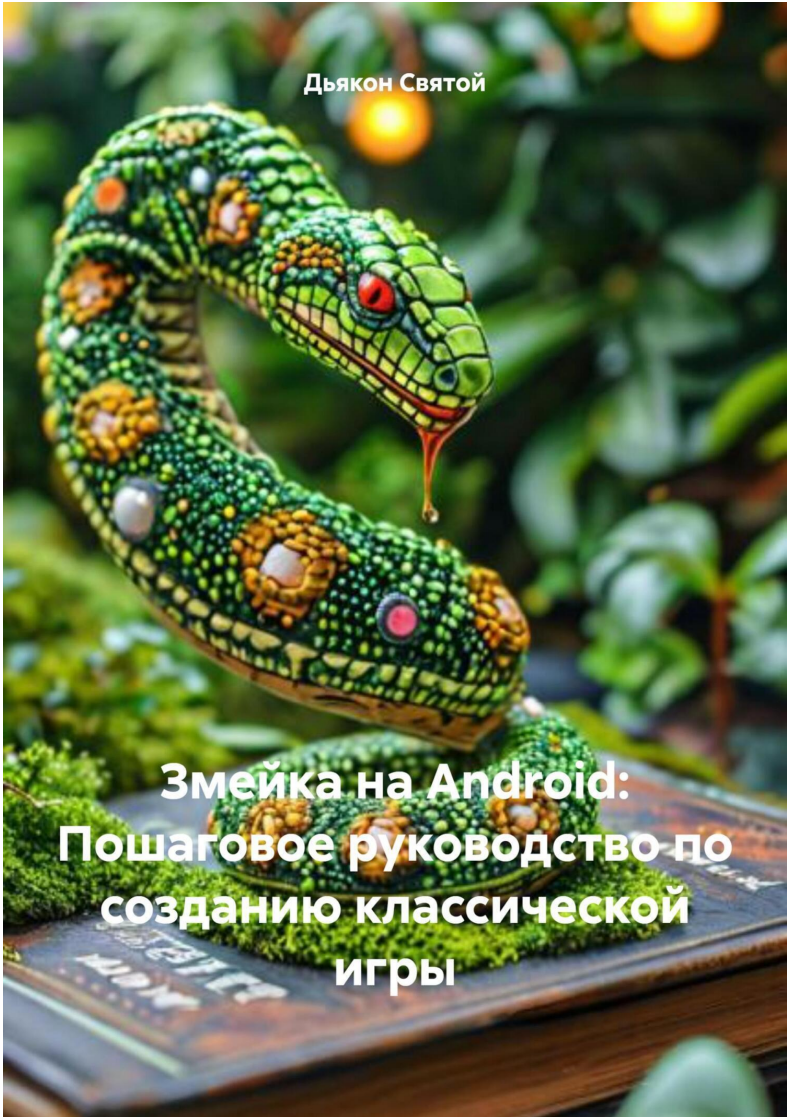
Дьякон Святой
Змейка на Android: Пошаговое руководство по созданию классической игры

Дьякон Джон Святой

Змейка на Android: Пошаговое руководство по созданию классической игры

http://www.litres.ru/pages/biblio_book/?art=72003268
SelfPub; 2025

Аннотация

Эта книга – ваше идеальное руководство по созданию игры "Змейка" на мобильной платформе Android с использованием языка программирования Java. Вы пройдете все этапы разработки, от установки программного обеспечения до создания полноценной игры, обладающей захватывающей графикой и удобным управлением. В книге представлены пошаговые инструкции, примеры кода и советы, которые помогут вам понять основные принципы разработки игр, включая обработку ввода, управление игровым циклом и оптимизацию производительности. Вы узнаете, как работать с пользовательским интерфейсом, добавлять уровень сложности и тестировать своё приложение на разных устройствах. Независимо от вашего уровня подготовки, это руководство станет вашим надежным помощником в мире разработки игр.

