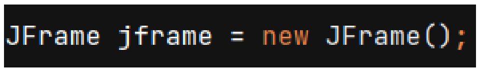
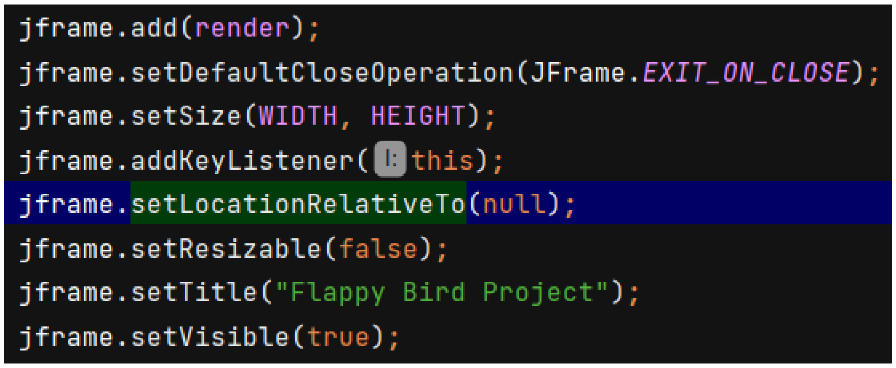
Window/Graphics:

Start creating the window by making an object called "jframe" using the JFrame class. The JFrame class provides close/title and other standard window features as you can see in the code.
code:

```
JFrame jframe = new JFrame();

jframe.add(render);

jframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

jframe.setSize(WIDTH, HEIGHT);

jframe.addKeyListener(this);

jframe.setLocationRelativeTo(null);

jframe.setResizable(false);

jframe.setTitle("Flappy Bird Project");

jframe.setVisible(true);
```



```
JFrame jframe = new JFrame();
```



```
jframe.add(render);
jframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
jframe.setSize(WIDTH, HEIGHT);
jframe.addKeyListener(l: this);
jframe.setLocationRelativeTo(null);
jframe.setResizable(false);
jframe.setTitle("Flappy Bird Project");
jframe.setVisible(true);
```

The JFrame is responsible for managing the layout and appearance of the other components (e.g. buttons, labels, text field, and panels), as well as responding to user inputs such as keystrokes and clicks.

Next, create a JPanel object to draw graphics on the screen. The Render class extends JPanel and overrides the paintComponent method of the JPanel class to draw the game elements onto the screen.

code:

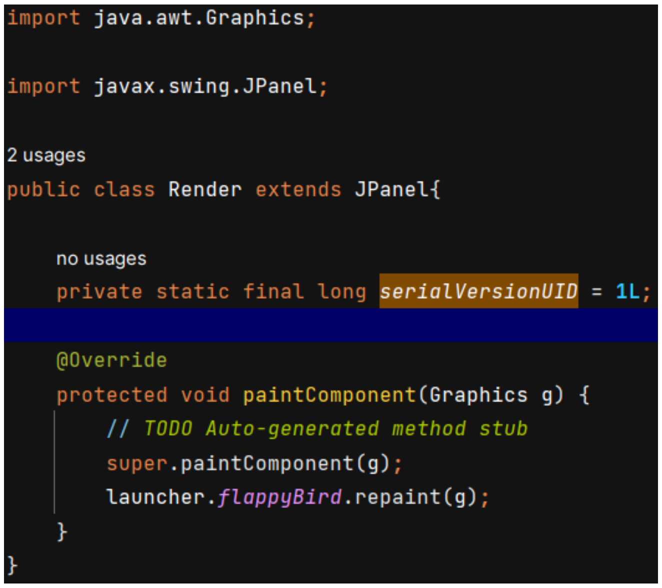
```
render = new Render();
```

```
render = new Render();
```

The paintComponent method takes the Graphics object as a parameter, which is used to draw graphics onto the screen. The super.paintComponent(g) call is important because it ensures that the panel is properly cleared and prepared for painting. The Graphics object g is passed in as an argument, and you can use it to draw shapes and images on the panel.

code:

```
import java.awt.Graphics;


import javax.swing.JPanel;


public class Render extends JPanel{


	private static final long serialVersionUID = 1L;


	@Override

	protected void paintComponent(Graphics g) {

		// TODO Auto-generated method stub

		super.paintComponent(g);

		launcher.flappyBird.repaint(g);

	}

}
```



```
import java.awt.Graphics;

import javax.swing.JPanel;

2 usages
public class Render extends JPanel{

    no usages
    private static final long serialVersionUID = 1L;

    @Override
    protected void paintComponent(Graphics g) {
        // TODO Auto-generated method stub
        super.paintComponent(g);
        launcher.flappyBird.repaint(g);
    }
}
```

This paints the bird, pipes, and text on the screen. The render object is added to the jframe using the add method.

code:

jframe.add(render);

```java
jframe.add(render);
```

The Repaint method is called in the actionPerformed method to repaint the render object and update the game screen according to the changes made during the game loop. In other words, render.repaint() will refresh the appearance of the Render component on the screen. This is useful when you want to update the content of the component or change its appearance in response to user input or other events.

Overall, the Render class provides the visual presentation of the game by drawing the bird and the pipes on the screen using the Graphics class methods such as fillRect, setColor, setFont, etc. It is responsible for handling the GUI aspects of the game and presenting the game state to the user in a graphical format.

