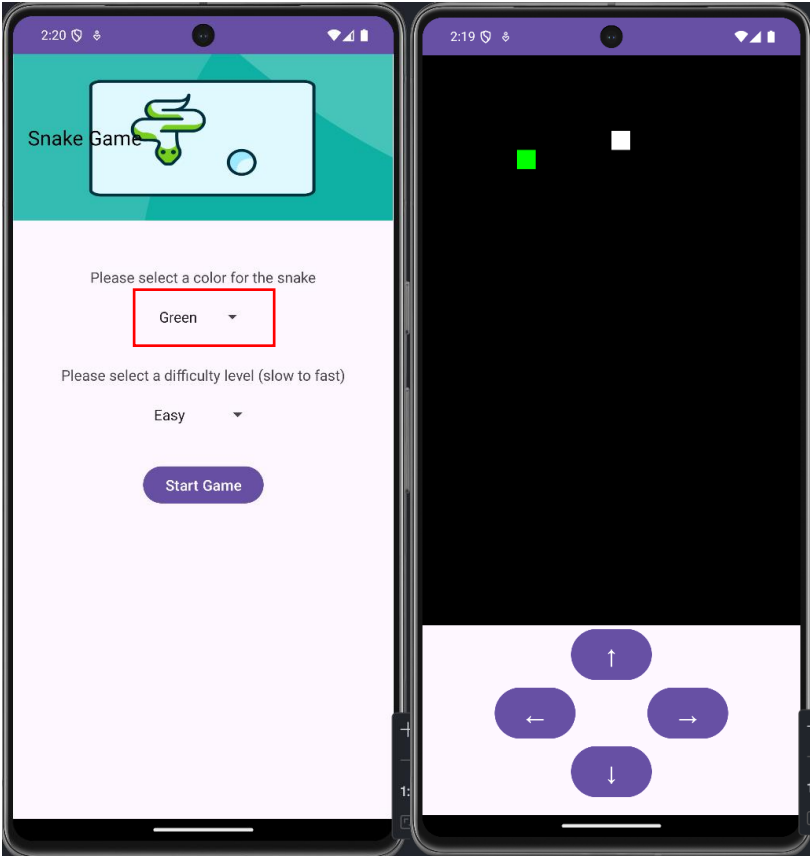
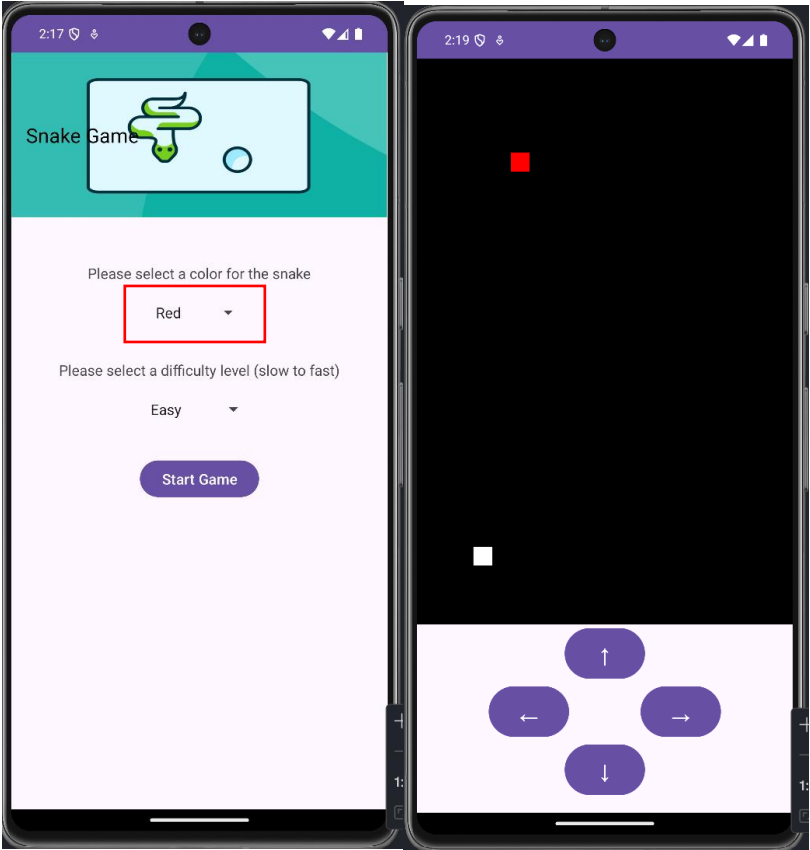
