**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ**

Курсовая работа

по дисциплине «Технологии и методы программирование»

на тему

«Dungeon Sokoban»

Группа: АВТ-043

Факультет: АВТФ

Студент: Сивирилова М.К.

Преподаватель: Дубков И.С.

Новосибирск 2022

**Оглавление**

[1. Введение 3](#_Toc105000615)

[2. Основная часть 4](#_Toc105000616)

[Проектирование программы 4](#_Toc105000617)

[Игровое поле 5](#_Toc105000618)

[Игровые элементы 6](#_Toc105000619)

[Главный герой 8](#_Toc105000620)

[Анимация 10](#_Toc105000621)

[Интерфейс 12](#_Toc105000622)

[Главное меню 17](#_Toc105000623)

[Игровой процесс 19](#_Toc105000624)

[Таблица рекордов 21](#_Toc105000625)

[Соревновательный режим 25](#_Toc105000626)

[Музыка 32](#_Toc105000627)

[3. Заключение 33](#_Toc105000628)

[4. Приложение 34](#_Toc105000629)

Введение

Сокобан – это двухмерная компьютерная игра-головоломка, в которой игроку необходимо расставить ящики по обозначенным местам лабиринта. Кладовщик одновременно может двигать только один ящик, толкая его вперёд.

Игра Sokoban, от которой пошло название жанра, была создана в 1981 году и издана в 1982 году в Японии. С тех пор концепцию повторяли множество раз, иногда привнося уникальные геймплейные механики. Однако суть оставалась прежней: игрок должен сдвинуть преграды на своем пути в правильном порядке, чтобы достигнуть цели.

В данной работе реализован не классический сокобан. За основу взята игра Helltaker 2020 года выпуска. Кроме ящиков в ней также реализованы механики врагов, шипов, ключа и сундука. Целью же является не поставить ящики на нужные точки, а добраться героем до определенной точки.

**Цель работы:**

Создать игру в жанре сокобан на Java, используя технологии ООП, базы данных, клиент-сервера.

**Задания:**

1. Реализовать core-механики игры:
   1. создание игрового поля;
   2. добавление объектов на игровое поле;
   3. взаимодействие игрока с героем, а героя с объектами;
2. Создать несколько уровней, отражающих возможности игры;
3. Создать главное меню;
4. Реализовать таблицу рекордов при помощи базы данных;
5. Реализовать соревновательный режим клиент-сервером.

Основная часть

Проектирование программы

Для визуализации используем библиотеку JavaFX, FXML-верстку и CSS-стили.

Запуск программы будет происходить в основном классе Client. Взаимодействие клиента и сервера будут происходить во внешнем классе Launcher. Все игровые механики, в том числе смена сцен главного меню и уровней, реализуем в классе LvlChanger.

Для смены сцен создадим два FXML-файла – один для главного меню, один для игрового поля. В каждом создадим несколько невидимых Pane, вызывающихся по нажатию на кнопку или клавишу. Таким образом, не нужно каждый раз генерировать окна паузы и победы, достаточно сделать их видимыми для игрока.

Взаимодействие между FXML и Java производятся при помощи контроллеров, в которых прописываются все элементы FXML и их взаимодействие с пользователем. Каждый FXML получит свой контроллер, чтобы не было конфликтов при вызове методов.

Звук реализуем в отдельном классе Sounds, где будут храниться и запускаться треки. Реализуем его при помощи MediaPlayer и Media.

Создание игрового поля будет происходить динамически из набора блоков по заранее заданным картам в отдельном файле. Таким же образом будут добавляться элементы, такие как ящики, щипы, сундуки и прочее. Все элементы, реализуемые либо при помощи ImageView, либо созданным классов Block.java, также добавляются в соответствующие списки, чтобы можно было их сдвинуть или уничтожить героем.

Герой, реализуемый классов Hero.java, спавнится как и другие элементы. Он подконтролен игроку, может перемещаться по свободным клеткам, сдвигать одиночные ящики перед собой, толкать врагов и уничтожать их о стены, терять ходы, попадая на шипы, подбирать ключ, отпирать ящик, а также побежать в случае нахождения в нужной точке карты. Также у героя есть заранее заданное число ходов. Если ходы кончились, герой умирает, а уровень перезапускается.

На каждом уровне можно сохранить свой рекорд в базу данных. Это реализуется в классе DataBase.java при подключении локального сервера PosgreSQL. Для отчета времени создан отдельный класс Timers, создающий потоки-счетчики, активирующиеся раз в секунду.

Игровое поле

Заранее создадим файлы, в которых матрицей построим игровой поле. Затем в классе LvlChanger, получив Pane, где оно должно располагаться, и номер текущего уровня, можно определить какой именно файл необходимо открыть и куда нужно добавить поле.

Используем GridPane для размещения блоков, пройдясь по всему файлу (матрице) и заполняя соответствующие ячейки GridPane. каждому символу матрицы сопоставим определенный тайл. Например, 1 – горизонтальная стена, 0 – пол, и т.д.

После заполнения GridPane устанавливаем ее по середине родительского блока и добавляем на экран внутрь Pane, посланного в функцию.

Листинг функции createGameField:

public GridPane createGameField(File fileMap) {  
 try {  
 //find matrix height  
 BufferedReader inLen = new BufferedReader((new FileReader(fileMap.getAbsoluteFile())));  
 int i = 0;  
 while (inLen.readLine() != null) i++;  
 matrix = new char[i][];  
 inLen.close();  
 //create matrix  
 BufferedReader in = new BufferedReader((new FileReader(fileMap.getAbsoluteFile())));  
 String str;  
 int j = 0;  
 while ((str = in.readLine()) != null)  
 matrix[j++] = str.toCharArray();  
 in.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 //create blocks and add into grid  
 int mHeight = matrix.length;  
 int mWidth = matrix[0].length;  
  
 GridPane grid = new GridPane();  
 for (int i = 0; i < mHeight; i++) {  
 for (int j = 0; j < mWidth; j++) {  
 switch (matrix[i][j]) {  
 case '0':  
 grid.add(new ImageView(imgFloor), i, j);  
 break;

}  
 }  
 }  
 //delete padding between cells  
 grid.setVgap(0);  
 grid.setHgap(0);  
 //grid to center  
 gWidth = grid.getColumnCount() \* blockSize;  
 gHeight = grid.getRowCount() \* blockSize;  
 grid.setTranslateX((WIDTH - gWidth) / 2d);  
 grid.setTranslateY((HEIGHT - 50 - gHeight) / 2d);  
 return grid;  
}

Для корректного вызова используется следующий код в функции loadLvl (Pane field, in num):

File fileMap = new File("levelConfig/maps/" + numLvl + ".conf");  
field.getChildren().add(createGameField(fileMap));

Результат работы функции:

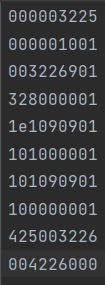


Рисунок 1. Создание игрового поля

Игровые элементы

Для игровых элементов вроде ящиков и врагов необходимо знать их расположение на карте и их тип. Они должны уметь двигаться при взаимодействии с игроком и друг другом. Например, враг при попадании на шипы должен умирать, а ящик блокировать урон герою, поскольку прикрывает шипы.

Для этого создадим отдельный класс Block.java:

public class Block {  
 public static int *blockSize* = 80;  
 String type;  
 int x, y; // coords on field  
 double errPos = 0.15 \* *blockSize*;  
 ImageView node;  
 Image imgDead = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0005.png")), 0.7 \* *blockSize*, 0.7 \* *blockSize*, false, true);  
 boolean isAlive = true;  
  
 public Block(int posX, int posY, ImageView img, String typeName) {  
 x = posX;  
 y = posY;  
 type = typeName;  
 node = img;  
 if (type.equals("Spike")) errPos = 0.05 \* *blockSize*;  
 node.setTranslateX(posX \* *blockSize* + errPos);  
 node.setTranslateY(posY \* *blockSize* + errPos);  
 }  
}

Для добавления блока на экран используем метод LvlChangerа:

public Pane createElements(File fileBlock) {  
 Pane elemBox = new Pane(), boxBox = new Pane(), enemyBox = new Pane(), spikeBox = new Pane();  
 try {  
 BufferedReader in = new BufferedReader((new FileReader(fileBlock.getAbsoluteFile())));  
 String str;  
 ImageView img;  
 while ((str = in.readLine()) != null) {  
 String[] strSplit = str.split(" ");  
 switch (strSplit[0]) {  
 case "box":  
 img = new ImageView(imgBox);  
 boxBox.getChildren().add(img);  
 listBoxAndEnemy.add(new Block(*parseInt*(strSplit[1]), *parseInt*(strSplit[2]), img, "Box"));  
 break;  
 }  
 }  
 in.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 if (key != null) elemBox.getChildren().add(key);  
 if (chest != null) elemBox.getChildren().add(chest);  
 elemBox.getChildren().addAll(spikeBox, enemyBox, boxBox, hero.getImg());  
 elemBox.setTranslateX((WIDTH - gWidth) / 2d);  
 elemBox.setTranslateY((HEIGHT - 50 - gHeight) / 2d);  
 return elemBox;  
}

Метод вызывается следующим кодом:

File fileBlock = new File("levelConfig/blocks/" + numLvl + ".conf");  
field.getChildren().add(createElements(fileBlock));

Результат работы методов:

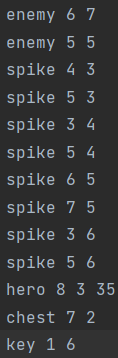


Рисунок 2. Создание игровых элементов

Главный герой

Главный герой, реализуемый классов Hero.java, похож на Block. Но кроме прочего у него свои взаимодействия с окружением и есть счетчик ходов.

public class Hero {  
  
 private final int blockSize = Block.*blockSize*;  
 private int x, y;  
 private int keyX, keyY;  
 private int chestX, chestY;  
 private final int animTime = 100;  
 private final Image redImg = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0164.png")), 0.7\*blockSize, 0.7\*blockSize, false, true);  
 private final ImageView node;  
 private int turns = 1000;  
 private boolean isKey = false;  
 private Sounds player = new Sounds();  
  
 public Hero(int posX, int posY, int countTurns, ImageView img, LvlChanger lvlChanger) {  
 x = posX;  
 y = posY;  
 if (lvlChanger.isSingleGame()) turns = countTurns;  
 node = img;  
 node.setTranslateX(x \* blockSize + 0.15 \* blockSize);  
 node.setTranslateY(y \* blockSize + 0.15 \* blockSize);  
 lvlChanger.getGameController().setTurnsLeft(turns);  
 }

При нажатии на клавиши WASD герой передвигается по полю:

public void initKey(Scene scene, GameController controller) {  
 scene.addEventFilter(KeyEvent.*KEY\_PRESSED*, evt -> {  
 Hero hero = getHero();  
 KeyCode code = evt.getCode();  
 if (!isOpenWindow)  
 switch (code) {  
 case *W*:  
 hero.move(hero.getX(), hero.getY() - 1, this);  
 break;

При передвижении проверяется возможность хода. Есть ли объект, преграждающий путь? Дошел ли герой до конца уровня? Есть ли у героя ходы?

public void move(int posX, int posY, LvlChanger lvlChanger) {  
 ArrayList<Block> listBoxAndEnemy = lvlChanger.getListBoxAndEnemy();  
 ArrayList<Block> listSpike = lvlChanger.getListSpike();  
 char[][] matrix = lvlChanger.getMatrix();  
 if (turns > 0) {  
 boolean isFree = true;  
 //check boxes and enemies  
 for (Block block : listBoxAndEnemy) {  
 isFree = checkBlock(posX, posY, block, lvlChanger);  
 if (!isFree) break;  
 }  
 //check spikes  
 for (Block spike : listSpike) {  
 if (posX == spike.getX() && posY == spike.getY() && isFree) {  
 spikeAnimation(posX, posY);  
 turns--;  
 player.playSpikeSound();  
 }  
 }  
 //check chest if hero has key  
 if (posX == chestX && posY == chestY && isKey) {  
 Pane pane = (Pane) lvlChanger.getGameController().getGameField().getChildren().get(1);  
 pane.getChildren().remove(lvlChanger.getChest());  
 player.playChestSound();  
 }  
 //check chest if hero hasn't key  
 if (posX == chestX && posY == chestY && !isKey) {  
 isFree = false;  
 }  
 //check key  
 if (posX == keyX && posY == keyY && isFree) {  
 isKey = true;  
 Pane pane = (Pane) lvlChanger.getGameController().getGameField().getChildren().get(1);  
 pane.getChildren().remove(lvlChanger.getKey());  
 player.playChestSound();  
 }  
 //if free then hero moves  
 if (isFree && matrix[posX][posY] == '0') {  
 moveAnimation(posX, posY, blockSize, animTime, false, node);  
 x = posX;  
 y = posY;  
 player.playMoveSound();  
 }  
 else if (matrix[posX][posY] == 'e' || matrix[posX][posY] == 'E') winAnimation(posX, posY, lvlChanger);  
 else {  
 player.playKickSound();  
 kickAnimation(posX,posY);  
 }  
 turns--;  
 lvlChanger.getGameController().setTurnsLeft(turns);  
 }  
 else {  
 lvlChanger.resetLvl(false);  
 player.playGameOverSound();  
 }  
}

Сама проверка блоков происходит в отдельном методе. Если блок перед героем обнаружен, нужно понять, можно ли его переместить или убить. В любом случае, если блок стоит на пути, герой не двигается. Происходит взаимодействие.

public boolean checkBlock(int posX, int posY, Block block, LvlChanger lvlChanger) {  
 if ((posX == block.getX()) && (posY == block.getY()) && (block.getIsAlive())) {  
 if (posX < x)  
 block.move(posX - 1, posY, lvlChanger);  
 if (posX > x)  
 block.move(posX + 1, posY, lvlChanger);  
 if (posY < y)  
 block.move(posX, posY - 1, lvlChanger);  
 if (posY > y)  
 block.move(posX, posY + 1, lvlChanger);  
 return false;  
 }  
 return true;  
}

Может ли блок умереть или сдвинуться проверяется уже внутри блока.

public void move(int posX, int posY, LvlChanger lvlChanger) {  
 ArrayList<Block> listBoxAndEnemy = lvlChanger.getListBoxAndEnemy();  
 ArrayList<Block> listSpike = lvlChanger.getListSpike();  
 char[][] matrix = lvlChanger.getMatrix();  
 boolean isFree = true;  
  
 for (Block block: listBoxAndEnemy)  
 if (posX == block.getX() && posY == block.getY() && isAlive) isFree = false;  
  
 for (Block spike: listSpike)  
 if (posX == spike.getX() && posY == spike.getY() && type.equals("Enemy")) kill();  
  
 if ((matrix[posX][posY] == '0') && (isFree)) {  
 moveAnimation(posX, posY, lvlChanger);  
 x = posX;  
 y = posY;  
 }  
 else if (matrix[posX][posY] != 0 && type.equals("Enemy")) kill();  
}

Здесь также проверяется наличие других блоков на пути, а также возможность умереть у врагов.

Анимация

Герой и блоки имеют анимацию для каждого движения. Анимации есть для перемещения на соседнюю клетку, удара (ящика, врага, стены – не важно), получения урона от шипов и спуска по победной лестнице.

Основной является анимация сдвига на определенной расстояние в одном из четырех направлений.

private void moveAnimation(int posX, int posY, double distance, int time, boolean isKick, ImageView nd) {  
 TranslateTransition anim = new TranslateTransition(Duration.*millis*(time), nd);  
  
 if (isKick) anim.setDelay(Duration.*millis*(time));  
  
 if (posX < x) {  
 anim.setByX(-distance);  
 if (isKick) anim.setFromX(node.getTranslateX() + distance);  
 }  
 if (posX > x) {  
 anim.setByX(distance);  
 if (isKick) anim.setFromX(node.getTranslateX() - distance);  
 }  
 if (posY < y) {  
 anim.setByY(-distance);  
 if (isKick) anim.setFromY(node.getTranslateY() + distance);  
 }  
 if (posY > y) {  
 anim.setByY(distance);  
 if (isKick) anim.setFromY(node.getTranslateY() - distance);  
 }  
 anim.play();  
}

Эта анимация используется для передвижения, передавая полный размер блока, чтобы герой сдвинулся на одну клетку в нужном направлении.

