Изображение выглядит как символ, эмблема, логотип, мультфильм

Автоматически созданное описаниеФГБОУ ВО Уральский государственный горный университет

Инженерно-экономический факультет

Кафедра информатики

**Курсовой проект**

По дисциплине программирование

На тему «Разработка приложения «Ballz»

|  |  |
| --- | --- |
|  | Выполнил:  Студент гр. АУБП-22-2  Писковский И. В. |
|  | Проверила: ст. преп. каф. информатики  Волкова Е. А. |

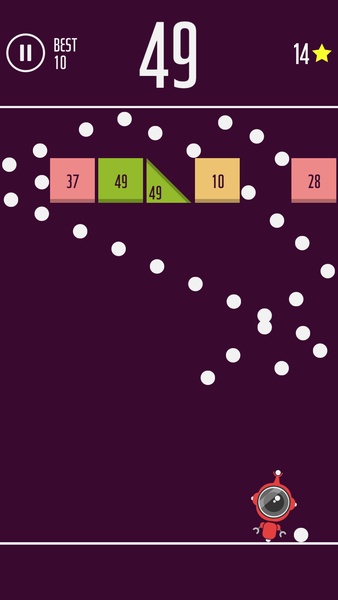
Екатеринбург, 2024

**Содержание**

1. Постановка задачи………………………………………………3
   1. Формулировка задания……………………………………3
   2. Алгоритмическое решение задачи……………………….4
   3. Контрольные примеры……………………………………7
2. Решение задачи………………………………………………...13
   1. Выбор средств реализации……………………………...13
   2. Описание основных классов……………………………14
   3. Интерфейс приложения…………………………………18
3. Тестирование приложения……………………………………
   1. Контрольный робот……………………………………..
   2. Работа приложения на контрольных тестах…………..
   3. Результаты работы………………………………………
4. **Постановка задачи**
   1. **Формулировка задания**

«Ballz»

В этой игре вам предстоит, используя пушку, стреляющую шариками, разбивать блоки и зарабатывать очки. Блоки имеют разную прочность, которая уменьшается от ударов шариком.



**Постановка задачи**

Реализовать алгоритм игры «Ballz»

* 1. **Алгоритмическое решение задачи**

Для решения задачи по разработке игры с управлением шаром (Ball), кирпичами (Brick) и игроком (Player) или ИИ (AIPlayer) используются разветвляющий и циклический алгоритмы. Алгоритм включает этапы: инициализацию игры, управление шаром и кирпичами, а также действия игрока и ИИ.

На этапе инициализации создается окно и интерфейс. Создается объект класса Stage для главного меню, где размещены кнопки для выбора режима игры (игрок или ИИ) и текстовые элементы для отображения названия игры. Устанавливаются обработчики событий для кнопок, которые запускают игровой процесс при нажатии.

После выбора режима игры вызывается метод startGame, в который передается режим игры (человек или ИИ, рис. 1).

Создание интерфейса меню

new GUI(menuStage);

Выбор режима игры

ИИ или игрок

startGame(false);

startGame(true);

Рис. 1 Выбор режима игры

При запуске уровня инициализируется игровое поле: создается объект класса Pane для игрового поля и объект класса Scene для его отображения. На игровом поле размещаются объекты класса Ball и Brick. Обрабатываются нажатия клавиш для управления углом запуска шара и его стрельбы.

Основной игровой цикл начинается с создания и запуска объекта AnimationTimer, который обновляет состояние игры в каждом кадре. В каждом кадре обновляется положение шара, проверяются столкновения шара с границами игрового поля и кирпичами, обновляется состояние кирпичей (уменьшается здоровье при попадании и удаляются, если здоровье равно нулю). При уничтожении всех кирпичей происходит переход на следующий уровень (рис. 2).

Обновление положения шариков

ball.update();

Проверка столкновений

if (ball intersects brick)

ball.reverseX() / ball.reverseY()

Обновление здоровья блоков

Brick.hit();

If (brick.getHitPoints() <= 0)

remove brick from gamePane

Все блоки уничтожены?

