НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

«КИЄВО – МОГИЛЯНСЬКА АКАДЕМІЯ»

Факультет інформатики

ЛАБОРАТОРНА РОБОТА №2

“Urban Escape”

Студент: Кальянова М. В.,

І р. н., ФІ.

Викладач: Кирієнко О. В.,

старший викладач

Київ-2023

**ЗМІСТ**

[1. Постановка задачі 3](#_Toc137547955)

[1.1. Опис проекту 3](#_Toc137547956)

[1.2. Визначення основних структур та процедур 3](#_Toc137547957)

[1.3. В процесі роботи використовувати Систему контролю версій — Git. 3](#_Toc137547958)

[1.4. Графічна програма 3](#_Toc137547959)

[1.5. GUI 3](#_Toc137547960)

[1.6. Інструкції користувача 4](#_Toc137547961)

[1.7. Фінальний проект 4](#_Toc137547962)

[2. Розподіл задач 5](#_Toc137547963)

[3. Структура програми - графічна ієрархічна структура класів, методів, зв'язків 6](#_Toc137547964)

[4 .Опис методів та класів 7](#_Toc137547965)

[4.1. Barrier.java 7](#_Toc137547966)

[4.2. ChooseLevel.java 7](#_Toc137547967)

[4.3. GoSecond.java 7](#_Toc137547968)

[4.4. GoThird.java 7](#_Toc137547969)

[4.5. Lost.java 7](#_Toc137547970)

[4.6. MovingBackground.java 8](#_Toc137547971)

[4.7. MP3Player.java 8](#_Toc137547972)

[4.8. Pause.java 8](#_Toc137547973)

[4.9. Settings.java 8](#_Toc137547974)

[4.10. StartMenu.java 8](#_Toc137547975)

[4.11. Win.java 8](#_Toc137547976)

[5. Інструкція користувача зі скріншотами працюючої гри 9](#_Toc137547977)

[5.1. Загальні відомості 9](#_Toc137547978)

[5.2. Виграш 12](#_Toc137547979)

[5.3. Програш 13](#_Toc137547980)

[5.4. Кінець гри 14](#_Toc137547981)

[6. Проблеми, які виникали та шляхи їх вирішення 15](#_Toc137547982)

[7. Висновки 16](#_Toc137547983)

[8. Програмний код 17](#_Toc137547984)

[8.1. Barrier.java 17](#_Toc137547985)

[8.2. ChooseLevel.java 25](#_Toc137547986)

[8.3. GoSecond.java 30](#_Toc137547987)

[8.4. GoThird.java 35](#_Toc137547988)

[8.5. Lost.java 40](#_Toc137547989)

[8.6. MovingBackground.java 46](#_Toc137547990)

[8.7. MP3Player.java 76](#_Toc137547991)

[8.8. Pause.java 78](#_Toc137547992)

[8.9. Settings.java 83](#_Toc137547993)

[8.10. StartMenu.java 87](#_Toc137547994)

[8.11. Win.java 94](#_Toc137547995)

**Список ілюстрацій**

[Рис. 1. UML діаграма 6](#_Toc137547996)

[Рис. 2. Стартовий скрін 9](#_Toc137547997)

[Рис. 3. Інструкції 10](#_Toc137547998)

[Рис. 4. Наземні перешкоди в місті 10](#_Toc137547999)

[Рис. 5. Повітряні перешкоди в місті 11](#_Toc137548000)

[Рис. 6. Наземні перешкоди в передмісті 11](#_Toc137548001)

[Рис. 7. Повітряні перешкоди в передмісті 11](#_Toc137548002)

[Рис. 8. Наземні перешкоди в селі 11](#_Toc137548003)

[Рис. 9. Повітряні перешкоди в селі 11](#_Toc137548004)

[Рис. 10. Перехід між рівнями 12](#_Toc137548005)

[Рис. 11. Програш 13](#_Toc137548006)

[Рис. 12. Виграш в грі 14](#_Toc137548007)

# 1. Постановка задачі

Оскільки звичайні ігри всім відомі, та й коди їхні у вільному доступі, то чекаємо від вас ускладнених та видозмінених ігор, зі своїми варіаціями, ускладненнями, бонусами тощо.

Вітається написання власної гри :)

## 1.1. Опис проекту

Завантажте опис проекту, що має містити наступні розділи:

1) Постановка і аналіз задачі.

2) Розподіл ролей в середині команди.

3) Опис технічних засобів розробки.

## 1.2. Визначення основних структур та процедур

Побудувати діаграму класів, що будуть використовуватися в проекті - у випадку використання об'єктно орієнтовної парадигми, або опис основних функцій та процедур в іншому випадку.

## 1.3. В процесі роботи використовувати Систему контролю версій — Git.

## 1.4. Графічна програма

Створити малюнок, що побудований з використанням графічних бібліотек

Написати графічну програму, в якій присутня анімація

Написати інтерактивну графічну гру. Мінімум 5 різних дій від користувача. Мінімум 3 рівні гри, кожен з яких складніший за попередній.

Корисні посилання:

<https://habrahabr.ru/company/alee/blog/117430/>

<https://habrahabr.ru/company/alee/blog/127518/>

<https://netbeans.org/kb/docs/java/gui-functionality_ru.html>

## 1.5. GUI

Має бути реалізований GUI (графічний інтерфейс користувача, дизайн).

Мають бути надані screenshots з програми.

Файли мають бути запаковані в один архів та завантажені на сайт

## 1.6. Інструкції користувача

Завантажте мануал для користувачів

Створення рекламного відео-ролика та відео-презентації гри

## 1.7. Фінальний проект

Завантажте фінальний проект або посилання на нього.

# 2. Розподіл задач

**Розподіл ролей:**

* Красовський Андрій – анімації, паралакс-фон, сцена та перешкоди, робота з оперативною пам’яттю, оптимізація використання ресурсів та їх вивільнення, вибір аудіоряду, створення MP3-плеєра, налаштування аудіо, створення формул для правильного масштабування та відображення елементів GUI, фінальне тестування гри, виправлення помилок попередніх етапів, усунення проблем зі сценами, перешкодами та меню.
* Кальянова Марія - графічний дизайн, меню, фінальне тестування гри, виправлення помилок попередніх етапів, усунення проблем зі сценами та перешкодами та меню.
* Тарасенко Михайло — сцена та перешкоди, меню.

# 3. Структура програми - графічна ієрархічна структура класів, методів, зв'язків

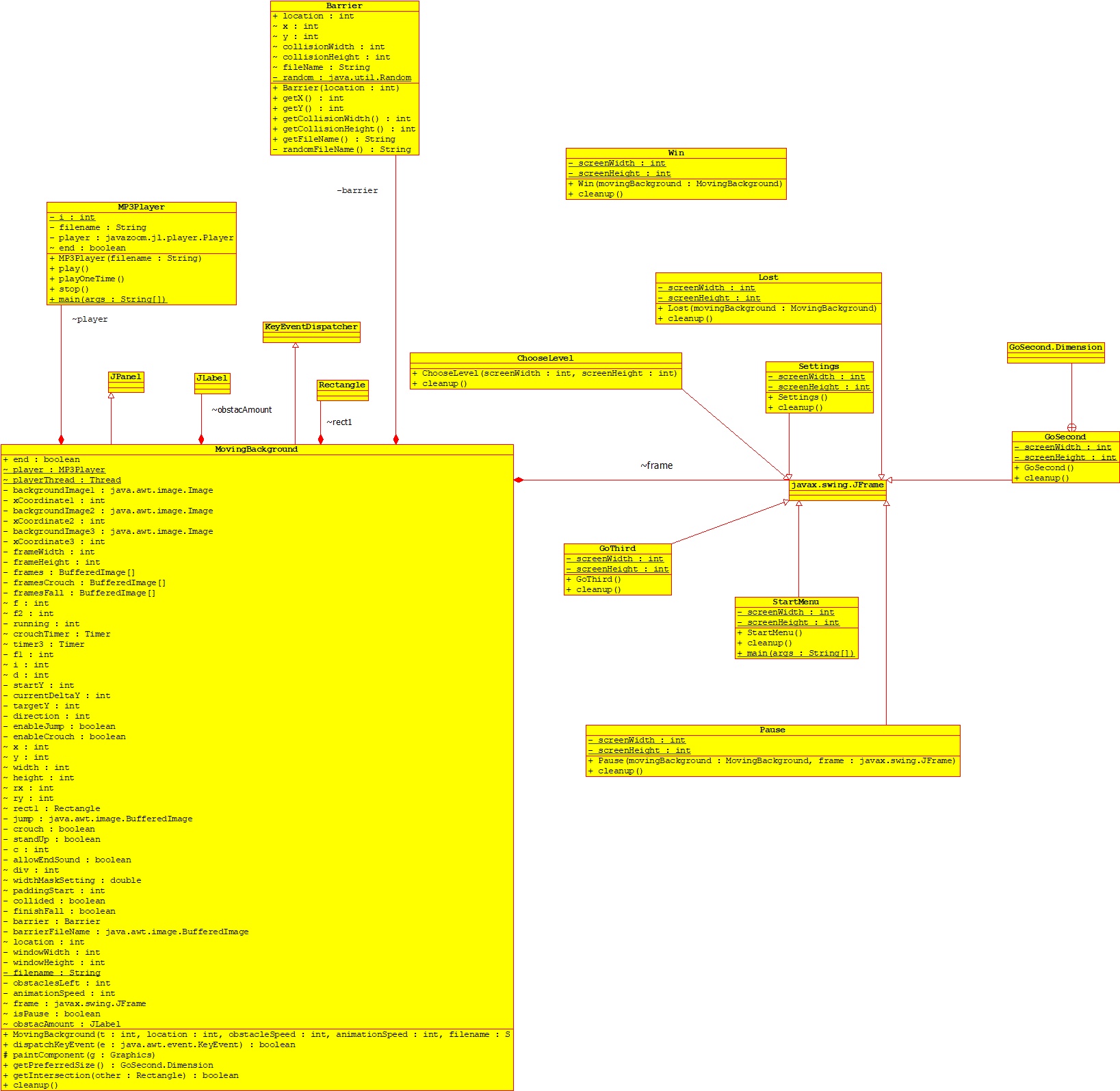


Рис. 1. UML діаграма

# 4 .Опис методів та класів

## 4.1. Barrier.java

public class Barrier /\* Barrier class \* @author Krasovskyy Andrii \* @author Tarasenko Misha \*/

private static Random random = new Random(); /\* Generator for randomly choosing obstacle \*/

public Barrier(int location) /\* Class constructor \* @param location \*/

public int getX() /\* Returns obstacle x position \* @return x \*/

public int getY() /\* Returns obstacle y position \* @return y \*/

public int getCollisionWidth() /\* Returns collision mask width \* @return collisionWidth \*/

public int getCollisionHeight() /\* Returns collision mask height \* @return collisionHeight \*/

public String getFileName() /\* Returns filepath for obstacle picture \* @return fileName\*/

private String randomFileName() /\* Randomly chooses obstacle and prepares for drawing in \* @return fileName \* @author Krasovskyy Andrii \* @author Tarasenko Misha \*/

## 4.2. ChooseLevel.java

public class ChooseLevel extends JFrame /\* Choose level class \* @author Krasovskyy Andrii \* @author Kalianova Maria \* @author Tarasenko Misha \*/

public ChooseLevel(int screenWidth, int screenHeight) /\* Class constructor \* @param screenWidth \* @param screenHeight \*/

public void cleanup() /\* Cleanup method to release resources \*/

## 4.3. GoSecond.java

public class GoSecond extends JFrame /\* Go on second level menu class \* @author Krasovskyy Andrii \* @author Kalianova Maria \* @author Tarasenko Misha \*/

## 4.4. GoThird.java

public class GoThird extends JFrame /\* Go on third level menu class \* @author Krasovskyy Andrii \* @author Kalianova Maria \* @author Tarasenko Misha \*/

## 4.5. Lost.java

public class Lost extends JFrame /\* Lose game menu class \* @author Krasovskyy Andrii \* @author Kalianova Maria \* @author Tarasenko Misha \*/

public Lost(MovingBackground movingBackground) /\* Class constructor \* @param movingBackground \*/