Кроме того, эта анимация используется для удара. Тогда вызываются две последовательные анимации смещения в длину четверти блока. Так герой двинется немного вперед, а затем вернется на место.

private void kickAnimation(int posX, int posY) {  
 moveAnimation(posX, posY, 0.25\*blockSize, animTime, false, node);  
 moveAnimation(posX, posY, -0.25\*blockSize, animTime, true, node);  
}

Кроме того, анимация смещения используется при победе. Герой двигается по лестнице, кроме того, он уменьшается и медленно исчезает. По истечению анимации уровень завершается.

private void winAnimation(int posX, int posY, LvlChanger lvlChanger) {  
 moveAnimation(posX, posY, 1.5\*blockSize, 3\*animTime, false, node);  
 ScaleTransition anim = new ScaleTransition(Duration.*millis*(3\*animTime), node);  
 anim.setByY(-0.2);  
 anim.setByX(-0.2);  
 anim.play();  
 FadeTransition anim1 = new FadeTransition(Duration.*millis*(3\*animTime), node);  
 anim1.setToValue(0.0);  
 anim1.play();  
 anim1.setOnFinished(e -> lvlChanger.winLvl());  
}

Самой интересной является анимация получения урона. Поскольку сложный объект нельзя просто немного перекрасить, как это делается с примитивами типа прямоугольников, то под героя добавляется его вторая версия, покрашенная в красный. Ей задается такое же смещение, как и у оригинала, а сам оригинал становится временно прозрачным. Таким образом, создается ощущение, что герой покраснел. После анимации красная тень удаляется.

private void spikeAnimation(int posX, int posY) {  
 //add red hero  
 ImageView redNode = new ImageView(redImg);  
 redNode.setTranslateX(node.getTranslateX());  
 redNode.setTranslateY(node.getTranslateY());  
 Pane pane = (Pane) node.getParent();  
 pane.getChildren().add(redNode);  
 node.toFront();  
 //move red hero as default hero  
 moveAnimation(posX, posY, blockSize, animTime, false, redNode);  
 //show red hero  
 FadeTransition damage = new FadeTransition(Duration.*millis*(animTime),node);  
 damage.setDelay(Duration.*millis*(0.5\*animTime));  
 damage.setToValue(0.6);  
 damage.setAutoReverse(true);  
 damage.setCycleCount(2);  
 damage.play();  
 //delete red hero after animation  
 damage.setOnFinished(e -> pane.getChildren().remove(redNode));  
}

Еще одна анимация используется при смерти персонажа или рестарте уровня. Она запускается в lvlChanger, временно создает белый прозрачный прямоугольник, меняет непрозрачность объекта и в этот момент уровень перезапускается.

public void resetLvl(boolean isRestart) {  
 Rectangle rct = new Rectangle(0, 0, WIDTH, HEIGHT);  
 rct.setFill(Color.*WHITE*);  
 gameController.getAnchorPane().getChildren().add(rct);  
  
 FadeTransition anim = new FadeTransition(Duration.*millis*(1500),rct);  
 anim.setFromValue(0.0);  
 anim.setToValue(1.0);  
 anim.play();  
  
 anim.setOnFinished(e -> {  
 timer.stop();  
 if (isRestart)  
 restartLvl(gameController.getGameField());  
 else loadLvl(gameController.getGameField(), numLvl);  
 FadeTransition anim1 = new FadeTransition(Duration.*millis*(1500),rct);  
 anim1.setFromValue(1.0);  
 anim1.setToValue(0.0);  
 anim1.play();  
 anim1.setOnFinished(ev -> gameController.getAnchorPane().getChildren().remove(rct));  
 });  
}

Интерфейс

Игроку необходимо знать, сколько ходов у него осталось, а также сколько времени прошло с начала уровня. Для их отображения в нижней части экрана добавим HBox, отображающий все необходимые показатели. Кроме того, в нем будет показываться краткая справка по клавишам и, если игрок нажмет клавишу «Н», то помощь в решении уровня.

Листинг верстки интерфейса:

<Pane fx:id="menuWindow" layoutX="285.0" layoutY="185.0" prefHeight="350.0" prefWidth="350.0" styleClass="pane\_dark" stylesheets="@style.css" visible="false">  
 <Label alignment="CENTER" layoutX="5.0" layoutY="32.0" prefHeight="38.0" prefWidth="341.0" styleClass="text\_large" stylesheets="@style.css" text="ПАУЗА" />  
 <Button fx:id="resetMenuBtn" layoutX="90.0" layoutY="150.0" minHeight="40.0" minWidth="170.0" mnemonicParsing="false" styleClass="button\_dark" stylesheets="@style.css" text="Начать заново" />  
 <Button fx:id="exitMenuBtn" layoutX="90.0" layoutY="270.0" minHeight="40.0" minWidth="170.0" mnemonicParsing="false" styleClass="button\_dark" stylesheets="@style.css" text="Выход из игры" />  
 <Button fx:id="continueMenuBtn" layoutX="90.0" layoutY="90.0" minHeight="40.0" minWidth="170.0" mnemonicParsing="false" styleClass="button\_dark" stylesheets="@style.css" text="Продолжить" />  
 <Button fx:id="mainMenuBtn" layoutX="90.0" layoutY="210.0" minHeight="40.0" minWidth="170.0" mnemonicParsing="false" styleClass="button\_dark" stylesheets="@style.css" text="Главное меню" />  
</Pane>

Выглядеть он будет следующим образом:



Рисунок 3. Внешний вид интерфейса

Смена ходов осуществляется внутри Hero.java в момент хода героя.

lvlChanger.getGameController().setTurnsLeft(turns);

Для этого используется метод в контроллере:

public void setTurnsLeft(int count) { statusTurns.setText("Ходов осталось: " + count);}

Для отображения подсказки игрок должен нажать «Н». В этот момент текст справки меняется на заранее записанную последовательность символов.



Рисунок 4. Интерфейс подсказок

Последовательность ходов заранее определим в файле help.txt.

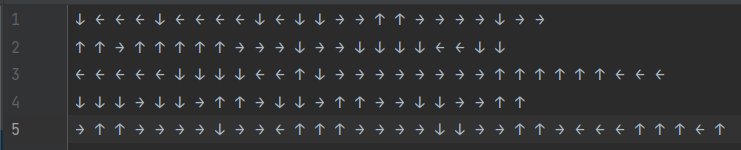


Рисунок 5. Содержание файла help.txt

Для смены текста внутри лейбла используется метод контроллера, в котором файл построчно считывается и на экран выводится определенная строка.

public void setStatusHelp(boolean isOpenHelp, int num) {  
 if (isOpenHelp) statusHelp.setText("R - заново, H - помощь, WASD - управление");  
 else  
 try {  
 File file = new File("levelConfig/help.txt");  
 BufferedReader in = new BufferedReader((new FileReader(file.getAbsoluteFile())));  
 int i = 1;  
 String str;  
 while ((str = in.readLine()) != null) {  
 if (i == num) {  
 statusHelp.setText(str);  
 break;  
 }  
 i++;  
 }  
 in.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
}

Для вызова метода добавляем событие при нажатии на клавишу H.

case *H*:  
 if (isSingleGame) {  
 controller.setStatusHelp(isOpenHelp, numLvl);  
 isOpenHelp = !isOpenHelp;  
 }  
 break;

Наконец, перейдем к таймеру. Для его создания необходимо инициализировать новый поток. Вынесем это в отдельный класс Timers.java. В конструкторе класса создается поток, засыпающий каждую секунду. Внутри потока инкрементируется переменная-счетчик секунд.

Кроме этого, поток не должен считать секунды во время открытой меню паузы. Поэтому счетчик оборачивается дополнительным условием, зависящим от булевой переменной объекта класса LvlChanger.

public class Timers {  
 int sec = 0;  
 int min = 0;  
 boolean isActive = true;  
  
 public Timers(Label statusTimer, LvlChanger lvlChanger) {  
 Thread timer = new Thread(() -> {  
 while(isActive) {  
 try {  
 Platform.*runLater*(() -> statusTimer.setText(" Время: " + setTime()));  
 Thread.*sleep*(1000);  
 if (lvlChanger.isSingleGame()) {  
 if (!lvlChanger.isOpenWindow())  
 sec++;  
 } else sec++;  
 if (sec == 60) {  
 sec = 0;  
 min++;  
 }  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 });  
 timer.start();  
 }

Для остановки таймера достаточно переключить isActive. Тогда поток завершится. Каждый уровень создается новый таймер. Проигрыш или рестарт также останавливает поток и создает новый.

public void stop() { isActive = false; }

Для форматированного вывода времени также используем отдельный метод:

private String setTime() {  
 String str;  
 if (min < 10) str = "0" + min + ":";  
 else str = min + ":";  
 if (sec < 10) str += "0" + sec;  
 else str += sec + "";  
 return str;  
}

При нажатии на Esc открывается (или закрывается) меню паузы.

if (code == *ESCAPE* && !controller.getWinWindow().isVisible()) {  
 if (isOpenWindow) { // open window  
 if (isSingleGame) controller.getMenuWindow().setVisible(false); // single game  
 else controller.getGiveupWindow().setVisible(false); // online game  
 } else { // close window  
 if (isSingleGame) controller.getMenuWindow().setVisible(true);  
 else controller.getGiveupWindow().setVisible(true);  
 }  
 reverseOpenWindow(); // reverse boolean marker  
}

Оно представляет из себя невидимый Pane c несколькими кнопками. В меню паузы можно:

* продолжить игру;
* начать уровень заново;
* выйти в главное меню;
* выйти из игры.

Каждая кнопка инициализируется при первой загрузке сцены с игровым полем.

continueMenuBtn.setOnAction(e -> {  
 menuWindow.setVisible(false);  
 lvlChanger.reverseOpenWindow();  
});  
resetMenuBtn.setOnAction(e -> {  
 lvlChanger.loadLvl(gameField,lvlChanger.getNumLvl());  
 lvlChanger.reverseOpenWindow();  
});  
mainMenuBtn.setOnAction(e -> {  
 lvlChanger.startMenu();  
 lvlChanger.reverseOpenWindow();  
});  
exitMenuBtn.setOnAction(e -> System.*exit*(0));



Рисунок 6. Внешний вид меню паузы

Еще один невидимый до времени Pane отвечает за окно победы. Когда герой доходит до точки интереса, оно становится видимым.

Листинг верстки победного окна:

<Pane fx:id="winWindow" layoutX="275.0" layoutY="275.0" prefHeight="200.0" prefWidth="350.0" styleClass="pane\_dark" stylesheets="@style.css" visible="false">  
 <Label alignment="CENTER" layoutX="-1.0" layoutY="27.0" prefHeight="38.0" prefWidth="353.0" stylesheets="@style.css" text="Поздравляем!" />  
 <Label alignment="CENTER" layoutY="71.0" prefHeight="58.0" prefWidth="350.0" styleClass="text\_large" stylesheets="@style.css" text="ЗАГАДКА РЕШЕНА" />  
 <Button fx:id="resetWinBtn" layoutX="22.0" layoutY="145.0" minHeight="40.0" minWidth="90.0" mnemonicParsing="false" styleClass="button\_dark" stylesheets="@style.css" text="Еще раз" />  
 <Button fx:id="nextWinBtn" layoutX="238.0" layoutY="145.0" minHeight="40.0" minWidth="90.0" mnemonicParsing="false" styleClass="button\_dark" stylesheets="@style.css" text="Дальше" />  
 <Button fx:id="saveWinBtn" layoutX="127.0" layoutY="145.0" maxWidth="102.0" minHeight="40.0" minWidth="90.0" mnemonicParsing="false" prefHeight="40.0" prefWidth="102.0" styleClass="button\_dark" stylesheets="@style.css" text="Рекорды" textAlignment="CENTER" wrapText="true" />  
</Pane>

Каждая кнопка также инициализируется при первой загрузке игрового поля, внутри контроллера.

nextWinBtn.setOnAction(e -> {  
 lvlChanger.loadLvl(gameField,lvlChanger.getNumLvl()+1);  
 lvlChanger.reverseOpenWindow();  
});  
saveWinBtn.setOnAction(e -> {  
 saveRecordBtn.setDisable(false);  
 lvlChanger.clearRecordTable(recordTable);  
 lvlChanger.getDB().writeToTable(recordTable, lvlChanger.getNumLvl());  
 recordWindow.toFront();  
 recordWindow.setVisible(true);  
});  
resetWinBtn.setOnAction(e -> {  
 lvlChanger.loadLvl(gameField,lvlChanger.getNumLvl());  
 lvlChanger.reverseOpenWindow();  
});



Рисунок 7. Внешний вид победного окна

Главное меню

Главное меню полностью создадим в FXML-файле menu.fxml. Оно содержит несколько невидимых Pane, появляющихся по нажатию пользователем определенных кнопок. Через меню можно:

* запустить новую игру, которая начнется с обучающего уровня;
* выбрать конкретный уровень (кроме обучающего);
* запустить соревновательный режим (кнопка недоступна, если пользователь не подключен к серверу);
* посмотреть таблицу рекордов из базы данных (кнопка недоступна, если пользователь не подключен к базе данных);
* выйти из игры.

Полный код представлен в приложении.

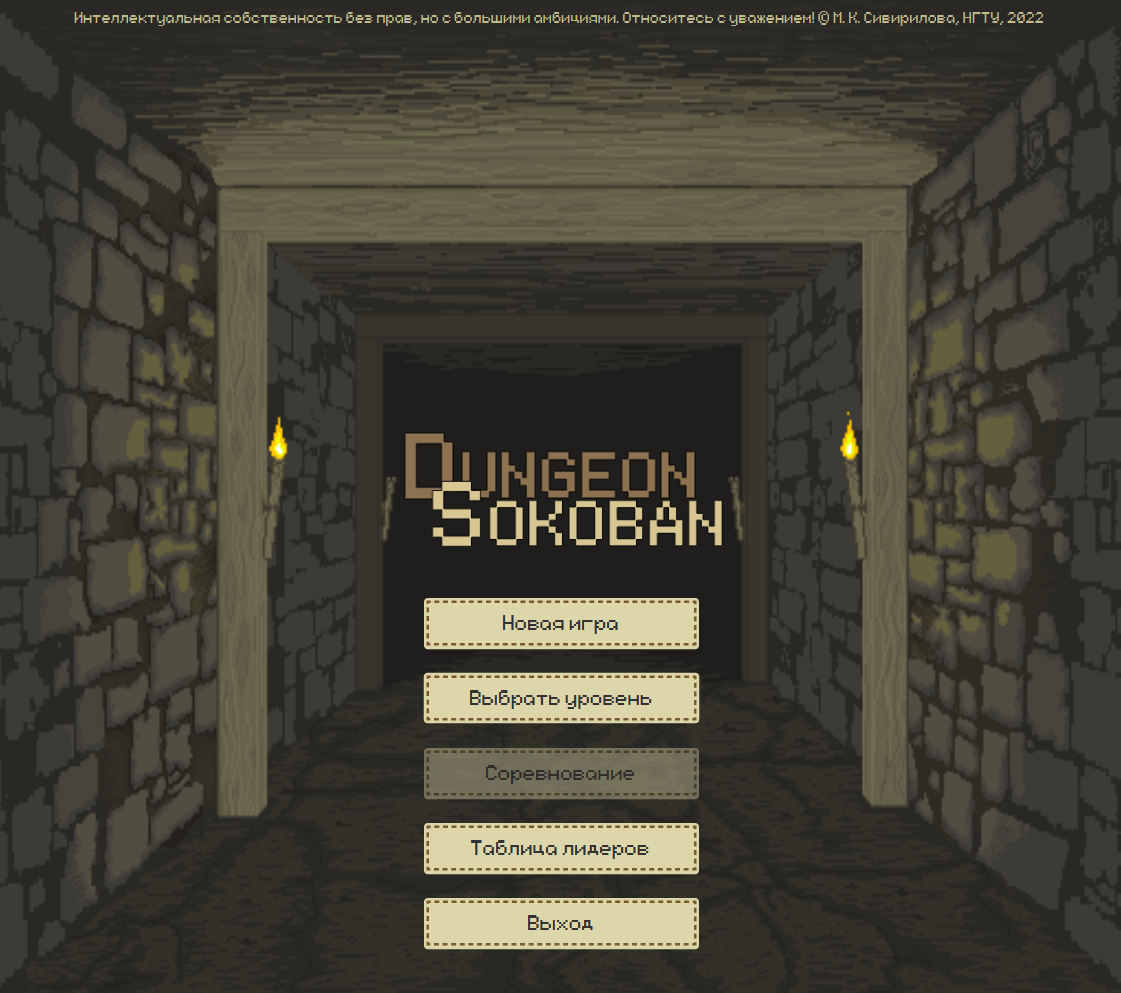


Рисунок 8. Внешний вид главного меню



Рисунок 9. Открытая панель выбора уровня

Меню открывается при запуске игры, а также при выходе с уровня. Для этого используется метод класса LvlChanger, в котором происходит смена сцены.

public void startMenu() {  
 try {  
 player.playMainMusic();  
 FXMLLoader fxml = new FXMLLoader(Client1.class.getResource("menu.fxml"));  
 Scene scene = new Scene(fxml.load(), WIDTH, HEIGHT);  
 menuController = fxml.getController();  
 menuController.initialize(this, launcher);  
 if (!launcher.isSocketActive()) menuController.setDisableOnlineBtn();  
 if (!DB.isActive()) menuController.setDisableRecordBtn();  
 stage.setScene(scene);  
 }  
 catch (IOException e) {  
 e.printStackTrace();  
 }  
}

Каждая кнопка инициализируется при загрузке главного меню. Это происходит внутри контроллера.