Переход на следующий уровень

startLevel(level + 1);

Рис. 2 Логика перехода на следующий уровень

Управление игрока включает обработку нажатий клавиш для управления углом запуска шара и его стрельбы, а также запуск шара при нажатии на пробел (рис. 3). Для управления ИИ реализуется логика выбора случайного угла и стрельбы с помощью метода shouldShoot, определяющего момент стрельбы (рис. 4).

Рис. 3 Логика выстрела игрока

Выстрел шаров

ball.shoot(launchAngle);

Пробел нажат?

If (event.getCode() == KeyCode.SPACE)

Обработка нажатий клавиш

gameScene.setOnKeyPressed(event);

* 1. **Контрольные примеры**

Для проверки работы игры приведем несколько контрольных примеров, отражающих управление шарами и обработку здоровья блоков при попадании в них шаров.

Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение

Автоматически созданное описаниеЗапустим приложение и выберем режим игры «Игрок». Ставим угол выстрела 90 градусов (вертикально вверх, рис. 5).

Рис. 5 Выбор вертикального угла

Шар летит вертикально вверх, как и ожидалось (рис. 6)

Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение

Автоматически созданное описаниеРис. 6 Полет шара вертикально вверх

Теперь выберем угол 45 градусов (рис. 7).

Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

Рис. 7 Выбор угла 45 градусов

Шар летит под углом 45 градусов, как и должно быть (рис. 8).

Изображение выглядит как снимок экрана, текст, луна

Автоматически созданное описание

Рис. 8 Полет шара под углом 45 градусов

Теперь пример того, что здоровье блоков уменьшается от попаданий шаров (рис. 9, рис. 10).

Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

Рис. 9 Изначальное состояние блоков

Изображение выглядит как снимок экрана, текст

Автоматически созданное описаниеРис. 10 Конечное состояние блоков (здоровье блока уменьшилось)

1. **Решение задачи**
   1. **Выбор средств реализации**

Для решения данной задачи был выбран высокоуровневый язык программирования Java. Java предлагает множество встроенных полезных функций и коллекций, что значительно упрощает работу с программой. Хотя можно было бы использовать C++ для решения этой задачи, Java предоставляет некоторые ключевые преимущества.

Java является объектно-ориентированным языком программирования, изначально разработанным компанией Sun Microsystems, которая позже была приобретена Oracle. Java-приложения обычно компилируются в байт-код, который может выполняться на любой виртуальной Java-машине (JVM), независимо от архитектуры компьютера. Этот байт-код исполняется JVM, которая действует как интерпретатор, передающий инструкции оборудованию. Это обеспечивает полную независимость байт-кода от операционной системы и аппаратного обеспечения, что позволяет запускать Java-приложения на любом устройстве, поддерживающем JVM.

Java также известна своей гибкой системой безопасности, благодаря которой выполнение программы полностью контролируется виртуальной машиной. Это обеспечивает немедленное прерывание любых операций, превышающих установленные полномочия программы, таких как попытки несанкционированного доступа к данным или подключения к другому компьютеру.

Для разработки был выбран Visual Studio Code. Хоть Visual Studio Code (далее VS Code) позиционируется как редактор кода, он может использоваться как мощная среда разработки. VS Code поддерживает рефакторинг, профилирование, подсветку синтаксиса, автодополнение кода, множество шаблонов и другие полезные функции.

* 1. **Описание основных классов**

Для решения задачи были разработаны 7 классов, которые показаны на диаграмме (рис. 11).

Изображение выглядит как текст, диаграмма, Параллельный, План

Автоматически созданное описание

Рис. 11 Диаграмма классов

Класс «App» запускает приложение и инициализирует меню приложения из класса «GUI». Класс «GUI» в свою очередь создает окно 500\*600 пикселей, и размещает две кнопки, с помощью которых выбирается режим игры.

Листинг класса «GUI» представлен в п. 2.3 «Интерфейс приложения».

