Природоматематическа гимназия “Акад. Иван Гюзелев” - гр. Габрово

Курсова работа

Тема:
“Разработка и имплементация на играта Tetris в Java”

Разработил: Елизабет Иванова Цуцова
Специалност: Системен програмист

Габрово, 2024

Резюме/Summary:

Тази дипломна работа представя разработката и имплементацията на класическата игра Тетрис с помощта на езика за програмиране Java. Проектът обхваща целия процес от концептуализацията, дизайна и програмирането до тестването и окончателното ѝ завършване. Основната цел е да се създаде функционална, заинтригуваща и визуално привлекателна версия на Тетрис, която може да се играе на съвременни системи. Дипломната работа описва подробно историческия контекст на Тетрис, целта и предимствата на проекта и предоставя задълбочен технически преглед на механиката и имплементирането на играта. Чрез този проект се изследва използването на Java Swing за разработка на графичен потребителски интерфейс (GUI), както и създаването на персонализирани текстури и решаване на проблеми по време на процеса на разработка. Освен това в проектът се обсъждат потенциални бъдещи подобрения по играта.

This thesis presents the development and implementation of the classic game Tetris using the Java programming language. The project encompasses the entire process from conceptualization, design, and coding to testing and final deployment. The primary objective is to create a functional, engaging, and visually appealing version of Tetris that can be played on modern systems. The thesis details Tetris's historical background, the project's purpose and benefits, and provides an in-depth technical overview of the game's mechanics and implementation. Through this project, the use of Java Swing for GUI development is explored, as well as custom texture creation, and problem-solving during the development process. Additionally, the thesis discusses potential future enhancements and improvements that could be made to the game.

Ключови думи/Keywords:

тетрис, тетромино, игрално поле, механика, текстури, игра, решетка, матрица, дъска, проблем, графичен потребителски интерфейс, интерфейс, дизайн, клас, метод, обект

Tetris, GUI, IDE, gameplay, refresh, mute/unmute, Java, Java JDK, Java Swing, Java AWT, preview

Съдържание:

1. Увод

1.1 Исторически контекст и значение

Tetris е една от най-емблематичните видео игри, създавани някога. Неговият прост, но пристрастяващ gameplay пленява милиони играчи по целия свят от самото начало. Играта е създадена от Алексей Пажитнов, руски софтуерен инженер, през 1984 г. Пажитнов проектира Tetris, докато работи в Изчислителния център на Дородницин на Съветската академия на науките в Москва. Неговото вдъхновение идва от традиционна пъзел игра, наречена "пентамино", която включва подреждане на форми, съставени от пет квадрата. Пажитнов опростява концепцията, за да създаде тетроминото, всяко състоящо се от четири квадрата, правейки играта по-достъпна и ангажираща.

Името "Тетрис" произлиза от гръцката дума "тетра", което означава четири, и "тенис", който е любимият спорт на Пажитнов. Играта първоначално се играе на Electronika 60, съветски компютър, но бързо се разпространява на други платформи. През 1986 г. Tetris е лицензиран от Nintendo, което води до пускането му на пазара чрез Game Boy през 1989 г. Тази преносима версия на Tetris се превръща в световна сензация, затвърждавайки мястото си в историята на игрите.

1.2 Цели на проекта

Основната цел на този проект е да разработи и имплементира напълно функционална версия на играта Tetris, използвайки езика за програмиране Java. Това включва не само пресъздаване на основната механика на играта, но и гарантиране, че играта е визуално привлекателна и удобна за потребителя. Чрез използването на Java Swing за графичния потребителски интерфейс, проектът има за цел да осигури гладко и приятно игрово изживяване.

1.2.1 Ползи и образователна стойност за потребителите

Играта Tetris предлага множество предимства за потребителите. Подобрява когнитивните умения като пространствено мислене, решаване на проблеми и координация око-ръка. Играта изисква играчите бързо да анализират и манипулират тетромино, за да ги поставят в най-изгодните позиции, насърчавайки бързото мислене и стратегическото планиране.

От образователна гледна точка разработването на Tetris в Java предоставя ценни възможности за учене. Той позволява на разработчиците да придобият практически опит с програмирането на Java, дизайна на GUI с помощта на Java Swing и принципите за разработка на игри. Проектът също така включва решаване на сложни проблеми и прилагане на ефективни алгоритми, които са основни умения в разработката на софтуер.

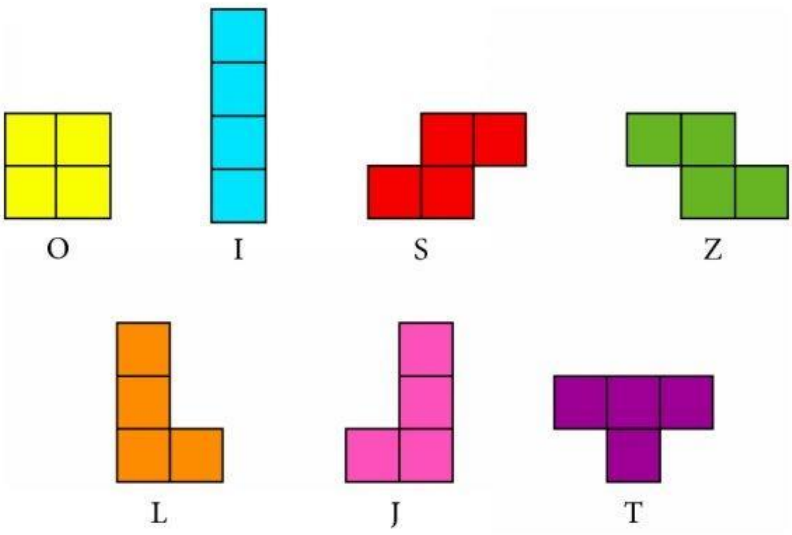
2. Изложение

2.1 Тетромино форми и механика на играта

Tetris се отнася до манипулирането на тетромино - форми, съставени от четири свързани квадрата. Всяко тетромино може да се върти и мести хоризонтално, докато пада на игралното поле. Целта е да се подредят така, че да образуват пълни хоризонтални линии без празни квадрати. Когато линия бъде завършена, тя изчезва и играчът печели точки. Когато тетромино фигури се натрупат до горната част на игралното поле, играта приключва.

