Specyfikacja Funkcjonalna

Rozdział 1: Opis Ogólny

Podrozdział 1.1: Nazwa programu.
Nazwa programu : 'Arkanoid'.

Podrozdział 1.2: Poruszany problem.
Program jest przykładem implementacji gry typu Arkanoid, której celem jest zniszczenie wszystkich bloków dostępnych na mapie za pomocą piłki, która jest sterowana za pomocą paletki.

Podrozdział 1.3: Użytkownik docelowy.
Program jest kierowany do pasjonatów gier komputerowych, w szczególności z czasów początków rynku gier.

Rozdział 2: Opis Funkcjonalności

Podrozdział 2.1: Jak korzystać z programu?
Do poprawnego funkcjonowania programu potrzebna jest działająca mysz komputerowa oraz opcjonalnie klawiatura.

Podrozdział 2.2: Uruchomienie programu.
W celu uruchomienia programu należy wywołać plik wykonywalny Arkanoid.jar a następnie za pomocą myszy wybierać odpowiednie opcje z menu.

Podrozdział 2.3: Możliwości programu.

1. Sterowanie odbywa się z wykorzystaniem głównie myszy komputerowej z opcjonalnym użyciem klawiatury.
2. Program pozwala wybrać jedną z 10 przygotowanych map gry, ale jednocześnie pliki zawierające dane map są w pełni edytowalne.
3. Za pomocą klawisza klawiatury przebieg gry może zostać zatrzymany i wznowiony.
4. W trakcie gry pojawiają się bonusy, których zebranie powoduje dynamiczną zmianę trudności gry poprzez zmianę parametrów gry.

Rozdział 3: Format danych i struktura plików

Podrozdział 3.1: Pojęcia i pola formularza
paletka – plansza na której umieszczane są 'żywe' komórki podane w pliku przy wywołaniu.
list – lista 'żywych' punktów.
simulator – struktura przechowująca dane do prowadzenia symulacji.

Podrozdział 3.1: Struktura pakietów

pakiet 'main' :

- Ball.java
- Brick.java
- CollisionManager.java
- GameScreenController.java
- Main.java
- MainMenuController.java
- MapChooserController.java

- Paddle.java

pakiet ‘main.Templates’ :

- GameScreen.fxml
- MainMenu.fxml
- MapChooser.fxml
- arkanoid_styles.css
- grid1.fxml
- grid2.fxml
- grid3.fxml
- grid4.fxml
- grid5.fxml
- grid6.fxml
- grid7.fxml
- grid8.fxml
- grid9.fxml
- gridX.fxml
- grid1.png
- grid2.png
- grid3.png
- grid4.png
- grid5.png
- grid6.png
- grid7.png
- grid8.png
- grid9.png
- gridX.png
- frame.png

Podrozdział 3.2: Dane wejściowe
Program jako dane wejściowe przyjmuje pliki o rozszerzeniu .fxml, z których są ładowane odpowiednie sceny okna, plik arkanoid_styles.css zawierające style do tych okien oraz zdjęcia typu .png, które są wyświetlane w odpowiednich miejscach w oknie.

Podrozdział 3.3: Funkcje wewnętrzne

Klasy w programie:

- ”Ball” - klasa zawierająca atrybuty i metody piłki
- ”Brick” - klasa zawierająca atrybuty i metody klocka

- "Paddle" - klasa zawierająca atrybuty i metody paletki odbijającej piłkę
- "Bonus" - klasa zawierająca atrybuty i metody bonusu
- "MapChooserController" - klasa kontrolera ekranu wyboru mapy
- "CollisionManager"- klasa odpowiedzialna za sprawdzanie kolizji i wyliczanie kąta po odbiciu się piłki od klocka
- "GameScreenController" - klasa kontrolera ekranu gry
- "MainMenuController" - klasa kontrolera menu głównego
- "Main" - klasa zawierająca inicjację okna

Metody klasy Ball:

- public Ball(Circle circle, double velocityX, double velocityY); - konstruktor klasy przyjmujący jako parametry okrąg będący reprezentacją piłki w pliku *.fxml oraz składową poziomą i pionową prędkości.
- public void setX(double x);
  - ustawia nową współrzędną poziomą piłki
- public void setY(double y);
  - ustawia nową współrzędną pionową piłki
- public double getX();
  - metoda dostępowa, zwraca aktualną wartość współrzędnej poziomej piłki
- public double getY();
  - metoda dostępowa, zwraca aktualną wartość współrzędnej pionowej piłki
- public double getRadius();
  - metoda dostępowa, zwraca stałą wartość promienia piłki
- public double getVelocityX();
  - metoda dostępowa, zwraca aktualną wartość składowej poziomej prędkości piłki
- public double getVelocityY();
  - metoda dostępowa, zwraca aktualną wartość składowej pionowej prędkości piłki
- public void setVelocityX(double value);
  - ustawia nową wartość składowej poziomej prędkości
- public void setVelocityY(double value);
  - ustawia nową wartość składowej pionowej prędkości
- public double getNextX();
  - wylicza nową wartość współrzędnej poziomej piłki
- public double getNextY();
  - wylicza nową wartość współrzędnej pionowej piłki
- public Circle getCircle();
  - metoda dostępowa, zwraca okrąg, będący prywatnym atrybutem klasy 'Ball'
- public Circle getNextCircle();
  - zwraca okrąg o nowych współrzędnych, służy on jedynie do wyliczenia kolizji piłki z klockiem
- public void makeBounceFromVertical();
  - zmienia zwrot składowej pionowej prędkości
- public void makeBounceFromHorizontal();
  - zmienia zwrot składowej poziomej predkości

Metody klasy Bonus:

- public Bonus(Type type, double x, double y, AnchorPane field);
  - konstruktor klasy przyjmujący jako parametry typ bonusu, jego współrzędne na ekranie i kontener do którego jest przypisany
- public Rectangle getRectangle();
  - metoda dostępowa, zwraca prostokąt, w którym znajduje się bonus
- public Bounds getBounds();
  - metoda dostępowa, zwraca obiekt zawierające informacje o pozycji i wielkości bonusu na ekranie
- public Type getType();
  - metoda dostępowa, zwraca typ bonusu

  public double getVelocityY();
  - metoda dostępowa, zwraca długość składowej pionowej prędkości
- public void setY(double y);
  - ustawia nową współrzędną pionową bonusu
- public double getX();
  - metoda dostępowa, zwraca aktualną wartość współrzędnej poziomej bonusu
- public double getY();
  - metoda dostępowa, zwraca aktualną wartość współrzędnej pionowej bonusu
- public double getNextY();
  - wylicza nową wartość współrzędnej pionowej bonusu
- public double getWidth();
  - metoda dostępowa, zwraca szerokość bonusu na ekranie
- public double getHeight();
  - metoda dostępowa, zwraca wysokość bonusu na ekranie
- public void removeFromFiedl(AnchorPane field);
  - usuwa bonus z kontenera na ekranie

Metody klasy Brick:

- public Brick(Rectangle rectangle, int initalHealthPoints);
  - konstruktor klasy przyjmujący jako parametr prostokąt będący reprezentacją klocka w pliku *.fxml oraz ilość odbić po których dany klocek zostanie zniszczony
- public Rectangle getRectangle();
  - metoda dostępowa, zwraca prostokąt będący odpowiednikiem klocka
- public int getHealthPoints();
  - metoda dostępowa, zwraca aktualną liczbę dozwolonych trafień piłki w klocek, po których nie zostanie on zniszczony
- public void decreaseHealthPoints();
  - zmniejsza liczbę dozwolonych trafień w klocek o 1
- public void setVisible(boolean value);
  - ustawia parametr odpowiadający za widoczność klocka w okienku aplikacji
- public boolean isVisible();
  - sprawdza czy zadany klocek znajduje się jeszcze na planszy i jest możliwy do zniszczenia
- public double getX();
  - metoda dostępowa, zwraca współrzędną poziomą zadanego klocka

- public double getY();
  - metoda dostępowa, zwraca współrzędną pionową zadanego klocka
- public double getWidth();
  - metoda dostępowa, zwraca szerokość zadanego klocka
- public double getHeight();
  -metoda dostępowa, zwraca wysokość zadanego klocka

Metody klasy 'CollisionManager':

- public CollisionManager(Ball ball, Paddle paddle, AnchorPane field);
  - konstruktor klasy przyjmujący jako parametry obiekt klasy 'Ball', 'Paddle' oraz 'AnchorPane'. Parametry odpowiadają za monitorowaną piłkę, paletkę oraz okienko gry.
- public void checkForCollisions(List <Brick>bricks);
  - sprawdza czy w następnym położeniu piłki nastąpi zderzenie z klockami z listy
- public boolean manageCollision(Rectangle rect, Circle circle);
  - zwraca wartość true jeśli nastąpiła kolizja, w przeciwnym wypadku zwraca false
- private void makeCornerBounce(Rectangle rect, Circle circle);
  - zarządza zmianą składowych prędkości po odbiciu od rogu prostokąta
- private void calculateCornerBounce(double cornerX, double cornerY);
  - wylicza kąt i ustawia nowe wartości składowych prędkości piłki przy odbiciu od kąta
- private CollisionType checkRectangleCircleIntersection(Rectangle rect, Circle circle);
  - zwraca typ kolizji pomiędzy kołem i prostokątem
- private boolean willBallCollideWithWall();
  - sprawdza czy piłka zderzyła się z pionową ścianą
- private boolean willBallCollideWithTop();
  - sprawdza czy piłka zderzyła się z górną ścianą

