Snake

Projekt na programowanie sem. III kierunek informatyka

Adam Czerwiński

Spis treści

Wstęp

Wstęp

Projekt jest oparty na grze z roku 1976, wydaną pod nazwą Blockade oraz udoskonalony o własne pomysły. Program został napisany w języku JAVA w wersji jdk 11.0.1 wraz z wykorzystaniem biblioteki JavaFX, która jest używana do tworzenia GUI. Korzystano również z narzędzia Scene Builder do szybkiego stworzenia układu wizualnego. Użyto środowiska IDE IntelliJ IDEA.

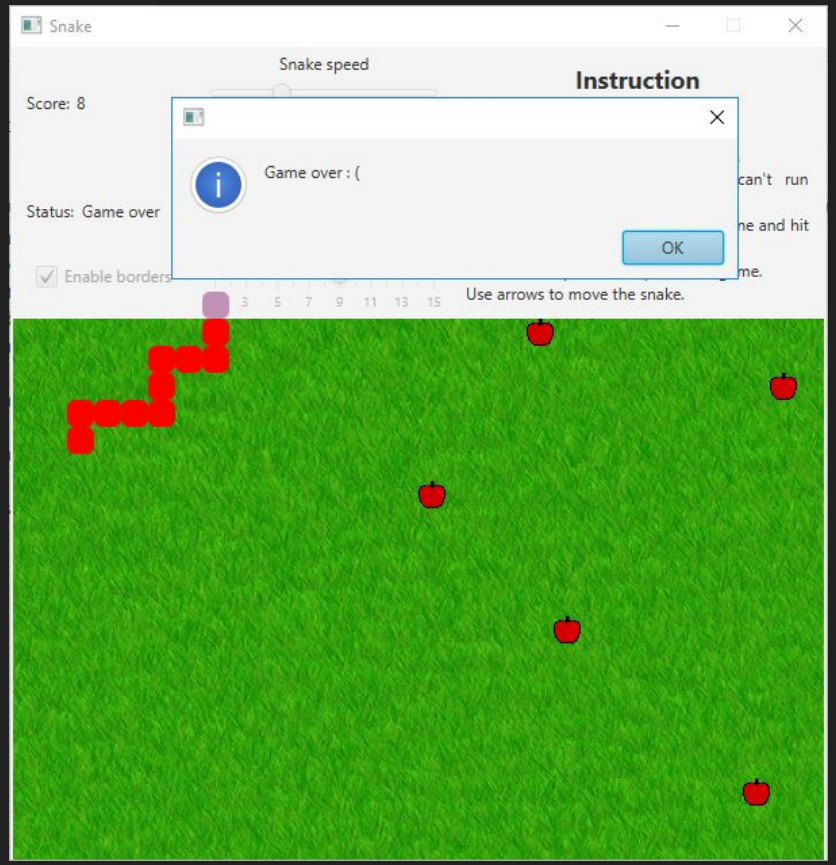
Działanie gry

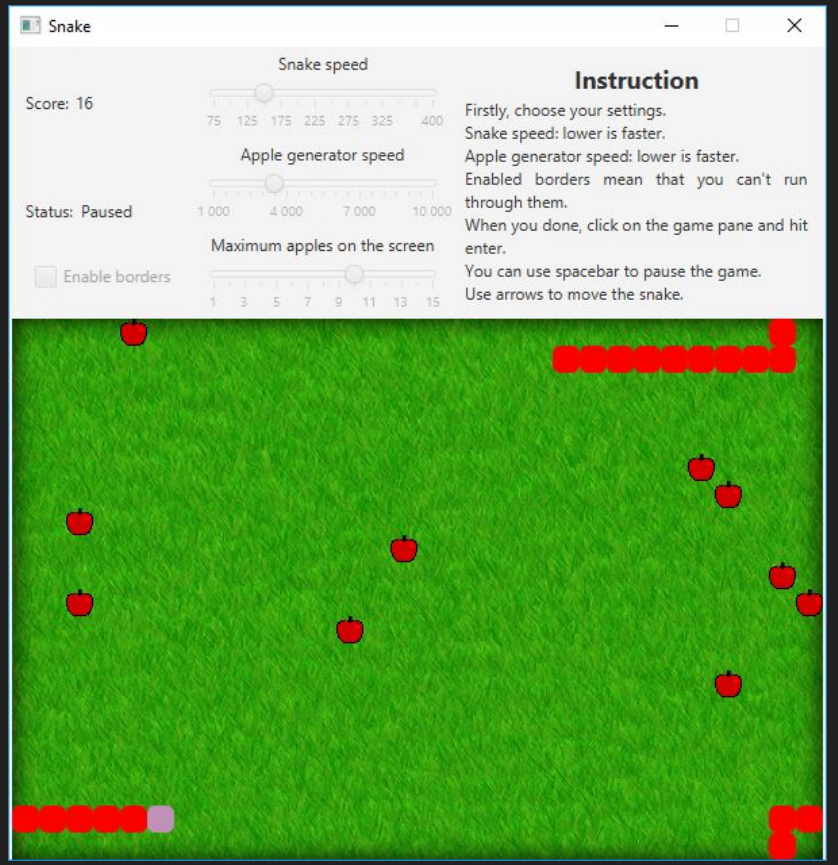
Działanie gry

Gra polega na jak największej ilości zebranych punktów. Punkty otrzymujemy za zjedzenie jabłek przez węża, wtedy również wąż rośnie. Koniec gry następuje po ugryzieniu samego siebie albo po uderzeniu w krawędzie (w zależności od opcji).

Do poruszania się wężem używamy strzałek. Jest możliwość wstrzymania gry za pomocą klawisza spacji lub przerwanie za pomocą klawisza escape.

Kolizja z krawędziami





Kod źródłowy

Main loop

Metoda GameLoop odpowiada za aktualizowanie informacji o obiektach co określony czas. Wykorzystano dwie pętle, jedna do aktualizowania informacji o wężu, a druga do aktualizowania informacji o jabłkach, ponieważ mogą mieć różne wartości czasu.

```
private void GameLoop() {
    //--------------GAME LOOP
    gameLoop = new Timeline(
            new KeyFrame(
                    Duration.millis(snakePace.getValue()),
                    event -> {
                        if (Settings.inGame && !Settings.isPause) {
                            Settings.isPressed = false;
                            if (snake.get(0).getDirection() == Directions.UP) {
                                moveUp();
                            } else if (snake.get(0).getDirection() == Directions.RIGHT) {
                                moveRight();
                            } else if (snake.get(0).getDirection() == Directions.DOWN) {
                                moveDown();