В Tetris има седем различни форми на тетромино, всяка с уникална конфигурация:

- I: Права линия от четири квадрата.
- O: Квадрат 2x2.
- T: T-образна форма.
- L: L-образна форма (потенциално може да се асоциира с буквата „Г“ на кирилица).
- J: Огледална L-образна форма.
- S: Зигзагообразна форма.
- Z: Огледална S-образна форма.



Приложение 1: Тетромино форми в Тетрис

Всяка форма може да се завърти на 90 градуса, за да се побере в различни пространства на игралното поле. Играта включва бързо вземане на решения и пространствено ориентиране за ефективно поставяне на тетромино и изчистване на възможно най-много линии.

2.2 Проследяване фазите на развитие на проекта

2.2.1 Цели на проекта и фази на развитие

Основната цел на този проект е да се разработи функционална и визуално привлекателна версия на Tetris, използваща Java. Процесът на разработка е разделен на няколко ключови фази:

- **Планиране и дизайн:** Тази фаза включва дефиниране на изискванията на играта и проектиране на цялостната архитектура, включително механиката на играта и дизайна на потребителския интерфейс.

- **Настройки и инициализация:** Настройване средата за разработка и инициализиране структурата на проекта, включително настройки на комплекта за разработка в Java (JDK) и създаване на първоначалните файлове на проекта.

- **Внедряване на основния gameplay:** Фокус върху разработването на основните механики на играта, като например създаване на тетроминото, игралното поле и логиката за движение и завъртане на формите.

- **Разработка на графичен потребителски интерфейс (GUI):** Внедряване визуалния интерфейс на играта с помощта на Java Swing, включително създаване на прозореца на играта, проектиране на оформлението и интегриране на графични елементи.

- **Персонализирани текстури и подобрения:** Проектиране и включване на персонализирани текстури, за да се подобри визуалната привлекателност на играта и добавяне на фонова музика с цел усъвършенстване на потребителското изживяване.

- **Тестване и отстраняване на грешки:** Провеждане на задълбочено тестване с цел идентифициране и коригиране на грешки, за да се осигури гладко и приятно игрово изживяване.

- **Окончателна имплементация:** Тази фаза включва подготовка на играта за окончателно внедряване, като се гарантира, че тя работи безпроблемно на различни системи и платформи.

2.2.2 Краен преглед на продукта

Крайният продукт е напълно функционална игра Tetris, разработена на Java, включваща:

- **Удобен за потребителя интерфейс:** Изчистен и интуитивен интерфейс, който позволява на потребителите лесно да стартират и да играят играта.

- **Отзивчиви контроли:** Плавни и отзивчиви контроли за преместване и въртене на тетроминото, както и бутоните за пауза, refresh и заглушаване/пускане на музика (mute/unmute).

- **Персонализирани текстури:** Визуално привлекателни текстури и графики, които подобряват цялостното игрово изживяване.

- **Фонова музика:** Въздействаща фонова музика, която допринася за завладяващия gameplay.

- **Система за точкуване:** Система за точкуване, която проследява напредъка на играча по време на всяка игрова сесия, въпреки че резултатите не се записват между сесиите.

Играта успешно възпроизвежда пристрастяващия и завладяващ gameplay на оригиналния Tetris, като същевременно включва модерни подобрения с цел усъвършенстване на потребителското изживяване.

2.3 Техническо изпълнение

2.3.1 Структура на проекта и връзките между класовете

Играта е структурирана с помощта на няколко ключови класа, всеки от които отговаря за различни аспекти на играта:

```
TetrisGame/
├── src/
│   └── pckTetris/
│       ├── Window.java
│       ├── Title.java
│       ├── Board.java
│       ├── Shape.java
│       └── ImageLoader.java
└── textures/
    ├── background.png
    ├── title.png
    ├── arrow.png
    ├── play1.png
    ├── play2.png
    ├── tiles.png
    ├── music.wav
    ├── music1.png
    ├── music2.png
    ├── pause.png
    └── refresh.png
```

- **Window.java:** Настройва основния прозорец на играта и управлява цялостния дисплей.
- **Title.java:** Управлява заглавния екран, който се показва при стартиране на играта.
- **Board.java:** Основният клас, който управлява дъската за игра, обработва поставянето на тетромино и съдържа основната логика на играта.
- **Shape.java:** Дефинира различните форми на тетромино, техните ротации и движения.
- **ImageLoader.java:** Отговаря за зареждането и управлението на графичните ресурси на играта, включително персонализирани текстури.

Тези класове си взаимодействат силно, за да създадат безпроблемно игрово изживяване. Например класът Board използва класа Shape за управление на тетромино и класа ImageLoader за изобразяването им. Класът Window обвързва всичко заедно, като показва игралната дъска и обработва въвеждането на потребителя.