@FXML  
public void initialize(LvlChanger lvlChanger, Launcher launcher) {  
 startBtn.setOnAction(e -> lvlChanger.startGame(0));  
 recordBtn.setOnAction(e -> {  
 lvlChanger.clearRecordTable(recordTable);  
 recordPane.setVisible(true);  
 btnPane.setVisible(false);  
 });  
 recordBtn1.setOnAction(e -> lvlChanger.getDB().writeToTable(recordTable, 1));  
 recordBtn2.setOnAction(e -> lvlChanger.getDB().writeToTable(recordTable, 2));  
 recordBtn3.setOnAction(e -> lvlChanger.getDB().writeToTable(recordTable, 3));  
 recordBtn4.setOnAction(e -> lvlChanger.getDB().writeToTable(recordTable, 4));  
 recordBtn5.setOnAction(e -> lvlChanger.getDB().writeToTable(recordTable, 5));  
 undoRecordBtn.setOnAction(e -> {  
 recordPane.setVisible(false);  
 btnPane.setVisible(true);  
 });  
 lvlBtn.setOnAction(e -> {  
 btnPane.setVisible(false);  
 lvlMenu.setVisible(true);  
 });  
 lvlBtn1.setOnAction(e -> lvlChanger.startGame(1));  
 lvlBtn2.setOnAction(e -> lvlChanger.startGame(2));  
 lvlBtn3.setOnAction(e -> lvlChanger.startGame(3));  
 lvlBtn4.setOnAction(e -> lvlChanger.startGame(4));  
 lvlBtn5.setOnAction(e -> lvlChanger.startGame(5));  
 undoBtn.setOnAction(e -> {  
 btnPane.setVisible(true);  
 lvlMenu.setVisible(false);  
 });  
 onlineBtn.setOnAction(e -> {  
 try {  
 if (!isOnline) {  
 readyOnlineBtn.setDisable(true);  
 onlineLabel.setText("Ожидание игрока...");  
 }  
 onlinePane.setVisible(true);  
 btnPane.setVisible(false);  
 OutputStream os = launcher.getSocket().getOutputStream();  
 ByteBuffer buffer = ByteBuffer.*allocate*(Integer.*BYTES*);  
 buffer.putInt(1);  
 os.write(buffer.array());  
 os.flush();  
 }  
 catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 });  
 readyOnlineBtn.setOnAction(e -> {  
 try {  
 readyOnlineBtn.setDisable(true);  
 OutputStream os = launcher.getSocket().getOutputStream();  
 os.write(new Message(launcher.getIdOpponent(),2).getBytes());  
 os.flush();  
 }  
 catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 });  
 undoOnlineBtn.setOnAction(e -> {  
 onlinePane.setVisible(false);  
 btnPane.setVisible(true);  
 });  
 exitBtn.setOnAction(e -> System.*exit*(0));  
}

Игровой процесс

При нажатии на новую игру или выборе определенного уровня, происходит смена сцены в методе класса LvlChander.java.

public void startGame(int num) {  
 try {  
 player.playGameMusic();  
 FXMLLoader fxml = new FXMLLoader(Client1.class.getResource("game.fxml"));  
 Scene scene = new Scene(fxml.load(), WIDTH, HEIGHT);  
 LvlChanger temp = this;  
 Platform.*runLater*(() -> {  
 stage.setScene(scene);  
 gameController = fxml.getController();  
 gameController.initialize(temp, launcher);  
 loadLvl(gameController.getGameField(), num);  
 initKey(scene, gameController);  
 });  
  
 }  
 catch (IOException e) {  
 e.printStackTrace();  
 }  
}

После инициализации управления и нового контроллера, начинается загрузка выбранного уровня.

public void loadLvl(Pane field, int num) {  
 clearLvl(field);  
 numLvl = num;  
 if (numLvl == 6) startMenu();  
 else {  
 //create map  
 File fileMap = new File("levelConfig/maps/" + numLvl + ".conf");  
 field.getChildren().add(createGameField(fileMap));  
 //create blocks and hero  
 File fileBlock = new File("levelConfig/blocks/" + numLvl + ".conf");  
 field.getChildren().add(createElements(fileBlock));  
 //create local level timer  
 timer = new Timers(gameController.getStatusTimer(), this);  
 }  
}

При этом если пройден последний уровень, то игрок попадает в главное меню – он прошел игру.

Перед каждой загрузкой необходимо очищать поле от предыдущих элементов и карты. Для этого обнуляются отображения картинок на экране, а также очищаются все списки. Переключатели окон переходят в исходное состояние.

public void clearLvl(Pane field) {  
 field.getChildren().clear();  
 key = null;  
 chest = null;  
 listBoxAndEnemy.clear();  
 listSpike.clear();  
 gameController.getStatusHelp().setTextFill(Color.*WHITE*);  
 gameController.getWinWindow().setVisible(false);  
 gameController.getMenuWindow().setVisible(false);  
}

При рестарте происходит примерно то же, только не переназначается новый уровень, поскольку перезапускается старый.

public void restartLvl(Pane field) {  
 clearLvl(field);  
 File fileMap = new File("levelConfig/maps/" + numLvl + ".conf");  
 field.getChildren().add(createGameField(fileMap));  
 File fileBlock = new File("levelConfig/blocks/" + numLvl + ".conf");  
 field.getChildren().add(createElements(fileBlock));  
 if (isSingleGame) timer = new Timers(gameController.getStatusTimer(), this);  
}

При победе игрока происходит остановка таймера, закрываются все окна, воспроизводится анимация спуска героя, а также включается отображение победного окна.

public void winLvl() {  
 timer.stop();  
 reverseOpenWindow();  
 closeHelp();  
 gameController.getSaveWinBtn().setDisable(numLvl == 0 || !DB.isActive()); // if lvl0 or DB isn't active  
 if (isSingleGame) { // single  
 gameController.getWinWindow().toFront();  
 gameController.getWinWindow().setVisible(true);  
 } else { // online  
 try {  
 gameController.getWinOnlineWindow().toFront();  
 gameController.getWinOnlineWindow().setVisible(true);  
 // send to server only if player wasn't win or lose before  
 if (getGameController().getStatusHelp().getTextFill() == Color.*WHITE*) {  
 OutputStream os = launcher.getSocket().getOutputStream();  
 os.write(new Message(launcher.getIdOpponent(), 3).getBytes());  
 os.flush();  
 }  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 }  
}

Таблица рекордов

Для таблицы рекордов используем локальный сервер, созданный при помощи PostgreSQL. В отдельном классе DataBase.java будем подключаться к заранее созданной базе данных. В конструкторе класса осуществим проверку на подключение. В случае отказа переключаем переменную-маркер и деактивируем все кнопки, позволяющие пользователю использовать базу данных.

public class DataBase {  
  
 private static final String *DRIVER* = "org.postgresql.Driver";  
 private static final String *DB\_URL* = "jdbc:postgresql://127.0.0.1:5432/Sokoban";  
 private static final String *USER* = "postgres";  
 private static final String *PASS* = "1234";  
  
 private boolean isActive;  
  
 public DataBase() {  
 try {  
 Class.*forName*(*DRIVER*);  
 Connection connection = DriverManager.*getConnection*(*DB\_URL*, *USER*, *PASS*);  
 Statement statement = connection.createStatement();  
 statement.close();  
 connection.close();  
 isActive = true;  
 } catch (Exception e) {  
 System.*out*.println("База данных не подключена!");  
 isActive = false;  
 }  
 }

Для каждого уровня создана своя таблица рекордов, состоящая из двух колонок: ник игрока и время, за которое он прошел уровень. Для добавления новых данных используем метод add, взаимодействующий с нужной таблицей.

public void add(String nickname, int time, int numLvl) {  
 try {  
 Class.*forName*(*DRIVER*);  
 Connection c = DriverManager.*getConnection*(*DB\_URL*, *USER*, *PASS*);  
 Statement statement = c.createStatement();  
  
 c.setAutoCommit(false);  
 String sql = "INSERT INTO LEVEL" + numLvl + " (NICKNAME, TIME)"  
 + "VALUES ('" + nickname + "', " + time + ");";  
 statement.executeUpdate(sql);  
  
 statement.close();  
 c.commit();  
 c.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
}

Чтобы вывести таблицу на экран, создадим структуру VBox, состоящую из нескольких HBox – каждый отвечает за свою строку в таблице базы данных. При помощи CSS-стилей впишем ее в окружающий интерфейс.



Рисунок 10. Внешний вид таблицы рекордов в главном меню



Рисунок 11. Внешний вид таблицы рекордов в игре

Для сохранения своего рекорда в таблицу необходимо ввести ник и нажать кнопку сохранить. Оставить пустое поле нельзя – программа выведет в него сообщение с просьбой ввести имя.

При нажатии на кнопку сохранить происходит запись в базу данных. Далее данные сортируются уже с новыми вводными и выводятся на экран в обновленном виде. Кнопка сохранения при этом деактивируется, чтобы нельзя было сохранять один результат дважды. При возврате на предыдущий экран кнопка рекордов также не будет доступна. Они активируются вновь только при повторном прохождении уровня.

saveWinBtn.setOnAction(e -> {  
 saveRecordBtn.setDisable(false);  
 lvlChanger.clearRecordTable(recordTable);  
 lvlChanger.getDB().writeToTable(recordTable, lvlChanger.getNumLvl());  
 recordWindow.toFront();  
 recordWindow.setVisible(true);  
});  
undoRecordBtn.setOnAction(e -> {  
 if (saveRecordBtn.isDisable()) {  
 saveWinBtn.setDisable(true);  
 saveOnlineWinBtn.setDisable(true);  
 }  
 recordWindow.setVisible(false);  
});  
saveRecordBtn.setOnAction(e -> {  
 if (!nickTextField.getText().equals("")) {  
 lvlChanger.saveRecordDB(nickTextField.getText(), recordTable);  
 saveRecordBtn.setDisable(true);  
 }  
 else nickTextField.setText("введите имя");  
});  
nickTextField.setOnMouseClicked(e -> {  
 if (nickTextField.getText().equals("введите имя (без пробелов)!")) nickTextField.setText("");  
});

Заполнение таблицы будет производиться в методе writeToTable класса DataBase.java. Сначала происходит подключение к базе данных, затем поиск нужной таблицы, после – занесение найденных данных в строки HBox, т.е. в VBoxы. Пустые строки заполняются значениями-заглушками.

public void writeToTable(VBox table, int numLvl) {  
 try {  
 Class.*forName*(*DRIVER*);  
 Connection c = DriverManager.*getConnection*(*DB\_URL*, *USER*, *PASS*);  
 Statement stmt = c.createStatement();  
 int i = 0;  
 ResultSet rs = stmt.executeQuery("SELECT NICKNAME, TIME FROM LEVEL" + numLvl + " ORDER BY TIME;");  
 while (rs.next()) {  
 HBox line = (HBox) table.getChildren().get(i + 1);  
 Label nickname = (Label) line.getChildren().get(1);  
 Label time = (Label) line.getChildren().get(2);  
 nickname.setText(rs.getString("nickname"));  
 time.setText(String.*valueOf*(rs.getInt("time")));  
 i++;  
 }  
 while (i < 10) { // fill empty lines  
 HBox line = (HBox) table.getChildren().get(i + 1);  
 Label nickname = (Label) line.getChildren().get(1);  
 Label time = (Label) line.getChildren().get(2);  
 nickname.setText("-");  
 time.setText("-");  
 i++;  
 }  
 stmt.close();  
 c.close();  
 } catch (Exception e) {  
 System.*err*.println(e.getClass().getName() + ": " + e.getMessage());  
 }  
}

Вызов методов нужно осуществлять последовательно: сначала добавляем в базу данных, затем вынимаем оттуда новые отсортированные данные.

public void saveRecordDB(String nick, VBox table) {  
 DB.add(nick, timer.getTime(), numLvl);  
 DB.writeToTable(table, numLvl);  
}

Кроме этого, также необходимо очищать заполнение таблицы рекордов при каждом ее отображении.

public void clearRecordTable(VBox table) {  
 for (int i = 0; i < 10; i++) {  
 HBox line = (HBox) table.getChildren().get(i + 1);  
 Label nickname = (Label) line.getChildren().get(1);  
 Label time = (Label) line.getChildren().get(2);  
 nickname.setText("-");  
 time.setText("-");  
 }  
}

Соревновательный режим

Сначала необходимо создать локальный сервер. Для этого используем класс Server.java.

public class Server {  
 ServerSocket server;  
 UUID player; // waiting player  
 Socket playerSocket; // waiting player's socket  
 int countPlayers = 0; // count of waiting player  
  
 {  
 try {  
 server = new ServerSocket(1984);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static void main(String[] args) {  
 try {  
 new Server();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }

В конструкторе класса создадим два потока.

Первый будет отлавливать новых игроков, генерировать и отсылать им уникальный UUID.

public Server() throws IOException {  
 System.*out*.println("Server started");  
 Map<UUID, Socket> sockets = new HashMap<>();  
  
 Thread socketThread = new Thread(() -> {  
 while (true) {  
 try {  
 Socket newSocket = server.accept();  
 UUID newSocketId = UUID.*randomUUID*();  
  
 synchronized (sockets) {  
 sockets.put(newSocketId, newSocket);  
 }  
  
 synchronized (sockets) {  
 sockets.forEach((uuid, socket) -> {  
 try {  
 //send yourself your uuid  
 OutputStream os = socket.getOutputStream();  
 if (uuid == newSocketId) os.write(new Message(newSocketId, 0).getBytes());  
 os.flush();  
 } catch (Exception e) {  
 e.printStackTrace();  
 synchronized (sockets) {  
 sockets.remove(uuid);  
 }  
 }  
 });  
 }  
  
 System.*out*.println("Accepted " + newSocketId);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 });  
  
 socketThread.start();

Второй будет отлавливать сообщения от клиентов и перенаправлять их другим клиентам.

while (true) {  
 synchronized (sockets) {  
 sockets.forEach((uuid, socket) -> {  
 try {  
 BufferedInputStream reader = new BufferedInputStream(socket.getInputStream());  
 List<Byte> byteList = new ArrayList<>();  
 while (reader.available() > 0) {  
 byteList.add((byte) reader.read());  
 }  
 if (byteList.size() != 0) {  
 byte[] bytes = new byte[byteList.size()];  
 for (int i = 0; i < byteList.size(); i++) {  
 bytes[i] = byteList.get(i);  
 }  
 ByteBuffer buffer = ByteBuffer.*wrap*(bytes);  
 int type = buffer.getInt();  
 byte[] newArray;  
 OutputStream os;  
 ByteBuffer buf;  
 switch (type) {  
 //case…  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 });  
 }  
 }  
}

Существует несколько ситуаций, когда клиент должен отправить сообщение на сервер.