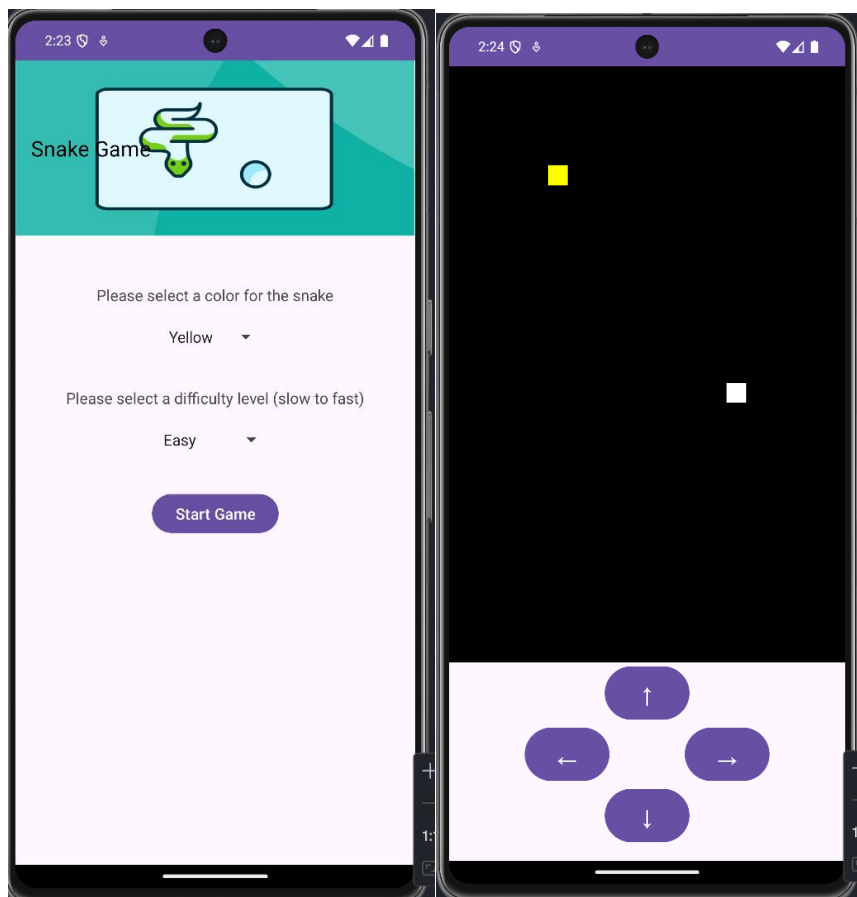
