Министерство образования и науки Российской Федерации

ФГБОУ ВО Ульяновский государственный технический университет

Кафедра «Вычислительная техника»

Лабораторная работа №4

«Простая игра»

Выполнил студент

группы ИВТАПбд-21

Мухаметзянов Т. А.

Ульяновск, 2022

**Задача:**

Написать программу, реализующую игру Лабиринт. Для отрисовки игрового процесса нужно использовать SurfaceView. В качестве объектов игрового мира использовать примитивные фигуры (окружности, прямоугольники, линии и т.д). Игру сделать состоящей из нескольких уровней, на каждом из которых повышается сложность игрового процесса. Использовать изображения в качестве объектов игрового мира и фона. Игру сделать состоящей из нескольких уровней, на каждом из которых повышается сложность игрового процесса. Сложность можно повышать плавно (в зависимости от варианта)

**Особенности реализации**

Вся отрисовка, функции загрузки уровней и отрисовки картинок и функции движения реализованы в потоке, что позволяет обновлять все это из фоновых потоков.

Загрузка нового уровня:

private void newLvl() {  
 switch (numberLvl) {  
 case 1:  
 x = 1;  
 y = 1;  
 break;  
 case 2:  
 arrLvl1 = arrLvl2;  
 x = 1;  
 y = 8;  
 break;  
 case 3:  
 arrLvl1 = arrLvl3;  
 x = 2;  
 y = 8;  
 break;  
 }  
}

Диалоговое окно о результате игры:

builder.setTitle(R.string.victoryTitle);  
builder.setMessage(R.string.victoryMessage1);  
AlertDialog victoryDialog1 = builder.create();

Обработка нажатий по экрану:

public boolean onTouchEvent(*@NonNull* MotionEvent *event*) {  
  
 // определение нажатия  
 if (*event*.getAction() == MotionEvent.ACTION\_DOWN) {  
 drawThread.doClick(*event*.getX(), *event*.getY());  
 return true;  
 }  
 // определение стороны движения, то есть куда нам двигаться  
 if (*event*.getAction() == MotionEvent.ACTION\_MOVE) {  
  
 // получение событий по осям  
 float x1 = *event*.getX();  
 float y1 = *event*.getY();  
 // определение центра ячейки игрока  
 // выбор направления движения  
 float dx = x1 - drawThread.getCx();  
 float dy = y1 - drawThread.getCy();  
 float absDx = Math.**abs**(dx);  
 float absDy = Math.**abs**(dy);  
  
 if (absDx > drawThread.getArrSize() || absDy > drawThread.getArrSize()) {  
 // выбор направления движения  
 if (absDx > absDy) {  
 // move in x-direction  
 if (dx > 0)  
 drawThread.movePlayer(Direction.RIGHT);  
 else  
 drawThread.movePlayer(Direction.LEFT);  
 } else {  
 // move in y-direction  
 if (dy > 0)  
 drawThread.movePlayer(Direction.DOWN);  
 else  
 drawThread.movePlayer(Direction.UP);  
 }  
 }  
 return true;  
 }  
 return super.onTouchEvent(*event*);  
}

Геттеры для получения размерности массива и состояний центра игрока по осям:

public float getCx() {  
 return playerCenterX;  
}  
public float getCy() {  
 return playerCenterY;  
}  
public float getArrSize() {  
 return arrSize;  
}

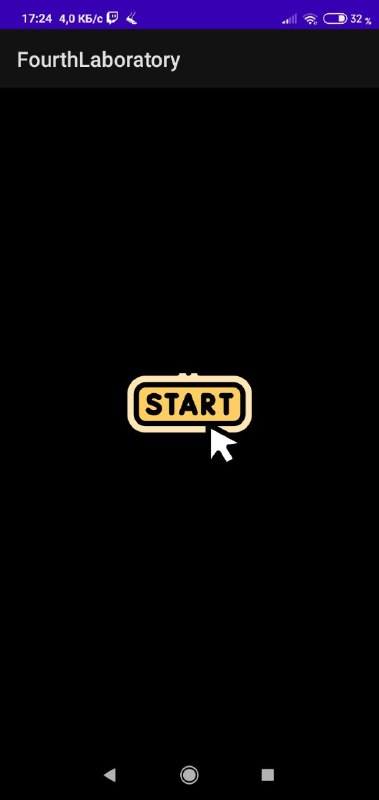
Отрисовка лабиринта:

for (int j = 0; j < h; j++) {  
 for (int i = 0; i < w; i++) {  
 switch (arrLvl1[j][i]) {  
 // field  
 case 0:  
 canvas.drawCircle(i \* 70, j \* 70, 35, yellowField);  
 break;  
 // wall  
 case 1:  
 canvas.drawCircle(i \* 70, j \* 70,35, white);  
 break;  
 // finish  
 case 2:  
 case 5:  
 case 6:  
 canvas.drawCircle(i \* 70, j \* 70, 35, blueFinish);  
 break;  
 // teleport  
 case 3:  
 canvas.drawCircle(i \* 70, j \* 70, 35, teleportPurple);  
 break;  
 // trap  
 case 4:  
 canvas.drawCircle(i \* 70, j \* 70, 35, trapGreen);  
 break;  
 }  
 }  
 }  
 canvas.drawCircle(x \* 70, y \* 70, 35, wallBlack);  
 canvas.drawCircle(x \* 70, y \* 70, 30, white);  
 canvas.drawCircle(x \* 70, y \* 70, 25, trapGreen);  
 canvas.drawCircle(x \* 70, y \* 70, 20, blueFinish);  
 canvas.drawCircle(x \* 70, y \* 70, 15, yellowField);  
 canvas.drawCircle(x \* 70, y \* 70, 10, teleportPurple);  
}

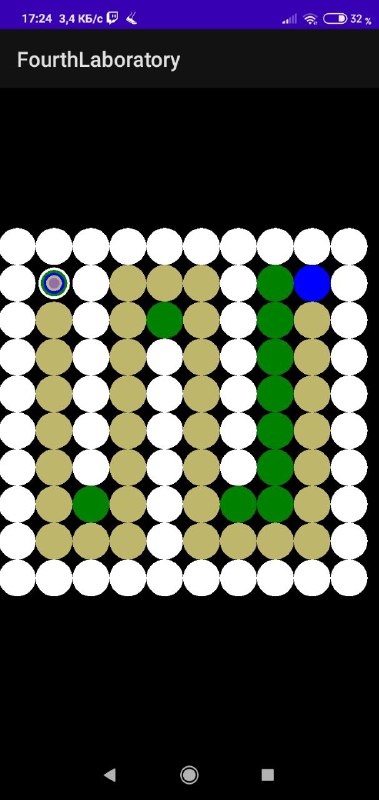
Автоматная модель загрузки потока:

*@Override*public void run() {  
 while (runFlag) {  
 switch (state) {  
 case 0:  
 doMenuAction();  
 break;  
 case 1:  
 doLvl1Action();  
 break;  
 case 2:  
 doGameAction();  
 break;  
 case 3:  
 doLvl2Action();  
 break;  
 case 4:  
 doLvl3Action();  
 break;  
 case 5:  
 doVictoryAction();  
 break;  
 }  
 }  
}

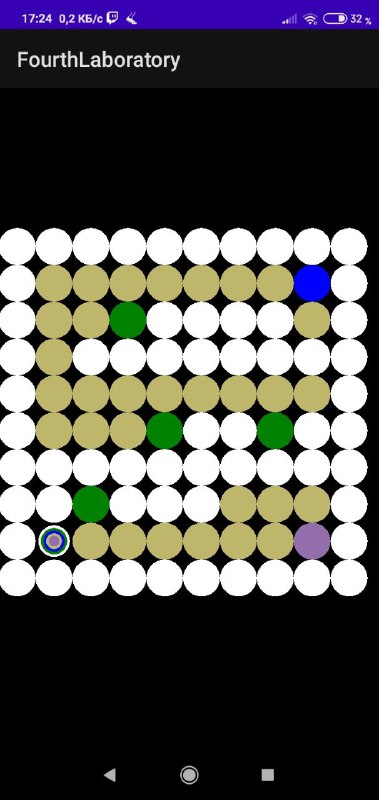
**Тестирование:**

****

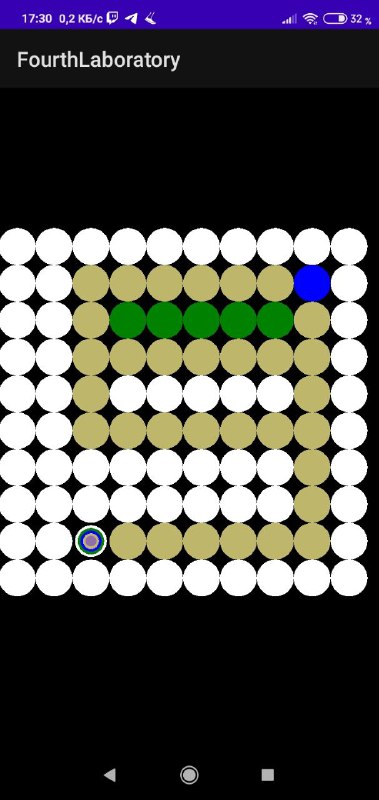
**Стартовое изображение**

****

**1 уровень**

****

**2 уровень**

****

**3 уровень**

**Вывод о проделанной работе:**

Работа с написанием простой игры в Android Studio выполнена успешно, программа работает корректно.