Report

Last Survivor: Nakhon Si Thammarat

Prepared by

Yanason Saksritrakun 6504062663095

Present to

Asst.Prof. Sathit Prasomphan

Object Oriented Programming subject

Term 1/2023

Introduction

Project name: Last Survivor: Nakhon Si Thammarat

Prepared by: Yanason Saksritrakun

Instructor: Asst.Prof. Sathit Prasomphan

Chapter 1: Introduction and Significance

This project aimed to assess proficiency in Object-Oriented Programming through the application of acquired concepts to develop a 2D game.

Project type:

Game

Project Benefits:

1. Enjoyment and Entertainment
2. Enhances Quick Decision-Making Skills

Table of Work Plan (October-November)

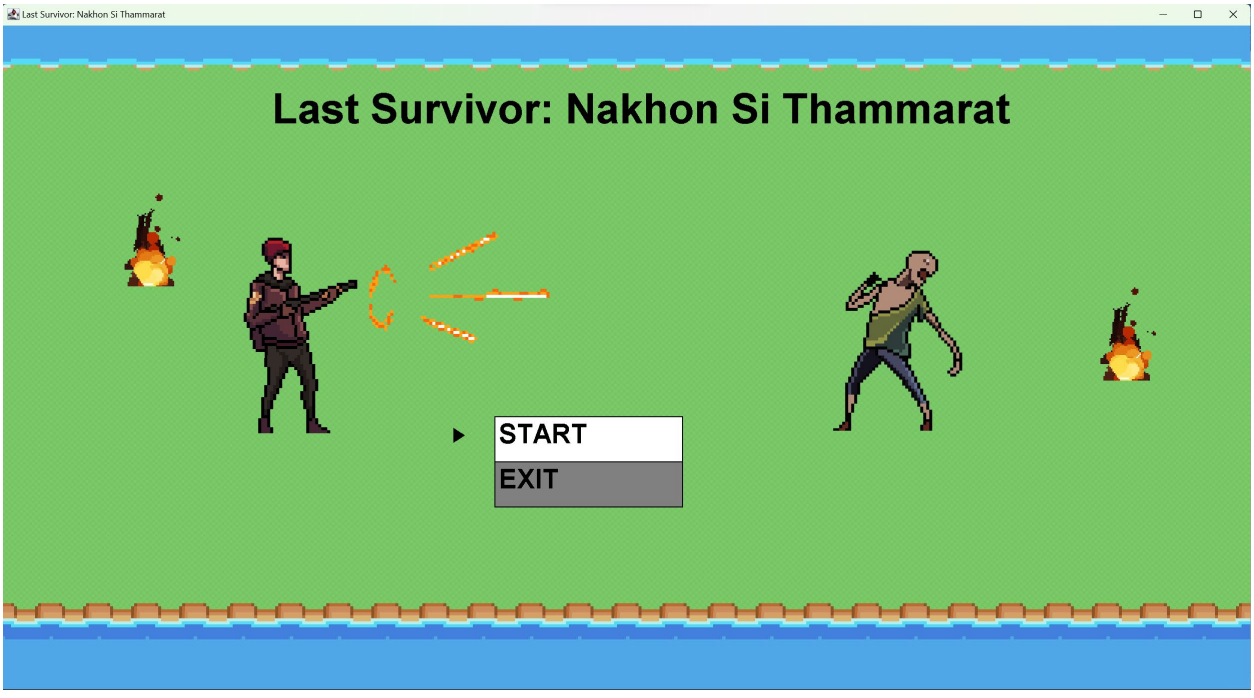
No.	Activity	17-24	25-1	2-8
1	Find Pictures, Graphics, or Designs in the Game			
2	Study Various Relevant Information			
3	Start Writing a Program			
4	Check for Errors			
5	Prepare Documents			