**public** **class** **App** **extends** Application {

**@Override**

**public** **void** **start**(Stage menuStage) {

GUI gui = **new** GUI(menuStage);

}

**public** **static** **void** **main**(String[] args) {

launch(args);

}

}

Листинг 1. Класс «App»

Класс «GameBoard» реализует основную логику приложения.

Листинги 2, 3 и 4 содержат алгоритмы, описанные в п. 1.2 «Алгоритмическое решение задачи».

**public** **GUI**(Stage menuStage) {

Text nameLabel = **new** Text(LABEL\_GAME);

nameLabel.setFont(**new** Font("Segoe Print", **28**));

nameLabel.setFill(Color.WHITE);

Button playAI = **new** Button("ИИ");

Button playPlayer = **new** Button("Игрок");

playAI.setPrefWidth(**200**);

playAI.setPrefHeight(**50**);

playAI.setFont(**new** Font("Calibri", **16**));

playPlayer.setPrefWidth(**200**);

playPlayer.setPrefHeight(**50**);

playPlayer.setFont(**new** Font("Calibri", **16**));

playAI.setOnAction(e -> startGame(menuStage, **true**));

playPlayer.setOnAction(e -> startGame(menuStage, **false**));

VBox nameLabelBox = **new** VBox(**10**, nameLabel);

nameLabelBox.setAlignment(Pos.CENTER);

nameLabelBox.setPadding(**new** Insets(**10**, **0**, **0**, **0**));

VBox buttonBox = **new** VBox(**10**, playAI, playPlayer);

buttonBox.setAlignment(Pos.CENTER);

BorderPane rootPane = **new** BorderPane();

rootPane.setTop(nameLabelBox);

rootPane.setCenter(buttonBox);

rootPane.setAlignment(nameLabel, Pos.TOP\_CENTER);

Scene mainScene = **new** Scene(rootPane, **500**, **600**);

rootPane.setStyle("-fx-background-color: " + BACKGROUND\_COLOR + ";");

menuStage.setTitle(LABEL\_GAME);

menuStage.setScene(mainScene);

menuStage.show();

}

**private** **void** **startGame**(Stage stage, **boolean** isAI) {

**new** **GameBoard**(stage, isAI);

}

Листинг 2. Реализация меню и выбора режима игры

**public** **void** **update**() {

circle.setCenterX(circle.getCenterX() + velocityX);

circle.setCenterY(circle.getCenterY() + velocityY);

}

**private** **void** **updateGame**() {

ball.update();

**for** (Brick brick : bricks) {

**if** (brick != **null** && ball.getCircle().getBoundsInParent().intersects(brick.getRectangle().getBoundsInParent())) {

System.out.println("Collision with brick!");

**double** ballCenterX = ball.getCircle().getCenterX();

**double** ballCenterY = ball.getCircle().getCenterY();

**double** brickLeft = brick.getRectangle().getX();

**double** brickRight = brickLeft + brick.getRectangle().getWidth();

**double** brickTop = brick.getRectangle().getY();

**double** brickBottom = brickTop + brick.getRectangle().getHeight();

**if** (ballCenterX >= brickLeft && ballCenterX <= brickRight) {

ball.reverseY();

}

**if** (ballCenterY >= brickTop && ballCenterY <= brickBottom) {

ball.reverseX();

}

**if** (Math.pow(ballCenterX - brickLeft, **2**) + Math.pow(ballCenterY - brickTop, **2**) <= Math.pow(ball.getCircle().getRadius(), **2**) ||

Math.pow(ballCenterX - brickRight, **2**) + Math.pow(ballCenterY - brickTop, **2**) <= Math.pow(ball.getCircle().getRadius(), **2**) ||

Math.pow(ballCenterX - brickLeft, **2**) + Math.pow(ballCenterY - brickBottom, **2**) <= Math.pow(ball.getCircle().getRadius(), **2**) ||

Math.pow(ballCenterX - brickRight, **2**) + Math.pow(ballCenterY - brickBottom, **2**) <= Math.pow(ball.getCircle().getRadius(), **2**)) {

ball.reverseX();

ball.reverseY();

}

brick.hit();