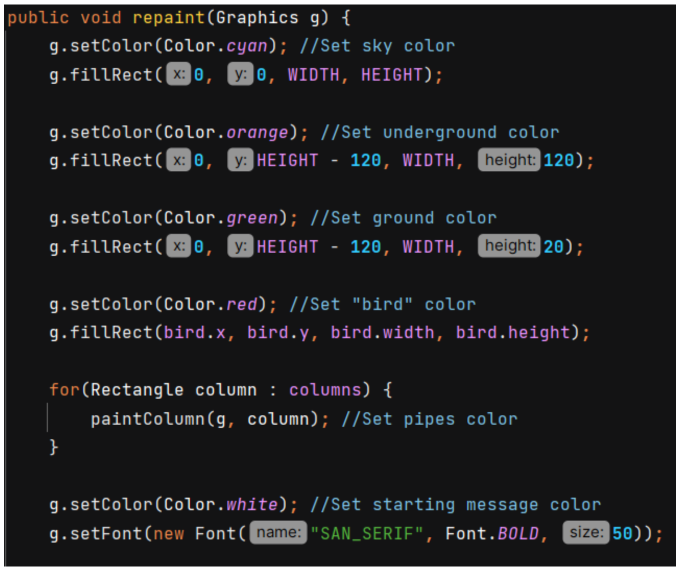
code:

render.repaint();

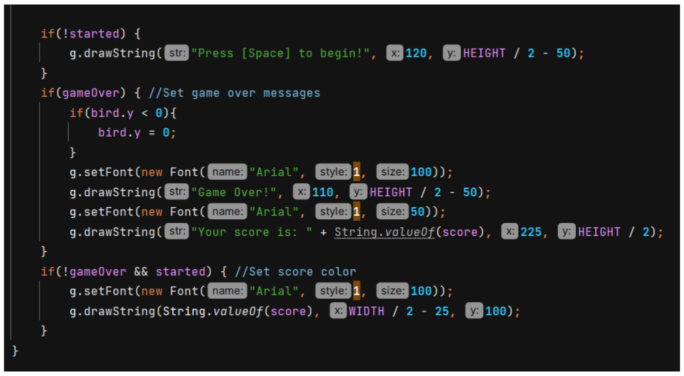
```java
render.repaint();
```

code:

```java
public void repaint(Graphics g) {

        g.setColor(Color.cyan); //Set sky color

        g.fillRect(0, 0, WIDTH, HEIGHT);


        g.setColor(Color.orange); //Set underground color

        g.fillRect(0, HEIGHT - 120, WIDTH, 120);


        g.setColor(Color.green); //Set ground color

        g.fillRect(0, HEIGHT - 120, WIDTH, 20);


        g.setColor(Color.red); //Set "bird" color

        g.fillRect(bird.x, bird.y, bird.width, bird.height);

        for(Rectangle column : columns) {
            paintColumn(g, column); //Set pipes color
        }

        g.setColor(Color.white); //Set starting message color
        g.setFont(new Font("SAN_SERIF", Font.BOLD, 50));

        if(!started) {
            g.drawString("Press [Space] to begin!", 120, HEIGHT / 2 - 50);
        }
        if(gameOver) { //Set game over messages
            if(bird.y < 0){
                bird.y = 0;
            }
            g.setFont(new Font("Arial", 1, 100));
            g.drawString("Game Over!", 110, HEIGHT / 2 - 50);
            g.setFont(new Font("Arial", 1, 50));
            g.drawString("Your score is: " + String.valueOf(score), 225, HEIGHT / 2);
        }
        if(!gameOver && started) { //Set score color
            g.setFont(new Font("Arial", 1, 100));
            g.drawString(String.valueOf(score), WIDTH / 2 - 25, 100);
        }
    }
```



```java
public void repaint(Graphics g) {
    g.setColor(Color.cyan); //Set sky color
    g.fillRect( x: 0,  y: 0, WIDTH, HEIGHT);

    g.setColor(Color.orange); //Set underground color
    g.fillRect( x: 0,  y: HEIGHT - 120, WIDTH,  height: 120);

    g.setColor(Color.green); //Set ground color
    g.fillRect( x: 0,  y: HEIGHT - 120, WIDTH,  height: 20);

    g.setColor(Color.red); //Set "bird" color
    g.fillRect(bird.x, bird.y, bird.width, bird.height);

    for(Rectangle column : columns) {
        paintColumn(g, column); //Set pipes color
    }

    g.setColor(Color.white); //Set starting message color
    g.setFont(new Font( name: "SAN_SERIF", Font.BOLD,  size: 50));
```



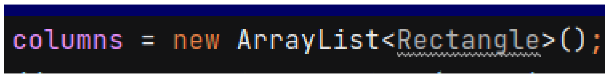
```java
    if(!started) {
        g.drawString( str: "Press [Space] to begin!",  x: 120,  y: HEIGHT / 2 - 50);
    }
    if(gameOver) { //Set game over messages
        if(bird.y < 0){
            bird.y = 0;
        }
        g.setFont(new Font( name: "Arial",  style: 1,  size: 100));
        g.drawString( str: "Game Over!",  x: 110,  y: HEIGHT / 2 - 50);
        g.setFont(new Font( name: "Arial",  style: 1,  size: 50));
        g.drawString( str: "Your score is: " + String.valueOf(score),  x: 225,  y: HEIGHT / 2);
    }
    if(!gameOver && started) { //Set score color
        g.setFont(new Font( name: "Arial",  style: 1,  size: 100));
        g.drawString(String.valueOf(score),  x: WIDTH / 2 - 25,  y: 100);
    }
}
```

Pipes:

First we add a arrayList called columns and make an object columns In the FlappyBird constructor:

code:

```java
columns = new ArrayList<Rectangle>();
```

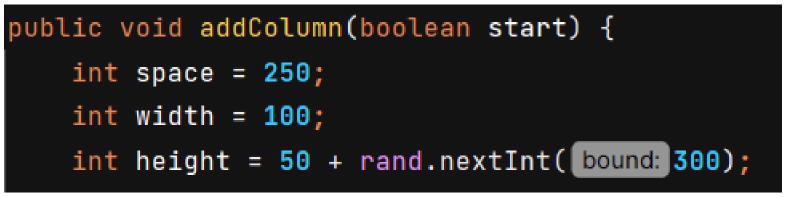


```java
columns = new ArrayList<Rectangle>();
```

Setting up the Pipes: The space is the Opening, And the height randomize the size from 50 to 300.

code:

```java
public void addColumn(Boolean start){

    int space = 250;

    int width = 100;
    int Height = 50 + rand.nextInt(300);
```



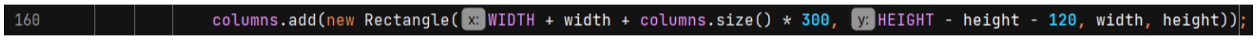
```java
public void addColumn(boolean start) {
    int space = 250;
    int width = 100;
    int height = 50 + rand.nextInt( bound: 300);
```

WIDTH So the pipe starts at far right of the screen + width so it starts outside of the screen + columns.size() * 300 so if there is another pipe it moves it over, by 300px if there is only 1, by (300+300=600px) if there are 2 etc. HEIGHT So the pipe starts at the bottom of the screen - height so it starts at a random height - 120 So it 100% starts at the top of the grass

width so the pipe has a width of 100px height so the pipe starts at the (HEIGHT - height - 120) and it has a random height going down till it reaches (HEIGHT - 120) => the grass

code:

columns.add(new Rectangle(WIDTH + width + columns.size() * 300, Height - height - 120, wish, height));

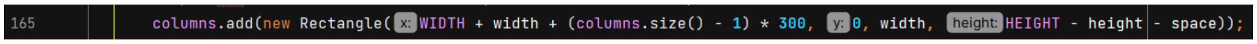


```java
        columns.add(new Rectangle( x: WIDTH + width + columns.size() * 300,  y: HEIGHT - height - 120, width, height));
```

WIDTH So the pipe starts at far right of the screen + width so it starts outside of the screen + columns.size() - 1 so it doesn't get moved over as we wanna have 2 pipes at the same place one on top one on bottom of the screen * 300 so if there is another pipe then it moves it over, by 300px if there is only 1, by (300+300=600px) if there are 2 etc. 0 So the pipe starts at the top of the screen width so the pipe has a width of 100px HEIGHT - height – space HEIGHT So the pipe ends at the bottom of the screen - height so it ends at a random height which is at least HEIGHT(800)-height(at least 50)=750px - space so it ends at least at 750-space(300)=at least 450px

code:

columns.add(new Rectangle(Width + width + (columns,size() - 1) * 300, 0, width, HEIGHT - height - space));



```java
        columns.add(new Rectangle( x: WIDTH + width + (columns.size() - 1) * 300,  y: 0, width,  height: HEIGHT - height - space));
```

But we want this columns to add in the beginning so we add four pairs of pipes are added to the columns array using the addColumn method with the start parameter set to true:

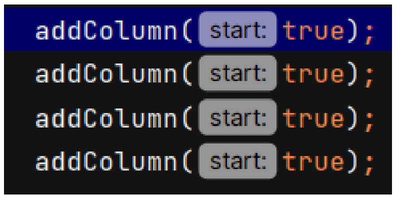
code:

addColumns(true);

addColumns(true);

addColumns(true);

addColumns(true);



```
addColumn( start: true);
addColumn( start: true);
addColumn( start: true);
addColumn( start: true);
```

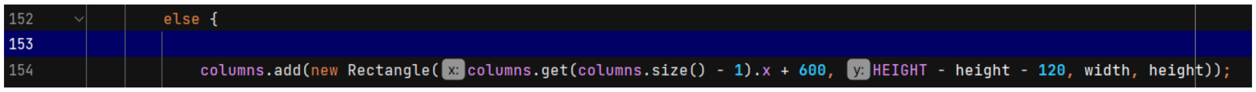
if it isn't the starting pipe then:

columns.get(columns.size() - 1).x columns.size() So we get the last pipe -1 cuz if there is only 1 pipe at the list then the columns.size() will return 1 but we want to get the pipe with the position of 0 + 600 so it goes at 600px more at right

code:

 else{

  columns.add(new Rectangle(columns.get(columns.size() - 1).x + 600, HEIGHT - height - 120, width, height));

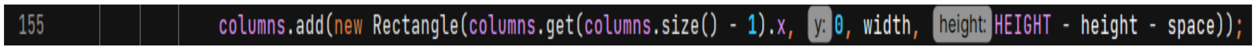


```
else {

    columns.add(new Rectangle( x: columns.get(columns.size() - 1).x + 600,  y: HEIGHT - height - 120, width, height));
```

columns.get(columns.size() - 1).x we dont need the +600 now cuz we will get the pipe that we already changed at the text above

code:

  columns.add(new Rectangle(columns.get(columns.size() - 1).x, 0, width, HEIGHT - height - space));



```
    columns.add(new Rectangle(columns.get(columns.size() - 1).x,  y: 0, width,  height: HEIGHT - height - space));
```

