PROJECT | Gravity Snake

By: SSR Team
Qingshan Yao
Shufan Huang
Xiangyu Lin
Feng Chang

Contents

Part 1

Overview

The motivation

-New play of classic games.
-More interesting and challenging.
-Enlightenment of Flappy Bird.
-Broad development space.

division of labor

Qingshan Yao

Team leader

Utilization of acceleration sensor
Connect some parts

Xiangyu Lin

Team member

User interface

Shufan Huang

Team member

Key algorithms

Feng Chang

Team member

Art designer
algorithms

Part 2

Utilization of acceleration sensor

getSystemService(SENSOR_SERVICE);
getSystemService(SENSOR_SERVICE);
getSystemService(SENSOR_SERVICE);
getSystemService(SENSOR_SERVICE);
getDefaultSensor(Sensor.TYPE_GRAVITY);

sensors

getSystemService(SENSOR_SERVICE);

↓

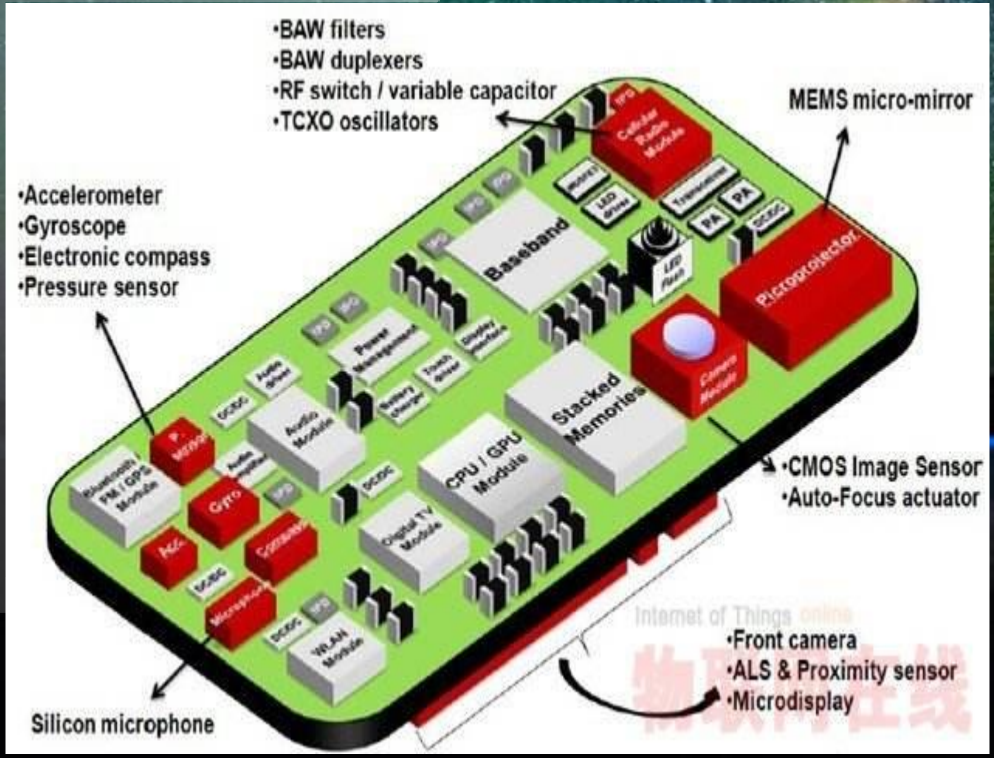
getDefaultSensor(Sensor.TYPE_GRAVITY);

↓

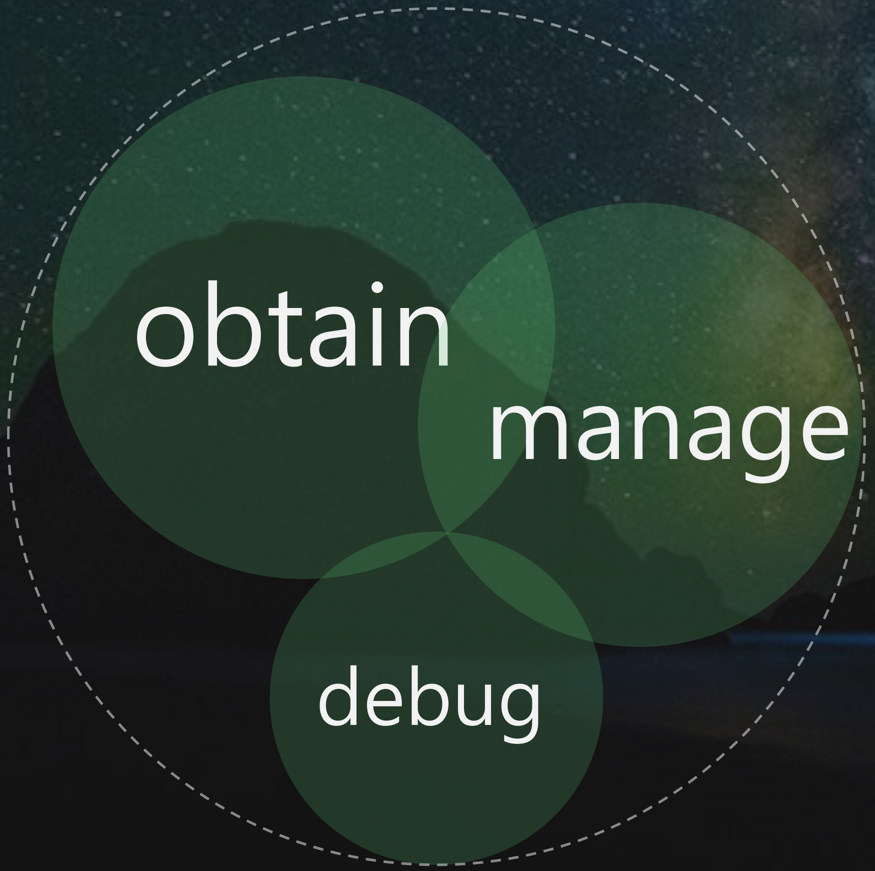
SensorEventListener

↓

onSensorChanged, onAccuracyChanged



Utilization



obtain

Use tools above to get the measurable statics

manage

Manage the threshold values and other conditions

debug

According to the main function, adjust the values

Part 3

User Interface

How to unite all the things beautifully and logically?

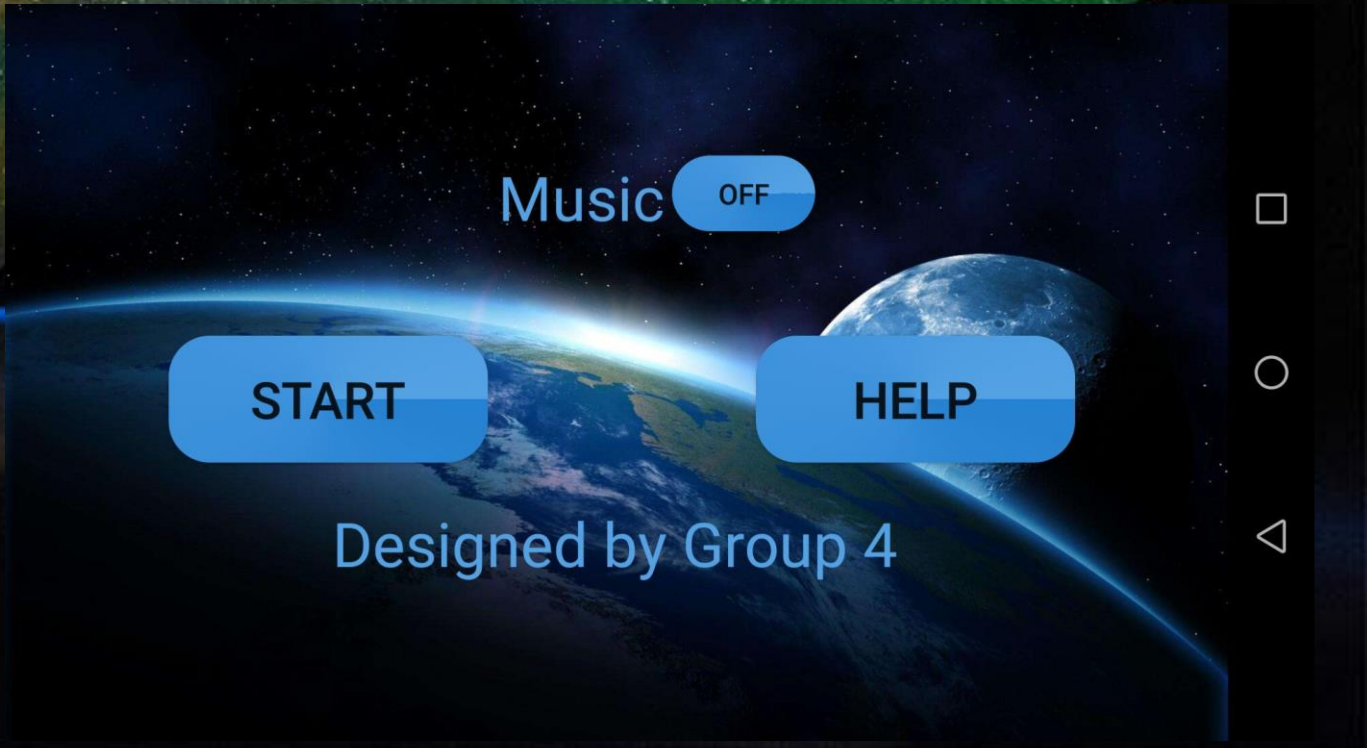
Music

Picture

Background

Snake

Button



How to use codes to create more beautiful things?

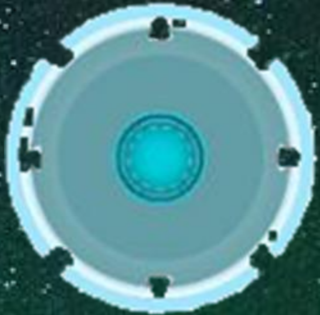
Artist?

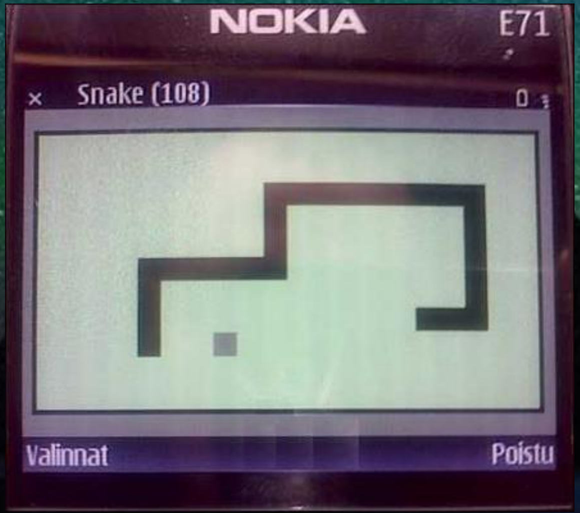
Programmer?

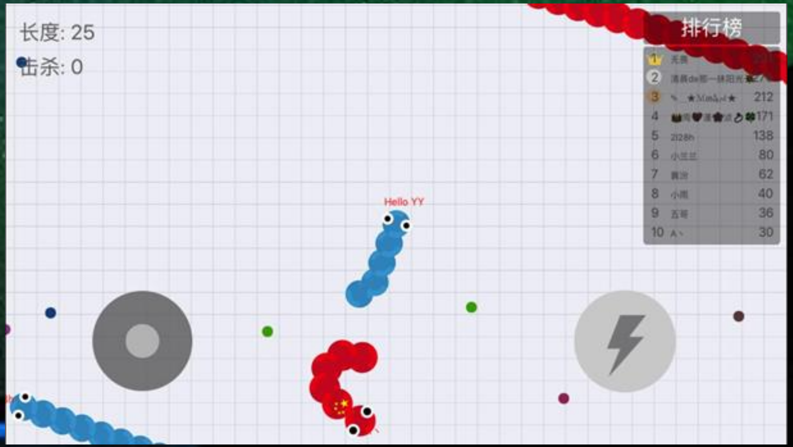
How to arrange the beautiful pictures and buttons with the most rational codes?

Part 4

Key algorithms



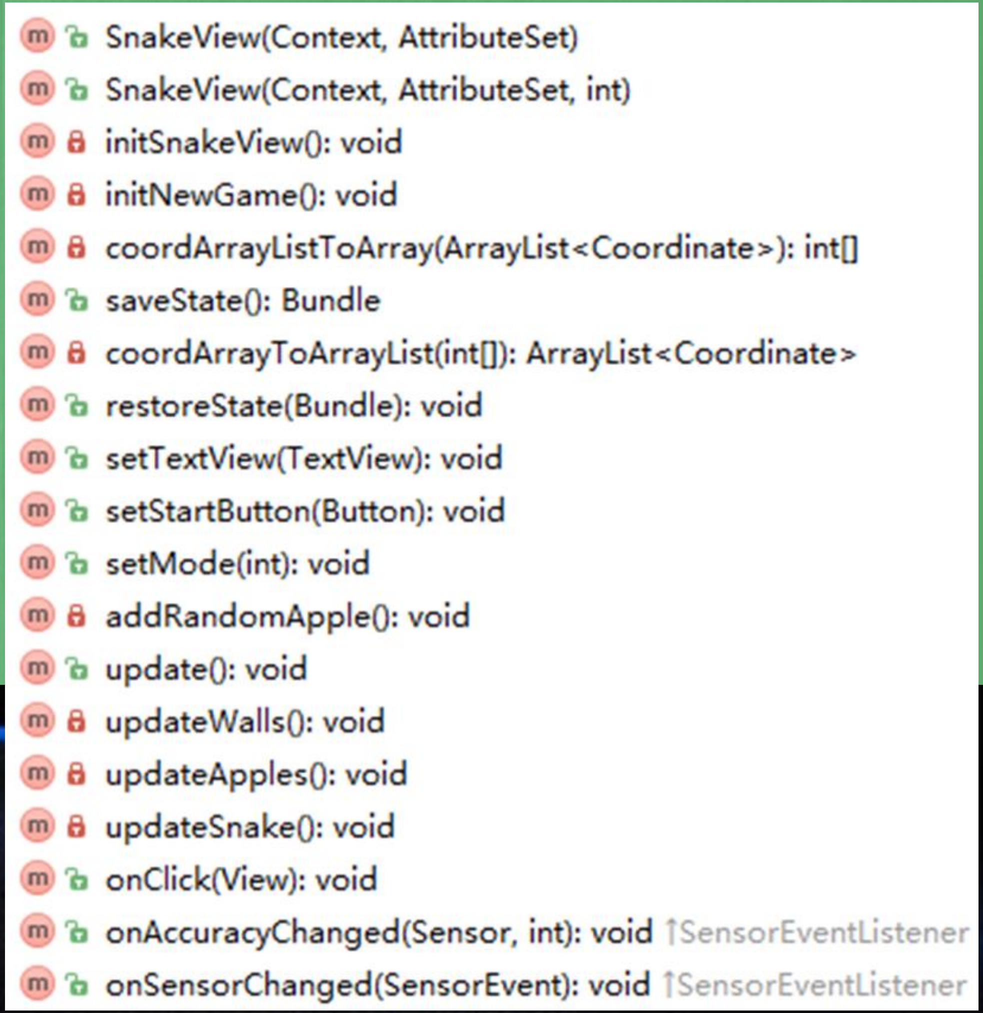




IDEA

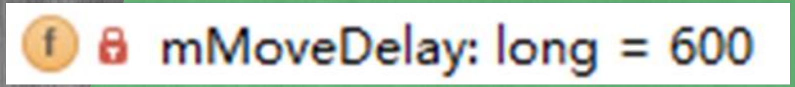
More advanced sensor→Evolution of games

SnakeView

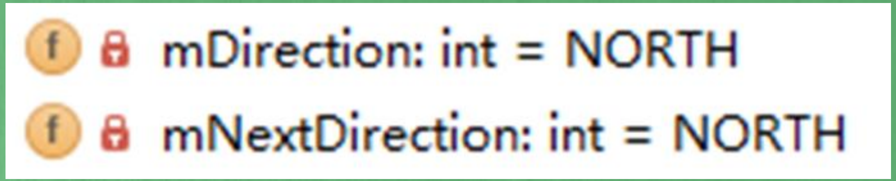


```
SnakeView(Context, AttributeSet)
SnakeView(Context, AttributeSet, int)
initSnakeView(): void
initNewGame(): void
coordArrayListToArray(ArrayList<Coordinate>): int[]
saveState(): Bundle
coordArrayToArrayList(int[]): ArrayList<Coordinate>
restoreState(Bundle): void
setTextView(TextView): void
setStartButton(Button): void
setMode(int): void
addRandomApple(): void
update(): void
updateWalls(): void
updateApples(): void
updateSnake(): void
onClick(View): void
onAccuracyChanged(Sensor, int): void ↑SensorEventListener
onSensorChanged(SensorEvent): void ↑SensorEventListener
```

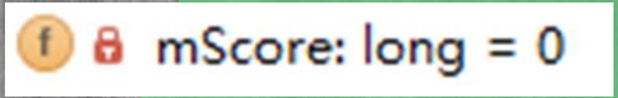
Important Variables



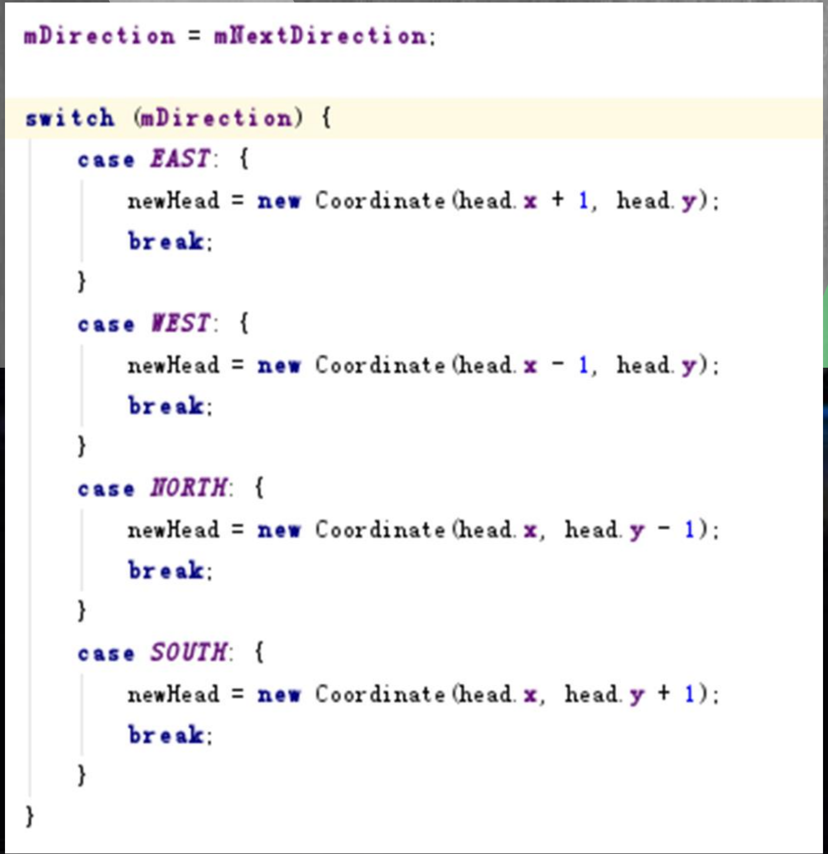
```
mMoveDelay: long = 600
```



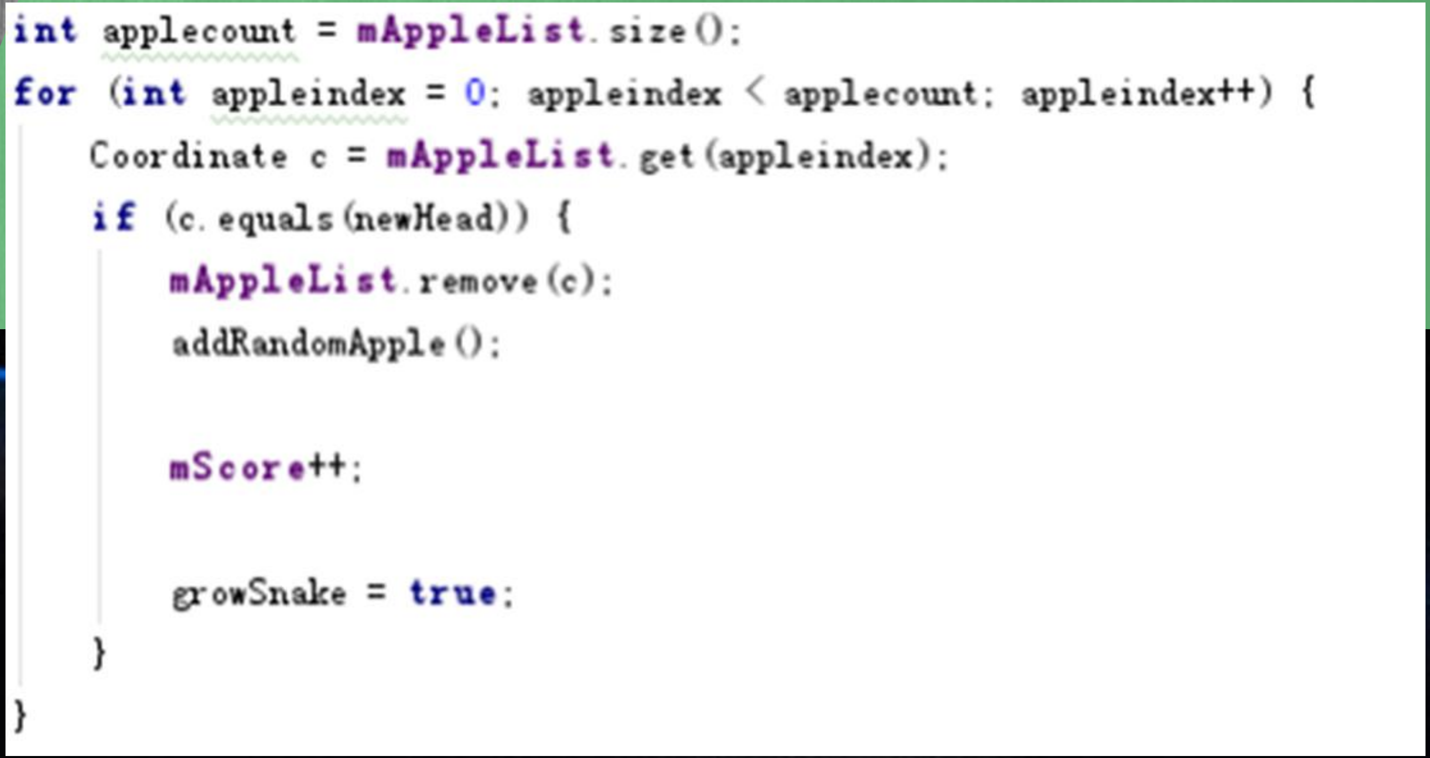
```
mDirection: int = NORTH
mNextDirection: int = NORTH
```



```
mScore: long = 0
```



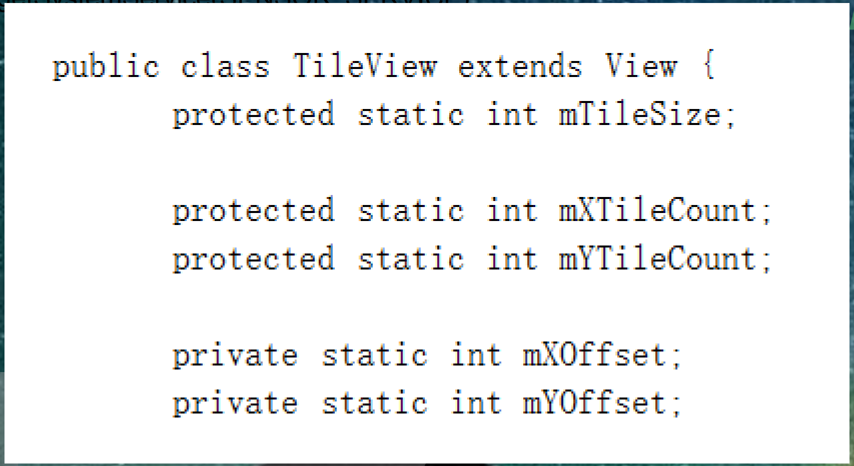
```java
mDirection = mNextDirection;

switch (mDirection) {
    case EAST: {
        newHead = new Coordinate(head.x + 1, head.y);
        break;
    }
    case WEST: {
        newHead = new Coordinate(head.x - 1, head.y);
        break;
    }
    case NORTH: {
        newHead = new Coordinate(head.x, head.y - 1);
        break;
    }
    case SOUTH: {
        newHead = new Coordinate(head.x, head.y + 1);
        break;
    }
}
```



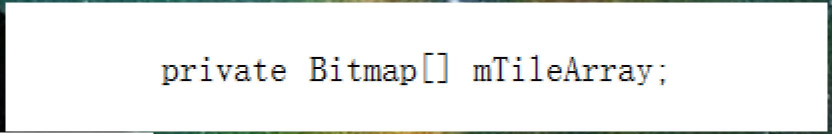
```java
int applecount = mAppleList.size();
for (int appleindex = 0; appleindex < applecount; appleindex++) {
    Coordinate c = mAppleList.get(appleindex);
    if (c.equals(newHead)) {
        mAppleList.remove(c);
        addRandomApple();

        mScore++;

        growSnake = true;
    }
}
```

getSystemService(SENSOR_SERVICE);
getSystemService(SENSOR_SERVICE);
getSystemService(SENSOR_SERVICE);
getSystemService(SENSOR_SERVICE);

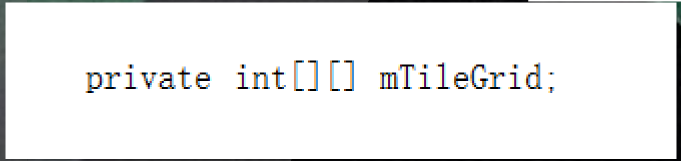
tileview



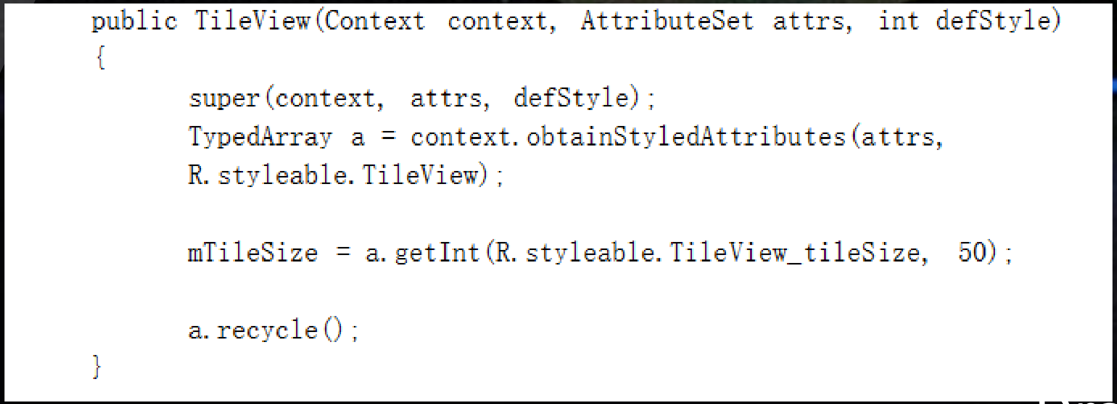
```
public class TileView extends View {
        protected static int mTileSize;

        protected static int mXTileCount;
        protected static int mYTileCount;

        private static int mXOffset;
        private static int mYOffset;
```



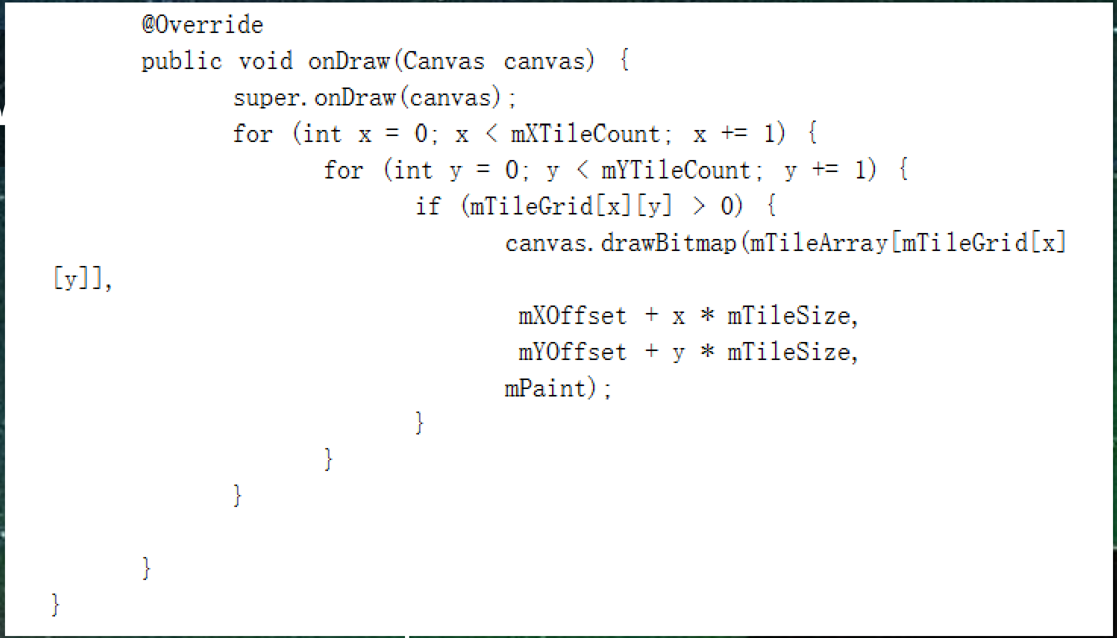
```
        private Bitmap[] mTileArray;
```



```
    private int[][] mTileGrid;
```



```
    public TileView(Context context, AttributeSet attrs, int defStyle)
    {
        super(context, attrs, defStyle);
        TypedArray a = context.obtainStyledAttributes(attrs,
        R.styleable.TileView);

        mTileSize = a.getInt(R.styleable.TileView_tileSize, 50);

        a.recycle();
    }
```



```
        @Override
        public void onDraw(Canvas canvas) {
                super.onDraw(canvas);
                for (int x = 0; x < mXTileCount; x += 1) {
                        for (int y = 0; y < mYTileCount; y += 1) {
                                if (mTileGrid[x][y] > 0) {
                                        canvas.drawBitmap(mTileArray[mTileGrid[x]
[y]],

                                         mXOffset + x * mTileSize,
                                         mYOffset + y * mTileSize,
                                        mPaint);
                                }
                        }
                }

        }
}
```

↓

A two-dimensional array of integers

↓

obtain the new attribute value

↓

Draw the canvas onto the mobile phone

Thanks