Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Глазовский государственный педагогический институт имени В.Г. Короленко»

**ОТЧЕТ**

**по учебной практике**

**ПМ 03Участие в интеграции программных модулей**

**Специальность 09.02.03 Программирование в компьютерных системах**

|  |
| --- |
| Выполнил(а): |
| студент(ка) ОП СПО группа 43 |
| Машковцев Р.М. |
|  |
| Руководитель практики: ст. преподаватель Касаткин К.А. |

г. Глазов 2019

Оглавление

[Введение 3](#_Toc4074454)

[Цель разработки 4](#_Toc4074455)

[Обзор аналогов 4](#_Toc4074456)

[Постановка задачи 4](#_Toc4074457)

[Описание алгоритма работы программы 12](#_Toc4074458)

[Исходный код программы 15](#_Toc4074459)

[Тестирование программы 21](#_Toc4074460)

[Руководство по установке 23](#_Toc4074461)

# Введение

На данный момент существует великое множество компьютерных игр, так же разнообразны подходы в их создании.

Например, популярные в последнее время небольшие игры (чаще всего многопользовательские, через сеть Интернет) создаются по технологии Flash с использованием языка ActionScript. Такие игры крайне популярны на разнообразных развлекательных интернет-ресурсах и а социальных сетях.

Большие, сложные игры с реалистичной 3D графикой пишутся на C++, отдельные модули к ним могут быть написаны на других языках (например AI - «искусственный интеллект» - на Lisp). Вообще, язык для создания игры выбирается как компромисс между требованиями к игре, к компьютерному железу и к стоимости разработки.

На подавляющем большинстве мобильных телефонов установлены ARM процессоры. Поэтому игры для мобильных телефонов на платформе Java пишутся на специальном языке для мобильных телефонов MobileBasic, а также на диалекте языка Pascal - MIDletPascal.

В рамках учебной практики необходимо разработать небольшую программу игры в «Сапёра» поле размера 9х9 клеток с применением графики на языке высокого уровня с использованием объектно-ориентированного программирования.

«Flappy Bird» - популярная игра людей всех возрастов. Она отличается простотой, способствует развитию внимания и элементарной логики. В данной работе эта игра реализована на языке Java.

# Цель разработки

Создать интересную и логическую игру, для детей и взрослых, игра будет в открытом доступе и все желающие могут воспользоваться.

# Обзор аналогов

**Alone**



Главная особенность игры Alone заключается в ее интенсивном темпе. Для того чтобы выжить, Вам нужно будет как можно быстрее преодолевать возникающие на Вашем пути препятствия. На первый взгляд, Вам может показаться, что эта игра слишком простая, однако это не так, ведь даже начальный уровень требует немалого мастерства.

**Badland**

****

Badland – это, без преувеличения, настоящий шедевр. Игра порадует Вас прекрасной графикой и забавным, но замысловатым сюжетом. Действие разворачивается в фантастическом мире, через который Вам нужно будет провести причудливое пушистое существо, всячески ограждая его от препятствий. Бонусы позволят существу увеличиваться или уменьшаться в размерах, замедляться и ускоряться, а также зарабатывать дополнительные «жизни».

**Retry**

****

Игру Retry создавали те же разработчики, которые придумали Angry Birds. Стоит отметить, что в прошлом году игра входила в список лучших приложений. Ваша задача – постараться выжить, управляя самолетом и выделывая фигуры высшего пилотажа.

**Jetpack Joyride**

****

Разработчиком Jetpack Joyride является Halfbrick Studios, знакомая многим по игре Fruit Ninja. Главная цель данной игры — продвинуться как можно дальше и при этом уцелеть. В игре можно (и нужно) зарабатывать бонусы, которые помогут собирать монеты, используемые для покупки различных предметов.

**Jupiter Jump**

****

Главный герой игры Jupiter Jump – космонавт, которой прыгает и обходит мины. В этом и заключается смысл, однако не стоит думать, что обходить мины – это легкая задача. По началу Ваш герой будет гибнуть, причем часто.

**Flop Rocket**

****

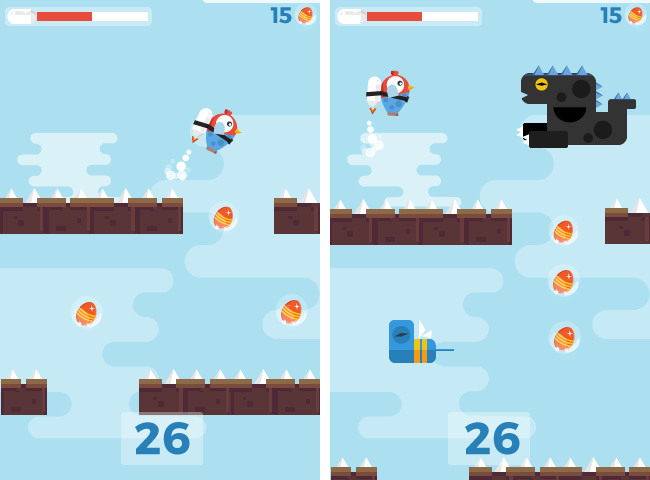
Правила игры Flop Rocket чрезвычайно простые – нужно обходить препятствия, избегать врагов, собирать бонусы и всегда держать топливный бак полным. На первый взгляд, все очень просто, но очень не многим удается освоить управление ракетой с первого раза. Подсказка: успешно пройти миссию поможет апгрейд корабля.

