

+void show(boolean won)	Displays the end game UI with options based on win or loss status**.**
-void stopTimer()	Stops the timer if it's running.
-void calculatePlaytime()	Calculates the total playtime by adding the time elapsed since the game started.
-String generateMessage(boolean won)	Generates a message based on the game's outcome.
-Object[] generateOptions(boolean won)	Generates options for the player based on whether they won or lost.
-void handleChoice(int choice, boolean won)	Handles the player's choice after the game ends: 1. If the player wins, it triggers the menu callback. 2. If the player loses, it processes choices to upgrade the player or

	return to the menu.

3.3)Class: MonsterHandler

3.3.1)Fields

-static ArrayList<Monster> monsters	A list that holds all the monsters in the game.
-Player player	The player object interacts with the monsters.

3.3.2)Constructor

+MonsterHandler(ArrayList<Monster> monsters, Player player): Initializes the MonsterHandler	Initializes the MonsterHandler with a list of monsters and the player.

3.3.3)Method

+void updateMonsters()	Loops through all monsters, updates them, checks for collisions with the player, and handles bullet collisions.
-void handlePlayerCollision(Monster monster)	Handles what happens when a monster collides with the player. 1. If the monster is a Boss or MiniBoss, the player takes damage. 2. Otherwise, the player takes damage and the monster is removed from the list.
-void removeDeadMonsters()	Iterates through the list of monsters and removes any that have zero or negative health.
+static ArrayList<Monster> getMonsters()	Returns the list of all monsters.

3.4)Enum:MonsterType

Value

BASIC	Represents a basic type of monster (e.g., BasicMonster).
TANK	Represents a tank-like monster (e.g., TankMonster).
FAST	Represents a fast-moving monster (e.g., FastMonster).
SHOOTER	Represents a monster that shoots (e.g., ShooterMonster).

3.5)Class:Player

3.5.1)Fields

-int x	The x-coordinate of the player.
-int y	The y-coordinate of the player.
-int currentHp	The player's current health points.
-int speed	The player's movement speed.
-int maxHp	The player's maximum health points.
-int savespeed	The player's saved speed for future use.
-boolean up, down, left, right	Directional movement flags.
-boolean isdmg	Indicates if the player is taking damage.
-ArrayList<Bullet> bullets	List of bullets fired by the player.
-ArrayList<Gun> guns	List of guns the player possesses.
-BufferedImage playerImage	The player's image.
-boolean isFacingRight	Indicates the player's facing direction.
-static Clip clip	Used to play sound effects (e.g., shooting

	sounds).

3.5.2)Constructor

+Player(int x, int y)	Initializes the player's position, health, speed, bullets, and guns. Also loads the player's image.

3.5.3)Methods

+void update()	Updates the player's position based on movement keys and updates bullets.
+void draw(Graphics g)	Draws the player and the bullets on the screen.
+void keyPressed(KeyEvent e)	Sets the corresponding directional flags when a key is
+void keyReleased(KeyEvent e)	Resets the corresponding directional flags when a key is
+void shoot(int mouseX, int mouseY)	Fires the player's current gun towards the mouse position, if the gun is ready to shoot.
-void playShotSound()	Plays the shooting sound based on the current weapon. It identifies the weapon type and plays the corresponding sound effect.
+void takeDamage()	1. Decreases Health: The currentHp is decremented by 1 when the player takes damage. 2. Speed Boost: If the player is not already in a damage state (if dmg is false), their speed is temporarily doubled. This gives the player a temporary burst of speed when they take damage 3. Damage State Flag: The is dmg flag is

	set to true, indicating that the player is currently in a damage state and preventing further speed boosts 4. Delayed Speed Reset: A new thread is started that: a. Waits for 3 seconds (Thread.sleep(3000)). b. After 3 seconds, the player's speed is reset back to the original value stored in savespeed. c. The is dmg flag is set to false, allowing the player to take damage again and enabling the speed boost effect if needed.
-void setCustomFont(Graphics g, int fontSize)	This method sets a custom font for rendering text on a Graphics object. It attempts to load a specific font ("PressStart2P.ttf") from the resources. If loading the custom font fails, it falls back to the default "SansSerif" font.
Generate getter of all fields.	

3.6)Class:Position

3.6.1)Fields

-final int x	The x-coordinate of the position.
-final int y	The y-coordinate of the position.