Дьякон Святой Змейка на Android: Пошаговое руководство по созданию классической игры

Введение в разработку игр

Разработка игр – это увлекательный и многогранный процесс, который объединяет креативность, технические навыки и аналитическое мышление. В этой главе мы рассмотрим основные аспекты разработки игр, их историческую эволюцию, основные жанры и ключевые моменты, на которые стоит обратить внимание, начиная свой путь в данной сфере. История разработки игр начинается с простых аркадных автоматов, таких как "Понг" и "Танк", которые быстро завоевали популярность в 70-х годах. Со временем технологии развивались, что привело к созданию более сложных и глубоких игр. Появление домашних игровых консолей и компьютеров произвело настоящую революцию, сделав игры доступными для широкой аудитории. На сегодняшний день существует масса жанров: от платформеров и ролевых игр до стратегий и симуляторов. Каждый жанр имеет свои уникальные особенности, механики и аудитории.

При разработке важно понимать, в каком направлении вы хотите двигаться, поскольку это определяет стиль игрового процесса, пользовательский интерфейс и даже дизайн уровней. Основные этапы разработки игр включают в себя концепцию, проектирование, программирование, тестирование и публикацию. Начальные этапы часто включают в себя создание прототипа – простой версии игры, которая позволяет протестировать основные идеи и механики. Это важный шаг, так как на нём можно выявить проблемные места и внести коррективы до того, как начнётся активная разработка. Программирование игр является одним из наиболее критически важных компонентов разработки.

Языки программирования, такие как Java, C#, C++ и Python, широко используются в зависимости от платформы. В контексте разработки на Android, Java является одним из самых популярных языков, что делает его идеальным выбором для новичков. Однако программирование – это только часть уравнения. Дизайн игры, включая визуальные элементы, звук и пользовательский интерфейс, также играет важную роль в создании увлекательного опыта для игроков. Умение работать с графикой, анимацией и звуковыми эффектами значительно улучшает игровой процесс и делает его более запоминающимся. Сейчас, когда вы получили общее представление о разработке игр и их компонентах, можно перейти к практике. В следующей главе мы подробно рассмотрим установку и настройку среды разработки Android

Studio, что станет вашим первым шагом на пути к созданию игр. Вы вооружитесь необходимыми инструментами и сможете приступить к воплощению своих идей в жизнь, погружаясь все глубже в захватывающий мир игровой разработки.

Установка и настройка Android Studio Перед тем как приступить к разработке игры на Android, необходимо установить и настроить среду разработки Android Studio. Этот процесс может показаться сложным для новичков, но следуя пошаговым инструкциям, вы сможете подготовить свою рабочую среду быстро и без проблем. Первым шагом является загрузка Android Studio с официального сайта. Перейдите на https://developer.android.com/studio и выберите версию, подходящую для вашей операционной системы – Windows, macOS или Linux. Убедитесь, что ваше устройство соответствует системным требованиям, которые указаны на странице загрузки. После загрузки установочного файла запустите его. На Windows вам, возможно, потребуется подтвердить разрешение от системы на установку. На этапе установки вы можете выбрать стандартный режим, который автоматически установит все необходимые компоненты, такие как Android SDK, эмуляторы и инструменты.

В процессе установки вам будет предложено выбрать компоненты, которые вы хотите установить. Рекомендуется оставить все параметры по умолчанию, так как они обеспечивают полноценную функциональность для разработки приложений. Скорее всего, вам также предложат установить

Java Development Kit (JDK), который необходим для работы с Java. Если он не установлен на вашем компьютере, просто следуйте инструкциям установщика. После завершения установки откройте Android Studio. При первом запуске программа проведет вас через процесс первоначальной настройки. Вам могут быть предложены различные опции: создать новый проект, импортировать существующий или загрузить примеры. Для начала обучения выберите создание нового проекта.