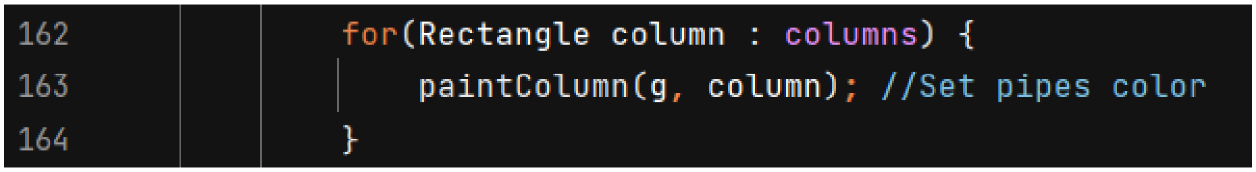
then we add paint for the rectangles or pipes:

code:

 for(Rectangle column : columns){

   paintColumn(g, column);

}

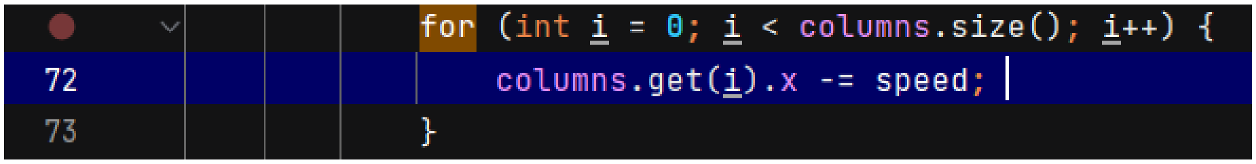


```
for(Rectangle column : columns) {
    paintColumn(g, column); //Set pipes color
}
```

Next we want the moved the column or pipes, I created for loop here inside created int i is equal to 0, i is less than columns.size, so if the column size is 1 this only call once, because we add one (i++) everytime

code:

```
  for(int i = 0; i < columns.size(); i++){
    columns.get(i).x -= speed;
}
```



```
for (int i = 0; i < columns.size(); i++) {
    columns.get(i).x -= speed;
}
```

Then we created integer named speed with a value 10, to set the speed of the movement of the columns(sprite). after that we test it did well but it stop appearing.

code:

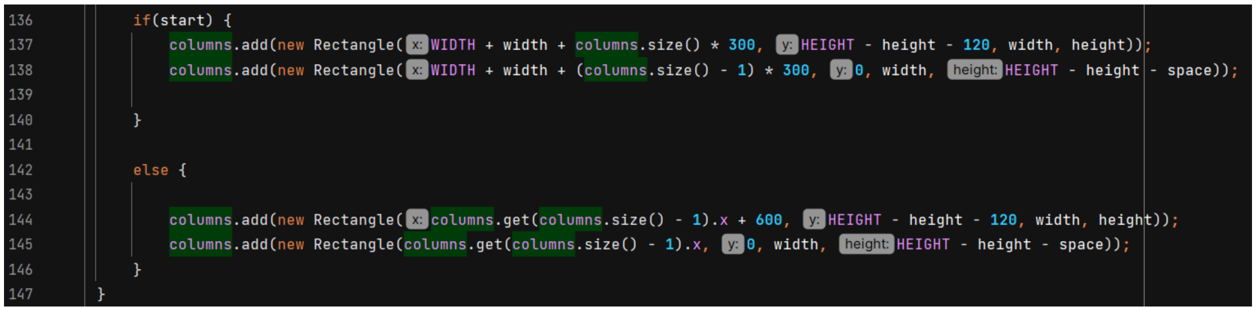
```
 int speed = 10;
```

```
int speed = 10;
```

thats why we have this if else here to make them appear.

code:

```
  if(start){
      columns.add(new Rectangle(WIDTH + width + columns.size() * 300, HEIGHT - height - 120, width, height));
     columns.add(new Rectangle(WIDTH + width + (columns.size() - 1) * 300, 0, width, HEIGHT - height - space));
}else{
  columns.add(new Rectangle(WIDTH + width + (columns.size() - 1) * 300, 0, width, HEIGHT - height - 120, width, height));
       columns.add(new Rectangle(WIDTH + width + (columns.size() - 1).x, 0, width, HEIGHT - height - space));
}
}
```

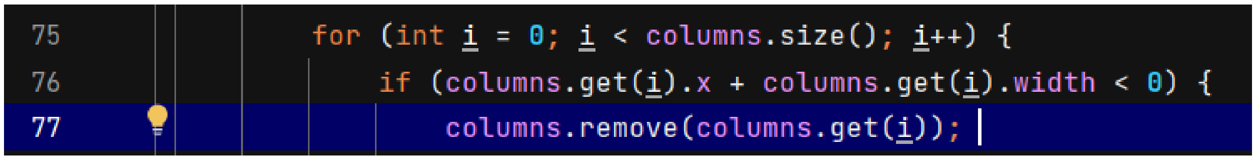


```
        if(start) {
            columns.add(new Rectangle( x: WIDTH + width + columns.size() * 300,  y: HEIGHT - height - 120, width, height));
            columns.add(new Rectangle( x: WIDTH + width + (columns.size() - 1) * 300,  y: 0, width,  height: HEIGHT - height - space));

        }

        else {

            columns.add(new Rectangle( x: columns.get(columns.size() - 1).x + 600,  y: HEIGHT - height - 120, width, height));
            columns.add(new Rectangle(columns.get(columns.size() - 1).x,  y: 0, width,  height: HEIGHT - height - space));
        }
    }
```

then created If because we want to remove the pipe to remove after it left on the screen.

code:

```
  for(int i = 0; i < columns.size(); i++){

    if(columns.get(i).x + columns.get(i).width < 0){

      columns.remove(columns.get(i));

}

}
```

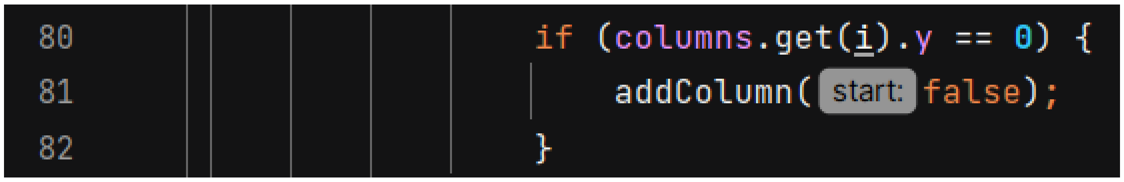


```
for (int i = 0; i < columns.size(); i++) {
    if (columns.get(i).x + columns.get(i).width < 0) {
        columns.remove(columns.get(i));
```

add another if, because this is saying its the top column so we dont want to do it twice because technically we have two columns going through and both of them arent going to be removed so pretty much we can do this by saying if its the top column so we know the y is always 0 for that one.so if its the top column then add another column and this will be infinite.

code:

```
  if(columns.get(i).y == 0){

      addColumn(false);

}
```



```
if (columns.get(i).y == 0) {
    addColumn( start: false);
}
```

The Gravity Function of the Bird or the Bird Itself:

First we created 2 integers the ticks and yMotion of the bird:

code:

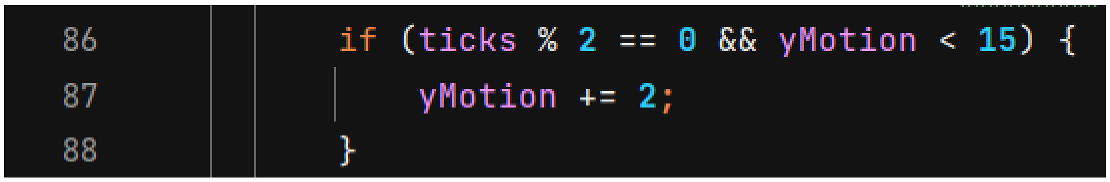
public int ticks, yMotion;

```
public int ticks, yMotion,
```

If remainder of ticks is equal to 0 and yMotion is less than 15 then it will call inside the curly brackets which is the yMotion plus equal 2

code:

```
 if(ticks % 2 == 0 && yMotion < 15){

      yMotion += 2;

}
```



```
if (ticks % 2 == 0 && yMotion < 15) {
    yMotion += 2;
}
```

then we add ticks++ to add:

code:

```
ticks++
```

```
ticks++;
```

next we add the bird.y += yMotion to make the bird to fall down or gravity:

code:

```
bird.y += yMotion;
```

```
bird.y += yMotion;
```

The Coordinate of the Bird:
20 , 20 Is the width/height of the bird, Width / 2 and Height / 2 are the coordinates of where it will start And then we do - 10 at the coordinates because the (Width / 2 & Height / 2) will put the top left corner of the bird in the middle of the screen And we already know that size of the bird is 20x20 so to put it exactly in the middle of the screen we do 20 / 2 = 10 So we put - 10 in the coordinates of the bird

code:

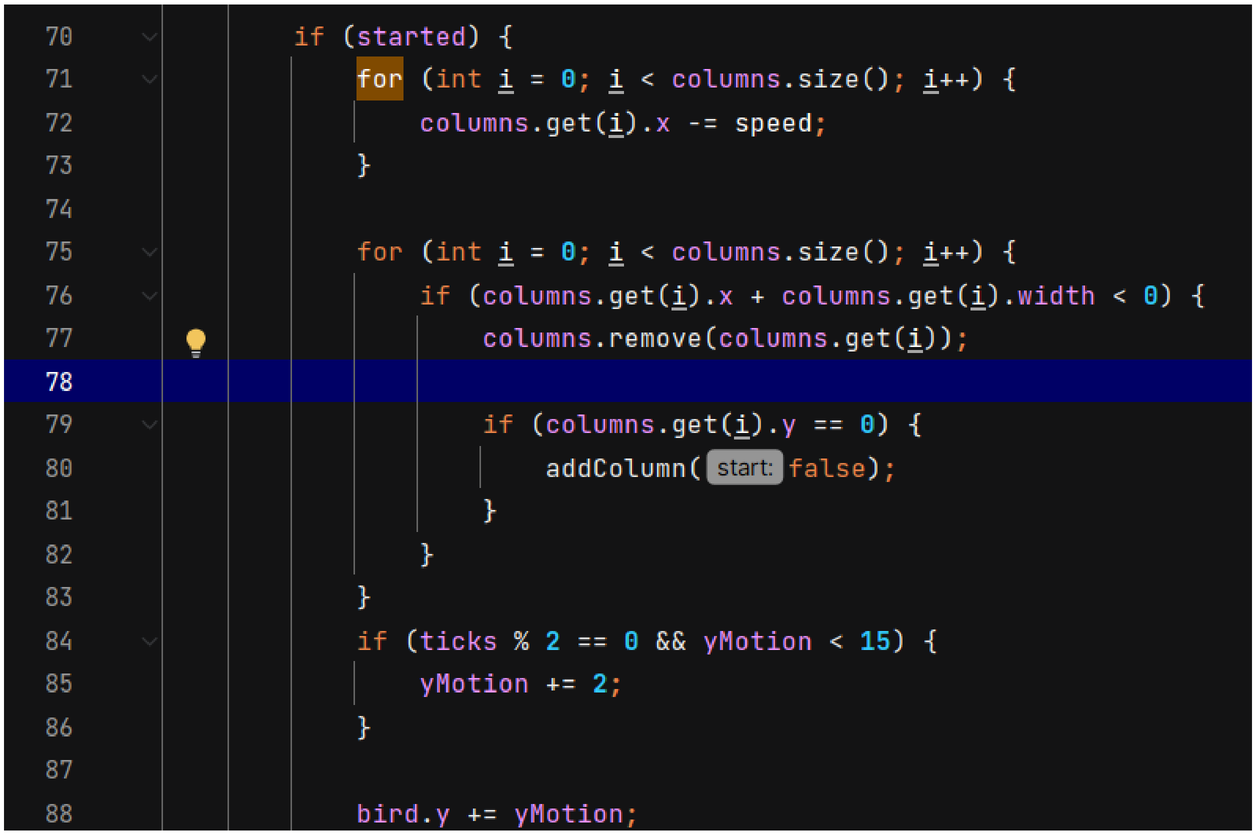
```
bird = new Rectangle(WIDTH / 2 - 20, HEIGHT / 2 - 20, 20, 20);
```

```
bird = new Rectangle( x: WIDTH / 2 - 20,  y: HEIGHT / 2 - 10,  width: 20,  height: 20);
```