3.6.2)Constructor

+Position(int x, int y)	Initializes a new position with specified x and y coordinates.

3.6.3)Methods

Generate getter of all fields.	

3.7)Class:RandomMonster

3.7.1)Fields

-final Random random	A Random object used to generate random numbers for monster selection.

3.7.2)Constructor

+RandomMonster()	Initializes the RandomMonster object, creating a new instance of the Random class.

3.7.3)Methods

+Monster createRandomMonster(int x, int y, int currentwave)	Creates a random monster based on the current wave. 1. Retrieves the list of allowed monster types based on the current wave using the getAllowedMonsterTypes() method. 2. Selects a random monster type from the list and creates the corresponding monster. 3. Returns a new instance of the selected monster type (BasicMonster, TankMonster, FastMonster, or ShooterMonster) with the specified x and y coordinates.

- List<MonsterType> getAllowedMonsterTypes(int currentwave)	Determines which types of monsters are allowed based on the current wave. 1. Always includes BasicMonster. 2. Adds TankMonster if the wave is 1 or higher, FastMonster if the wave is 2 or higher, and ShooterMonster if the wave is 3 or higher. 3. Returns the list of allowed monster types.

3.8)CLass:WaveManager

3.8.1)Fields

-final WaveSpawner wavespawner	The object responsible for spawning waves of monsters.
-final ArrayList<Integer> waves	A list representing the number of monsters to spawn in each wave.
-final Player player;	The player participating in the game.
-int currentWave;	Tracks the current wave number.
-boolean spawningWave;	Flag indicating if a wave is currently spawning.
-boolean showItemSelection;	Flag to show item selection after each wave.
-ArrayList<Item> rewardItems	List of items to be offered as rewards after each wave.
-volatile boolean stopSpawning	Flag to stop spawning waves.

3.8.2)Constructor

+WaveManager(WaveSpawner waveSpawner, Player player)	Initializes the WaveManager with a WaveSpawner and a Player. Sets initial values for currentWave, spawningWave, and showItemSelection.

3.8.3)Methods

+void stopWaveThread()	Stops the wave spawning thread by setting stopSpawning to true.
+void startWaveThread()	Starts a new thread to spawn waves. The method: 1. Loops through the list of waves, spawning a new wave and waiting for the monsters to be cleared before proceeding. 2. Calls generateRewardItems() to generate items for the player to choose from after each wave. 3. Waits for the player to select an item with waitForItemSelection(). 4. Calls cooldownBeforeNextWave() to wait 5 seconds before the next wave starts.
-void waitForMonstersToClear()	Waits until all monsters are cleared from the screen before proceeding to the next wave.
-void generateRewardItems()	Generates a list of reward items after each wave: 1. In the first wave, it adds various guns as rewards. 2. For subsequent waves, it adds power-up items like increased damage, speed, health, and faster reload.
-void waitForItemSelection()	Waits for the player to select an item within 15 seconds after each wave.
-void cooldownBeforeNextWave()	Waits for 5 seconds before starting the next wave.

-void delay(int milliseconds)	Pauses the execution for the specified number of milliseconds.
+int getCurrentWave()	Returns the current wave number.
+void setCurrentWave(int currentWave)	Sets the current wave number.
+ArrayList<Item> getRewardItems()	Returns the list of reward items.
+ArrayList<Integer> getWaves()	Returns the list of wave sizes.
+boolean isShowItemSelection()	Returns whether the item selection is being shown.
+void setShowItemSelection(boolean showItemSelection)	Sets whether to show the item selection screen.
+void setSpawningWave(boolean spawningWave)	Sets the flag indicating whether a wave is currently spawning.

3.9)Class:WaveSpawner

3.9.1)Fields

-final Player player	The player participating in the game.
-final RandomMonster randomMonster	The object responsible for generating random monsters.

3.9.2)Constructor

+WaveSpawner(Player player, RandomMonster randomMonster)	Initializes the WaveSpawner with a Player and a RandomMonster instance.