**Fly Catbug Fly**

****

Игра Fly Catbug Fly наверняка придется по душе тем, кто является поклонником мультсериала «Храбрейшие воины». В процессе игры Вы будете путешествовать по причудливым мирам, собирать предметы, обменивать их на наличные и покупать КотоЖуку (Catbug) необходимую утварь.

**Rocket Romeo**

****

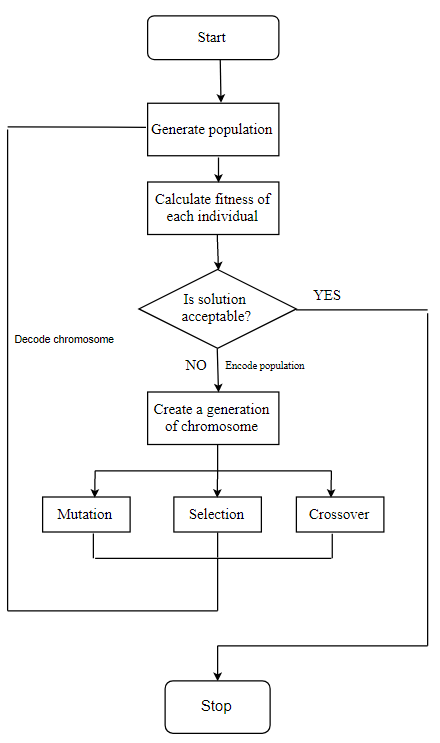
Rocket Romeo — простая, но удивительно забавная игра, где Вы должны будете использовать ракетные ускорители, чтобы избежать контакта с шипами.

# Постановка задачи

1. Написать программу, реализующую идею игры. Использовать классы и функции. Использовать графический режим.

# Описание алгоритма работы программы

1. Отрисовка игрового поля.
2. Прием команды с мышки пользователя.
3. Соответствующее действие на принятую команду.
4. Сообщение о проигрыше или выйгрыше.



# Исходный код программы

Для написания игры, я взял язык программирования Java. Так как этот язык очень удобный, и я хотел работать с ним.

# DesktopLauncher

**package** com.mygdx.game.desktop;  
  
**import** com.badlogic.gdx.backends.lwjgl.LwjglApplication;  
**import** com.badlogic.gdx.backends.lwjgl.LwjglApplicationConfiguration;  
**import** com.mygdx.game.MyGdxGame;  
  
**public class** DesktopLauncher {  
 **public static void** main (String[] arg) {  
 LwjglApplicationConfiguration config = **new** LwjglApplicationConfiguration();  
 config.**width** = 800;  
 config.**height** = 600;  
 **new** LwjglApplication(**new** MyGdxGame(), config);  
 }  
}

**MyGdxGame**

**package** com.mygdx.game;  
  
**import** com.badlogic.gdx.ApplicationAdapter;  
**import** com.badlogic.gdx.Gdx;  
**import** com.badlogic.gdx.Input;  
**import** com.badlogic.gdx.graphics.GL20;  
**import** com.badlogic.gdx.graphics.Texture;  
**import** com.badlogic.gdx.graphics.g2d.SpriteBatch;  
  
**public class** MyGdxGame **extends** ApplicationAdapter {  
 SpriteBatch **batch**;  
 Background **bg**;  
 Bird **bird**;  
 Obstacles **obstacles**;  
 **boolean gameOver**;  
 Texture **restartTexture**;  
  
  
  
 @Override  
 **public void** create () {  
 **batch** = **new** SpriteBatch();  
 **bg** = **new** Background();  
 **bird** = **new** Bird();  
 **obstacles** = **new** Obstacles();  
 **gameOver** = **false**;  
 **restartTexture** = **new** Texture(**"RestartBtn.png"**);  
  
 }  
  
 @Override  
 **public void** render () {  
 update();  
 Gdx.*gl*.glClearColor(1, 1, 1, 1);  
 Gdx.*gl*.glClear(GL20.***GL\_COLOR\_BUFFER\_BIT***);  
 **batch**.begin();  
 **bg**.render(**batch**);  
 **obstacles**.render(**batch**);  
 **if**(!**gameOver**) {  
 **bird**.render(**batch**);  
 }**else**{  
 **batch**.draw(**restartTexture**, 200, 200);  
 }  
 **batch**.end();  
 }  
  