Во-первых, когда нажимается кнопка «Соревнование» и игрок начинает поиск оппонента. Тогда сервер получает сообщение с текстом «1» и помещает игрока в лист ожидания.

onlineBtn.setOnAction(e -> {  
 try {  
 if (!isOnline) {  
 readyOnlineBtn.setDisable(true);  
 onlineLabel.setText("Ожидание игрока...");  
 }  
 onlinePane.setVisible(true);  
 btnPane.setVisible(false);  
 OutputStream os = launcher.getSocket().getOutputStream();  
 ByteBuffer buffer = ByteBuffer.*allocate*(Integer.*BYTES*);  
 buffer.putInt(1);  
 os.write(buffer.array());  
 os.flush();  
 }  
 catch (IOException ex) {  
 ex.printStackTrace();  
 }  
});

Как только следующий игрок решит нажать кнопку «Соревнование», он также отправит серверу сообщение с «1» и сервер подсоединит его к ожидающему игроку. Это произойдет следующим образом: сервер отошлет клиенту, от которого получил запрос, и клиенту, который ожидает ответа, сообщения с UUID соперника.

case 1:  
 System.*out*.println("Player " + uuid + " connected");  
 if (player == null) {  
 player = uuid;  
 playerSocket = socket;  
 } else {  
 os = socket.getOutputStream();  
 os.write(new Message(player, 1).getBytes());  
 os.flush();  
 os = playerSocket.getOutputStream();  
 os.write(new Message(uuid, 1).getBytes());  
 os.flush();  
 player = null;  
 playerSocket = null;  
 }  
 break;

На стороне клиента также создаем поток в классе Launcher.java, ожидающий сообщения от сервера. В нем будет происходить их обработка. Например, как только клиент получит сообщение с заголовком «1» и чужим UUID, то запишет его в себе в оппоненты, обновит лейбл поиска игрока и позволит пользователю нажать кнопку «Готов!»

public void startSocket() {  
 try {  
 socket = new Socket("localhost", 1984);  
 isSocketActive = true;  
 } catch (IOException e) {  
 System.*out*.println("Сервер не подключен!");  
 isSocketActive = false;  
 //e.printStackTrace();  
 }  
  
 Thread socketThread = new Thread(() -> {  
 while (true) {  
 try {  
 if (isSocketActive) {  
 BufferedInputStream reader = new BufferedInputStream(socket.getInputStream());  
 List<Byte> bytes = new ArrayList<>();  
 while (reader.available() > 0) {  
 bytes.add((byte) reader.read());  
 }  
  
 if (bytes.size() > 0) {  
 byte[] byteArray = new byte[bytes.size()];  
 for (int i = 0; i < bytes.size(); i++) {  
 byteArray[i] = bytes.get(i);  
 }  
 ByteBuffer buffer = ByteBuffer.*wrap*(byteArray);  
 int type = buffer.getInt();  
 byte[] newArray;  
 switch (type) {  
 case 0: //own uuid  
 newArray = new byte[buffer.array().length - buffer.position()];  
 buffer.get(newArray, buffer.arrayOffset(), newArray.length);  
 Message uuid = new Message(newArray);  
 id = uuid.getId();  
  
 System.*out*.println("own: " + id);  
 break;  
  
 case 1: //opponent uuid  
 newArray = new byte[buffer.array().length - buffer.position()];  
 buffer.get(newArray, buffer.arrayOffset(), newArray.length);  
 Message message = new Message(newArray);  
 idOpponent = message.getId();  
 lvlChanger.getMenuController().setOnlineOpponent();  
 System.*out*.println("opponent: " + message.getId());  
 break;  
  
 }  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 });  
  
 socketThread.start();  
}

Для клиента это будет выглядеть так:

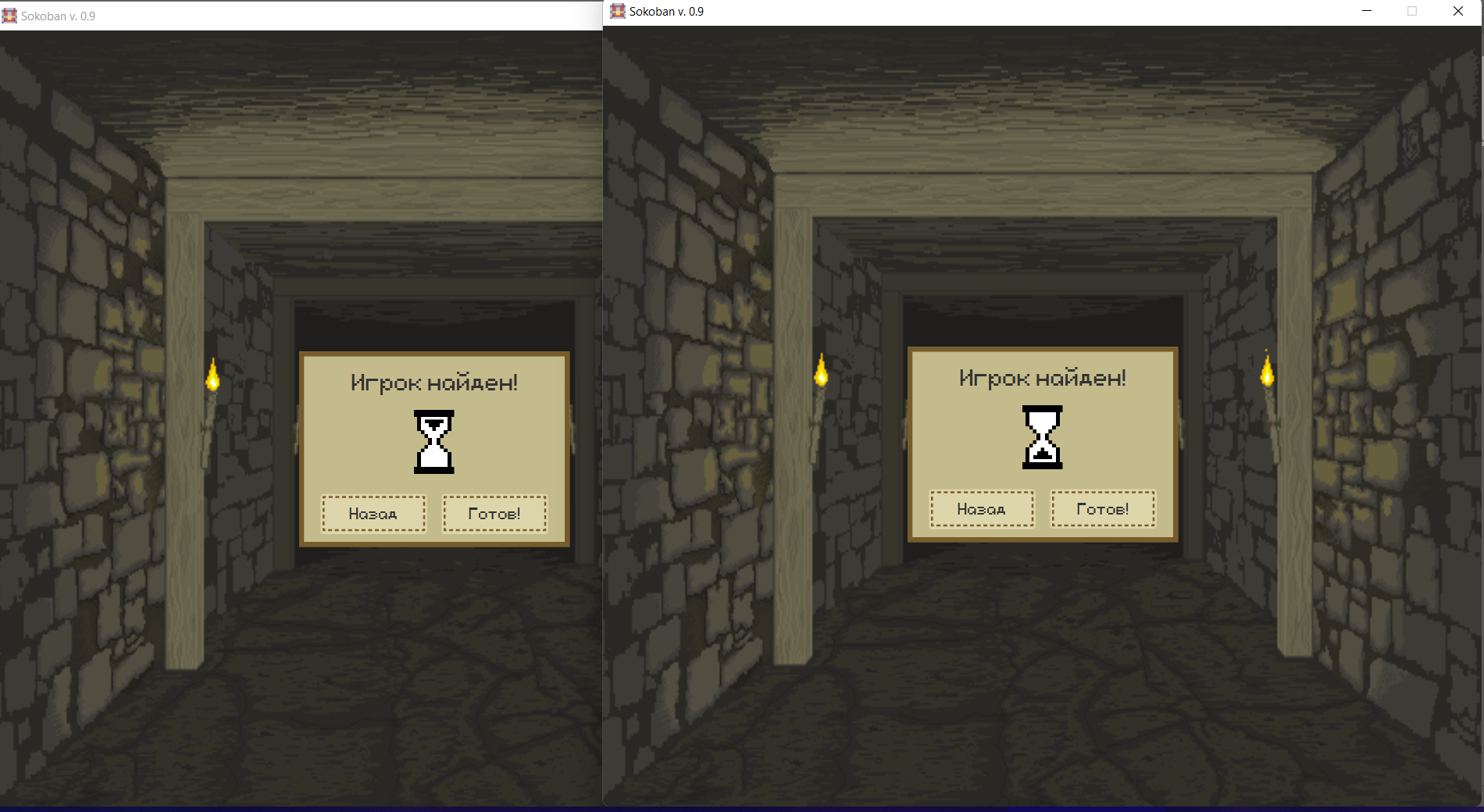


Рисунок 12. Поиск игрока в соревновательном режиме

При нажатии кнопки «Готов!» на сервер будет отправляться сообщение с заголовком «2» и UUID соперника. Это необходимо, поскольку сам сервер не запоминает, какой игрок с каким соревнуется.

readyOnlineBtn.setOnAction(e -> {  
 try {  
 readyOnlineBtn.setDisable(true);  
 OutputStream os = launcher.getSocket().getOutputStream();  
 os.write(new Message(launcher.getIdOpponent(),2).getBytes());  
 os.flush();  
 }  
 catch (IOException ex) {  
 ex.printStackTrace();  
 }  
});

Получив от клиента такое сообщение, сервер считает готовых игроков. Как только оба игрока будут готовы, счетчик готовых игроков обнуляется, на сервере рандомится уровень и отправляется обоим клиентам вместе с заголовком «2».

case 2:  
 System.*out*.println("Player " + uuid + " ready");  
 countPlayers++;  
 newArray = new byte[buffer.array().length - buffer.position()];  
 buffer.get(newArray, buffer.arrayOffset(), newArray.length);  
 Message message = new Message(newArray);  
 System.*out*.println("Opponent: " + message.getId());  
 if (countPlayers == 2) {  
 countPlayers = 0;  
 buf = ByteBuffer.*allocate*(Integer.*BYTES* \* 2);  
 buf.putInt(2);  
 buf.putInt((int) (Math.*random*() \* 3 + 1));  
 os = socket.getOutputStream();  
 os.write(buf.array());  
 os.flush();  
 os = sockets.get(message.getId()).getOutputStream();  
 os.write(buf.array());  
 os.flush();  
 }

Клиент, получив от сервера такое сообщение, запускает уровень с заданным номером.

case 2: //all players ready  
 lvlChanger.getMenuController().setOnlineReady();  
 lvlChanger.reverseSingleGame();  
 lvlChanger.startGame(buffer.getInt());  
 break;

Следующее взаимодействие клиента и сервера произойдет, когда один из игроков победит или сдаться. Для каждой ситуации клиент генерирует сообщение серверу с заголовком «3» и «4» соответственно, а также с UUID.

При победе клиент отправляет:

try {  
 gameController.getWinOnlineWindow().toFront();  
 gameController.getWinOnlineWindow().setVisible(true);  
 // send to server only if player wasn't win or lose before  
 if (getGameController().getStatusHelp().getTextFill() == Color.*WHITE*) {  
 OutputStream os = launcher.getSocket().getOutputStream();  
 os.write(new Message(launcher.getIdOpponent(), 3).getBytes());  
 os.flush();  
 }  
} catch (IOException ex) {  
 ex.printStackTrace();  
}

При победе сервер получает и перенаправляет:

case 3:  
 System.*out*.println("Player " + uuid + " win");  
 newArray = new byte[buffer.array().length - buffer.position()];  
 buffer.get(newArray, buffer.arrayOffset(), newArray.length);  
 Message loser = new Message(newArray);  
 os = sockets.get(loser.getId()).getOutputStream();  
 buf = ByteBuffer.*allocate*(Integer.*BYTES*);  
 buf.putInt(3);  
 os.write(buf.array());  
 os.flush();  
 break;

При признании поражения клиент отправляет:

yesOnlineBtn.setOnAction(e -> {  
 try {  
 if (statusHelp.getTextFill() == Color.*WHITE*) {  
 OutputStream os = launcher.getSocket().getOutputStream();  
 os.write(new Message(launcher.getIdOpponent(), 4).getBytes());  
 os.flush();  
 }  
 lvlChanger.startMenu();  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
});

При признании поражения сервер получает и перенаправляет:

case 4:  
 System.*out*.println("Player " + uuid + " give up");  
 newArray = new byte[buffer.array().length - buffer.position()];  
 buffer.get(newArray, buffer.arrayOffset(), newArray.length);  
 Message opponent = new Message(newArray);  
 os = sockets.get(opponent.getId()).getOutputStream();  
 buf = ByteBuffer.*allocate*(Integer.*BYTES*);  
 buf.putInt(4);  
 os.write(buf.array());  
 os.flush();  
 break;

Само признание поражения возможно при вызове меню на Esc. Для режима соревнования оно изменено и содержит лишь две кнопки. Также при нахождении в нем Timers не переключается, а значит время продолжает идти.



Рисунок 13. Меню признания поражения

Получив подобные сообщения, сервер шлет оппоненту уведомление о проигрыше или победе с заголовками «3» и «4» соответственно. На стороне клиента оно отображается как красная или зеленая надпись в окне справки.

case 3: //opponent win  
 Platform.*runLater*(() -> {  
 lvlChanger.getGameController().getStatusHelp().setText("Соперник победил!");  
 lvlChanger.getGameController().getStatusHelp().setTextFill(Color.*INDIANRED*);  
 });  
 break;  
  
case 4: //opponent give up  
 Platform.*runLater*(() -> {  
 lvlChanger.getGameController().getStatusHelp().setText("Соперник сдался!");  
 lvlChanger.getGameController().getStatusHelp().setTextFill(Color.*GREEN*);  
 });  
 break;

Для обмена сообщениями между сервером и клиентом используется буфер байтов, а также объекты класса Message.java. Данный класс позволяет записать UUID в буфер байтов.

У него два конструктора. Один позволяет получить данные напрямую, другой – сформировать UUID из буфера.

public class Message {  
 public int TYPE;  
 private static final int *SIZE* = Integer.*BYTES* + Long.*BYTES* \* 2;  
 private final UUID id;  
  
  
 public Message(UUID id, int type) {  
 this.id = id;  
 TYPE = type;  
 }  
  
 public Message(byte[] bytes) {  
 ByteBuffer buffer = ByteBuffer.*wrap*(bytes);  
 this.id = new UUID(buffer.getLong(), buffer.getLong());  
 }

Вытащить байтовый эквивалент данных сообщения позволяет следующий метод:

public byte[] getBytes() {  
 ByteBuffer buffer = ByteBuffer.*allocate*(*SIZE*);  
  
 buffer.putInt(TYPE);  
 buffer.putLong(id.getMostSignificantBits());  
 buffer.putLong(id.getLeastSignificantBits());  
  
 return buffer.array();  
}

В соревновательном режиме не работают подсказки, а у героя 1000 ходов. Рестарт работает, но не сбрасывает таймер. Побеждает самый быстрый игрок. Таким образом, не обязательно использовать самое малозатратное решение. Достаточно лишь быстрее добраться до цели.

Музыка

Для воспроизведения музыки используем отдельный класс Sound.java. В нем определим пути к музыкальным трекам, а также методы, проигрывающие либо музыку на заднем плане (с бесконечным повтором), либо звук определенного действия (воспроизводится один раз).

public class Sounds {  
  
 private MediaPlayer player;  
 private final Media mainTheme = new Media(Objects.*requireNonNull*(getClass().getResource("/sounds/mainTheme.mp3")).toString());  
 private final Media move = new Media(Objects.*requireNonNull*(getClass().getResource("/sounds/move.wav")).toString());  
  
 private void playSound(Media sound) {  
 player = new MediaPlayer(sound);  
 player.play();  
 }  
  
 private void playMusic(Media music) {  
 if (player != null) player.stop();  
 player = new MediaPlayer(music);  
 player.setCycleCount(MediaPlayer.*INDEFINITE*);  
 player.setVolume(0.1);  
 player.play();  
 }

У фоновой музыки уменьшим громкость, чтобы та не заглушала звуки действий.

Заключение

В ходе работы была создана игра в жанре сокобан. Реализовано главное меню, пять основных и один обучающий уровень, связь с локальной базой данных и сервером для игры по локальной сети. Основными элементами игры стали: главный герой, ящики, враги, сундук и ключ, шипы. В игре есть звуки и музыка. Реализован интерфейс, включающий в себя оставшееся количество ходов, время игры и справку, также отображающую подсказки для прохождения.

Во время работы возник ряд сложностей, которые потребуют дальнейшей разработки.

Во-первых, при онлайн-игре более двух клиентов не происходит привязка готовых игроков. Т.е. может случится ситуация, когда сформировалось две пары оппонентов, в каждой из них только один игрок нажал «Готов!», но запустились обе партии. Необходимо добавить отслеживание готовности конкретного оппонента.

Во-вторых, не хватило времени для реализации настроек. Смена разрешения изображения и полноэкранный режим не являются обязательными, но стали бы приятным дополнением. Также в настройках можно было бы добавить отключение (изменение громкости) музыки (звуков) и смену привязки клавиш к управлению.

В-третьих, добавление нескольких музыкальных треков в меню и самой игре. И большее количество контекстных звуков также не помешало бы. Например, звук нажатия на кнопку или звук сохранения результата в таблицу рекордов (особенно если этот результат входит в топ).

В-четвертых, добавление более проработанной анимации. Не простое «сдвинулся туда-обратно», а действительно удар. Или, например, не смену картинок при смерти врага, а падение или что-то в этом духе.

Однако как готовый бета-продукт игра вышла хорошей.

Все задания выполнены. Цель работы достигнута.

Приложение

Листинг программы:

Client.java

package game.sokoban;  
  
import game.sokoban.gameplay.Launcher;  
import javafx.application.Application;  
import javafx.stage.Stage;  
  
public class Client1 extends Application {  
 @Override  
 public void start(Stage stage) {  
 new Launcher(stage);  
 }  
  
 public static void main(String[] args) {  
 *launch*();  
 }  
}

Launcher.java

package game.sokoban.gameplay;  
  
import game.sokoban.gameplay.services.clientServer.Message;  
import javafx.application.Platform;  
import javafx.scene.paint.Color;  
import javafx.stage.Stage;  
  
import java.io.BufferedInputStream;  
import java.io.IOException;  
import java.net.Socket;  
import java.nio.ByteBuffer;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
import java.util.UUID;  
  
public class Launcher {  
  
 private boolean isSocketActive;  
 private Socket socket;  
 private UUID id, idOpponent;  
 private final LvlChanger lvlChanger;  
  
 public Launcher(Stage stage) {  
 startSocket();  
 javafx.scene.text.Font.*loadFont*(Objects.*requireNonNull*(getClass().getResource("/fonts/NineteenNinetyThree.ttf")).toExternalForm(), 16f);  
 stage.setTitle("Sokoban v. 0.9");  
 stage.setResizable(false);  
 lvlChanger = new LvlChanger(stage, this);  
 lvlChanger.startMenu();  
 stage.show();  
 }  
  
 public Socket getSocket() { return socket; }  
 public UUID getIdOpponent() { return idOpponent; }  
 public boolean isSocketActive() { return isSocketActive; }  
  
 public void startSocket() {  
 try {  
 socket = new Socket("localhost", 1984);  
 isSocketActive = true;  
 } catch (IOException e) {  
 System.*out*.println("Сервер не подключен!");  
 isSocketActive = false;  
 //e.printStackTrace();  
 }  
  
 Thread socketThread = new Thread(() -> {  
 while (true) {  
 try {  
 if (isSocketActive) {  
 BufferedInputStream reader = new BufferedInputStream(socket.getInputStream());  
 List<Byte> bytes = new ArrayList<>();  
 while (reader.available() > 0) {  
 bytes.add((byte) reader.read());  
 }  
  
 if (bytes.size() > 0) {  
 byte[] byteArray = new byte[bytes.size()];  
 for (int i = 0; i < bytes.size(); i++) {  
 byteArray[i] = bytes.get(i);  
 }  
 ByteBuffer buffer = ByteBuffer.*wrap*(byteArray);  
 int type = buffer.getInt();  
 byte[] newArray;  
 switch (type) {  
 case 0: //own uuid  
 newArray = new byte[buffer.array().length - buffer.position()];  
 buffer.get(newArray, buffer.arrayOffset(), newArray.length);  
 Message uuid = new Message(newArray);  
 id = uuid.getId();  
  