Metody klasy 'GameScreenController':

- public void handleSceneClick();
  - metoda zarządza rozpoczęciem gry, oczekując na kliknięcie myszy
- private void Loop();
  - zarządza zachowanie gry, z zależności od jej stanu
- private GameState getGameState();
  - zwraca aktualny stan gry
- public void initialize(URL url, ResourceBundle rb);
  - inicjuje mechanizmy działania aplikacji

  Metody klasy 'MapChooserController':
- private void handlePlayButton(ActionEvent event);
  - ustawia wybraną planszę na ekran i umożliwia rozpoczęcie rozgrywki
- private void handleMapButtons(ActionEvent event);
  - przechwytuje odpowiedni wzór planszy i oznacza ją jako wybraną
- public void initialize(URL url, ResourceBundle rb);
  - ustawia wybraną planszę

Metody klasy 'Paddle' :

- public Paddle(Rectangle rectangle);
  - konstruktor klasy przyjmujący jako parametr obiekt klasy Rectangle
- public Rectangle getRectangle();
  - metoda dostępowa, zwraca obiekt klasy Rectangle
- public double getX();
  - metoda dostępowa, zwraca współrzędną poziomą obiektu
- public double getY();
  - metoda dostępowa, zwraca współrzędną pionową obiektu
- public void setX(double x);
  - zmienia wartość współrzędnej poziomej na wartość przyjętą jako parametr
- public double getWidth();
  - metoda dostępowa, zwraca szerokość obiektu
- public double getHeight();
  - metoda dostępowa, zwraca wysokość obiektu

Rozdział 4: Scenariusz pracy z programem
1) Uruchomienie programu
Aby uruchomić program należy wywołać go w systemie operacyjnym zawierającym oprogramowanie Java.

2) Menu Główne
Użytkownik w menu głównym może wybrać opcję "PLAY", która przenosi go do menu wyboru mapy lub "EXIT", która spowoduje wyłączenie programu.

3) Menu Wyboru Mapy
W tym menu użytkownik może zaznaczyć wedle upodobań mapę, na której chciałby zagrać, a następnie potwierdzić wybór przyciskiem "Play".

4) Ekran Gry
Na ekranie gry znajdują się następujące elementy:

- górny pasek, który zawiera informacje na temat pozostałych żyć oraz czas od rozpoczęcia gry,
- kolorowe prostokątne bloki, których zniszczenie jest celem użytkownika, w zależności od ilości uderzeń potrzebnych do zniszczenia przyjmują inny kolor,
- piłka, której kolizja z blokiem powoduje zmniejszenie liczby uderzeń potrzebnych do zniszczenia bloui,
- paletka, której pozycja jest ustawiana na podstawie pozycji kursora, służy do odbijania piłki w wybranym kierunku i zapobiegania opuszczenia boiska przez piłkę oraz zbierania bonusów,
- bonusy, które losowo pojawiają się po zniszczeniu bloku, powodują zmianę parametrów gry: wielkość paletki, ilość piłek, wielkość piłki, prędkość piłki.

Na początku gry piłka jest zaczepiona do paletki. Kliknięcie lewym przyciskiem myszy na okno powoduje wystrzelenie piłki pod losowym kątem. Od tego momentu następuje odliczanie czasu gry i kliknięcie klawisza "P" na klawiaturze powoduje zatrzymanie gry. Piłka przemieszcza się ze stałą prędkością, odbijając się od ścan bocznych i sufitu oraz bloków, co powoduje ich zniszczenie lub zmniejszenie liczby uderzeń pozostałych do zniszczenia. Przy zniszczeniu bloku istnieje 10% szansy na pojawienie się losowego typu bonusu, które opadają i mogą zostać zebrane za pomocą paletki. Typy bonusów to: zwiększenie/zmniejszenie piłki, zwiększenie/zmniejszenie paletki, zwiększenie/zmniejszenie prędkości piłki, pojawienie się dodatkowych dwóch piłek z każdej istniejącej.
Zadaniem użytkownika jest zniszczenie wszystkich obecnych bloków, co powoduje wygraną i przeniesienie do Menu Głównego. W trakcie gry użytkownik posiada określoną liczbę żyć, która wynosi 3 na początku gry. Jeśli wszyskie piłki opuszczą boisko poprzez dolną ścianę okna użytkownik traci jedno z żyć i pojawia się nowa piłka przyczepiona do paletki, którą użytkownik może ponownie wystrzelić. Jeśli liczba żyć osiągnie 0 użytkownik przegrywa grę i jest przenoszony do Menu Głównego.