3.9.3)Methods

+void spawnWave(int currentWave, int monstersToSpawn)	Spawns a wave of monsters by: 1. Looping monstersToSpawn times, creating a new monster each iteration. 2. If it is the last monster of the wave

	and the current wave is 4 or higher, it spawns a boss or mini-boss instead of a regular monster. 3. Each monster is spawned at a valid position determined by the findValidSpawnPosition() method. 4. Introduces a 2-second delay between monster spawns.
-boolean isLastMonster(int currentWave, int i, int monstersToSpawn)	Checks if the current monster is the last one in the wave, which will trigger a special boss or mini-boss spawn if the current wave is 4 or higher.
-void spawnBossOrMiniBoss(int currentWave)	Spawns a boss or mini-boss based on the current wave: 1. Wave 4 spawns a MiniBoss. 2. Wave 5 spawns two MiniBosses. 3. Wave 6 spawns a Boss.
-Position findValidSpawnPosition()	Finds a valid spawn position for a monster by: 1. Generating a random position within the screen boundaries. 2. Ensuring the position does not overlap with the player's safe zone or existing monsters.
-boolean overlapsWithExistingMonsters(int x, int y)	Checks if a monster at the given position (x, y) overlaps with any existing monsters.
-void delay(int milliseconds)	Pauses the execution for the specified number of milliseconds.

4)Package:gun

4.1)Abstract Class:Gun Implements Item

4.1.1)Fields

-int firerate	The fire rate of the gun, determining how fast the gun can shoot.
-int damage	The base damage dealt by the gun.
-int bonusdamage	An optional bonus damage that can be added to the gun's damage. Default is 0.
-boolean canshoot	A flag indicating whether the gun is ready to shoot.
-long lastshottime	The timestamp of the last time the gun was fired.

4.1.2)Constructor

+Gun(int damage, int firerate)	Initializes the Gun with the specified damage and fire rate. The canshoot flag is set to true, and bonus damage is set to 0 by default.

4.1.3)Methods

+abstract void fire(int mouseX, int mouseY, ArrayList<Bullet> bullets, Player p)	An abstract method that must be implemented in subclasses. It defines how the gun fires, adding bullets to the bullets list and interacting with the player.
+ String getCooldownTimerText()	Returns a formatted string indicating the remaining cooldown time of the gun. If the gun is ready to shoot, it returns "Ready". Otherwise, it returns the remaining cooldown in seconds.
+ void use(Player player)	Updates the player's current gun by

	removing the existing gun and adding this gun to the player's inventory.
Getter and Setter for all fields	

4.2)Class:BasicGun extends Gun

4.2.1)Fields

-String name	The name of the gun, set to "Pistol"

4.2.2)Constructor

+BasicGun(int damage, int firerate)	Initializes the BasicGun with a specified damage and fire rate. The name is set to "Pistol"

4.2.3)Methods

+void fire(int mouseX, int mouseY, ArrayList<Bullet> bullets, Player p)	Fires a bullet from the player's position towards the mouse coordinates, adding the bullet to the bullets list. 1. Checks if the gun is ready to shoot (isCanshoot()). 2. If ready, creates a new Bullet with the player's position and the target mouse coordinates. 3. Sets Canshoot to false to prevent immediate re-shooting, and updates the time of the last shot. 4. Starts a new thread that waits for the fire rate duration before allowing another shot (Canshoot is set back to true).
+String getName()	Returns the name of the gun, which is "Pistol".

+ String getType()	Returns “Gun-Pistol”.

4.3)Class:MachineGun extends BasicGun

4.3.1)Fields

-String name	The name of the gun.

4.3.2)Constructor

+Machinegun(int damage, int firerate)	Initializes the MachineGun with specified damage and fire rate.

4.3.3)Methods

+String getName()	Returns the name of the gun, which is “MachineGun”.
+String getType()	Returns ”Gun - MachineGun”.

4.4)Class:Bazooka extends Gun

4.4.1)Fields

-String name	The name of the gun, set to "Bazooka".

4.4.2)Constructor

+Bazooka(int damage, int firerate)	Initializes the Bazooka with a specified damage and fire rate. The name is set to "Bazooka".

4.4.3)Methods

+ String getName()	Returns the name of the gun, which is "Bazooka".
+ String getType()	Returns the type of the gun, which is "Gun -

	Bazooka".
+ void fire(int mouseX, int mouseY, ArrayList<Bullet> bullets, Player p)	Fires a missile bullet from the player's position towards the mouse coordinates, adding the missile to the bullets list. 1. Checks if the gun is ready to shoot (isCanshoot()). 2. If ready, creates a new MissileBullet with the player's position and target mouse coordinates. 3. Sets Canshoot to false to prevent immediate re-shooting, and updates the time of the last shot. 4. Starts a new thread that waits for the fire rate duration before allowing another shot (Canshoot is set back to true).