Откъм зависимости играта разчита на стандартната библиотека на Java, по-специално на пакетите java.awt и java.awt.image за изобразяване на графики и обработка на изображения. Графичните елементи на играта се съхраняват в папката textures и са от съществено значение за визуалното изживяване на играта. Управляват се чрез класа ImageLoader който зарежда и организира всички текстури, използвани в играта, а именно:

- **background.png**: Фоновото изображение за играта.
- **title.png**: Изображението, показано на заглавния екран.
- **arrow.png**: Изображението за стрелките за навигация, служещо като упътване.
- **play1.png** и **play2.png**: Изображения за бутона за възпроизвеждане в различни състояния.
- **tiles.png**: Изображението за блоковете тетромино.
- **music1.png** и **music2.png**: Изображения за бутона за музика в различни състояния.
- **pause.png**: Изображението за бутона за пауза.
- **refresh.png**: Изображението за бутона за опресняване.
- **music.wav**: Фоновата музика за играта.

2.3.2 Ролята на матрицата и логиката в играта

В играта Tetris игралното поле или grid (решетка) служи като основната матрица, където се реализират всички взаимодействия на играта. В конкретната реализация класът Board управлява тази матрица, която има фиксиран размер от 20 реда (височина) на 10 колони (ширина).

Класът Board е отговорен за различни задачи, които са от съществено значение за функционалността на играта, включително изобразяване на елементите на играта, обработка на въведени от потребителя данни и актуализиране на състоянието на играта. По-долу е представена част от това как матрицата се използва и управлява в този клас:

```
private final int boardHeight = 20, boardWidth = 10;
private int[][] board = new int[boardHeight][boardWidth];
```

Тук матрицата (игралното поле) се инициализира като решетка 20x10. Всяка клетка в матрицата може да съдържа цяло число, представляващо цвета на блок, или да остане празна (стойност 0).

Когато фигура се движи или завърта, методът за актуализиране в класа Shape проверява за сблъсъци и съответно обновява матрицата, за да отразява текущата позиция на блоковете. Програмният сегмент по-долу гарантира, че когато форма се сблъска и не може да се движи по-нататък, тя се заключва на място в матрицата:

```
if (collision) {
    for (int row = 0; row < coords.length; row++) {
        for (int col = 0; col < coords[0].length; col++) {
            if (coords[row][col] != 0) {
                board.getBoard()[y + row][x + col] = color;
            }
        }
    }
}
```

Методът paintComponent в класа Board отговаря за изчертаването на текущото състояние на матрицата на екрана. Всяка клетка в матрицата, която съдържа ненулева стойност, се изобразява като блок със съответния цвят.

```
for (int row = 0; row < board.length; row++) {
    for (int col = 0; col < board[row].length; col++) {
        if (board[row][col] != 0) {
            g.drawImage(blocks.getSubimage((board[row][col]    -    1)    *
blockSize, 0, blockSize, blockSize), col * blockSize, row * blockSize, null);
        }
    }
}
```

Този цикъл преминава през матрицата и изчертава всеки блок на съответната му позиция на игралното поле.

Методът checkLine в класа Shape проверява за пълни линии, които трябва да бъдат изчистени и отговаря за актуализиране резултата на играча въз основа на броя изчистени линии. Когато редът е завършен, всички блокове в този ред се премахват, а блоковете над него падат надолу, за да запълнят пространството.

```
private void checkLine() {
    int size = board.getBoard().length - 1;
    int linesCleared = 0;

    for (int i = board.getBoard().length - 1; i > 0; i--) {
        int count = 0;
        for (int j = 0; j < board.getBoard()[0].length; j++) {
            if (board.getBoard()[i][j] != 0) {
                count++;
            }
            board.getBoard()[size][j] = board.getBoard()[i][j];
        }
        if (count < board.getBoard()[0].length) {
            size--;
        } else {
            linesCleared++;
        }
    }
```

```
    if (linesCleared > 0) {
        board.addScore(10 * linesCleared);
    }
}
```

Освен за изчертаване на игралното поле, матрицата служи за визуалното представяне на тетроминото в играта, като използва персонализирани текстури, предоставени във файл с изображение (tiles.png). Това изображение съдържа различните типове блокове, всеки със своя уникален цвят.

Приложение 2: Ресурсът tiles.png

Текстурата се зарежда в играта с помощта на класа ImageLoader, който управлява зареждането на ресурсите на изображението.

```
blocks = ImageLoader.loadImage("/tiles.png");
```

На всяка форма тетромино се присвоява част от текстурата, представляваща нейния цвят. Например I-образното тетромино използва първия блок от текстурата, T-образното тетромино втория и т.н.:

```
shapes[0] = new Shape(new int[][] {
 { 1, 1, 1, 1 } },
 blocks.getSubimage(0, 0, blockSize, blockSize), this, 1);

shapes[1] = new Shape(new int[][] {
 { 1, 1, 1 },
 { 0, 1, 0 }},
 blocks.getSubimage(blockSize, 0, blockSize, blockSize), this, 2);
```

Тези матрици не само диктуват подреждането на блоковете във всяка форма, но също така улесняват ротацията и процесите на изобразяване. Въртенето на тетромино в Tetris включва завъртане на фигура около центъра ѝ, за да се промени ориентацията ѝ, като същевременно се гарантира, че се вписва в игралното поле (матрицата). Процесът на ротация на фигурите се състои от три основни стъпки:

- **Транспониране на матрицата:**

За да се завърти фигурата, първо трябва да се транспонира нейната матрица. Транспонирането включва обръщане на матрицата върху нейния диагонал, което превръща редовете в колони.

```
for (int y = 0; y < coords.length; y++) {
    for (int x = 0; x < coords[0].length; x++) {
        rotatedMatrix[x][y] = coords[y][x];
    }
}
```

➢ **Обръщане на редовете (завъртане на 90 градуса по часовниковата стрелка):** След транспониране, обръщането на всеки ред от транспонираната матрица води до завъртане на 90 градуса по часовниковата стрелка.

```
for (int y = 0; y < rotatedMatrix.length; y++) {
    for (int x = 0; x < rotatedMatrix[y].length / 2; x++) {
        int temp = rotatedMatrix[y][x];
        rotatedMatrix[y][x] = rotatedMatrix[y][rotatedMatrix[y].length - x
- 1];
        rotatedMatrix[y][rotatedMatrix[y].length - x - 1] = temp;
    }
}
```

➢ **Проверка за сблъсък:**

```
if (!collision(x, y, rotatedMatrix)) {
    coords = rotatedMatrix;
}
```

Преди да приложи новата ориентация, кодът проверява дали завъртяната фигура ще се сблъска със съществуващи/поставени блокове или границите на решетката. Ако няма сблъсък, координатите (coords) на формата се актуализират до новото завъртяно състояние.

Ако фигура е близо до краищата на решетката, завъртането ѝ може да доведе до излизането ѝ извън границите на игралното поле. Чрез булевата променлива collision се гарантира, че подобни завъртания са предотвратени и фигурата няма да се завърти, ако е доведена до невалидна позиция.

Всяка фигура има уникална структура и техните въртения може леко да се различават, особено за формата O (квадрат), която не се променя при въртене. Кодът може да поеме тези вариации, като използва специфична логика, пригодена за всеки тип форма.

Чрез прилагането на тази логика, играта гарантира, че формите се въртят правилно в рамките на ограниченията на матрицата, като същевременно се избягват сблъсъци с други блокове или границите на решетката.

2.3.3 Системни изисквания

За да се стартира играта Tetris трябва компютърът да отговаря на необходимите системни изисквания и да има инсталиран подходящ софтуер. По-долу има подробно ръководство за системните изисквания и зависимости.

Хардуерни изисквания:

- **Процесор:** Intel Core i3 или еквивалентен (минимум); Intel Core i5 или по-висок (препоръчително)
- **RAM:** 4 GB (минимум); 8 GB или повече (препоръчително)
- **Памет:** Поне 100 MB свободно дисково пространство
- **Графика:** Интегрирана графична карта (минимум); специална графична карта за по-добра производителност (препоръчително)
- **Операционна система:** Windows 7/8/10, macOS 10.12 или по-нова версия или всяка модерна Linux дистрибуция

Софтуерни изисквания:

- **Java Development Kit (JDK):** JDK 8 или по-нова
- **Интегрирана среда за разработка (IDE):** IntelliJ IDEA, Eclipse или NetBeans (по избор)

2.3.4 Софтуерно осигуряване

За разработването на проекта TetrisGame е използван Eclipse, широко използвана интегрирана среда за разработка (IDE) за програмиране на Java. Именно тази платформа е избрана, заради обширната поддръжка за разработка на Java, включително мощни инструменти за отстраняване на грешки, допълване на код и подчертаване на синтаксиса. Това улеснява писането, тестването и отстраняването на грешки. Eclipse има огромна колекция от плъгини, които могат да разширят неговата функционалност. Тази гъвкавост позволява допълнително интегриране на инструменти и библиотеки без да се налага превключване към различни софтуерни среди. IDE включва набор от интегрирани инструменти за контрол на версиите, автоматизация на изграждането и управление на проекти. Това гарантира, че при работния процес може ефективно да се управлява жизнения цикъл на проекта от програмиране и тестване до внедряване. Eclipse има силна общност и обширна документация, предоставяйки достатъчно ресурси за отстраняване на проблеми и обучение, което е особено полезно за преодоляване на предизвикателствата по време на процеса на разработка. Софтуерът също е съвместим с множество операционни системи, включително Windows, macOS и Linux, като по този начин разширява достъпа до разнообразен кръг потребители и увеличава обхвата си в рамките на потенциалния потребителски пазар. Тези характеристики заедно допринасят за ефикасен и ефективен процес на разработка.

2.3.5 Подробна реализация в Java с помощта на Swing и AWT

При разработването на играта Tetris изборът на Java Swing и Abstract Window Toolkit (AWT) се ръководи от няколко ключови фактора, които са в съответствие както с техническите изисквания, така и с целите на проекта. Swing и AWT са неразделна част от Java Foundation Classes (JFC), осигурявайки стабилна рамка за изграждане на графични потребителски интерфейси (GUI) в Java приложения.

Swing предлага широка гама от леки, независими от платформата GUI компоненти. Това включва бутони, етикети, текстови полета, таблици, trees и други, което позволява създаването на силно интерактивен потребителски интерфейс. Например JPanel, използван в класовете Title и Board, служи като контейнер за други компоненти и персонализирани изображения.

AWT осигурява основни градивни елементи за създаване на GUI приложения, като например възможности за обработка на събития (event handling), рисуване и манипулиране на изображения. Компоненти като Graphics и Color се използват за изобразяване на елементите на играта.

Методът paintComponent на Swing, наследен от класовете Title и Board, позволява персонализирано изобразяване на компоненти. Това е от решаващо значение за играта Tetris, която изисква динамично актуализиране и изчертаване на части от играта, фон и интерактивни бутони.

Както Swing, така и AWT предлагат стабилни механизми за обработка на събития, които са от съществено значение за интерактивни приложения като игри. Интерфейсите MouseListener и MouseMotionListener, внедрени в класове Title и Board, улесняват взаимодействието на потребителите чрез събития с мишката. Интерфейсът KeyListener в класа Board следи клавиатурата, позволявайки на играчите да контролират играта с помощта на клавишите стрелки.

