

Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу "Разработка кроссплатформенных приложений"

на тему "Разработка многомодульного приложения на языке Java"

Выполнили студент группы 21ввв1:

Гришин С.В.

30.05.24
отлично

[Signature]

Принял к.т.н., доцент:

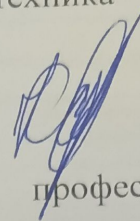
Юрова О.В.

Пенза 2024

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Вычислительной техники

Кафедра "Вычислительная техника"



"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

профессор М. А. Митрохин

« ____ » _____ 20__ г

ЗАДАНИЕ

на курсовое проектирование по курсу

Разработка кроссплатформенных приложений
Студенту Тришину Сергею Вадимовичу Группа 21ВВВ1
Тема проекта Разработка многопользовательского приложения на языке Java

Исходные данные (технические требования) на проектирование

Разработать игровое приложение "Панчики".
Язык программирования Java, среда разработки NetBeans.
Приложение должно обладать графическим интерфейсом и использовать следующие технологии:

1. Java Collections Framework
2. Механизм обработки исключительных ситуаций
3. Java Stream API
4. Java Multithreading
5. Сетевое взаимодействие

Объем работы по курсу

1. Расчетная часть

2. Графическая часть

Диаграммы UML: диаграмма вариантов использования;
диаграмма классов;
диаграмма деятельности;
диаграмма развертывания;
диаграмма последовательности

3. Экспериментальная часть

Тестирование многомодульного приложения

Срок выполнения проекта по разделам

1 В соответствии с графиком

2

3

4

5

6

7

8

Дата выдачи задания "06" 02.2024

Дата защиты проекта " " "

Руководитель Юрова О. В.

Задание получил "27" марта

Студент Гринин Сергей Вадимович

2024г.

Содержание

Содержание.....	2
Введение	3
1. Постановка задачи.....	4
2. Выбор решения	5
2.1. Пользовательский интерфейс.....	5
3. Описание разработки программы	7
3.1. Создание интерфейса	7
3.2. Организация структуры программы	8
3.3. Представление объектов на панели уровня.....	9
4. Организация сетевого взаимодействия	10
5. Отладка и тестирование	11
6. Руководство пользователя	12
Заключение.....	14
Список литературы.....	15
Приложение А. Листинг программы.....	16

Введение

Разделение приложения на модули в наши дни, похоже, является неотделимой частью разработки любого стоящего продукта. Но почему это так? Каковы преимущества для разработчика и команды в целом? Как должно выглядеть модульное приложение? И как начать разделять свое приложение на модули?

Модульность действительно помогает во многих аспектах:

1. Ускоряет сборку приложения;
2. Позволяет подгружать компоненты приложения “на лету”;
3. Упрощает разработку;
4. Позволяет повторно использовать модули в приложениях;
5. Допускает эксперименты с новыми технологиями;
6. Удобна для крупных команд разработчиков;
7. Делает рефакторинг проще;
8. Упрощает автоматизацию тестирования;

Модульность невероятно мощна, что позволяет ускорить сборку, упростить разработку и существенно расширить команду. Кроме того, она обеспечивает интересные варианты использования, такие как мгновенные приложения (instant apps), и облегчает экспериментирование с новыми технологиями.

Целью данной работы является разработка многомодульного приложения, демонстрирующего преимущества такого подхода к разработке.

1. Постановка задачи

Требуется разработать игру – “Танчики”, цель которой заключается в уничтожении базы противника. Игрок управляет танком, вид сверху. Два игрока и их базы появляются на разных половинах уровня. При запуске игра пытается подключиться к уже запущенной игре, если такой нет, то она переходит в режим ожидания подключения.

Используемый язык программирования Java, среда разработки – NetBeans.

Приложение должно использовать Java Collections Framework и Java Stream API, обрабатывать исключительные ситуации, работать в многопоточном режиме и осуществлять сетевое взаимодействие.

2. Выбор решения

2.1. Пользовательский интерфейс

Были проанализированы варианты использования программы. Диаграмма вариантов использования представлена на рисунке 1.



Рисунок 1 – UML диаграмма вариантов использования

Окно приложения будет отображать только сам уровень

Диаграмма деятельности представлена на рисунке 2.