На следующем экране вам нужно будет ввести название вашего проекта, редактировать путь к папке проекта и определить, с какой версией Android вы будете работать. Рекомендуется выбрать последнюю стабильную версию SDK, чтобы воспользоваться всеми новыми функциями и улучшениями. После этого вам будет предложено выбрать шаблон для вашего приложения. Для создания самой простой версии вашей игры "Змейка" рекомендуется выбрать пустое активити, поскольку это даст больше свободы в разработке. Когда основной проект создан, стоит ознакомиться с интерфейсом Android Studio. Основные элементы интерфейса включают панель инструментов, окно проекта, редактор кода и консоль для вывода сообщений. Знакомство с этими элементами поможет вам лучше ориентироваться в процессе разработки. Не забудьте настроить эмулятор или подключить физическое устройство для тестирования. Для создания нового эмулятора откройте AVD Manager из панели инструментов.

Здесь вы можете выбрать тип устройства, установить версию Android и другие параметры. После создания эмулятора вы сможете запускать на нём свои приложения.

Также важно отметить, что при разработке под Android может понадобиться настройка профилей разрешений. Чтобы ваше приложение корректно функционировало на устройствах пользователей, необходимо указать необходимые права в файле манифеста. Это можно сделать в разделе "AndroidManifest.xml", который удобно открывать и редактировать в Android Studio. После того как все настройки завершены, вы готовы перейти к следующему шагу – созданию вашего первого проекта. Но перед этим рекомендуется протестировать собранное приложение на эмуляторе или реальном устройстве. Убедитесь, что все работает так, как задумано, и сделайте заметки по возможным улучшениям. С установкой и настройкой Android Studio завершено, перед вами открываются двери в мир разработки игр на платформе Android.

В следующей главе мы на практике создадим первый проект, что даст вам возможность увидеть, как теоретические концепции воплощаются в коде. Будьте готовы к увлекательному процессу, в котором ваши идеи будут постепенно превращаться в реальность. Создание первого проекта После успешной установки и настройки Android Studio, вы готовы приступить к созданию первого проекта, что является важным шагом на пути к разработке вашей игры "Змейка". Да-

вайте разберем этот процесс поэтапно, чтобы вам было удобно следовать инструкциям и понимать, что происходит на каждом этапе. Запустите Android Studio и на главном экране выберите опцию для создания нового проекта. Вам будет предложено выбрать название и местоположение для вашего проекта. Назовите его, например, "SnakeGame", и выберите директорию, в которой он будет сохранён. Убедитесь, что путь содержит только латинские буквы и не содержит специальных символов, так как это может потенциально вызвать проблемы. На следующем экране вам нужно будет выбрать тип приложения. Для нашей игры "Змейка" идеальным выбором будет "Empty Activity", так как это предоставит чистый старт для нашей разработки. В данном случае вы сможете самостоятельно определять структуру и элементы интерфейса. Обязательно укажите название главной активности, например, "MainActivity". После этого Android Studio предложит настроить файл манифеста, который содержит основную информацию о приложении, такую как название, иконка приложения, права доступа и другие детали. Для нашего проекта стандартные настройки подойдут, но не забудьте ознакомится с ними, поскольку они могут потребоваться в дальнейшем, когда вы будете добавлять новые функции. Нажмите «Finish», и Android Studio создаст новый проект.

Это может занять несколько минут, так как среда загрузит все необходимые библиотеки и настройки. После заверше-

ния процесса вы увидите структуру вашего проекта в панели слева. Важной частью разработки является ознакомление с основными файлами и папками, которые были созданы. Обратите внимание на папку "app", где находятся исходные файлы вашего приложения. В ней вы увидите директорию "src", в которой, в свою очередь, находится "main" – это основное место для написания кода. Файл "MainActivity.java" – это ваше главное место для кода, где будет происходить основная логика приложения. Чтобы увидеть, как выглядит ваша игра на устройстве, вам нужно будет запустить приложение. Вы можете использовать эмулятор, который вы настраивали ранее, или подключить физическое устройство через USB для тестирования. В случае с эмулятором, выберите его в верхнем меню и нажмите на кнопку запуска. Android Studio автоматически соберёт проект и запустит его на выбранном эмуляторе.