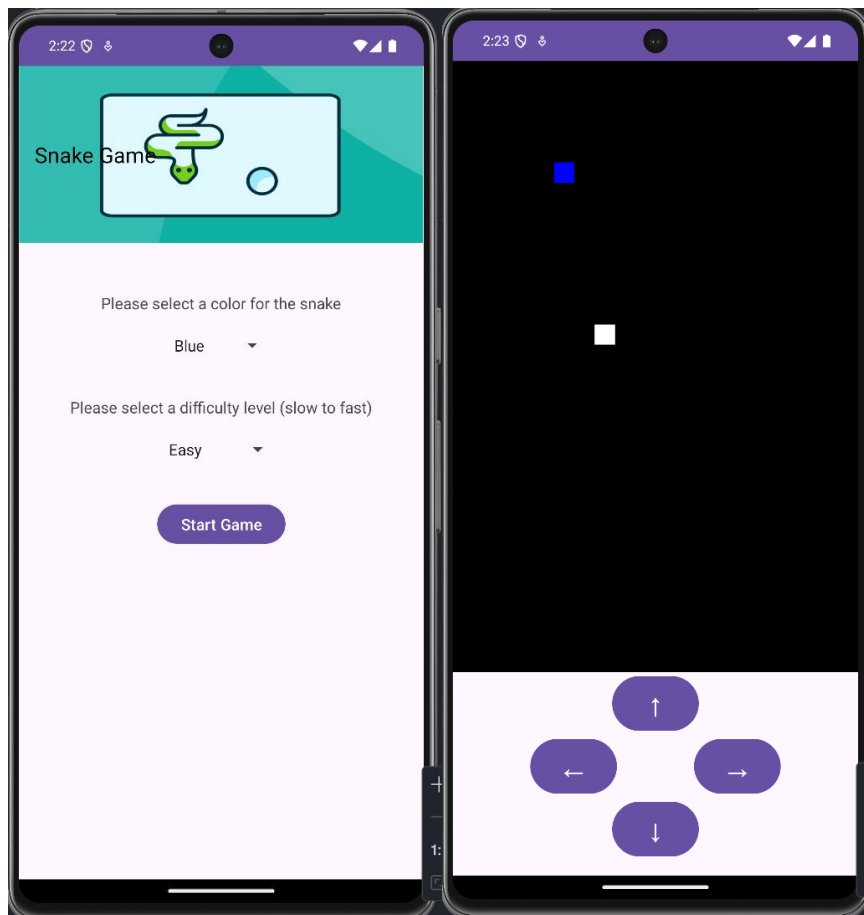
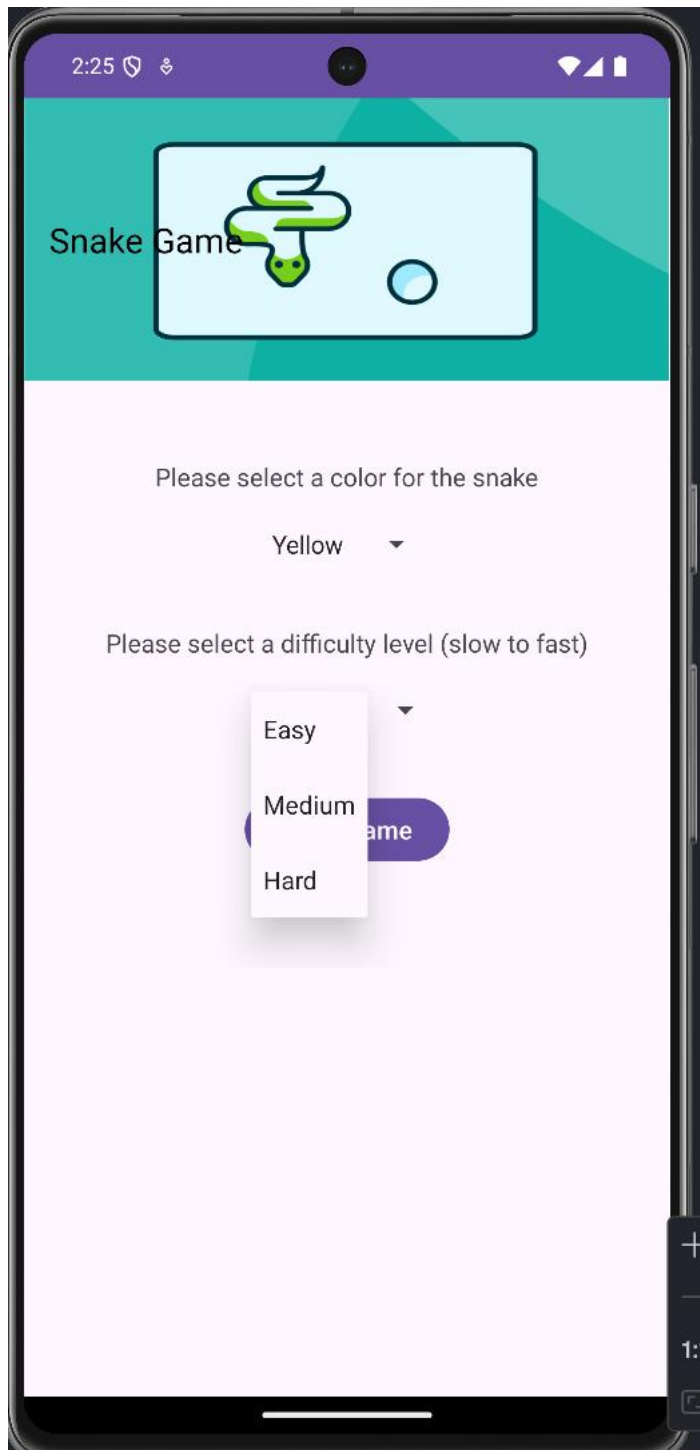