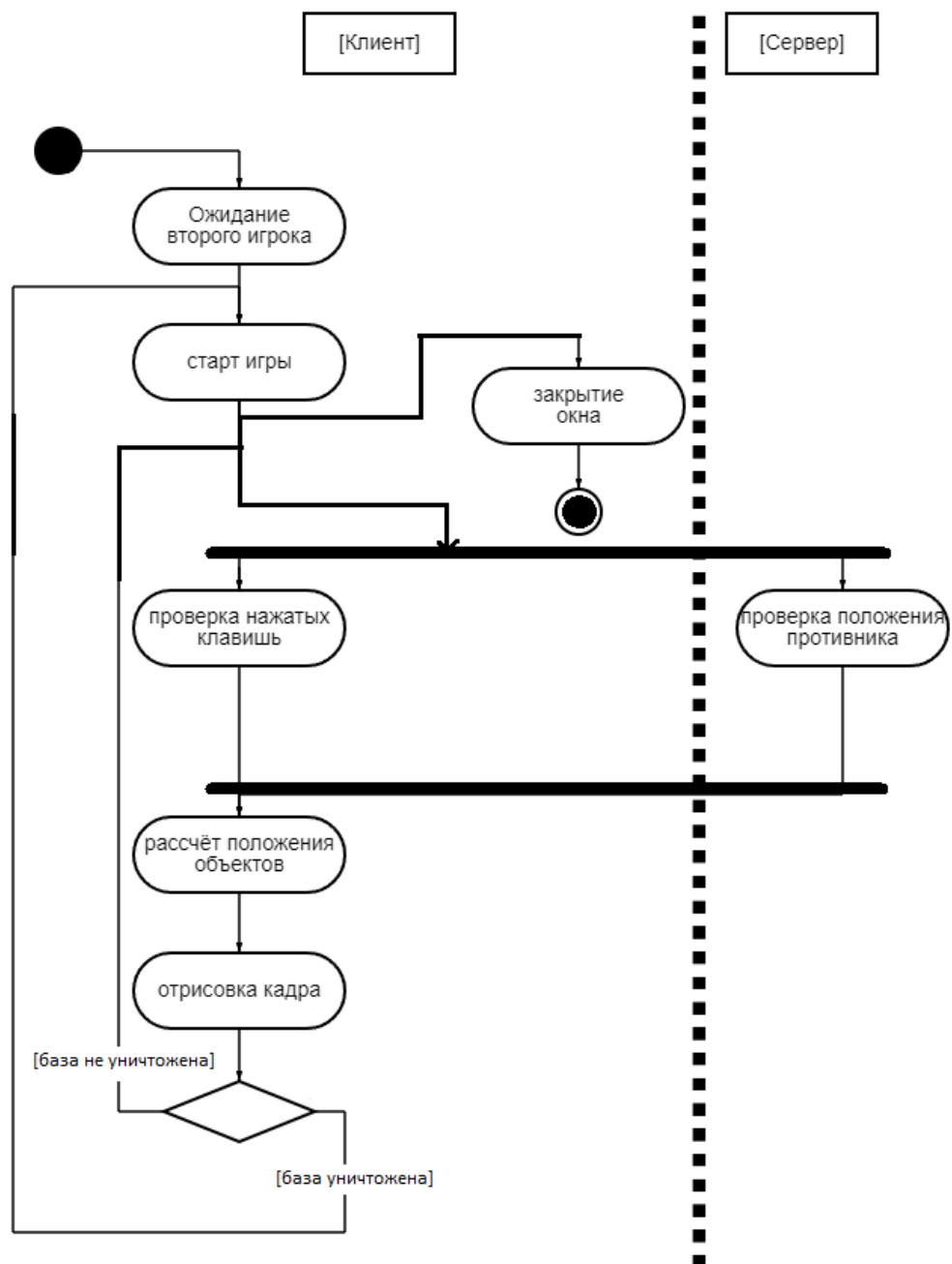


Рисунок 2 - UML диаграмма деятельности

3. Описание разработки программы

3.1. Создание интерфейса

Было создано 2 основных класса: Taper и Engine. Оба работают в отдельных потоках.

Taper отвечает за подключение или ожидание второго игрока, а также создает основное окно и перезапускает уровень при необходимости. Engine следит за обработкой и отрисовкой всех объектов, так же он получает информацию о положении второго игрока и его действиях. Диаграмма последовательности представлена на рисунке 3.

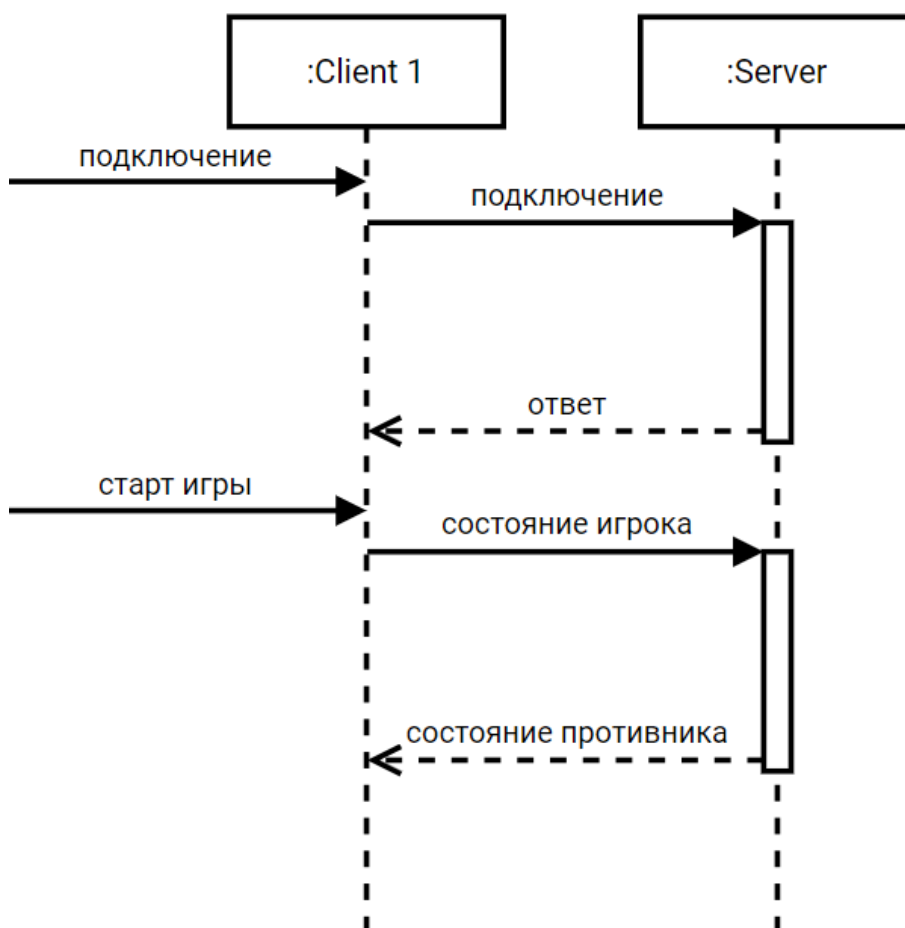


Рисунок 3 - UML диаграмма последовательности

3.2. Организация структуры программы

Структуру разрабатываемой программы составляют классы, отражающие свойства основных сущностей программы. Диаграмма классов представлена на рисунке 4.