Chapter 2 : Development

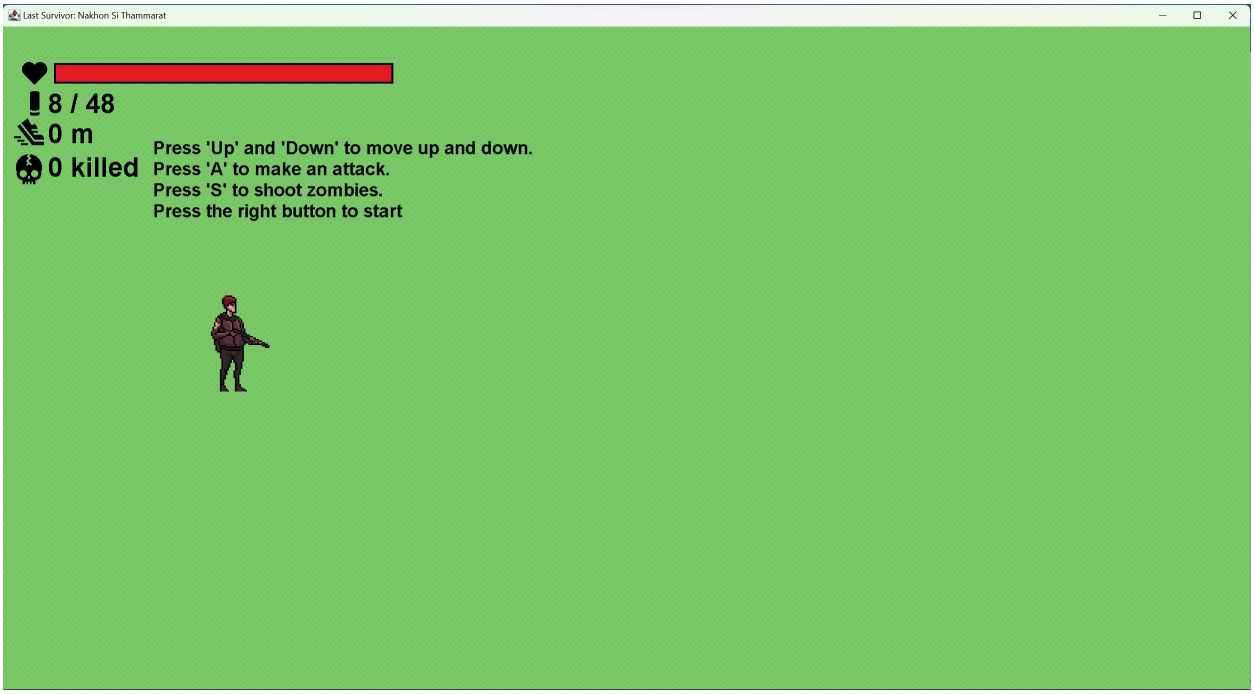
Story:

In a desolate world, "Film" stands as the lone survivor on Earth. Prepared to navigate the chaotic horde of zombies in the heart of Nakhon Si Thammarat, survival is the only goal.

How to play?



Select “START” using up&down button and press spacebar to play

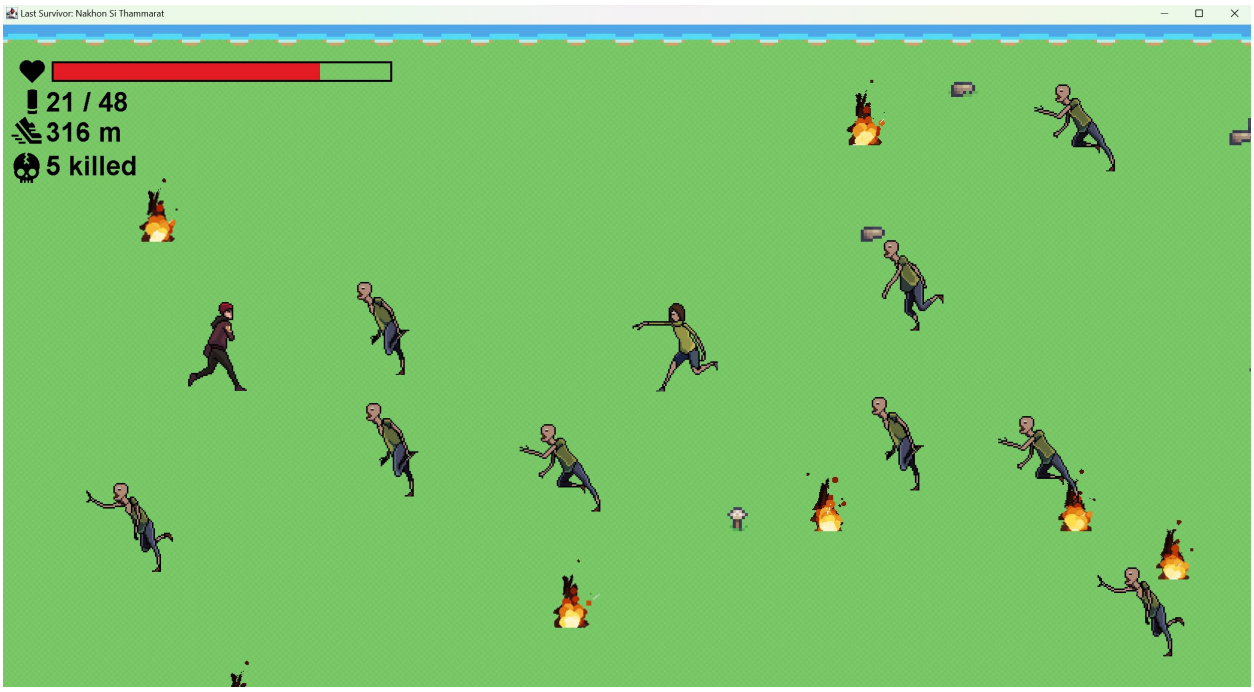


Press “Up” and “Down” to move.

