Project #3: The Jumper

Team:

1. Ahmed Atif (achaud16)
2. Everest Young (eyoung24)
3. Jay Yoo (gyoo3)
4. Syed Bilal Hussain (shussa11)

Description:

The goal of this game is to reach the coin at the top of the frame. The user must jump their way to the top using the arrow keys on the keyboard (left, right, up). The objective must be achieved within a given timeframe that is displayed via a timer on the top of the frame. When the timer runs out and the user has not reached the coin, the game is over. Otherwise, the users score gets incremented by 1 and the map changes.

Flourish:

Randomized collidable blocks.

Something You Invent (from prompt):

- There is animation
- It is interactive (arrow keys)
- There is a scoring mechanism and it is visible to the player (timer + score counter)
- There is a definitive ending mechanism (running out of time gives “Game Over”)
- There is a physical mechanism (gravity)
- There is collision detection (walls, coin, blocks)
- It is creative
- You implement a flourish (Randomized collidables)

Class Definitions:

Public class the Jumper {

We created the mainframe for our game using MainFrame(). We also created a progress bar using JProgressBar() for the timer. Firstly, for the main frame, we set the size of our frame to 1000 (width) by 775 (height), and added basic frame requirements such as borders, title, and the color of the frame background. Secondly, for the progress bar, we added background color and a foreground to make the timer aesthetic. Afterwards, we created a while loop, where we set the condition making it so that the counter variable will be decreasing incrementally until it hits 0. The thread.sleep() allows it to change the frequency or how fast the progress bar goes down. Lastly, we then added the progress bar to a JPanel, which we then added to the JFrame; this resulted in the progress bar appearing above the actual game frame.

}

Public class MainFrame extends JFrame {

In this mainframe, we added the GamePanel.java file to the frame. We set the size of our actual game panel, and make the actual playable game panel in the center of our mainframe. We also added a key listener so everything in the GamePanel.java can be interactive. This keylistener is directly linked to the KeyChecker method below.
}

Public class KeyChecker extends KeyAdapter{
As briefly mentioned above, this part adds the GamePanel
}

MainFramel extends JFrame{
Creates a game panel within the frame, set location, set size to be as big as the frame, set background color and add key listener to the frame. Contains a method to get gameFinish boolean and a method to reset the game.
}

Public class Coin{
This file contains the method to set the coordinates, size, hit box of the coin which the user must reach. First, variables are created for the x coordinate, y coordinate, height and width of the coin. The interaction of the coin with the user is done using a rectangular hitbox. A rectangle, approximately the size of the coin, in area, is used to represent the hitbox of the coin. The interaction between the user and the coin are explained in the Player file but are essentially based on the same principles as those between the player and and the wall.
After this, we used graphics to design the coin using the variables previously mentioned.
}

public class Wall {
Similar to Coin, variables are created for the drawing walls. The walls also have rectangular hitboxes which have collision mechanisms. This file creates a template for what a singular box in the wall would look like using graphics. The color is set and the size is set. Later in the Player file and GamePanel file, this code is used to create the walls.
}

public class GamePanel extends JPanel implements KeyListener {
- Creates a player
- Creates arraylists to store generated walls and coin
- Repaints player every 0.017s
- Contains a method to generate boundary and random platforms
- Contains a method to generate a coin for the objective
- Contains key listeners where arrow keys are
- Contains a method to return a boolean for player finishing the objective

}

public class player {

- Defines size, (x, y) coordinates and velocities of the player
- Has defined positive acceleration in the positive y direction (gravity)
- Contains a method that sets the position of the player and its hit box if the key is certain keys are pressed.
    - If no keys are pressed x speed is constantly multiplied by 0.8
    - If left arrow key is pressed: decrement x position by x speed
    - If right arrow key is pressed: increment x position by x speed
    - Set speed limit
    - Set speed to 0 when the speed is small enough to prevent sliding of the player
    - If the up arrow is pressed, make the player and the hit box move up at a certain speed only when the player is touching the ground.
- Contains horizontal & vertical wall & coin collision and set “finish” to true when player touches the coin
- Method to draw player
- Method to return “finish” boolean

}