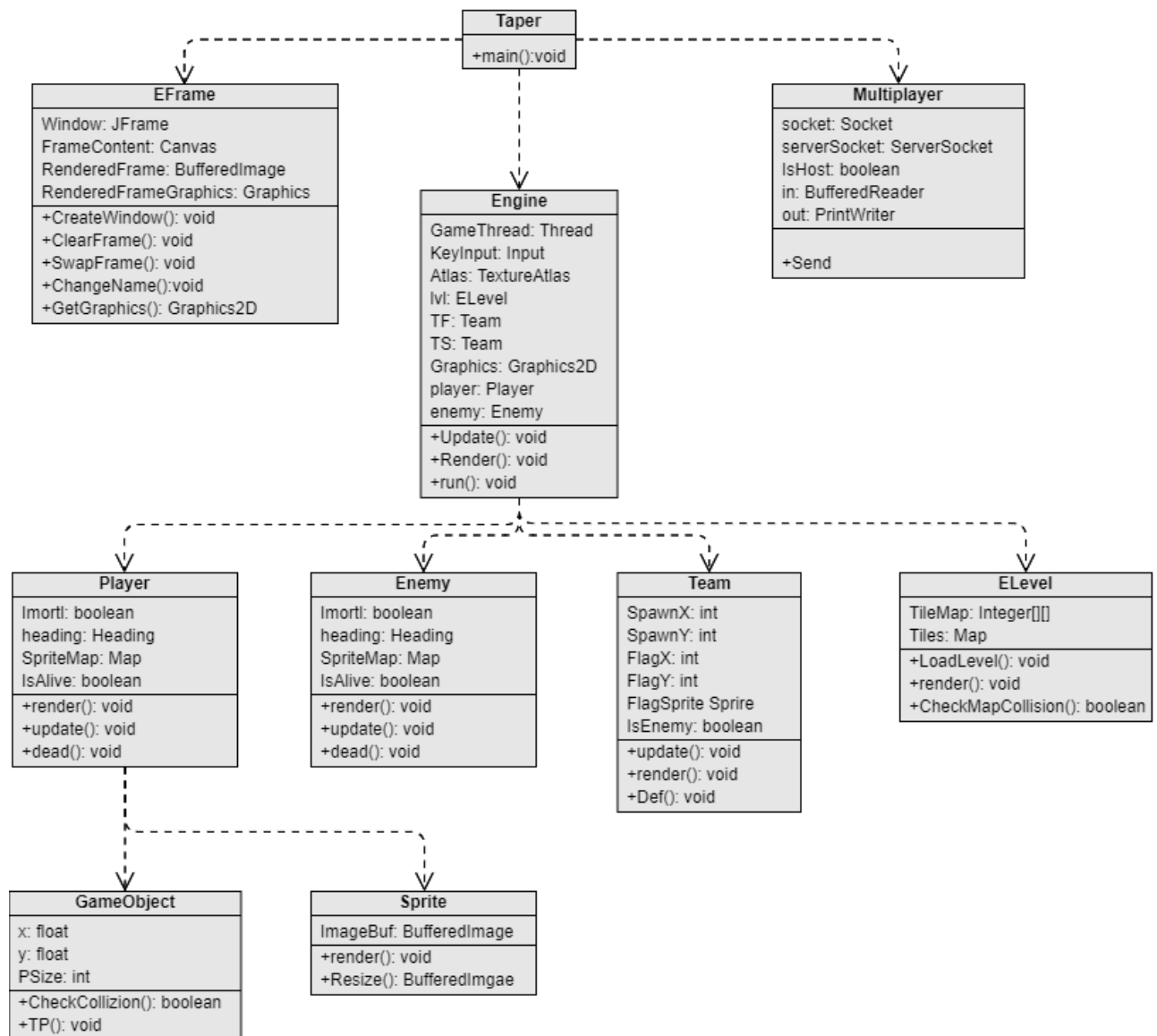


Рисунок 4 - UML диаграмма классов

3.3. Представление объектов на панели уровня

Отображение уровня осуществляется на основе текущего состояния. Частота обновления изображения на экране составляет ~75 кадров в секунду. Такое обновление организовано при помощи таймера, который вызывает событие обновления.

Расположение объектов на панели производится относительно начала координат (левый верхний угол).

Объекты наследуют класс `GameObject`, который хранит информацию о положении объекта на сцене, а также имеет метод `CheckCollision`, который проверяет коллизию между двумя объектами. Так же данный класс содержит методы `update`, вызываемый каждый кадр в классе `Engine` и отвечающий за расчёт движения объекта и его взаимодействия с другими объектами, и метод `render`, который так же вызывается каждый кадр и рисует объект на сцене .

4. Организация сетевого взаимодействия

Сервер осуществляет прием запросов на 9642 порте. Клиент и сервер обмениваются данными о положении игровых объектов каждый раз, когда один из игроков совершает какое-либо действие (передвижение или выстрел)

Диаграмма развертывания приведена на рисунке 5.

