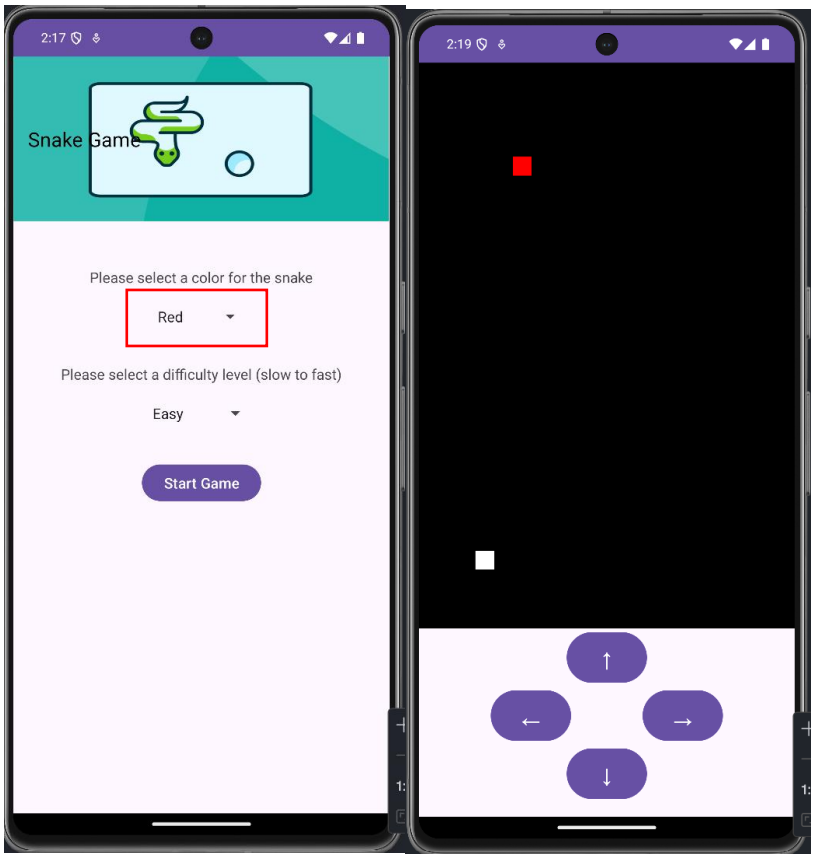
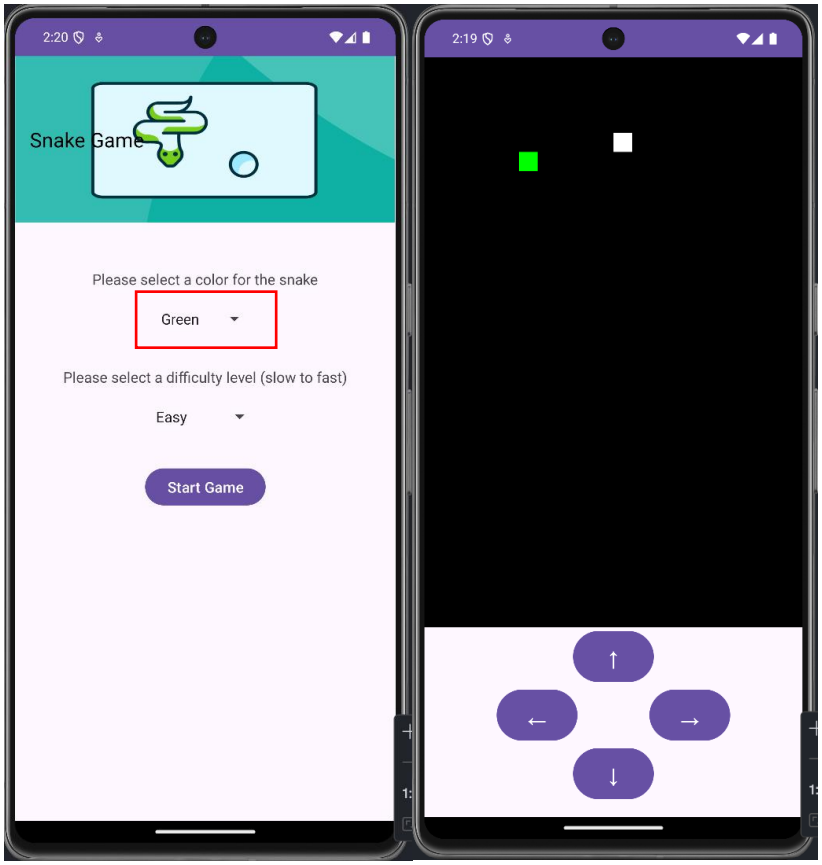