Когда приложение запустится, вы увидите экран, на котором должно быть написано "Hello World!" – это стандартный текст, который генерируется по умолчанию. Этот текст вы можете заменить, добавив свою логику и элементы интерфейса. Подумайте о том, как будет выглядеть интерфейс вашей игры "Змейка". В этом разделе мы будем добавлять графику, контроллеры и другие элементы. Чтобы изменить текст, откройте файл "activity_main.xml", который находится в директории "res/layout". Здесь вы сможете редактировать разметку вашего пользовательского интерфейса. Вы може-

те использовать XML для описания визуальных компонентов, или воспользоваться графическим редактором, который предоставляет Android Studio. Давайте изменим стандартный текст на более подходящий для нашей игры. Удалите существующий TextView и добавьте новый элемент. Для начала вы можете добавить простую разметку с текстом "Змейка".

Пример кода может выглядеть так:

```xml
<TextView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:text="Змейка"
    android:textSize="30sp" />
```

После внесения изменений сохраните файл и запустите приложение еще раз. Теперь вы должны увидеть текст "Змейка" в центре экрана. На этом этапе вы создали базовую структуру проекта и настроили пользовательский интерфейс. У вас есть все необходимое для перехода к кодированию логики игры. В следующей главе мы разберем основы создания структуры игры, классов и объектов, которые составят основу вашей "Змейки". Не бойтесь экспериментировать с кодом и интерфейсом – это отличный способ научиться и понять, как работает Android-платформа. Каждое изменение, которое вы вносите, приближает вас к созданию полноценной игры. Основы пользовательского интерфейса

Когда мы говорим о разработке игр на платформе Android, одним из ключевых аспектов является создание интуитивно понятного и привлекательного пользовательского интерфейса (UI).

Пользовательский интерфейс – это то, как игроки взаимодействуют с вашим приложением, и он играет важную роль в создании общего впечатления от игры. В этой главе мы рассмотрим основные элементы пользовательского интерфейса, которые понадобятся для вашей игры "Змейка".

Начнем с разметки. Android использует XML для описания пользовательского интерфейса. Разметка, которая отвечает за отображение элементов на экране, создается в файлах с расширением .xml, находящихся в папке res/layout. Наш первый шаг – создать разметку для основного экрана игры. Создайте файл activity_game.xml, в котором вы определите элементы интерфейса для вашей игры. Мы можем добавить следующие элементы: игровое поле, кнопки управления сменой направления змейки и текстовую строку для отображения счета.

Пример структуры XML для игрового интерфейса может выглядеть так:

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent">
```

```
    <TextView
    android:id="@+id/tvScore"

    android:layout_width="wrap_content"
android:layout_height="wrap_content"
    android:text="Счет: 0"
    android:textSize="24sp"
    android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"/>
    <SurfaceView
    android:id="@+id/surfaceView"
    android:layout_width="match_parent"
android:layout_height="match_parent"
    android:layout_below="@id/tvScore"/>
    <LinearLayout          android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_alignParentBottom="true"
    android:orientation="horizontal"
    android:gravity="center">
    <Button
    android:id="@+id/btnLeft"
    android:layout_width="wrap_content"
android:layout_height="wrap_content"
    android:text="Влево" />
    <Button                              android:id="@+id/btnRight"
android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:text="Вправо" />
    <Button
    android:id="@+id/btnUp"

    android:layout_width="wrap_content"
android:layout_height="wrap_content"
    android:text="Вверх" />
    <Button
    android:id="@+id/btnDown"
    android:layout_width="wrap_content"
android:layout_height="wrap_content"
    android:text="Вниз" />
    </LinearLayout> </RelativeLayout> ```
```

В данном примере мы создали основной контейнер RelativeLayout, который содержит текстовое поле для счета, игровую область SurfaceView и кнопки для управления. SurfaceView представляет собой специальный вид, который идеально подходит для отображения графики в реальном времени, что делает его идеальным для игр. После того как вы создали разметку, следующим шагом будет подключение этой разметки к вашему классу активности. Откройте основной класс вашей активности (MainActivity), и добавьте в него следующий код для связывания интерфейса с логикой игры:

```java public class MainActivity extends AppCompatActivity { private TextView tvScore;
private SurfaceView surfaceView;
@Override protected void onCreate(Bundle savedInstanceState)
{ super.onCreate(savedInstanceState); setContentView(R.layout.activity_game);
tvScore = findViewById(R.id.tvScore);
surfaceView = findViewById(R.id.surfaceView);
// Инициализация игрового процесса } } ```

Теперь мы можем собрать и запустить приложение, чтобы увидеть, как созданный интерфейс отображается на экране устройства. Вы уже можете заметить, что текущее состояние игрового интерфейса позволяет вам просмотреть счёт, а также использовать кнопки для управления змейкой. Далее рассмотрим более детально взаимодействие с элементами интерфейса. Для этого вам необходимо установить обработчики событий на кнопки управления.

Добавьте следующий код в метод onCreate вашего MainActivity: Button btnLeft = findViewById(R.id.btnLeft);
Button btnRight = findViewById(R.id.btnRight);
Button btnUp = findViewById(R.id.btnUp);
Button btnDown = findViewById(R.id.btnDown);
btnLeft.setOnClickListener(v -> {
// Логика для движения влево });
```

```
btnRight.setOnClickListener(v -> {
    // Логика для движения вправо });
btnUp.setOnClickListener(v -> {
    // Логика для движения вверх });
btnDown.setOnClickListener(v -> {
    // Логика для движения вниз });
```

Каждое нажатие на кнопку будет вызывать соответствующий обработчик, в котором вы сможете реализовать логику движения вашей змейки. На этом этапе важно знать, как обрабатывать события ввода с помощью кнопок, так как это основа для управления игровым процессом. Кроме кнопок, вы также можете добавить другие элементы пользовательского интерфейса, такие как панель с уровнями сложности или элемент для отображения времени. Все это поможет сделать вашу игру более интерактивной и интересной. Также стоит обратить внимание на адаптивный дизайн. Вы должны убедиться, что интерфейс будет хорошо отображаться на устройствах с разными размерами экранов и разрешениями. Одним из решений может быть использование ресурсов и альтернативных ресурсов (например, drawable-mdpi, drawable-hdpi и т.д.) для различных экранов.

Будьте внимательны к тому, чтобы избежать жестко закодированных значений ширины и высоты, как это не рекомендуется в современных приложениях. На этом этапе у вас уже есть основа для пользовательского интерфейса вашей игры "Змейка". Вы реализовали основные элементы, такие как иг-

ровое поле, обработку ввода и счёт. Это будет вашей стартовой точкой для дальнейших описаний игрового процесса, когда мы будем добавлять логику игры и графику. Не бойтесь экспериментировать с элементами интерфейса – это поможет вам лучше понять, как создавать удобные и привлекательные приложения. В следующих главах мы продолжим развивать вашу игру, добавляя интерактивность и графику.
Структура игры: классы и объекты

При разработке игры "Змейка" на платформе Android важно правильно организовать код, чтобы он был понятным, удобным для сопровождения и расширяемым. Основой такой организации является объектно-ориентированное программирование (ООП), которое позволяет разбить вашу игру на небольшие, управляемые части, называемые классами. В этой главе мы рассмотрим, как создаются классы и объекты для вашей игры, а также как их правильно структурировать. Первая структура, которую мы создадим, будет представлять саму игру. В нашем случае это будет класс SnakeGame. Он будет отвечать за управление игровым процессом, включая логику движения змейки, генерацию еды и отслеживание счета.

Ниже представлен упрощенный вариант класса SnakeGame: public class SnakeGame { private Snake snake; private Food food; private int score; public SnakeGame() { this.snake = new Snake(); this.food = new Food(); this.score = 0; }