Press “A” to attack zombies.

Press “S” to shoot zombies.

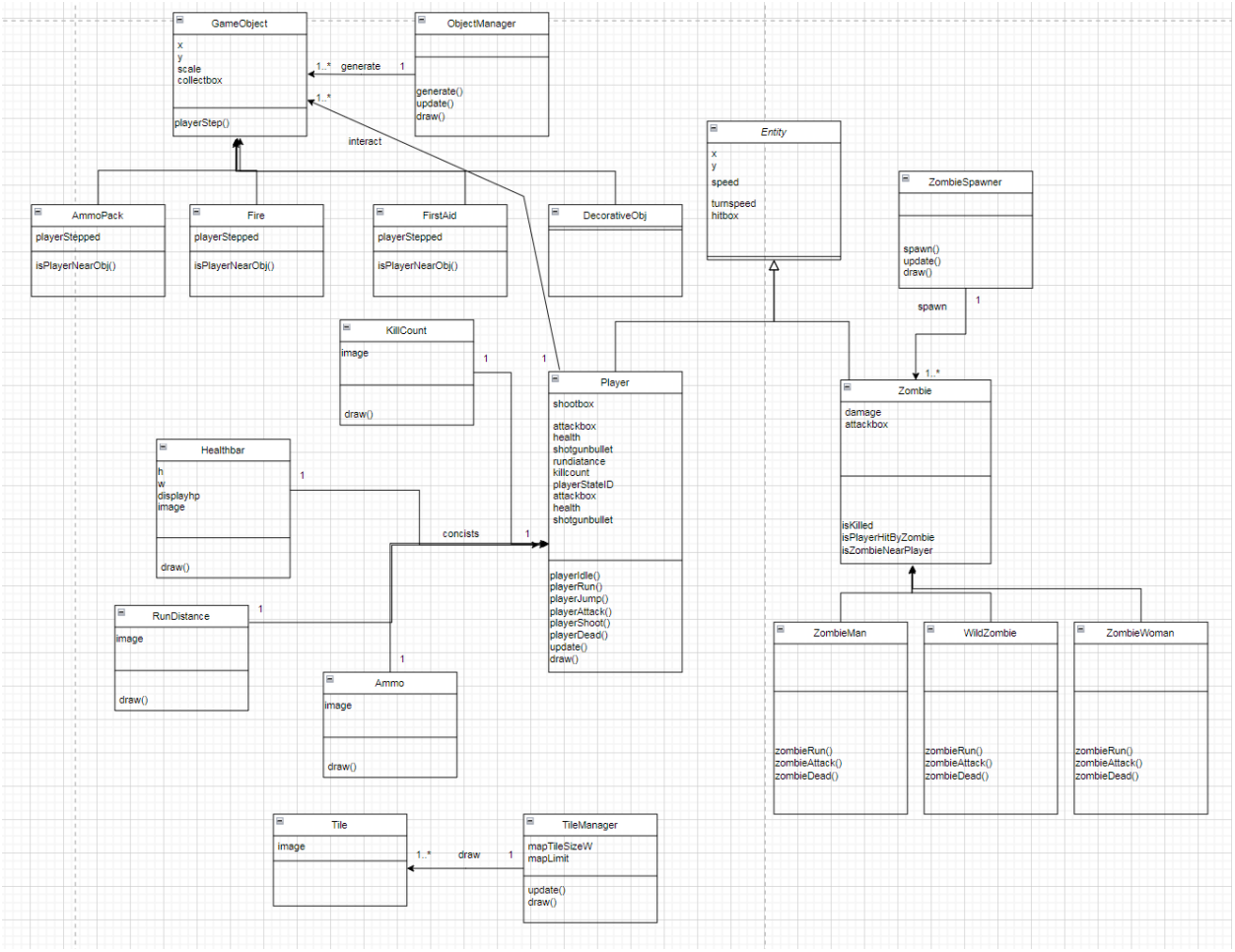
Press the “Right” button to start running.



- Try to avoid the zombies and fire or you will be attacked. If you can't avoid it, shoot them.
- Each zombie has a different speed, be careful.
- Run as far as you can.

After game over, choose to restart or exit.