4.5)Class:ShotGun extends Gun

4.5.1)Fields

-String name	The name of the gun, set to "Triple Gun".

4.5.2)Constructor

+ShotGun(int damage, int firerate)	Initializes the ShotGun with the specified damage and fire rate. The name is set to "TrippleGun".

4.5.3)Methods

+String getName()	Returns the name of the gun, which is "TrippleGun".
+String getType()	Returns the type of the gun, which is "Gun - TrippleGun".
+void fire(int mouseX, int mouseY,	Fires three bullets (shotgun blast) from the

ArrayList<Bullet> bullets, Player p)	player's position towards the mouse coordinates, adding the bullets to the bullets list. 1. The factor variable is used to slightly offset the position of each bullet, creating a spread effect. 2. Checks if the gun is ready to shoot (isCanshoot()). 3. If ready, it adds three bullets at slightly different positions by adjusting their x and y coordinates. 4. Sets Canshoot to false to prevent immediate re-shooting and updates the time of the last shot. 5. Starts a new thread that waits for the fire rate duration before allowing another shot (Canshoot is set back to true).

5)Package:main

5.1)Class:Game extends JPanel implements ActionListener, KeyListener, MouseListener, MouseMotionListener

5.1.1)Fields

-static Timer timer	The timer that controls the game loop.
-static Player player	The player object in the game.
-ArrayList<Integer> keep	A list used for tracking or storing integer values (the exact use is not clear from the provided code).
-static long startTime	The start time of the game.
-long totalPlaytime	The total time the game has been played.
-Image backgroundImage	The background image of the game.

-Image overlayImage	An overlay image (possibly used for UI elements like health bars or status effects).
-Clip clip	An audio clip (possibly for background music or sound effects).
-JFrame frame	The main game window.
-static WaveManager waveManager	Responsible for managing and spawning game waves.
-RandomMonster randomMonster	The random monster generator used to spawn monsters.
-static MonsterHandler monsterHandler	Manages the list of monsters and handles interactions with them.

5.1.2)Constructor

+Game(JFrame frame)	Initializes the Game object with the specified frame. The constructor: 1. Initializes the player, game timer, wave manager, and monster handler. 2. Loads background and overlay images. 3. Adds key and mouse listeners for user input. 4. Sets the preferred size of the game panel to 1920x1080. 5. Starts the game and wave threads.

5.1.3)Methods

+static void resetGame()	Resets the game to its initial state: 1. Stops the current wave thread. 2. Clears existing monsters. 3. Resets the player’s health and equipment.

	4. Reinitializes the wave manager and monster handler. 5. Starts the game and wave threads again.
+void addNotify()	Ensures the game panel has focus when it is added to the window.
+void showGameEndUI(boolean won)	Displays the game over UI (either won or lost) and clears the monsters from the game.
+void goToMenu()	Switches to the main menu by removing the game panel and adding the Menu panel to the frame.
+static void startWaveThread()	Starts the wave spawning thread in the WaveManager.
-void delay(int milliseconds)	Pauses the game for the specified number of milliseconds.

5.2)Class:Menu extends JPanel implements ActionListener

5.2.1)Fields

-JFrame frame	The main application frame contains the menu.
-Image backgroundImage	The background image of the menu.
-Image titleImage	The title image displayed on the menu.
-JButton startButton	The button to start the game.
-JButton helpButton	The button to display help or instructions.
-JButton exitButton	The button to exit the game.
-static Clip clip	The audio clip for playing background music.

5.2.2)Constructor

+ Menu(JFrame frame)	Initializes the menu panel with a given frame 1. Loads and sets the backgroundImage and titleImage from resources. 2. Configures the panel layout and size (1920x1080). 3. Creates and styles the buttons (Start, Help, Exit) with the createStyledButton() method, adding them to the panel. 4. Stops any currently playing background music and starts the menu's background music.