**if** (brick.getHitPoints() <= **0**) {

gamePane.getChildren().remove(brick.getRectangle());

gamePane.getChildren().remove(brick.getHealthText());

}

}

}

**if** (ball.getCircle().getCenterX() <= **0** || ball.getCircle().getCenterX() >= gamePane.getWidth()) {

ball.reverseX();

}

**if** (ball.getCircle().getCenterY() <= **0** || ball.getCircle().getCenterY() >= gamePane.getHeight()) {

ball.reverseY();

}

}

**public** **void** **hit**() {

hitPoints--;

healthText.setText(String.valueOf(hitPoints));

}

**private** **void** **startLevel**(**int** level) {

gamePane.getChildren().clear();

generateBricks(level);

ball.reset(**250**, **550**);

gamePane.getChildren().add(ball.getCircle());

launchLine.setVisible(**true**);

launchAngle = **45**;

updateLaunchLine();

}

Листинг 3. Логика перехода на следующий уровень и обновления игры

gameScene.setOnKeyPressed(event -> {

**switch** (event.getCode()) {

**case** **LEFT:**

launchAngle = (launchAngle + **5**) % **360**;

updateLaunchLine();

System.out.println("Left pressed");

**break**;

**case** **RIGHT:**

launchAngle = (launchAngle - **5**) % **360**;

updateLaunchLine();

System.out.println("Right pressed");

**break**;

**case** **SPACE:**

ball.shoot(launchAngle);

System.out.println("Space pressed");

**break**;

**default**:

**break**;

}

});

**public** **void** **shoot**(**double** angle) {

velocityX = SPEED \* Math.cos(Math.toRadians(angle));

velocityY = -SPEED \* Math.sin(Math.toRadians(angle));

}

Листинг 4. Реализация стрельбы игроком

* 1. **Интерфейс приложения**

Интерфейс приложения реализован очень просто. При запуске приложения появляется главное меню (рис. 12), в котором можно выбрать режим игры: игрок или ИИ.

Изображение выглядит как текст, снимок экрана, Шрифт, Прямоугольник

Автоматически созданное описание

Рис. 12 Меню приложения

После выбора режима игры запускается поле с блоками и шаром (рис. 13) в выбранном режиме игры.

Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение

Автоматически созданное описание

Рис. 13 Интерфейс игры

Отображение меню реализовано в листинге 2, методы для отрисовки игрового поля отображены в листингах ….

**private** **void** **generateBricks**(**int** level) {

bricks = **new** ArrayList<>();

Random random = **new** Random();

**int** rows = **3**;

**int** cols = **5**;

**int** brickSize = **50**;

**int** padding = **10**;

**int** startX = (**int**) (gamePane.getWidth() - (cols \* (brickSize + padding))) / **2**;

**for** (**int** row = **0**; row < rows; row++) {

**int** y = row \* (brickSize + padding) + **50** - level \* (brickSize + padding);

**for** (**int** col = **0**; col < cols; col++) {

**int** x = startX + col \* (brickSize + padding);

**int** hitPoints = random.nextInt(**30**) + **10**;

Brick brick = **new** Brick(x, y, brickSize, hitPoints);

bricks.add(brick);

gamePane.getChildren().add(brick.getRectangle());

gamePane.getChildren().add(brick.getHealthText());

}

}

}

**private** **void** **updateLaunchLine**() {

**double** ballX = ball.getCircle().getCenterX();

**double** ballY = ball.getCircle().getCenterY();

**double** endX = ballX + Math.cos(Math.toRadians(launchAngle)) \* **50**;

**double** endY = ballY - Math.sin(Math.toRadians(launchAngle)) \* **50**;

System.out.println("Launch Line Start: (" + ballX + ", " + ballY + ")");

System.out.println("Launch Line End: (" + endX + ", " + endY + ")");

launchLine.setStartX(ballX);

launchLine.setStartY(ballY);

launchLine.setEndX(endX);

launchLine.setEndY(endY);

launchLine.setStroke(Color.WHITE);

gamePane.getChildren().remove(launchLine);

gamePane.getChildren().add(launchLine);

}

Листинг 5. Отрисовка поля игры