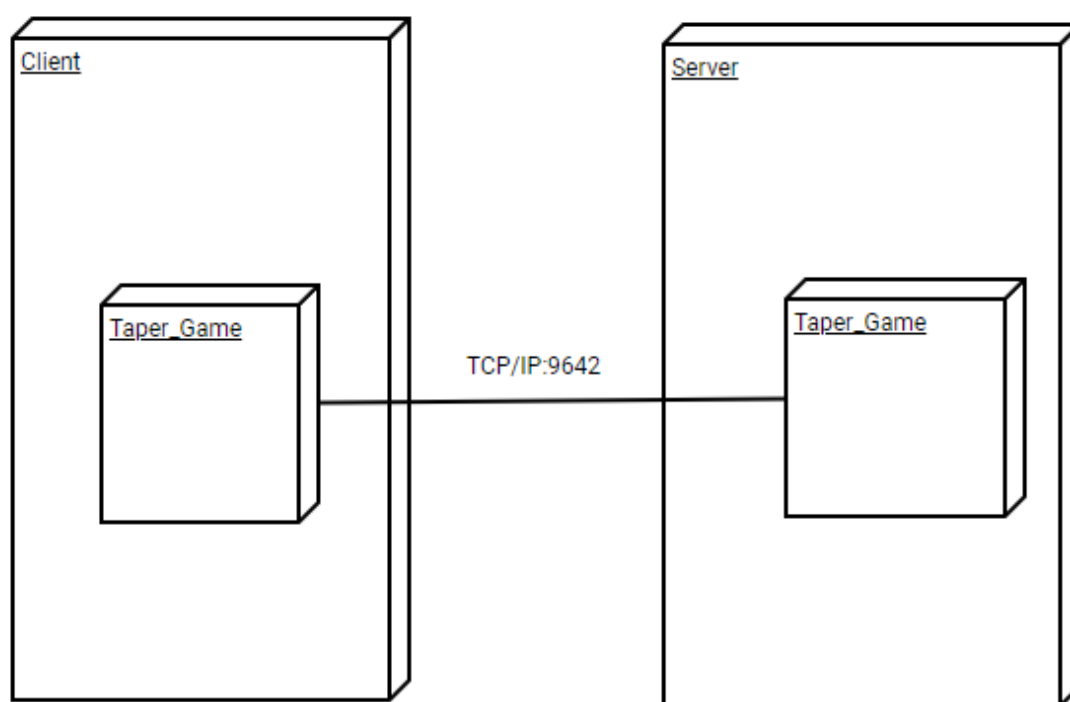


Рисунок 5 - UML диаграмма развертывания

5. Отладка и тестирование

При отладке и тестировании использовались такие инструменты среды программирования, как вывод в консоль значений переменных, анализ содержимого локальных и глобальных переменных.

Программа была протестирована путем моделирования множества возможных ситуаций, теоретически способных негативно сказаться на взаимодействии. Кроме того, был оценен объем используемых программой ресурсов, который оставался в пределах допустимых значений.

В ходе тестирования программа показала свою надежность и гибкость.

Обработка исключительных ситуаций производится с помощью конструкции try..catch.

При ошибке подключения к серверу выводится окно с сообщением об этом (рис.6).

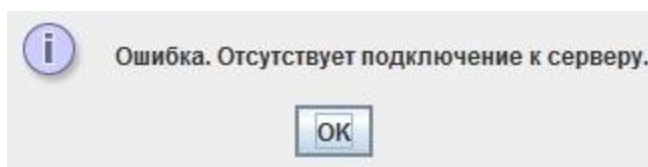


Рисунок 6 - Ошибка при подключении к серверу

Такая обработка происходит как при попытке неудачной попытке запуска игры в качестве сервера.

6. Руководство пользователя

6.1. Запуск

Запускаем игру. В случае, если кто-то уже ожидает подключения, игра сразу загрузит игровой уровень, иначе приложение переходит в режим сервера и ожидает, пока к нему не подключится второй игрок. В случае, когда оба игрока успешно подключились, вы увидите игровой уровень (рисунок 7).

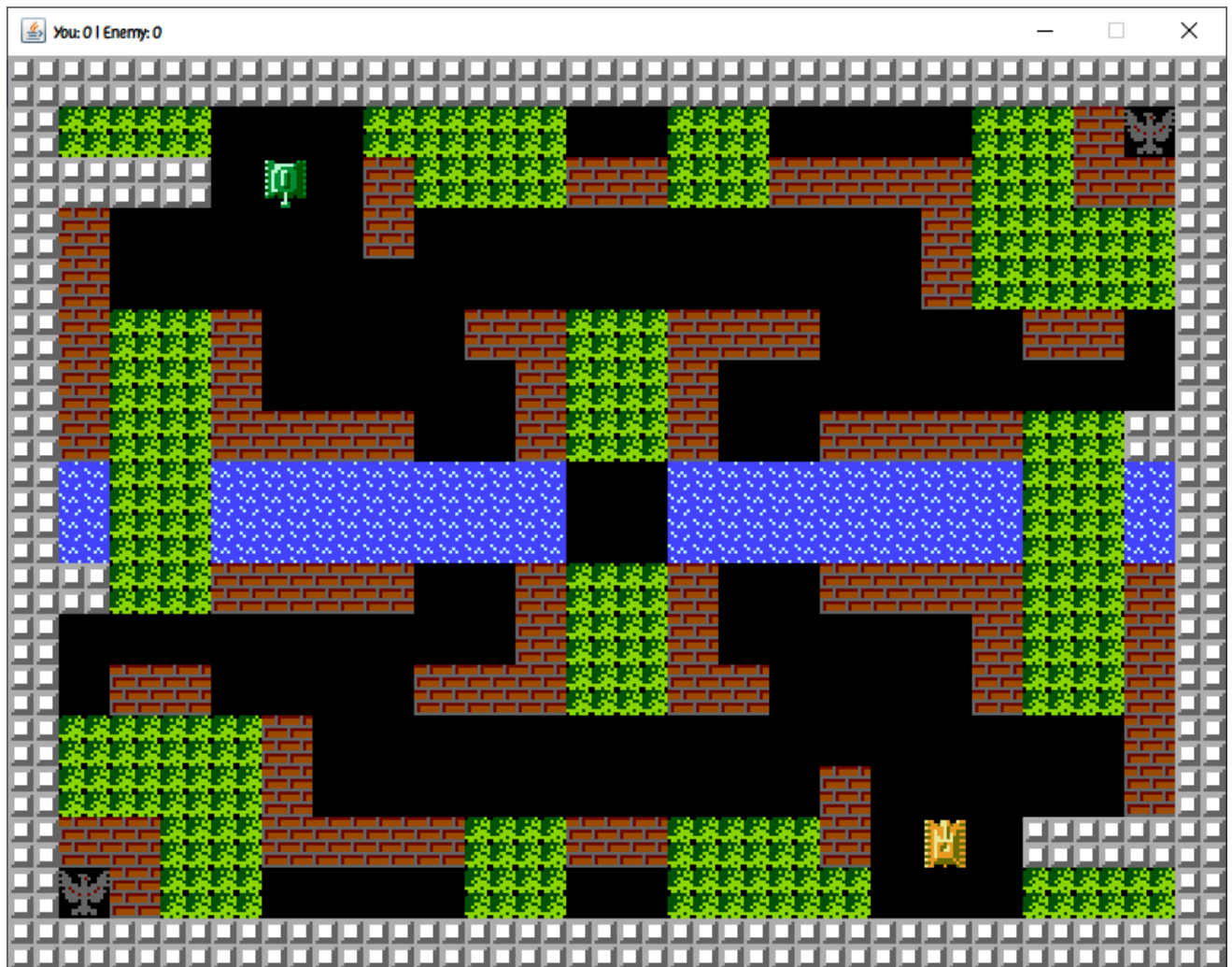


Рисунок 7 – Начало игры

6.2. Игровой процесс

Вы управляете жёлтым танком (рисунок 8), а противник – зелёным (рисунок 9).



Рисунок 8 – Танк игрока

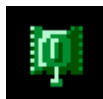


Рисунок 9 – Танк противника

Для управления танком используются стрелочки. Чтобы выстрелить нужно нажать пробел. Пуля может уничтожать противника, и кирпичные блоки (рисунок 10), остальные липы блоков не разрушаемые.

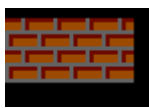


Рисунок 10 – Кирпичные блоки

При уничтожении игрока или противника, уничтоженный на несколько секунд выбывает из игры, затем появляется в своей точке появления, при этом имеет временную неуязвимость.

Для победы необходимо уничтожить базу противника. Обратите внимание, что база противника находится на том же берегу, где появляется противник. В случае неудачного выстрела, вы можете сами уничтожить свою базу и отдать победу противнику.

После уничтожения одной из баз игра перезапускается с обновлённым счётом.

Заключение

При выполнении данной курсовой работы были получены навыки разработки многомодульного приложения с пользовательским интерфейсом Swing на языке программирования Java, а также работа в среде разработки NetBeans.

В рамках выполнения курсового проекта было разработано игровое приложение “Танчики”. Приложение легко может быть усовершенствовано, что является следствием проработанной архитектуры организации при помощи модулей.

Список литературы

1. Герберт Шилдт – Java 8. Руководство для начинающих – Oracle, 2015 – 1377 с.
2. Тидвелл Дженнифер – Разработка пользовательских интерфейсов - Питер, 2011 – 480 с.
3. Портал JavaRush [Электронный ресурс]. – URL: <https://www.javarush.ru/>

Приложение А.

Листинг программы

Приложение А.1 Файл “Taper.java”

```
package taper;

import DefPack.EFrame;
import DefPack.ESettings;
import Game.Engine;
import Game.Multiplayer;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.*;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Taper{

    public static void main(String[] args) throws InterruptedException {
        try {
            System.out.println("start");
```

```

        Multiplayer.socket = new Socket("localhost",9643);

        Multiplayer.IsHost = false;

        Multiplayer.in = new BufferedReader(new
InputStreamReader(Multiplayer.socket.getInputStream()));

        Multiplayer.out = new
PrintWriter(Multiplayer.socket.getOutputStream(), true);

    } catch (IOException ex)
    {

        Multiplayer.IsHost = true;

        try {

            Multiplayer.serverSocket = new ServerSocket(9643);

            Multiplayer.socket = Multiplayer.serverSocket.accept();

            Multiplayer.in = new BufferedReader(new
InputStreamReader(Multiplayer.socket.getInputStream()));

            Multiplayer.out = new
PrintWriter(Multiplayer.socket.getOutputStream(), true);

        } catch (IOException ex1) {

            Logger.getLogger(Taper.class.getName()).log(Level.SEVERE, null,
ex1);

            return;

        }

    }

    EFrame.CreateWindow(ESettings.Name, ESettings.Width, ESettings.Height,
ESSettings.ClearColor, ESettings BuffersCount);

    while(true)

    {

        EFrame.ChangeName("You: " + ESettings.YouScore + " | Enemy: " +
ESSettings.EnemyScore);

```

```

    Engine GM = new Engine();

    GM.start();

    ESettings.IsRunning = true;

    GM.join();

    EFrame.ChangeName("END");

    EFrame.ClearFrame();

}

}

}

```

Приложение А.2 Файл “Engine.java”

```

package Game;

import java.awt.*;
import DefPack.*;
import EGraphics.Sprite;
import EGraphics.SpriteAnimation;
import EGraphics.TextureAtlas;
import GameObjects.*;
import java.awt.event.KeyEvent;
import java.util.ArrayList;

public class Engine extends Thread
{
    private Thread GameThread;
    private Input KeyInput;

```

```

private TextureAtlas Atlas;

private ELevel lvl;


private Team TF;

private Team TS;


private Graphics2D Graphics;


private Player player;

private Enemy enemy;


public Engine()
{
    ESettings.IsRunning = false;

    //EFrame.CreateWindow(ESettings.Name, ESettings.Width,
    ESettings.Height, ESettings.ClearColor, ESettings BuffersCount);

    Graphics = EFrame.GetGraphics();

    ESettings.MainGraphics = Graphics;

    KeyInput = new Input();

    EFrame.AddInputListener(KeyInput);

    Atlas = new TextureAtlas(ESettings.MainTextureName,
    ESettings.SpriteSize);

    ESettings.Atlas = Atlas;

    if(Multiplayer.IsHost)
    {
        TF = new Team(5,2,22,1,Atlas);
    }
}