Класът javax.swing.Timer се използва за управление на игрови цикли и анимации, осигурявайки прост, но ефективен начин за планиране на повтарящи се задачи. В класа Title се използва таймер за периодично извикване на repaint(), като се гарантира, че интерфейсът остава отзивчив и актуализиран. По същия начин, в класа Board, Timer задвижва основния цикъл на играта, извиквайки методите update() и repaint() за прогресиране на състоянието на играта.

Swing и AWT улесняват лесното зареждане и манипулиране на изображения и аудио. Класът ImageIO се използва в класовете Title и Board за зареждане на изображения за игрови активи като заглавия, бутони и фигури от играта. Относно музиката класът Clip от пакета javax.sound.sampled обработва фонова музика и звукови ефекти, подобрявайки игровото изживяване. Класът ImageLoader също включва методи за зареждане и контрол на аудио клипове.

Независимият от платформата характер на Java гарантира, че играта Tetris може да работи безпроблемно на различни операционни системи без модификация. Тази широка съвместимост е значително предимство за образователни инструменти и игри, предназначени за различни потребителски бази.

2.3.6 Създаване и интегриране на персонализирани текстури

Текстурите имат жизненоважна роля за подобряване на визуалната привлекателност и потребителското изживяване на играта. Тази част от проекта се възползва от възможностите на Adobe Photoshop CS6 и Canva за проектиране на различни графични елементи и бутони, необходими за интерфейса на играта. Canva, удобен за потребителя инструмент за графичен дизайн, е използван за създаване на първоначалния дизайн на всички текстури и графични елементи. Неговата обширна библиотека от шаблони и елементи улеснява бързото създаване на визуално привлекателни ресурси. Adobe Photoshop CS6 е използван за обработка на създадените текстури. Този мощен софтуер за редактиране на изображения позволява прецизност и предоставя усъвършенствани инструменти за подобряване на качеството и детайлите на интерактивните елементи (бутони).

Бутонът за стартиране на играта има 2 състояния: нормално състояние (когато потребителят не взаимодейства с него), когато курсорът на мишката е върху него, предоставяйки визуален сигнал за интерактивност.

Приложение 3: Ресурс play1.png – нормално състояние

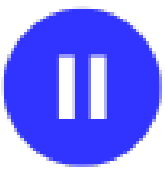
Приложение 4: Ресурс play2.png – при взаимодействие

Бутонът за изключване/включване на звука действа подобно на бутона за стартиране на играта.

Приложение 4: Ресурс music1.png - mute

Приложение 4: Ресурс music1.png - unmute

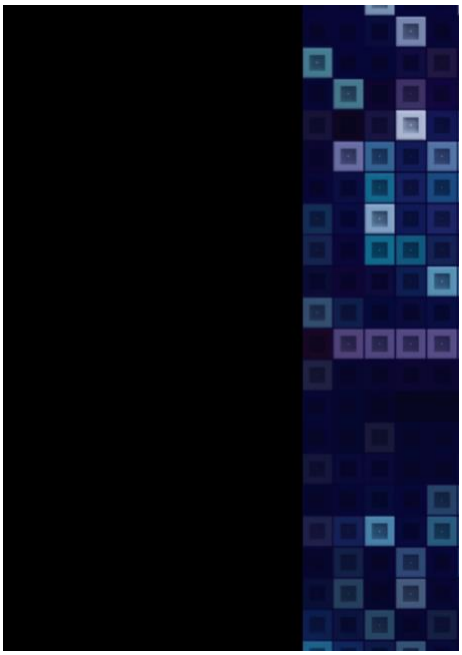
Бутонът за пауза позволява на играча временно да спре играта, а **бутонът за опресняване** нулира играта, предоставяйки на потребителя нов старт.



Приложение 5: Ресурс pause.png - пауза

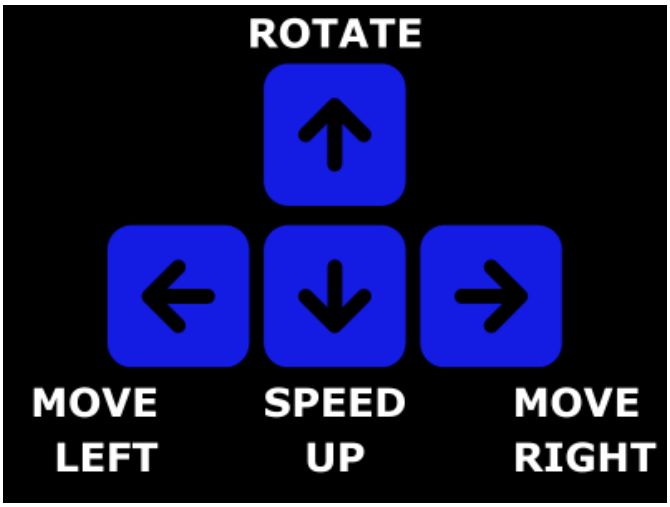
Приложение 6: Ресурс refresh.png - опресняване

Фоновото изображение се придържа строго към темата на играта и допринася за настроението на потребителя.



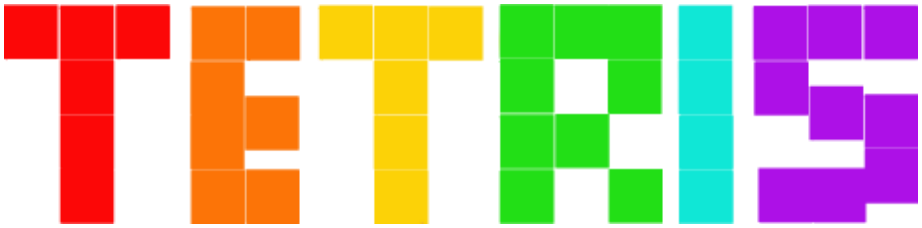
Приложение 7: Ресурс background.png – фоновото изображение на играта в умален размер

Инструктивното изображение представлява бързо упътване, което улеснява играча. Тази текстура се появява в началото на играта, показваща основните контроли и механики на играта.