 **public void** update(){  
 **bg**.update();  
 **bird**.update();  
 **obstacles**.update();  
 **for** (**int** i = 0; i < Obstacles.*obs*.**length**; i++) {  
 **if**(**bird**.**position**.**x** > Obstacles.*obs*[i].**position**.**x** && **bird**.**position**.**x** < Obstacles.*obs*[i].**position**.**x**+50){  
 **if**(!Obstacles.*obs*[i].**emptySpace**.contains(**bird**.**position**)){  
 **gameOver** = **true**;  
 }  
 }  
 }  
 **if**(**bird**.**position**.**y** <0 || **bird**.**position**.**y** > 600){  
 **gameOver** = **true**;  
 }  
 **if**(Gdx.*input*.isKeyPressed(Input.Keys.***SPACE***) && **gameOver**){  
 recreate();  
 }  
  
  
 }  
  
 @Override  
 **public void** dispose () {  
 **batch**.dispose();  
 }  
  
 **public void** recreate(){  
 **bird**.recreate();  
 **obstacles**.recreate();  
 **gameOver** = **false**;  
 }  
}

**Background**

**package** com.mygdx.game;  
  
**import** com.badlogic.gdx.graphics.Texture;  
**import** com.badlogic.gdx.graphics.g2d.SpriteBatch;  
**import** com.badlogic.gdx.math.Vector2;  
  
**public class** Background {  
  
 **class** BGPicture{  
 **private** Texture **tx**;  
 **private** Vector2 **pos**;  
  
 **public** BGPicture(Vector2 pos){  
 **tx** = **new** Texture(**"back.png"**);  
 **this**.**pos** = pos;  
 }  
  
 }  
  
 **private int speed**;  
 **private** BGPicture[] **backs**;  
  
 **public** Background(){  
 **speed** = 4;  
 **backs** = **new** BGPicture[2];  
 **backs**[0] = **new** BGPicture(**new** Vector2(0,0));  
 **backs**[1] = **new** BGPicture(**new** Vector2(800,0));  
 }  
  
 **public void** render(SpriteBatch batch){  
 **for** (**int** i = 0; i < **backs**.**length**; i++) {  
 batch.draw(**backs**[i].**tx**, **backs**[i].**pos**.**x**, **backs**[i].**pos**.**y**);  
 }  
 }  
  
 **public void** update(){  
 **for** (**int** i = 0; i < **backs**.**length**; i++) {  
 **backs**[i].**pos**.**x** -= **speed**;  
 }  
  
 **if**(**backs**[0].**pos**.**x** < -800){  
 **backs**[0].**pos**.**x** = 0;  
 **backs**[1].**pos**.**x** = 800;  
 }  
 }  
}

**Bird**

**package** com.mygdx.game;  
  
**import** com.badlogic.gdx.Gdx;  
**import** com.badlogic.gdx.Input;  
**import** com.badlogic.gdx.graphics.Texture;  
**import** com.badlogic.gdx.graphics.g2d.SpriteBatch;  
**import** com.badlogic.gdx.math.Vector2;  
  
**public class** Bird {  
 Texture **img**;  
 Vector2 **position**;  
 **float vy**;  
 **float gravity**;  
  
  
 **public** Bird(){  
 **img** = **new** Texture(**"bird1.png"**);  
 **position** = **new** Vector2(100,380);  
 **vy** = 0;  
 **gravity** = - 0.7f;  
 }  
  
 **public void** render(SpriteBatch batch){  
 batch.draw(**img**, **position**.**x**, **position**.**y**);  
 }  
  
 **public void** update(){  
  
 **if** (Gdx.*input*.isKeyPressed(Input.Keys.***SPACE***)){  
 **vy** = 10;  
 }  
 **vy** += **gravity**;  
 **position**.**y** += **vy**;  
 }  
  
 **public void** recreate(){  
 **position** = **new** Vector2(100,380);  
 **vy** = 0;  
 }  
}

**Obstacles**

**package** com.mygdx.game;  
  
**import** com.badlogic.gdx.Gdx;  
**import** com.badlogic.gdx.Input;  
**import** com.badlogic.gdx.graphics.Texture;  
**import** com.badlogic.gdx.graphics.g2d.SpriteBatch;  
**import** com.badlogic.gdx.math.Vector2;  
  
**public class** Bird {  
 Texture **img**;  
 Vector2 **position**;  
 **float vy**;  
 **float gravity**;  
  
  
 **public** Bird(){  
 **img** = **new** Texture(**"bird1.png"**);  
 **position** = **new** Vector2(100,380);  
 **vy** = 0;  
 **gravity** = - 0.7f;  
 }  
  
 **public void** render(SpriteBatch batch){  
 batch.draw(**img**, **position**.**x**, **position**.**y**);  
 }  
  
 **public void** update(){  
  
 **if** (Gdx.*input*.isKeyPressed(Input.Keys.***SPACE***)){  
 **vy** = 10;  
 }  
 **vy** += **gravity**;  
 **position**.**y** += **vy**;  
 }  
  
 **public void** recreate(){  
 **position** = **new** Vector2(100,380);  
 **vy** = 0;  
 }  
}

# Тестирование программы