```

```

    ESettings.TF = TF;

    TF.IsEnemy = false;

    TS = new Team(18,15,1,16,Atlas);

    ESettings.TS = TS;

    TF.IsEnemy = true;
}
else
{
    TS = new Team(5,2,22,1,Atlas);

    ESettings.TS = TS;

    TS.IsEnemy = false;

    TF = new Team(18,15,1,16,Atlas);

    ESettings.TF = TF;

    TF.IsEnemy = true;
}

lvl = new ELevel(Atlas);

if(Multiplayer.IsHost) player = new Player(TF.SpawnX,TF.SpawnY,
10000f,Player Heading.Down);

else player = new Player(TF.SpawnX,TF.SpawnY,
10000f,Player Heading.Up);

ESSettings.player = player;

if(Multiplayer.IsHost) enemy = new Enemy(TS.SpawnX,TS.SpawnY,
Enemy Heading.Up);

else enemy = new Enemy(TS.SpawnX,TS.SpawnY, Enemy Heading.Down);

ESSettings.enemy = enemy;

ESSettings.lvl = lvl;

```

```
}
```

```
public synchronized void StartGame()
```

```
{
```

```
    if(ESettings.IsRunning) return;
```

```
    ESettings.IsRunning = true;
```

```
    GameThread = new Thread(this);
```

```
    GameThread.start();
```

```
}
```

```
public synchronized void StopGame()
```

```
{
```

```
    if(!ESettings.IsRunning) return;
```

```
    try
```

```
    {
```

```
        GameThread.join();
```

```
    } catch (InterruptedException e)
```

```
    {
```

```
        e.printStackTrace();
```

```
    }
```

```
    Clean();
```

```
}
```

```

private void Update()
{
    player.Update(KeyInput);
    for(Bullet B: Bullet.BulletList)
    {
        B.Update();
    }
    for(Bullet B: Bullet.DeleteList)
    {
        Bullet.BulletList.remove(B);
    }
    Bullet.DeleteList.clear();

    if(KeyInput.GetKey(KeyEvent.VK_F3)) ESettings.ShowDebug =
!ESettings.ShowDebug;
}

```

```

private void Render()
{
    EFrame.ClearFrame();

    lvl.Render(Graphics);

    player.Render(Graphics);

    enemy.Render(Graphics);
}

```

```
for(Bullet B: Bullet.BulletList)
{
    B.Render(Graphics);
}
```

```
TF.Render(Graphics);
```

```
TS.Render(Graphics);
```

```
lvl.RenderGrass(Graphics);
```

```
EFrame.SwapFrame();
}
```

```
private void Clean()
```

```
{
    EFrame.DestroyWindow();
}
```

```
public void run()
```

```
{

    int c = 0;

    int FPS = 0;

    int Upd = 0;

    int UpdL = 0;
```

```

float UpdateDelta = 0;

ESettings.enemy.Update();

long Count = 0;

ETime.LastTime = ETime.GetTime();
while(ESettings.IsRunning)
{
    c++;

    //Обновление времени

    ETime.SwapTime();

    Count += ETime.Diff;

    UpdateDelta += (ETime.Diff/ ESettings.FrameInterval);

    //расчёт физики

    boolean FrameIsChanged = false;
    while(UpdateDelta > 1)
    {
        Update();

        Upd++;

        UpdateDelta--;

        if(FrameIsChanged) UpdL++;
        else FrameIsChanged = true;
    }
}

```

```

//рендер кадра
if(FrameIsChanged)
{
    Render();
    FPS++;
}else
{
    try
    {
        Thread.sleep(ESettings.ThreadSaveTime);
    } catch (InterruptedException e)
    {
        e.printStackTrace();
    }
}
if(Count >= ETime.Second)
{
    //EFrame.ChangeName(ESettings.Name + " | FPS: " + FPS + " | Upd: " +
Upd + " | UpdL: " + UpdL);
    Upd = 0;
    FPS = 0;
    UpdL = 0;
    Count = 0;
}
}
}

```

```
}
```

Приложение A.3 Файл “Multiplayer.java”

```
package Game;
```

```
import java.io.BufferedReader;
```

```
import java.io.PrintWriter;
```

```
import java.net.*;
```

```
public class Multiplayer
```

```
{
```

```
    public static Socket socket;
```

```
    public static ServerSocket serverSocket;
```

```
    public static boolean IsHost;
```

```
    public static BufferedReader in;
```

```
    public static PrintWriter out;
```

```
    public static void Send(String message)
```

```
    {
```

```
        out.println(message);
```

```
    }
```

```
}
```

Приложение A.4 Файл “EFrame.java”

```
package DefPack;
```

```
import Game.Input;
```

```

import javax.swing.*;

import java.awt.*;

import java.awt.image.*;

import java.util.Arrays;


public abstract class EFrame
{
    private static JFrame Window;

    private static Canvas FrameContent;


    private static BufferedImage RenderedFrame;

    private static int[] RenderedFrameData;

    private static Graphics RenderedFrameGraphics;

    private static int ClearColor;

    private static BufferStrategy RenderedFrameStrategy;


    public static void CreateWindow(String Name, int Wight, int Height, int
NewClearColor, int BuffersCount)
    {
        ClearColor = NewClearColor;


        FrameContent = new Canvas();

        FrameContent.setPreferredSize(new Dimension(Wight,Height));

```

```

Window = new JFrame(Name);

Window.setResizable(false);

Window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

Window.getContentPane().add(FrameContent);

Window.pack();

Window.setVisible(true);