Приложение 8: Ресурс arrow.png – упътването в умален размер

Заглавието на играта е създадено с помощта на персонализиран шрифт от FontSpace, уебсайт, предлагащ голямо разнообразие от безплатни шрифтове. Този уникален дизайн на заглавието помага за брандирането на играта и изпъкването ѝ.



Приложение 9: Ресурс title.png – заглавието на играта

2.3.7 Проблеми и решения

При разработването на играта Tetris се срещат няколко предизвикателства, които изискват обмислени решения. По-долу е представено подробно описание на проблемите и съответните решения, внедрени, за да се осигури гладко и приятно игрово изживяване.

- **Проблем с подравняването на игралното поле (решетката):**

Проблем: Първоначално матрицата на играта не се подравняваше идеално в решетка 20x10, което причиняваше проблеми с оформлението. Това неправилно подравняване доведе до изобразяване на блокове извън определените им клетки, нарушавайки визуалната последователност и геймплея.

Решение: Проблемът беше разрешен, като се гарантира, че размерите на матрицата съответстват на дефинираната височина и ширина на дъската. Логиката на изобразяване на мрежата в метода paintComponent беше внимателно приведена в съответствие с размерите на матрицата. Това включваше проверка дали всеки блок е позициониран точно в клетките на решетката, като по този начин се гарантира, че матрицата на играта съответства точно на структурата 20x10.

- **Непълно оцветяване на тетроминото:**

Проблем: Фигурите не бяха напълно оцветени, когато бяха показани на игралното поле, което доведе до частична или неправилна визуализация на тетроминото.

Решение: Този проблем беше разрешен чрез интегриране на Java Swing и AWT за изобразяване, заедно с използване на персонализирани текстурни плочки (tiles.png). Чрез зареждането на тези текстури и картографирането им правилно върху формите беше постигнато пълно оцветяване. Изображението на блоковете беше правилно начертано в метода paintComponent и вградения метод Shape.render. Това гарантира, че всяко тетромино се показва с пълен и точен цвят, подобрявайки визуалната привлекателност на играта.

- **Неправилно завъртане на формата:**

Проблем: Първоначалната логика на завъртане беше погрешна, което караше формите да се завъртят само веднъж или да не се въртят напълно. Това възпрепятства геймплея, тъй като играчите не могат да маневрират ефективно с фигурите.

Решение: Алгоритъмът за ротация беше подобрен чрез внедряване на транспониране на матрица и обръщане в метода rotateShape. Този метод гарантира, че формите се въртят правилно в границите на мрежата, като проверява за сблъсъци и грешки извън границите, преди да финализира въртенето:

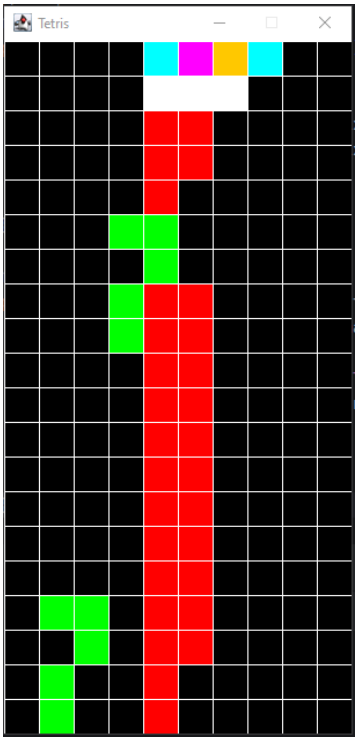
```
public void rotateShape() {
    int[][] rotatedShape = transposeMatrix(coords);
    rotatedShape = reverseRows(rotatedShape);
    if((x + rotatedShape[0].length > 10) || (y + rotatedShape.length > 20))
{
        return;
    }
    for(int row = 0; row < rotatedShape.length; row++) {
        for(int col = 0; col < rotatedShape[row].length; col ++) {
            if(rotatedShape[row][col] != 0) {
                if(board.getBoard()[y + row][x + col] != 0) {
                    return;
                }
            }
        }
    }
    coords = rotatedShape;
}
```

- **Остатъчни позиции на тетроминото:**

Проблем: След завъртане на фигура, старата ѝ позиция не беше изчистена, което доведе до припокриващи се блокове и разхвърляна игрална дъска.

Решение: Гарантирането, че матрицата на игралното поле се актуализира правилно след всяко движение или завъртане, реши този проблем. Методът за актуализиране в класа Shape проверява за сблъсъци и заключва формата на място, актуализирайки по подходящ начин матрицата:

```
if (collision) {
    for (int row = 0; row < coords.length; row++) {
        for (int col = 0; col < coords[0].length; col++) {
            if (coords[row][col] != 0) {
                board.getBoard()[y + row][x + col] = color;
            }
        }
    }
    checkLine();
    board.addScore(1);
    board.setCurrentShape();
}
```



Приложение 10: Илюстриране на проблема с остатъчните позиции при завъртане на тетроминото

- **Контрол на паузата на играта:**

Проблем: Дори когато играта беше поставена на пауза, играчите все още можеха да местят тетроминото с помощта на клавишите, нарушавайки предвидената функционалност за паузата.