 System.*out*.println("own: " + id);  
 break;  
  
 case 1: //opponent uuid  
 newArray = new byte[buffer.array().length - buffer.position()];  
 buffer.get(newArray, buffer.arrayOffset(), newArray.length);  
 Message message = new Message(newArray);  
 idOpponent = message.getId();  
 lvlChanger.getMenuController().setOnlineOpponent();  
 System.*out*.println("opponent: " + message.getId());  
 break;  
  
 case 2: //all players ready  
 lvlChanger.getMenuController().setOnlineReady();  
 lvlChanger.reverseSingleGame();  
 lvlChanger.startGame(buffer.getInt());  
 break;  
  
 case 3: //opponent win  
 Platform.*runLater*(() -> {  
 lvlChanger.getGameController().getStatusHelp().setText("Соперник победил!");  
 lvlChanger.getGameController().getStatusHelp().setTextFill(Color.*INDIANRED*);  
 });  
 break;  
  
 case 4: //opponent give up  
 Platform.*runLater*(() -> {  
 lvlChanger.getGameController().getStatusHelp().setText("Соперник сдался!");  
 lvlChanger.getGameController().getStatusHelp().setTextFill(Color.*GREEN*);  
 });  
 break;  
 }  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 });  
  
 socketThread.start();  
 }  
}

LvlChanger.java

package game.sokoban.gameplay;  
  
import game.sokoban.Client1;  
import game.sokoban.gameplay.services.clientServer.Message;  
import game.sokoban.gameplay.controllers.GameController;  
import game.sokoban.gameplay.controllers.MenuController;  
import game.sokoban.gameplay.elements.\*;  
import game.sokoban.gameplay.services.DataBase;  
import game.sokoban.gameplay.services.Sounds;  
import game.sokoban.gameplay.services.Timers;  
import javafx.animation.FadeTransition;  
import javafx.application.Platform;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.scene.control.Label;  
import javafx.scene.image.Image;  
import javafx.scene.image.ImageView;  
import javafx.scene.input.KeyCode;  
import javafx.scene.input.KeyEvent;  
import javafx.scene.layout.GridPane;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.Pane;  
import javafx.scene.layout.VBox;  
import javafx.scene.paint.Color;  
import javafx.scene.shape.Rectangle;  
import javafx.stage.Stage;  
import javafx.util.Duration;  
  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.Objects;  
  
import static java.lang.Integer.*parseInt*;  
import static javafx.scene.input.KeyCode.*ESCAPE*;  
  
public class LvlChanger {  
  
 private final Stage stage;  
 private final int blockSize = Block.*blockSize*;  
 private final double errPos = 0.15\*blockSize;  
 private final int WIDTH = 900;  
 private final int HEIGHT = 800;  
 private int gWidth, gHeight;  
 private MenuController menuController;  
 private GameController gameController;  
 private final Launcher launcher;  
 private boolean isOpenWindow = false;  
 private boolean isOpenHelp = false;  
 private boolean isSingleGame = true;  
  
 private Timers timer;  
 private final DataBase DB = new DataBase();  
 private final Sounds player = new Sounds();  
  
 private int numLvl = 0;  
 private char[][] matrix;  
 private Hero hero;  
 private ImageView chest, key;  
 private final ArrayList<Block> listBoxAndEnemy = new ArrayList<>(), listSpike = new ArrayList<>();  
  
 Image imgFloor = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0017.png")), blockSize, blockSize, false, true);  
 Image imgWall1 = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0145.png")), blockSize, blockSize, false, true);  
 Image imgWall2 = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0148.png")), blockSize, blockSize, false, true);  
 Image imgWall3 = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0144.png")), blockSize, blockSize, false, true);  
 Image imgWall4 = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0147.png")), blockSize, blockSize, false, true);  
 Image imgWall5 = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0150.png")), blockSize, blockSize, false, true);  
 Image imgWall6 = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0153.png")), blockSize, blockSize, false, true);  
 Image imgWall7 = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0162.png")), blockSize, blockSize, false, true);  
 Image imgWall8 = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0163.png")), blockSize, blockSize, false, true);  
 Image imgColumn9 = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0030.png")), blockSize, blockSize, false, true);  
 Image imgBox = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0096.png")), 0.7\*blockSize, 0.7\*blockSize, false, true);  
 Image imgEnemy = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0010.png")), 0.7\*blockSize, 0.7\*blockSize, false, true);  
 Image imgSpike = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0018.png")), 0.9\*blockSize, 0.9\*blockSize, false, true);  
 Image imgHero = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0160.png")), 0.7\*blockSize, 0.7\*blockSize, false, true);  
 Image imgExit = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0052.png")), blockSize, blockSize, false, true);  
 Image imgExitEnd = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0161.png")), blockSize, blockSize, false, true);  
 Image imgChest = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0057.png")), 0.7\*blockSize, 0.7\*blockSize, false, true);  
 Image imgKey = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0090.png")), 0.4\*blockSize, 0.4\*blockSize, false, true);  
  
 LvlChanger(Stage primaryStage, Launcher launcher) {  
 stage = primaryStage;  
 stage.getIcons().add(imgChest);  
 this.launcher = launcher;  
 }  
  
 public void startMenu() {  
 try {  
 player.playMainMusic();  
 FXMLLoader fxml = new FXMLLoader(Client1.class.getResource("menu.fxml"));  
 Scene scene = new Scene(fxml.load(), WIDTH, HEIGHT);  
 menuController = fxml.getController();  
 menuController.initialize(this, launcher);  
 if (!launcher.isSocketActive()) menuController.setDisableOnlineBtn();  
 if (!DB.isActive()) menuController.setDisableRecordBtn();  
 stage.setScene(scene);  
 }  
 catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void startGame(int num) {  
 try {  
 player.playGameMusic();  
 FXMLLoader fxml = new FXMLLoader(Client1.class.getResource("game.fxml"));  
 Scene scene = new Scene(fxml.load(), WIDTH, HEIGHT);  
 LvlChanger temp = this;  
 Platform.*runLater*(() -> {  
 stage.setScene(scene);  
 gameController = fxml.getController();  
 gameController.initialize(temp, launcher);  
 loadLvl(gameController.getGameField(), num);  
 initKey(scene, gameController);  
 });  
  
 }  
 catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void initKey(Scene scene, GameController controller) {  
 scene.addEventFilter(KeyEvent.*KEY\_PRESSED*, evt -> {  
 Hero hero = getHero();  
 KeyCode code = evt.getCode();  
 if (!isOpenWindow)  
 switch (code) {  
 case *W*:  
 hero.move(hero.getX(), hero.getY() - 1, this);  
 break;  
 case *S*:  
 hero.move(hero.getX(), hero.getY() + 1, this);  
 break;  
 case *A*:  
 hero.move(hero.getX() - 1, hero.getY(), this);  
 break;  
 case *D*:  
 hero.move(hero.getX() + 1, hero.getY(), this);  
 break;  
 case *R*:  
 if (isSingleGame) timer.stop();  
 resetLvl(true);  
 break;  
 case *H*:  
 if (isSingleGame) {  
 controller.setStatusHelp(isOpenHelp, numLvl);  
 isOpenHelp = !isOpenHelp;  
 }  
 break;  
 }  
 if (code == *ESCAPE* && !controller.getWinWindow().isVisible()) {  
 if (isOpenWindow) { // open window  
 if (isSingleGame) controller.getMenuWindow().setVisible(false); // single game  
 else controller.getGiveupWindow().setVisible(false); // online game  
 } else { // close window  
 if (isSingleGame) controller.getMenuWindow().setVisible(true);  
 else controller.getGiveupWindow().setVisible(true);  
 }  
 reverseOpenWindow(); // reverse boolean marker  
 }  
 });  
 }  
  
 public void winLvl() {  
 timer.stop();  
 reverseOpenWindow();  
 closeHelp();  
 gameController.getSaveWinBtn().setDisable(numLvl == 0 || !DB.isActive()); // if lvl0 or DB isn't active  
 if (isSingleGame) { // single  
 gameController.getWinWindow().toFront();  
 gameController.getWinWindow().setVisible(true);  
 } else { // online  
 try {  
 gameController.getWinOnlineWindow().toFront();  
 gameController.getWinOnlineWindow().setVisible(true);  
 // send to server only if player wasn't win or lose before  
 if (getGameController().getStatusHelp().getTextFill() == Color.*WHITE*) {  
 OutputStream os = launcher.getSocket().getOutputStream();  
 os.write(new Message(launcher.getIdOpponent(), 3).getBytes());  
 os.flush();  
 }  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 }  
 }  
  
 public void saveRecordDB(String nick, VBox table) {  
 DB.add(nick, timer.getTime(), numLvl);  
 DB.writeToTable(table, numLvl);  
 }  
  
 public void clearRecordTable(VBox table) {  
 for (int i = 0; i < 10; i++) {  
 HBox line = (HBox) table.getChildren().get(i + 1);  
 Label nickname = (Label) line.getChildren().get(1);  
 Label time = (Label) line.getChildren().get(2);  
 nickname.setText("-");  
 time.setText("-");  
 }  
 }  
  
 public void resetLvl(boolean isRestart) {  
 Rectangle rct = new Rectangle(0, 0, WIDTH, HEIGHT);  
 rct.setFill(Color.*WHITE*);  
 gameController.getAnchorPane().getChildren().add(rct);  
  
 FadeTransition anim = new FadeTransition(Duration.*millis*(1500),rct);  
 anim.setFromValue(0.0);  
 anim.setToValue(1.0);  
 anim.play();  
  
 anim.setOnFinished(e -> {  
 timer.stop();  
 if (isRestart)  
 restartLvl(gameController.getGameField());  
 else loadLvl(gameController.getGameField(), numLvl);  
 FadeTransition anim1 = new FadeTransition(Duration.*millis*(1500),rct);  
 anim1.setFromValue(1.0);  
 anim1.setToValue(0.0);  
 anim1.play();  
 anim1.setOnFinished(ev -> gameController.getAnchorPane().getChildren().remove(rct));  
 });  
 }  
  
 public void clearLvl(Pane field) {  
 field.getChildren().clear();  
 key = null;  
 chest = null;  
 listBoxAndEnemy.clear();  
 listSpike.clear();  
 gameController.getStatusHelp().setTextFill(Color.*WHITE*);  
 gameController.getWinWindow().setVisible(false);  
 gameController.getMenuWindow().setVisible(false);  
 }  
  
 public void loadLvl(Pane field, int num) {  
 clearLvl(field);  
 numLvl = num;  
 if (numLvl == 6) startMenu();  
 else {  
 //create map  
 File fileMap = new File("levelConfig/maps/" + numLvl + ".conf");  
 field.getChildren().add(createGameField(fileMap));  
 //create blocks and hero  
 File fileBlock = new File("levelConfig/blocks/" + numLvl + ".conf");  
 field.getChildren().add(createElements(fileBlock));  
 //create local level timer  
 timer = new Timers(gameController.getStatusTimer(), this);  
 }  
 }  
  
 public void restartLvl(Pane field) {  
 clearLvl(field);  
 File fileMap = new File("levelConfig/maps/" + numLvl + ".conf");  
 field.getChildren().add(createGameField(fileMap));  
 File fileBlock = new File("levelConfig/blocks/" + numLvl + ".conf");  
 field.getChildren().add(createElements(fileBlock));  
 if (isSingleGame) timer = new Timers(gameController.getStatusTimer(), this);  
 }  
  
 public GridPane createGameField(File fileMap) {  
 try {  
 //find matrix height  
 BufferedReader inLen = new BufferedReader((new FileReader(fileMap.getAbsoluteFile())));  
 int i = 0;  
 while (inLen.readLine() != null) i++;  
 matrix = new char[i][];  
 inLen.close();  
 //create matrix  
 BufferedReader in = new BufferedReader((new FileReader(fileMap.getAbsoluteFile())));  
 String str;  
 int j = 0;  
 while ((str = in.readLine()) != null)  
 matrix[j++] = str.toCharArray();  
 in.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 //create blocks and add into grid  
 int mHeight = matrix.length;  
 int mWidth = matrix[0].length;  
  
 GridPane grid = new GridPane();  
 for (int i = 0; i < mHeight; i++) {  
 for (int j = 0; j < mWidth; j++) {  
 switch (matrix[i][j]) {  
 case '0':  
 grid.add(new ImageView(imgFloor), i, j);  
 break;  
 case '1':  
 grid.add(new ImageView(imgWall1), i, j);  
 break;  
 case '2':  
 grid.add(new ImageView(imgWall2), i, j);  
 break;  
 case '3':  
 grid.add(new ImageView(imgWall3), i, j);  
 break;  
 case '4':  
 grid.add(new ImageView(imgWall4), i, j);  
 break;  
 case '5':  
 grid.add(new ImageView(imgWall5), i, j);  
 break;  
 case '6':  
 grid.add(new ImageView(imgWall6), i, j);  
 break;  
 case '7':  
 grid.add(new ImageView(imgWall7), i, j);  
 break;  
 case '8':  
 grid.add(new ImageView(imgWall8), i, j);  
 break;  
 case '9':  
 grid.add(new ImageView(imgColumn9), i, j);  
 break;  
 case 'e':  
 grid.add(new ImageView(imgExit), i, j);  
 break;  
 case 'E':  
 grid.add(new ImageView(imgExitEnd), i, j);  
 break;  
 }  
 }  
 }  
 //delete padding between cells  
 grid.setVgap(0);  
 grid.setHgap(0);  
 //grid to center  
 gWidth = grid.getColumnCount() \* blockSize;  
 gHeight = grid.getRowCount() \* blockSize;  
 grid.setTranslateX((WIDTH - gWidth) / 2d);  
 grid.setTranslateY((HEIGHT - 50 - gHeight) / 2d);  
 return grid;  
 }  
  
 public Pane createElements(File fileBlock) {  
 Pane elemBox = new Pane(), boxBox = new Pane(), enemyBox = new Pane(), spikeBox = new Pane();  
 try {  
 BufferedReader in = new BufferedReader((new FileReader(fileBlock.getAbsoluteFile())));  
 String str;  
 ImageView img;  
 while ((str = in.readLine()) != null) {  
 String[] strSplit = str.split(" ");  
 switch (strSplit[0]) {  
 case "box":  
 img = new ImageView(imgBox);  
 boxBox.getChildren().add(img);  
 listBoxAndEnemy.add(new Block(*parseInt*(strSplit[1]), *parseInt*(strSplit[2]), img, "Box"));  
 break;  
 case "enemy":  
 img = new ImageView(imgEnemy);  
 enemyBox.getChildren().add(img);  
 listBoxAndEnemy.add(new Block(*parseInt*(strSplit[1]), *parseInt*(strSplit[2]), img, "Enemy"));  
 break;  
 case "spike":  
 img = new ImageView(imgSpike);  
 spikeBox.getChildren().add(img);  
 listSpike.add(new Block(*parseInt*(strSplit[1]), *parseInt*(strSplit[2]), img, "Spike"));  
 break;  
 case "chest":  
 chest = new ImageView(imgChest);  
 chest.setTranslateX(*parseInt*(strSplit[1]) \* blockSize + errPos);  
 chest.setTranslateY(*parseInt*(strSplit[2]) \* blockSize + errPos);  
 hero.setChestX(*parseInt*(strSplit[1]));  
 hero.setChestY(*parseInt*(strSplit[2]));  
 break;  
 case "key":  
 key = new ImageView(imgKey);  
 key.setTranslateX(*parseInt*(strSplit[1]) \* blockSize + 2\*errPos);  
 key.setTranslateY(*parseInt*(strSplit[2]) \* blockSize + 2\*errPos);  
 hero.setKeyX(*parseInt*(strSplit[1]));  
 hero.setKeyY(*parseInt*(strSplit[2]));  
 break;  
 case "hero":  
 img = new ImageView(imgHero);  
 hero = new Hero(*parseInt*(strSplit[1]), *parseInt*(strSplit[2]), *parseInt*(strSplit[3]), img, this);  
 break;  
 }  
 }  
 in.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 if (key != null) elemBox.getChildren().add(key);  
 if (chest != null) elemBox.getChildren().add(chest);  
 elemBox.getChildren().addAll(spikeBox, enemyBox, boxBox, hero.getImg());  
 elemBox.setTranslateX((WIDTH - gWidth) / 2d);  
 elemBox.setTranslateY((HEIGHT - 50 - gHeight) / 2d);  
 return elemBox;  
 }  
  
 public void closeHelp() {  
 gameController.setStatusHelp(true, numLvl);  
 isOpenHelp = false;  
 }  
 public void reverseOpenWindow() { isOpenWindow = !isOpenWindow; }  
 public void reverseSingleGame() { isSingleGame = !isSingleGame; }  
  
 public boolean isSingleGame() { return isSingleGame; }  
 public boolean isOpenWindow() { return isOpenWindow; }  
 public GameController getGameController() { return gameController; }  
 public MenuController getMenuController() { return menuController; }  
 public int getNumLvl() { return numLvl; }  
 public DataBase getDB() { return DB; }  
 public Hero getHero() { return hero; }  
 public ArrayList<Block> getListBoxAndEnemy() { return listBoxAndEnemy; }  
 public ArrayList<Block> getListSpike() { return listSpike; }  
 public char[][] getMatrix() { return matrix; }  
 public ImageView getKey() { return key; }  
 public ImageView getChest() { return chest; }  
}