Class diagram



A solitary player embarks on the journey, armed with a health rating, attack range, and capacity to withstand damage. The arsenal includes shotgun shells for interacting with objects—

taking damage from fire, collecting items. Among the diverse objects are decorations, lights, first aid kits, and ammunition. The player faces various zombie adversaries, including zombies men, wild zombies, and zombie women

OOP

Constructor

Entity class (for player, zombie)

Set size and frames (sprite sheet)

```
public Entity() {
    setPostion(x:0, y:0);
    setSize(DefaultPanel.tileSize, DefaultPanel.tileSize);
    this.BI = new BufferedImage[20];
}

public Entity(int sizeW, int sizeH, int maxFrames) {
    setPostion(x:0, y:0);
    setSize(DefaultPanel.tileSize*sizeW, DefaultPanel.tileSize*sizeH);
    this.BI = new BufferedImage[maxFrames];
}

public Entity(int x, int y, int sizeW, int sizeH, int maxFrames) {
    setPostion(x, y);
    setSize(DefaultPanel.tileSize*sizeW, DefaultPanel.tileSize*sizeH);
    this.BI = new BufferedImage[maxFrames];
}
{
    framedelay = frames;
}

public void setHitbox(int x, int y, int w, int h) {
    hitbox[0] = x;
    hitbox[1] = y;
    hitbox[2] = w;
    hitbox[3] = h;
}
```

Tile (map image)

Pass id for setting image

```java
public class Tile {
    public BufferedImage image;

    public Tile(int id) {
        try {
            this.image = ImageIO.read(getClass().getResourceAsStream(String.format(format:"/image/tiles/%d.png", id)));
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
}
```

TileManager (Draw map)

Pass object Player, GamePanel, KeyHandler

Get gameStateID from class GamePanel.

KeyHandler uses for move player by camera view.

And get player speed.

```java
public class TileManager {
    GamePanel gp;
    KeyHandler keyH;
    Player player;
    Tile[] tile;
    public int temp_distanceX, temp_distanceY;

    private final int mapTileSizeW = 16;
    private final int[] mapLimit = {mapTileSizeW*-1, mapTileSizeW+DefaultPanel.maxScreenRow};

    public TileManager(GamePanel gp, KeyHandler keyH, Player player) {
        this.temp_distanceY = 0;
        this.gp = gp;
        this.keyH = keyH;
        this.player = player;
        this.tile = new Tile[10];
        setTileImage();
    }
```

ObjectManager (generate object such as fire, items)

Pass object Player, TileManager

Get player coordinates from TileManager (coordinates by camera view)

And get player x, y to check if item is collected by player.

```java
public class ObjectManager {
    private GamePanel gp;
    private Player player;
    private TileManager tm;
    private ArrayList<GameObject> go;

    public ObjectManager(GamePanel gp, TileManager tm, Player player) {
        this.gp = gp;
        this.tm = tm;
        this.player = player;
        this.go = new ArrayList<GameObject>();
    }

    public void update() {
        if (this.gp.gameState == 1) {
            generate(new Fire(), 0.09*(1+gp.dificultRate));
            generate(new AmmoPack(), rate:0.003);
            generate(new FirstAid(), rate:0.003);
            generate(new DecorativeObj(), rate:0.1);
        }
```

GamePanel (Draw all objects)

Pass JFrame to swap a panel to another panel.

```
KeyHandler keyH;
Player player;
HealthBar hb;
Ammo ammo;
KillCount kc;
RunDistance rd;
TileManager tileM;
ObjectManager objM;
ZombieSpawner zbs;

private String[] buttonText = { "Restart", "Exit" };
private int selectedButtonIndex = 0;

public GamePanel(JFrame window) {
    this.window = window;
    this.keyH = new KeyHandler();
    this.player = new Player(this, keyH);
    this.hb = new HealthBar(player);
    this.ammo = new Ammo(player);
    this.kc = new KillCount(player);
    this.rd = new RunDistance(this, player);
    this.tileM = new TileManager(this, keyH, player);
    this.objM = new ObjectManager(this, tileM, player);
    this.zbs = new ZombieSpawner(this, tileM, player);
```

Player stat class

Get some attribute from player, set image and draw.

```
public class RunDistance {
    private Player player;
    private GamePanel gp;
    private BufferedImage image = null;

    public RunDistance(GamePanel gp, Player player) {
        this.player = player;
        this.gp = gp;
        try {
            image = ImageIO.read(getClass().getResourceAsStream(name:"/image/run.png"));
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
```

```java
public class Ammo {
    private Player player;
    private BufferedImage image = null;

    public Ammo(Player player) {
        this.player = player;
        try {
            image = ImageIO.read(getClass().getResourceAsStream(name:"/image/ammoicon.png"));
        } catch (IOException e) {
            e.getStackTrace();
        }
    }
```

```java
public class HealthBar {
    private Player player;
    private int displayhp;
    private BufferedImage image = null;
    private final int h = 25;
    private final int w = 450;

    public HealthBar(Player player) {
        this.player = player;
        this.displayhp = player.maxHealth;
        try {
            image = ImageIO.read(getClass().getResourceAsStream(name:"/image/heart.png"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```java
public class KillCount {
    private Player player;
    private BufferedImage image = null;

    public KillCount(Player player) {
        this.player = player;
        try {
            image = ImageIO.read(getClass().getResourceAsStream(name:"/image/skull.png"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

Abstract zombie class that inherits from Entity class