```
public void moveSnake(Direction direction) {
snake.move(direction);
// Логика проверки столкновения и генерации еды }
public void increaseScore() {
score++; // Обновление счета в UI }
// Геттеры и другие методы }
```

Класс Snake представляет собой змейку. Он включает в себя информацию о текущих координатах и направлении движения, а также методы, которые позволяют изменять состояние змейки:

```
public class Snake { private List<Point> body;
private Direction direction; public Snake() {
body = new ArrayList<>();
// Инициализация змейки }
public void move(Direction direction) {
// Логика движения змейки с добавлением новой головы и удалением хвоста }
public void grow() {
// Логика увеличения длины змейки }
// Геттеры и другие методы }
```

В данном примере класс Snake содержит список координат тела змейки и направление движения. Метод move отвечает за перемещение змейки, а метод grow позволяет увеличивать её длину, когда она "съедает" еду. Следующий класс, который нам нужен, – это класс Food, который описывает еду для змейки.

Он будет хранить координаты еды и метод для генерации еды в случайной позиции:

```
public class Food {
private Point position;
public Food() {
spawn(); }
public void spawn() {
// Логика генерации случайной позиции для еды
}
// Геттеры и другие методы }
```

Класс Food отвечает за создание и место расположения еды в игровом поле. Метод spawn позволяет случайным образом позиционировать еду на игровом экране.

Для управления направление движения змейки можно использовать перечисление Direction:

```
public enum Direction { UP, DOWN, LEFT, RIGHT }
```

Эта простая структура классов позволит вам разбить вашу игру на логические блоки, что делает код более понятным и структурированным. Все классы взаимодействуют между собой, например, класс

SnakeGame будет использовать методы класса Snake для движения и роста змейки и класс Food для генерации еды.

Теперь, когда основные классы созданы, у вас есть возможность добавлять новые функции, такие как разные уровни сложности, новые типы еды или даже специальные улучшения для змейки. Вы можете создать дополнительные клас-

сы для этих элементов, что также сделает вашу игру более расширяемой. Важно помнить, что при разработке игры вам, возможно, придется пересматривать структуру классов по мере добавления новых функций или элементов. Хорошая практика заключается в том, чтобы не делать классы слишком большими и не перегружать их функциями. Каждый класс должен иметь одно четкое назначение. Когда мы переходили к следующими этапам разработки, такими как реализация логики игры и отрисовка графики, ваша структура классов поддержит чистоту и ясность кода. Это особенно важно, когда вы вернетесь к проекту через несколько месяцев или разрабатываете его совместно с другими программистами. Подводя итог, создание структуру классов и объектов в вашей игре "Змейка" является фундаментальным шагом к организации кода.

Правильное применение принципов ООП поможет вам создать логику игры, которая будет легко изменять и поддерживать. В следующих главах мы будем интегрировать эту структуру с другими элементами игры, такими как управление и графика, что поможет вам разработать готовый продукт, способный привлечь внимание пользователей. Логика игры: создание класса SnakeGame Создание класса SnakeGame является одним из ключевых этапов в разработке игры "Змейка".

Этот класс будет служить основным контроллером игры, объединяя логику и управление различными её аспектами. В

этой главе мы разберем, как реализовать класс SnakeGame, который будет управлять состоянием игры, следить за движением змейки и взаимодействием с объектами, такими как еда и стены. Наш класс SnakeGame будет включать в себя несколько ключевых элементов: инициализацию змейки и еды, управление игровым процессом, обработку логики столкновений и обновление состояния игр. