Université des Mascareignes

Faculté des technologies de l'information et de la

Communication

Bsc Hons Informatique Appliquée

Project “JUMP”

Soumis par : BALLAH Vayuna

Norbert Ronan Adrien

Hachim Hassani Bacar

Soumis à : Mr. Khadimoullah Ramoth

Date : 26/02/2024

Table of Contents **1**

Table of figures

UNIVERSITÉ DES MASCAREIGNES
SAVOIR, C'EST POUVOIR

Université de Limoges

JUMP

Introduction

Created with JavaFX, "Jump" is an exciting platformer game in which players control a cube through a network of obstacles to gather yellow squares, each of which will add up to the score. Inspired by games like Geometry Dash, "Jump" has quick action and a precision-based gaming mechanic. Yellow cubes that players earn as they advance are shown on a unique score screen, heightening the tension and conflict. But they have to dodge dangerous red cubes that are positioned as formidable obstacles along the way. "Jump" promises an immersive experience that will keep players on the edge of their seats as they leap, dodge, and strive for mastery through each thrilling level. It does this by having a minimalist yet engaging design and simple yet intuitive controls.

Summary

"Jump" is a JavaFX platformer game inspired by Geometry Dash. Controlling a cube, players must avoid red cubes, which act as obstacles, in order to gather yellow squares, which represent coins. The game has a simple, minimalistic design, precision gameplay, and fast-paced action. As they go through levels, players collect coins that are shown on a score panel, which heightens the difficulty and excitement. "Jump" provides an immersive gameplay experience with easy-to-use controls, allowing players to traverse through exciting levels full of obstacles and rewards.

Methodology

The first step in creating "Jump" with JavaFX in Eclipse was to conceptualise the layout and gameplay of the game. In addition to creating the user interface layout for screens such as the start, tutorial, game, and end screens, we also specified the essential gameplay components such as coin collection, obstacle avoidance, and player controls. We used a simple CSS-based style with complimentary colours to improve the game's aesthetics while keeping things simple. The next step was to set up Eclipse for JavaFX development, making sure we had the correct environment to execute our concept.

We started coding the various panels and capabilities throughout the implementation phase. Players were met with the start screen, which offered options to launch the game, go to the tutorial, or end it. Within the tutorial screen, the tutorial level offered clear directions on gaming mechanics. Regarding the main game, we created a challenging environment in which players had to move through stages, gather coins, and dodge obstacles. The final screen displayed the player's performance, their score, and options to finish the game, go back to the menu, or start over.

We used inline CSS to achieve the desired visual aesthetic, and JavaFX was our main tool for handling user input, game logic, and screen transitions during the developing process. We tested all of the game's components during development to make sure they worked together and as a whole.

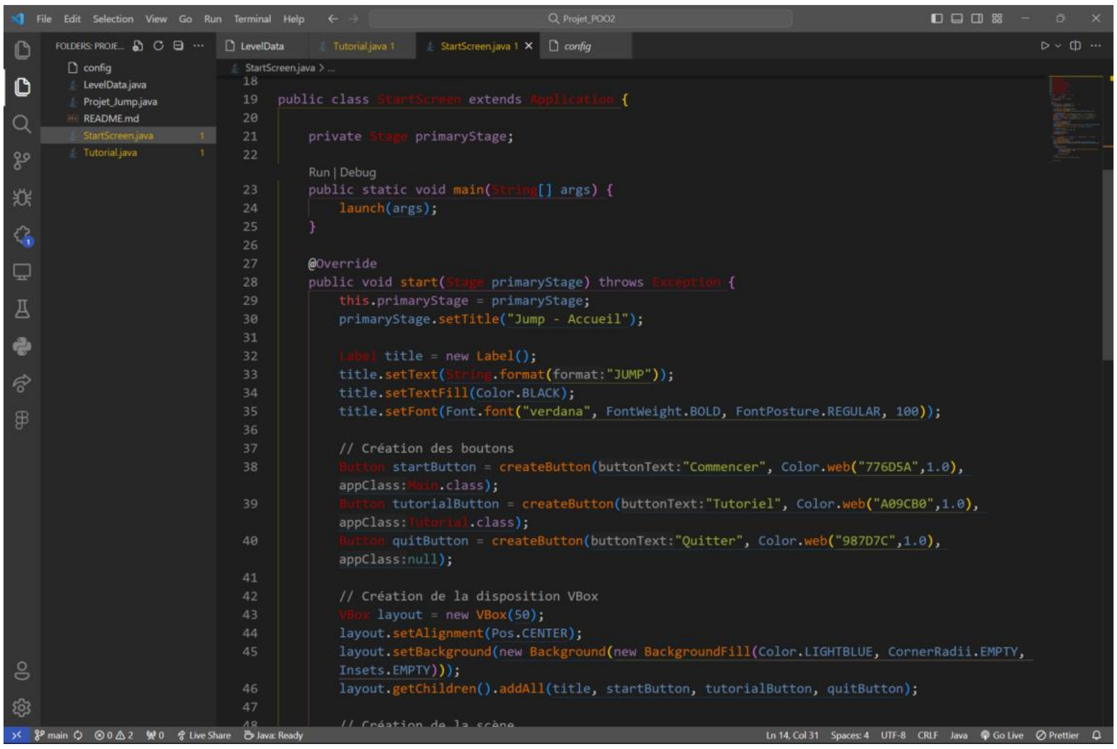
Development Process

Start Screen java

The Start Screen is basically the screen that the user will interact with to play the game. The code below shows how to build up the main screen of the JavaFX game "Jump." It builds a graphical user interface (GUI) with buttons for launching the game, viewing a tutorial, and ending it by utilising a number of JavaFX features and components. To arrange the UI elements nicely, a vertical box (VBox) is used in the layout. The buttons are designed with particular fonts, colours, and sizes in mind to improve both use and aesthetic appeal. The code also has button click event handlers, allowing for things like launching new game stages or shutting down the application. All things considered, this implementation demonstrates how versatile JavaFX is for creating engaging and eye-catching user interfaces for Java applications.

Mr. Norbert A. R was in charge of the development of the game mechanics like movement, jump and death. Mr. Bacar H. H was in charge of the screens, for example: the start screen, death screens and win screens. Ms. Ballah V was responsible for the aesthetics and design of the levels in the general game and the tutorial.

Below is a little snippet of the code



```java
public class StartScreen extends Application {

    private Stage primaryStage;

    Run | Debug
    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) throws Exception {
        this.primaryStage = primaryStage;
        primaryStage.setTitle("Jump - Accueil");

        Label title = new Label();
        title.setText(String.format(format:"JUMP"));
        title.setTextFill(Color.BLACK);
        title.setFont(Font.font("verdana", FontWeight.BOLD, FontPosture.REGULAR, 100));

        // Création des boutons
        Button startButton = createButton(buttonText:"Commencer", Color.web("776D5A",1.0),
        appClass:Main.class);
        Button tutorialButton = createButton(buttonText:"Tutoriel", Color.web("A09CB0",1.0),
        appClass:Tutorial.class);
        Button quitButton = createButton(buttonText:"Quitter", Color.web("987D7C",1.0),
        appClass:null);

        // Création de la disposition VBox
        VBox layout = new VBox(50);
        layout.setAlignment(Pos.CENTER);
        layout.setBackground(new Background(new BackgroundFill(Color.LIGHTBLUE, CornerRadii.EMPTY,
        Insets.EMPTY)));
        layout.getChildren().addAll(title, startButton, tutorialButton, quitButton);

        // Création de la scène
```

Figure 1

Below is the Start Screen with simple CSS styling

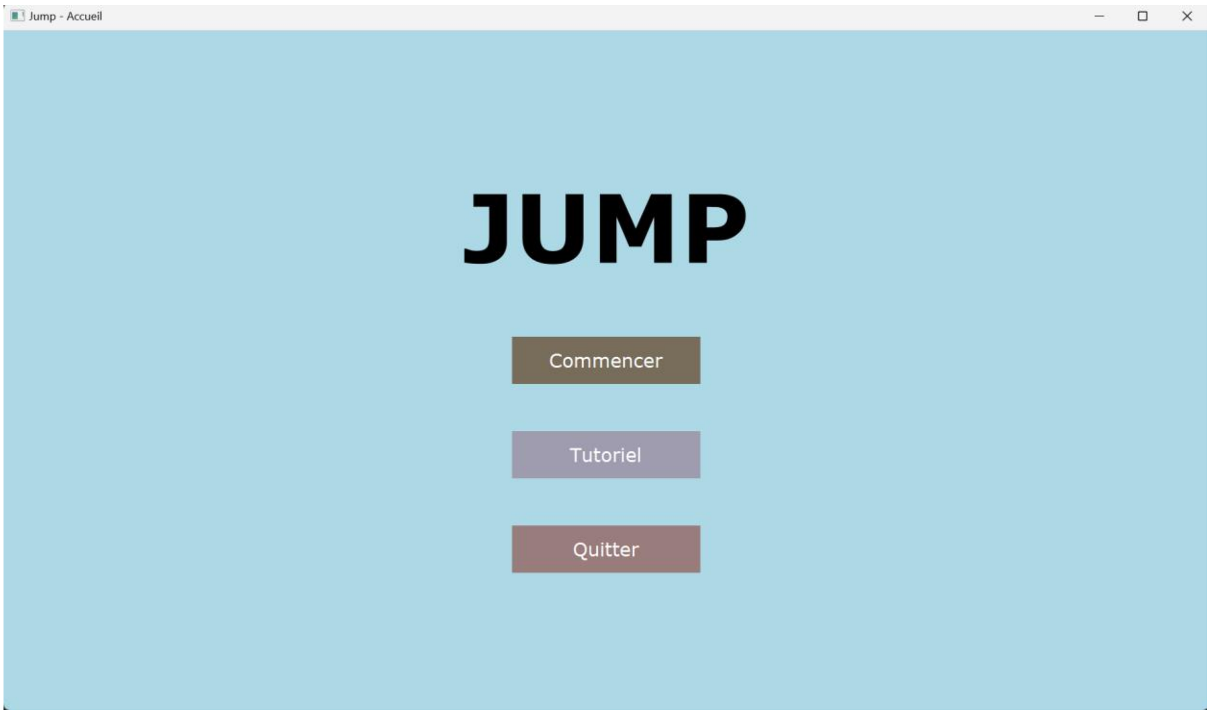
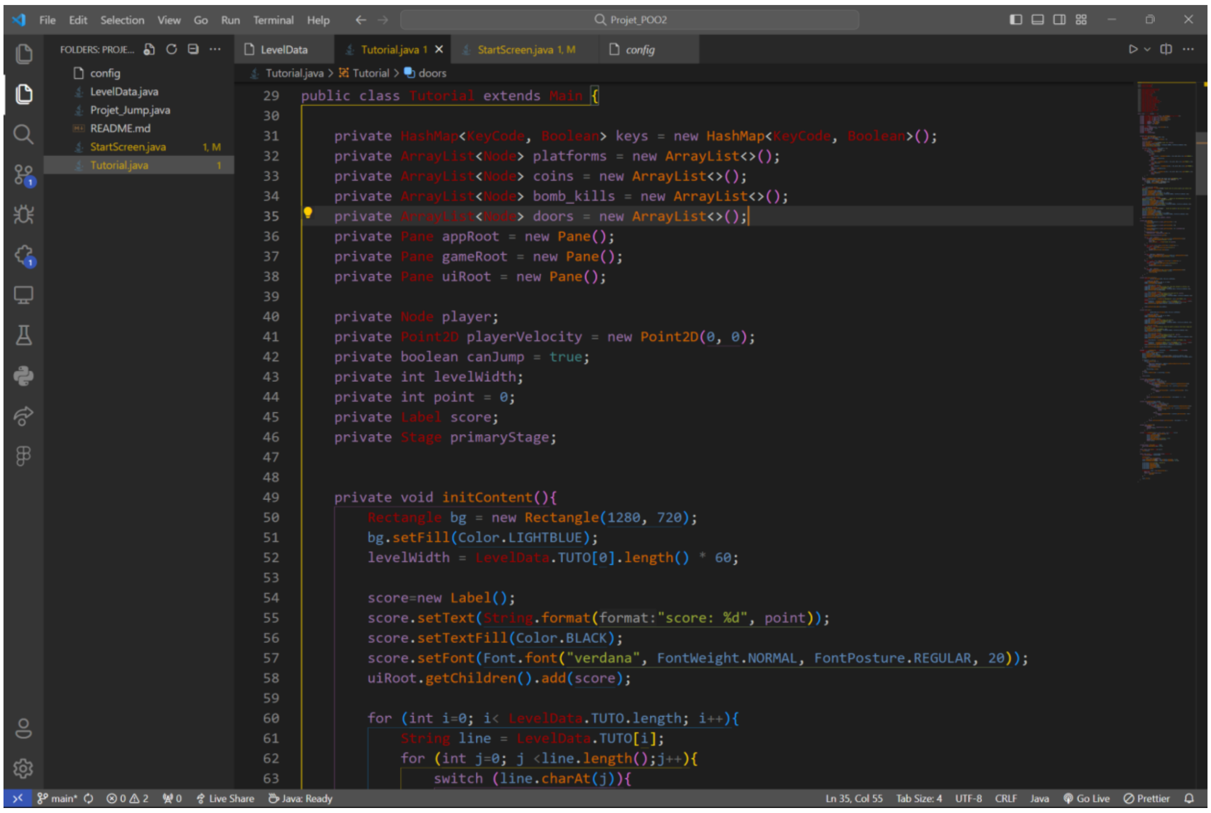


Figure 2

Tutorial Level Data

Players are guided through the gaming mechanics in the "Tutorial" level of the "Jump" game by having simple and understandable explanations provided at strategic points. As they go, gamers will come across tutorial instructions that walk them through each step of how to play the game properly. The placement of these messages is intentional in order to align with important gameplay features like platform jumping, gathering points from yellow squares, and avoiding red obstacles. The incorporation of instructional messaging into the gameplay experience facilitates player comprehension of the game concepts and helps them overcome obstacles. This methodology guarantees that players obtain prompt instruction and assistance, augmenting their comprehension of the game and promoting a more pleasurable gaming encounter.

Below is a little snippet of the code


File Edit Selection View Go Run Terminal Help
Projet_POO2
FOLDERS: PROJE...
config
LevelData.java
Projet_Jump.java
README.md
StartScreen.java 1, M
Tutorial.java 1
LevelData
Tutorial.java 1
StartScreen.java 1, M
config
Tutorial.java > Tutorial > doors
main* ⊗ 0 ⚠ 2 Live Share Java: Ready
Ln 35, Col 55 Tab Size: 4 UTF-8 CRLF Java Go Live Prettier

```java
public class Tutorial extends Main {

    private HashMap<KeyCode, Boolean> keys = new HashMap<KeyCode, Boolean>();
    private ArrayList<Node> platforms = new ArrayList<>();
    private ArrayList<Node> coins = new ArrayList<>();
    private ArrayList<Node> bomb_kills = new ArrayList<>();
    private ArrayList<Node> doors = new ArrayList<>();
    private Pane appRoot = new Pane();
    private Pane gameRoot = new Pane();
    private Pane uiRoot = new Pane();

    private Node player;
    private Point2D playerVelocity = new Point2D(0, 0);
    private boolean canJump = true;
    private int levelWidth;
    private int point = 0;
    private Label score;
    private Stage primaryStage;


    private void initContent(){
        Rectangle bg = new Rectangle(1280, 720);
        bg.setFill(Color.LIGHTBLUE);
        levelWidth = LevelData.TUTO[0].length() * 60;

        score=new Label();
        score.setText(String.format(format:"score: %d", point));
        score.setTextFill(Color.BLACK);
        score.setFont(Font.font("verdana", FontWeight.NORMAL, FontPosture.REGULAR, 20));
        uiRoot.getChildren().add(score);

        for (int i=0; i< LevelData.TUTO.length; i++){
            String line = LevelData.TUTO[i];
            for (int j=0; j <line.length();j++){
                switch (line.charAt(j)){
```

Figure 3

Below is the tutorial screen

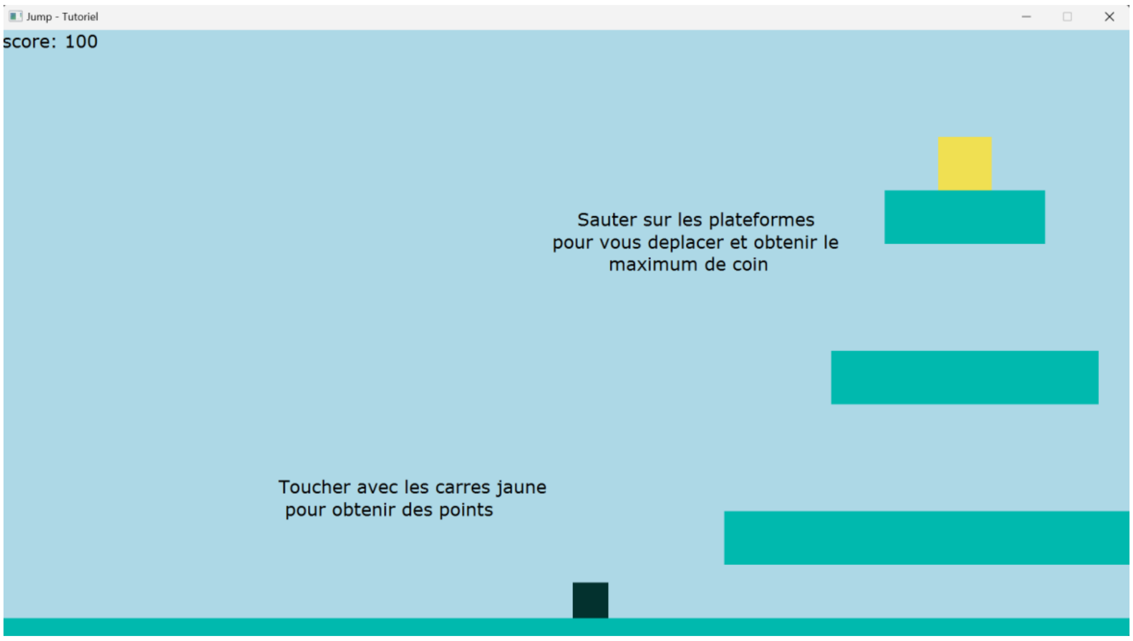

Jump - Tutoriel
score: 100
Sauter sur les plateformes
pour vous deplacer et obtenir le
maximum de coin
Toucher avec les carres jaune
pour obtenir des points

Figure 4

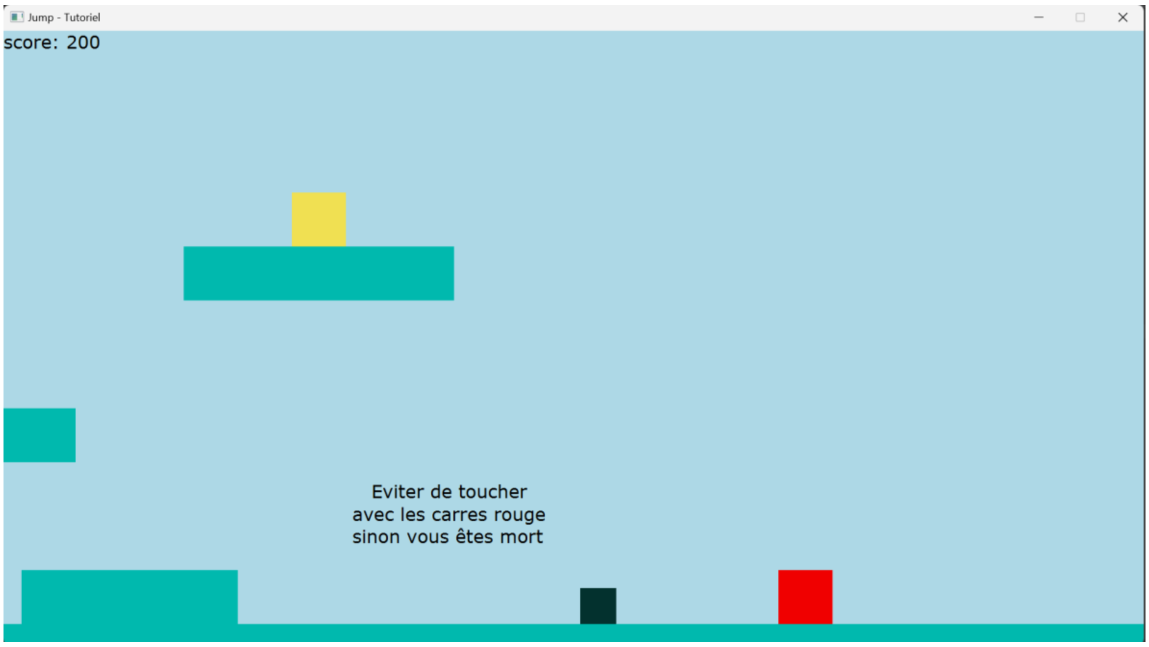
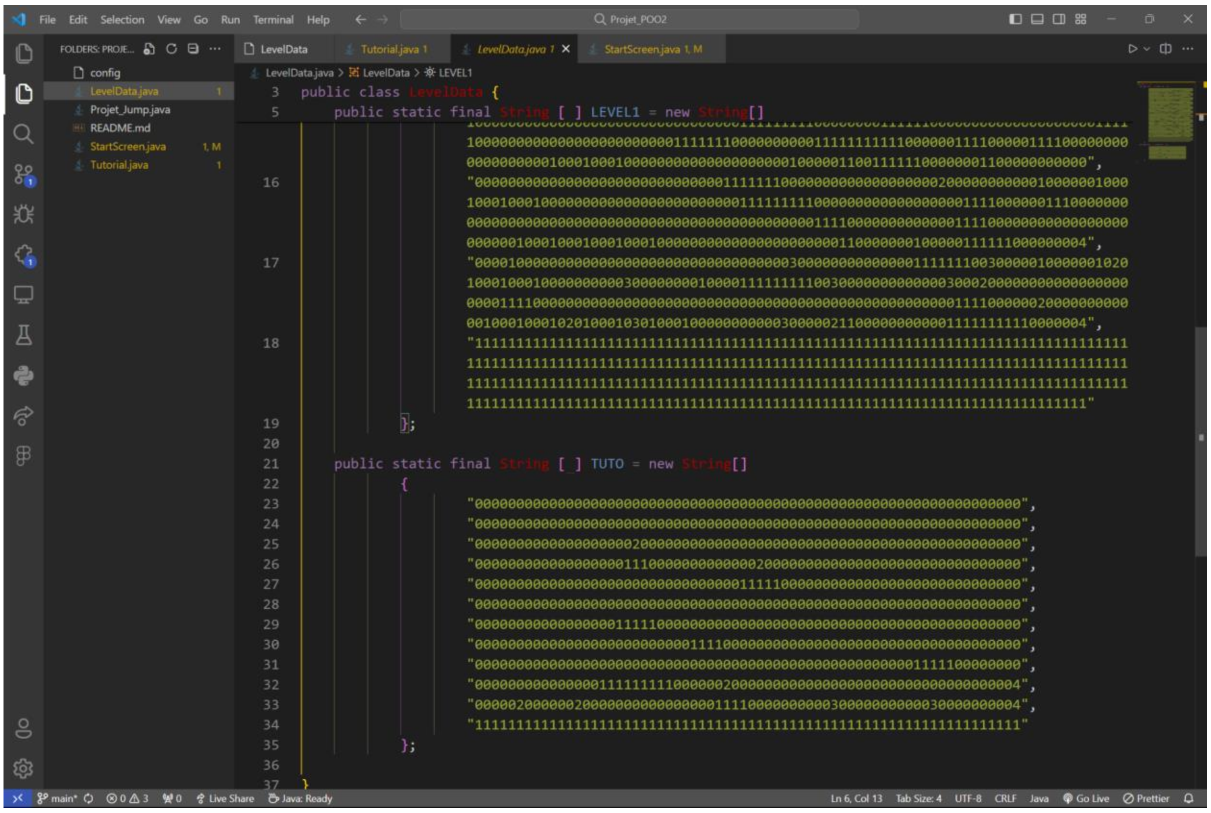


Figure 5

Level Data

The LevelData class is crucial to the "Jump" game since it controls how the various game levels are set up. The main game levels and the tutorial level are the two main kinds of levels that players face. Every level has been carefully designed to provide an unique enjoyable experience.
Players navigate through various surroundings featuring platforms, coins to collect, obstacles to avoid, and doors to open in the main game levels. As players move through the levels, they must improve their skills and tactics due to the levels' increasing complexity and challenge.
However, the Tutorial level acts as an introduction, giving players all the necessary instructions on how to play the game. Players learn the basics of jumping, collecting coins, and avoiding obstacles in this level. The tutorial level is made to be easier to understand and less intimidating than the main game levels, giving players a safe space to get familiar with the game concepts.

Below is a snippet of the code



```java
public class LevelData {
    public static final String [ ] LEVEL1 = new String[]
                        10000000000000000000000000111111000000000000111111111100000011110000011110000000000
                        0000000000100010001000000000000000000000001000001100111111000000011000000000000",
                        "000000000000000000000000000001111110000000000000000000020000000000010000001000
                        10001000100000000000000000000000001111111100000000000000000000111100000011100000000
                        00000000000000000000000000000000000000000001111000000000000001111000000000000000000
                        0000001000100010001000100000000000000000000000011000000010000011111100000000004",
                        "00001000000000000000000000000000000000003000000000000000111111003000001000000001020
                        10001000100000000000300000000100001111111100300000000000000030002000000000000000000
                        00001111000000000000000000000000000000000000000000000000000011110000000200000000000
                        0010001000102010001030100010000000000003000002110000000000001111111110000004",
                        "11111111111111111111111111111111111111111111111111111111111111111111111111111111111
                        111111111111111111111111111111111111111111111111111111111111111111111111111111111111
                        111111111111111111111111111111111111111111111111111111111111111111111111111111111111
                        11111111111111111111111111111111111111111111111111111111111111111111111111111111"
                };

    public static final String [ ] TUTO = new String[]
                {
                        "0000000000000000000000000000000000000000000000000000000000000000000000",
                        "0000000000000000000000000000000000000000000000000000000000000000000000",
                        "0000000000000000000020000000000000000000000000000000000000000000000000",
                        "0000000000000000000111000000000000002000000000000000000000000000000000",
                        "0000000000000000000000000000000000111110000000000000000000000000000000",
                        "0000000000000000000000000000000000000000000000000000000000000000000000",
                        "0000000000000000001111100000000000000000000000000000000000000000000000",
                        "0000000000000000000000000000111100000000000000000000000000000000000000",
                        "0000000000000000000000000000000000000000000000000000000001111100000000",
                        "0000000000000000111111111000000200000000000000000000000000000000000004",
                        "0000020000000200000000000000000001111000000000003000000000000030000000004",
                        "1111111111111111111111111111111111111111111111111111111111111111111111"
                };

}
```

Figure 6

Project Jump

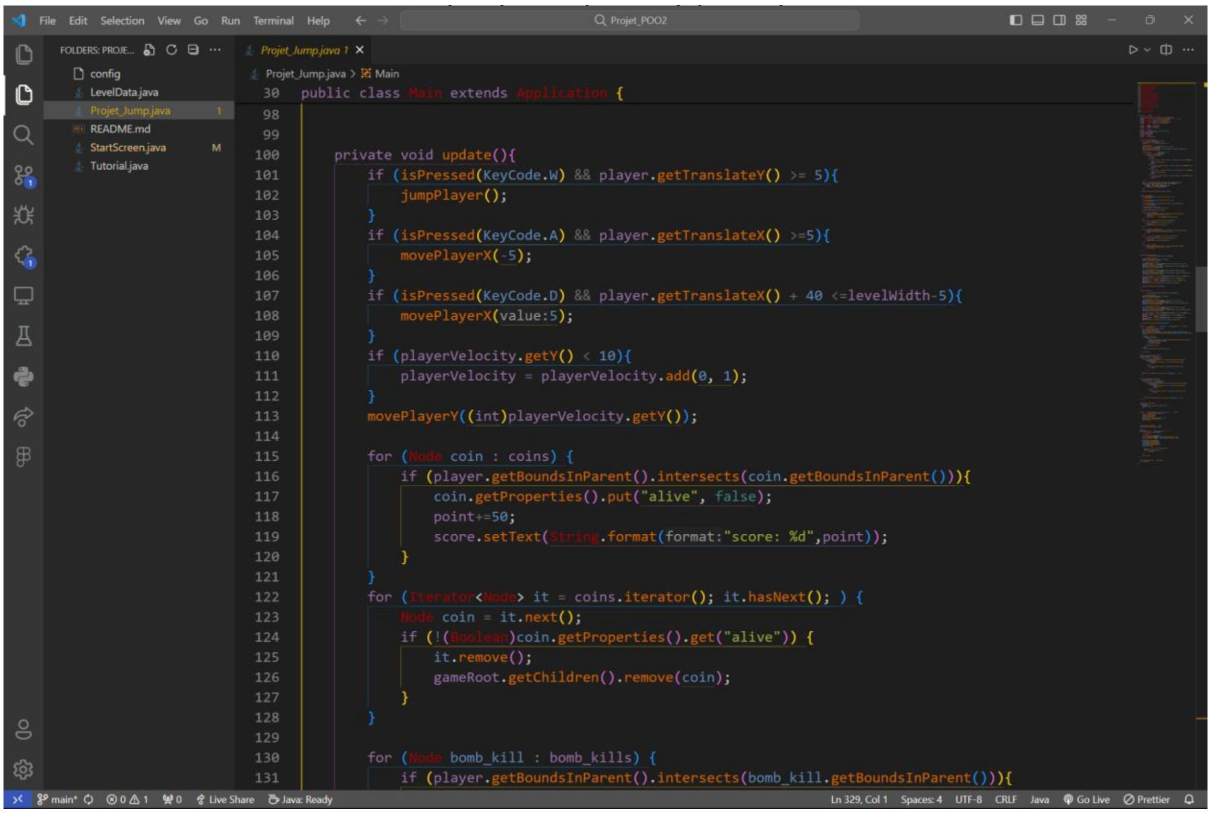
Players take control of a character in this game as they move through a variety of levels that are filled with doors, platforms, coins, and bombs. The primary goal is to lead the player's character from the beginning of each level to the door at the finish while increasing their score. With the help of the game's simple controls, players can jump with the 'W' key and move left and right using ‘A’ and ‘D’ respectively.

A grid-based design is used to build the game world, and various platform arrangements are used to provide difficult terrain for the player to navigate. The rectangular nodes that represent each platform are used by the player character to move both vertically and horizontally over them. The player character interacts with other game elements including doors, bombs, and coins. The player character is represented as a coloured rectangle.

Throughout the game, coins are placed as collectibles that earn points for the player when they are found. The user interface shows the player's score, which is updated in real time as the player gathers coins. Bombs are dangerous for the player character; if the player collides with one, the game is lost. Collision detection is used in the game to precisely manage interactions between the player character and other game elements.

A key component of the logic of the game is handling events, which permits responsive gameplay and processing of user input. To find out when particular keyboard keys are pushed or released, the code makes use of event listeners. For instance, the player's character can jump by hitting the 'W' key, giving them the ability to move vertically. In a similar manner, the player character can be moved horizontally over the platforms by using the left or right arrow keys.

Furthermore, event handling is used in the game to control how the gameplay unfolds. The player moves on to the following level after gathering all the coins and arriving at the door at the conclusion of the level. On the other hand, hitting a bomb causes a game over event, which shows the player's score at the end of the game along with a choice to either restart it or go back to the main menu.



```java
public class Main extends Application {


    private void update(){
        if (isPressed(KeyCode.W) && player.getTranslateY() >= 5){
            jumpPlayer();
        }
        if (isPressed(KeyCode.A) && player.getTranslateX() >=5){
            movePlayerX(-5);
        }
        if (isPressed(KeyCode.D) && player.getTranslateX() + 40 <=levelWidth-5){
            movePlayerX(value:5);
        }
        if (playerVelocity.getY() < 10){
            playerVelocity = playerVelocity.add(0, 1);
        }
        movePlayerY((int)playerVelocity.getY());

        for (Node coin : coins) {
            if (player.getBoundsInParent().intersects(coin.getBoundsInParent())){
                coin.getProperties().put("alive", false);
                point+=50;
                score.setText(String.format(format:"score: %d",point));
            }
        }
        for (Iterator<Node> it = coins.iterator(); it.hasNext(); ) {
            Node coin = it.next();
            if (!(Boolean)coin.getProperties().get("alive")) {
                it.remove();
                gameRoot.getChildren().remove(coin);
            }
        }

        for (Node bomb_kill : bomb_kills) {
            if (player.getBoundsInParent().intersects(bomb_kill.getBoundsInParent())){
```

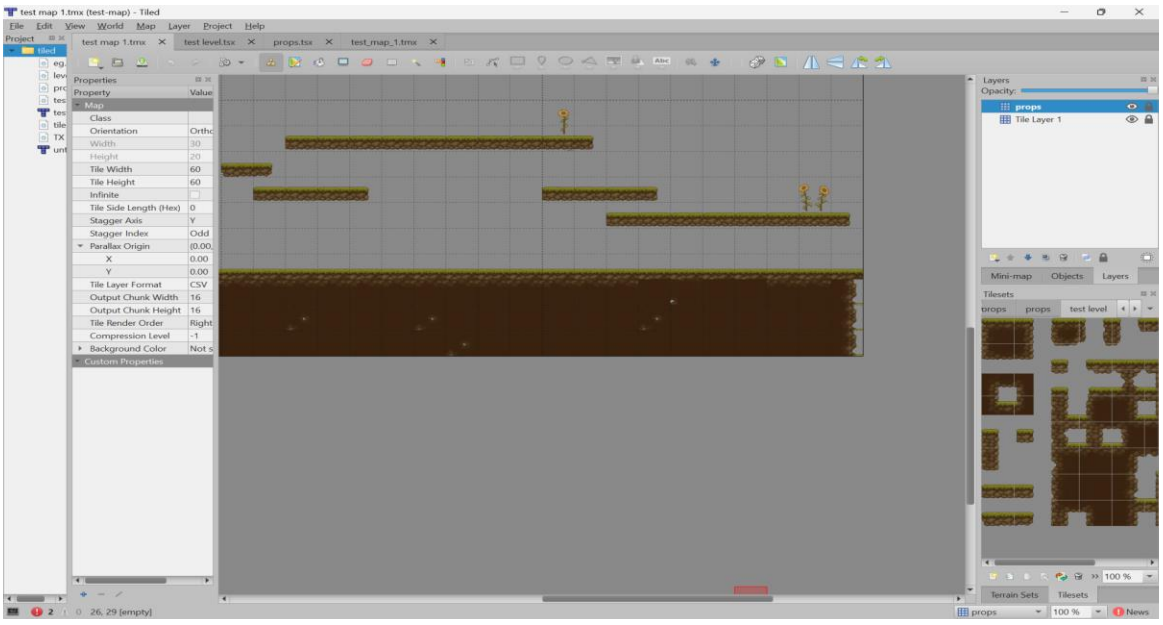
Figure 7

Problems

One popular method for improving level design is to use Tiled and a tileset to create a base layout for a 2D platform game. However, because the underlying technologies differ, there can be difficulties integrating this into a JavaFX application. The main function of JavaFX is to render graphical user interfaces (GUIs); Tiled's Lua-based tile layer format is not natively supported by JavaFX.

Therefore, building a direct interface between Tiled's Lua file and JavaFX would require a substantial amount of proprietary parsing and rendering code. Moreover, Tiled's tile-based maps are not natively supported by JavaFX, which makes it challenging to effectively translate Tiled maps into the JavaFX scene graph. Because of this, developers frequently turn to workarounds like manually exporting Tiled maps to a format that JavaFX can readily parse or utilising game development tools, which provide superior support for Tiled maps and tilesets.

The picture below is the original tileset i used to create a game platform using Tiled but was not able to implement this due to the file being in Lua format which is incompatible when working with JavaFx.



Property	Value
Map	
Class	
Orientation	Ortho
Width	30
Height	20
Tile Width	60
Tile Height	60
Infinite	☐
Tile Side Length (Hex)	0
Stagger Axis	Y
Stagger Index	Odd
Parallax Origin	(0.00,
X	0.00
Y	0.00
Tile Layer Format	CSV
Output Chunk Width	16
Output Chunk Height	16
Tile Render Order	Right
Compression Level	-1
Background Color	Not s
Custom Properties	

Figure 8

Conclusion

The dynamic platformer game "Jump" provides players with an engaging and fun experience. Through levels full of different challenges and obstacles, players may go through them with ease thanks to the intuitive gameplay mechanics and fluid controls. The protagonist of the game is able to walk left and right, jump, and interact with various objects like doors, coins, platforms, and bombs. The goal is to dodge obstacles like red cubes that can terminate the game and instead gather coins that are scattered around the levels to raise the player's score.

Annex

Link GitHub

This link below is our GitHub link for the entirety of the code.
https://github.com/NORBS06/Projet_POO2.git

Demo video

This link below is a video, a demo of the actual game.
final demo.mp4