```

```

    RenderedFrame = new
    BufferedImage(Wight,Height,BufferedImage.TYPE_INT_ARGB);

    RenderedFrameData =
    ((DataBufferInt)RenderedFrame.getRaster().getDataBuffer()).getData();

    RenderedFrameGraphics = RenderedFrame.getGraphics();

    ((Graphics2D)RenderedFrameGraphics).setRenderingHint(RenderingHints.KEY_
    ANTIALIASING,RenderingHints.VALUE_ANTIALIAS_ON);

    FrameContent.createBufferStrategy(BuffersCount);

    RenderedFrameStrategy = FrameContent.getBufferStrategy();
}

```

```

public static void ClearFrame()
{
    Arrays.fill(RenderedFrameData,ClearColor);
}

```

```

public static void SwapFrame()

```

```

{
    Graphics g = RenderedFrameStrategy.getDrawGraphics();
    g.drawImage(RenderedFrame,0,0,null);
    RenderedFrameStrategy.show();
}

```

```

public static Graphics2D GetGraphics()
{
    return (Graphics2D)RenderedFrameGraphics;
}

```

```

public static void DestroyWindow()
{
    Window.dispose();
}

```

```

public static void ChangeName(String NewName)
{
    Window.setTitle(NewName);
}

```

```

public static void AddInputListener(Input IL)
{
    Window.add(IL);
}

```

```
}
```

Приложение A.5 Файл “Player.java”

```
package GameObjects;

import DefPack.*;
import EGraphics.*;
import Game.ELevel;
import Game.Input;
import Game.Multiplayer;
import java.awt.Graphics2D;
import java.awt.event.KeyEvent;
import java.awt.image.BufferedImage;
import java.util.HashMap;
import java.util.Map;

public class Player extends GameObject
{
    public static enum Heading{
        Up    (0, 0),
        Left  (2, 0),
        Down  (4, 0),
        Right (6, 0);

        private int x,y;
```

```

Heading(int x, int y)
{
    this.x = x;
    this.y = y;
}

protected BufferedImage Texture(TextureAtlas Atlas)
{
    return Atlas.Cut(x, y, 2);
}
}

private long lastShotTime;
private Heading heading;
private Map<Heading, SpriteAnimation> SpriteMap;
private float AnimationTime;
private float CAT;
private int CurrentFrame;
private int MH;

public boolean Imortl;
public static boolean IsAlive;

public Player(float x, float y, float AnimationTime, Heading heading)
{

```

```
super(GameObjectType.Player, x*ESettings.BlockSize,  
y*ESettings.BlockSize,ESettings.SpriteSize);
```

```
Imortl = false;
```

```
IsAlive = true;
```

```
lastShotTime = System.currentTimeMillis();
```

```
this.heading = heading;
```

```
if(this.heading == Player.Heading.Up)MH = 0;
```

```
if(this.heading == Player.Heading.Left)MH = 1;
```

```
if(this.heading == Player.Heading.Down)MH = 2;
```

```
if(this.heading == Player.Heading.Right)MH = 3;
```

```
SpriteMap = new HashMap<Heading,SpriteAnimation>();
```

```
for(Heading h: Heading.values())
```

```
{
```

```
    SpriteAnimation SA = new  
SpriteAnimation(h.Texture(ESettings.Atlas),2,ESettings.SpriteSize);
```

```
    Sprite sprite = new Sprite(SA,ESettings.BlockScale);
```

```
    SpriteMap.put(h, SA);
```

```
}
```

```
this.AnimationTime = AnimationTime;
```

```
CurrentFrame = 0;
```

```

        System.out.println("Player ready");
    }

    public void Update(Input input)
    {
        if(IsAlive)
        {
            float NewX = x;

            float NewY = y;

            if(input.GetKey(KeyEvent.VK_SPACE))
            {
                long currentTime = System.currentTimeMillis();

                if (currentTime - lastShotTime >= 1400)
                {
                    if(heading == Heading.Up)Bullet.Spawn(Bullet.Heading.Up,
(int)x+ESettings.BlockSize/2-3, (int)y - 5);

                    if(heading == Heading.Down)Bullet.Spawn(Bullet.Heading.Down,
(int)x + ESettings.BlockSize/2-3, ((int)y + ESettings.BlockSize) + 5);

                    if(heading == Heading.Left)Bullet.Spawn(Bullet.Heading.Left, (int)x
- 5, (int)y + ESettings.BlockSize/2-3);

                    if(heading == Heading.Right)Bullet.Spawn(Bullet.Heading.Right,
((int)x + ESettings.BlockSize)+5, (int)y + ESettings.BlockSize/2-3);

                    Multiplayer.out.println("SHOOT " + (int)x + " " + (int)y + " " +
CurrentFrame + " " + MH);

                    lastShotTime = currentTime;
                }
            }
        }
    }

```

```

}

if(input.GetKey(KeyEvent.VK_UP))
{
    NewY -= ESettings.PlayerSpeed;
    CAT -= ETime.Diff;
    heading = Heading.Up;
    MH = 0;
}else if(input.GetKey(KeyEvent.VK_LEFT))
{
    NewX -= ESettings.PlayerSpeed;
    CAT -= ETime.Diff;
    heading = Heading.Left;
    MH = 1;
}else if(input.GetKey(KeyEvent.VK_DOWN))
{
    NewY += ESettings.PlayerSpeed;
    CAT -= ETime.Diff;
    heading = Heading.Down;
    MH = 2;
}else if(input.GetKey(KeyEvent.VK_RIGHT))
{
    NewX += ESettings.PlayerSpeed;
    CAT -= ETime.Diff;
    heading = Heading.Right;
    MH = 3;
}