thats all about how the Pipe generation work if you started the game the program will call this started.(The Pipe and The bird)

code:

```
public void actionPerformed(ActionEvent arg0) {
        ticks++;
        int speed = 10;
        if (started) {
                for (int i = 0; i < columns.size(); i++) {
                        columns.get(i).x -= speed;
                }
                for (int i = 0; i < columns.size(); i++) {
                        if (columns.get(i).x + columns.get(i).width < 0) {
                                columns.remove(columns.get(i));
                                if (columns.get(i).y == 0) {
                                        addColumn(false);
                                }
                        }
                }
                if (ticks % 2 == 0 && yMotion < 15) {
                        yMotion += 2;
        }
```



```
if (started) {
    for (int i = 0; i < columns.size(); i++) {
        columns.get(i).x -= speed;
    }

    for (int i = 0; i < columns.size(); i++) {
        if (columns.get(i).x + columns.get(i).width < 0) {
            columns.remove(columns.get(i));

            if (columns.get(i).y == 0) {
                addColumn( start: false);
            }
        }
    }
    if (ticks % 2 == 0 && yMotion < 15) {
        yMotion += 2;
    }

    bird.y += yMotion;
```

Collision (to check if its Scores or Gameover) Function:

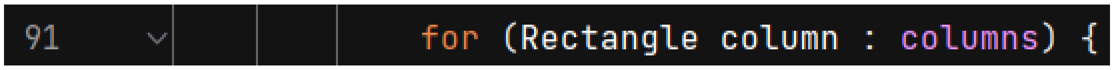
So to check the collision first thing we did we do another column thing. For rectangle column, columns, So for each column inside columns do if column intersects bird or bird intersect columns then we want to do game over equals true.

code:

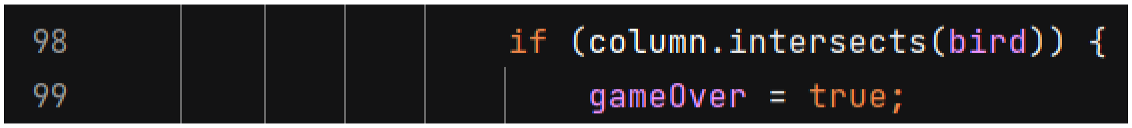
for (Rectangle column : columns){

if(column.intersects(bird)){

gameOver = true;



```
for (Rectangle column : columns) {
```

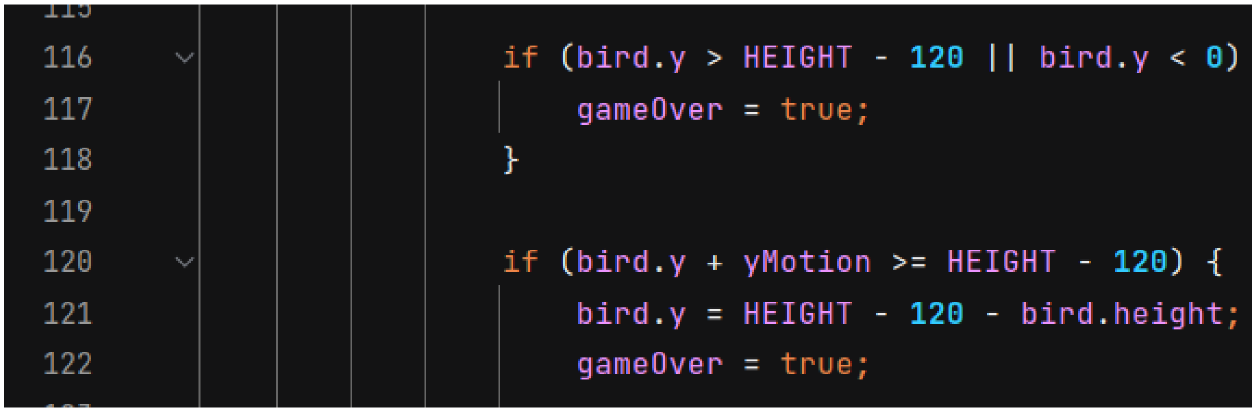


```
    if (column.intersects(bird)) {
        gameOver = true;
```

