МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ

КАЛИНИНГРАДСКОЙ ОБЛАСТИ

ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ

«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СТРОИТЕЛЬСТВА»

(ГБУ КО ПОО «КИТиС»)

|  |  |  |
| --- | --- | --- |
| Курсовой проект  допущен к защите  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата)  Зам.директора по УМР  \_\_\_\_\_\_\_\_\_\_\_\_\_Павленко Г.Я.  (подпись) |  | Курсовой проект  защищен с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  Руководитель проект  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Пояснительная записка к курсовому проекту

по дисциплине: МДК 01.01 Разработка программных модулей

Тема: Разработка игры “Змейка” под управлением ОС Android

Специальность 09.02.07 «Информационные системы и программирование»

|  |  |
| --- | --- |
| Выполнил: | студент 3 курса,  группы ИСп19-2к  Магомедов Артур Арсенович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Руководитель: | Большакова-Стрекалова Анна Викторовна, преподаватель ГБУ КО ПОО КИТиС  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Калининград

2022

Содержание

[Введение](#_Toc101951928) 3

[1 Описание предмотной области](#_Toc101951929) 4

1.1 Аналоги разрабатываемого приложения  [4](#_Toc101951930)

[1.2 Техническое задание 7](#_Toc101951931)

[1.3 Описание структуры приложения 8](#_Toc101951932)

[2 Описание разработки приложения 9](#_Toc101951934)

[2.1 Обоснование средств разработки](#_Toc101951935) 9

[2.2 Разработка интерфейса 1](#_Toc101951936)4

[2.3Разработка логики работы приложения. Схема взаимодействия компонентов проекта](#_Toc101951938) 16

[2.4Описание переменных, компонентов, классов и подпрограмм](#_Toc101951938) 17

[3 Тестирование и установка приложения](#_Toc101951934) 18

[Заключение](#_Toc101951934) 22

[Список использованной литературы](#_Toc101951934) 23

[Приложение А. Листинг рабочего кода с подробными комментариями](#_Toc101951934) 24

## Введение

С развитием цифровых технологий компьютеры все больше вливаются в жизнь человека. Если раньше ЭВМ использовались исключительно для сложных математических вычислений, то сегодня сфера их применения существенно расширилась. Компьютерные игры - одно из наиболее массовых применений электронных вычислительных машин.

Развитие игровой индустрии шло стремительным темпом, и особенно пользовалось популярностью у подростков. Первые игры отличались простотой интерфейса и логики, но со временем они становились все сложнее и сложнее, над их созданием работал уже не один человек, а целая команда разработчиков.

Современные игры требуют достаточно большой производительности от телефона, и не каждый смартфон в силах воспроизводить требовательные игры. Однако для отдыха от монотонной работы зачастую достаточно простой, не требовательной к технике, игры. Именно такой разработке посвящен данный курсовой проект - игра «Змейка».

Впервые эта игра возникла еще в 1977 году, но и по сей день, она пользуется популярностью, особенно на мобильных платформах. Игрок управляет длинным, тонким существом, напоминающим змею, которое ползает по плоскости (как правило, ограниченной стенками), собирая еду (или другие предметы), избегая столкновения с собственным хвостом и краями игрового поля.

## Описание предметной области

Основная роль игры в змейку заключается в поедании яблок с необходимостью обползать свой хвост и не врезаться в стенки, в игре один режим с подсчётом очков и записью рекорда

## Аналоги разрабатываемого приложения

Был проведен анализ похожих существующих приложений.~~~~

Рисунок 1 - Blocky Snake

Результаты сравнения представлены в таблице 1.

Таблица 1 - Таблица сравнения

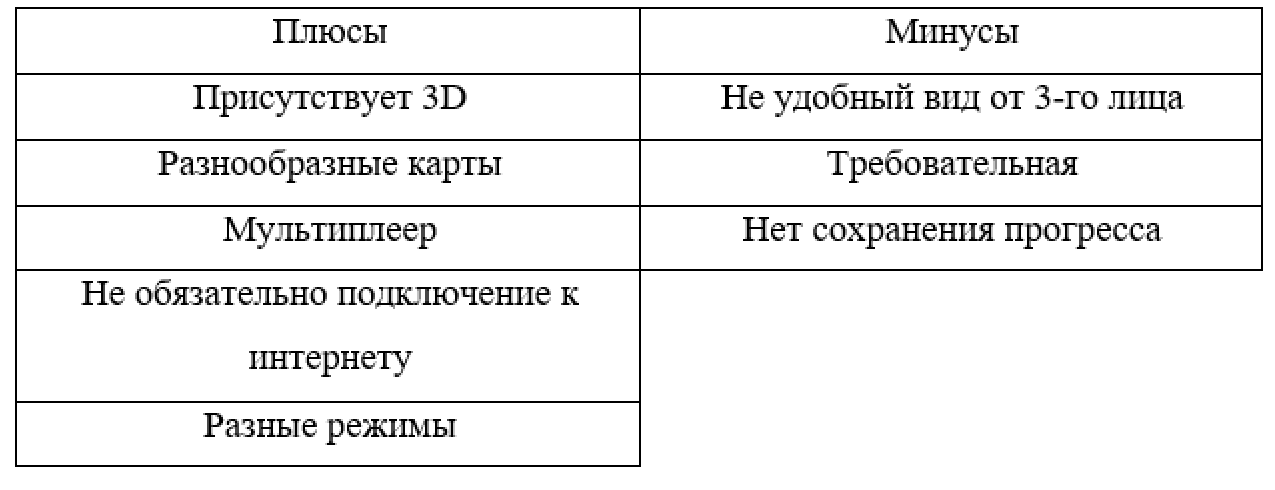




Рисунок 2 – Venom Angry Crashy Rush Online

Результаты сравнения представлены в таблице 2.

Таблица 2 - Таблица сравнения

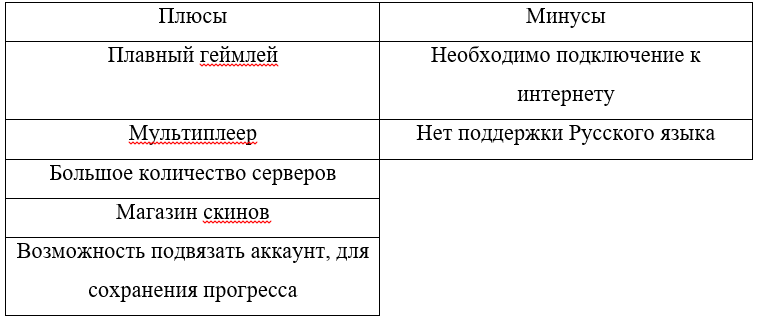




Рисунок 3 – Classic nokia snake game

Результаты сравнения представлены в таблице 3.

Таблица 3 - Таблица сравнения”

## 1.2 Техническое задание

Игра «Змейка» для телефона. Данный программный продукт представляет из себя Android-приложение для телефонов.