Different colors to choose from

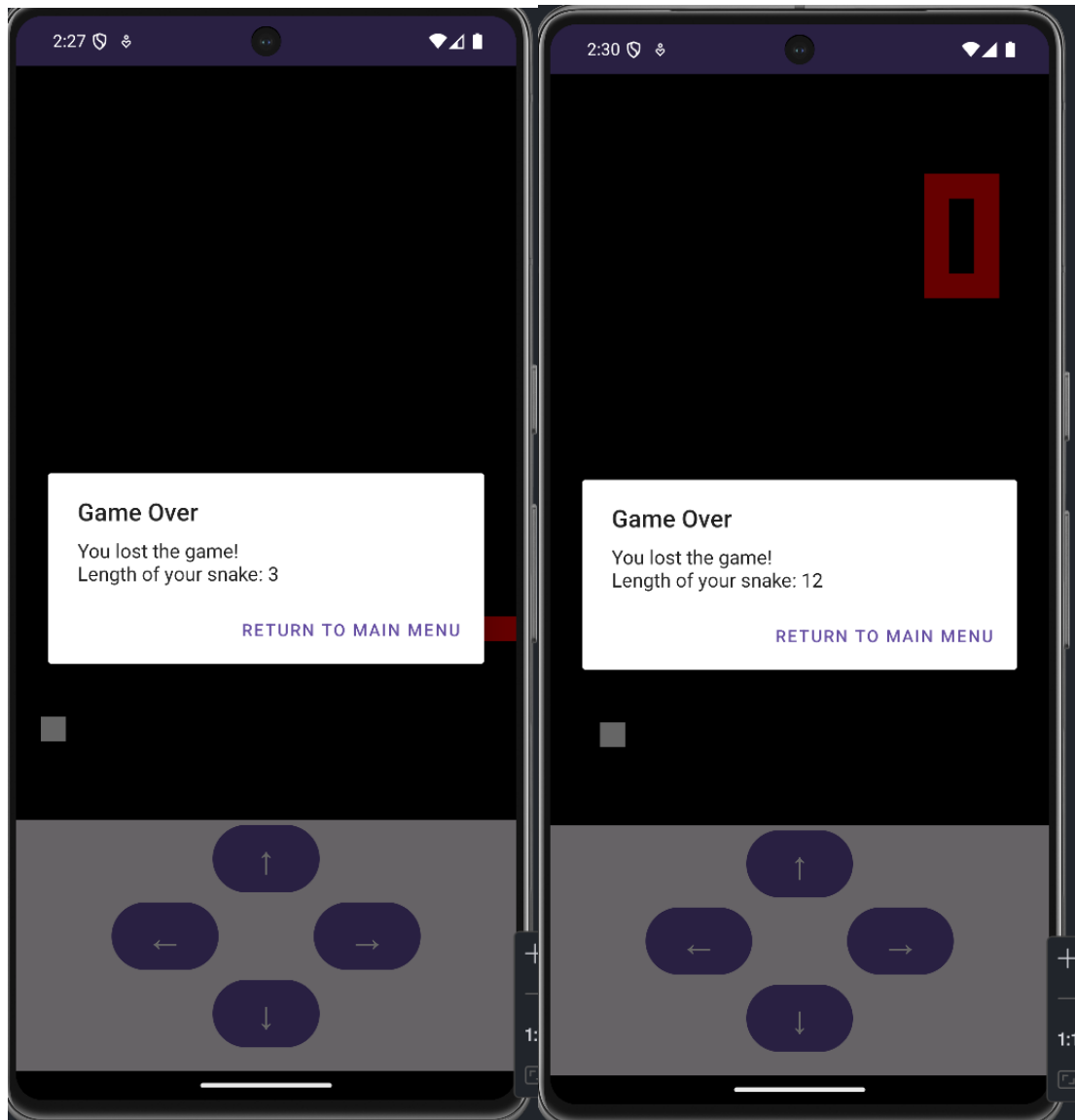




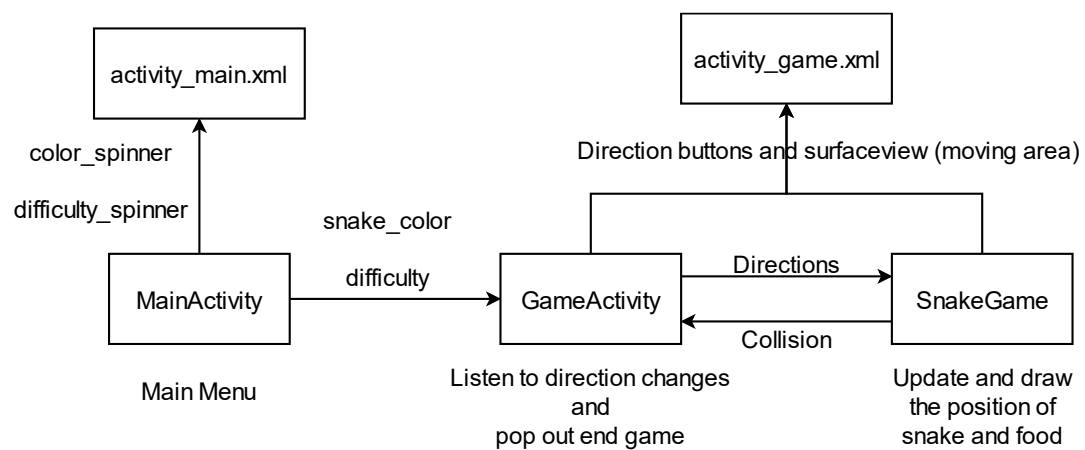
Different difficulty means different moving (update) speed of snake, which will be described later in code part.



Collision with the wall or itself will lead to the end of game, and the length of snake will be displayed.



Project Architecture

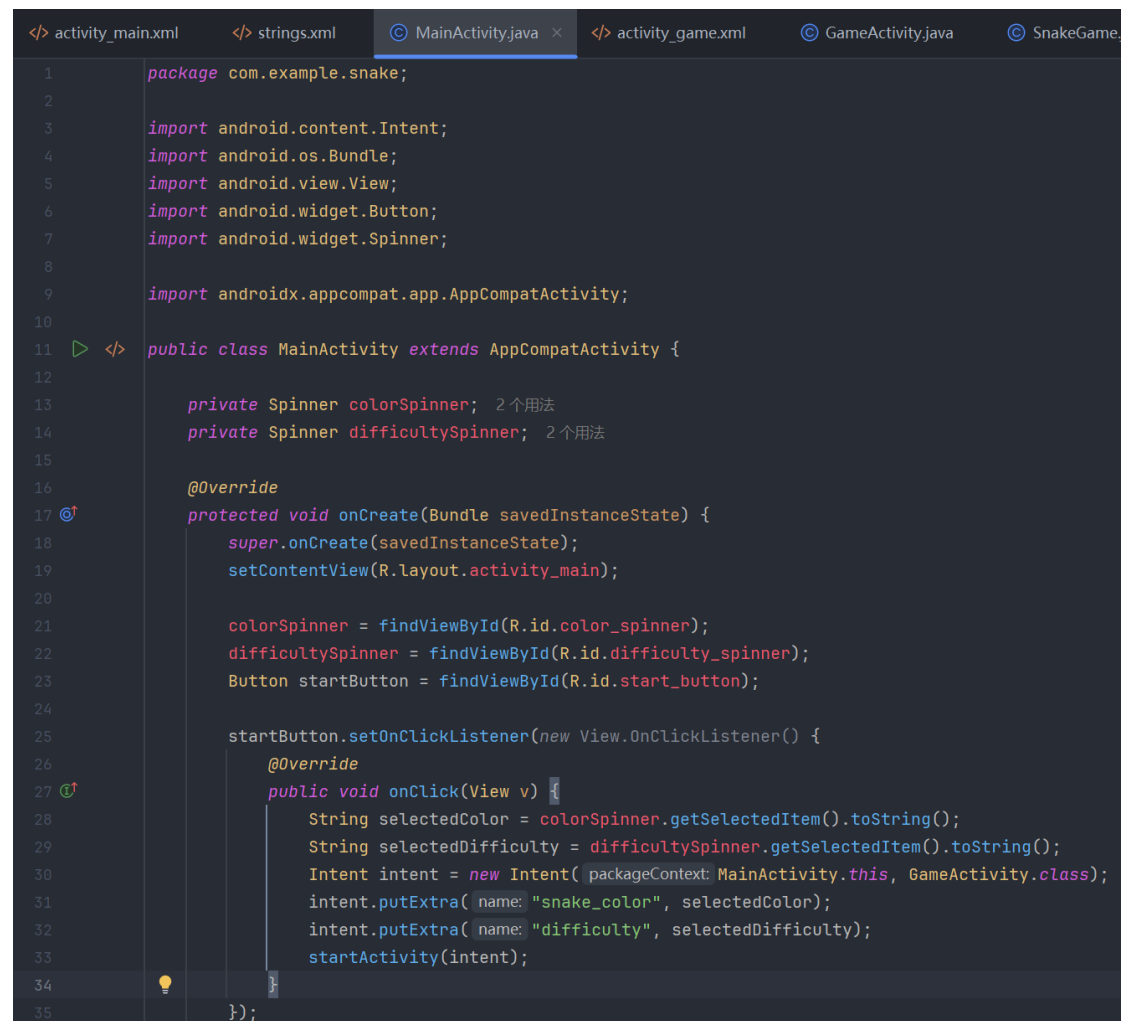


```
activity_main.xml x strings.xml MainActivity.java activity_game.xml Gar
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:orientation="vertical"
8     tools:context=".MainActivity">
9
10    <com.google.android.material.appbar.AppBarLayout
11        android:layout_width="match_parent"
12        android:layout_height="wrap_content"
13        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">
14
15        <androidx.appcompat.widget.Toolbar
16            android:id="@+id/toolbar"
17            android:layout_width="match_parent"
18            android:layout_height="180dp"
19            android:background="@drawable/snake"
20            app:title="Snake Game"
21            app:titleTextColor="@android:color/black" />
22    </com.google.android.material.appbar.AppBarLayout>
23
24    <TextView
25        android:layout_width="wrap_content"
26        android:layout_height="wrap_content"
27        android:layout_gravity="center"
28        android:layout_marginTop="50dp"
29        android:textSize="16sp"
30        android:text="Please select a color for the snake" />
31
32    <Spinner
33        android:id="@+id/color_spinner"
34        android:layout_width="wrap_content"
35        android:layout_height="wrap_content"
```

```
</> activity_main.xml × </> strings.xml © MainActivity.java </> activity_game.xml © Game

 2      <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
32      <Spinner
36          android:layout_gravity="center"
37          android:layout_margin="20dp"
38          android:entries="@array/snake_colors" />
39
40      <TextView
41          android:layout_width="wrap_content"
42          android:layout_height="wrap_content"
43          android:layout_gravity="center"
44          android:layout_marginTop="20dp"
45          android:textSize="16sp"
46          android:text="Please select a difficulty level (slow to fast)" />
47
48      <Spinner
49          android:id="@+id/difficulty_spinner"
50          android:layout_width="wrap_content"
51          android:layout_height="wrap_content"
52          android:layout_gravity="center"
53          android:layout_margin="20dp"
54          android:entries="@array/difficulty_levels" />
55
56      <Button
57          android:id="@+id/start_button"
58          android:layout_width="wrap_content"
59          android:layout_height="wrap_content"
60          android:layout_gravity="center"
61          android:layout_margin="20dp"
62          android:textSize="16sp"
63          android:text="Start Game" />
64
65  </LinearLayout>
```

Pass snake **color** and **difficulty** to **GameActivity** using intent.



```
1 package com.example.snake;
2
3 import android.content.Intent;
4 import android.os.Bundle;
5 import android.view.View;
6 import android.widget.Button;
7 import android.widget.Spinner;
8
9 import androidx.appcompat.app.AppCompatActivity;
10
11 public class MainActivity extends AppCompatActivity {
12
13     private Spinner colorSpinner; 2个用法
14     private Spinner difficultySpinner; 2个用法
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.activity_main);
20
21         colorSpinner = findViewById(R.id.color_spinner);
22         difficultySpinner = findViewById(R.id.difficulty_spinner);
23         Button startButton = findViewById(R.id.start_button);
24
25         startButton.setOnClickListener(new View.OnClickListener() {
26             @Override
27             public void onClick(View v) {
28                 String selectedColor = colorSpinner.getSelectedItem().toString();
29                 String selectedDifficulty = difficultySpinner.getSelectedItem().toString();
30                 Intent intent = new Intent( packageContext MainActivity.this, GameActivity.class);
31                 intent.putExtra( name: "snake_color", selectedColor);
32                 intent.putExtra( name: "difficulty", selectedDifficulty);
33                 startActivity(intent);
34             }
35         });
36     }
37 }
```

```
<? activity_main.xml    <? strings.xml    MainActivity.java    <? activity_game.xml    GameActivity.java    SnakeGame.java
1  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
2      xmlns:app="http://schemas.android.com/apk/res-auto"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      tools:context=".GameActivity">
7
8      <SurfaceView
9          android:id="@+id/game_view"
10         android:layout_width="0dp"
11         android:layout_height="620dp"
12         app:layout_constraintTop_toTopOf="parent"
13         app:layout_constraintBottom_toTopOf="@id/up_button"
14         app:layout_constraintStart_toStartOf="parent"
15         app:layout_constraintEnd_toEndOf="parent" />
16
17      <Button
18          android:id="@+id/up_button"
19          android:layout_width="wrap_content"
20          android:layout_height="wrap_content"
21          android:text="↑"
22          android:textSize="32sp"
23          app:layout_constraintTop_toBottomOf="@id/game_view"
24          app:layout_constraintStart_toStartOf="parent"
25          app:layout_constraintEnd_toEndOf="parent" />
26
27      <Button
28          android:id="@+id/down_button"
29          android:layout_width="wrap_content"
30          android:layout_height="wrap_content"
31          android:text="↓"
32          android:textSize="32sp"
33          app:layout_constraintTop_toBottomOf="@id/left_button"
34          app:layout_constraintStart_toStartOf="parent"
35          app:layout_constraintEnd_toEndOf="parent" />
```

```
<? activity_main.xml    <? strings.xml    MainActivity.java    <? activity_game.xml    GameActivity.java    SnakeGame.java
1      <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
27      <Button
30          android:layout_height="wrap_content"
31          android:text="↓"
32          android:textSize="32sp"
33          app:layout_constraintTop_toBottomOf="@id/left_button"
34          app:layout_constraintStart_toStartOf="parent"
35          app:layout_constraintEnd_toEndOf="parent" />
36
37      <Button
38          android:id="@+id/left_button"
39          android:layout_width="wrap_content"
40          android:layout_height="wrap_content"
41          android:text="←"
42          android:textSize="32sp"
43          app:layout_constraintTop_toBottomOf="@id/up_button"
44          app:layout_constraintStart_toStartOf="parent"
45          app:layout_constraintEnd_toStartOf="@id/right_button" />
46
47      <Button
48          android:id="@+id/right_button"
49          android:layout_width="wrap_content"
50          android:layout_height="wrap_content"
51          android:text="→"
52          android:textSize="32sp"
53          app:layout_constraintTop_toBottomOf="@id/up_button"
54          app:layout_constraintStart_toEndOf="@id/left_button"
55          app:layout_constraintEnd_toEndOf="parent" />
56
57  </androidx.constraintlayout.widget.ConstraintLayout>
```


Listen to the input of direction buttons and pass to **SnakeGame**.

```
</> activity_main.xml    </> strings.xml    MainActivity.java    </> activity_game.xml    GameActivity.java
13  <> public class GameActivity extends AppCompatActivity {
14
15      private SnakeGame snakeGame; 8个用法
16
17      @Override
18      protected void onCreate(Bundle savedInstanceState) {
19          super.onCreate(savedInstanceState);
20          setContentView(R.layout.activity_game);
21
22          SurfaceView gameView = findViewById(R.id.game_view);
23          String snakeColorString = getIntent().getStringExtra( name: "snake_color");
24          int snakeColor = Color.parseColor(snakeColorString);
25          String difficulty = getIntent().getStringExtra( name: "difficulty");
26
27          snakeGame = new SnakeGame(gameView, snakeColor, difficulty);
28          snakeGame.start();
29
30          Button upButton = findViewById(R.id.up_button);
31          Button downButton = findViewById(R.id.down_button);
32          Button leftButton = findViewById(R.id.left_button);
33          Button rightButton = findViewById(R.id.right_button);
34
35          upButton.setOnClickListener(new View.OnClickListener() {
36              @Override
37              public void onClick(View v) {
38                  snakeGame.setDirection(SnakeGame.Direction.UP);
39              }
40          });
41
42          downButton.setOnClickListener(new View.OnClickListener() {
43              @Override
44              public void onClick(View v) {
45                  snakeGame.setDirection(SnakeGame.Direction.DOWN);
46              }
47          });
48      }
49  }
```

```
</> activity_main.xml    </> strings.xml    MainActivity.java    </> activity_game.xml
13      public class GameActivity extends AppCompatActivity {
18      protected void onCreate(Bundle savedInstanceState) {
49          leftButton.setOnClickListener(new View.OnClickListener() {
50              @Override
51              public void onClick(View v) {
52                  snakeGame.setDirection(SnakeGame.Direction.LEFT);
53              }
54          });
55
56          rightButton.setOnClickListener(new View.OnClickListener() {
57              @Override
58              public void onClick(View v) {
59                  snakeGame.setDirection(SnakeGame.Direction.RIGHT);
60              }
61          });
62      }
63
64      @Override
65      protected void onDestroy() {
66          super.onDestroy();
67          if (snakeGame != null) {
68              snakeGame.stopGame();
69          }
70      }
71  }
```

Pop out the game over dialog.

```
activity_main.xml x strings.xml MainActivity.java activity_game.xml GameActivity.java x SnakeGame.java
13 public class GameActivity extends AppCompatActivity {
65     protected void onDestroy() {
66         super.onDestroy();
67         if (snakeGame != null) {
68             snakeGame.stopGame();
69         }
70     }
71
72     public void showGameOverDialog(final int snakeLength) { 1 个用法
73         runOnUiThread(new Runnable() {
74             @Override
75             public void run() {
76                 new AlertDialog.Builder(context: GameActivity.this)
77                     .setTitle("Game Over")
78                     .setMessage("You lost the game!\nLength of your snake: " + snakeLength)
79                     .setPositiveButton(text: "Return to Main Menu", new DialogInterface.OnClickListener() {
80                         @Override
81                         public void onClick(DialogInterface dialog, int which) {
82                             dialog.dismiss();
83                             Intent intent = new Intent(packageContext: GameActivity.this, MainActivity.class);
84                             intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_NEW_TASK);
85                             startActivity(intent);
86                             finish();
87                         }
88                     })
89                     .show();
90             }
91         });
92     }
93 }
```

Definition of **SnakeGame** class.

```
y_main.xml SnakeGame.java x strings.xml MainActivity.j
public class SnakeGame extends Thread { 6 个用法

    private final SurfaceView surfaceView; 7 个用法
    private final Paint paint; 5 个用法
    private final int snakeColor; 2 个用法
    private final List<int[]> snake; 13 个用法
    private int[] food; 9 个用法
    private Direction direction; 12 个用法
    private boolean running; 4 个用法
    private boolean paused; 3 个用法
    private int sleepTime; 4 个用法
    private int gridSize; 11 个用法
    private int gridCountX; 4 个用法
    private int gridCountY; 4 个用法
    private static final String TAG = "SnakeGame"; 0 个用法
    private final Handler handler; 4 个用法

    public enum Direction { 19 个用法
        UP, DOWN, LEFT, RIGHT 6 个用法
    }
}
```

Pass the **surfaceview** (moving area of snake) ,**color** and **difficulty** when initializing.

```
public class SnakeGame extends Thread { 6 个用法

    public SnakeGame(SurfaceView surfaceView, int snakeColor, String difficulty) {
        this.surfaceView = surfaceView;
        this.snakeColor = snakeColor;
        this.paint = new Paint();
        this.snake = new ArrayList<>();
        this.direction = Direction.RIGHT;
        this.running = true;
        this.paused = true;
        this.handler = new Handler(Looper.getMainLooper());

        snake.add(new int[]{5, 5});
        switch (difficulty) {
            case "Medium":
                this.sleepTime = 120;
                break;
            case "Hard":
                this.sleepTime = 100;
                break;
            case "Easy":
            default:
                this.sleepTime = 160;
                break;
        }
    }

    @Override
    public void run() {
        handler.post(gameLoop);
    }
}
```

The running loop using handler.

```
ty_main.xml  SnakeGame.java  </> strings.xml  MainActivity.java  </> activity

public class SnakeGame extends Thread { 6 个用法

    @Override
    public void run() {
        handler.post(gameLoop);
    }

    private final Runnable gameLoop = new Runnable() { 2 个用法
        @Override
        public void run() {
            if (running) {
                draw();
                if (!paused) {
                    update();
                    draw();
                }
                handler.postDelayed(this, sleepTime);
            }
        }
    };
}
```

Pass directions to **update()** method and update the **snake list** storing the locations of snake body. When player takes opposite direction to current direction, call **reverseSnake()** method to make a reverse in case of **misjudged collision**.

```
public class SnakeGame extends Thread { 6个用法

    public void setDirection(Direction newDirection) { 4个用法
        // Check if the new direction is opposite to the current direction
        if ((this.direction == Direction.UP && newDirection == Direction.DOWN) ||
            (this.direction == Direction.DOWN && newDirection == Direction.UP) ||
            (this.direction == Direction.LEFT && newDirection == Direction.RIGHT) ||
            (this.direction == Direction.RIGHT && newDirection == Direction.LEFT)) {
            // Reverse the snake
            reverseSnake();
        } else {
            this.direction = newDirection;
        }
        this.paused = false;
        //Log.d(TAG, "Direction set to: " + newDirection);
    }
}
```

```
private void reverseSnake() { 1个用法
    List<int[]> reversedSnake = new ArrayList<>();
    for (int i = snake.size() - 1; i >= 0; i--) {
        reversedSnake.add(snake.get(i));
    }
    snake.clear();
    snake.addAll(reversedSnake);
    // Reverse the direction
    switch (this.direction) {
        case UP:
            this.direction = Direction.DOWN;
            break;
        case DOWN:
            this.direction = Direction.UP;
            break;
        case LEFT:
            this.direction = Direction.RIGHT;
            break;
        case RIGHT:
            this.direction = Direction.LEFT;
            break;
    }
}
```