then created another game over if the bird Y is greater than the capital height which is 800 - 120 or bird Y is less than 0 is equal to game over, in other words if the bird leaves the game ground then add this if bird Y equal to capital height minus 120 minus bird.height to make the stop at the ground which is the grass

code:

```
if (bird.y > HEIGHT - 120 || bird.y < 0) {// If the bird leaves the game ground then game over
                    gameOver = true;
          }
          if (bird.y + yMotion >= HEIGHT - 120) {
                    bird.y = HEIGHT - 120 - bird.height; //move the bird on top of the ground
                    gameOver = true;
          }
}
```



```
if (bird.y > HEIGHT - 120 || bird.y < 0)
    gameOver = true;
}

if (bird.y + yMotion >= HEIGHT - 120) {
    bird.y = HEIGHT - 120 - bird.height;
    gameOver = true;
```

then we add this the bird x equal to column x minus bird width because we want the bird stop when they hit the pipe

code:

```
bird.x = column.x - bird.width;
```

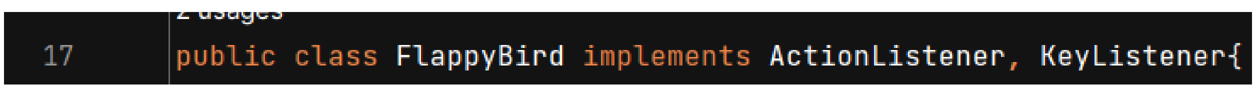
```
bird.x = column.x - bird.width;
```

Movement(Action):

So we want to add command key first thing we did is we implement KeyListener

code:

public class FlappyBird implements ActionListener, KeyListener{



```
2 usages
public class FlappyBird implements ActionListener, KeyListener{
```

setting up the command key space bar

code:

@override

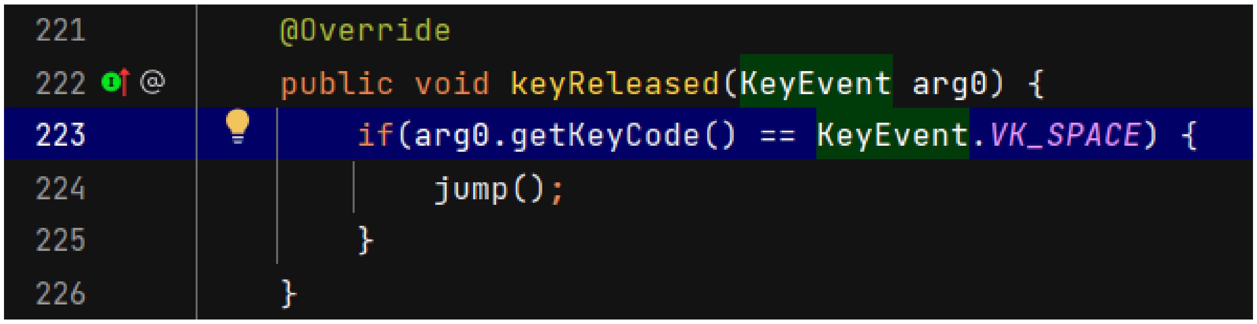
public void keyReleased(KeyEvent arg0){

if(arg0.getKeyCode() == KeyEvent.VK_SPACE){

jump();

}

}



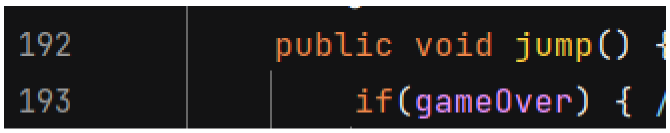
```
    @Override
    public void keyReleased(KeyEvent arg0) {
        if(arg0.getKeyCode() == KeyEvent.VK_SPACE) {
            jump();
        }
    }
```

then we created a constructor Jump() and created to if's, if the game started and GameOver

code:

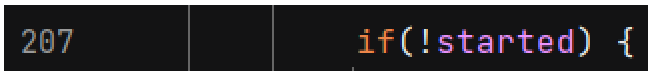
public void jump(){

if(gameOver){



```
    public void jump() {
        if(gameOver) {
```

if(!started){

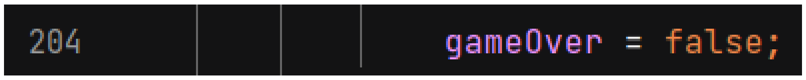


```
        if(!started) {
```

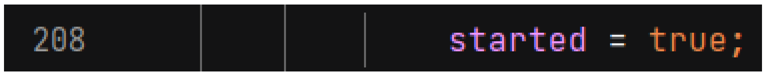
then created inside the curly brackets gameOver equal to false, and started equals to true this mean to start the game.

public void jump() {

```
        if(gameOver) {
        gameOver = false;
}
if(!started) {
        started = true;
}
```



```
gameOver = false;
```



```
started = true;
```

then copy paste the code to the if gameover:

code:

```
bird = new Rectangle(WIDTH / 2 - 20, HEIGHT / 2 - 10, 20, 20);
columns = new ArrayList<Rectangle>();
addColumn(true);
addColumn(true);
addColumn(true);
addColumn(true);
```

to this

code:

```
if(gameOver) {
        bird = new Rectangle(WIDTH / 2 - 10, HEIGHT / 2 - 10, 20, 20);
        columns.clear();
        yMotion = 0;
        score = 0;

        addColumn(true);
        addColumn(true);
```

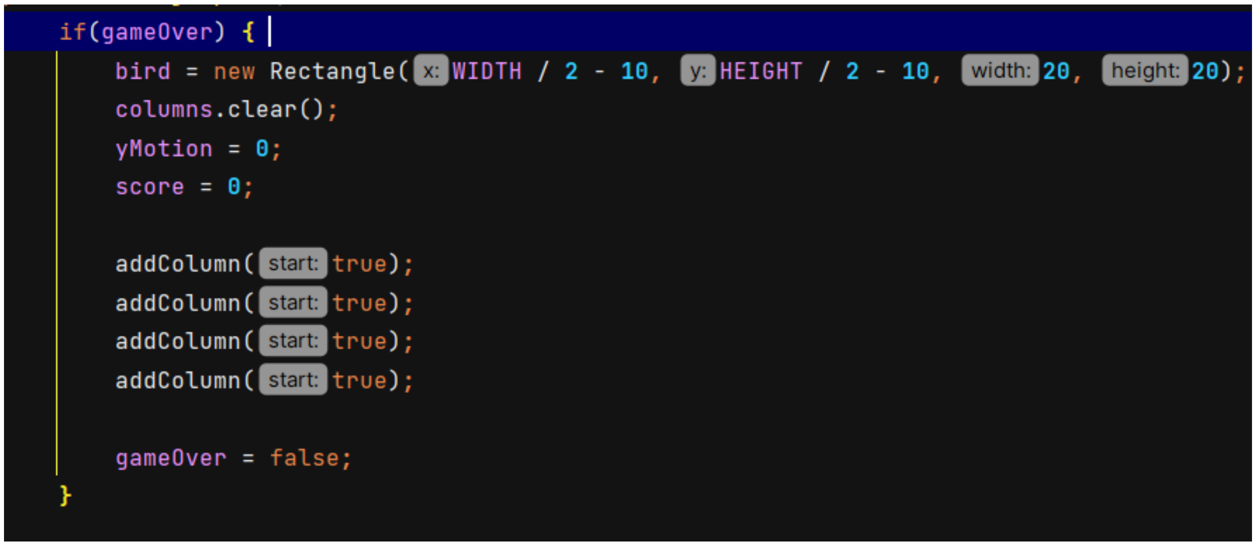
addColumn(true);

addColumn(true);

gameOver = false;

}

we clear the rectangle object and turn it into columns.clear this method will remove the columns that already exist in the game so other word If the game is over then by pressing [Space] the game will restart



```
if(gameOver) {
    bird = new Rectangle(x: WIDTH / 2 - 10, y: HEIGHT / 2 - 10, width: 20, height: 20);
    columns.clear();
    yMotion = 0;
    score = 0;

    addColumn(start: true);
    addColumn(start: true);
    addColumn(start: true);
    addColumn(start: true);

    gameOver = false;
}
```

next we add else if under the not started, else if not gameover

code;

else if(!gameOver){

}

next we set up yMotion and score here.

code:

  yMotion = 0;

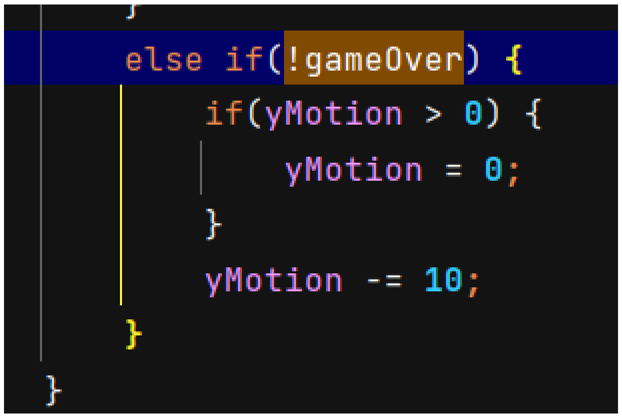
  score = 0;

then we set the else if not GameOver to make the bird Jump.

code:

 else if(!GameOver){

```
   if(yMotion > 0){
       yMotion = 0;
}
yMotion -= 10;
}
}
```

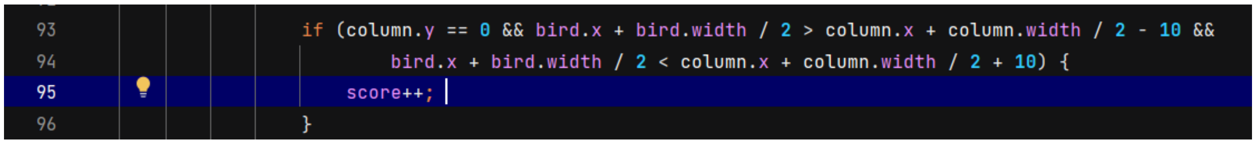


```
    }
    else if(!gameOver) {
        if(yMotion > 0) {
            yMotion = 0;
        }
        yMotion -= 10;
    }
}
```

now to setup the score we want the bird pass through the columns or pipes.

code:

```
if (column.y == 0 && bird.x + bird.width / 2 > column.x + column.width / 2 - 10 &&
                        bird.x + bird.width / 2 < column.x + column.width / 2 + 10) {
            score++;
}
```



```
if (column.y == 0 && bird.x + bird.width / 2 > column.x + column.width / 2 - 10 &&
        bird.x + bird.width / 2 < column.x + column.width / 2 + 10) {
    score++;
}
```