5.2.3)Methods

-void setCustomFont(JButton b, int fontSize)	-Applies a custom font to a button using the PressStart2P.ttf font resource. -Falls back to the "SansSerif" font if the custom font cannot be loaded.
+void playBackgroundMusic()	-Plays the background music for the menu in a loop using the Theme.wav file. -Throws an exception if the music file is not found or cannot be loaded.
+static void stopBackgroundMusic()	Stops the currently playing background music if it is running.
-JButton createStyledButton(String text, int x, int y)	1. Creates a styled button with the given text, x, and y position. 2. Adds custom styling (e.g., font, color, border) and mouse hover effects. 3. Adds the current instance as an ActionListener.
#void paintComponent(Graphics g)	-Paints the background and title images

	onto the menu panel. -Ensures the images scale to the panel size and centers the title image horizontally.

6)Package:monster

6.1)Abstract Class:Monster

6.1.1)Fields

#int x	The x-coordinate of the monster's position.
#int y	The y-coordinate of the monster's position.
#int hp	The monster's health points.
#int speed	The speed at which the monster moves (default: 2).
#bool isFacingRight	Indicates whether the monster is facing right.

6.1.2)Constructor

+Monster(int x, int y, int hp)	Initializes the monster's position (x, y) and health points (hp).

6.1.3)Methods

+void update(Player player, ArrayList<Monster> allMonsters)	-Moves the monster toward the player's position while avoiding collisions with other monsters. -Updates the monster's position (x, y) only if the new position does not result in a collision with other monsters.
#void setCustomFont(Graphics g, int fontSize)	Loads and sets a custom font for rendering text. Falls back to "SansSerif" if the custom font cannot be loaded.

+abstract void attack(Player player)	Defines an abstract method for monster-specific attack behavior. Subclasses must implement this method.
+int getX()	Returns the x-coordinate of the monster's position.
+int getY()	Returns the y-coordinate of the monster's position.
+abstract void draw(Graphics g)	Draws the monster. Subclasses must provide the implementation.
+boolean isColliding(Player player)	Checks if the monster collides with the player's bounding box.
+boolean isColliding(Bullet bullet)	Checks if the monster collides with a bullet's bounding box.
+int getHp()	Returns the current health points of the monster.
+void setHp(int hp)	Sets the monster's health points.
+void damage(Bullet bullet)	Reduces the monster's health by the damage value of the bullet.
+boolean isFacingRight()	Returns whether the monster is facing right.
+abstract Rectangle getBounds()	Returns the Monster hitbox.

6.2)Class:BasicMonster extends Monster

6.2.1)Fields

-BufferedImage monsterImage	The image representing the BasicMonster

6.2.2)Constructor

+BasicMonster(int x, int y)	Initializes the BasicMonster at the specified position (x, y) with a default health of 15. 1. Randomly selects one of four

	images (BasicMonster1.png, BasicMonster2.png, etc.) to represent the monster. 2. Attempts to load the selected image. If loading fails, an error message is printed, and no image is set.

6.2.3)Methods

+void attack(Player player)	A placeholder for the attack behavior. Currently, this method is empty but can be customized for specific attacks.
+void draw(Graphics g)	1. Draws the BasicMonster on the screen a. If an image is available, it draws the monster's image. The image is flipped horizontally if the monster is facing right. b. If no image is available, it falls back to drawing a red rectangle. 2. Displays the monster's health (hp) above it: a. Draws the text shadow in black and the main text in red. b. Uses a custom font for the text (PressStart2P.ttf) if available.
+Rectangle getBounds()	Returns BasicMonster hitbox.

6.3)Class:FastMonster extends Monster

6.3.1)Fields

-BufferedImage monsterImage	The image represents the FastMonster.

6.3.2)Constructor

+FastMonster(int x, int y)	Initializes a FastMonster at the specified position (x, y) with default health (10) and increased speed (4). 1. Attempts to load the image file "FastMonster.png" from the resources. 2. If the image cannot be loaded, an error message is printed.

6.3.3)Methods

+void attack(Player player)	Placeholder method for attack behavior. Currently empty, allowing for future customization.
+void draw(Graphics g)	1. Draws the FastMonster on the screen: a. If the image is available, it is drawn at the monster's position. The image is flipped horizontally if the monster is facing right. b. If the image is unavailable, an orange rectangle is drawn as a fallback. 2. Displays the monster's health (hp) above it: a. A black shadow is drawn beneath the health text for readability. b. The health text is drawn in

	red using a custom font (PressStart2P.ttf) if available.
+Rectangle getBounds()	Returns FastMonster hitbox.