```
abstract class Zombie extends Entity {
    String path;
    int zombieStateID;
    int xDiff, yDiff, sizeW, sizeH;
    int damage;
    public boolean attacked = false;
    public boolean isAlive = true;
    public int[] attackbox = new int[4];

    BufferedImage image;

    public abstract void zombieRun();
    public abstract void zombieAttack();
    public abstract void zombieDead();

    public Zombie(int x, int y) {
        super(x, y, sizeW:3, sizeH:3, maxFrames:10);
        setAttackbox(x, y, x, y);
        zombieRun();
    }

    public void setImage() {
        try {
            for (int i=0; i < frames; i++) {
                BI[i] = ImageIO.read(getClass().getResourceAsStream(String.format(path, i+1)));
            }
        } catch (IOException e) {
            e.printStackTrace();
```

Type of zombie

```java
class ZombieMan extends Zombie {
    public ZombieMan(int x, int y) {
        super(x, y);
    }

    public void zombieRun() {
        this.spriteNum = 0;
        this.zombieStateID = 1;
        this.damage = 5;
        setFrame(n:7);
        setSpeed(s:2);
        setTurnSpeed(s:1);
        setFrameDelay(s:8);
        this.xDiff = 80;
        this.yDiff = 90;
        this.sizeW = 35;
        this.sizeH = 50;
        path = "/image/zombiemanrun/Run_%d.png";
        setImage();

    }
```

```java
class ZombieWoman extends Zombie {
    public ZombieWoman(int x, int y) {
        super(x, y);
    }

    public void zombieRun() {
        spriteNum = 0;
        this.zombieStateID = 1;
        this.damage = 25;
        setFrame(n:7);
        setSpeed(s:10);
        setTurnSpeed(s:5);
        setFrameDelay(s:3);
        this.xDiff = 80;
        this.yDiff = 90;
        this.sizeW = 35;
        this.sizeH = 50;
        path = "/image/zombiewomanrun/Run_%d.png";
        setImage();


    }
```

```java
class WildZombie extends Zombie {
    public WildZombie(int x, int y) {
        super(x, y);
    }

    public void zombieRun() {
        spriteNum = 0;
        this.damage = 15;
        this.zombieStateID = 1;
        setFrame(n:8);
        setSpeed(s:4);
        setTurnSpeed(s:3);
        setFrameDelay(s:6);
        this.xDiff = 70;
        this.yDiff = 100;
        this.sizeW = 35;
        this.sizeH = 50;
        path = "/image/wildzombierun/Run_%d.png";
        setImage();

    }
```

Each zombie has a different speed, turn speed, damage, and frames (for animation)

Player is a type of Entity too

```
public class Player extends Entity {
    GamePanel gp;
    KeyHandler keyH;
    String path;
    public boolean isAlive = true;
    public int[] shootbox = {280, (DefaultPanel.screenHeight / 2) - 130, 230, 100};
    public int[] attackbox = {310, (DefaultPanel.screenHeight / 2) - 100, 50, 50};
    public int[] stepbox = {270, (DefaultPanel.screenHeight / 2) - 20, 45, 25};
    public final int maxHealth = 100;
    public int health;
    public int shotgunBullet;
    public int runDistance;
    public int killCount;
    public int playerStateID = 0;
    boolean on_action = false;


    public Player(GamePanel gp, KeyHandler keyH) {
        super(x:182, (DefaultPanel.screenHeight / 2) - 250, sizeW:4, sizeH:4, maxFrames:20);
        this.gp = gp;
        this.keyH = keyH;
        this.health = this.maxHealth;
        this.runDistance = 0;
        this.shotgunBullet = 8;
        this.killCount = 0;

        playerIdle();
        setImage();
    }
```

GameObject (Super abstract class)

```
public abstract class GameObject {
    boolean playerStepped = false;
    BufferedImage[] BI;
    int x, y;
    public int scale;
    public int spriteCounter = 0;
    public int spriteNum = 0;
    public int frames;
    public int framedelay;
    public int[] collectbox = new int[4];
    public int xDiff, yDiff;

    abstract public void setImage();

    public void playerStep(Player player) {}
```

Sub class of GameObject

```
public class FirstAid extends GameObject {
    boolean playerStepped = false;
    public FirstAid() {
        this.frames = 1;
        this.scale = 1;
        this.framedelay = 1;
        this.xDiff = 20;
        this.yDiff = 20;
        this.BI = new BufferedImage[this.frames];
        setCollectbox(this.x, this.y, w:20, h:20);
        setImage();
    }
```

```
public class Fire extends GameObject {
    boolean playerStepped = false;
    public Fire() {
        this.frames = 6;
        this.scale = 2;
        this.framedelay = 2;
        this.xDiff = 55;
        this.yDiff = 110;
        this.BI = new BufferedImage[this.frames];
        setCollectbox(this.x, this.y, w:20, h:20);
        setImage();
    }
```

```
public class AmmoPack extends GameObject {
    boolean playerStepped = false;

    public AmmoPack() {
        this.frames = 1;
        this.scale = 1;
        this.framedelay = 1;
        this.xDiff = 20;
        this.yDiff = 20;
        this.BI = new BufferedImage[this.frames];
        setCollectbox(this.x, this.y, w:20, h:20);
        setImage();
    }
```

```java
public class DecorativeObj extends GameObject {
    public int id;
    public DecorativeObj() {
        this.frames = 1;
        this.scale = 1;
        this.BI = new BufferedImage[1];
        setCollectbox(x:0, y:0, w:0, h:0);
        setImage();
    }

    public void setImage() {
        try {
            this.BI[0] = ImageIO.read(getClass().getResourceAsStream(String.format(format:"/image/decorativeobj/%d.png", RD.random(min:0, max:11))));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void update() {}


}
```

Polymorphism

```java
public class ObjectManager {
    private GamePanel gp;
    private Player player;
    private TileManager tm;
    private ArrayList<GameObject> go;

    public ObjectManager(GamePanel gp, TileManager tm, Player player) {
        this.gp = gp;
        this.tm = tm;
        this.player = player;
        this.go = new ArrayList<GameObject>();
    }
```

```java
Iterator<GameObject> it = this.go.iterator();
while (it.hasNext()) {
    GameObject gameObject = it.next();
    gameObject.spriteCounter++;

    if (gameObject.spriteCounter >= gameObject.framedelay) {
        gameObject.spriteNum++;
        gameObject.spriteCounter = 0;
    }
    if (gameObject.spriteNum >= gameObject.frames) {
        gameObject.spriteNum = 0;
    }
    gameObject.x -= player.speed;
    gameObject.updateCollectbox(tm.temp_distanceY);
    gameObject.playerStep(player);
    if (gameObject instanceof FirstAid) {
        FirstAid fa = (FirstAid)gameObject;
        if (fa.playerStepped) {
            it.remove();
            continue;
        }
```

In ObjectManager this code stores arraylist of GameObject

In ZombieSpawner

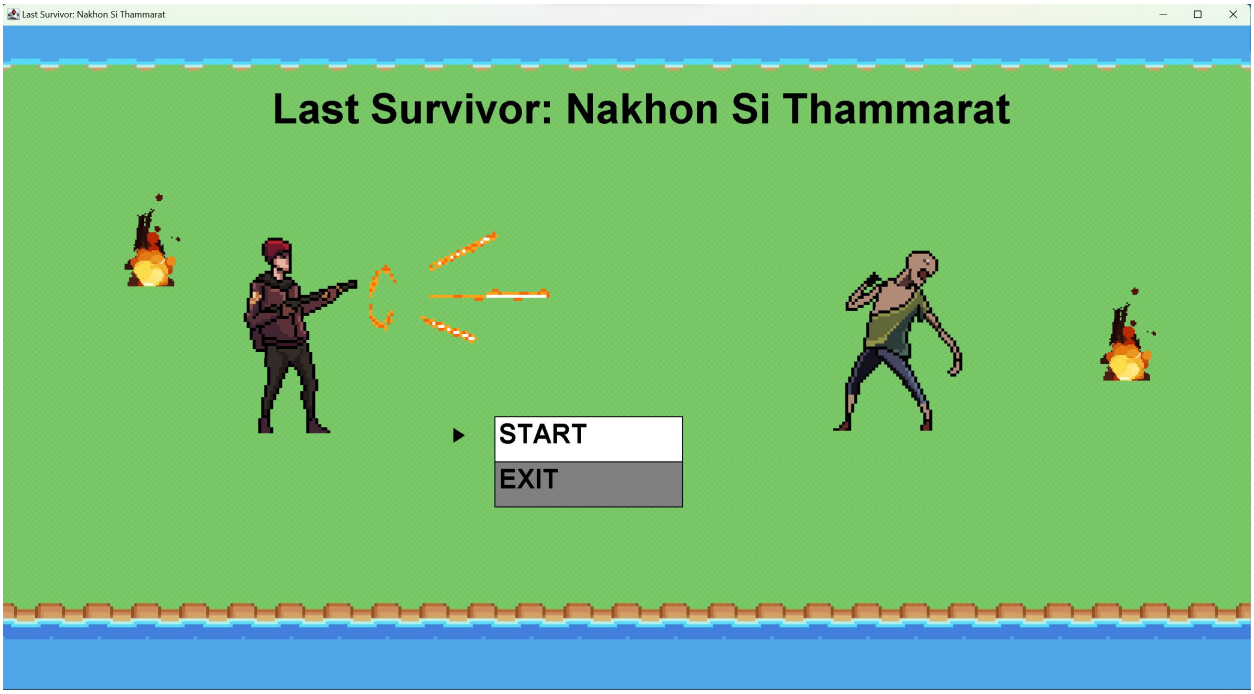
```
public class ZombieSpawner {
    Player player;
    GamePanel gp;
    TileManager tm;
    ArrayList<Zombie> zombies;

