Tetri
s

OBJECT - ORIENTED PROGRAMMING

Group Members

Nguyễn Phước Thịnh	ITITWE22106
Trịnh Minh Khoa	ITITIU21228
Trần Lê Minh Quân	ITITWE23042

Table of Content

Game Rules

Class design

Demo

Game Rules

A TEAM

ide

Game Rules

Items

a. Mino_Bar
b. Mino_L1
c.Mino_L2
d.Mino_Square
e. Mino_Z1
f. Mino_T
g. Mino_Z2

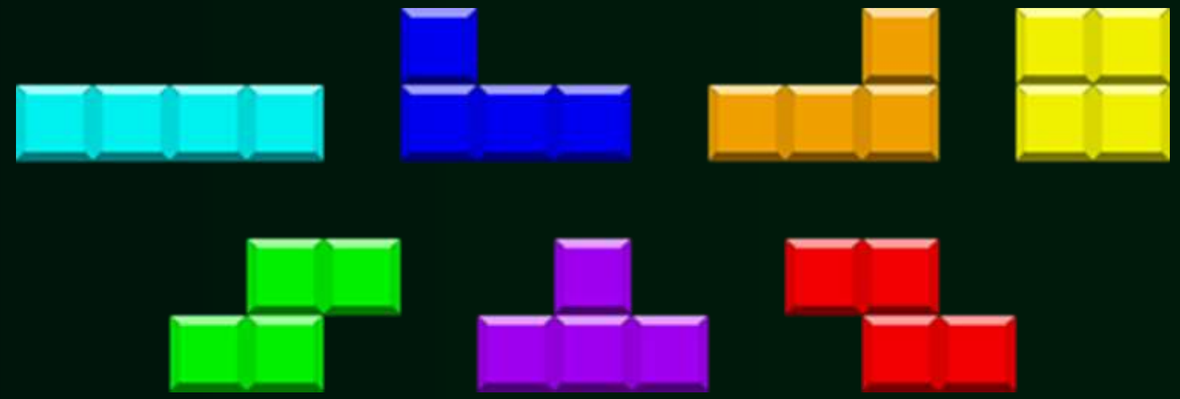


Figure 1 : The designed items

Game Rules

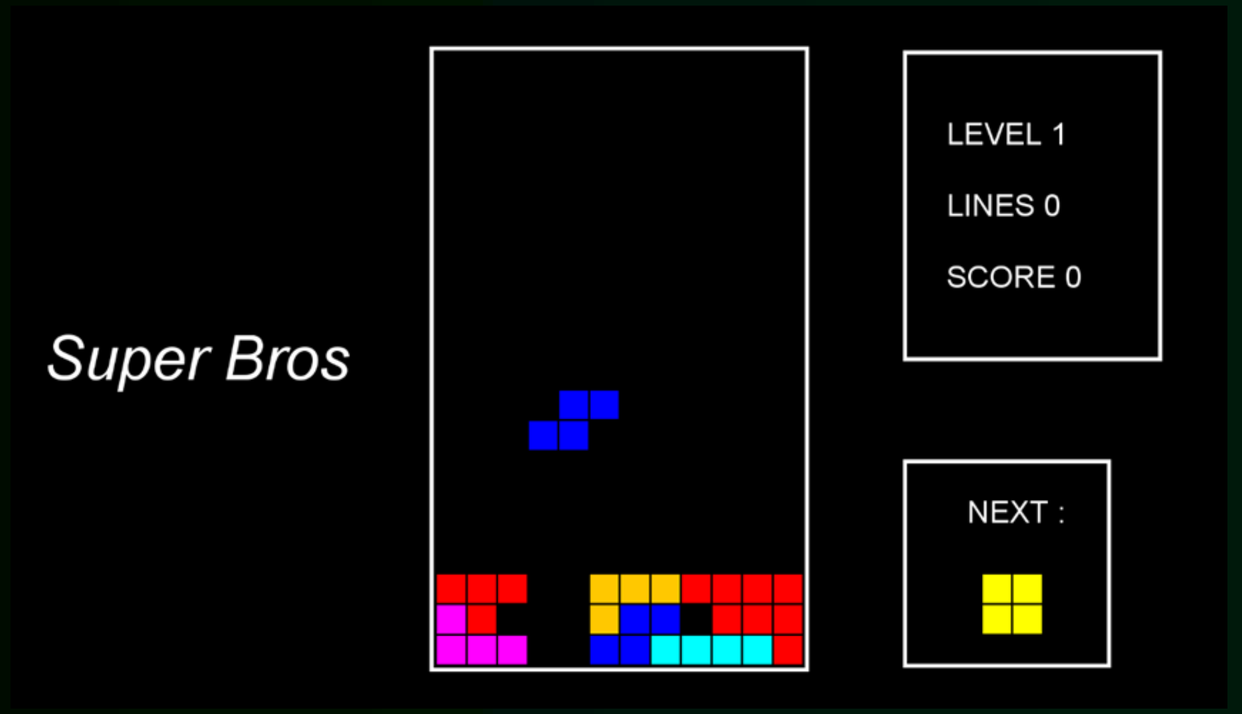


Figure 2.The Game Interface

Game Rules

Figure 3 : The Status Panel

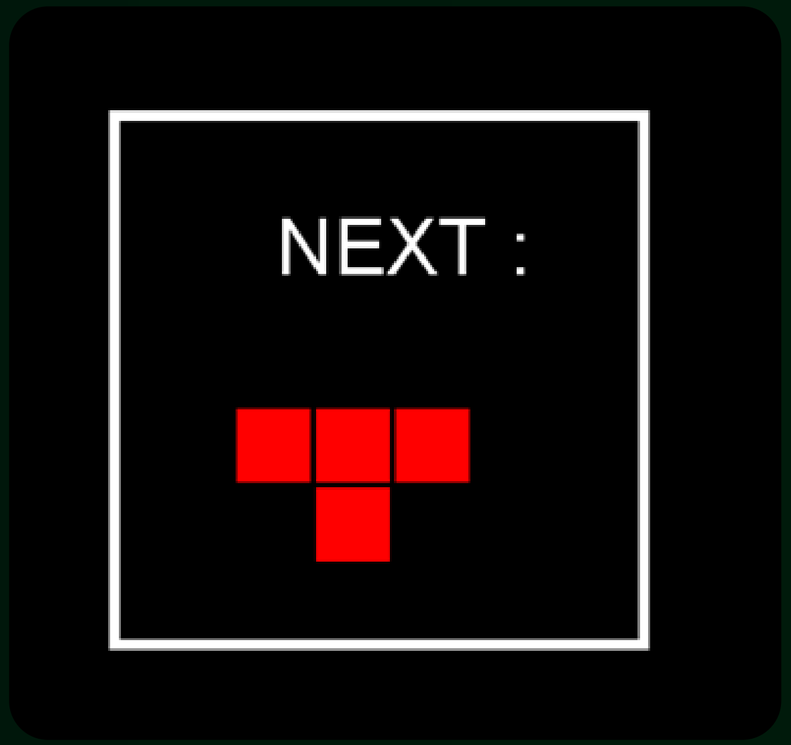


Figure 4 :The Data Frame(Next Brick Panel)

Game Rules

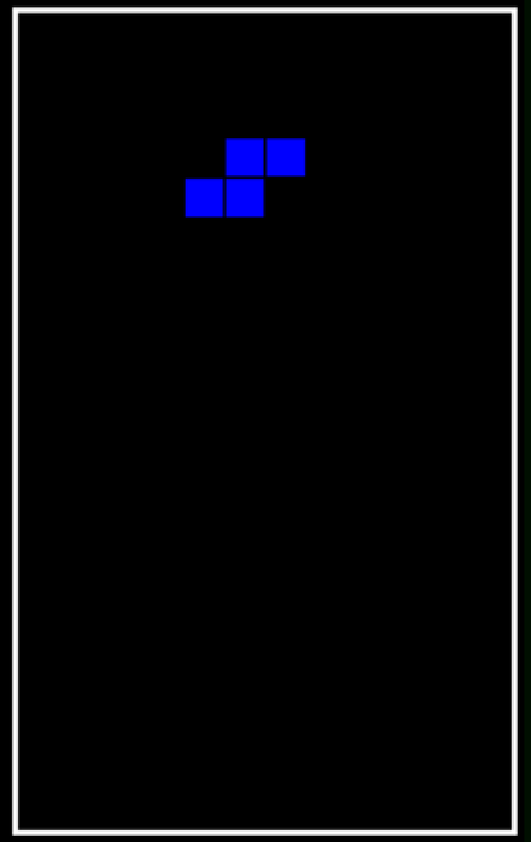


Figure 5 : GamePanel

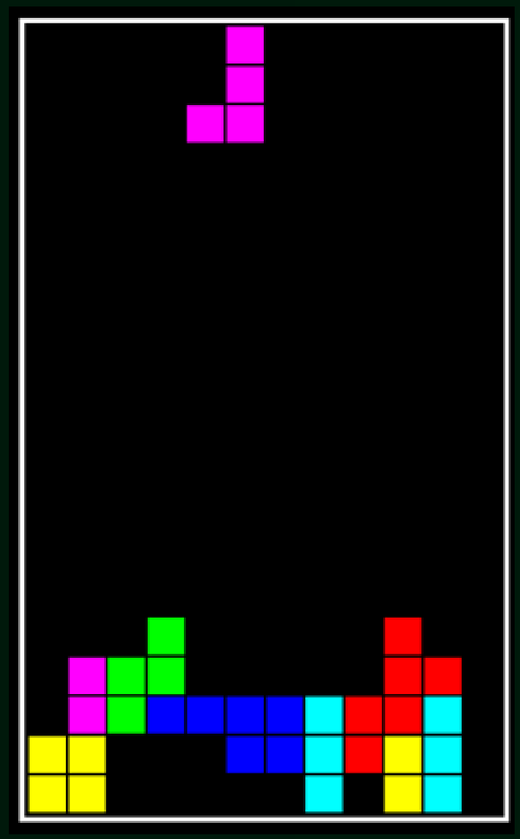


Figure 6: GamePanel

Game Rules

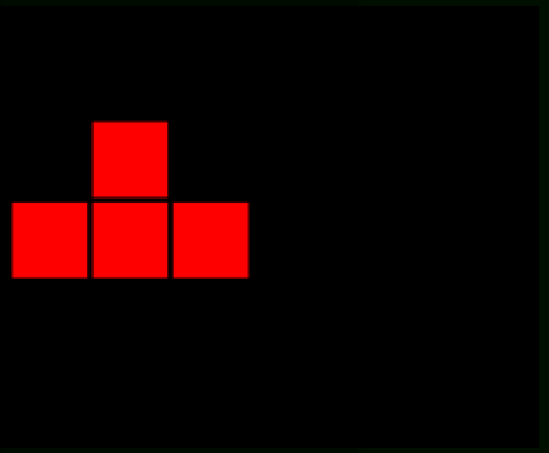


Figure 7: Mino before rotation

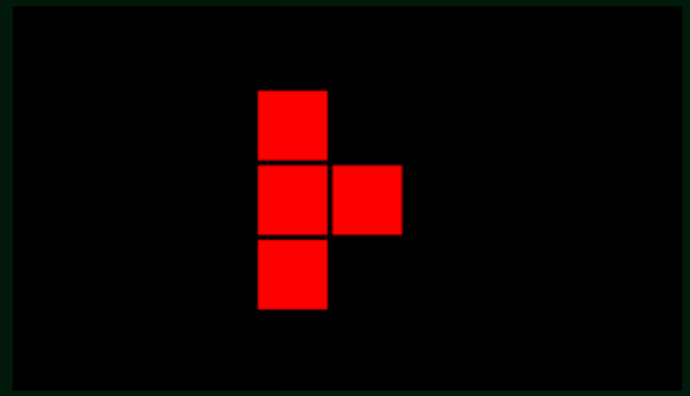


Figure 8: Mino after rotation

Game Rules

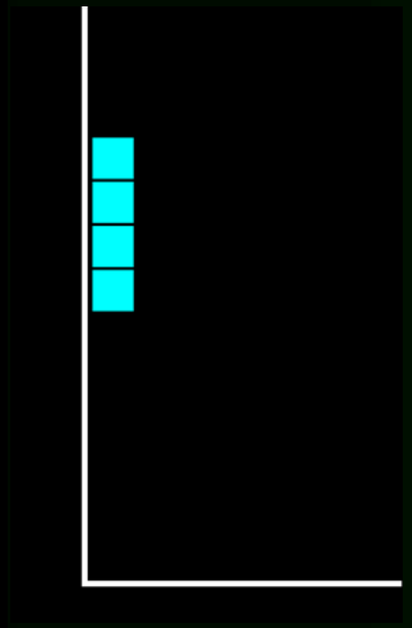


Figure 9: Mino_Bar cannot rotate when collide the wall

Figure 10: Mino cannot rotate when collide the floor

Game Rules

Game Pause

Player can take a break in the middle of the game by pressed the key to pop up the Pause Panel

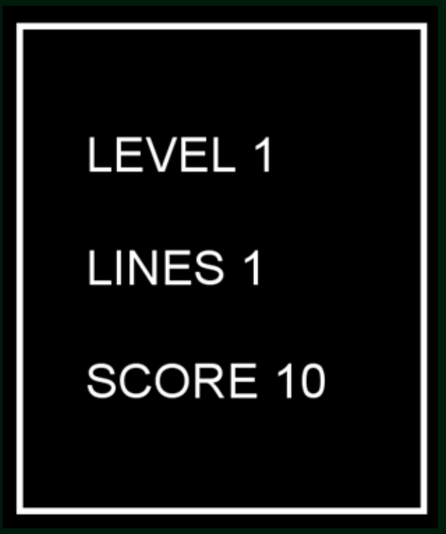
Figure 11 : PauseScreen

Game Rules

Figure 11 : Game Over screen

Score

After the perfect 12 Block in line disappear , you will receive for each 1 + line and the score will be summed for 10 points.



Game Over

When the brick cannot go down anymore the game will be over , then the Game over Panel(figure 7) will pop up and stop the game

Class Design

A TEAM

ide

Class Design

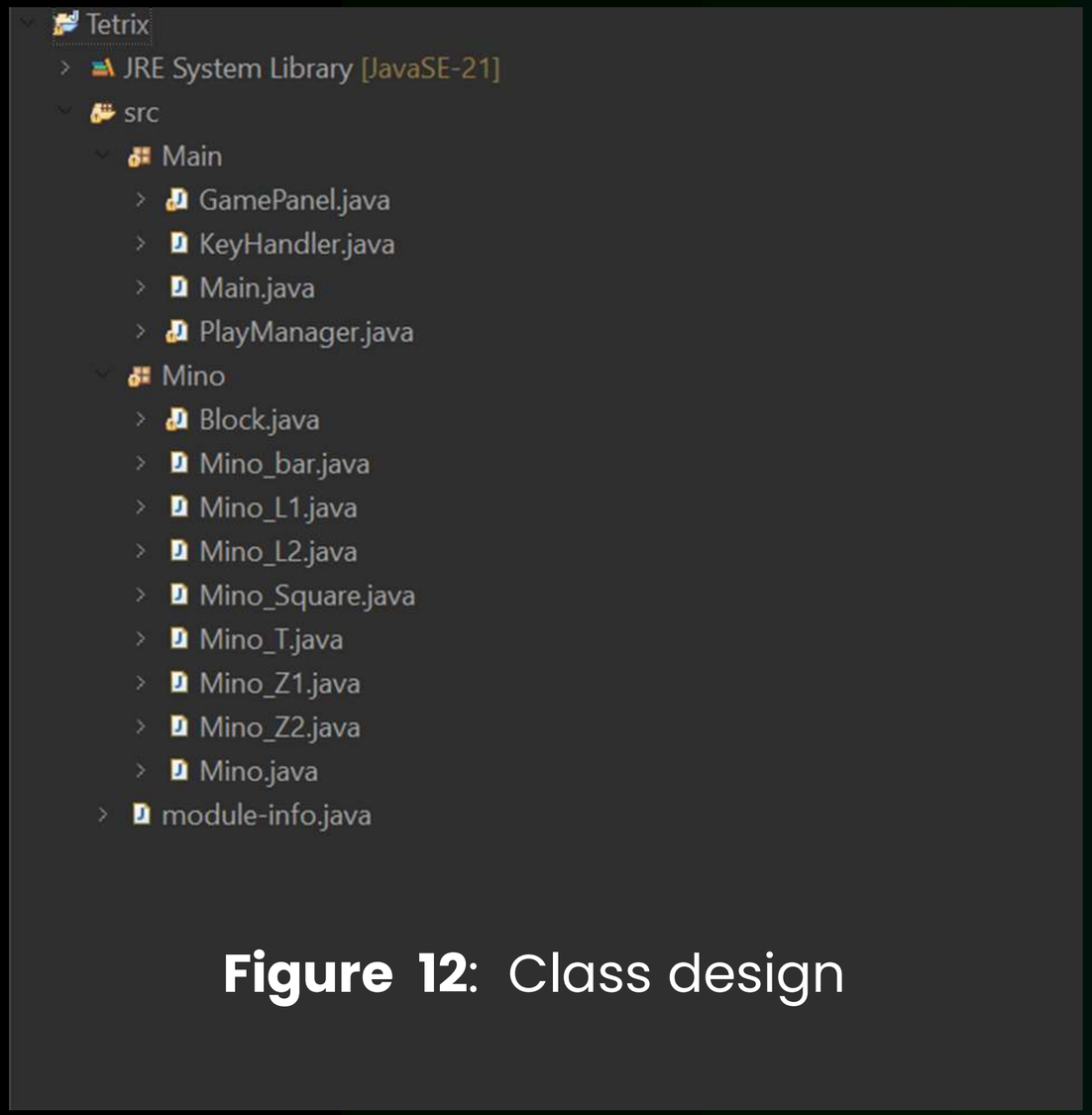


Figure 12: Class design

how the core game work that 2 class the **GamePanel** and **PlayManager**

Game Panel Class

```
1  package Main;
2
3  import java.awt.Color;
9
10 public class GamePanel extends JPanel implements Runnable {
11
12         public static final int WIDTH =1200;
13         public static final int HEIGHT= 720;
14         final int FPS=60;
15         Thread gameThread;
16         PlayManager pm;
17
18
19     public GamePanel () {
33 public void launchGame() {
34     gameThread=new Thread(this);
35     gameThread.start();
36 }
37
38
40     public void run() {
53         public void update() {
58
59         public void paintComponent(Graphics g) {
76
77     }
78
79
```

Figure 14: GamePanel class

Able game loop functionality. This class manages the graphical user interface (GUI), game loop, and interactions with other game components.

Mino Class

```
1  package Mino;
2
3  import java.awt.Color;
8
9  public class Mino {
10
11
12     public Block b[]=new Block[4];
13     public Block tempB[]= new Block[4];
14     int autodrop =0;
15     public int direction =1 ;
16     boolean leftCollision, rightCollision , bottomCollision;
17     public boolean active =true ;
18     public boolean deactivating;
19     int deactivatingCounter =0;
20
21
22
23     public void create(Color c ) {
34     public void setXY(int x , int y ) {}
35     public void updateXY(int direction) {
53
54     public void getDirection1() {}
55     public void getDirection2() {}
56     public void getDirection3() {}
57     public void getDirection4() {}
58     public void checkMovementCollision() {
88     public void checkRotationCollision() {
116
117    public void checkStaticBlockCollision() {
145    public void update(){
220    private void deactivating() {
236    public void draw(Graphics g2) {
237
238
239        int margin=1;
240        g2.setColor(b[0].c);
241        g2.fillRect(b[0].x+ margin, b[0].y+margin, Block.SIZE-(margin*2), Block.SIZE-(margin*2));
242        g2.fillRect(b[1].x+ margin, b[1].y+margin, Block.SIZE-(margin*2), Block.SIZE-(margin*2));
243        g2.fillRect(b[2].x+ margin, b[2].y+margin, Block.SIZE-(margin*2), Block.SIZE-(margin*2));
244        g2.fillRect(b[3].x+ margin, b[3].y+margin, Block.SIZE-(margin*2), Block.SIZE-(margin*2));
```

Figure 14: GamePanel class

oBlocks:

- o Movement and State:
- o Collision Flags:

PlayManager Class

```
public class PlayManager {

    //main playarea

    final int WIDTH = 360;
    final int HEIGHT = 600;
    public static int left_x;
    public static int right_x;
    public static int top_y;
    public static int bottom_y;


    //Mino
    Mino currentMino;
    final int MINO_START_X;
    final int MINO_START_Y;
    Mino nextMino;
    final int NEXTMINO_X;
    final int NEXTMINO_Y;
    public static ArrayList<Block> staticBlocks =new ArrayList<>();

    //other
    public static int dropInterval =60;// the speed of the drop is 60 frames
    boolean gameOver;

    // effect
    boolean eco;
    int ec;
    ArrayList<Integer> effectY = new ArrayList<>();

    //score
    int level =1;
    int lines;
    int score;

    public PlayManager() {
    private Mino pickMino() {
```

Figure 15: PlayManager class

The PlayManager class serves as the primary handler for gameplay mechanics, including managing the play area, controlling Tetris pieces (Mino), and updating the game state. It is responsible for drawing the game area, managing static blocks, handling gameplay actions, and providing visual effects and scoring.

Block Class

```java
package Mino;

import java.awt.Color;

public class Block extends Rectangle  {


    public int x,y;
    public static final int SIZE =30;//30x30 block
    public Color c ;

    public Block(Color c) {
        this.c=c;
    }
    public void draw (Graphics2D g2 ) {
        int margine =1;
        g2.setColor(c);
        g2.fillRect(x+margine, y+margine, SIZE-(2*margine), SIZE-(2*margine));
    }
    }
```

Figure 16: Block class

Which extends Rectangle, represents an individual square unit that makes up a Tetris piece (Mino) or forms part of the static blocks on the playfield. It is a fundamental building block of the game.

```java
package Mino;

import java.awt.Color;

public class Mino_L1 extends Mino{

    public Mino_L1() {


    public void setXY(int x, int y ) {
    public void getDirection1() {
    public void getDirection2() {
        //
        // 0 0 0
        // 0

                tempB[0].x=b[0].x;
                tempB[0].y=b[0].y;
                tempB[1].x= b[0].x+ Block.SIZE;
                tempB[1].y= b[0].y ;
                tempB[2].x= b[0].x- Block.SIZE;
                tempB[2].y= b[0].y ;
                tempB[3].x= b[0].x - Block.SIZE;
                tempB[3].y= b[0].y + Block.SIZE;

                updateXY(2);
    }
public void getDirection3() {
    public void getDirection4() {
```

Figure 17: Mino_L1 class

All Mino Class

Which were built the same and it need to take 1 block out to be the static block and all other 3 Blocks around it will be made base on the data of that static Block

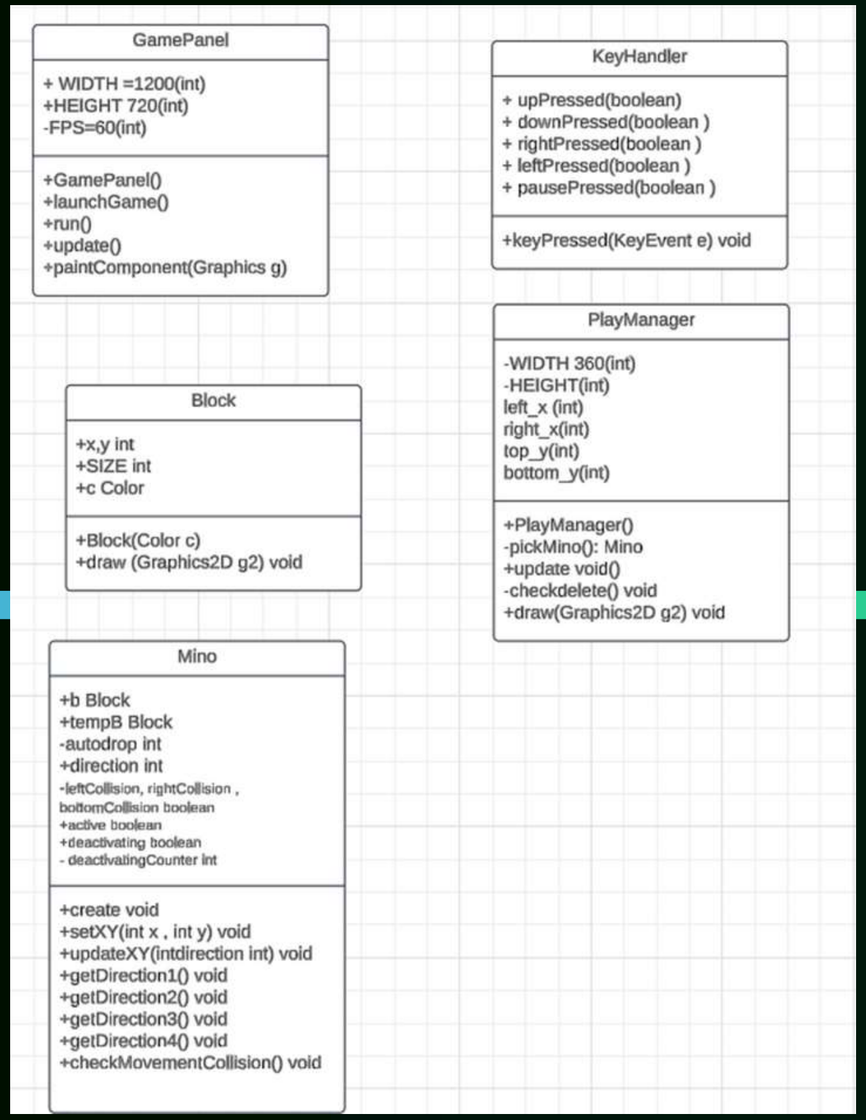


Figure 18: Project UML

Dem

A TEAM

ide

Thank You

FOR YOUR ATTENTION