EIE3109 Assignment Report

Author: LIU Minghao

1 Introduction

Welcome to "Save Github Cat," an engaging and thrilling gaming experience. In this official mobile application, players will embark on a mission to rescue adorable Github cats in a captivating gameplay scenario. As a Github cat descends from the heavens above, your objective as the player is to skillfully maneuver a rectangular catcher, precisely positioning it to intercept the falling feline. Upon contact with the designated rectangle, the cat's trajectory undergoes a remarkable reversal, allowing it to safely return to the sky. However, beware! Any unfortunate collision with the ground will result in the cat's disappearance.

With each successful rescue, the game diligently keeps track of the number of Github cats present on the playing field, adding an element of challenge and competitiveness to the experience.

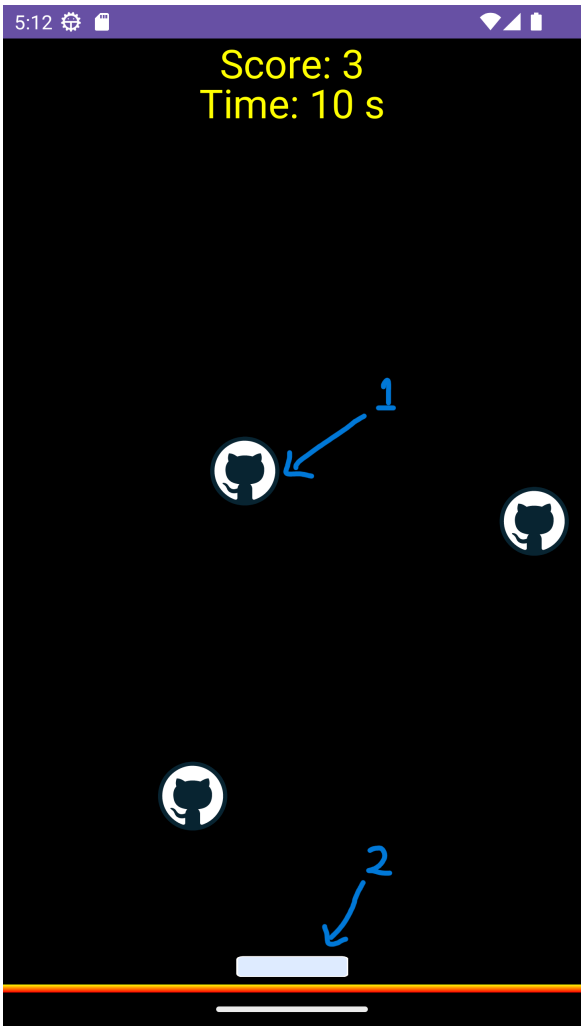

Figure 1: Introduction

As depicted in the aforementioned illustration, "Save Github Cat" boasts a simplistic and user-friendly gameplay experience. The visual representation showcases an intuitive interface, allowing players to seamlessly engage with the game mechanics. Notably, the Github cat, symbolized by the number 1 in the figure, takes center stage as the endearing character that requires rescue. Positioned strategically beneath it is the rectangular catcher, denoted by the number 2, serving as the player's control mechanism.

2 Description

There is a new class named `Rectangle` to represent rectangle catcher in the game, which is similar as the GraphicObject Class. The class has two attributes: `bitmap` and `coordinates`. The `bitmap` attribute represents the image or visual representation of the rectangle， which type is `Bitmap`. The `coordinates` attribute represents the position or location of the rectangle on the game screen. It is of type `Coordinates`, which is another class used to store and manage the x and y coordinates of a graphic object.

In the constructor `Rectangle(Bitmap bitmap)`, a Bitmap object is passed as a parameter, representing the image that will be associated with the rectangle. The constructor initializes the bitmap attribute with the provided bitmap and creates a new instance of Coordinates using the bitmap's dimensions. This allows the Rectangle object to have the appropriate size and position based on the image it represents.