## 4.6. MovingBackground.java

public class MovingBackground extends JPanel implements KeyEventDispatcher /\* GUI class (draws all components, moves them and process animations) \* @author Krasovskyy Andrii \*/

public MovingBackground(int t, int location, int obstacleSpeed, int animationSpeed, String filename, int obstaclesLeft, JFrame frame) throws IOException /\* Class constructor (starts the GUI and timer animations) \* @param t \* @throws IOException \* @author Krasovskyy Andrii \*/

protected void paintComponent(Graphics g) /\* Repaints graphics (updates image) \* @author Krasovskyy Andrii \*/

## 4.7. MP3Player.java

public class MP3Player /\* MP3 player class \* @author Krasovskyy Andrii \*/

public MP3Player(String filename) /\*Class constructor \* @param filename \*/

public void play() /\* Plays mp3 track in cycle \*/

public void playOneTime() /\* Plays mp3 track \*/

public void stop() /\* Stops the mp3 track \*/

## 4.8. Pause.java

public class Pause extends JFrame /\* Pause menu class \* @author Krasovskyy Andrii \* @author Kalianova Maria \* @author Tarasenko Misha \*/

public Pause(MovingBackground movingBackground, JFrame frame) /\* Class constructor \* @param movingBackground \* @param frame \*/

## 4.9. Settings.java

public class Settings extends JFrame /\* Instruction class \* @author Krasovskyy Andrii \* @author Kalianova Maria \* @author Tarasenko Misha \*/

public Settings() /\* Class constructor \*/

## 4.10. StartMenu.java

public class StartMenu extends JFrame /\* Start menu class \* @author Krasovskyy Andrii \* @author Kalianova Maria \* @author Tarasenko Misha \*/

public StartMenu() /\* Class constructor (set up the GUI) \*/

## 4.11. Win.java

public class Win extends JFrame /\* Win game menu (finished 3rd level) class \* @author Krasovskyy Andrii \* @author Kalianova Maria \* @author Tarasenko Misha \*/

public Win(MovingBackground movingBackground) /\* Class constructor \* @param movingBackground \*/

# 5. Інструкція користувача зі скріншотами працюючої гри

## 5.1. Загальні відомості



Рис. 2. Стартовий скрін

Чоловік вирішує залишити місто, в якому він жив і працював, і вирушити в спокійне й тихе місце. Його подорож є серією випробувань і перешкод, які він повинен подолати, щоб досягти своєї мети.

***Перша локація — Місто:***

Місто є жвавим і переповненим місцем, де людині необхідно впоратися з різними труднощами, щоб почати свою подорож. Тут можна зіткнутися з заторами на дорогах, заплутатися в переплетених вулицях, зазнати стресу від шуму та суєти міського життя. Ви маєте знайти вихід із цього хаосу та знайти шлях до наступної локації.

***Друга локація — Передмістя:***

Передмістя є перехідною зоною між містом та селом. Тут людина може потрапити в більш спокійну атмосферу та природу, але тут все ще існують перешкоди. Наприклад, герой може зіткнутися з перепонами на своєму шляху. Можливо, йому доведеться подолати фізичні чи емоційні перепони, щоб продовжити свою подорож.

***Третя локація — Село:***

Село є місцем, де герой сподівається знайти спокій і спокій. Він прагне злитися зі простішим способом життя, відсутністю суєти та стресу міста. Однак і тут є перешкоди. Він повинен знайти своє місце в цьому новому середовищі й подолати останні випробування, щоб знайти справжнє благополуччя.

У результаті, через подолання всіх трьох локацій, герой зможе досягти своєї мети та знайти бажане усамітнення та спокій, залишивши суєту міста та зустрівши нові виклики долі та можливості у селі.

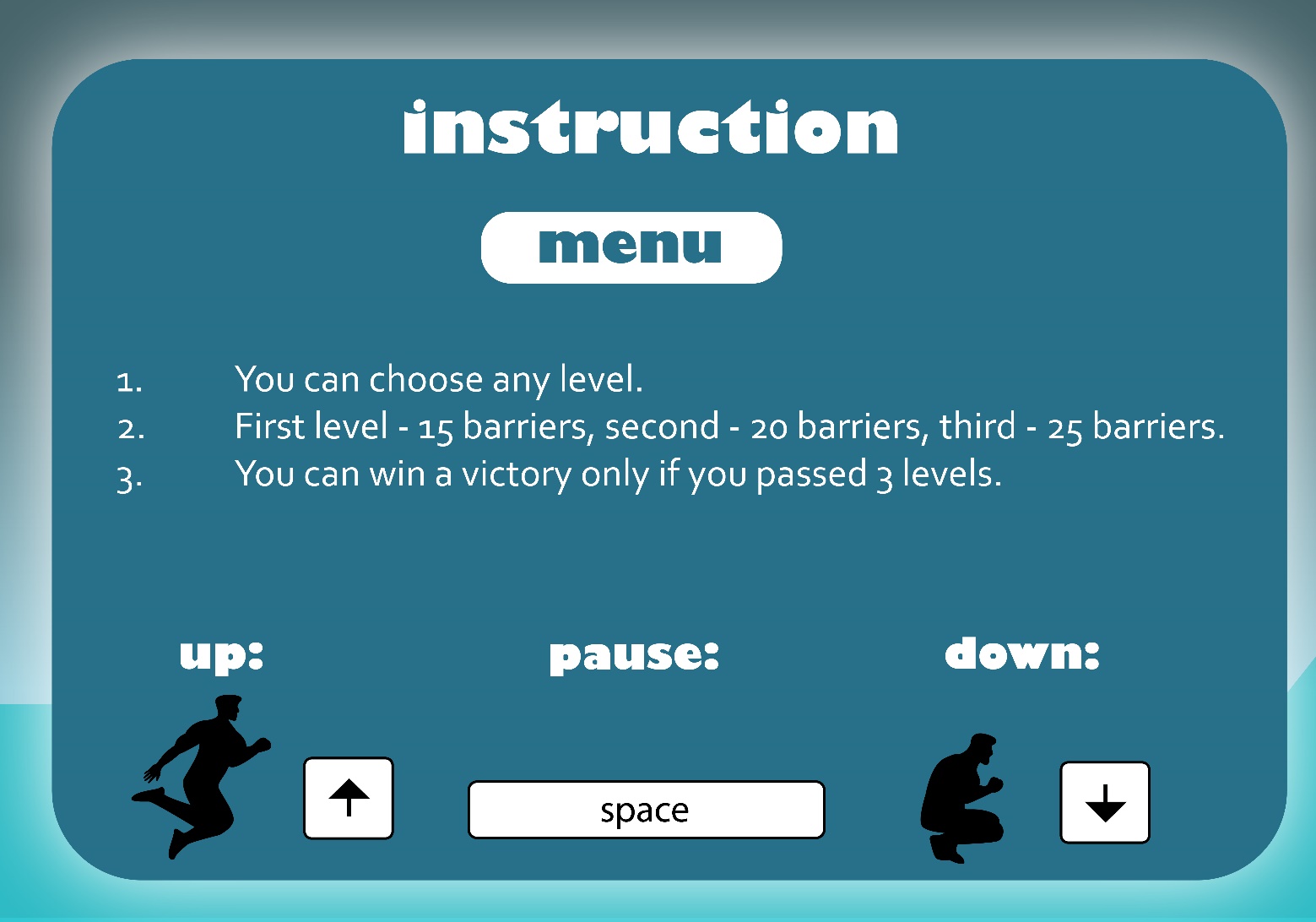


Рис. 3. Інструкції



Рис. 4. Наземні перешкоди в місті



Рис. 5. Повітряні перешкоди в місті



Рис. 6. Наземні перешкоди в передмісті



Рис. 7. Повітряні перешкоди в передмісті



Рис. 8. Наземні перешкоди в селі



Рис. 9. Повітряні перешкоди в селі

## 5.2. Виграш

Коли гравець успішно пройшов через всі випробування та перешкоди в місті, йому з'являється повідомлення "[CONGRATS! You went throught 20 barriers]". Якщо гравець бажає продовжити свою подорож та втекти з міста, він мусить натиснути на кнопку миші та обрати відповідний пункт: **menu** – перехід у головне меню програми, **next level (2)** [**next level (3)**] – для переходу на наступний рівень, **repeat** – повторно пройти поточний рівень. При переході до наступного етапу гри, кількість пройдених перешкод обнуляється.

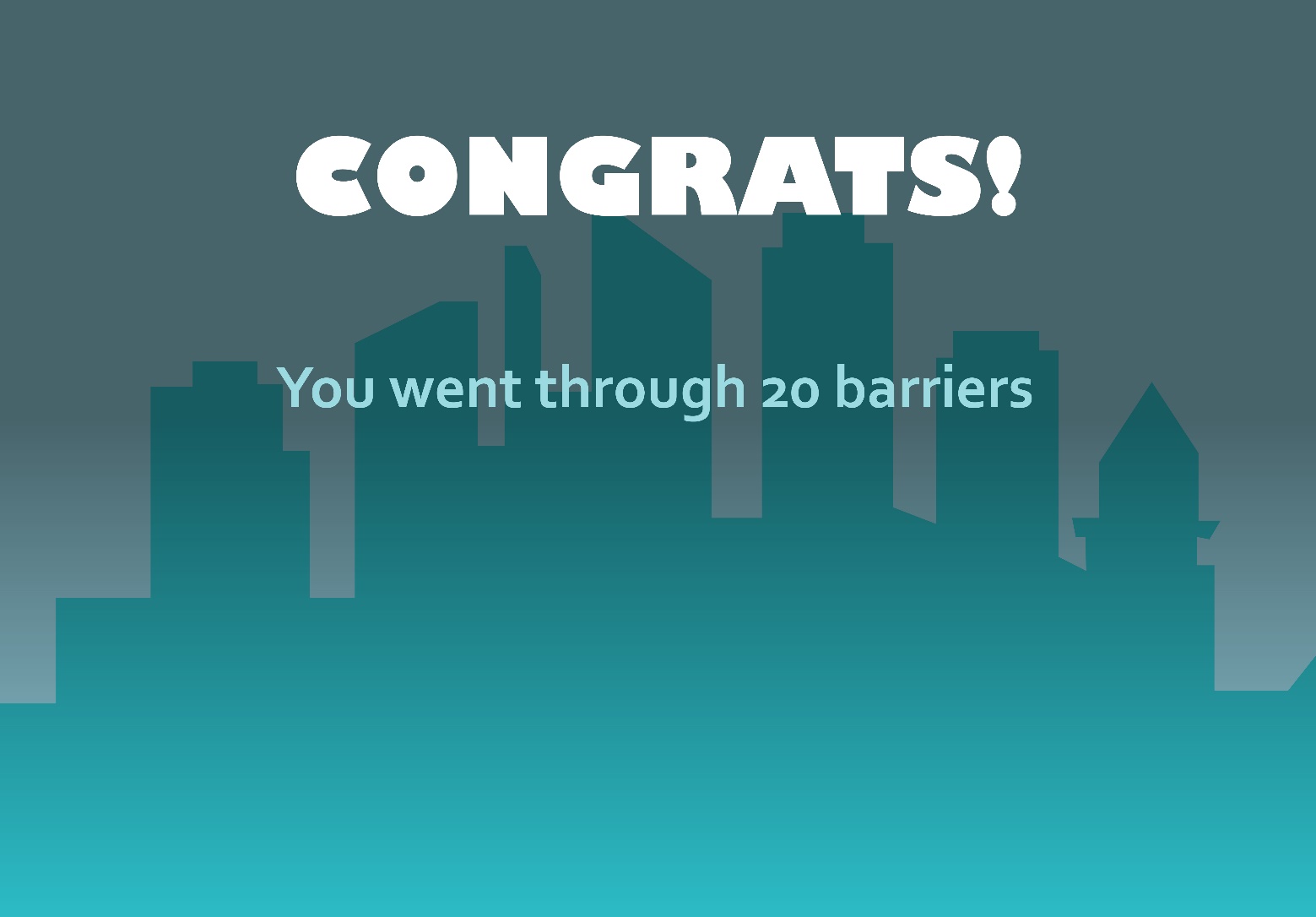
 

Рис. 10. Перехід між рівнями

## 5.3. Програш

Якщо користувач переміг на рівні, він отримує повідомлення [unfortunately… YOU LOST], далі він мусить натиснути на кнопку миші та обрати відповідний пункт: **menu** – перехід у головне меню програми, **exit** – для виходу з програми, **repeat** – повторно пройти останній рівень. При переході до наступного етапу гри, кількість пройдених перешкод обнуляється.

Рис. 11. Програш

## 5.4. Кінець гри

Якщо користувач переміг на останньому рівні, він отримує повідомлення про завершення гри [CONGRATS! You won the game and passed in summary 60 barriers! The noise of then city won’t bother you anymore] далі він мусить натиснути на кнопку миші та обрати відповідний пункт: **menu** – перехід у головне меню програми, **exit** – для виходу з програми, **repeat** – повторно пройти останній рівень.

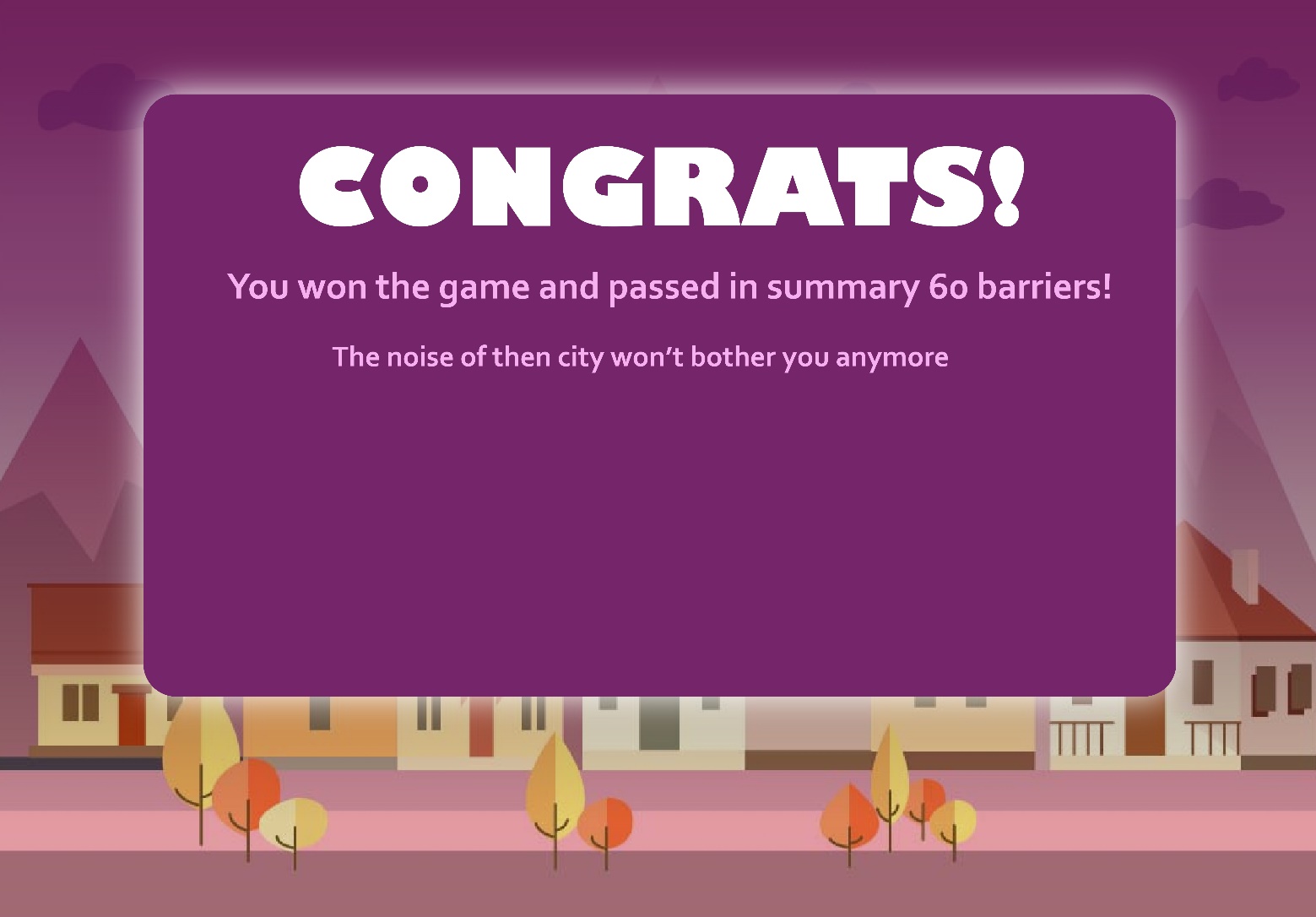


Рис. 12. Виграш в грі

# 6. Проблеми, які виникали та шляхи їх вирішення

*Красовський О.:*

Основною проблемою було використання програмою оперативної пам'яті. У грі використовуються якісні зображення, тому їх переміщення, обробка та масштабування займає приблизно 1 ГБ оперативної пам'яті. Місця, де не вивільняються ресурси, я виявляв за допомогою періодичного знімання даних заповненості heap через Runtime. Тому вирішив розробити методи для очистки ресурсів, що стало ще однією проблемою, адже gc в Java не можна викликати, можна лише передати рекомендацію на запуск. В процесі вирішення проблеми я перебрав кілька варіантів, поки дані статистки використання пам'яті не показали, що вивільнено ~1 ГБ після запуску метода cleanup() в класі MovingBackground.

Створення паралакс-фону було досить цікавою та складною задачею, адже треба зробити з одного зображення рухому стрічку, використовуючи при цьому мінімум ресурсів. Я створив алгоритм з використання Timer та методу paintComponent.

Анімація бігу, стрибка та присідання забрала багато часу, адже необхідно було розробити компактний код для зміни кадрів, щоб все рухалося плавно. Тут я використав алгоритм подібний до того, що створює рухомий фон.

Іншою проблемою була перевірка на зіткнення, адже необхідно було передбачити всі можливі ситуації та події, які можуть відбутися з персонажем. Цю проблему я вирішив побудовою алгоритму спочатку на папері, а вже потім у програмі. Також за допомогою розписаного алгоритму я уявляв, як має поводитися персонаж при зіткненні, щоб точно все перенести в програмний код.

Також необхідно було правильно вибрати спосіб перевірки колізій, адже персонаж є людиною і має складну форму.

Необхідно було розробити MP3-програвач, для чого довелося використати бібліотеку javazoom. Також мені довелося вивчити основи роботи з потоками, адже музику необхідно було програвати паралельно з графічною складовою гри.

Також гра має стартувати на різних комп'ютерах, тому необхідно було врахувати, що в різних користувачів різні розміри екрану, а також може бути присутня панель інструментів знизу екрану. Для цього я розробив формули масштабування та розміщення всіх елементів, а також формули для розрахунку робочої області екрану.

Загалом більшість проблем полягала в усуненні дрібних недоліків та збільшенні зручності використання гри.

# 7. Висновки

Гра має три рівні: Місто, Передмістя та Село. Кожен рівень має свої особливості та перешкоди, які гравець повинен подолати.

У грі є наземні та повітряні перешкоди, які гравець повинен подолати, щоб прогресувати.

Є можливість перемоги або програшу на кожному рівні. При перемозі гравець отримує відповідне повідомлення, а при програшу має можливість повторити рівень або перейти до наступного.

При успішному проходженні всіх рівнів гравець отримує повідомлення про завершення гри та досягнення своєї мети.

У процесі розробки гри виникали різні проблеми, такі як використання оперативної пам'яті, створення паралакс-фону, анімації персонажа, перевірки на зіткнення та інші. Я використовував різноманітні алгоритми та методи, щоб вирішити ці проблеми.

Гра також має враховувати різні розміри екрана та працювати на різних комп'ютерах, тому були розроблені формули для коректного масштабування та розміщення елементів гри.

Загальною метою розробки гри було забезпечити комфортне та зручне користування для гравців, усунути недоліки та поліпшити функціональність гри.

# 8. Програмний код

## 8.1. Barrier.java

import java.awt.\*;

import java.io.File;

import java.io.IOException;

import java.util.Random;

import javax.imageio.ImageIO;

/\*\*

\* Barrier class

\* @author Krasovskyy Andrii

\* @author Tarasenko Misha

\*/

public class Barrier {

/\*\*

\* Contains current level number

\*/

int location;

/\*\*

\* x position for obstacle

\*/

int x;

/\*\*

\* y position for obstacle

\*/

int y;

/\*\*

\* Collision mask width

\*/

int collisionWidth;

/\*\*

\* Collision mask height

\*/

int collisionHeight;

/\*\*

\* Contains obstacle picture filepath

\*/

String fileName;

/\*\*

\* Generator for randomly choosing obstacle

\*/

private static Random random = new Random();

/\*\*

\* Class constructor

\* @param location

\*/

public Barrier(int location) {

this.location = location;

this.fileName = randomFileName();

}

/\*\*

\* Returns obstacle x position

\* @return x

\*/

public int getX() {

return x;

}

/\*\*

\* Returns obstacle y position

\* @return y

\*/

public int getY() {

return y;

}

/\*\*

\* Returns collision mask width

\* @return collisionWidth

\*/

public int getCollisionWidth() {

return collisionWidth;

}

/\*\*

\* Returns collision mask height

\* @return collisionHeight

\*/

public int getCollisionHeight() {

return collisionHeight;

}

/\*\*

\* Returns filepath for obstacle picture

\* @return fileName

\*/

public String getFileName() {

return fileName;

}

/\*\*

\* Randomly chooses obstacle and prepares for drawing in

\* @return fileName

\* @author Krasovskyy Andrii

\* @author Tarasenko Misha

\*/

private String randomFileName() {

String fileName = "city\_ground1.png";

int number, groundOrSky;

Toolkit toolkit = Toolkit.getDefaultToolkit();

Dimension screenSize = toolkit.getScreenSize();

int windowWidth = screenSize.width;

int windowHeight = (int) (screenSize.height\*0.95);

GraphicsEnvironment env = GraphicsEnvironment.getLocalGraphicsEnvironment();

GraphicsDevice device = env.getDefaultScreenDevice();

GraphicsConfiguration config = device.getDefaultConfiguration();

Insets insets = Toolkit.getDefaultToolkit().getScreenInsets(config);

int toolbarHeight = insets.top + insets.bottom;

windowWidth = screenSize.width;

windowHeight = screenSize.height - toolbarHeight;

System.out.println(windowWidth + "-" + windowHeight + "-" + toolbarHeight);

switch (location) {

case 1: {

groundOrSky = random.nextInt(2);;

if(groundOrSky < 1) {

number = random.nextInt(7) + 1;

fileName = "city\_ground" + number + ".png";

try {

this.collisionWidth = ImageIO.read(new File(fileName)).getWidth() / 17 - 10;

this.collisionHeight = ImageIO.read(new File(fileName)).getHeight() / 15;

} catch (IOException e) {

e.printStackTrace();

}

this.x = (int) (windowWidth);

this.y = (int) (windowHeight - 35 - collisionHeight);

}else {

number = random.nextInt(3) + 1;

fileName = "city\_sky" + number + ".png";

try {

this.collisionWidth = ImageIO.read(new File(fileName)).getWidth() / 25 - 10;

this.collisionHeight = ImageIO.read(new File(fileName)).getHeight() / 25;

} catch (IOException e) {

e.printStackTrace();

}

this.x = (int) (windowWidth);

this.y = (int) (windowHeight \* 0.65 - collisionHeight);

}

break;

}

case 2:{

groundOrSky = random.nextInt(2);

if(groundOrSky < 1) {

number = random.nextInt(7) + 1;

fileName = "suburb\_ground" + number + ".png";

try {

this.collisionWidth = ImageIO.read(new File(fileName)).getWidth() / 17 - 10;

this.collisionHeight = ImageIO.read(new File(fileName)).getHeight() / 15;

} catch (IOException e) {

e.printStackTrace();

}

this.x = (int) (windowWidth);

this.y = (int) (windowHeight - 35 - collisionHeight);

}

else {

number = random.nextInt(3) + 1;

fileName = "suburb\_sky" + number + ".png";

try {

this.collisionWidth = ImageIO.read(new File(fileName)).getWidth() / 25 - 10;

this.collisionHeight = ImageIO.read(new File(fileName)).getHeight() / 25;

} catch (IOException e) {

e.printStackTrace();

}

this.x = (int) (windowWidth);

this.y = (int) (windowHeight \* 0.65 - collisionHeight);

}

break;

}

case 3:{

groundOrSky = random.nextInt(2);

if(groundOrSky < 1) {

number = random.nextInt(8) + 1;

fileName = "village\_ground" + number + ".png";

try {

this.collisionWidth = ImageIO.read(new File(fileName)).getWidth() / 17 - 10;

this.collisionHeight = ImageIO.read(new File(fileName)).getHeight() / 15;

} catch (IOException e) {

e.printStackTrace();

}

this.x = (int) (windowWidth);

this.y = (int) (windowHeight - 35 - collisionHeight);

}

else {

number = random.nextInt(3) + 1;

fileName = "village\_sky" + number + ".png";

try {

this.collisionWidth = ImageIO.read(new File(fileName)).getWidth() / 25 - 10;

this.collisionHeight = ImageIO.read(new File(fileName)).getHeight() / 25;

} catch (IOException e) {

e.printStackTrace();

}

this.x = (int) (windowWidth);

this.y = (int) (windowHeight \* 0.65 - collisionHeight);

}

break;

}

}

System.out.println(fileName);

return fileName;

}

}

## 8.2. ChooseLevel.java

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.io.IOException;

/\*\*

\* Choose level class

\* @author Krasovskyy Andrii

\* @author Kalianova Maria

\* @author Tarasenko Misha

\*/

public class ChooseLevel extends JFrame{

/\*\*

\* Class constructor

\* @param screenWidth

\* @param screenHeight

\*/

public ChooseLevel(int screenWidth, int screenHeight) {

Toolkit toolkit = Toolkit.getDefaultToolkit();

Dimension screenSize = toolkit.getScreenSize();

GraphicsEnvironment env = GraphicsEnvironment.getLocalGraphicsEnvironment();

GraphicsDevice device = env.getDefaultScreenDevice();

GraphicsConfiguration config = device.getDefaultConfiguration();

Insets insets = Toolkit.getDefaultToolkit().getScreenInsets(config);

int toolbarHeight = insets.top + insets.bottom;

screenWidth = screenSize.width;

screenHeight = screenSize.height - toolbarHeight;

System.out.println(screenWidth + "-" + screenHeight + "-" + toolbarHeight);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(screenWidth, screenHeight);

setLocationRelativeTo(null);

setTitle("start menu");

setLayout(null);

ImageIcon buttonIcon = new ImageIcon("level\_choice2.png");

Image button = buttonIcon.getImage();

Image scaledButton = button.getScaledInstance((int) (screenWidth \* 0.3), (int) (screenHeight \* 0.43), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon = new ImageIcon(scaledButton);

JLabel buttonLabel = new JLabel(scaledButtonIcon);

buttonLabel.setBounds((int) (screenWidth / 2 - screenWidth \* 0.115), (int) (screenHeight / 2 - screenHeight \* 0.12), (int) (screenWidth \* 0.26), (int) (screenHeight \* 0.28));

buttonLabel.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

JFrame frame = new JFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

MovingBackground backgroundPanel;

try {

backgroundPanel = new MovingBackground(10, 2, 40, 70, "2.mp3", 20, frame);

//backgroundPanel = new MovingBackground(10, 2, 40, 70, "2.mp3", 3, frame);

frame.add(backgroundPanel);

frame.pack();

frame.setExtendedState(JFrame.MAXIMIZED\_BOTH);

frame.setVisible(true);

}

catch (IOException e1) {

e1.printStackTrace();

}

cleanup();

dispose();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {}

});

add(buttonLabel);

ImageIcon buttonIcon2 = new ImageIcon("level\_choice1.png");

Image button2 = buttonIcon2.getImage();

Image scaledButton2 = button2.getScaledInstance((int) (screenWidth \* 0.3), (int) (screenHeight \* 0.43), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon2 = new ImageIcon(scaledButton2);

JLabel buttonLabel2 = new JLabel(scaledButtonIcon2);

buttonLabel2.setBounds((int) (screenWidth / 2 - screenWidth \* 0.440), (int) (screenHeight / 2 - screenHeight \* 0.12), (int) (screenWidth \* 0.26), (int) (screenHeight \* 0.28));

buttonLabel2.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e){

JFrame frame = new JFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

MovingBackground backgroundPanel;

try {

backgroundPanel = new MovingBackground(10, 1, 60, 80, "1.mp3", 15, frame);

//backgroundPanel = new MovingBackground(10, 1, 60, 80, "1.mp3", 3, frame);

frame.add(backgroundPanel);

frame.pack();

frame.setExtendedState(JFrame.MAXIMIZED\_BOTH);

frame.setVisible(true);

} catch (IOException e1) {

e1.printStackTrace();

}

cleanup();

dispose();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {}

});

add(buttonLabel2);

ImageIcon buttonIcon3 = new ImageIcon("level\_choice3.png");

Image button3 = buttonIcon3.getImage();

Image scaledButton3 = button3.getScaledInstance((int) (screenWidth \* 0.3), (int) (screenHeight \* 0.43), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon3 = new ImageIcon(scaledButton3);

JLabel buttonLabel3 = new JLabel(scaledButtonIcon3);

buttonLabel3.setBounds((int) (screenWidth / 2 + screenWidth \* 0.235), (int) (screenHeight / 2 - screenHeight \* 0.12), (int) (screenWidth \* 0.26), (int) (screenHeight \* 0.28));

buttonLabel3.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

JFrame frame = new JFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

MovingBackground backgroundPanel;

try {

backgroundPanel = new MovingBackground(10, 3, 20, 60, "3.mp3", 25, frame);

//backgroundPanel = new MovingBackground(10, 3, 20, 60, "3.mp3", 3, frame);

frame.add(backgroundPanel);

frame.pack();

frame.setExtendedState(JFrame.MAXIMIZED\_BOTH);

frame.setVisible(true);

} catch (IOException e1) {

e1.printStackTrace();

}

cleanup();

dispose();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {

//ChooseLevel level = new ChooseLevel(finalScreenWidth, finalScreenHeight);

//dispose();

}

});

add(buttonLabel3);

ImageIcon imageIcon = new ImageIcon("main2.jpg");

Image image = imageIcon.getImage();

Image scaledImage = image.getScaledInstance(screenWidth, screenHeight, Image.SCALE\_SMOOTH);

ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

JLabel label = new JLabel(scaledImageIcon);

label.setBounds(0, 0, screenWidth, screenHeight);

add(label);

setVisible(true);

}

/\*\*

\* Cleanup method to release resources

\*/

public void cleanup() {

removeAll();

System.gc();

}

}

## 8.3. GoSecond.java

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.io.IOException;

/\*\*

\* Go on second level menu class

\* @author Krasovskyy Andrii

\* @author Kalianova Maria

\* @author Tarasenko Misha

\*/

public class GoSecond extends JFrame {

/\*\*

\* Screen width

\*/

private static int screenWidth;

/\*\*

\* Screen height

\*/

private static int screenHeight;

/\*\*

\* Class constructor

\*/

public GoSecond() {

Toolkit toolkit = Toolkit.getDefaultToolkit();

Dimension screenSize = toolkit.getScreenSize();

screenWidth = screenSize.width;

screenHeight = screenSize.height;

GraphicsEnvironment env = GraphicsEnvironment.getLocalGraphicsEnvironment();

GraphicsDevice device = env.getDefaultScreenDevice();

GraphicsConfiguration config = device.getDefaultConfiguration();

Insets insets = Toolkit.getDefaultToolkit().getScreenInsets(config);

int toolbarHeight = insets.top + insets.bottom;

screenWidth = screenSize.width;

screenHeight = screenSize.height - toolbarHeight;

System.out.println(screenWidth + "-" + screenHeight + "-" + toolbarHeight);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(screenWidth, screenHeight);

setLocationRelativeTo(null);

setTitle("pause menu");

setLayout(null);

ImageIcon buttonIcon = new ImageIcon("gosecond\_menu.png");

Image button = buttonIcon.getImage();

Image scaledButton = button.getScaledInstance((int) (screenWidth \* 0.4), (int) (screenHeight \* 0.1), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon = new ImageIcon(scaledButton);

JLabel buttonLabel = new JLabel(scaledButtonIcon);

buttonLabel.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight \* (0.48 + 0.045)), (int) (screenWidth \* 0.4), (int) (screenHeight \* 0.11));

buttonLabel.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

StartMenu startMenu = new StartMenu();

cleanup();

dispose();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {

// ChooseLevel level = new ChooseLevel(finalScreenWidth1, finalScreenHeight1);

// dispose();

}

});

add(buttonLabel);

ImageIcon buttonIcon1 = new ImageIcon("gosecond\_next.png");

Image button1 = buttonIcon1.getImage();

Image scaledButton1 = button1.getScaledInstance((int) (screenWidth \* 0.4), (int) (screenHeight \* 0.1), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon1 = new ImageIcon(scaledButton1);

JLabel buttonLabel1 = new JLabel(scaledButtonIcon1);

buttonLabel1.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight / 2 + screenHeight \* 0.05), (int) (screenWidth \* 0.4), (int) (screenHeight \* 0.11));

buttonLabel1.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight \* (0.48 + 0.045\*2 + 0.1)), (int) (screenWidth \* 0.4), (int) (screenHeight \* 0.11));

buttonLabel1.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

JFrame frame = new JFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

MovingBackground backgroundPanel;

try {

backgroundPanel = new MovingBackground(10, 2, 40, 70, "2.mp3", 20, frame);

//backgroundPanel = new MovingBackground(10, 2, 40, 70, "2.mp3", 3, frame);

frame.add(backgroundPanel);

frame.pack();

frame.setExtendedState(JFrame.MAXIMIZED\_BOTH);

frame.setVisible(true);

} catch (IOException e1) {

e1.printStackTrace();

}

cleanup();

dispose();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {}

});

add(buttonLabel1);

ImageIcon buttonIcon2 = new ImageIcon("gosecond\_repeat.png");

Image button2 = buttonIcon2.getImage();

Image scaledButton2 = button2.getScaledInstance((int) (screenWidth \* 0.4), (int) (screenHeight \* 0.1), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon2 = new ImageIcon(scaledButton2);

JLabel buttonLabel2 = new JLabel(scaledButtonIcon2);

buttonLabel2.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight / 2 + screenHeight \* 0.18), (int) (screenWidth \* 0.4), (int) (screenHeight \* 0.11));

buttonLabel2.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight \* (0.48 + 0.045\*3 + 0.1\*2)), (int) (screenWidth \* 0.4), (int) (screenHeight \* 0.11));

buttonLabel2.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

JFrame frame = new JFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

MovingBackground backgroundPanel;

try {

backgroundPanel = new MovingBackground(10, 1, 60, 80, "1.mp3", 15, frame);

//backgroundPanel = new MovingBackground(10, 2, 40, 70, "1.mp3", 3, frame);

frame.add(backgroundPanel);

frame.pack();

frame.setExtendedState(JFrame.MAXIMIZED\_BOTH);

frame.setVisible(true);

} catch (IOException e1) {

e1.printStackTrace();

}

cleanup();

dispose();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {

//ChooseLevel level = new ChooseLevel(finalScreenWidth1, finalScreenHeight1);

//dispose();

}

});

add(buttonLabel2);

ImageIcon imageIcon = new ImageIcon("first\_back.jpg");

Image image = imageIcon.getImage();

Image scaledImage = image.getScaledInstance(screenWidth, (int) (screenHeight), Image.SCALE\_SMOOTH);

ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

JLabel label = new JLabel(scaledImageIcon);

label.setBounds(0, 0, screenWidth, (int) (screenHeight));

add(label);

setVisible(true);

}

/\*\*

\* Cleanup method to release resources

\*/

public void cleanup() {

removeAll();

screenWidth = 0;

screenHeight = 0;

System.gc();

}

}

## 8.4. GoThird.java

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.io.IOException;

/\*\*

\* Go on third level menu class

\* @author Krasovskyy Andrii

\* @author Kalianova Maria

\* @author Tarasenko Misha

\*/

public class GoThird extends JFrame {

/\*\*

\* Screen width

\*/

private static int screenWidth;

/\*\*

\* Screen height

\*/

private static int screenHeight;

/\*\*

\* Class constructor

\*/

public GoThird() {

Toolkit toolkit = Toolkit.getDefaultToolkit();

Dimension screenSize = toolkit.getScreenSize();

screenWidth = screenSize.width;

screenHeight = screenSize.height;

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(screenWidth, screenHeight);

setLocationRelativeTo(null);

setTitle("pause menu");

setLayout(null);

ImageIcon buttonIcon = new ImageIcon("gothird\_menu.png");

Image button = buttonIcon.getImage();

Image scaledButton = button.getScaledInstance((int) (screenWidth \* 0.4), (int) (screenHeight \* 0.1), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon = new ImageIcon(scaledButton);

JLabel buttonLabel = new JLabel(scaledButtonIcon);

buttonLabel.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight \* (0.48 + 0.045)), (int) (screenWidth \* 0.4), (int) (screenHeight \* 0.11));

buttonLabel.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

StartMenu startMenu = new StartMenu();

cleanup();

dispose();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {

//ChooseLevel level = new ChooseLevel(finalScreenWidth1, finalScreenHeight1);

//dispose();

}

});

add(buttonLabel);

ImageIcon buttonIcon1 = new ImageIcon("gothird\_next.png");

Image button1 = buttonIcon1.getImage();

Image scaledButton1 = button1.getScaledInstance((int) (screenWidth \* 0.4), (int) (screenHeight \* 0.1), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon1 = new ImageIcon(scaledButton1);

JLabel buttonLabel1 = new JLabel(scaledButtonIcon1);

buttonLabel1.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight \* (0.48 + 0.045\*2 + 0.1)), (int) (screenWidth \* 0.4), (int) (screenHeight \* 0.1));

buttonLabel1.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

JFrame frame = new JFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

MovingBackground backgroundPanel;

try {

backgroundPanel = new MovingBackground(10, 3, 20, 60, "3.mp3", 25, frame);

//backgroundPanel = new MovingBackground(10, 3, 20, 60, "3.mp3", 3, frame);

frame.add(backgroundPanel);

frame.pack();

frame.setExtendedState(JFrame.MAXIMIZED\_BOTH);

frame.setVisible(true);

} catch (IOException e1) {

e1.printStackTrace();

}

cleanup();

dispose();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {

dispose();

}

});

add(buttonLabel1);

ImageIcon buttonIcon2 = new ImageIcon("gothird\_repeat.png");

Image button2 = buttonIcon2.getImage();

Image scaledButton2 = button2.getScaledInstance((int) (screenWidth \* 0.4), (int) (screenHeight \* 0.1), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon2 = new ImageIcon(scaledButton2);

JLabel buttonLabel2 = new JLabel(scaledButtonIcon2);

buttonLabel2.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight \* (0.48 + 0.045\*3 + 0.1\*2)), (int) (screenWidth \* 0.4), (int) (screenHeight \* 0.11));

buttonLabel2.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

JFrame frame = new JFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

MovingBackground backgroundPanel;

try {

backgroundPanel = new MovingBackground(10, 2, 40, 70, "2.mp3", 20, frame);

//backgroundPanel = new MovingBackground(10, 2, 40, 70, "2.mp3", 3, frame);

frame.add(backgroundPanel);

frame.pack();

frame.setExtendedState(JFrame.MAXIMIZED\_BOTH);

frame.setVisible(true);

} catch (IOException e1) {

e1.printStackTrace();

}

cleanup();

dispose();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {

//ChooseLevel level = new ChooseLevel(finalScreenWidth1, finalScreenHeight1);

//dispose();

}

});

add(buttonLabel2);

ImageIcon imageIcon = new ImageIcon("second\_back.jpg");

Image image = imageIcon.getImage();

Image scaledImage = image.getScaledInstance(screenWidth, (int) (screenHeight), Image.SCALE\_SMOOTH);

ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

JLabel label = new JLabel(scaledImageIcon);

label.setBounds(0, 0, screenWidth, (int) (screenHeight));

add(label);

setVisible(true);

}

/\*\*

\* Cleanup method to release resources

\*/

public void cleanup() {

removeAll();

screenWidth = 0;

screenHeight = 0;

System.gc();

}

}

## 8.5. Lost.java

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.io.IOException;

import java.util.concurrent.atomic.AtomicReference;

/\*\*

\* Lose game menu class

\* @author Krasovskyy Andrii

\* @author Kalianova Maria

\* @author Tarasenko Misha

\*/

public class Lost extends JFrame {

/\*\*

\* Screen width

\*/

private static int screenWidth;

/\*\*

\* Screen height

\*/

private static int screenHeight;

/\*\*

\* Class constructor

\* @param movingBackground

\*/

public Lost(MovingBackground movingBackground) {

Toolkit toolkit = Toolkit.getDefaultToolkit();

Dimension screenSize = toolkit.getScreenSize();

screenWidth = screenSize.width;

screenHeight = screenSize.height;

GraphicsEnvironment env = GraphicsEnvironment.getLocalGraphicsEnvironment();

GraphicsDevice device = env.getDefaultScreenDevice();

GraphicsConfiguration config = device.getDefaultConfiguration();

Insets insets = Toolkit.getDefaultToolkit().getScreenInsets(config);

int toolbarHeight = insets.top + insets.bottom;

screenWidth = screenSize.width;

screenHeight = screenSize.height - toolbarHeight;

System.out.println(screenWidth + "-" + screenHeight + "-" + toolbarHeight);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(screenWidth, screenHeight);

setLocationRelativeTo(null);

setTitle("pause menu");

setLayout(null);

ImageIcon buttonIcon = new ImageIcon("lose\_repeat.png");

Image button = buttonIcon.getImage();

Image scaledButton = button.getScaledInstance((int) (screenWidth \* 0.4), (int) (screenHeight \* 0.14), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon = new ImageIcon(scaledButton);

JLabel buttonLabel = new JLabel(scaledButtonIcon);

buttonLabel.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight \* (0.2625 + 0.0625)), (int) (screenWidth \* 0.4), (int) (screenHeight \* 0.14));

buttonLabel.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

if(movingBackground.location == 1){

Runtime runtime = Runtime.getRuntime();

long usedMemory = runtime.totalMemory() - runtime.freeMemory();

long maxMemory = runtime.maxMemory();

System.out.println("Використано пам'яті: " + usedMemory / (1024 \* 1024) + " МБ");

System.out.println("Максимально доступна пам'ять: " + maxMemory / (1024 \* 1024) + " МБ");

JFrame frame = new JFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

MovingBackground backgroundPanel;

try {

backgroundPanel = new MovingBackground (10, 1, 60, 80, "1.mp3", 15, frame);

//backgroundPanel = new MovingBackground (10, 1, 60, 80, "1.mp3", 3, frame);

frame.add(backgroundPanel);

runtime = Runtime.getRuntime();

usedMemory = runtime.totalMemory() - runtime.freeMemory();

maxMemory = runtime.maxMemory();

System.out.println("Використано пам'яті: " + usedMemory / (1024 \* 1024) + " МБ");

System.out.println("Максимально доступна пам'ять: " + maxMemory / (1024 \* 1024) + " МБ");

} catch (IOException e1) {

System.out.println("Exception in IO during reload");

e1.printStackTrace();

}

frame.pack();

frame.setExtendedState(JFrame.MAXIMIZED\_BOTH);

frame.setVisible(true);

dispose();

backgroundPanel = null;

System.gc();

} else if(movingBackground.location == 2){

JFrame frame = new JFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

MovingBackground backgroundPanel;

try {

backgroundPanel = new MovingBackground(10, 2, 40, 70, "2.mp3", 20, frame);

//backgroundPanel = new MovingBackground(10, 2, 40, 70, "2.mp3", 3, frame);

frame.add(backgroundPanel);

frame.pack();

frame.setExtendedState(JFrame.MAXIMIZED\_BOTH);

frame.setVisible(true);

} catch (IOException e1) {

e1.printStackTrace();

}

dispose();

}else{

JFrame frame = new JFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

MovingBackground backgroundPanel;

try {

backgroundPanel = new MovingBackground(10, 3, 20, 60, "3.mp3", 25, frame);

//backgroundPanel = new MovingBackground(10, 3, 20, 60, "3.mp3", 3, frame);

frame.add(backgroundPanel);

frame.pack();

frame.setExtendedState(JFrame.MAXIMIZED\_BOTH);

frame.setVisible(true);

} catch (IOException e1) {

e1.printStackTrace();

}

cleanup();

dispose();

}

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {

// ChooseLevel level = new ChooseLevel(finalScreenWidth1, finalScreenHeight1);

// dispose();

}

});

add(buttonLabel);

ImageIcon buttonIcon1 = new ImageIcon("lose\_exit.png");

Image button1 = buttonIcon1.getImage();

Image scaledButton1 = button1.getScaledInstance((int) (screenWidth \* 0.4), (int) (screenHeight \* 0.14), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon1 = new ImageIcon(scaledButton1);

JLabel buttonLabel1 = new JLabel(scaledButtonIcon1);

buttonLabel1.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight \* (0.2625 + 0.0625\*2 + 0.14)), (int) (screenWidth \* 0.4), (int) (screenHeight \* 0.14));

buttonLabel1.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

cleanup();

dispose();

System.exit(0);

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {

dispose();

}

});

add(buttonLabel1);

ImageIcon buttonIcon2 = new ImageIcon("lose\_menu.png");

Image button2 = buttonIcon2.getImage();

Image scaledButton2 = button2.getScaledInstance((int) (screenWidth \* 0.4), (int) (screenHeight \* 0.14), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon2 = new ImageIcon(scaledButton2);

JLabel buttonLabel2 = new JLabel(scaledButtonIcon2);

buttonLabel2.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight \* (0.2625 + 0.0625\*3 + 0.14\*2)), (int) (screenWidth \* 0.4), (int) (screenHeight \* 0.14));

buttonLabel2.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

StartMenu startMenu = new StartMenu();

cleanup();

dispose();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {

// ChooseLevel level = new ChooseLevel(finalScreenWidth1, finalScreenHeight1);

// dispose();

}

});

add(buttonLabel2);

ImageIcon imageIcon = new ImageIcon("lost\_back.jpg");

Image image = imageIcon.getImage();

Image scaledImage = image.getScaledInstance(screenWidth, screenHeight, Image.SCALE\_SMOOTH);

ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

JLabel label = new JLabel(scaledImageIcon);

label.setBounds(0, 0, screenWidth, (int) (screenHeight));

add(label);

setVisible(true);

}

/\*\*

\* Cleanup method to release resources

\*/

public void cleanup() {

removeAll();

screenWidth = 0;

screenHeight = 0;

System.gc();

}

}

## 8.6. MovingBackground.java

import javax.imageio.ImageIO;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.KeyEvent;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

/\*\*

\* GUI class (draws all components, moves them and process animations)

\* @author Krasovskyy Andrii

\*/

public class MovingBackground extends JPanel implements KeyEventDispatcher {

/\*\*

\* Game state

\*/

public boolean end = false;

/\*\*

\* Player for background music

\*/

static MP3Player player;

/\*\*

\* Thread for music

\*/

static Thread playerThread;

/\*\*

\* Background image 1

\*/

private Image backgroundImage1;

private int xCoordinate1;

/\*\*

\* Background image 2

\*/

private Image backgroundImage2;

private int xCoordinate2;

/\*\*

\* Background image 3

\*/

private Image backgroundImage3;

private int xCoordinate3;

/\*\*

\* Character frame width

\*/

private int frameWidth;

/\*\*

\* Character frame height

\*/

private int frameHeight;

/\*\*

\* Run animation frames

\*/

private BufferedImage[] frames;

/\*\*

\* Crouch animation frames

\*/

private BufferedImage[] framesCrouch;

/\*\*

\* Fall animation frames

\*/

private BufferedImage[] framesFall;

/\*\*

\* Run frames counter

\*/

int f = 0;

/\*\*

\* Fall frames counter

\*/

int f2 = 0;

/\*\*

\* Movement state

\*/

private int running = 1;

Timer crouchTimer;

Timer timer3;

/\*\*

\* Crouch frames counter

\*/

private int f1 = 0;

/\*\*

\* Jump counter

\*/

int i = 0;

/\*\*

\* Delta for jump move (OY axis)

\*/

int d = 25;

private int startY;

/\*\*

\* Current delta from start position (OY axis)

\*/

private int currentDeltaY = 0;

private int targetY;

private int direction;

/\*\*

\* Enable jump state

\*/

private boolean enableJump = true;

/\*\*

\* Enable crouch state

\*/

private boolean enableCrouch = true;

/\*\*

\* Character's collision mask x position

\*/

int x;

/\*\*

\* Character's collision mask y position

\*/

int y;

/\*\*

\* Character's collision mask width

\*/

int width;

/\*\*

\* Character's collision mask height

\*/

int height;

int rx = 1700;

int ry = 1700;

Rectangle rect1;

/\*\*

\* Jump frame

\*/

private BufferedImage jump;

private boolean crouch = false;

/\*\*

\* End of crouch (stand up) state

\*/

private boolean standUp = false;

private int c = 0;

private boolean allowEndSound = true;

/\*\*

\* Padding regulation (for collision mask)

\*/

int div = 4;

double widthMaskSetting = 2.5;

/\*\*

\* Padding regulation (for collision mask)

\*/

int paddingStart = 20;

/\*\*

\* Flag for collision

\*/

private boolean collided = false;

/\*\*

\* Flag for finishing fall animation

\*/

private boolean finishFall = false;

/\*\*

\* Contains obstacle

\*/

private Barrier barrier;

/\*\*

\* Image of obstacle

\*/

private BufferedImage barrierFileName;

int location;

private int windowWidth;

private int windowHeight;

/\*\*

\* Filename to load background music

\*/

private static String filename;

private int obstaclesLeft;

private int animationSpeed;

JFrame frame;

boolean isPause = false;

JLabel obstacAmount;

/\*\*

\* Class constructor (starts the GUI and timer animations)

\* @param t

\* @throws IOException

\* @author Krasovskyy Andrii

\*/

public MovingBackground(int t, int location, int obstacleSpeed, int animationSpeed, String filename, int obstaclesLeft, JFrame frame) throws IOException {

MovingBackground.filename = filename;

this.location = location;

this.obstaclesLeft = obstaclesLeft;

this.frame = frame;

// Получаем экземпляр класса Toolkit

Toolkit toolkit = Toolkit.getDefaultToolkit();

// Получаем размер экрана в пикселях

Dimension screenSize = toolkit.getScreenSize();

windowWidth = screenSize.width;

windowHeight = (int) (screenSize.height\*0.95);

GraphicsEnvironment env = GraphicsEnvironment.getLocalGraphicsEnvironment();

GraphicsDevice device = env.getDefaultScreenDevice();

GraphicsConfiguration config = device.getDefaultConfiguration();

Insets insets = Toolkit.getDefaultToolkit().getScreenInsets(config);

//Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

int toolbarHeight = insets.top + insets.bottom;

windowWidth = screenSize.width;

windowHeight = screenSize.height - toolbarHeight;

System.out.println(windowWidth + "-" + windowHeight + "-" + toolbarHeight);

System.out.println("width: " + windowWidth);

System.out.println("height: " + windowHeight);

KeyboardFocusManager manager = KeyboardFocusManager.getCurrentKeyboardFocusManager();

manager.addKeyEventDispatcher(this);

//Встановлюємо розмір кадру анімації персонажа

this.frameWidth = ImageIO.read(new File("run1.png")).getWidth()/10;

this.frameHeight = ImageIO.read(new File("run1.png")).getHeight()/10;

System.out.println(frameWidth + "-" + frameHeight);

//Встановлюємо координати маски колізії та її розмір (для персонажа)

x = paddingStart + frameWidth/div;

y = (int) (windowHeight - this.frameHeight - currentDeltaY + 30);

System.out.println(x);

System.out.println(y);

width = (int) (frameWidth/widthMaskSetting);

height = 350;

barrier = new Barrier(this.location);

barrierFileName = ImageIO.read(new File(barrier.getFileName()));

rx = barrier.getX();

ry = barrier.getY();

rect1 = new Rectangle(rx, ry, barrier.collisionWidth, barrier.collisionHeight);

if(location == 1){

obstacAmount = new JLabel(obstaclesLeft + " / 15");

obstacAmount.setFont(new Font(Font.SERIF, Font.PLAIN, 35));

obstacAmount.setBounds(50, 50 , 100, 50);

add(obstacAmount);

}else if(location == 2){

obstacAmount = new JLabel(obstaclesLeft + " / 20");

obstacAmount.setFont(new Font(Font.SERIF, Font.PLAIN, 35));

obstacAmount.setBounds(50, 50 , 100, 50);

add(obstacAmount);

}else{

obstacAmount = new JLabel(obstaclesLeft + " / 25");

obstacAmount.setFont(new Font(Font.SERIF, Font.PLAIN, 35));

obstacAmount.setBounds(50, 50 , 100, 50);

add(obstacAmount);

}

frames = new BufferedImage[6];

framesCrouch = new BufferedImage[4];

framesFall = new BufferedImage[5];

for (int i = 1; i <= 6; ++i) {

frames[i-1] = ImageIO.read(new File("run"+i+".png"));

}

for (int i = 1; i <= 3; ++i) {

framesCrouch[i-1] = ImageIO.read(new File("sil"+i+".png"));

}

for (int i = 1; i <= 5; ++i) {

framesFall[i-1] = ImageIO.read(new File("fall"+i+".png"));

}

framesCrouch[3] = ImageIO.read(new File("going\_sit.png"));

jump = ImageIO.read(new File("jump.png"));

playerThread = new Thread(() -> {

player = new MP3Player(filename);

player.play();

});

playerThread.start();

//Параметри для стрибка

currentDeltaY = 0;

direction = -15;

switch (this.location) {

case 1: {

backgroundImage1 = new ImageIcon("Game\_city1\_city.png").getImage().getScaledInstance(windowWidth, windowHeight, 0);

backgroundImage2 = new ImageIcon("Game\_city1\_back\_city.png").getImage().getScaledInstance(windowWidth, windowHeight, 0);

backgroundImage3 = new ImageIcon("Game\_city1\_back\_sky.png").getImage().getScaledInstance(windowWidth, windowHeight, 0);

break;

}

case 2:{

backgroundImage1 = new ImageIcon("Game\_suburb1\_city.png").getImage().getScaledInstance(windowWidth, windowHeight, 0);

backgroundImage2 = new ImageIcon("Game\_suburb1\_back\_city.png").getImage().getScaledInstance(windowWidth, windowHeight, 0);

backgroundImage3 = new ImageIcon("Game\_suburb1\_back\_sky.png").getImage().getScaledInstance(windowWidth, windowHeight, 0);

break;

}

case 3:{

backgroundImage1 = new ImageIcon("Game\_village1\_village.png").getImage().getScaledInstance(windowWidth, windowHeight, 0);

backgroundImage2 = new ImageIcon("Game\_village1\_back\_mount.png").getImage().getScaledInstance(windowWidth, windowHeight, 0);

backgroundImage3 = new ImageIcon("Game\_village1\_back\_sky.png").getImage().getScaledInstance(windowWidth, windowHeight, 0);

break;

}

}

xCoordinate1 = 0;

xCoordinate2 = 0;

xCoordinate3 = 0;

//Timer for background movement

Timer timer = new Timer(t, e -> {

if (end == false) {

//Тут ми рухаємо фон

xCoordinate1 -= 3;

xCoordinate2 -= 2;

xCoordinate3 -= 1;

if (xCoordinate1 <= -backgroundImage1.getWidth(null)) {

xCoordinate1 = 0;

}

repaint();

}

});

Timer barrierTimer = new Timer(obstacleSpeed, e -> {

if (end == false) {

rx -= 20;

if (rx <= -250) {

barrier = new Barrier(this.location);

try {

barrierFileName = ImageIO.read(new File(barrier.getFileName()));

this.obstaclesLeft--;

if (this.obstaclesLeft <= 0) {

end = true;

if(location == 1){

GoSecond goSecond = new GoSecond();

this.removeAll();

frame.dispose();

cleanup();

}else if(location == 2){

GoThird goThird = new GoThird();

this.removeAll();

frame.dispose();

cleanup();

}else{

Win win = new Win(this);

this.removeAll();

frame.dispose();

cleanup();

}

System.out.println("END OF LEVEL!!!");

}

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

rx = barrier.getX();

ry = barrier.getY();

rect1 = new Rectangle(rx, ry, barrier.collisionWidth, barrier.collisionHeight);

}

repaint();

}

});

//Timer for running and jumping animations

Timer timer1 = new Timer(animationSpeed, e -> {

if(!end) {

//Here should be your collision check (add checking through masks ArrayList)

//It's quite possible to move check in a separate method

Rectangle other = new Rectangle(rx, ry, barrier.getCollisionWidth(), barrier.getCollisionHeight());

if (getIntersection(other)) {

if (crouchTimer.isRunning()) {

crouchTimer.stop();

}

repaint();

player.stop();

playerThread.interrupt();

collided = true;

if (running == 0) {

f2 = 2;

}

running = 3;

}

//Змінюємо кадри бігу

if (running == 1) {

if(f == 5) {

f = 0;

}

else {

f++;

}

}

if (running == 0) {

f = 0;

}

repaint();

if (running == 3) {

if (f2 < 4) {

f2++;

}

repaint();

if (f2 == 4) {

finishFall = true;

end = true;

if (end == true && allowEndSound == true) {

allowEndSound = false;

playerThread = new Thread(() -> {

player = new MP3Player("e1.mp3");

player.playOneTime();

});

playerThread.start();

Thread playerThread1 = new Thread(() -> {

MP3Player player1 = new MP3Player("e2.mp3");

player1.playOneTime();

allowEndSound = false;

});

playerThread1.start();

Runtime runtime = Runtime.getRuntime();

long usedMemory = runtime.totalMemory() - runtime.freeMemory();

long maxMemory = runtime.maxMemory();

System.out.println("Використано пам'яті: " + usedMemory / (1024 \* 1024) + " МБ");

System.out.println("Максимально доступна пам'ять: " + maxMemory / (1024 \* 1024) + " МБ");

Lost lost = new Lost(this);

runtime = Runtime.getRuntime();

usedMemory = runtime.totalMemory() - runtime.freeMemory();

maxMemory = runtime.maxMemory();

System.out.println("Використано пам'яті: " + usedMemory / (1024 \* 1024) + " МБ");

System.out.println("Максимально доступна пам'ять: " + maxMemory / (1024 \* 1024) + " МБ");

this.removeAll();

frame.dispose();

frame.removeAll();

cleanup();

runtime = Runtime.getRuntime();

usedMemory = runtime.totalMemory() - runtime.freeMemory();

maxMemory = runtime.maxMemory();

System.out.println("Використано пам'яті: " + usedMemory / (1024 \* 1024) + " МБ");

System.out.println("Максимально доступна пам'ять: " + maxMemory / (1024 \* 1024) + " МБ");

}

}

}

if (running == 2 || (running == 3 && finishFall != true)) {

if (i == 3) {

d = 17;

}

else if (i == 6) {

d = 10;

}

else if (i == 9) {

d = 3;

}

else if (i == 12) {

d = -3;

}

else if (i == 15) {

d = -10;

}

else if (i == 18) {

d = -17;

}

else if (i == 21) {

d = -25;

}

else if (i == 24) {

d = 25;

i=0;

currentDeltaY = 0;

if (running == 2) {

enableJump = true;

enableCrouch = true;

}

if(running == 2) {

running = 1;

}

else {

running = 3;

}

currentDeltaY = 0;

//Change collision mask

x = paddingStart + frameWidth/div;

y = (int) (windowHeight - this.frameHeight - currentDeltaY + 30);

System.out.println(x);

System.out.println(y);

width = (int) (frameWidth/widthMaskSetting);

height = 350;

}

currentDeltaY = currentDeltaY + d;

if (running == 2 && i != 24) {

y = (int) (windowHeight - this.frameHeight - currentDeltaY + 30);

}

i++;

}

}

});

//Timer for crouch animations

crouchTimer = new Timer(animationSpeed, e -> {

if(standUp == false) {

if (running == 0 && enableCrouch == true) {

if(f1 == 2 || f1 == 3) {

if (f1 == 3) {

f1 = 2;

}

else if (f1 == 2) {

f1 = 3;

}

}

else {

if (f1 == 0) {

x = paddingStart + frameWidth/div;

y = (int) (windowHeight - this.frameHeight + 30);

System.out.println(x);

System.out.println(y);

width = (int) (frameWidth/widthMaskSetting);

height = 350;

}

if (f1 == 0) {

x = paddingStart + frameWidth/div;

y = (int) (windowHeight - this.frameHeight + 30 + 55);

System.out.println(x);

System.out.println(y);

width = (int) (frameWidth/widthMaskSetting);

height = 350 - 55;

}

if (f1 == 1) {

x = paddingStart + frameWidth/div;

y = (int) (windowHeight - this.frameHeight + 30 + 125);

System.out.println(x);

System.out.println(y);

width = (int) (frameWidth/widthMaskSetting);

height = 350 - 125;

}

repaint();

f1++;

}

}

}

else {

if(f1 == 2 || f1== 3) {

x = paddingStart + frameWidth/div;

y = (int) (windowHeight - this.frameHeight + 30 + 55);

System.out.println(x);

System.out.println(y);

width = (int) (frameWidth/widthMaskSetting);

height = 350 - 55;

f1 = 1;

repaint();

}

else if (f1 == 1) {

x = paddingStart + frameWidth/div;

y = (int) (windowHeight - this.frameHeight + 30);

System.out.println(x);

System.out.println(y);

width = (int) (frameWidth/widthMaskSetting);

height = 350;

f1--;

repaint();

}

else if (f1 == 0) {

running = 1;

crouchTimer.stop();

repaint();

enableJump = true;

enableCrouch = true;

}

}

});

timer.start();

timer1.start();

barrierTimer.start();

}

@Override

/\*\*

\* Key listeners for playing game

\* @author Krasovskyy Andrii

\*/

public boolean dispatchKeyEvent(KeyEvent e) {

if (collided == false && end == false) {

if (e.getID() == KeyEvent.KEY\_PRESSED) {

if (e.getKeyCode() == KeyEvent.VK\_DOWN) {

standUp = false;

enableJump = false;

if (enableCrouch == true) {

running = 0;

crouchTimer.start();

}

}

else if (e.getKeyCode() == KeyEvent.VK\_UP) {

enableCrouch = false;

if (enableJump == true) {

running = 2;

enableJump = false;

enableCrouch = false;

x = paddingStart + frameWidth/div;

y = windowHeight - this.frameHeight + 30;

System.out.println(x);

System.out.println(y);

width = (int) (frameWidth/widthMaskSetting);

height = 280;

}

}

}

if (e.getID() == KeyEvent.KEY\_RELEASED) {

if (e.getKeyCode() == KeyEvent.VK\_DOWN) {

standUp = true;

}

}

}

if (e.getID() == KeyEvent.KEY\_PRESSED) {

if (e.getKeyCode() == KeyEvent.VK\_SPACE && !isPause) {

if(end == true && isPause == true) {

end = false;

}

else {

end = true;

isPause = true;

Pause pause = new Pause(this, frame);

pause.setVisible(true);

if(!end){

try {

Thread.sleep(1000);

} catch (InterruptedException ex) {

throw new RuntimeException(ex);

}

}

}

}

}

return false;

}

@Override

/\*\*

\* Repaints graphics (updates image)

\* @author Krasovskyy Andrii

\*/

protected void paintComponent(Graphics g) {

super.paintComponent(g);

int panelWidth = getWidth();

int panelHeight = getHeight();

startY = panelHeight-this.frameHeight-100;

Image buffer = createImage(panelWidth, panelHeight);

Graphics2D g2d = (Graphics2D) buffer.getGraphics();

g2d.setComposite(AlphaComposite.SrcOver);

for (int x = xCoordinate3; x < panelWidth; x += backgroundImage3.getWidth(null)) {

g2d.drawImage(backgroundImage3, x, 0, panelWidth, panelHeight, null);

}

for (int x = xCoordinate2; x < panelWidth; x += backgroundImage2.getWidth(null)) {

g2d.drawImage(backgroundImage2, x, 0, panelWidth, panelHeight, null);

}

for (int x = xCoordinate1; x < panelWidth; x += backgroundImage1.getWidth(null)) {

g2d.drawImage(backgroundImage1, x, 0, panelWidth, panelHeight, null);

//Малює маску колізії (потім прибрати) (devtools)

/\*g2d.setColor(Color.WHITE);

g2d.fillRect(0, 0, WIDTH, HEIGHT);

g2d.setColor(Color.WHITE);

g2d.fillRect(this.x, y, width, height);\*/

if (running == 1) {

g2d.drawImage(frames[f], 10, windowHeight-this.frameHeight, frameWidth, frameHeight, null);

}

else if (running == 0) {

g2d.drawImage(framesCrouch[f1], 10, windowHeight-this.frameHeight, frameWidth, frameHeight, null);

}

else if (running == 2) {

g2d.drawImage(jump , 10, windowHeight-this.frameHeight - currentDeltaY, frameWidth, frameHeight, null);

}

else if (running == 3) {

if (f2 == 0 || f2 == 1) {

g2d.drawImage(framesFall[f2] , 10, windowHeight-this.frameHeight, (int) (frameWidth\*1.27), frameHeight, null);

}

if (f2 == 2 || f2 == 3) {

g2d.drawImage(framesFall[f2] , 10, windowHeight-this.frameHeight, (int) (frameWidth\*1.45), frameHeight, null);

}

if (f2 == 4) {

g2d.drawImage(framesFall[f2] , 10, windowHeight-this.frameHeight + 245, (int) (frameHeight/1.1), (int) (frameWidth/(1.7\*1.1)), null);

}

}

g2d.drawImage(barrierFileName, rx, ry, rect1.width, rect1.height, null);

remove(obstacAmount);

if(location == 1){

obstacAmount = new JLabel(15 - obstaclesLeft + " / 15");

obstacAmount.setFont(new Font(Font.SERIF, Font.PLAIN, 35));

obstacAmount.setBounds(50, 50 , 150, 50);

add(obstacAmount);

}else if(location == 2){

obstacAmount = new JLabel(20 - obstaclesLeft + " / 20");

obstacAmount.setFont(new Font(Font.SERIF, Font.PLAIN, 35));

obstacAmount.setBounds(50, 50 , 150, 50);

add(obstacAmount);

}else{

obstacAmount = new JLabel(25 - obstaclesLeft + " / 25");

obstacAmount.setFont(new Font(Font.SERIF, Font.PLAIN, 35));

obstacAmount.setBounds(50, 50 , 150, 50);

add(obstacAmount);

}

//Theese two lines are for drawing obstacle collision mask (devtool)

//g2d.setColor(Color.BLACK);

//g2d.fillRect(rx, ry, barrier.getCollisionWidth(), barrier.getCollisionHeight());

}

g.drawImage(buffer, 0, 0, null);

}

@Override

/\*\*

\* Makes the window fit the background size

\* @author Krsovskyy Andrii

\* @return Dimension

\*/

public Dimension getPreferredSize() {

return new Dimension(backgroundImage1.getWidth(null), backgroundImage1.getHeight(null));

}

/\*\*

\* Checks for collision with mask

\* @param other

\* @return boolean

\* @author Krasovskyy Andrii

\*/

public boolean getIntersection (Rectangle other) {

int x1 = Math.max(this.x, other.x);

int y1 = Math.max(this.y, other.y);

int x2 = Math.min(this.x + this.width, other.x + other.width);

int y2 = Math.min(this.y + this.height, other.y + other.height);

if (x2 < x1 || y2 < y1) {

return false;

}

System.out.println("Collision!");

return true;

}

/\*\*

\* Cleanup method to release resources (up to 1 GB)

\*/

public void cleanup() {

player.stop();

playerThread.interrupt();

backgroundImage1 = null;

backgroundImage2 = null;

backgroundImage3 = null;

jump = null;

frames = null;

framesCrouch = null;

framesFall = null;

barrierFileName = null;

rect1 = null;

obstacAmount = null;

frame = null;

System.gc();

}

}

## 8.7. MP3Player.java

import javazoom.jl.player.Player;

import java.io.FileInputStream;

/\*\*

\* MP3 player class

\* @author Krasovskyy Andrii

\*/

public class MP3Player {

private static int i = 0;

private String filename;

private Player player;

boolean end = false;

/\*\*

\* Class constructor

\* @param filename

\*/

public MP3Player(String filename) {

this.filename = filename;

}

/\*\*

\* Plays mp3 track in cycle

\*/

public void play() {

while (!end) {

try {

FileInputStream fis = new FileInputStream(filename);

player = new Player(fis);

player.play();

} catch (Exception e) {

System.out.println("Ошибка при проигрывании файла: " + e);

}

}

}

/\*\*

\* Plays mp3 track

\*/

public void playOneTime() {

try {

FileInputStream fis = new FileInputStream(filename);

player = new Player(fis);

player.play();

} catch (Exception e) {

System.out.println("Ошибка при проигрывании файла: " + e);

}

}

/\*\*

\* Stops the mp3 track

\*/

public void stop() {

if (player != null) {

end = true;

player.close();

}

}

public static void main(String[] args) {

MP3Player player = new MP3Player("ForceMaker.mp3");

player.play();

}

}

## 8.8. Pause.java

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

/\*\*

\* Pause menu class

\* @author Krasovskyy Andrii

\* @author Kalianova Maria

\* @author Tarasenko Misha

\*/

public class Pause extends JFrame {

/\*\*

\* Screen width

\*/

private static int screenWidth;

/\*\*

\* Screen height

\*/

private static int screenHeight;

/\*\*

\* Class constructor

\* @param movingBackground

\* @param frame

\*/

public Pause(MovingBackground movingBackground, JFrame frame) {

movingBackground.setEnabled(false);

Toolkit toolkit = Toolkit.getDefaultToolkit();

Dimension screenSize = toolkit.getScreenSize();

screenWidth = screenSize.width;

screenHeight = screenSize.height;

GraphicsEnvironment env = GraphicsEnvironment.getLocalGraphicsEnvironment();

GraphicsDevice device = env.getDefaultScreenDevice();

GraphicsConfiguration config = device.getDefaultConfiguration();

Insets insets = Toolkit.getDefaultToolkit().getScreenInsets(config);

int toolbarHeight = insets.top + insets.bottom;

screenWidth = screenSize.width;

screenHeight = screenSize.height - toolbarHeight;

System.out.println(screenWidth + "-" + screenHeight + "-" + toolbarHeight);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(screenWidth, screenHeight);

setLocationRelativeTo(null);

setTitle("pause menu");

setLayout(null);

ImageIcon buttonIcon = new ImageIcon("pause\_continue.png");

Image button = buttonIcon.getImage();

Image scaledButton = button.getScaledInstance((int) (screenWidth \* 0.4), (int) (screenHeight \* 0.14), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon = new ImageIcon(scaledButton);

JLabel buttonLabel = new JLabel(scaledButtonIcon);

buttonLabel.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight \* (0.2625 + 0.0625)), (int) (screenWidth \* 0.4), (int) (screenHeight \* 0.14));

buttonLabel.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

movingBackground.setEnabled(true);

movingBackground.end = false;

movingBackground.isPause = false;

cleanup();

dispose();

System.gc();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {

//ChooseLevel level = new ChooseLevel(finalScreenWidth1, finalScreenHeight1);

//dispose();

}

});

add(buttonLabel);

ImageIcon buttonIcon1 = new ImageIcon("pause\_exit.png");

Image button1 = buttonIcon1.getImage();

Image scaledButton1 = button1.getScaledInstance((int) (screenWidth \* 0.4), (int) (screenHeight \* 0.14), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon1 = new ImageIcon(scaledButton1);

JLabel buttonLabel1 = new JLabel(scaledButtonIcon1);

buttonLabel1.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight \* (0.2625 + 0.0625\*2 + 0.14)), (int) (screenWidth \* 0.4), (int) (screenHeight \* 0.14));

buttonLabel1.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

cleanup();

frame.dispose();

dispose();

System.exit(0);

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {

dispose();

}

});

add(buttonLabel1);

ImageIcon buttonIcon2 = new ImageIcon("pause\_menu.png");

Image button2 = buttonIcon2.getImage();

Image scaledButton2 = button2.getScaledInstance((int) (screenWidth \* 0.4), (int) (screenHeight \* 0.14), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon2 = new ImageIcon(scaledButton2);

JLabel buttonLabel2 = new JLabel(scaledButtonIcon2);

buttonLabel2.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight \* (0.2625 + 0.0625\*3 + 0.14\*2)), (int) (screenWidth \* 0.4), (int) (screenHeight \* 0.14));

buttonLabel2.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

StartMenu menu = new StartMenu();

cleanup();

movingBackground.player.stop();

movingBackground.cleanup();

frame.dispose();

dispose();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {

//ChooseLevel level = new ChooseLevel(finalScreenWidth1, finalScreenHeight1);

//dispose();

}

});

add(buttonLabel2);

ImageIcon imageIcon = new ImageIcon("pause\_back.jpg");

Image image = imageIcon.getImage();

Image scaledImage = image.getScaledInstance(screenWidth, screenHeight, Image.SCALE\_SMOOTH);

ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

JLabel label = new JLabel(scaledImageIcon);

label.setBounds(0, 0, screenWidth, (int) (screenHeight\*0.9\*10/9));

add(label);

setVisible(true);

}

/\*\*

\* Cleanup method to release resources

\*/

public void cleanup() {

removeAll();

screenWidth = 0;

screenHeight = 0;

System.gc();

}

}

## 8.9. Settings.java

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

/\*\*

\* Instruction class

\* @author Krasovskyy Andrii

\* @author Kalianova Maria

\* @author Tarasenko Misha

\*/

public class Settings extends JFrame {

/\*\*

\* Screen width

\*/

private static int screenWidth;

/\*\*

\* Screen height

\*/

private static int screenHeight;

/\*\*

\* Class constructor

\*/

public Settings() {

Toolkit toolkit = Toolkit.getDefaultToolkit();

Dimension screenSize = toolkit.getScreenSize();

screenWidth = screenSize.width;

screenHeight = screenSize.height;

GraphicsEnvironment env = GraphicsEnvironment.getLocalGraphicsEnvironment();

GraphicsDevice device = env.getDefaultScreenDevice();

GraphicsConfiguration config = device.getDefaultConfiguration();

Insets insets = Toolkit.getDefaultToolkit().getScreenInsets(config);

int toolbarHeight = insets.top + insets.bottom;

screenWidth = screenSize.width;

screenHeight = screenSize.height - toolbarHeight;

screenWidth = screenSize.width;

screenHeight = screenSize.height - toolbarHeight;

System.out.println(screenWidth + "-" + screenHeight + "-" + toolbarHeight);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(screenWidth, screenHeight);

setLocationRelativeTo(null);

setTitle("settings");

setLayout(null);

ImageIcon buttonIcon = new ImageIcon("instr\_menu.png");

Image button = buttonIcon.getImage();

Image scaledButton = button.getScaledInstance((int) (screenWidth \* 0.38), (int) (screenHeight \* 0.14), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon = new ImageIcon(scaledButton);

JLabel buttonLabel = new JLabel(scaledButtonIcon);

buttonLabel.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight \* (0.03 + 0.115 + 0.067)), (int) (screenWidth \* 0.38), (int) (screenHeight \* 0.14));

System.out.println((int) (screenHeight \* (0.03 + 0.115 + 0.67)));

buttonLabel.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

StartMenu startMenu = new StartMenu();

cleanup();

dispose();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {}

});

add(buttonLabel);

ImageIcon imageIcon = new ImageIcon("intr.jpg");

Image image = imageIcon.getImage();

Image scaledImage = image.getScaledInstance(screenWidth, screenHeight, Image.SCALE\_SMOOTH);

ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

JLabel label = new JLabel(scaledImageIcon);

label.setBounds(0, 0, screenWidth, (int) (screenHeight));

add(label);

setVisible(true);

}

/\*\*

\* Cleanup method to release resources

\*/

public void cleanup() {

removeAll();

screenWidth = 0;

screenHeight = 0;

System.gc();

}

}

## 8.10. StartMenu.java

import java.awt.Dimension;

import java.awt.GraphicsConfiguration;

import java.awt.GraphicsDevice;

import java.awt.GraphicsEnvironment;

import java.awt.Image;

import java.awt.Insets;

import java.awt.Toolkit;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import javax.swing.ImageIcon;

import javax.swing.JFrame;

import javax.swing.JLabel;

/\*\*

\* Start menu class

\* @author Krasovskyy Andrii

\* @author Kalianova Maria

\* @author Tarasenko Misha

\*/

public class StartMenu extends JFrame {

/\*\*

\* Screen width

\*/

private static int screenWidth;

/\*\*

\* Screen height

\*/

private static int screenHeight;

/\*\*

\* Class constructor (set up the GUI)

\*/

public StartMenu() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Toolkit toolkit = Toolkit.getDefaultToolkit();

Dimension screenSize = toolkit.getScreenSize();

screenWidth = screenSize.width;

screenHeight = screenSize.height - 50;

GraphicsEnvironment env = GraphicsEnvironment.getLocalGraphicsEnvironment();

GraphicsDevice device = env.getDefaultScreenDevice();

GraphicsConfiguration config = device.getDefaultConfiguration();

Insets insets = Toolkit.getDefaultToolkit().getScreenInsets(config);

int toolbarHeight = insets.top + insets.bottom;

System.out.println(insets.top + "-" + insets.bottom);

screenWidth = screenSize.width;

screenHeight = screenSize.height - toolbarHeight;

System.out.println(screenWidth + "-" + screenHeight + "-" + toolbarHeight);

setSize(screenWidth, screenHeight);

setLocationRelativeTo(null);

setTitle("Start menu");

setLayout(null);

ImageIcon buttonIcon = new ImageIcon("prev\_start.png");

Image button = buttonIcon.getImage();

Image scaledButton = button.getScaledInstance((int) (screenWidth\*0.26), (int) (screenHeight \* 0.13), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon = new ImageIcon(scaledButton);

JLabel buttonLabel = new JLabel(scaledButtonIcon);

buttonLabel.setBounds((int) (screenWidth/2 - screenWidth\*0.13), (int) (screenHeight/4\*3 + screenHeight \*0.034), (int) (screenWidth\*0.26), (int) (screenHeight\*0.13));

buttonLabel.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

ChooseLevel level = new ChooseLevel(screenWidth, screenHeight);

cleanup();

dispose();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {

}

});

add(buttonLabel);

ImageIcon buttonIcon2 = new ImageIcon("prev\_exit.png");

Image button2 = buttonIcon2.getImage();

Image scaledButton2 = button2.getScaledInstance((int) (screenWidth\*0.14), (int) (screenHeight \* 0.1), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon2 = new ImageIcon(scaledButton2);

JLabel buttonLabel2 = new JLabel(scaledButtonIcon2);

buttonLabel2.setBounds((int) (screenWidth/2 - screenWidth\*0.34), (int) (screenHeight/4\*3 + screenHeight\*0.05), (int) (screenWidth\*0.14), (int) (screenHeight\*0.1));

buttonLabel2.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

cleanup();

dispose();

System.exit(0);

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {}

});

add(buttonLabel2);

ImageIcon buttonIcon3 = new ImageIcon("prev\_settings.png");

Image button3 = buttonIcon3.getImage();

Image scaledButton3 = button3.getScaledInstance((int) (screenWidth\*0.14), (int) (screenHeight \* 0.1), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon3 = new ImageIcon(scaledButton3);

JLabel buttonLabel3 = new JLabel(scaledButtonIcon3);

buttonLabel3.setBounds((int) (screenWidth/2 + screenWidth\*0.2), (int) (screenHeight/4\*3 + screenHeight \* 0.05), (int) (screenWidth\*0.14), (int) (screenHeight\*0.1));

buttonLabel3.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

Settings settings = new Settings();

cleanup();

dispose();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {}

});

add(buttonLabel3);

ImageIcon imageIcon = new ImageIcon("pre.jpg");

Image image = imageIcon.getImage();

Image scaledImage = image.getScaledInstance(screenWidth, screenHeight, Image.SCALE\_SMOOTH);

ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

JLabel label = new JLabel(scaledImageIcon);

label.setBounds(0, 0, screenWidth, screenHeight);

add(label);

setVisible(true);

}

/\*\*

\* Cleanup method to release resources

\*/

public void cleanup() {

removeAll();

screenWidth = 0;

screenHeight = 0;

System.gc();

}

public static void main(String[] args) {

StartMenu frame = new StartMenu();

frame.setVisible(true);

}

}

## 8.11. Win.java

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.io.IOException;

/\*\*

\* Win game menu (finished 3rd level) class

\* @author Krasovskyy Andrii

\* @author Kalianova Maria

\* @author Tarasenko Misha

\*/

public class Win extends JFrame {

/\*\*

\* Screen width

\*/

private static int screenWidth;

/\*\*

\* Screen height

\*/

private static int screenHeight;

/\*\*

\* Class constructor

\* @param movingBackground

\*/

public Win(MovingBackground movingBackground) {

Toolkit toolkit = Toolkit.getDefaultToolkit();

Dimension screenSize = toolkit.getScreenSize();

screenWidth = screenSize.width;

screenHeight = screenSize.height;

GraphicsEnvironment env = GraphicsEnvironment.getLocalGraphicsEnvironment();

GraphicsDevice device = env.getDefaultScreenDevice();

GraphicsConfiguration config = device.getDefaultConfiguration();

Insets insets = Toolkit.getDefaultToolkit().getScreenInsets(config);

int toolbarHeight = insets.top + insets.bottom;

screenWidth = screenSize.width;

screenHeight = screenSize.height - toolbarHeight;

System.out.println(screenWidth + "-" + screenHeight + "-" + toolbarHeight);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(screenWidth, screenHeight);

setLocationRelativeTo(null);

setTitle("pause menu");

setLayout(null);

ImageIcon buttonIcon = new ImageIcon("menu\_final.png");

Image button = buttonIcon.getImage();

Image scaledButton = button.getScaledInstance((int) (screenWidth \* 0.38), (int) (screenHeight \* 0.1), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon = new ImageIcon(scaledButton);

JLabel buttonLabel = new JLabel(scaledButtonIcon);

buttonLabel.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight \* (0.105 + 0.307 + 0.016)), (int) (screenWidth \* 0.38), (int) (screenHeight \* 0.1));

buttonLabel.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

StartMenu startMenu = new StartMenu();

cleanup();

dispose();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {

// ChooseLevel level = new ChooseLevel(finalScreenWidth1, finalScreenHeight1);

// dispose();

}

});

add(buttonLabel);

ImageIcon buttonIcon1 = new ImageIcon("exit\_final.png");

Image button1 = buttonIcon1.getImage();

Image scaledButton1 = button1.getScaledInstance((int) (screenWidth \* 0.38), (int) (screenHeight \* 0.1), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon1 = new ImageIcon(scaledButton1);

JLabel buttonLabel1 = new JLabel(scaledButtonIcon1);

buttonLabel1.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight \* (0.105 + 0.307 + 0.016\*2 + 0.1)), (int) (screenWidth \* 0.38), (int) (screenHeight \* 0.1));

buttonLabel1.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

cleanup();

dispose();

System.exit(0);

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {

cleanup();

dispose();

}

});

add(buttonLabel1);

ImageIcon buttonIcon2 = new ImageIcon("repeat\_final.png");

Image button2 = buttonIcon2.getImage();

Image scaledButton2 = button2.getScaledInstance((int) (screenWidth \* 0.38), (int) (screenHeight \* 0.1), Image.SCALE\_SMOOTH);

ImageIcon scaledButtonIcon2 = new ImageIcon(scaledButton2);

JLabel buttonLabel2 = new JLabel(scaledButtonIcon2);

buttonLabel2.setBounds((int) (screenWidth / 2 - screenWidth \* 0.19), (int) (screenHeight \* (0.105 + 0.307 + 0.016\*3 + 0.1\*2)), (int) (screenWidth \* 0.38), (int) (screenHeight \* 0.1));

buttonLabel2.addMouseListener(new MouseListener() {

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

JFrame frame = new JFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

MovingBackground backgroundPanel;

try {

backgroundPanel = new MovingBackground(10, 3, 20, 60, "3.mp3", 25, frame);

//backgroundPanel = new MovingBackground(10, 3, 20, 60, "3.mp3", 3, frame);

frame.add(backgroundPanel);

frame.pack();

frame.setExtendedState(JFrame.MAXIMIZED\_BOTH);

frame.setVisible(true);

} catch (IOException e1) {

e1.printStackTrace();

}

cleanup();

dispose();

}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {}

});

add(buttonLabel2);

ImageIcon imageIcon = new ImageIcon("win\_back.jpg");

Image image = imageIcon.getImage();

Image scaledImage = image.getScaledInstance(screenWidth, screenHeight, Image.SCALE\_SMOOTH);

ImageIcon scaledImageIcon = new ImageIcon(scaledImage);

JLabel label = new JLabel(scaledImageIcon);

label.setBounds(0, 0, screenWidth, (int) (screenHeight));

add(label);

setVisible(true);

}

/\*\*

\* Cleanup method to release resources

\*/

public void cleanup() {

removeAll();

screenWidth = 0;

screenHeight = 0;

System.gc();

}

}