                            } else if (snake.get(0).getDirection() == Directions.LEFT) {
                                moveLeft();
                            }
                            moveRest();
                            update();
                            updateStatus();
                        }
                    }
            )
    );//////END OF GAME LOOP
    gameLoop.setCycleCount(Animation.INDEFINITE);
    gameLoop.play();

    appleGenerator = new Timeline(
            new KeyFrame(
                    Duration.millis(appleGeneratorPace.getValue()),
                    event -> {
                        if (Settings.inGame && !Settings.isPause && apples.size() < maximumApplesOnScreen.getValue()) {
                            generateNewApple();
                        }
                    }
            )
```

Poruszanie wężem

Metody moveUp, moveRight, moveDown, moveLeft służą do zmiany kierunku głowy węża, a metoda moveRest służy do poruszania reszty ciała węża. Poruszanie węża polega na przypisaniu obiektowi wartości z obiektu go poprzedzającego, które te są przechowywane w pomocniczej tablicy obiektów klasy Point2D.

```java
private void moveRight() {
    saveLastPositionSnake();
    snake.get(0).setX(snake.get(0).getX() + 1);
    snake.get(0).setRectangleX();
}

private void moveUp() {
    saveLastPositionSnake();
    snake.get(0).setY(snake.get(0).getY() - 1);
    snake.get(0).setRectangleY();
}

private void saveLastPositionSnake() {
    snakeCoordinates.get(0).setX(snake.get(0).getX());
    snakeCoordinates.get(0).setY(snake.get(0).getY());
}

private void moveRest() {
    for (int i = 1; i < snake.size(); i++)
    {
        snakeCoordinates.get(i).setX(snake.get(i).getX());
        snakeCoordinates.get(i).setY(snake.get(i).getY());
        snake.get(i).setX(snakeCoordinates.get(i - 1).getX());
        snake.get(i).setY(snakeCoordinates.get(i - 1).getY());
        snake.get(i).setRectangleX();
        snake.get(i).setRectangleY();
    }
}
```

Sprawdzanie kolizji

Metoda update() zawiera między innymi sprawdzanie kolizji. Sprawdzana jest kolizja z krawędziami - w zależności od zaznaczonej opcji, kolizja z samym sobą (wężem) oraz kolizja z jabłkami. Jeżeli wystąpi kolizja tworzony jest alert informujący gracza o przegranej grze, a następnie zostaje załadowane menu główne gry.

```
private void update() {
    if (collisionSnake() || (collisionBorder() && isBorders.get())) {
        quitGame();
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle(null);
        alert.setHeaderText(null);
        alert.setContentText("Game over : (");

        alert.setOnHidden(evt -> mainController.loadMenu());

        alert.show();
    }

    if (collisionBorder() && !isBorders.get()) {
        outOfBorders();
    }

    int index = collisionApple();
    if (index >= 0) {
        increaseScore();
        addSnake();
        generateAppleCoordinates(index);
    }
}
```

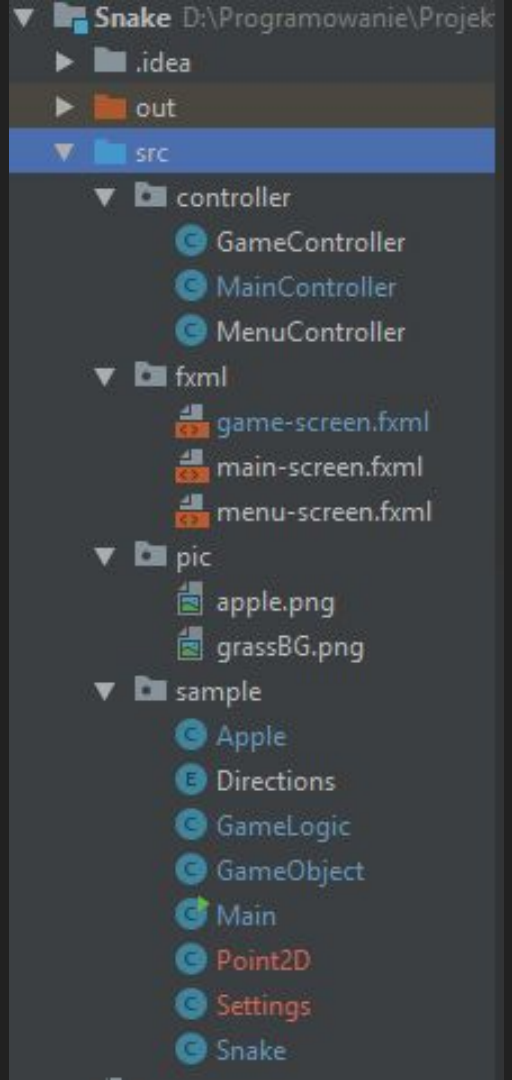
Podział na pliki

Podział na pliki

JavaFX korzysta z plików o rozszerzeniu FXML, bazowane na języku XML. Służą one do odseparowania struktury logicznej ze strukturą wizualną programu. Do modyfikacji wyglądu m.in. kontrolek można wykorzystać język CSS, z którego w tym projekcie nie korzystałem.
Pliki FXML są tworzone automatycznie przy użyciu programu Scene Builder, nie są one obowiązkowe w programie i można osiągnąć taki sam efekt wyłączenie za pomocą samych klas.

Struktura

- Pakiet o nazwie controller zawiera w sobie klasy, które są powiązane z plikami FXML i mają bezpośredni dostęp do kontrolek stworzonych w odpowiadających im plikach FXML.
- Pakiet fxml zawiera pliki fxml.
- Pakiet pic zawiera pliki z rozszerzeniem .png.
- Pakiet sample zawiera klasy odpowiadające za logikę programu.



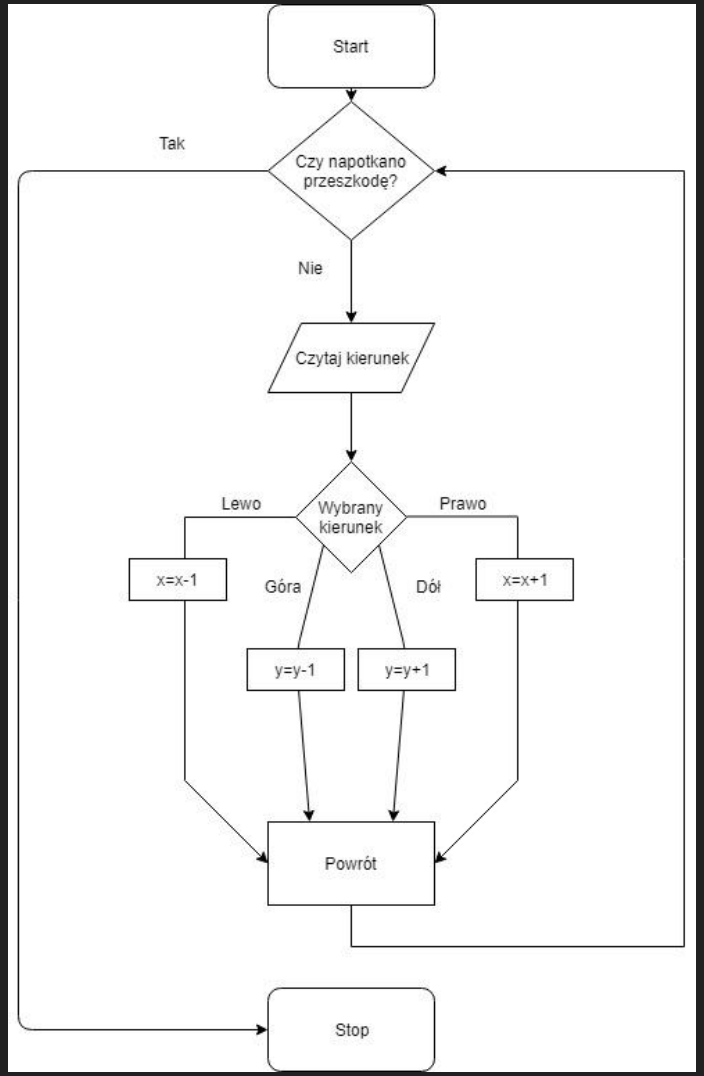
Przykładowy plik FXML

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.text.*?>
<?import javafx.scene.effect.*?>
<?import javafx.geometry.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<Pane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="600.0"
      prefWidth="600.0" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controller.MenuController">
   <children>
      <VBox alignment="CENTER" layoutX="235.0" layoutY="240.0" spacing="40.0">
         <children>
            <Button mnemonicParsing="false" onAction="#onActionPlayButton" prefHeight="50.0" prefWidth="130.0" text="Play Game">
               <font>
                  <Font size="20.0" />
               </font>
            </Button>
            <Button mnemonicParsing="false" onAction="#onActionExitButton" prefHeight="30.0" prefWidth="90.0" text="Exit">
               <font>
                  <Font size="14.0" />
               </font>
            </Button>
         </children>
      </VBox>
   </children>
</Pane>
```

Schemat blokowy


Start
Czy napotkano przeszkodę?
Tak
Nie
Czytaj kierunek
Wybrany kierunek
Lewo
Prawo
Góra
Dół
x=x-1
x=x+1
y=y-1
y=y+1
Powrót
Stop