GameController.java

package game.sokoban.gameplay.controllers;  
  
import game.sokoban.gameplay.services.clientServer.Message;  
import game.sokoban.gameplay.Launcher;  
import game.sokoban.gameplay.LvlChanger;  
import javafx.fxml.FXML;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.scene.control.TextField;  
import javafx.scene.layout.AnchorPane;  
import javafx.scene.layout.Pane;  
import javafx.scene.layout.VBox;  
import javafx.scene.paint.Color;  
  
import java.io.\*;  
  
public class GameController {  
  
 @FXML  
 private AnchorPane anchorPane;  
 @FXML  
 private Pane gameField;  
 @FXML  
 private Label statusTurns;  
 @FXML  
 private Label statusTimer;  
 @FXML  
 private Label statusHelp;  
 @FXML  
 private Pane winWindow;  
 @FXML  
 private Button resetWinBtn;  
 @FXML  
 private Button saveWinBtn;  
 @FXML  
 private Pane recordWindow;  
 @FXML  
 private VBox recordTable;  
 @FXML  
 private TextField nickTextField;  
 @FXML  
 private Button undoRecordBtn;  
 @FXML  
 private Button saveRecordBtn;  
 @FXML  
 private Button nextWinBtn;  
 @FXML  
 private Pane menuWindow;  
 @FXML  
 private Button exitMenuBtn;  
 @FXML  
 private Button mainMenuBtn;  
 @FXML  
 private Button resetMenuBtn;  
 @FXML  
 private Button continueMenuBtn;  
 @FXML  
 private Pane giveupWindow;  
 @FXML  
 private Button noOnlineBtn;  
 @FXML  
 private Button yesOnlineBtn;  
 @FXML  
 private Pane winOnlineWindow;  
 @FXML  
 private Button saveOnlineWinBtn;  
 @FXML  
 private Button exitOnlineWinBtn;  
  
 public AnchorPane getAnchorPane() { return anchorPane; }  
 public Pane getGameField() { return gameField; }  
 public Pane getWinWindow() { return winWindow; }  
 public Pane getMenuWindow() { return menuWindow; }  
 public Pane getGiveupWindow() { return giveupWindow; }  
 public Pane getWinOnlineWindow() { return winOnlineWindow; }  
 public Label getStatusTimer() { return statusTimer; }  
 public Label getStatusHelp() { return statusHelp; }  
 public Button getSaveWinBtn() { return saveWinBtn; }  
  
 @FXML  
 public void initialize(LvlChanger lvlChanger, Launcher launcher) {  
 nextWinBtn.setOnAction(e -> {  
 lvlChanger.loadLvl(gameField,lvlChanger.getNumLvl()+1);  
 lvlChanger.reverseOpenWindow();  
 });  
 saveWinBtn.setOnAction(e -> {  
 saveRecordBtn.setDisable(false);  
 lvlChanger.clearRecordTable(recordTable);  
 lvlChanger.getDB().writeToTable(recordTable, lvlChanger.getNumLvl());  
 recordWindow.toFront();  
 recordWindow.setVisible(true);  
 });  
 undoRecordBtn.setOnAction(e -> {  
 if (saveRecordBtn.isDisable()) {  
 saveWinBtn.setDisable(true);  
 saveOnlineWinBtn.setDisable(true);  
 }  
 recordWindow.setVisible(false);  
 });  
 saveRecordBtn.setOnAction(e -> {  
 if (!nickTextField.getText().equals("")) {  
 lvlChanger.saveRecordDB(nickTextField.getText(), recordTable);  
 saveRecordBtn.setDisable(true);  
 }  
 else nickTextField.setText("введите имя");  
 });  
 nickTextField.setOnMouseClicked(e -> {  
 if (nickTextField.getText().equals("введите имя (без пробелов)!")) nickTextField.setText("");  
 });  
 resetWinBtn.setOnAction(e -> {  
 lvlChanger.loadLvl(gameField,lvlChanger.getNumLvl());  
 lvlChanger.reverseOpenWindow();  
 });  
 continueMenuBtn.setOnAction(e -> {  
 menuWindow.setVisible(false);  
 lvlChanger.reverseOpenWindow();  
 });  
 resetMenuBtn.setOnAction(e -> {  
 lvlChanger.loadLvl(gameField,lvlChanger.getNumLvl());  
 lvlChanger.reverseOpenWindow();  
 });  
 mainMenuBtn.setOnAction(e -> {  
 lvlChanger.startMenu();  
 lvlChanger.reverseOpenWindow();  
 });  
 exitMenuBtn.setOnAction(e -> System.*exit*(0));  
 yesOnlineBtn.setOnAction(e -> {  
 try {  
 if (statusHelp.getTextFill() == Color.*WHITE*) {  
 OutputStream os = launcher.getSocket().getOutputStream();  
 os.write(new Message(launcher.getIdOpponent(), 4).getBytes());  
 os.flush();  
 }  
 lvlChanger.startMenu();  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 });  
 noOnlineBtn.setOnAction(e -> {  
 giveupWindow.setVisible(false);  
 lvlChanger.reverseOpenWindow();  
 });  
 saveOnlineWinBtn.setOnAction(e -> {  
 saveRecordBtn.setDisable(false);  
 lvlChanger.clearRecordTable(recordTable);  
 lvlChanger.getDB().writeToTable(recordTable, lvlChanger.getNumLvl());  
 recordWindow.toFront();  
 recordWindow.setVisible(true);  
 });  
 exitOnlineWinBtn.setOnAction(e -> lvlChanger.startMenu());  
 }  
  
 public void setStatusHelp(boolean isOpenHelp, int num) {  
 if (isOpenHelp) statusHelp.setText("R - заново, H - помощь, WASD - управление");  
 else  
 try {  
 File file = new File("levelConfig/help.txt");  
 BufferedReader in = new BufferedReader((new FileReader(file.getAbsoluteFile())));  
 int i = 1;  
 String str;  
 while ((str = in.readLine()) != null) {  
 if (i == num) {  
 statusHelp.setText(str);  
 break;  
 }  
 i++;  
 }  
 in.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void setTurnsLeft(int count) { statusTurns.setText("Ходов осталось: " + count);}  
}

MenuController.java

package game.sokoban.gameplay.controllers;  
  
import game.sokoban.gameplay.services.clientServer.Message;  
import game.sokoban.gameplay.Launcher;  
import game.sokoban.gameplay.LvlChanger;  
import javafx.application.Platform;  
import javafx.fxml.FXML;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.scene.layout.Pane;  
import javafx.scene.layout.VBox;  
  
import java.io.IOException;  
import java.io.OutputStream;  
import java.nio.ByteBuffer;  
  
public class MenuController {  
  
 @FXML  
 private Pane btnPane;  
  
 @FXML  
 private Pane lvlMenu;  
  
 @FXML  
 private Button startBtn;  
  
 @FXML  
 private Button lvlBtn;  
  
 @FXML  
 private Button lvlBtn1;  
  
 @FXML  
 private Button lvlBtn2;  
  
 @FXML  
 private Button lvlBtn3;  
  
 @FXML  
 private Button lvlBtn4;  
  
 @FXML  
 private Button lvlBtn5;  
  
 @FXML  
 private Button undoBtn;  
  
 @FXML  
 private Button onlineBtn;  
  
 @FXML  
 private Button recordBtn;  
  
 @FXML  
 private Pane recordPane;  
  
 @FXML  
 private Button recordBtn1;  
  
 @FXML  
 private Button recordBtn2;  
  
 @FXML  
 private Button recordBtn3;  
  
 @FXML  
 private Button recordBtn4;  
  
 @FXML  
 private Button recordBtn5;  
  
 @FXML  
 private VBox recordTable;  
  
 @FXML  
 private Button undoRecordBtn;  
  
 @FXML  
 private Pane onlinePane;  
  
 @FXML  
 private Label onlineLabel;  
  
 @FXML  
 private Button undoOnlineBtn;  
  
 @FXML  
 private Button readyOnlineBtn;  
  
 @FXML  
 private Button exitBtn;  
  
 @FXML  
 public void initialize(LvlChanger lvlChanger, Launcher launcher) {  
 startBtn.setOnAction(e -> lvlChanger.startGame(0));  
 recordBtn.setOnAction(e -> {  
 lvlChanger.clearRecordTable(recordTable);  
 recordPane.setVisible(true);  
 btnPane.setVisible(false);  
 });  
 recordBtn1.setOnAction(e -> lvlChanger.getDB().writeToTable(recordTable, 1));  
 recordBtn2.setOnAction(e -> lvlChanger.getDB().writeToTable(recordTable, 2));  
 recordBtn3.setOnAction(e -> lvlChanger.getDB().writeToTable(recordTable, 3));  
 recordBtn4.setOnAction(e -> lvlChanger.getDB().writeToTable(recordTable, 4));  
 recordBtn5.setOnAction(e -> lvlChanger.getDB().writeToTable(recordTable, 5));  
 undoRecordBtn.setOnAction(e -> {  
 recordPane.setVisible(false);  
 btnPane.setVisible(true);  
 });  
 lvlBtn.setOnAction(e -> {  
 btnPane.setVisible(false);  
 lvlMenu.setVisible(true);  
 });  
 lvlBtn1.setOnAction(e -> lvlChanger.startGame(1));  
 lvlBtn2.setOnAction(e -> lvlChanger.startGame(2));  
 lvlBtn3.setOnAction(e -> lvlChanger.startGame(3));  
 lvlBtn4.setOnAction(e -> lvlChanger.startGame(4));  
 lvlBtn5.setOnAction(e -> lvlChanger.startGame(5));  
 undoBtn.setOnAction(e -> {  
 btnPane.setVisible(true);  
 lvlMenu.setVisible(false);  
 });  
 onlineBtn.setOnAction(e -> {  
 try {  
 if (!isOnline) {  
 readyOnlineBtn.setDisable(true);  
 onlineLabel.setText("Ожидание игрока...");  
 }  
 onlinePane.setVisible(true);  
 btnPane.setVisible(false);  
 OutputStream os = launcher.getSocket().getOutputStream();  
 ByteBuffer buffer = ByteBuffer.*allocate*(Integer.*BYTES*);  
 buffer.putInt(1);  
 os.write(buffer.array());  
 os.flush();  
 }  
 catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 });  
 readyOnlineBtn.setOnAction(e -> {  
 try {  
 readyOnlineBtn.setDisable(true);  
 OutputStream os = launcher.getSocket().getOutputStream();  
 os.write(new Message(launcher.getIdOpponent(),2).getBytes());  
 os.flush();  
 }  
 catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 });  
 undoOnlineBtn.setOnAction(e -> {  
 onlinePane.setVisible(false);  
 btnPane.setVisible(true);  
 });  
 exitBtn.setOnAction(e -> System.*exit*(0));  
 }  
  
 boolean isOnline = false;  
  
 public void setOnlineOpponent() {  
 isOnline = true;  
 Platform.*runLater*(() -> onlineLabel.setText("Игрок найден!"));  
 readyOnlineBtn.setDisable(false);  
 }  
  
 public void setOnlineReady() {  
 Platform.*runLater*(() -> onlineLabel.setText("Игрок готов!"));  
 }  
  
 public void setDisableOnlineBtn() { onlineBtn.setDisable(true); }  
 public void setDisableRecordBtn() { recordBtn.setDisable(true); }  
}

Block.java

package game.sokoban.gameplay.elements;  
  
import game.sokoban.gameplay.LvlChanger;  
import javafx.animation.TranslateTransition;  
import javafx.scene.image.Image;  
import javafx.scene.image.ImageView;  
import javafx.util.Duration;  
  
import java.util.ArrayList;  
import java.util.Objects;  
  
public class Block {  
 public static int *blockSize* = 80;  
 String type;  
 int x, y; // coords on field  
 double errPos = 0.15\**blockSize*;  
 ImageView node;  
 Image imgDead = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0005.png")), 0.7\**blockSize*, 0.7\**blockSize*, false, true);  
 boolean isAlive = true;  
  
 public Block(int posX, int posY, ImageView img, String typeName) {  
 x = posX;  
 y = posY;  
 type = typeName;  
 node = img;  
 if (type.equals("Spike")) errPos = 0.05\**blockSize*;  
 node.setTranslateX(posX \* *blockSize* + errPos);  
 node.setTranslateY(posY \* *blockSize* + errPos);  
 }  
  
 public void move(int posX, int posY, LvlChanger lvlChanger) {  
 ArrayList<Block> listBoxAndEnemy = lvlChanger.getListBoxAndEnemy();  
 ArrayList<Block> listSpike = lvlChanger.getListSpike();  
 char[][] matrix = lvlChanger.getMatrix();  
 boolean isFree = true;  
  
 for (Block block: listBoxAndEnemy)  
 if (posX == block.getX() && posY == block.getY() && isAlive) isFree = false;  
  
 for (Block spike: listSpike)  
 if (posX == spike.getX() && posY == spike.getY() && type.equals("Enemy")) kill();  
  
 if ((matrix[posX][posY] == '0') && (isFree)) {  
 moveAnimation(posX, posY, lvlChanger);  
 x = posX;  
 y = posY;  
 }  
 else if (matrix[posX][posY] != '0' && type.equals("Enemy") && isFree) kill();  
 }  
  
 private void moveAnimation(int posX, int posY, LvlChanger lvlChanger) {  
 TranslateTransition anim = new TranslateTransition();  
 anim.setDuration(Duration.*millis*(lvlChanger.getHero().getAnimTime()));  
 anim.setNode(node);  
 if (posX < x) anim.setByX(-*blockSize*);  
 if (posX > x) anim.setByX(*blockSize*);  
 if (posY < y) anim.setByY(-*blockSize*);  
 if (posY > y) anim.setByY(*blockSize*);  
 anim.play();  
 }  
  
 public int getX() { return x; }  
 public int getY() { return y; }  
  
 public boolean getIsAlive() { return isAlive; }  
 public void kill() {  
 node.setImage(imgDead);  
 isAlive = false;  
 }  
}

Hero.java

package game.sokoban.gameplay.elements;  
  
import game.sokoban.gameplay.LvlChanger;  
import game.sokoban.gameplay.services.Sounds;  
import javafx.animation.FadeTransition;  
import javafx.animation.ScaleTransition;  
import javafx.animation.TranslateTransition;  
import javafx.scene.image.Image;  
import javafx.scene.image.ImageView;  
import javafx.scene.layout.Pane;  
import javafx.util.Duration;  
  
import java.util.ArrayList;  
import java.util.Objects;  
  
public class Hero {  
  
 private final int blockSize = Block.*blockSize*;  
 private int x, y;  
 private int keyX, keyY;  
 private int chestX, chestY;  
 private final int animTime = 100;  
 private final Image redImg = new Image(Objects.*requireNonNull*(getClass().getResourceAsStream("/images/tiles/tile\_0164.png")), 0.7\*blockSize, 0.7\*blockSize, false, true);  
 private final ImageView node;  
 private int turns = 1000;  
 private boolean isKey = false;  
 private Sounds player = new Sounds();  
  
  
 public Hero(int posX, int posY, int countTurns, ImageView img, LvlChanger lvlChanger) {  
 x = posX;  
 y = posY;  
 if (lvlChanger.isSingleGame()) turns = countTurns;  
 node = img;  
 node.setTranslateX(x \* blockSize + 0.15 \* blockSize);  
 node.setTranslateY(y \* blockSize + 0.15 \* blockSize);  
 lvlChanger.getGameController().setTurnsLeft(turns);  
 }  
  
 public void move(int posX, int posY, LvlChanger lvlChanger) {  
 ArrayList<Block> listBoxAndEnemy = lvlChanger.getListBoxAndEnemy();  
 ArrayList<Block> listSpike = lvlChanger.getListSpike();  
 char[][] matrix = lvlChanger.getMatrix();  
 if (turns > 0) {  
 boolean isFree = true;  
 //check boxes and enemies  
 for (Block block : listBoxAndEnemy) {  
 isFree = checkBlock(posX, posY, block, lvlChanger);  
 if (!isFree) break;  
 }  
 //check spikes  
 for (Block spike : listSpike) {  
 if (posX == spike.getX() && posY == spike.getY() && isFree) {  
 spikeAnimation(posX, posY);  
 turns--;  
 player.playSpikeSound();  
 }  
 }  
 //check chest if hero has key  
 if (posX == chestX && posY == chestY && isKey) {  
 Pane pane = (Pane) lvlChanger.getGameController().getGameField().getChildren().get(1);  
 pane.getChildren().remove(lvlChanger.getChest());  
 player.playChestSound();  
 }  
 //check chest if hero hasn't key  
 if (posX == chestX && posY == chestY && !isKey) {  
 isFree = false;  
 }  
 //check key  
 if (posX == keyX && posY == keyY && isFree) {  
 isKey = true;  
 Pane pane = (Pane) lvlChanger.getGameController().getGameField().getChildren().get(1);  
 pane.getChildren().remove(lvlChanger.getKey());  
 player.playChestSound();  
 }  
 //if free then hero moves  
 if (isFree && matrix[posX][posY] == '0') {  
 moveAnimation(posX, posY, blockSize, animTime, false, node);  
 x = posX;  
 y = posY;  
 player.playMoveSound();  
 }  
 else if (matrix[posX][posY] == 'e' || matrix[posX][posY] == 'E') winAnimation(posX, posY, lvlChanger);  
 else {  
 player.playKickSound();  
 kickAnimation(posX,posY);  
 }  
 turns--;  
 lvlChanger.getGameController().setTurnsLeft(turns);  
 }  
 else {  
 lvlChanger.resetLvl(false);  
 player.playGameOverSound();  
 }  
 }  
  