    public ZombieSpawner(GamePanel gp, TileManager tm, Player player) {
        this.gp = gp;
        this.tm = tm;
        this.player = player;
        this.zombies = new ArrayList<Zombie>();
    }
```

```
Iterator<Zombie> it = this.zombies.iterator();

while (it.hasNext()) {
    Zombie z = it.next();
```

For each item in ArrayList also uses polymorphism!

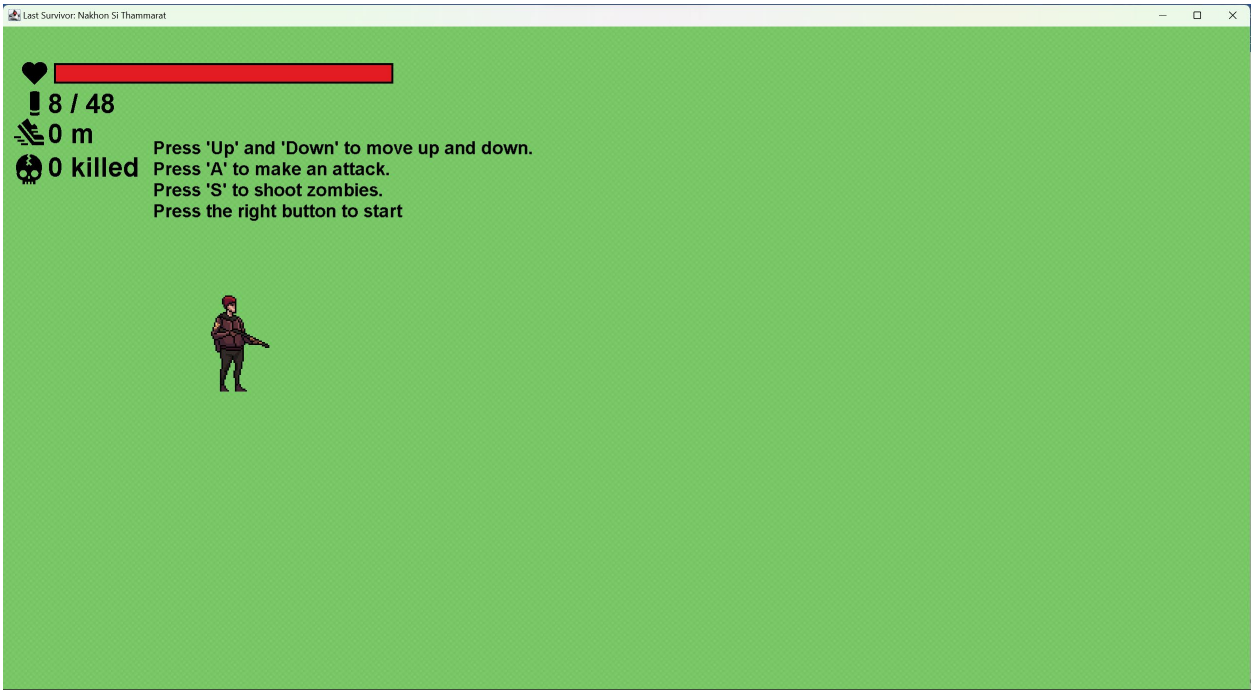


Components:

- Game title
- Start button
- Exit button
- Back ground

Event handling

- Press up&down
- Press spacebar to select



Components:

- Player health
- Ammo
- Run distance
- Kill count
- How to play message
- Icon

Event handling

- Press up&down to move
- Press A to attack
- Press S to shoot

- Restart button
- Exit to home button

Event handling

- Press up&down
- Press spacebar to select

My algorithms

Random integer between min to max. It's used to random position of generate game object