6.4)Class:TankMonster extends Monster

6.4.1)Fields

-BufferedImage monsterImage	The image representing the TankMonster

6.4.2)Constructor

+TankMonster(int x, int y)	Initializes the TankMonster at the specified position (x, y) with high health points (40). 1. Randomly selects one of three image variations (TankMonster1.png, TankMonster2.png, TankMonster3.png). 2. Attempts to load the selected image from resources. 3. Prints an error message if the image fails to load.

6.4.3)Methods

+void draw(Graphics g)	1. Draws the TankMonster on the screen: a. If an image is available, it is drawn at the monster's position. The image is flipped horizontally if the monster is facing right. b. If the image is unavailable, a green rectangle is drawn as a fallback. 2. Displays the monster's health (hp) above it:

	a. A black shadow is drawn beneath the health text for readability. b. The health text is drawn in red using a custom font (PressStart2P.ttf) if available.
+Rectangle getBounds()	-Returns TankMonster hitbox.

6.5)Class: ShooterMonster extends Monster

6.5.1)Fields

-long lastShotTime	Tracks the time of the last shot fired by the monster.
-ArrayList<Bullet> bullets	A list of bullets fired by the ShooterMonster.
-BufferedImage monsterImage	The image represents the ShooterMonster.

6.5.2)Constructor

+ShooterMonster(int x, int y)	Initializes a ShooterMonster at the specified position (x, y) with default health (10). 1. Attempts to load the image file "ShooterMonster1.png" from resources. 2. Prints an error message if the image fails to load.

6.5.3)Methods

+void attack(Player player)	-Fires a bullet at the player if at least 2 seconds have passed since the last shot. -Creates a new Bullet aimed at the player's position with a damage value of 1.

	-Adds the bullet to the bullets list and updates lastShotTime.
+void draw(Graphics g)	1. Draws the ShooterMonster on the screen: a. If the image is available, it is drawn at the monster's position. The image is flipped horizontally if the monster is facing right. b. If the image is unavailable, a magenta rectangle is drawn as a fallback. 2. Displays the monster's health (hp) above it: a. A black shadow is drawn beneath the health text for readability. b. The health text is drawn in red using a custom font (PressStart2P.ttf) if available. c. Draws all the bullets fired by the ShooterMonster.
+Rectangle getBounds()	Returns ShooterMonster hitbox

6.6)Class:MiniBoss extends Monster

6.6.1)Fields

-long lastShotTime	Tracks the time of the last shot fired by the MiniBoss.
-long bullets	A list of bullets fired by the MiniBoss.
-long miniBossImage	The image represents the MiniBoss.

6.6.2)Constructor

+MiniBoss(int x, int y)	Initializes a MiniBoss at the specified

	position (x, y) with a default health of 200. 1. Attempts to load the image file "MiniBoss.png" from resources. 2. Prints an error message if the image fails to load.

6.6.3)Methods

+void attack(Player player)	-Fires a burst of bullets toward the player if at least 1.3 seconds have passed since the last shot. -Bullets are fired in a spread pattern: 1. 7 bullets are created with increasing offsets from the player's position. -Each bullet has a damage value of 2.
+void update(Player player, ArrayList<Monster> allMonsters)	Moves the monster toward the player by calling super.update(). 1. Calls the attack() method to fire bullets at the player. 2. Updates all bullets in the bullets list, moving them and checking for: a. If a bullet hits the player, it deals damage and is removed. b. Bullets that go off-screen are removed.
+void draw(Graphics g)	1. Draws the MiniBoss on the screen: a. If the image is available, it is drawn at the monster's position. The image is flipped horizontally if the monster is facing right. b. If the image is unavailable,

	an orange rectangle is drawn as a fallback. 2. Displays the monster's health (hp) above it: a. A black shadow is drawn beneath the health text for readability. b. The health text is drawn in red using a custom font (PressStart2P.ttf) if available. 3. Draws all the bullets fired by the MiniBoss in red.
+Rectangle getBounds()	Return MiniBoss hitbox.

6.7)Class:Boss extends Monster

6.7.1)Fields

-long lastShotTime	Tracks the time of the last attack by the Boss.
-ArrayList<Bullet> bullets	A list of bullets fired by the Boss.
-Random random	A Random object to introduce variability in the Boss's attacks.
-BufferedImage bossImage	The image representing the Boss.