Решение: Методите keyPressed и keyReleased бяха модифицирани, за да игнорират входове, когато играта е на пауза или е приключила. Това гарантира, че не могат да се извършват движения по време на тези състояния:

```
@Override
public void keyPressed(KeyEvent e) {
    if (gamePaused || gameOver) {
        return;
    }
    // останалия код
}
```

- **Покрити съобщения:**

Проблем: Съобщенията „Game Over“ и „Game Paused“ бяха скрити от решетката, което затрудняваше играчите да виждат тези важни известия.

Решение: Коригирането на реда на рисуване в метода paintComponent гарантира, че тези съобщения се изобразяват отгоре. Чрез задаване на свойствата на цвета и шрифта по подходящ начин и изчертаване на съобщенията последни, тяхната видимост беше подобрена:

```
if (gamePaused) {
    g.setColor(Color.WHITE);
    g.setFont(new Font("Verdana", Font.BOLD, 30));
```

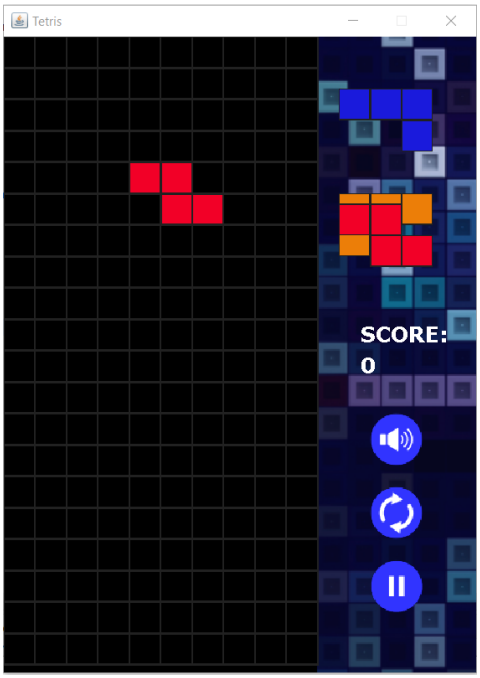
```
    g.drawString("GAME PAUSED", 35, Window.HEIGHT / 2);
}
if (gameOver) {
    g.setColor(Color.WHITE);
    g.setFont(new Font("Verdana", Font.BOLD, 30));
    g.drawString("GAME OVER", 50, Window.HEIGHT / 2);
}
```

- **Визуализация (preview) на следващите фигури:**

Проблем: Използването на Math.random за визуализации на следващите фигури предизвика объркване, тъй като формите се превключваха непредвидимо, което доведе до липса на яснота относно предстоящите форми.

Решение: Въвеждане на променливи nextShape1 и nextShape2, разрешени за последователни визуализации. Методите setNextShapes и setCurrentShape управляват тези визуализации, като гарантират, че следващите форми са избрани по контролиран и предвидим начин:

```
public void setNextShapes() {
    int index1 = (int) (Math.random() * shapes.length);
    int index2 = (int) (Math.random() * shapes.length);
    while (index2 == index1) {
        index2 = (int) (Math.random() * shapes.length);
    }
    nextShape1 = new Shape(shapes[index1].getCoords(),
shapes[index1].getBlock(), this, shapes[index1].getColor());
    nextShape2 = new Shape(shapes[index2].getCoords(),
shapes[index2].getBlock(), this, shapes[index2].getColor());
}
```



Приложение 11: Илюстриране на проблема с preview на следващите фигури

- **Механизъм за актуализиране на резултата:**

Проблем: Методът addScore не награждаваше адекватно играчите за изчистване на линии, което можеше да ги демотивира.
Решение: Механизмът за точкуване беше подобрен, като сега добавя по 10 точки за всяка изчистена линия. Това е приложено в метода checkLine, който отчита броя на изчистените линии и съответно актуализира резултата:

```
private void checkLine() {
    int size = board.getBoard().length - 1;
    int linesCleared = 0;
    for (int i = board.getBoard().length - 1; i > 0; i--) {
        int count = 0;
        for (int j = 0; j < board.getBoard()[0].length; j++) {
            if (board.getBoard()[i][j] != 0) {
                count++;
            }
            board.getBoard()[size][j] = board.getBoard()[i][j];
        }
        if (count < board.getBoard()[0].length) {
            size--;
        } else {
            linesCleared++;
        }
    }
    if (linesCleared > 0) {
        board.addScore(10 * linesCleared);
    }
}
```

- **Проблем с ширината на панела:**

Проблем: Задаването на ширината на панела на нечетно число доведе до появяване на бяла линия от дясната страна, поради изчисления за изобразяване, включващи деление.
Решение: Задаването на ширината на панела на четно число (466) разреши проблема. Това вероятно се дължи на изчисленията за позициониране на изображенията, изискващи деление на 2, което създава проблеми с нечетните числа. Осигуряването на ширината предотврати всякакви проблеми с частични пиксели, като по този начин елиминира нежеланата бяла линия.

Чрез справянето с тези проблеми функционалността на играта и потребителското изживяване са значително подобрени. Всяко решение е внедрено чрез усъвършенстване на логиката на играта и техниките за изобразяване.

2.4 Бъдещи планове за подобрение

Текущата реализация на играта Tetris поставя солидна основа с базисни механики и функционален потребителски интерфейс. Няколко подобрения обаче могат допълнително да подобрят производителността на играта и потребителското изживяване.

Замяната на текущите текстури с по-детайлни и завладяващи, включително графики с по-висока разделителна способност и по-сложни дизайнерски елементи, може да създаде визуално привлекателна игрова среда. Подобрените текстури могат да направят играта по-ангажираща, давайки на играчите по-богато и по-завладяващо изживяване.