1. Область применения

Использование рядовыми пользователями телефонов на Android в развлекательных, а так же в развивающих целях.

1. Основание для разработки

Игра «Змейка» для телефона предназначена для игрового времяпрепровождения. Во время, которого пользователям представляется возможность управлять в Android-приложении змеёй, которая поедает яблоки.

1. Функциями приложения являются:

- Игровое времяпрепровождение;

- Предоставление инструкций к игре;

- Статистические данные;

- Развитие навыков реакции и логики.

1. Требования к программе

Игра «Змейка» для телефона должна выполнять следующие функции:

- Вести статистику рекордов;

- Воспроизводить звуки съедания яблок и смерти;

- Останавливать игру при соприкосновении с хвостом или стенами;

- Развивать реакцию пользователя.

1. Функциональные требования:

* Возможность управления змейкой свайпами;
* Работать на Android;
* Реализовать вывод рекорда/ прогресса.

## 1.3 Описание структуры приложения

При заходе в приложение перед пользователем должно открываться окно с написанным рекордом который он набрал играя ранее, если же приложение открыто в первые, рекорд будет равен нулю.

Дабы начать игры пользователю необходимо нажать на кнопку старт для начала игры.

Задача игрока набрать максимальное количество очков поедая яблоки а так же попытаться не коснуться стенки и своего хвоста или же это приведёт к окончанию игры.

## Описание разработки приложения

### 2.1 Обоснование средств разработки

Выбор операционной системы:

Для iOS необходимо владение Swift и Objective-C. Для Android – Kotlin или Java. Android лидирует среди всех операционных систем на мировом рынке.

Android:

Преимущества для разработчиков:

* Для работы подойдет любой компьютер;
* Большое сообщество разработчиков;
* Чзык программирования Java может быть использован при написании не только мобильных, но и веб-приложений, игр;
* На Android работают не только приложения для смартфонов, но и для умных часов, Smart TV, VR/AR;
* Среда разработки анализирует код и может автоматически формировать некоторые его части;
* Хороший фреймворк; можно найти готовые модули под самые разные цели;

Платформу можно адаптировать под любую специализированную задачу;

* Исходный код Android открыт для всех; есть платформа Android Open Source Project, где компания публикует все данные, и разработчики тоже могут это делать;
* Публикация приложений в Google Play происходит практически мгновенно, обычно без участия модераторов – не нужно долго ждать подтверждения;
* Небольшая плата за размещение приложений в Google Play ($25 один раз, при регистрации).

Недостатки:

* Многообразие устройств и их параметров создает почву для багов, неодинаковой работы приложения на разных смартфонах;
* Многообразие версий операционной системы: в отличие от iOS, у Android остаются актуальными и могут работать без обновлений устаревшие версии;
* Не все обучающие материалы и ответы на форумах актуальны: система существует давно, нюансы разработки меняются;
* Приложение и аккаунт могут быть заблокированы без объяснения причин, на жалобы в техподдержку отвечает бот.

iOS:

Преимущества для разработчиков:

* Строго ограниченное количество версий смартфона, размеров экрана и самой операционной системы;
* Все изменения в ОС прозрачны: Apple ежегодно выпускает гайдлайны по разработке приложений для iOS и объясняет, как заменить элементы кода на более производительные;
* Приложений и разработчиков меньше, конкуренция ниже.

Недостатки:

* Для работы необходима дорогостоящая техника;
* Чтобы загружать приложения в AppStore, нужно ежегодно продлевать аккаунт разработчика за $99;
* Публикация приложения может занять больше недели, потому что модерация производится вручную.

Вывод: Андройд лучше по причинам перечисленным выше.

Выбор среды разработки:

1. Eclipse

Eclipse – это бесплатная среда разработки от некоммерческой организации Eclipse Foundation.

Благодаря активному развитию, а также поддержке со стороны компании и сторонних разработчиков, на данный момент у этой IDE имеются следующие преимущества:

* Официальная русификация интерфейса и документации;
* Отличная производительность на слабых машинах;
* Большое число дополнений (например, для работы с сервером, базой данных и т. д.);
* Возможность подключения модулей (об этом было сказано выше);
* Возможность групповой разработки.

Eclipse была очень популярна несколько лет назад и считалась монополистом на рынке IDE для Android. Однако в связи с выходом Android Studio, в 2014 г. Google перестала поддерживать Eclipse как основную среду для разработки приложений под Android.

1. IntelliJ IDEA

Разработкой данной среды программирования занимается отечественная компания JetBrains.

Если рассматривать программирование под Android между IntelliJ IDEA и Eclipse, то первый вариант предпочтительнее, т. к. у этой среды имеются неоспоримые преимущества относительно своего конкурента:

* Более быстрая отладка значений;
* Автозаполнение методов (также реализовано в Eclipse, но пока в тестовом варианте);
* Наличие рефакторинга (автоматического подбора значений);
* Более удобный интерфейс;
* Отлично подходит для программирования на Java.

  Главный недостаток – это наличие платной версии.

1. Android Studio

Видя возрастающую популярность своей системы, Google не могли оставаться в стороне, поэтому принялись за создание официальной среды разработки под Android.

В настоящий момент Android Studio – это официальная среда разработки под Android. Конечно, некоторые программисты остаются верны Eclipse или IntelliJ IDEA, но у них есть огромный опыт в программировании. Новичкам же дико рекомендуется использовать официальную IDE – Android Studio.

Какую же среду выбрать?

Использовать Eclipse лучше в следующих случаях:

* ПК не обладает хорошей мощностью (например, имеет всего 1 ГБ оперативной памяти);
* Программы будут создаваться на нескольких языках программирования;
* Будущий разработчик совершенно незнаком с английским языком.

Стоит сказать, что последний пункт довольно спорный. Любому программисту придётся иметь дело с иностранной документацией, поэтому использование русскоязычных программ – это не выход. Гораздо эффективнее пользоваться программами на английском языке. Постоянное использование позволит сильно повысить уровень знания иностранного языка.

Теперь перейдём к IntelliJ IDEA. Она отлично подойдёт если:

* Разработка ведётся на нескольких языках программирования;
* Компьютер достаточно мощный (минимум 2 ГБ оперативной памяти).

И, наконец, Android Studio. Её следует использовать если:

* ПК обладает достаточной мощностью (минимум 2 ГБ оперативной памяти);
* Разработка будет осуществляться только под ОС Android;
* Вы не обладаете достаточным опытом программирования.

Выбор языка программирования:

Kotlin — молодой язык, и неизвестно, что будет дальше, в то время как Java характеризуется кроссплатформенностью: на ней не только мобильная разработка держится, но и бэкенд с десктопом.

Сейчас Kotlin развивается достаточно предсказуемо и ориентирован именно на Android-разработку. Его всё чаще выбирают молодые компании, которые берут старт в создании мобильных приложений, и сейчас наблюдается нехватка Kotlin-разработчиков, а потому язык весьма востребован. Но вряд ли в обозримом будущем он заменит Java.