Update the location of **head and food**. **Add length** when the head come across the food. When collision, set running to false to end the game and pass the **length of snake** to **GameActivity**.

```
private void update() { 1个用法
    int[] head = snake.get(0);
    int[] newHead = new int[]{head[0], head[1]};

    switch (direction) {
        case UP:
            newHead[1]--;
            break;
        case DOWN:
            newHead[1]++;
            break;
        case LEFT:
            newHead[0]--;
            break;
        case RIGHT:
            newHead[0]++;
            break;
    }

    if (newHead[0] < 0 || newHead[0] >= gridCountX || newHead[1] < 0 || newHead[1] >= gridCountY || isCollision(newHead)) {
        running = false;
        ((GameActivity) surfaceView.getContext()).showGameOverDialog(snake.size());
        return;
    }

    snake.add(0, newHead);

    if (newHead[0] == food[0] && newHead[1] == food[1]) {
        generateFood();
    } else {
        snake.remove(0, snake.size() - 1);
    }
}
```

Divide the surfaceview (black) as **grid** and draw the **snake and food** (white) on it.

```
private void draw() { 2个用法
    SurfaceHolder holder = surfaceView.getHolder();
    Canvas canvas = holder.lockCanvas();
    if (canvas != null) {
        // Calculate grid size and count based on SurfaceView dimensions
        gridSize = Math.min(surfaceView.getWidth() / 20, surfaceView.getHeight() / 20);
        gridCountX = surfaceView.getWidth() / gridSize;
        gridCountY = surfaceView.getHeight() / gridSize;

        //Log.d(TAG, "Grid Size: " + gridSize + ", Grid Count X: " + gridCountX + ", Grid Count Y: " + gridCountY);

        if (food == null) {
            generateFood();
        }

        canvas.drawColor(Color.BLACK);
        paint.setColor(snakeColor);
        for (int[] segment : snake) {
            //Log.d(TAG, "Drawing snake segment at: (" + segment[0] + ", " + segment[1] + ")");
            canvas.drawRect( left: segment[0] * gridSize, top: segment[1] * gridSize, right: (segment[0] + 1) * gridSize, bottom: (segment[1] + 1) * gridSize, paint);
        }
        paint.setColor(Color.WHITE);
        //Log.d(TAG, "Drawing food at: (" + food[0] + ", " + food[1] + ")");
        canvas.drawRect( left: food[0] * gridSize, top: food[1] * gridSize, right: (food[0] + 1) * gridSize, bottom: (food[1] + 1) * gridSize, paint);
        holder.unlockCanvasAndPost(canvas);
    }
}
```

Randomly generate food on the surfaceview.

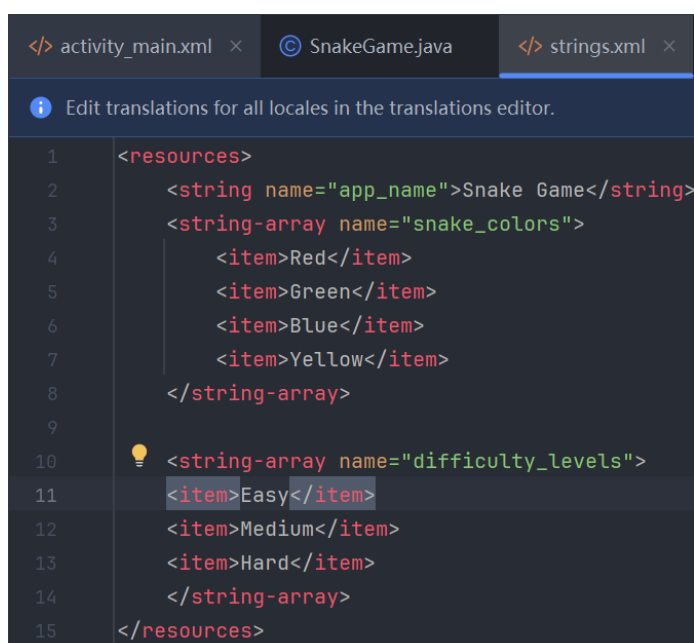
```
private void generateFood() { 2个用法
    Random random = new Random();
    // Ensure gridCountX and gridCountY are positive
    if (gridCountX > 0 && gridCountY > 0) {
        food = new int[]{random.nextInt( bound: gridCountX - 2) + 1, random.nextInt( bound: gridCountY - 2) + 1};
    } else {
        food = new int[]{1, 1}; // Default value if gridCountX or gridCountY is not positive
    }
    //Log.d(TAG, "Generated food at: (" + food[0] + ", " + food[1] + ")");
}
```

Separate the **head and body** using segment, if the head has **the same position** as any block of body indicates a collision.

```
private boolean isCollision(int[] position) { 1个用法
    for (int[] segment : snake) {
        if (segment[0] == position[0] && segment[1] == position[1]) {
            return true;
        }
    }
    return false;
}
```

```
public void stopGame() { 1个用法
    running = false;
    handler.removeCallbacks(gameLoop);
}
```

Values of spinners.



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Snake Game</string>
    <string-array name="snake_colors">
        <item>Red</item>
        <item>Green</item>
        <item>Blue</item>
        <item>Yellow</item>
    </string-array>
    <string-array name="difficulty_levels">
        <item>Easy</item>
        <item>Medium</item>
        <item>Hard</item>
    </string-array>
</resources>
```