```java
package random;
public class RD {
    public static int random(int min, int max) {
        return (int) (Math.random() * (max - min + 1)) + min;
    }
}
```

Generate game object to screen with probability (this.go is an array list of GameObject)

```java
generate(new Fire(), 0.09*(1+gp.dificultRate));
generate(new AmmoPack(), rate:0.003);
generate(new FirstAid(), rate:0.003);
generate(new DecorativeObj(), rate:0.1);
```

0.3% per 1/30 seconds to generate Ammopack

10% per 1/30 seconds to generate Decoration

```java
public void generate(GameObject go, double rate) {
    boolean shouldGenerate = Math.random() < rate;
    if (shouldGenerate) {
        go.setPostion(DefaultPanel.screenWidth + 100, RD.random(-5, max:18) * DefaultPanel.tileSize);
        this.go.add(go);
    }
}
```

Check hitbox of player and attackbox of zombies (Check if two rectangle intersect)

```java
public boolean isPlayerHitByZombie(Player p) {
    int playerX = p.hitbox[0];
    int playerY = p.hitbox[1];
    int playerWidth = p.hitbox[2];
    int playerHeight = p.hitbox[3];

    int attackboxX = this.attackbox[0];
    int attackboxY = this.attackbox[1];
    int attackboxWidth = this.attackbox[2];
    int attackboxHeight = this.attackbox[3];

    int playerCenterX = playerX + playerWidth / 2;
    int playerCenterY = playerY + playerHeight / 2;

    int attackboxCenterX = attackboxX + attackboxWidth / 2;
    int attackboxCenterY = attackboxY + attackboxHeight / 2;

    int playerHalfWidth = playerWidth / 2;
    int playerHalfHeight = playerHeight / 2;
    int attackboxHalfWidth = attackboxWidth / 2;
    int attackboxHalfHeight = attackboxHeight / 2;

    boolean isPlayerHit = (Math.abs(playerCenterX - attackboxCenterX) < (playerHalfWidth + attackboxHalfWidth))
            && (Math.abs(playerCenterY - attackboxCenterY) < (playerHalfHeight + attackboxHalfHeight));

    return isPlayerHit;
}
```

```java
public boolean isKilled(Player p) {
    int shootboxX = p.shootbox[0];
    int shootboxY = p.shootbox[1];
    int shootboxWidth = p.shootbox[2];
    int shootboxHeight = p.shootbox[3];

    int attackboxX = p.attackbox[0];
    int attackboxY = p.attackbox[1];
    int attackboxWidth = p.attackbox[2];
    int attackboxHeight = p.attackbox[3];

    int zombieX = this.hitbox[0];
    int zombieY = this.hitbox[1];
    int zombieWidth = this.hitbox[2];
    int zombieHeight = this.hitbox[3];

    // Check if the shootbox and zombiehitbox intersect
    boolean isShot = (shootboxX + shootboxWidth >= zombieX && shootboxX <= zombieX + zombieWidth)
                  && (shootboxY + shootboxHeight >= zombieY && shootboxY <= zombieY + zombieHeight);
    boolean isAttacked = (attackboxX + attackboxWidth >= zombieX && attackboxX <= zombieX + zombieWidth)
                  && (attackboxY + attackboxHeight >= zombieY && attackboxY <= zombieY + zombieHeight);


    return (isShot && p.playerStateID == 4 && p.spriteNum < 6)
            || (isAttacked && p.playerStateID == 3);
}
```

Health bar will change slowly and smoothly

```
public void update() {
    if (this.displayhp > player.health)
        this.displayhp--;
    else if (this.displayhp < player.health)
        this.displayhp++;

}

public void draw(Graphics2D g2) {

    g2.drawImage(image, x:20, y:40, this.h+20, this.h+20, observer:null);
    g2.setColor(new Color(r:227, g:27, b:35));
    g2.fillRect(x:70, y:50, w * this.displayhp / player.maxHealth, this.h);
    g2.setColor(Color.BLACK);
    g2.setStroke(new BasicStroke(width:3));
    g2.drawRect(x:70, y:50, w, h);
    g2.setStroke(new BasicStroke(width:1));



}
```

When player step on ammo pack then increase player shotgun ammo by 8. And 48 is limit using Math.min()

```
if (isCollected) {
    p.shotgunBullet = Math.min(a:48, p.shotgunBullet+8);
    this.playerStepped = true;
}
```

Chapter 3: Summary

Problems encountered during development:

1. Game lag, difficult to fix.
2. The game didn't go as planned due to time constraints.

Highlights of the program:

1. Fun game, difficult to play

Suggestions:

The game should be simpler, with fewer requirements since it's just a mini-project.