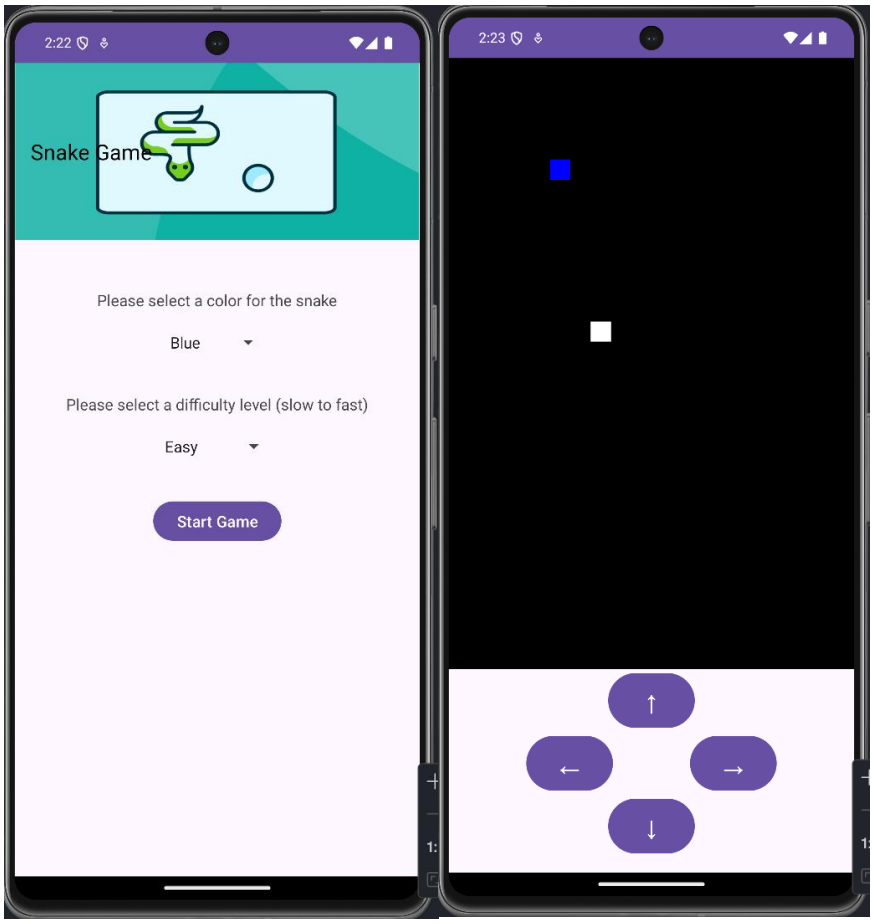
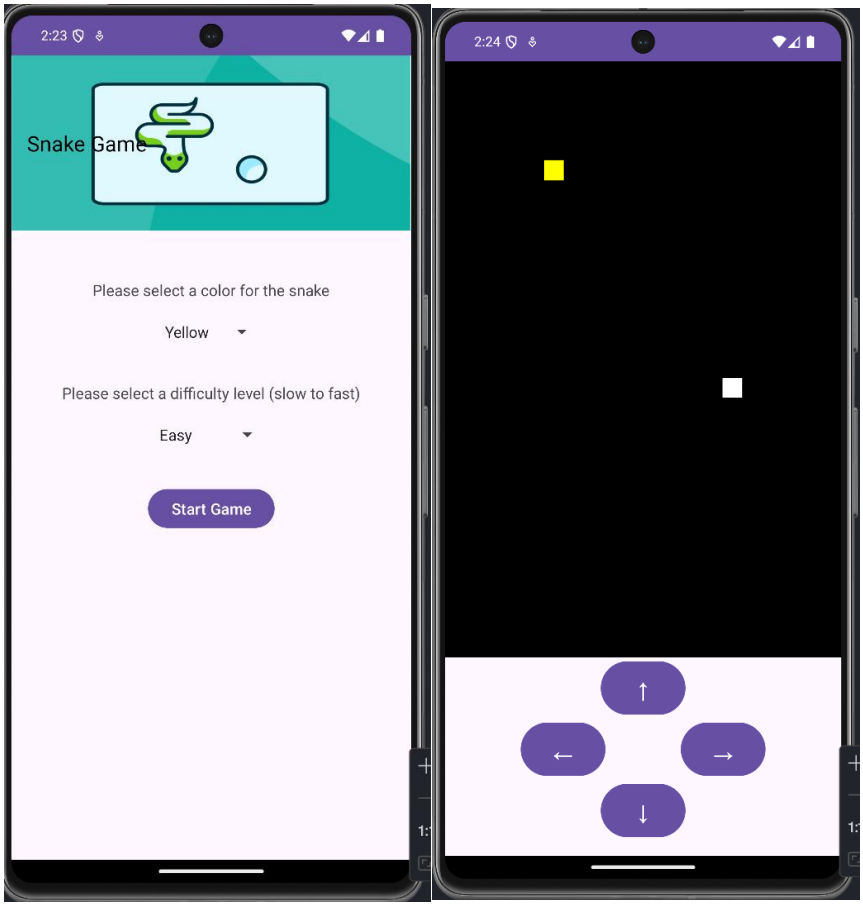
Different colors to choose from

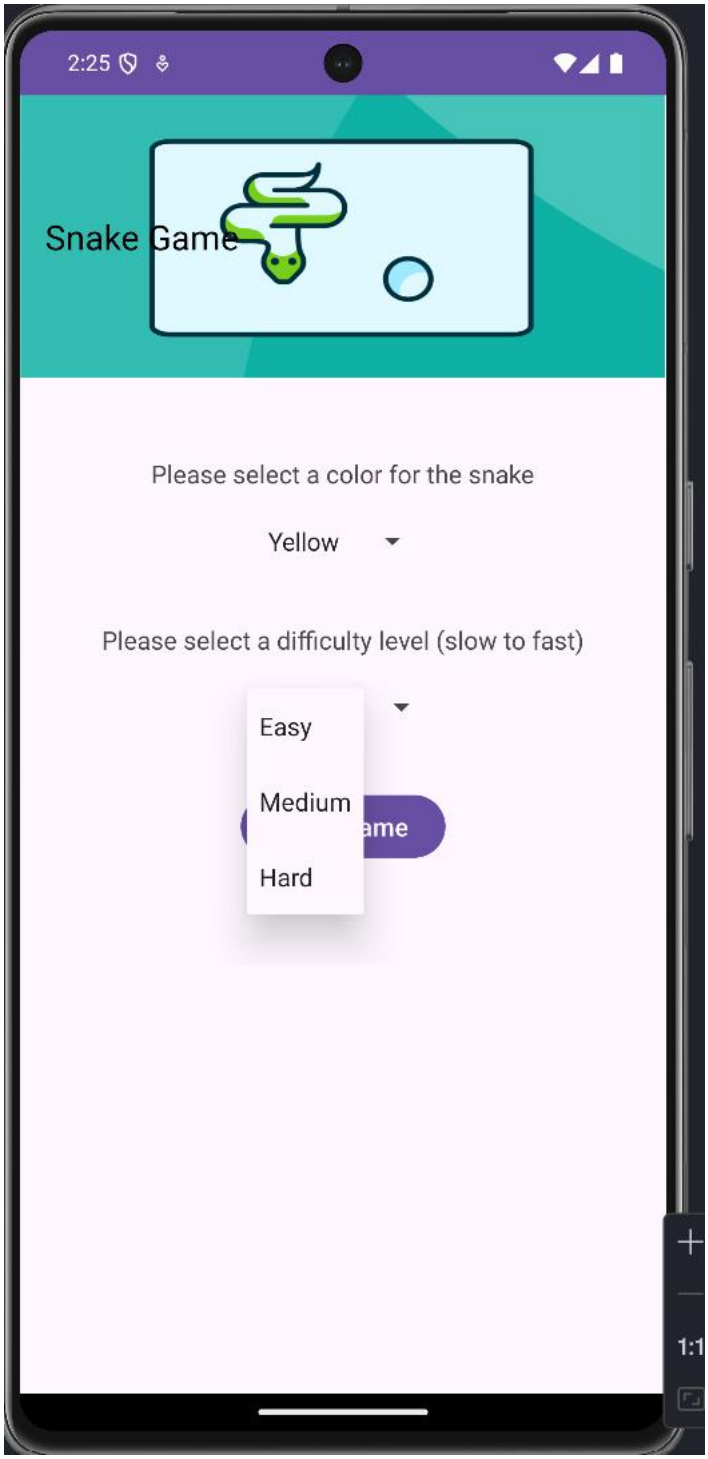




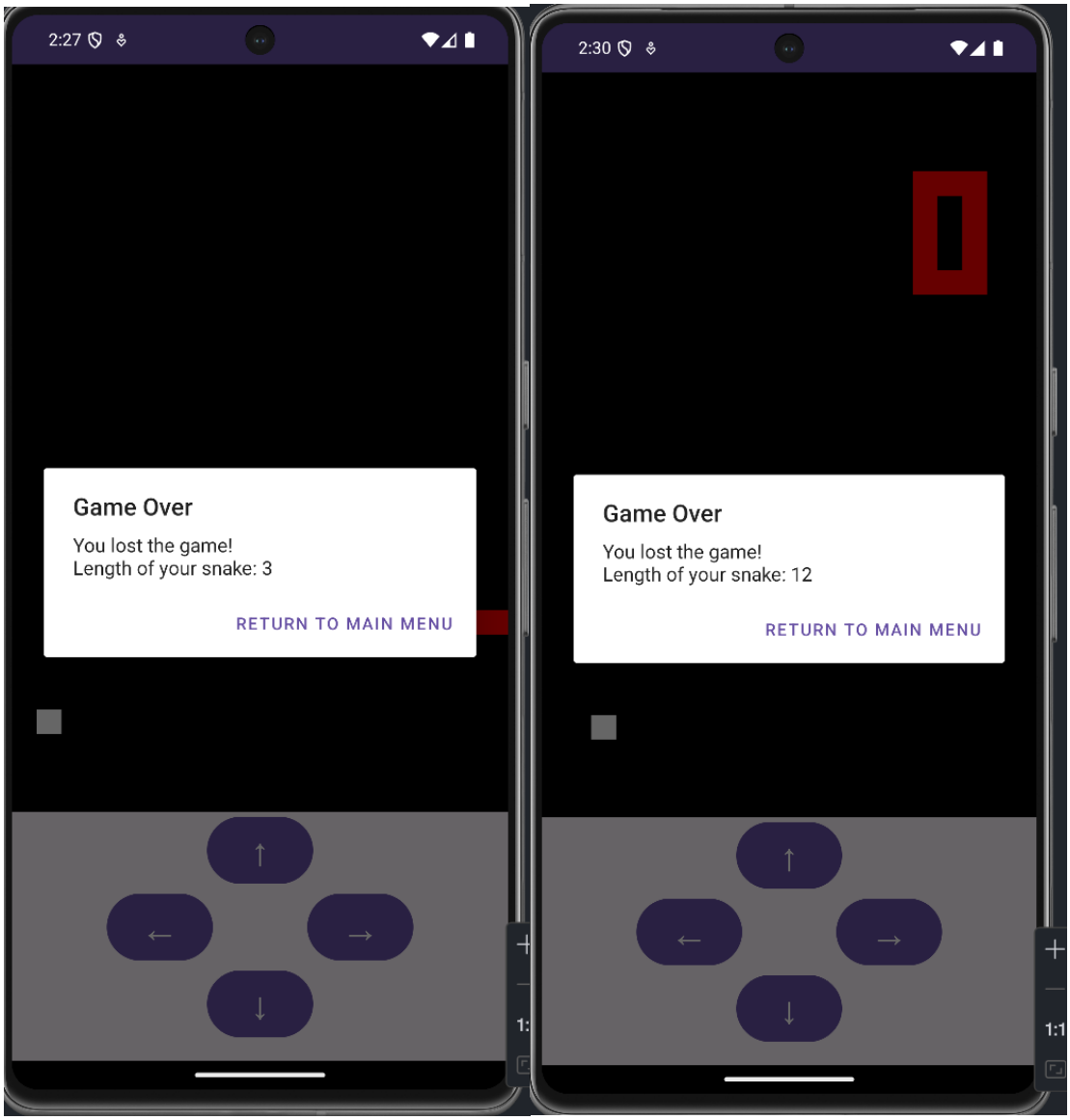

2:22
Snake Game
Please select a color for the snake
Blue
Please select a difficulty level (slow to fast)
Easy
Start Game
2:23
↑
←
→
↓


2:23
Snake Game
Please select a color for the snake
Yellow
Please select a difficulty level (slow to fast)
Easy
Start Game
2:24
↑
←
→
↓

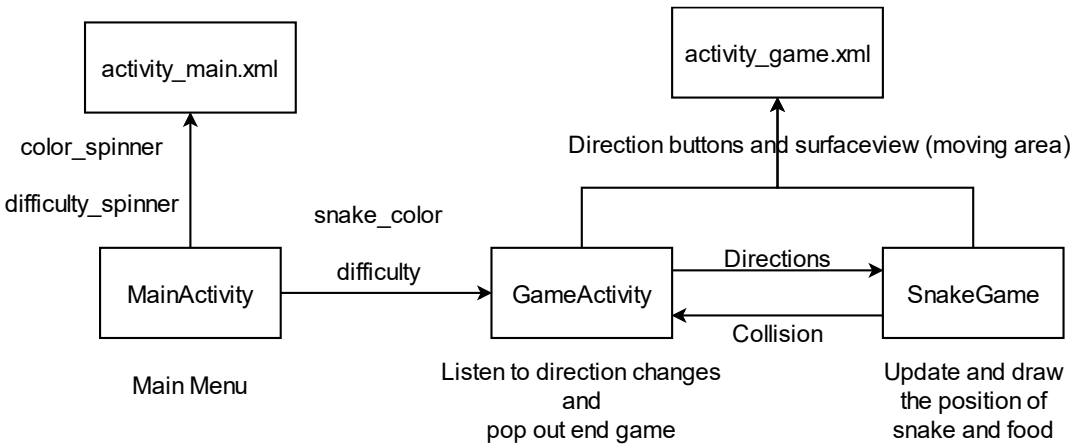
Different difficulty means different moving (update) speed of snake, which will be described later in code part.



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 | strings.xml | MainActivity.java | activity_game.xml | Gar

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="180dp"
            android:background="@drawable/snake"
            app:title="Snake Game"
            app:titleTextColor="@android:color/black" />
    </com.google.android.material.appbar.AppBarLayout>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="50dp"
        android:textSize="16sp"
        android:text="Please select a color for the snake" />

    <Spinner
        android:id="@+id/color_spinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

activity_main.xml | strings.xml | MainActivity.java | activity_game.xml | Game

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    <Spinner
        android:layout_gravity="center"
        android:layout_margin="20dp"
        android:entries="@array/snake_colors" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="20dp"
        android:textSize="16sp"
        android:text="Please select a difficulty level (slow to fast)" />

    <Spinner
        android:id="@+id/difficulty_spinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="20dp"
        android:entries="@array/difficulty_levels" />

    <Button
        android:id="@+id/start_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="20dp"
        android:textSize="16sp"
        android:text="Start Game" />

</LinearLayout>
```

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

activity_main.xml | strings.xml | MainActivity.java | activity_game.xml | GameActivity.java | SnakeGame.