 public boolean checkBlock(int posX, int posY, Block block, LvlChanger lvlChanger) {  
 if ((posX == block.getX()) && (posY == block.getY()) && (block.getIsAlive())) {  
 if (posX < x)  
 block.move(posX - 1, posY, lvlChanger);  
 if (posX > x)  
 block.move(posX + 1, posY, lvlChanger);  
 if (posY < y)  
 block.move(posX, posY - 1, lvlChanger);  
 if (posY > y)  
 block.move(posX, posY + 1, lvlChanger);  
 return false;  
 }  
 return true;  
 }  
  
 private void moveAnimation(int posX, int posY, double distance, int time, boolean isKick, ImageView nd) {  
 TranslateTransition anim = new TranslateTransition(Duration.*millis*(time), nd);  
  
 if (isKick) anim.setDelay(Duration.*millis*(time));  
  
 if (posX < x) {  
 anim.setByX(-distance);  
 if (isKick) anim.setFromX(node.getTranslateX() + distance);  
 }  
 if (posX > x) {  
 anim.setByX(distance);  
 if (isKick) anim.setFromX(node.getTranslateX() - distance);  
 }  
 if (posY < y) {  
 anim.setByY(-distance);  
 if (isKick) anim.setFromY(node.getTranslateY() + distance);  
 }  
 if (posY > y) {  
 anim.setByY(distance);  
 if (isKick) anim.setFromY(node.getTranslateY() - distance);  
 }  
 anim.play();  
 }  
  
 private void spikeAnimation(int posX, int posY) {  
 //add red hero  
 ImageView redNode = new ImageView(redImg);  
 redNode.setTranslateX(node.getTranslateX());  
 redNode.setTranslateY(node.getTranslateY());  
 Pane pane = (Pane) node.getParent();  
 pane.getChildren().add(redNode);  
 node.toFront();  
 //move red hero as default hero  
 moveAnimation(posX, posY, blockSize, animTime, false, redNode);  
 //show red hero  
 FadeTransition damage = new FadeTransition(Duration.*millis*(animTime),node);  
 damage.setDelay(Duration.*millis*(0.5\*animTime));  
 damage.setToValue(0.6);  
 damage.setAutoReverse(true);  
 damage.setCycleCount(2);  
 damage.play();  
 //delete red hero after animation  
 damage.setOnFinished(e -> pane.getChildren().remove(redNode));  
 }  
  
 private void kickAnimation(int posX, int posY) {  
 moveAnimation(posX, posY, 0.25\*blockSize, animTime, false, node);  
 moveAnimation(posX, posY, -0.25\*blockSize, animTime, true, node);  
 }  
  
 private void winAnimation(int posX, int posY, LvlChanger lvlChanger) {  
 moveAnimation(posX, posY, 1.5\*blockSize, 3\*animTime, false, node);  
 ScaleTransition anim = new ScaleTransition(Duration.*millis*(3\*animTime), node);  
 anim.setByY(-0.2);  
 anim.setByX(-0.2);  
 anim.play();  
 FadeTransition anim1 = new FadeTransition(Duration.*millis*(3\*animTime), node);  
 anim1.setToValue(0.0);  
 anim1.play();  
 anim1.setOnFinished(e -> lvlChanger.winLvl());  
 }  
  
 public int getX() { return x; }  
 public int getY() { return y; }  
 public ImageView getImg() { return node; }  
 public int getAnimTime() { return animTime; }  
 public void setKeyX(int x) { keyX = x; }  
 public void setKeyY(int y) { keyY = y; }  
 public void setChestX(int x) { chestX = x; }  
 public void setChestY(int y) { chestY = y; }  
}

Timers.java

package game.sokoban.gameplay.services;  
  
import game.sokoban.gameplay.LvlChanger;  
import javafx.application.Platform;  
import javafx.scene.control.Label;  
  
public class Timers {  
 int sec = 0;  
 int min = 0;  
 boolean isActive = true;  
  
 public Timers(Label statusTimer, LvlChanger lvlChanger) {  
 Thread timer = new Thread(() -> {  
 while(isActive) {  
 try {  
 Platform.*runLater*(() -> statusTimer.setText(" Время: " + setTime()));  
 Thread.*sleep*(1000);  
 if (lvlChanger.isSingleGame()) {  
 if (!lvlChanger.isOpenWindow())  
 sec++;  
 } else sec++;  
 if (sec == 60) {  
 sec = 0;  
 min++;  
 }  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 });  
 timer.start();  
 }  
  
 private String setTime() {  
 String str;  
 if (min < 10) str = "0" + min + ":";  
 else str = min + ":";  
 if (sec < 10) str += "0" + sec;  
 else str += sec + "";  
 return str;  
 }  
  
 public void stop() { isActive = false; }  
  
 public int getTime() { return min\*60 + sec - 1; }  
}

Sounds.java

package game.sokoban.gameplay.services;  
  
import javafx.scene.media.Media;  
import javafx.scene.media.MediaPlayer;  
  
import java.util.Objects;  
  
public class Sounds {  
  
 private MediaPlayer player;  
 private final Media mainTheme = new Media(Objects.*requireNonNull*(getClass().getResource("/sounds/mainTheme.mp3")).toString());  
 private final Media gameTheme = new Media(Objects.*requireNonNull*(getClass().getResource("/sounds/sound.mp3")).toString());  
 private final Media move = new Media(Objects.*requireNonNull*(getClass().getResource("/sounds/move.wav")).toString());  
 private final Media moveSpike = new Media(Objects.*requireNonNull*(getClass().getResource("/sounds/moveSpike.wav")).toString());  
 private final Media moveChest = new Media(Objects.*requireNonNull*(getClass().getResource("/sounds/moveChest.mp3")).toString());  
 private final Media kick = new Media(Objects.*requireNonNull*(getClass().getResource("/sounds/kick.wav")).toString());  
 private final Media gameOver = new Media(Objects.*requireNonNull*(getClass().getResource("/sounds/gameover.mp3")).toString());  
  
 private void playSound(Media sound) {  
 player = new MediaPlayer(sound);  
 player.play();  
 }  
  
 private void playMusic(Media music) {  
 if (player != null) player.stop();  
 player = new MediaPlayer(music);  
 player.setCycleCount(MediaPlayer.*INDEFINITE*);  
 player.setVolume(0.1);  
 player.play();  
 }  
  
 public void playMainMusic() { playMusic(mainTheme); }  
 public void playGameMusic() { playMusic(gameTheme); }  
 public void playMoveSound() { playSound(move); }  
 public void playSpikeSound() { playSound(moveSpike); }  
 public void playChestSound() { playSound(moveChest); }  
 public void playKickSound() { playSound(kick); }  
 public void playGameOverSound() { playSound(gameOver); }  
}

DataBase.java

package game.sokoban.gameplay.services;  
  
import javafx.scene.control.Label;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.VBox;  
  
import java.sql.\*;  
  
public class DataBase {  
  
 private static final String *DRIVER* = "org.postgresql.Driver";  
 private static final String *DB\_URL* = "jdbc:postgresql://127.0.0.1:5432/Sokoban";  
 private static final String *USER* = "postgres";  
 private static final String *PASS* = "1234";  
  
 private boolean isActive;  
  
 public DataBase() {  
 try {  
 Class.*forName*(*DRIVER*);  
 Connection connection = DriverManager.*getConnection*(*DB\_URL*, *USER*, *PASS*);  
 Statement statement = connection.createStatement();  
 statement.close();  
 connection.close();  
 isActive = true;  
 } catch (Exception e) {  
 System.*out*.println("База данных не подключена!");  
 isActive = false;  
 //e.printStackTrace();  
 }  
 }  
  
 public void add(String nickname, int time, int numLvl) {  
 try {  
 Class.*forName*(*DRIVER*);  
 Connection c = DriverManager.*getConnection*(*DB\_URL*, *USER*, *PASS*);  
 Statement statement = c.createStatement();  
  
 c.setAutoCommit(false);  
 String sql = "INSERT INTO LEVEL" + numLvl + " (NICKNAME, TIME)"  
 + "VALUES ('" + nickname + "', " + time + ");";  
 statement.executeUpdate(sql);  
  
 statement.close();  
 c.commit();  
 c.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void writeToTable(VBox table, int numLvl) {  
 try {  
 Class.*forName*(*DRIVER*);  
 Connection c = DriverManager.*getConnection*(*DB\_URL*, *USER*, *PASS*);  
 Statement stmt = c.createStatement();  
 int i = 0;  
 ResultSet rs = stmt.executeQuery("SELECT NICKNAME, TIME FROM LEVEL" + numLvl + " ORDER BY TIME;");  
 while (rs.next()) {  
 HBox line = (HBox) table.getChildren().get(i + 1);  
 Label nickname = (Label) line.getChildren().get(1);  
 Label time = (Label) line.getChildren().get(2);  
 nickname.setText(rs.getString("nickname"));  
 time.setText(String.*valueOf*(rs.getInt("time")));  
 i++;  
 }  
 while (i < 10) { // fill empty lines  
 HBox line = (HBox) table.getChildren().get(i + 1);  
 Label nickname = (Label) line.getChildren().get(1);  
 Label time = (Label) line.getChildren().get(2);  
 nickname.setText("-");  
 time.setText("-");  
 i++;  
 }  
 stmt.close();  
 c.close();  
 } catch (Exception e) {  
 System.*err*.println(e.getClass().getName() + ": " + e.getMessage());  
 }  
 }  
  
 public boolean isActive() { return isActive; }  
}

Server.java

package game.sokoban.gameplay.services.clientServer;  
  
import java.io.BufferedInputStream;  
import java.io.IOException;  
import java.io.OutputStream;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.nio.ByteBuffer;  
import java.util.\*;  
  
public class Server {  
 ServerSocket server;  
 UUID player; // waiting player  
 Socket playerSocket; // waiting player's socket  
 int countPlayers = 0; // count of waiting player  
  
 {  
 try {  
 server = new ServerSocket(1984);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static void main(String[] args) {  
 try {  
 new Server();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public Server() throws IOException {  
 System.*out*.println("Server started");  
 Map<UUID, Socket> sockets = new HashMap<>();  
  
 Thread socketThread = new Thread(() -> {  
 while (true) {  
 try {  
 Socket newSocket = server.accept();  
 UUID newSocketId = UUID.*randomUUID*();  
  
 synchronized (sockets) {  
 sockets.put(newSocketId, newSocket);  
 }  
  
 synchronized (sockets) {  
 sockets.forEach((uuid, socket) -> {  
 try {  
 //send yourself your uuid  
 OutputStream os = socket.getOutputStream();  
 if (uuid == newSocketId) os.write(new Message(newSocketId, 0).getBytes());  
 os.flush();  
 } catch (Exception e) {  
 e.printStackTrace();  
 synchronized (sockets) {  
 sockets.remove(uuid);  
 }  
 }  
 });  
 }  
  
 System.*out*.println("Accepted " + newSocketId);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 });  
  
 socketThread.start();  
  
 while (true) {  
 synchronized (sockets) {  
 sockets.forEach((uuid, socket) -> {  
 try {  
 BufferedInputStream reader = new BufferedInputStream(socket.getInputStream());  
 List<Byte> byteList = new ArrayList<>();  
 while (reader.available() > 0) {  
 byteList.add((byte) reader.read());  
 }  
 if (byteList.size() != 0) {  
 byte[] bytes = new byte[byteList.size()];  
 for (int i = 0; i < byteList.size(); i++) {  
 bytes[i] = byteList.get(i);  
 }  
 ByteBuffer buffer = ByteBuffer.*wrap*(bytes);  
 int type = buffer.getInt();  
 byte[] newArray;  
 OutputStream os;  
 ByteBuffer buf;  
 switch (type) {  
 case 1:  
 System.*out*.println("Player " + uuid + " connected");  
 if (player == null) {  
 player = uuid;  
 playerSocket = socket;  
 } else {  
 os = socket.getOutputStream();  
 os.write(new Message(player, 1).getBytes());  
 os.flush();  
 os = playerSocket.getOutputStream();  
 os.write(new Message(uuid, 1).getBytes());  
 os.flush();  
 player = null;  
 playerSocket = null;  
 }  
 break;  
  
 case 2:  
 System.*out*.println("Player " + uuid + " ready");  
 countPlayers++;  
 newArray = new byte[buffer.array().length - buffer.position()];  
 buffer.get(newArray, buffer.arrayOffset(), newArray.length);  
 Message message = new Message(newArray);  
 System.*out*.println("Opponent: " + message.getId());  
 if (countPlayers == 2) {  
 countPlayers = 0;  
 buf = ByteBuffer.*allocate*(Integer.*BYTES* \* 2);  
 buf.putInt(2);  
 int num = (int) (Math.*random*() \* 3 + 1);  
 System.*out*.println("Random level " + num);  
 buf.putInt(num);  
 os = socket.getOutputStream();  
 os.write(buf.array());  
 os.flush();  
 os = sockets.get(message.getId()).getOutputStream();  
 os.write(buf.array());  
 os.flush();  
 }  
 break;  
  
 case 3:  
 System.*out*.println("Player " + uuid + " win");  
 newArray = new byte[buffer.array().length - buffer.position()];  
 buffer.get(newArray, buffer.arrayOffset(), newArray.length);  
 Message loser = new Message(newArray);  
 os = sockets.get(loser.getId()).getOutputStream();  
 buf = ByteBuffer.*allocate*(Integer.*BYTES*);  
 buf.putInt(3);  
 os.write(buf.array());  
 os.flush();  
 break;  
  
 case 4:  
 System.*out*.println("Player " + uuid + " give up");  
 newArray = new byte[buffer.array().length - buffer.position()];  
 buffer.get(newArray, buffer.arrayOffset(), newArray.length);  
 Message opponent = new Message(newArray);  
 os = sockets.get(opponent.getId()).getOutputStream();  
 buf = ByteBuffer.*allocate*(Integer.*BYTES*);  
 buf.putInt(4);  
 os.write(buf.array());  
 os.flush();  
 break;  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 });  
 }  
 }  
 }  
}

Message.java

package game.sokoban.gameplay.services.clientServer;  
  
import java.nio.ByteBuffer;  
import java.util.UUID;  
  
public class Message {  
 public int TYPE;  
 private static final int *SIZE* = Integer.*BYTES* + Long.*BYTES* \* 2;  
 private final UUID id;  
  
  
 public Message(UUID id, int type) {  
 this.id = id;  
 TYPE = type;  
 }  
  
 public Message(byte[] bytes) {  
 ByteBuffer buffer = ByteBuffer.*wrap*(bytes);  
 this.id = new UUID(buffer.getLong(), buffer.getLong());  
 }  
  
 public byte[] getBytes() {  
 ByteBuffer buffer = ByteBuffer.*allocate*(*SIZE*);  
  
 buffer.putInt(TYPE);  
 buffer.putLong(id.getMostSignificantBits());  
 buffer.putLong(id.getLeastSignificantBits());  
  
 return buffer.array();  
 }  
  
 public UUID getId() {  
 return id;  
 }  
}

game.fxml

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.geometry.Insets?>  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.HBox?>  
<?import javafx.scene.layout.Pane?>  
<?import javafx.scene.layout.VBox?>  
  
<AnchorPane fx:id="anchorPane" minHeight="800.0" minWidth="900.0" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1" fx:controller="game.sokoban.gameplay.controllers.GameController">  
  
 <Pane fx:id="gameField" minHeight="770.0" minWidth="900.0" style="-fx-background-color: #222323;" />  
  