Друго подобрение е разширяването на панела на играта, за да включва определена зона, където играчите могат да запазват тетромино за по-късна употреба. Тази функция, често срещана в много версии на Tetris, позволява на играчите да съхраняват временно фигура и да я разменят, когато е необходимо. Добавянето на този механизъм придава стратегическа дълбочина на играта, позволявайки повече тактически решения, и е по-привлекателен за опитни играчи, които обичат стратегически предизвикателства.

Система за класиране, която проследява и показва най-добрите резултати на играчите, може да въведе състезателен елемент в играта. Това може да включва както местни класации за резултати, постигнати на едно и също устройство, така и онлайн класации за резултати, споделени между множество играчи по целия свят. Системата за класиране може да мотивира играчите да подобрят своите резултати и да се изкачат в класацията, насърчавайки участието на общността и чувството за конкуренция и постижение.

Добавянето на усъвършенствани звукови ефекти за допълване на съществуващата фонова музика може значително да подобри цялостното игрово изживяване. Това включва звукови ефекти за движение на фигури, завъртания и изчистване на линии. Висококачествените звукови ефекти могат да направят играта по-завладяваща и приятна, като предоставят звукова обратна връзка на играчите и подобряват дейността им по време на игра.

Предлагането на играчите на опцията да избират различни теми за игралното поле и тетромино могат допълнително да персонализират игровото изживяване. Това може да включва различни цветови схеми, фонови изображения и блокови дизайни. Персонализираните теми позволяват на играчите да приспособят игровата си среда към своите предпочитания, като повишават удовлетворението и насладата, и се харесват на по-широка аудитория с разнообразни естетически вкусове.

Разработването на изчерпателен tutorial и раздел за помощ, който обяснява подробно механиката, контролите и стратегиите на играта, може да направи играта по-достъпна за нови играчи. Предоставянето на насоки и стратегии може да помогне на играчите да подобрят своя gameplay и да се насладят на играта по-пълноценно.

3. Заключение

Разработването на проекта за игра Tetris е успешно начинание, което води до напълно функционално приложение, демонстрирайки както технически умения, така и творческо решаване на проблеми. Проектът има за цел да пресъздаде класическата игра Tetris, като същевременно осигури образователна стойност чрез процеса на разработване. Тази цел е постигната чрез внимателно планиране, проектиране и изпълнение.

3.1 Обобщение на ключовите постижения

Едно от ключовите постижения на проекта е създаването на напълно работеща Tetris игра, използваща Java, включваща Swing и AWT за графичния потребителски интерфейс. Играта включва начален екран, функционална игрална дъска и отзивчиви бутони, допринасящи за потребителското изживяване. Интегрирани са персонализирани текстури и графики, за да се подобри визуалната привлекателност на играта.

Техническото изпълнение се фокусира върху осигуряването на модулност и поддръжка. Класовата структура на играта и взаимовръзките между различните компоненти са проектирани с това предвид. Матрицата и логиката на играта, включително обработката на тетромино, откриване на сблъсък и изчисляване на резултата, са внедрени, за да възпроизведат класическия gameplay на Tetris.

По време на процеса на разработка са срещнати няколко предизвикателства, които са успешно разрешени. Те включват проблеми, свързани с графично изобразяване, управление на състоянието на играта и внедряване на звукови ефекти. Решенията на тези проблеми подобряват устойчивостта на играта и предоставят представа за често срещани проблеми при разработването на игри.

Проектът също така идентифицира потенциални области за бъдещи подобрения, като добавяне на разширени функции и допълнително оптимизиране на кода на играта. Тези предложения предоставят готов план за непрекъснато развитие и подобряване на играта.

Този проект ефективно прилага уменията за програмиране и способностите за решаване на проблеми, за да създаде функционална и увлекателна игра. Ключовите постижения на проекта включват разработването на играта, преодоляването на техническите предизвикателства и идентифицирането на бъдещи подобрения, отразяващи успеха на проекта и потенциала за продължаващо развитие.

3.2 Източници

Разработването на този проект за игра Tetris използва различни свободно достъпни ресурси. По-долу е изчерпателен списък на материалите, използвани в проекта:

- 24hourfonts - Дейвид Мартин „Tetris Blocks“: персонализирани шрифтове, използвани за играта.
- Статия в Уикипедия за Алексей Пажитнов: Подробна основна информация за създателя на Tetris.
- Уроци за разработка на Java игри от Jade Engineer: Изчерпателен плейлист в YouTube за техники за разработка на Tetris с Java.
- Урок за разработка на игри Tetris от RyiSnow: Конкретен видео урок за създаване на игра Tetris в Java.
- Уроци по програмиране и разработка на Java игри от FoX DeN: Друг ценен плейлист в YouTube за изучаване на Java и разработка на игри.
- CopyAssignment.com „Игра Tetris в Java“: Онлайн урок, описващ подробно стъпките за създаване на игра Tetris с помощта на Java.
- „Tetris в Java“ на Zetcode: Практическо ръководство за изграждане на Tetris в Java.
- Икони на Canva от Google Design Icons - Икони, използвани в интерфейса на играта, произхождащи от библиотеката на Canva.
- Бутон за начало на играта от @sketchify в Canva - Елементи на дизайна, използвани в играта, получени от Canva и създадени от потребителя @sketchify.
- Фоново изображение от @mediawhalestock в Canva - Снимка, използвана като част от фона на играта, получена от Canva и създадена от потребител @mediawhalestock.
- Музика: aespa - "Hold on Tight" Instrumental - Фонова музика, използвана в играта, инструментална версия от aespa.