```

```
}
```

```
if(CAT < 0)
```

```
{
```

```
    CAT = AnimationTime;
```

```
    if(CurrentFrame > 1)
```

```
    {
```

```
        CurrentFrame = 0;
```

```
    }
```

```
    else
```

```
    {
```

```
        CurrentFrame++;
```

```
    }
```

```
}
```

```
    if(ELevel.CheckMapCollision(new  
GameObject(this.Type,NewX,NewY,ESettings.SpriteSize), ESettings.lvl)) return;
```

```
    if(GameObject.CheckCollision(new  
GameObject(this.Type,NewX,NewY,ESettings.SpriteSize), ESettings.enemy))  
return;
```

```
    if(GameObject.CheckCollision(new  
GameObject(this.Type,NewX,NewY,ESettings.SpriteSize), new  
GameObject(GameObjectType.Flag,ESettings.TF.FlagX *ESettings.BlockSize  
,ESettings.TF.FlagY * ESettings.BlockSize,ESettings.SpriteSize))) return;
```

```
    if(GameObject.CheckCollision(new  
GameObject(this.Type,NewX,NewY,ESettings.SpriteSize), new  
GameObject(GameObjectType.Flag,ESettings.TS.FlagX *ESettings.BlockSize  
,ESettings.TS.FlagY * ESettings.BlockSize,ESettings.SpriteSize))) return;
```

```
    x = NewX;
```

```

        y = NewY;

        if(IsAlive)Multiplayer.out.println("POS " + (int)x + " " + (int)y + " " +
CurrentFrame + " " + MH);

    }

}

public void Render(Graphics2D g)
{
    Sprite.render(g, x, y,
Sprite.Resize(SpriteMap.get(heading).GetSprite(CurrentFrame),ESettings.BlockSc
ale));
}

public void Dead()
{
    Thread TT = new Thread(new ETime(GameObjectType.Player));
    TT.start();
}
}

```

Приложение А.6 Файл “Team.java”

```

package GameObjects;

import DefPack.*;

```

```

import EGraphics.*;
import java.awt.Graphics2D;

public class Team
{
    public int SpawnX;
    public int SpawnY;
    public boolean InGame;

    private Sprite FlagSprite;
    private Sprite DefFlagSprite;

    public int FlagX;
    public int FlagY;

    public boolean IsEnemy;

    public Team(int SpawnX,int SpawnY,int FlagX,int FlagY, TextureAtlas TA)
    {
        InGame = true;

        this.SpawnX = SpawnX;
        this.SpawnY = SpawnY;
    }

```

```
this.FlagX = FlagX;
```

```
this.FlagY = FlagY;
```

```
SpriteAnimation SA = new SpriteAnimation(TA.Cut(4,  
3,1),1,ESettings.SpriteSize);
```

```
FlagSprite = new Sprite(SA,ESettings.BlockScale);
```

```
SA = new SpriteAnimation(TA.Cut(5, 3,1),1,ESettings.SpriteSize);
```

```
DefFlagSprite = new Sprite(SA,ESettings.BlockScale);
```

```
}
```

```
public void Update()
```

```
{
```

```
}
```

```
public void Render(Graphics2D g)
```

```
{
```

```
FlagSprite.render(g, FlagX * ESettings.BlockSize,  
FlagY*ESettings.BlockSize);
```

```
}
```

```
public void Def()
```

```
{
```

```
FlagSprite = DefFlagSprite;
```

```
if(IsEnemy)
```

```
{
```

```

        ESettings.EnemyScore++;
    }
    else
    {
        ESettings.YouScore++;
    }
    InGame = false;
    ESettings.IsRunning = false;
}
}

```

Приложение A.7 Файл “GameObject.java”

```

package DefPack;

import DefPack.*;
import Game.ELevel;
import Game.Input;
import java.awt.Graphics2D;
import java.awt.List;
import java.util.ArrayList;

public class GameObject
{
    public final GameObjectType Type;
    public float x;
    public float y;
}

```

```
public int PSize;
```

```
public GameObject(GameObjectType Type, float x, float y , int PSize)
{
    this.Type = Type;
    this.x = x;
    this.y = y;
    this.PSize = PSize;
}
```

```
public float x(){return x;}
```

```
public float y(){return y;}
```

```
public static boolean CheckCollision(GameObject o1,GameObject o2)
{
    float BufO1 = ESettings.BlockScale * o1.PSize;
    float BufO2 = ESettings.BlockScale * o2.PSize;
    if(o1.x + BufO1 > o2.x &&
        o1.x < o2.x + BufO2 &&
        o1.y + BufO1 > o2.y &&
        o1.y < o2.y + BufO2)
    {
        return true;
    }
}
```

```

        return false;
    }

    public void TP(int x, int y)
    {
        this.x = x * PSize * ESettings.BlockScale;
        this.y = y * PSize * ESettings.BlockScale;
    }
}

```

Приложение А.8 Файл “Sprite.java”

```

package EGraphics;

import java.awt.*;
import java.awt.image.BufferedImage;

public class Sprite
{
    private SpriteAnimation Image;
    private BufferedImage ImageBuf;

    public Sprite(SpriteAnimation SA, float Scale)
    {
        Image = SA;
        ImageBuf = Image.GetSprite(0);
        ImageBuf = Sprite.Resize(ImageBuf, Scale);
    }
}

```

```
}
```

```
public void render(Graphics2D g, float x, float y)
```

```
{
```

```
    g.drawImage(ImageBuf,(int)(x),(int)(y),null);
```

```
}
```

```
public static void render(Graphics2D g, float x, float y, BufferedImage ImageR)
```

```
{
```

```
    g.drawImage(ImageR,(int)(x),(int)(y),null);
```

```
}
```

```
public static BufferedImage Resize(BufferedImage image, float Scale)
```

```
{
```

```
    int Width = (int)(image.getWidth() * Scale);
```

```
    int Height = (int)(image.getHeight() * Scale);
```

```
    BufferedImage Buffer = new  
    BufferedImage(Width,Height,BufferedImage.TYPE_INT_ARGB);
```

```
    Buffer.getGraphics().drawImage(image,0,0,Width,Height,null);
```

```
    return Buffer;
```

```
}
```

```
}
```