6.7.2)Constructor

+ Boss(int x, int y)	Initializes a Boss at the specified position (x, y) with high health points (500). 1. Attempts to load the "Boss.png" image from resources. 2. Prints an error message if the image fails to load.

6.7.3)Methods

+void attack(Player player)	1. Alternates between two attack

	patterns every second: a. Spread Attack: Fires 10 bullets in a spread pattern aimed at the player's position. Each bullet deals 3 damage. b. Missile Attack: Fires a single MissileBullet targeting the player's position with 2 damage. 2. Uses random.nextBoolean() to decide the attack type. 3. Updates lastShotTime to ensure attacks occur at least 1 second apart.
+void update(Player player, ArrayList<Monster> allMonsters)	1. Calls the attack() method to fire bullets at the player. 2. Iterates through the bullets list to: a. Update each bullet's position. b. Remove bullets that collide with the player or go out of screen bounds.
+void draw(Graphics g)	1. Draws the Boss on the screen: a. If the image is available, it is drawn scaled to a size of 200x200 centered on the Boss's position. b. If the image fails to load, a red rectangle is drawn as a fallback. 2. Displays the Boss's health (hp) above it using a custom font (PressStart2P.ttf) if available. 3. Draws all bullets fired by the Boss in red.

+Rectangle getBounds()	-Returns the bounding box of the Boss for collision detection. -The bounds are based on the Boss's position (x, y) and size (200x200).

7)Package:ui

7.1)Class:EntityDrawer

7.1.1)Fields

-Player player	The player object to be drawn on the screen.
-List<Monster> monsters	A list of monsters to be drawn on the screen.

7.1.2)Constructor

+EntityDrawer(Player player, List<Monster> monsters)	Initializes the EntityDrawer with a Player and a list of Monster objects.

7.1.3)Methods

+void draw(Graphics g)	-Draws the Player and all Monster objects onto the screen. -First, the player.draw(g) method is called to render the player. -Then, for each monster in the monsters list, the monster.draw(g) method is called to render each monster.

7.2)Class:HUDDrawer

7.2.1)Fields

-Player player	The player whose information (such as health) is displayed in the HUD (Heads-Up Display).

-long startTime	The start time of the current game session used to track elapsed time.
-int currentWave	The current wave number of the game.
-int waveSize	The total number of waves in the game.

7.2.2)Constructor

+HUDDrawer(Player player, long startTime, long totalPlaytime, int currentWave, int waveSize)	Initializes the HUDDrawer with the given player, start time, current wave, and wave size.

7.2.3)Methods

-void setCustomFont(Graphics g, int fontSize)	-Loads and sets a custom font (PressStart2P.ttf) for drawing the HUD text. -Falls back to the SansSerif font if the custom font cannot be loaded.
+void draw(Graphics g)	1. Draws the HUD elements on the screen, such as: a. Player's HP: Displays the current health of the player. b. Current Wave: Shows the current wave out of the total wave size. c. Elapsed Time: Displays the time elapsed since the game started, in seconds. 2. Uses the custom font and sets the color to PINK for text drawing.

7.3)Class:ItemSelection

7.3.1)Fields

-ArrayList<Item> rewardItems	A list of available reward items to be displayed.

-ArrayList<Integer> keep	A list of selected indices that will display the reward items.

7.3.2)Constructor

+ItemSelectionDrawer(ArrayList<Item> rewardItems, ArrayList<Integer> keep)	Initializes the ItemSelectionDrawer with the provided list of rewardItems and the keep list for selected items.

7.3.3)Methods

- void setCustomFont(Graphics g, int fontSize)	-Loads and sets a custom font (PressStart2P.ttf) for the text displayed in the item selection screen. -Falls back to the default "SansSerif" font if the custom font cannot be loaded.
+void draw(Graphics g, Point mousePosition)	Renders the item selection UI: 1. Background and Title: a. Displays a semi-transparent background and the title "Choose a reward item". 2. Item Selection: a. If the keep list is empty, it randomly selects 3 items (without duplicates) from the rewardItems list. 3. Item List: a. Draws the names of the selected reward items (keep) on the screen. 4. Hover Effects: a. Highlights the hovered item with a yellow color when the mouse is over an item.