Вывод:

### 2*.*2 Разработка интерфейса

Пользовательский интерфейс - это средства общения пользователя с программой. которые могут включать в себя изображения, звуки и текст. Ориентируясь на среднего пользователя интерфейс должен быть простым и удобным. Это снизит вероятность ошибок.

Интерфейс состоит из 5-х компонентов image, с 1-й кнопкой button, 2 компонентами text. Картинки в интерфейсе:grass, grass03, apple, snake, swipe.

Image: grass - png Рисунок травы, grass03 - вторая Рисунок травы (т.к игровое поле в клетках добавлена вторая png для точечного перемещения по клеточкам)

Рисунок 4 - Трава

Apple - png название даёт понять (то что съедает наш ползучий кусок бамбука)

Рисунок 5 - Яблоко

Snake - наш синий дружок, персонаж которым мы управляем на протяжении всей игры.



Рисунок 6 - Змея в разрезе

Swipe - png Рисунок руки которая даёт понять что нужно провести рукой по полю что бы дать понять куда ползти нашему другу.



Рисунок 7 - Палец

Button - кнопка которую нужно нажать для начала игры.

Рисунок 8 - Кнопка

Text: 1 text нужен для показа съеденных яблок, второй же сохраняет максимальное количество яблок съеденных пользователем за игру.

Принцип работы игры состоит в том, что при свайпе в определённую сторону наш змей поползёт куда ему приказали, и если это будет яблоко то он его сьест я и получит “очко” в свой счётчик съеденных им яблок. необходимо набрать максимально большое количество очков для личного рекорда или что бы просто хорошо провести время.

### 2.3 Разработка логики работы приложения. Схема взаимодействия компонентов проекта

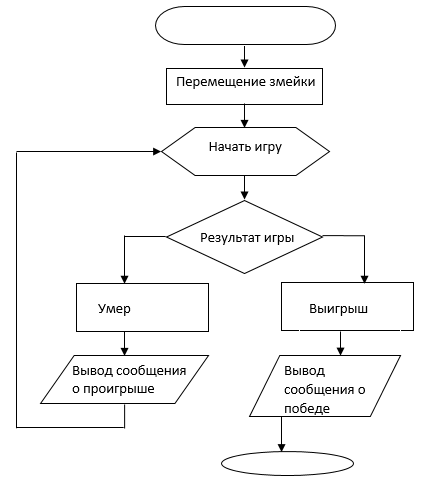


Рисунок 9 - Схема работы приложения

### 2.4 Описание переменных, компонентов, классов и подпрограмм

Таблица 4 - Методы

|  |  |
| --- | --- |
| Методы | Описание |
| Draw | Прорисовка холста. |
| onLoadComplete | Загрузка звука. |
| gameOver | Окончание игры. |
| reset | Перезапуск игры. |
| dialogScore | Форма рекорда |
| onCreate | Привязка к активностям. |
| Update | Обновление до последний версии java |

Таблица 5 – Классы

|  |  |
| --- | --- |
| Классы | Описание |
| public class Apple | Класс яблока |
| public class Constants | Неизменяемые переменные |
| public class Grass | Класс травы |
| public class PartSnake | Класс тела змеи |
| public class Snake | Класс змеи |

## 3 Тестирование и установка приложения

Запускаем проект, и даем разрешение телефону.

Тестирование приложение.

Автоматическое тестирование приложений для Android

Принцип автоматического тестирования приложений

Задача состоит в том, чтобы с наибольшей точностью автоматизировать действия, которые будут выполняться вручную. Будем использовать приложения и устройства с Android. Для каждого приложения и каждого устройства нужно выполнить следующие действия:

Установить приложение на устройство

1. Запустить приложение
2. Протестировать приложение, используя выбранный метод
3. Удалить приложение
4. Сбросить устройство в исходное состояние
5. На каждом этапе нужно собирать и анализировать данные (журналы и снимки экрана).

Управление устройствами Android

Методы автоматического тестирования Android

Тестирование с помощью Monkey\*

Допустим, что устройство с приложением оказалось в руках обезьяны: программа Monkey представляет эту такую ситуацию. Monkey, входящая в состав Android SDK, отправляет поток случайных действий пользователя. В командной строке можно указать количество действий пользователя, долю действий каждого типа и имя пакета

Недостатки тестирования с помощью Monkey:

* Monkey не может имитировать сложные нагрузки, такие как проверка подлинности. В таких случаях функциональность приложений остается непротестированной;
* Игры со сложным управлением, требующие быстрой реакции пользователей и сложных жестов, будут выполнены с самого начала или же вовсе не запустятся;
* Крайне сложно воспроизводить ошибки, обнаруженные с помощью Monkey;
* Monkey не проверяет состояние приложения во время тестирования.

Автоматическое тестирование с помощью Monkey можно считать неплохой начальной проверкой для любого приложения. Этот метод может дать достаточно полезные результаты для определенного приложения. Но при низком качестве тестирования следует использовать другие методы.

Так же в проекте были с нуля созданы интерфейс и дизайн приложения согласно техническому заданию.

Из этого следует, что создание мобильного приложения процесс трудоемкий и затрачивает он очень много времени из-за необходимости тестов и перестройки модели приложения.