 <Pane fx:id="recordWindow" layoutX="266.0" layoutY="143.0" prefHeight="490.0" prefWidth="368.0" styleClass="pane\_dark" stylesheets="@style.css" visible="false">  
 <Label alignment="TOP\_LEFT" layoutX="22.0" layoutY="400.0" prefHeight="20.0" prefWidth="136.0" styleClass="text\_small" stylesheets="@style.css" text="Введите свое имя:" />  
 <Label alignment="CENTER" layoutX="9.0" layoutY="14.0" prefHeight="58.0" prefWidth="350.0" styleClass="text\_large" stylesheets="@style.css" text="ТАБЛИЦА РЕКОРДОВ" />  
 <Button fx:id="undoRecordBtn" layoutX="37.0" layoutY="436.0" minHeight="30.0" minWidth="130.0" mnemonicParsing="false" styleClass="button\_dark" stylesheets="@style.css" text="Назад" />  
 <Button fx:id="saveRecordBtn" layoutX="196.0" layoutY="436.0" maxWidth="115.0" minHeight="30.0" minWidth="130.0" mnemonicParsing="false" styleClass="button\_dark" stylesheets="@style.css" text="Сохранить" />  
 <TextField fx:id="nickTextField" layoutX="159.0" layoutY="397.0" prefHeight="25.0" prefWidth="184.0" />  
 <VBox fx:id="recordTable" layoutX="22.0" layoutY="72.0" prefHeight="310.0" prefWidth="324.0" style="-fx-background-color: #222323; -fx-border-width: 5; -fx-border-color: darkgrey;">  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: darkgrey;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="№" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="НИКНЕЙМ" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="ВРЕМЯ (СЕК)" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: darkgrey;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="1" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: darkgrey;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="2" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: darkgrey;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="3" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: darkgrey;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="4" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: darkgrey;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="5" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: darkgrey;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="6" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: darkgrey;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="7" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: darkgrey;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="8" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: darkgrey;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="9" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: darkgrey;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="10" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" style="-fx-border-color: darkgrey; -fx-border-width: 0 1 0 0;" styleClass="text\_small" stylesheets="@style.css" text="-" textAlignment="CENTER" textFill="LIGHTGRAY" />  
 </HBox>  
 </VBox>  
 </Pane>  
  
 <Pane fx:id="winWindow" layoutX="275.0" layoutY="275.0" prefHeight="200.0" prefWidth="350.0" styleClass="pane\_dark" stylesheets="@style.css" visible="false">  
 <Label alignment="CENTER" layoutX="-1.0" layoutY="27.0" prefHeight="38.0" prefWidth="353.0" stylesheets="@style.css" text="Поздравляем!" />  
 <Label alignment="CENTER" layoutY="71.0" prefHeight="58.0" prefWidth="350.0" styleClass="text\_large" stylesheets="@style.css" text="ЗАГАДКА РЕШЕНА" />  
 <Button fx:id="resetWinBtn" layoutX="22.0" layoutY="145.0" minHeight="40.0" minWidth="90.0" mnemonicParsing="false" styleClass="button\_dark" stylesheets="@style.css" text="Еще раз" />  
 <Button fx:id="nextWinBtn" layoutX="238.0" layoutY="145.0" minHeight="40.0" minWidth="90.0" mnemonicParsing="false" styleClass="button\_dark" stylesheets="@style.css" text="Дальше" />  
 <Button fx:id="saveWinBtn" layoutX="127.0" layoutY="145.0" maxWidth="102.0" minHeight="40.0" minWidth="90.0" mnemonicParsing="false" prefHeight="40.0" prefWidth="102.0" styleClass="button\_dark" stylesheets="@style.css" text="Рекорды" textAlignment="CENTER" wrapText="true" />  
 </Pane>  
  
 <Pane fx:id="menuWindow" layoutX="285.0" layoutY="185.0" prefHeight="350.0" prefWidth="350.0" styleClass="pane\_dark" stylesheets="@style.css" visible="false">  
 <Label alignment="CENTER" layoutX="5.0" layoutY="32.0" prefHeight="38.0" prefWidth="341.0" styleClass="text\_large" stylesheets="@style.css" text="ПАУЗА" />  
 <Button fx:id="resetMenuBtn" layoutX="90.0" layoutY="150.0" minHeight="40.0" minWidth="170.0" mnemonicParsing="false" styleClass="button\_dark" stylesheets="@style.css" text="Начать заново" />  
 <Button fx:id="exitMenuBtn" layoutX="90.0" layoutY="270.0" minHeight="40.0" minWidth="170.0" mnemonicParsing="false" styleClass="button\_dark" stylesheets="@style.css" text="Выход из игры" />  
 <Button fx:id="continueMenuBtn" layoutX="90.0" layoutY="90.0" minHeight="40.0" minWidth="170.0" mnemonicParsing="false" styleClass="button\_dark" stylesheets="@style.css" text="Продолжить" />  
 <Button fx:id="mainMenuBtn" layoutX="90.0" layoutY="210.0" minHeight="40.0" minWidth="170.0" mnemonicParsing="false" styleClass="button\_dark" stylesheets="@style.css" text="Главное меню" />  
 </Pane>  
  
 <HBox id="HBox" alignment="CENTER\_LEFT" layoutY="770.0" minHeight="30.0" minWidth="900.0" spacing="5.0" style="-fx-background-color: #222323;">  
 <padding>  
 <Insets bottom="3.0" left="3.0" right="3.0" top="3.0" />  
 </padding>  
 <Label fx:id="statusTimer" maxHeight="1.7976931348623157E308" prefHeight="24.0" prefWidth="118.0" styleClass="text\_small" stylesheets="@style.css" text=" Время:" textFill="WHITE" HBox.hgrow="ALWAYS" />  
 <Pane prefHeight="24.0" prefWidth="585.0" HBox.hgrow="ALWAYS">  
 <Label fx:id="statusHelp" alignment="CENTER" prefHeight="24.0" prefWidth="602.0" styleClass="text\_small" stylesheets="@style.css" text="R - заново, H - помощь, WASD - управление" textFill="WHITE" />  
 </Pane>  
 <Label fx:id="statusTurns" prefHeight="20.0" prefWidth="164.0" styleClass="text\_small" stylesheets="@style.css" text="Ходов осталось:" textFill="WHITE" HBox.hgrow="NEVER" />  
 </HBox>  
 <Pane fx:id="giveupWindow" layoutX="316.0" layoutY="275.0" prefHeight="200.0" prefWidth="274.0" styleClass="pane\_dark" stylesheets="@style.css" visible="false">  
 <children>  
 <Label alignment="CENTER" layoutY="24.0" prefHeight="38.0" prefWidth="274.0" stylesheets="@style.css" text="Cдаться?" />  
 <Button fx:id="yesOnlineBtn" layoutX="86.0" layoutY="81.0" maxWidth="102.0" minHeight="40.0" minWidth="90.0" mnemonicParsing="false" prefHeight="40.0" prefWidth="102.0" styleClass="button\_dark" stylesheets="@style.css" text="Да" textAlignment="CENTER" wrapText="true" />  
 <Button fx:id="noOnlineBtn" layoutX="86.0" layoutY="139.0" maxWidth="102.0" minHeight="40.0" minWidth="90.0" mnemonicParsing="false" prefHeight="40.0" prefWidth="102.0" styleClass="button\_dark" stylesheets="@style.css" text="Нет" textAlignment="CENTER" wrapText="true" />  
 </children>  
 </Pane>  
 <Pane fx:id="winOnlineWindow" layoutX="285.0" layoutY="285.0" prefHeight="200.0" prefWidth="350.0" styleClass="pane\_dark" stylesheets="@style.css" visible="false">  
 <children>  
 <Label alignment="CENTER" layoutX="-1.0" layoutY="27.0" prefHeight="38.0" prefWidth="353.0" stylesheets="@style.css" text="Поздравляем!" />  
 <Label alignment="CENTER" layoutY="71.0" prefHeight="58.0" prefWidth="350.0" styleClass="text\_large" stylesheets="@style.css" text="ВЫ ПОБЕДИЛИ" />  
 <Button fx:id="exitOnlineWinBtn" layoutX="184.0" layoutY="145.0" minHeight="40.0" minWidth="150.0" mnemonicParsing="false" styleClass="button\_dark" stylesheets="@style.css" text="Главное меню" />  
 <Button fx:id="saveOnlineWinBtn" layoutX="17.0" layoutY="145.0" maxWidth="102.0" minHeight="40.0" minWidth="150.0" mnemonicParsing="false" prefHeight="40.0" prefWidth="102.0" styleClass="button\_dark" stylesheets="@style.css" text="Рекорды" textAlignment="CENTER" wrapText="true" />  
 </children>  
 </Pane>  
</AnchorPane>

menu.fxml

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.image.Image?>  
<?import javafx.scene.image.ImageView?>  
<?import javafx.scene.layout.HBox?>  
<?import javafx.scene.layout.Pane?>  
<?import javafx.scene.layout.VBox?>  
  
<Pane minHeight="800.0" minWidth="900.0" style="-fx-background-image: url(https://repository-images.githubusercontent.com/191390728/aa696780-8c69-11e9-84b1-687f30a61b27); -fx-background-repeat: stretch; -fx-background-size: 900 800;" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1" fx:controller="game.sokoban.gameplay.controllers.MenuController">  
 <Pane fx:id="btnPane" layoutX="340.0" layoutY="480.0" prefHeight="280.0" prefWidth="220.0">  
 <Button fx:id="onlineBtn" layoutY="120.0" minHeight="40.0" minWidth="220.0" mnemonicParsing="false" stylesheets="@style.css" text="Соревнование" />  
 <Button fx:id="lvlBtn" layoutY="60.0" minHeight="40.0" minWidth="220.0" mnemonicParsing="false" stylesheets="@style.css" text="Выбрать уровень" />  
 <Button fx:id="startBtn" minHeight="40.0" minWidth="220.0" mnemonicParsing="false" stylesheets="@style.css" text="Новая игра" />  
 <Button fx:id="recordBtn" layoutY="180.0" minHeight="40.0" minWidth="220.0" mnemonicParsing="false" stylesheets="@style.css" text="Таблица лидеров" />  
 <Button fx:id="exitBtn" layoutY="240.0" minHeight="40.0" minWidth="220.0" mnemonicParsing="false" stylesheets="@style.css" text="Выход" />  
 </Pane>  
 <Pane fx:id="lvlMenu" layoutX="140.0" layoutY="500.0" prefHeight="187.0" prefWidth="622.0" visible="false">  
 <Button fx:id="undoBtn" layoutX="234.0" layoutY="129.0" minHeight="50.0" minWidth="150.0" mnemonicParsing="false" stylesheets="@style.css" text="Назад" />  
 <Button fx:id="lvlBtn1" minHeight="109.0" minWidth="109.0" mnemonicParsing="false" styleClass="text\_large" stylesheets="@style.css" text="1" />  
 <Button fx:id="lvlBtn2" layoutX="128.0" minHeight="109.0" minWidth="109.0" mnemonicParsing="false" styleClass="text\_large" stylesheets="@style.css" text="2" />  
 <Button fx:id="lvlBtn3" layoutX="254.0" minHeight="109.0" minWidth="109.0" mnemonicParsing="false" styleClass="text\_large" stylesheets="@style.css" text="3" />  
 <Button fx:id="lvlBtn4" layoutX="382.0" minHeight="109.0" minWidth="109.0" mnemonicParsing="false" styleClass="text\_large" stylesheets="@style.css" text="4" />  
 <Button fx:id="lvlBtn5" layoutX="511.0" minHeight="109.0" minWidth="109.0" mnemonicParsing="false" prefHeight="109.0" prefWidth="109.0" styleClass="text\_large" stylesheets="@style.css" text="5" />  
 </Pane>  
 <ImageView fitHeight="214.0" fitWidth="357.0" layoutX="269.0" layoutY="293.0">  
 <Image url="@../../images/title2.png" />  
 </ImageView>  
 <Pane fx:id="recordPane" layoutX="202.0" layoutY="191.0" prefHeight="418.0" prefWidth="499.0" styleClass="pane" stylesheets="@style.css" visible="false">  
 <Label alignment="CENTER" layoutX="75.0" layoutY="14.0" prefHeight="58.0" prefWidth="350.0" styleClass="text\_large" stylesheets="@style.css" text="ТАБЛИЦА РЕКОРДОВ" />  
 <Button fx:id="undoRecordBtn" layoutX="360.0" layoutY="341.0" minHeight="30.0" minWidth="120.0" mnemonicParsing="false" stylesheets="@style.css" text="Назад" />  
 <VBox fx:id="recordTable" layoutX="22.0" layoutY="72.0" prefHeight="310.0" prefWidth="324.0" styleClass="pane" stylesheets="@style.css">  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: #775c2a;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" styleClass="hbox" stylesheets="@style.css" text="№" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="НИКНЕЙМ" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="ВРЕМЯ (СЕК)" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: #775c2a;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" styleClass="hbox" stylesheets="@style.css" text="1" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: #775c2a;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" styleClass="hbox" stylesheets="@style.css" text="2" textAlignment="CENTER" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: #775c2a;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" styleClass="hbox" stylesheets="@style.css" text="3" textAlignment="CENTER" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: #775c2a;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" styleClass="hbox" stylesheets="@style.css" text="4" textAlignment="CENTER" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: #775c2a;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" styleClass="hbox" stylesheets="@style.css" text="5" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: #775c2a;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" styleClass="hbox" stylesheets="@style.css" text="6" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: #775c2a;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" styleClass="hbox" stylesheets="@style.css" text="7" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: #775c2a;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" styleClass="hbox" stylesheets="@style.css" text="8" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: #775c2a;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" styleClass="hbox" stylesheets="@style.css" text="9" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 </HBox>  
 <HBox maxHeight="28.0" prefHeight="38.0" prefWidth="314.0" style="-fx-border-color: #775c2a;">  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="24.0" minWidth="24.0" styleClass="hbox" stylesheets="@style.css" text="10" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 <Label alignment="CENTER" maxHeight="28.0" maxWidth="150.0" minWidth="24.0" prefHeight="26.0" prefWidth="150.0" styleClass="hbox" stylesheets="@style.css" text="-" />  
 </HBox>  
 </VBox>  
 <Button fx:id="recordBtn1" layoutX="375.0" layoutY="80.0" minHeight="45.0" minWidth="90.0" mnemonicParsing="false" stylesheets="@style.css" text="1" />  
 <Button fx:id="recordBtn2" layoutX="375.0" layoutY="130.0" minHeight="45.0" minWidth="90.0" mnemonicParsing="false" stylesheets="@style.css" text="2" />  
 <Button fx:id="recordBtn3" layoutX="375.0" layoutY="180.0" minHeight="45.0" minWidth="90.0" mnemonicParsing="false" stylesheets="@style.css" text="3" />  
 <Button fx:id="recordBtn4" layoutX="375.0" layoutY="230.0" minHeight="45.0" minWidth="90.0" mnemonicParsing="false" stylesheets="@style.css" text="4" />  
 <Button fx:id="recordBtn5" layoutX="375.0" layoutY="280.0" minHeight="45.0" minWidth="90.0" mnemonicParsing="false" stylesheets="@style.css" text="5" />  
 </Pane>  
 <Pane fx:id="onlinePane" layoutX="312.0" layoutY="329.0" prefHeight="200.0" prefWidth="277.0" styleClass="pane" stylesheets="@style.css" visible="false">  
 <children>  
 <Button fx:id="readyOnlineBtn" disable="true" layoutX="146.0" layoutY="146.0" minHeight="40.0" minWidth="109.0" mnemonicParsing="false" prefHeight="40.0" prefWidth="109.0" stylesheets="@style.css" text="Готов!" />  
 <Button fx:id="undoOnlineBtn" layoutX="22.0" layoutY="146.0" minHeight="40.0" minWidth="109.0" mnemonicParsing="false" prefHeight="40.0" prefWidth="109.0" stylesheets="@style.css" text="Назад" />  
 <ImageView fitHeight="79.0" fitWidth="79.0" layoutX="99.0" layoutY="53.0">  
 <image>  
 <Image url="@../../images/loading.gif" />  
 </image>  
 </ImageView>  
 <Label fx:id="onlineLabel" alignment="CENTER" layoutX="14.0" layoutY="15.0" prefHeight="32.0" prefWidth="250.0" stylesheets="@style.css" text="Поиск соперника..." />  
 </children>  
 </Pane>  
 <Label alignment="CENTER" layoutX="-2.0" layoutY="7.0" prefHeight="16.0" prefWidth="900.0" style="-fx-text-fill: #c4bb8c;" styleClass="text\_small" stylesheets="@style.css" text="Интеллектуальная собственность без прав, но с большими амбициями. Относитесь с уважением! © М. К. Сивирилова, НГТУ, 2022" />  
</Pane>

style.css

.button {  
 -fx-font-family: "Nineteen Ninety Three";  
 -fx-font-size: 15;  
 -fx-background-color: #ded6ab;  
 -fx-border-width: 2;  
 -fx-border-color: #775c2a;  
 -fx-border-insets: 2;  
 -fx-border-style: dashed;  
}  
.button\_dark {  
 -fx-background-color: darkgrey;  
 -fx-border-color: #222323;  
}  
.label {  
 -fx-font-family: "Nineteen Ninety Three";  
 -fx-font-size: 22;  
}  
.pane {  
 -fx-background-color: #c4bb8c;  
 -fx-border-color: #775c2a;  
 -fx-border-width: 5;  
}  
.pane\_dark {  
 -fx-background-color: lightgrey;  
 -fx-border-color: darkgrey;  
 -fx-border-width: 5;  
}  
.hbox {  
 -fx-border-color: #775c2a;  
 -fx-border-width: 0 1 0 0;  
 -fx-font-family: "Nineteen Ninety Three";  
 -fx-font-size: 11;  
 -fx-text-fill: black;  
 -fx-text-alignment: CENTER;  
}  
.text\_large {  
 -fx-font-family: "Nineteen Ninety Three";  
 -fx-font-size: 26;  
}  
.text\_small {  
 -fx-font-family: "Nineteen Ninety Three";  
 -fx-font-size: 11;