The `getGraphic()` method returns the bitmap attribute, providing access to the image representation of the rectangle. The `getCoordinates()` method returns the coordinates attribute, allowing other parts of the code to retrieve the position of the rectangle on the game screen. This can be useful for various operations, such as collision detection or updating the position of the rectangle during gameplay.

Overall, this new class is very similar to the original graphicobject class, and I'll be looking at simplifying the code in future work (adding a field or judging bitmap names)

3 Methodology

3.1 Program Flow

The approximate program flow is shown below:

1. The program starts by initializing the `panel`, which displays graphics and allows user interaction.
2. The panel periodically creates new graphic objects and adds them to the graphics list.
3. The `onDraw()` method is responsible for drawing the graphics on the panel's canvas.
4. The `updateMovement()` method updates the movement of the graphics and handles collision detection.
5. The `onTouchEvent()` method handles touch events on the panel, allowing users to move the rectangle and interact with graphics.
6. The program continues running until the game is exited or terminated

As you can see, the program flow of the new game is actually still dominated by the `Panel` class. For new class `Rectangle`, it is only used to represent the rectangle object in the game. The `Panel` class is still responsible for the game logic and the game flow.

3.2 New Features

In order to implement the game, I have added some new features to the `Panel` class.

1. In the `updateMovement()` method, I've added an update rectangle and some judgement on the movement of the github cat after the collision
    - First, the code checks to see if the Github cat and the rectangle catcher collide (the right boundary of the Github cat intersects the left boundary of the rectangle, and the left boundary of the Github cat intersects the right boundary of the rectangle).
    - After a collision, the code switches the direction of Y-axis movement of the graphical object. This means that the graphical object will change the direction of movement along the vertical axis.
    - The code will also verify that the horizontal center point of the Github cat lies outside the left and right boundaries of the rectangle catcher when the collision occurs.
    - In this case, the code switches the direction of movement of the X-axis of the graphical object. This means that the graphic object will change the direction of movement along the horizontal axis.
    - This realizes the problem of updating the motion of the GitHub cat and the rectangle catcher after both collide.

```
if (bmpX + graphic.getGraphic().getWidth() >= recX && bmpX <= recX + recW &&
    bmpY + graphic.getGraphic().getHeight() >= recY && bmpY <= recY + recH)
    movement.toggleYDirection();
    coordinates.setY(recY - graphic.getGraphic().getHeight());

    int bmpMidX = bmpX + graphic.getGraphic().getWidth() / 2;
    if (bmpMidX < recX || bmpMidX > recX + recW)
        movement.toggleXDirection();
```

1. In the `onTouchEvent()` method, I've added a new feature that allows the user to move the rectangle object by touching the screen.
    - Initially, the code checks whether the touch point lies within the boundaries of the rectangle object.If the touch point falls within the rectangle, the code proceeds to record the x coordinate of the touch point.
    - Next, the code verifies if the touch point remains within the boundaries of the rectangle object.If the touch point is still within the rectangle, the code calculates the distance between the current touch point and the last touch point.Using this distance, the code moves the rectangle object by the same amount, ensuring synchronized movement with the touch gesture.
    - Finally, the code checks if the rectangle object has moved out of the screen.If the rectangle object is found to be outside the screen, the code repositions the object back to the edge of the screen.
    - By incorporating these steps, the code enables the movement of the rectangle object by simply touching the screen. This user interaction provides an intuitive and engaging way to control the position of the rectangle object within the game environment.

```
case MotionEvent.ACTION_DOWN:
    // record the last touch x coordinate
    break;
case MotionEvent.ACTION_MOVE:
    if (lastTouchRecX != -1)
    // Calculate the distance between the current coordinate and the last coordinate
    // check if the rectangle is out of the screen
    // Move the rectangle according to the distance
    break;
case MotionEvent.ACTION_UP:
    // Reset the last touch x coordinate
    lastTouchRecX = -1;
    break;
```

Figure 2: Rectangle Catcher

3. Create a new method `autoCreateGraphic()` to automatically create a new github cat object and add it to the graphics list.
    - To initiate the process, the code creates a new GraphicObject object utilizing the bmp attribute. This object represents the Github cat.
    - Subsequently, the code generates random x and y coordinates for the Github cat object. These coordinates determine the initial position of the cat within the game environment.
    - Finally, the code adds the newly created Github cat object to the graphics list. This list is responsible for managing and rendering all graphical elements in the game.
    - By following these steps, the code ensures the automatic creation of the Github cat object every 2 seconds. This periodic creation adds an element of surprise and challenge to the game, as players need to react quickly to save the falling cats.

```
timer.schedule(new TimerTask() {
    @Override
    public void run() {
        autoCreateGraphic();
    }
}, 0, 2000);

private void autoCreateGraphic() {
    // Create a new GraphicObject object using the bmp attribute
    // Randomly generate the x and y coordinates of the bmp
    // Add the bmp to the graphics list
}
```

3.3 How to Play game

The gameplay of the game is straightforward and easy to understand. The objective is to control the movement of the rectangular object and catch the falling Github cats. Here are the key gameplay mechanics:

1. Control the Rectangle: Players need to maneuver the rectangular object horizontally to align it with the falling Github cats.
2. Catching Github Cats: When a Github cat touches the rectangle, its trajectory reverses, and it ascends back into the sky.
3. Missing Github Cats: If a Github cat hits the ground without being caught, it will disappear. This results in a deduction from the player's score.
4. Scoreboard: The game features a scoreboard located at the top of the screen. It tracks the number of Github cats present in the game field. The score increases by 1 whenever a new Github cat is created, and it decreases by 1 when a Github cat hits the ground.

By effectively controlling the rectangle and maximizing the number of caught Github cats while avoiding misses, players can aim for high scores and challenge themselves to improve their performance in the game. Enjoy the simple yet addictive gameplay of catching the adorable Github cats!

4 Conclusion

4.1 Problems Encountered

One major challenge I encountered was the need for my rectangles to move horizontally and the requirement to handle the click on the Github cat within the `onTouchEvent()` method. However, I faced difficulty in synchronizing these actions simultaneously.

To address this issue, I discovered that the actions were not synchronized and implemented the solution of adding `synchronized (graphics)` at the outermost level of the function.

By adding the synchronized keyword followed by the object graphics, I ensured that concurrent access to the graphics object is synchronized. This synchronization allows for proper handling of both the movement of my rectangles and the click on the Github cat within the `onTouchEvent()` method, resolving the problem I encountered.

With this synchronization in place, I can now handle both actions simultaneously and ensure smooth and coordinated gameplay between the rectangle catcher movement and interaction with the Github cats.

Another problem I have encountered is that the Github cat is not displayed on the screen. After referring to the information in the provided link ('http://hk.javashuo.com/article/p-byipoyat-hn.html'), I was able to address this issue.

By following the instructions and code examples from the linked resource, I made the necessary modifications to ensure that the Github cat is properly displayed on the screen from the beginning of the game. This resolved

the problem of the Github cat not appearing.

4.2 Future Work

In my future work, I plan to improve "Save Github Cat" by enhancing the user interface, exploring more gameplay options, and adding interesting props to make the gameplay more engaging. Also, it will be possible to make a bounce once after Github cat into two Github cats

Additionally, I would like to consider integrating Bluetooth or WiFi connectivity to enable communication and collaboration between multiple devices, allowing for multiplayer modes and social interactions.

Furthermore, I believe incorporating audio and video capabilities into the app would enhance the immersive experience, bringing the adorable Github cats to life with captivating sound effects and animations.

These future developments aim to create a more visually appealing, interactive, and entertaining gaming experience for players. Stay tuned for the exciting updates to come!