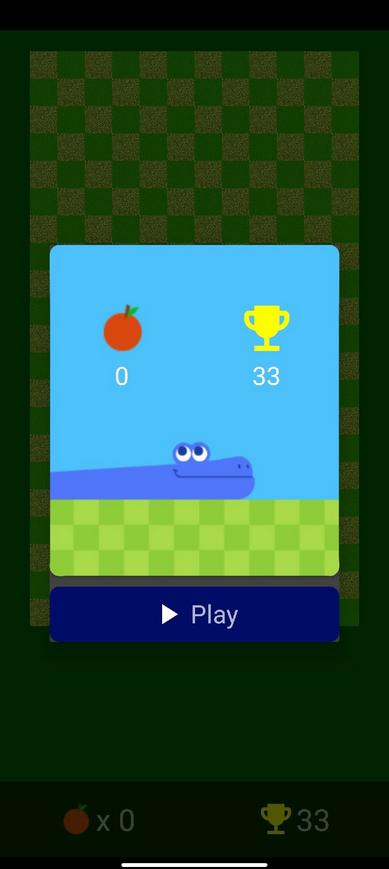


Рисунок 9. Запуск игры

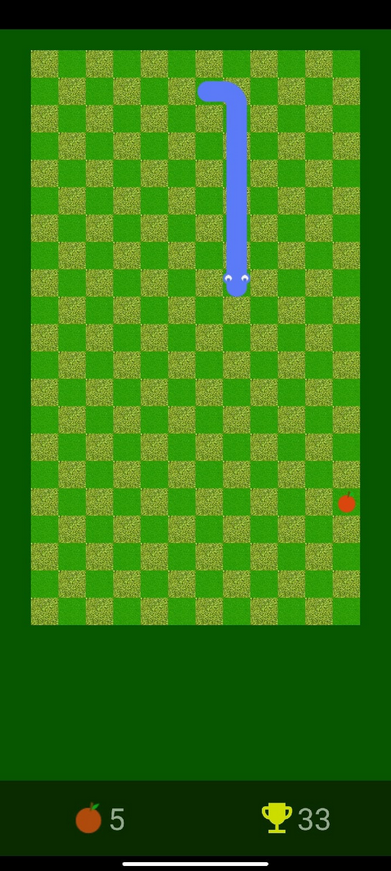


Рисунок 10. Проверка на работоспособность

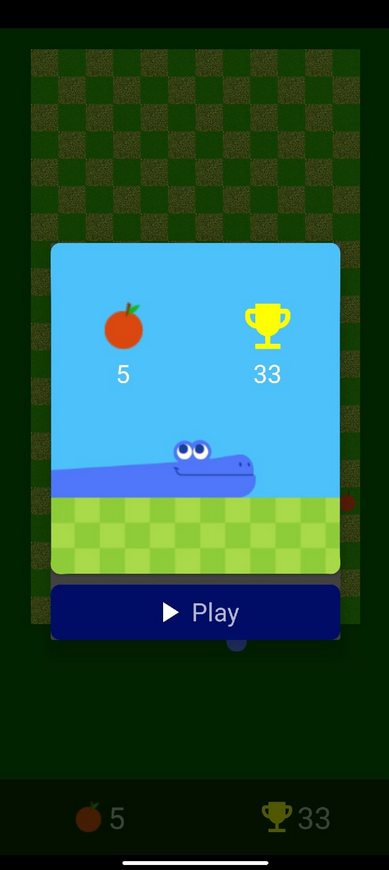


Рисунок 11. Смерть змейки

## Заключение

В ходе выполнения данной работы на основе различных источников данных была проанализирована заданная предметная область.

В процессе кодирования была изучена Java в части пользовательского интерфейса и технологий отображения графики.

В результате выполнения проекта была полностью реализована игра «Змейка»,

Были выполнены следующие задачи:

* Поворот змеи путём свайпа в определённую сторону экрана;
* Прибавление очка к рекорду при поедании яблока;
* Увеличение хвоста змеи;
* Смерть при соприкосновении змеи с своим телом или стенами;
* При смерти змеи вылезала форма с очками набранными во время игры.

В процессе тестирования приложения ошибок не обнаружено.

## Список использованной литературы

1. М. Федотенко «Разработка мобильных приложений. Первые шаги»

2. К. Марсикано, К. Стюарт, Б. Филлипс, Б. Харди «Android. Программирование для профессионалов»

3. Эд Бурнет «Привет, Android! Разработка мобильных приложений»

4. Андерс Ерансон «Эффективное использование потоков в операционной системе Android»

5. Герберт Шилдт «Java: Полное руководство»

6. Разработка игры "Змейка" на Java

7. «Изучаем Java», Кэти Сьерра, Берт Бейтс;

8. «Java. Руководство для начинающих», Герберт Шилдт*;*

9.«Java. Библиотека профессионала», Кей Хорстманн.

10. Java. Эффективное программирование Книга, Джошуа Блох

## Приложения

Приложение А. Листинг кода

Код Apple:

package com.example.snake;

import android.graphics.Bitmap; /\*для работы с Рисунокми.\*/

import android.graphics.Canvas; /\* предоставляет методы для рисования, которые отображают графические примитивы на исходном растровом изображении. \*/

import android.graphics.Rect; /\*это класс из API Android Draw\*/

public class Apple { /\*публичный, общедоступный класс\*/

private Bitmap bm; /\*закрытый класс\*/

private int x, y; /\*закрытый класс, оси\*/

private Rect r; /\*закрытый класс\*/

public Apple(Bitmap bm, int x, int y) { /\*публичный, общедоступный класс\*/

this.bm = bm; /\*используется в теле любого метода для ссылки на текущий объект.\*/

this.x = x; /\*используется в теле любого метода для ссылки на текущий объект.\*/

this.y = y; /\*используется в теле любого метода для ссылки на текущий объект.\*/

}

public void draw(Canvas canvas){ /\*означает, что метод не возвращает никаких данных.\*/

canvas.drawBitmap(bm, x, y, null); /\*позволяет перемещать изображение на холсте с одной точки на другую в любом направлении\*/

}

public void reset(int nx, int ny){ /\*Увеличение стороны прямоугольника\*/

this.x = nx; /\*используется в теле любого метода для ссылки на текущий объект.\*/

this.y = ny; /\*используется в теле любого метода для ссылки на текущий объект.\*/

}

public Bitmap getBm() { /\*публичный, общедоступный класс\*/

return bm; /\*Возвращает bm\*/

}

public void setBm(Bitmap bm) { /\*метод не возвращает никаких данных.\*/

this.bm = bm; /\*Ссылка на текущий объект\*/

}

public int getX() { /\*публичный, общедоступный класс\*/

return x; /\*Возвращает x\*/

}

public void setX(int x) { /\*метод не возвращает никаких данных.\*/

this.x = x; /\*Ссылка на текущий объект\*/

}

public int getY() { /\*публичный, общедоступный класс\*/

return y; /\*Возвращает y\*/

}

public void setY(int y) { /\*метод не возвращает никаких данных.\*/

this.y = y; /\*Ссылка на текущий объект\*/

}

public Rect getR() { /\*публичный, общедоступный класс API Android Draw\*/

return new Rect(this.x, this.y, this.x+GameView.sizeElementMap, this.y+GameView.sizeElementMap); /\*Возвращает r\*/

}

public void setR(Rect r) { /\*метод не возвращает никаких данных.\*/

this.r = r; /\*Ссылка на текущий объект\*/

}

}

Код GameView:

package com.example.snake;

import android.content.Context; /\*это объект, который предоставляет доступ к базовым функциям приложения: доступ к ресурсам, к файловой системе, вызов активности и т.д.\*/

import android.content.SharedPreferences; /\*для получения доступа к настройкам в коде приложения\*/

import android.graphics.Bitmap; /\*для работы с Рисунокми.\*/

import android.graphics.BitmapFactory; /\*Создает объекты Bitmap из различных источников,\*/

import android.graphics.Canvas; /\* предоставляет методы для рисования, которые отображают графические примитивы на исходном растровом изображении. \*/

import android.graphics.Color; /\*содержит несколько констант и методов для конвертации и извлечения цветового компонента.\*/

import android.graphics.Paint; /\*Класс Paint используется с холстом для рисования объектов. Он содержит информацию о цвете и стиле.\*/