```java
package com.example.snake;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Spinner;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    private Spinner colorSpinner;  2 个用法
    private Spinner difficultySpinner;  2 个用法

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        colorSpinner = findViewById(R.id.color_spinner);
        difficultySpinner = findViewById(R.id.difficulty_spinner);
        Button startButton = findViewById(R.id.start_button);

        startButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String selectedColor = colorSpinner.getSelectedItem().toString();
                String selectedDifficulty = difficultySpinner.getSelectedItem().toString();
                Intent intent = new Intent( packageContext: MainActivity.this, GameActivity.class);
                intent.putExtra( name: "snake_color", selectedColor);
                intent.putExtra( name: "difficulty", selectedDifficulty);
                startActivity(intent);
            }
        });
```

activity_main.xml | strings.xml | MainActivity.java | activity_game.xml | GameActivity.java | SnakeGame.java

```xml
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".GameActivity">

    <SurfaceView
        android:id="@+id/game_view"
        android:layout_width="0dp"
        android:layout_height="620dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@id/up_button"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <Button
        android:id="@+id/up_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="↑"
        android:textSize="32sp"
        app:layout_constraintTop_toBottomOf="@id/game_view"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <Button
        android:id="@+id/down_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="↓"
        android:textSize="32sp"
        app:layout_constraintTop_toBottomOf="@id/left_button"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />
```

activity_main.xml | strings.xml | MainActivity.java | activity_game.xml | GameActivity.java | SnakeGame.java

```xml
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    <Button
        android:layout_height="wrap_content"
        android:text="↓"
        android:textSize="32sp"
        app:layout_constraintTop_toBottomOf="@id/left_button"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <Button
        android:id="@+id/left_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="←"
        android:textSize="32sp"
        app:layout_constraintTop_toBottomOf="@id/up_button"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toStartOf="@id/right_button" />

    <Button
        android:id="@+id/right_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="→"
        android:textSize="32sp"
        app:layout_constraintTop_toBottomOf="@id/up_button"
        app:layout_constraintStart_toEndOf="@id/left_button"
        app:layout_constraintEnd_toEndOf="parent" />

</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

```java
public class GameActivity extends AppCompatActivity {

    private SnakeGame snakeGame;  8 个用法

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_game);

        SurfaceView gameView = findViewById(R.id.game_view);
        String snakeColorString = getIntent().getStringExtra( name: "snake_color");
        int snakeColor = Color.parseColor(snakeColorString);
        String difficulty = getIntent().getStringExtra( name: "difficulty");

        snakeGame = new SnakeGame(gameView, snakeColor, difficulty);
        snakeGame.start();

        Button upButton = findViewById(R.id.up_button);
        Button downButton = findViewById(R.id.down_button);
        Button leftButton = findViewById(R.id.left_button);
        Button rightButton = findViewById(R.id.right_button);

        upButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                snakeGame.setDirection(SnakeGame.Direction.UP);
            }
        });

        downButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                snakeGame.setDirection(SnakeGame.Direction.DOWN);
            }
```

activity_main.xml | strings.xml | MainActivity.java | activity_game.xml

```java
public class GameActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        leftButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                snakeGame.setDirection(SnakeGame.Direction.LEFT);
            }
        });

        rightButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                snakeGame.setDirection(SnakeGame.Direction.RIGHT);
            }
        });
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (snakeGame != null) {
            snakeGame.stopGame();
        }
    }
```

Pop out the game over dialog.

```java
activity_main.xml | strings.xml | MainActivity.java | activity_game.xml | GameActivity.java | SnakeGame.java

public class GameActivity extends AppCompatActivity {
    protected void onDestroy() {
        super.onDestroy();
        if (snakeGame != null) {
            snakeGame.stopGame();
        }
    }

    public void showGameOverDialog(final int snakeLength) {  1 个用法
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                new AlertDialog.Builder( context: GameActivity.this)
                        .setTitle("Game Over")
                        .setMessage("You lost the game!\nLength of your snake: " + snakeLength)
                        .setPositiveButton( text: "Return to Main Menu", new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                dialog.dismiss();
                                Intent intent = new Intent( packageContext: GameActivity.this, MainActivity.class);
                                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_NEW_TASK);
                                startActivity(intent);
                                finish();
                            }
                        })
                        .show();
            }
        });
    }
}
```

Definition of **SnakeGame** class.

```java
_main.xml | SnakeGame.java | 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.

```java
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( r: 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**.

```java
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);
    }
```

```java
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**.

```java
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( i: 0, newHead);

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

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

```java
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.

```java
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.

```java
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;
}
```

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

Values of spinners.

activity_main.xml | SnakeGame.java | strings.xml

Edit translations for all locales in the translations editor.

```xml
<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>
```