import android.graphics.Rect; /\*это класс из API Android Draw\*/

import android.media.AudioAttributes; /\*Воспроизведение звуковых эффектов\*/

import android.media.AudioManager; /\*Воспроизведение звуковых эффектов\*/

import java.util.ArrayList; /\*это класс, позволяющий создавать списки, входит в состав библиотеки java. util.\*/

import java.util.Random; /\*генератор случайных чисел.\*/

public class GameView extends View { /\*публичный, общедоступный класс, который создает поле для игры и управляет ей наследуется от View\*/

private Bitmap bmGrass1, bmGrass2, bmSnake1, bmApple; /\*закрытый класс, рисует на карте данные обьекты\*/

private ArrayList<Grass> arrGrass = new ArrayList<>(); /\*\*/

private int w = 12, h=21; /\*закрытый класс, выставляет размер\*/

public static int sizeElementMap = 75\*Constants.SCREEN\_WIDTH/1080; /\*публичный, общедоступный класс, выставляет разрешение экрана\*/

private Snake snake; /\*закрытый класс, змея\*/

private SoundPool soundPool; /\*закрытый класс, тип переменной | название\*/

public GameView(Context context, @Nullable AttributeSet attrs) { /\*публичный, общедоступный класс, \*/

super(context, attrs); /\*используется тогда, когда подклассу требуется сослаться на его непосредственный супер класс, интерфейс предоставляющий глобальную информацию о среде приложения, это еще один термин для поля. Обычно это общедоступная константа или общедоступная переменная, к которой можно получить прямой доступ. \*/

this.context = context; /\*используется в теле любого метода для ссылки на текущий объект, интерфейс предоставляющий глобальную информацию о среде приложения, это еще один термин для поля.\*/

SharedPreferences sp = context.getSharedPreferences("gamesetting", Context.MODE\_PRIVATE); /\*для получения доступа к настройкам в коде приложения\*/

if(sp!=null){ /\*Оператор, литерал, похожий на true и false.\*/

if(Build.VERSION.SDK\_INT>=21){ /\*Установка версии SDK\*/

AudioAttributes audioAttributes = new AudioAttributes.Builder() /\*Установка версии SDK\*/

.setUsage(AudioAttributes.USAGE\_GAME) /\*Установка версии SDK\*/

.setContentType(AudioAttributes.CONTENT\_TYPE\_SONIFICATION) /\*Установка версии SDK\*/

.build(); /\*Установка версии SDK\*/

SoundPool.Builder builder = new SoundPool.Builder(); /\*нужен для массивов и ссылок на объекты\*/

builder.setAudioAttributes(audioAttributes).setMaxStreams(5); /\*Настройка библиотеки звука\*/

this.soundPool = builder.build(); /\*Настройка звука\*/

}else{ /\*Оператор если\*/

soundPool = new SoundPool(5, AudioManager.STREAM\_MUSIC, 0); /\*Настройка звука\*/

}

this.soundPool.setOnLoadCompleteListener(new SoundPool.OnLoadCompleteListener() { /\*используется в теле любого метода для ссылки на текущий объект\*/

@Override /\*это возможность реализовать метод так, чтобы он имел идентичную сигнатуру с методом класса-предка\*/

public void onLoadComplete(SoundPool soundPool, int sampleId, int status) { /\* Публичный обьект, метод не возвращает никаких данных\*/

loadedsound = true; /\*Запуск звука\*/

}

});

soundEat = this.soundPool.load(context, R.raw.eat, 1); /\*запуск звука съедания яблок\*/

soundDie = this.soundPool.load(context, R.raw.die, 1); /\*запуск звука смерти\*/

}

private int[] randomApple(){ /\*Рандомизация спавна яблока\*/

int []xy = new int[2]; /\*Рандомизация спавна яблока\*/

Random r = new Random(); /\*Рандомизация спавна яблока\*/

xy[0] = r.nextInt(arrGrass.size()-1); /\*Рандомизация спавна яблока\*/

xy[1] = r.nextInt(arrGrass.size()-1); /\*Рандомизация спавна яблока\*/

Rect rect = new Rect(arrGrass.get(xy[0]).getX(), arrGrass.get(xy[1]).getY(), arrGrass.get(xy[0]).getX()+sizeElementMap, arrGrass.get(xy[1]).getY()+sizeElementMap); /\*Рандомизация спавна яблока\*/

boolean check = true; /\*Рандомизация спавна яблока\*/

while (check){ /\*Рандомизация спавна яблока\*/

check = false; /\*Рандомизация спавна яблока\*/

for (int i = 0; i < snake.getArrPartSnake().size(); i++){ /\*Рандомизация спавна яблока\*/

if(rect.intersect(snake.getArrPartSnake().get(i).getrBody())){ /\*Рандомизация спавна яблока\*/

check = true; /\*Рандомизация спавна яблока\*/

xy[0] = r.nextInt(arrGrass.size()-1); /\*Рандомизация спавна яблока\*/

xy[1] = r.nextInt(arrGrass.size()-1); /\*Рандомизация спавна яблока\*/

rect = new Rect(arrGrass.get(xy[0]).getX(), arrGrass.get(xy[1]).getY(), arrGrass.get(xy[0]).getX()+sizeElementMap, arrGrass.get(xy[1]).getY()+sizeElementMap); /\*Рандомизация спавна яблока\*/

}

}

}

return xy; /\*Вохврат к xy\*/

}

@Override/\*это возможность реализовать метод так, чтобы он имел идентичную сигнатуру с методом класса-предка\*/

public boolean onTouchEvent(MotionEvent event) { /\*Код для движения змеи\*/

if(score > bestScore){ /\*если очков больше чем рекорда\*/

bestScore = score; /\*рекорд становится набранными очками\*/

SharedPreferences sp = context.getSharedPreferences("gamesetting", Context.MODE\_PRIVATE); /\*настройка кода в приложении\*/

SharedPreferences.Editor editor = sp.edit(); /\*Получение доступа к коду\*/

editor.putInt("bestscore", bestScore); /\*Изменение числа во время игры\*/

editor.apply(); /\*Примимение\*/

MainActivity.txt\_best\_score.setText(bestScore+""); /\*Обновление рекорда\*/

}

}

handler.postDelayed(r, 100); /\*вызов функции объекта, которая принимает 2 аргумента (в скобках)\*/

}

private void gameOver() { /\*Публичный класс, окончание игры\*/

isPlaying = false; /\*проверка игры\*/

MainActivity.dialogScore.show(); /\*появление окна с очками\*/

MainActivity.txt\_dialog\_best\_score.setText(bestScore+""); /\*Лучший рекорд\*/

MainActivity.txt\_dialog\_score.setText(score+""); /\*рекорд\*/

if(loadedsound){ /\*хагрузка звука\*/

int streamId = this.soundPool.play(this.soundDie, (float)0.5, (float)0.5, 1, 0, 1f); /\*звук смерти\*/

}

}

public void reset(){ /\*Публичный класс, перезапуск игры\*/

for(int i = 0; i < h; i++){ /\*код расставления травы на экране\*/

for (int j = 0; j < w; j++){ /\*код расставления травы на экране\*/

if((j+i)%2==0){ /\*код расставления травы на экране\*/

arrGrass.add(new Grass(bmGrass1, j\*bmGrass1.getWidth() + Constants.SCREEN\_WIDTH/2 - (w/2)\*bmGrass1.getWidth(), i\*bmGrass1.getHeight()+50\*Constants.SCREEN\_HEIGHT/1920, bmGrass1.getWidth(), bmGrass1.getHeight())); /\*выставление травы по разрешению экрана\*/

}else{ /\*оператор или\*/

arrGrass.add(new Grass(bmGrass2, j\*bmGrass2.getWidth() + Constants.SCREEN\_WIDTH/2 - (w/2)\*bmGrass2.getWidth(), i\*bmGrass2.getHeight()+50\*Constants.SCREEN\_HEIGHT/1920, bmGrass2.getWidth(), bmGrass2.getHeight())); /\*выставление травы по разрешению экрана\*/

}

}

}

snake = new Snake(bmSnake1,arrGrass.get(126).getX(),arrGrass.get(126).getY(), 4); /\*повторное появление змейки на кординатах\*/

apple = new Apple(bmApple, arrGrass.get(randomApple()[0]).getX(), arrGrass.get(randomApple()[1]).getY()); /\*появление яблока в рандом месте\*/

score = 0; /\*Начальные очки\*/

}

}

Код Grass:

package com.example.snake;

import android.graphics.Bitmap; /\*для работы с Рисунокми.\*/

public class Grass { /\*публичный класс для травы\*/

private Bitmap bm; /\*привватный класс для работы с Рисунокми\*/

private int x, y, width, height; /\*оси и длина/ширина\*/

public Grass(Bitmap bm, int x, int y, int width, int height) { /\*публичный класс травы, для обозначения осей и длины ширины\*/

this.bm = bm; /\*используется в теле любого метода для ссылки на текущий объект\*/

this.x = x; /\*используется в теле любого метода для ссылки на текущий объект\*/

this.y = y; /\*используется в теле любого метода для ссылки на текущий объект\*/

this.width = width; /\*используется в теле любого метода для ссылки на текущий объект\*/

this.height = height; /\*используется в теле любого метода для ссылки на текущий объект\*/

}

public Bitmap getBm() { /\*публичный, общедоступный класс\*/

return bm; /\*возвращает bm\*/

}

public void setBm(Bitmap bm) {/\*метод не возвращает никаких данных.\*/

this.bm = bm; /\*Ссылка на текущий объект\*/

}

public int getX() { /\*публичный, общедоступный класс\*/

return x; /\*возвращает x\*/

}

public void setX(int x) { /\*метод не возвращает никаких данных.\*/

this.x = x; /\*Ссылка на текущий объект\*/

}

public int getY() { /\*публичный, общедоступный класс\*/

return y; /\*возвращает y\*/

}

public void setY(int y) { /\*метод не возвращает никаких данных.\*/

this.y = y; /\*Ссылка на текущий объект\*/

}

rl\_start.setOnClickListener(new View.OnClickListener() { /\*начинатьигру при закрытии меню\*/

@Override /\*это возможность реализовать метод так, чтобы он имел идентичную сигнатуру с методом класса-предка\*/

public void onClick(View v) { /\* Публичный обьект, метод не возвращает никаких данных\*/

img\_swipe.setVisibility(View.VISIBLE); /\*показывать форму\*/

gv.reset(); /\*метод, который сбрасывает переменные, сохраняя сумму в ноль.\*/

dialogScore.dismiss(); /\*отменяет форму рекорда\*/

}

});

dialogScore.show(); /\*показывает форму рекорда\*/

}

}

Код PartSnake:

package com.example.snake;

import android.graphics.Bitmap; /\*Для работы с Рисунокми.\*/

import android.graphics.Rect; /\*Это класс из API Android Draw\*/

public class PartSnake { /\*Аубличный класс\*/

private Bitmap bm; /\*закрытый класс\*/

private int x, y; /\*закрытый класс, оси\*/

private Rect rBody, rTop, rBottom, rRight, rLeft; /\*закрытый класс, части тела\*/

public PartSnake(Bitmap bm, int x, int y) { /\*публичный, общедоступный класс\*/

this.bm = bm; /\*используется в теле любого метода для ссылки на текущий объект.\*/

this.x = x; /\*используется в теле любого метода для ссылки на текущий объект.\*/

this.y = y; /\*используется в теле любого метода для ссылки на текущий объект.\*/

}

public Bitmap getBm() { /\*публичный, общедоступный класс\*/

return bm; /\*Возвращает bm\*/

}

public void setBm(Bitmap bm) { /\*метод не возвращает никаких данных.\*/

this.bm = bm; /\*Ссылка на текущий объект\*/

}

public int getX() { /\*публичный, общедоступный класс\*/

return x; /\*Возвращает x\*/

}

public void setX(int x) { /\*метод не возвращает никаких данных.\*/

this.x = x; /\*Ссылка на текущий объект\*/

}

public void setrBody(Rect rBody) { /\*метод не возвращает никаких данных.\*/

this.rBody = rBody; /\*Ссылка на текущий объект\*/

}

}

Код Snake:

package com.example.snake;

import android.graphics.Bitmap; /\*для работы с Рисунокми.\*/

import android.graphics.BitmapFactory; /\*Создает объекты Bitmap из различных источников,\*/

import android.graphics.Canvas; /\* предоставляет методы для рисования, которые отображают графические примитивы на исходном растровом изображении. \*/

import android.graphics.Color; /\*содержит несколько констант и методов для конвертации и извлечения цветового компонента.\*/

import android.graphics.Paint; /\*Класс Paint используется с холстом для рисования объектов. Он содержит информацию о цвете и стиле.\*/

import android.graphics.Rect; /\*это класс из API Android Draw\*/

import java.util.ArrayList; /\*это класс, позволяющий создавать списки, входит в состав библиотеки java. util.\*/

public class Snake { /\*публичный класс змеи\*/

private Bitmap bm, bm\_head\_down, bm\_head\_left, bm\_head\_right, bm\_head\_up,

bm\_body\_vertical, bm\_body\_horizontal, bm\_body\_bottom\_left, bm\_body\_bottom\_right, bm\_body\_top\_left, bm\_body\_top\_right,

bm\_tail\_up, bm\_tail\_down, bm\_tail\_right, bm\_tail\_left; /\*Закрытый класс,прорисовка обьектов\*/

private ArrayList<PartSnake> arrPartSnake = new ArrayList<>(); /\*присвоение\*/

private int length; /\*Элемент для находа размера массива\*/

private boolean move\_left, move\_right, move\_up, move\_down; /\*закрытый класс, оператор правды\*/

public Snake(Bitmap bm, int x, int y, int length) { /\*публичный класс для настройки размеров змеи\*/

this.bm = bm; /\*Присвоение\*/

this.length = length; /\*Присвоение\*/

bm\_body\_bottom\_left = Bitmap.createBitmap(bm, 0, 0, bm.getWidth()/14, bm.getHeight()); /\*настройка размеров\*/

bm\_body\_bottom\_right = Bitmap.createBitmap(bm, bm.getWidth()/14, 0, bm.getWidth()/14, bm.getHeight()); /\*настройка размеров\*/

bm\_head\_left = Bitmap.createBitmap(bm, 7\*bm.getWidth()/14, 0, bm.getWidth()/14, bm.getHeight()); /\*настройка размеров\*/

bm\_tail\_down = Bitmap.createBitmap(bm, 13\*bm.getWidth()/14, 0, bm.getWidth()/14, bm.getHeight()); /\*настройка размеров\*/

setMove\_right(true); /\*Выставление стороны змеи при спавне\*/

arrPartSnake.add(new PartSnake(bm\_head\_right, x, y)); /\*голова змеи\*/

for (int i = 1; i < length-1; i++){ /\*код для места появления\*/

this.arrPartSnake.add(new PartSnake(bm\_body\_horizontal, this.arrPartSnake.get(i-1).getX()-GameView.sizeElementMap, y)); /\*код для места появления\*/

}

arrPartSnake.add(new PartSnake(bm\_tail\_right, arrPartSnake.get(length-2).getX()-GameView.sizeElementMap, arrPartSnake.get(length-2).getY())); /\*код для места появления\*/

}

public void update(){ /\*Публичный обьект, метод не возвращает никаких данны, Функция, которая позволяет гарантировать наличие на компьютере под управлением Windows последних версий Java.\*/

for(int i = length-1; i > 0; i--){ /\*код для выставления размера змеи\*/

arrPartSnake.get(i).setX(arrPartSnake.get(i-1).getX()); /\*выставление змеи по оси\*/

arrPartSnake.get(i).setY(arrPartSnake.get(i-1).getY()); /\*выставление змеи по оси\*/

}

if(move\_right){ /\*оператор если, поворот вправо\*/

arrPartSnake.get(0).setX(arrPartSnake.get(0).getX()+GameView.sizeElementMap); /\*движение змеи при вовороте в эту сторону\*/

arrPartSnake.get(0).setBm(bm\_head\_right); /\*движение змеи при вовороте в эту сторону\*/

}else if(move\_down){ /\*оператор или если, поворот вниз\*/

arrPartSnake.get(0).setY(arrPartSnake.get(0).getY()+GameView.sizeElementMap); /\*движение змеи при вовороте в эту сторону\*/

arrPartSnake.get(0).setBm(bm\_head\_down); /\*движение змеи при вовороте в эту сторону\*/

}else{ /\*оператор если, поворот в сторону не упомянутую выше (влево)\*/

arrPartSnake.get(0).setX(arrPartSnake.get(0).getX()-GameView.sizeElementMap); /\*движение змеи при вовороте в эту сторону\*/

arrPartSnake.get(0).setBm(bm\_head\_left); /\*движение змеи при вовороте в эту сторону\*/

}

for (int i = 1; i < length - 1; i++){ /\*код для смещения хвоста\*/

if(arrPartSnake.get(i).getrLeft().intersect(arrPartSnake.get(i+1).getrBody()) /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

&&arrPartSnake.get(i).getrBottom().intersect(arrPartSnake.get(i-1).getrBody()) /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

||arrPartSnake.get(i).getrBottom().intersect(arrPartSnake.get(i+1).getrBody()) /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

}else if (arrPartSnake.get(i).getrRight().intersect(arrPartSnake.get(i+1).getrBody())

&&arrPartSnake.get(i).getrTop().intersect(arrPartSnake.get(i-1).getrBody()) /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

||arrPartSnake.get(i).getrRight().intersect(arrPartSnake.get(i-1).getrBody()) /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

&&arrPartSnake.get(i).getrTop().intersect(arrPartSnake.get(i+1).getrBody())) { /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

arrPartSnake.get(i).setBm(bm\_body\_top\_right); /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

}else if(arrPartSnake.get(i).getrRight().intersect(arrPartSnake.get(i+1).getrBody()) /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

&&arrPartSnake.get(i).getrBottom().intersect(arrPartSnake.get(i-1).getrBody()) /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

||arrPartSnake.get(i).getrRight().intersect(arrPartSnake.get(i-1).getrBody()) /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

&&arrPartSnake.get(i).getrBottom().intersect(arrPartSnake.get(i+1).getrBody())){ /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

arrPartSnake.get(i).setBm(bm\_body\_bottom\_right); /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

}else if(arrPartSnake.get(i).getrLeft().intersect(arrPartSnake.get(i-1).getrBody()) /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

&&arrPartSnake.get(i).getrRight().intersect(arrPartSnake.get(i+1).getrBody()) /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

||arrPartSnake.get(i).getrLeft().intersect(arrPartSnake.get(i+1).getrBody()) /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

&&arrPartSnake.get(i).getrRight().intersect(arrPartSnake.get(i-1).getrBody())){ /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

arrPartSnake.get(i).setBm(bm\_body\_horizontal); /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

}else if(arrPartSnake.get(i).getrTop().intersect(arrPartSnake.get(i-1).getrBody()) /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

&&arrPartSnake.get(i).getrBottom().intersect(arrPartSnake.get(i+1).getrBody()) /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

||arrPartSnake.get(i).getrTop().intersect(arrPartSnake.get(i+1).getrBody()) /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

&&arrPartSnake.get(i).getrBottom().intersect(arrPartSnake.get(i-1).getrBody())){ /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

arrPartSnake.get(i).setBm(bm\_body\_vertical); /\*если происходит поворот, змея поворачивается и увеличивает свой рамер на 1 тайл, а хвост уменьшается\*/

arrPartSnake.get(i).setBm(bm\_body\_vertical); /\*змея повёрнутая вертикально\*/

}else{ /\*оператор или, поворот влево\*/

arrPartSnake.get(i).setBm(bm\_body\_horizontal); /\*змея повёрнутая в горизонтально\*/

}

}

}

if(arrPartSnake.get(length-1).getrRight().intersect(arrPartSnake.get(length-2).getrBody())){ /\*оператор если, поворот вправо\*/

arrPartSnake.get(length-1).setBm(bm\_tail\_right);

}else if(arrPartSnake.get(length-1).getrLeft().intersect(arrPartSnake.get(length-2).getrBody())){ /\*оператор или если, поворот влево\*/

arrPartSnake.get(length-1).setBm(bm\_tail\_left);

}else if(arrPartSnake.get(length-1).getrBottom().intersect(arrPartSnake.get(length-2).getrBody())){ /\*оператор или если, поворот вниз\*/

arrPartSnake.get(length-1).setBm(bm\_tail\_down);

}else{ /\*оператор или, поворот в сторону не упомянутую выше (вверх)\*/

arrPartSnake.get(length-1).setBm(bm\_tail\_up);

}

}

public void drawSnake(Canvas canvas){ /\*общий класс\*/

for(int i = length-1; i >= 0; i--){ /\*код для части змеи\*/

canvas.drawBitmap(arrPartSnake.get(i).getBm(), arrPartSnake.get(i).getX(), arrPartSnake.get(i).getY(), null); /\*прорисовка части змеи\*/

}

}

public Bitmap getBm() { /\*публичный, общедоступный класс\*/

return bm; /\*возврат к bm\*/

}

public void setBm(Bitmap bm) { /\*метод не возвращает никаких данных.\*/

this.bm = bm; /\*Ссылка на текущий объект\*/

}

public Bitmap getBm\_head\_down() { /\*публичный, общедоступный класс\*/

return bm\_head\_down; /\*возврат к bm\_head\_down\*/

}

public Bitmap getBm\_tail\_up() { /\*публичный, общедоступный класс\*/

return bm\_tail\_up; /\*возврат к bm\_tail\_up\*/

}

public void setBm\_tail\_up(Bitmap bm\_tail\_up) { /\*метод не возвращает никаких данных.\*/

this.bm\_tail\_up = bm\_tail\_up; /\*Ссылка на текущий объект\*/

}

public Bitmap getBm\_tail\_down() { /\*публичный, общедоступный класс\*/

return bm\_tail\_down; /\*возврат к bm\_tail\_down\*/

}

public void setBm\_tail\_down(Bitmap bm\_tail\_down) { /\*метод не возвращает никаких данных.\*/

this.bm\_tail\_down = bm\_tail\_down; /\*Ссылка на текущий объект\*/

}

public Bitmap getBm\_tail\_right() { /\*публичный, общедоступный класс\*/

return bm\_tail\_right; /\*возврат к bm\_tail\_right\*/

}

public void setBm\_tail\_right(Bitmap bm\_tail\_right) { /\*метод не возвращает никаких данных.\*/

this.bm\_tail\_right = bm\_tail\_right; /\*Ссылка на текущий объект\*/

}

public Bitmap getBm\_tail\_left() { /\*публичный, общедоступный класс\*/

return bm\_tail\_left; /\*возврат к bm\_tail\_left\*/

}

public void setBm\_tail\_left(Bitmap bm\_tail\_left) { /\*метод не возвращает никаких данных.\*/

this.bm\_tail\_left = bm\_tail\_left; /\*Ссылка на текущий объект\*/

}

public ArrayList<PartSnake> getArrPartSnake() { /\*публичный класс части змеи\*/

return arrPartSnake; /\*возврат к arrPartSnake\*/

}

public void setArrPartSnake(ArrayList<PartSnake> arrPartSnake) { /\*метод не возвращает никаких данных.\*/

this.arrPartSnake = arrPartSnake; /\*Ссылка на текущий объект\*/

}

public int getLength() { /\*публичный, общедоступный класс\*/

return length; /\*возврат к length\*/

}

public void setLength(int length) { /\*метод не возвращает никаких данных.\*/

this.length = length; /\*Ссылка на текущий объект\*/

}

public boolean isMove\_left() { /\*общедоступный класс, оператор правды\*/

return move\_left; /\*возврат к move\_left\*/

}

public void setMove\_left(boolean move\_left) { /\*метод не возвращает никаких данных, оператор правды\*/

this.setup(); /\*используется в теле любого метода для ссылки на текущий объект\*/

this.move\_left = move\_left; /\*Ссылка на текущий объект, присвоение\*/

}

public boolean isMove\_right() { /\*общедоступный класс, оператор правды\*/

return move\_right; /\*возврат к move\_right\*/

}

this.arrPartSnake.add(new PartSnake(bm\_tail\_left, p.getX()+GameView.sizeElementMap, p.getY())); /\*код для змеи который увеличивает её размер при повороте в эту сторону\*/

}else if(p.getBm()==bm\_tail\_up){ /\*оператор или если, поворот вверх\*/

this.arrPartSnake.add(new PartSnake(bm\_tail\_up, p.getX(), p.getY()+GameView.sizeElementMap)); /\*код для змеи который увеличивает её размер при повороте в эту сторону\*/

}else if(p.getBm()==bm\_tail\_down){ /\*оператор если, поворот вниз\*/

this.arrPartSnake.add(new PartSnake(bm\_tail\_up, p.getX(), p.getY()-GameView.sizeElementMap)); /\*код для змеи который увеличивает её размер при повороте в эту сторону\*/

}

}

}

Приложение Б. Руководство пользователя

Руководство пользователя мобильного приложения для устройств на ОС Android

1. Требования к аппаратуре и программному обеспечению

Для устройств на базе ОС Android:

Android — версия 9.0 и старше; сенсорный экран.

2. Установка, обновление, удаление мобильного приложения

Установка мобильного приложения выполняется через Android Studio.

Для удаления мобильного приложения необходимо на мобильном устройстве через настройки -> приложения, удалить мобильное приложение.

3. Запуск мобильного приложения

Для запуска приложения нужно найти его в списке и нажать на иконку. При этом будет открыто основное окно мобильного приложения.

4. Интерфейс системы

Интерфейс мобильной платформы ориентирован на то, что в один момент времени на экране отображается одна форма. Под нее освобождается максимум свободного места на экране. Первой видимой формой является главный экран. Команды, открывающие другие экраны, находятся в главном меню приложения. Также это меню содержит команду возврата на рабочий стол, команды отображения информации о программе и перехода к списку приложений. Вызов главного меню приложения осуществляется с помощью системной кнопки вызова меню.

5.Геймплейl:

Задача игрока нажать на кнопку start и начнётся игра, свайпами необходимо усправлять змеёй которая ест яблоки и обходит свой хвост и края экрана

6.Выход из приложения

Для выхода из приложения достаточно свайпнуть по экрану согласно системе или нажать кнопку назад на нижней панели управления.

Рецензия

В работе на тему Разработка игры “Змейка” под управлением ОС Android, было разработано приложение. Проект состоит из введения, трёх глав, заключения, списка использованных источников и приложения А. Во введении кратко рассказывается о истории возникновения игры, а также отмечены: цель, задачи. В первой главе были проанализированы приложения- конкуренты, разработано техническое задание. Вторая глава посвящена практической части, где обусловлен выбор среды разработки и языка программирования А в третьей главе описывается тестирование приложения В заключении данного курсового проекта развернуто подытожена работа.

Можно сделать вывод, что в целом, автор выполнил работу по всем требованиям. Каждая часть проекта рассматривалась с применением необходимой профессиональной терминологией. Студент излагал работу с логичной последовательностью, все названия глав полностью соответствуют тексту.

https://github.com/Perun1967RUS/Snake.git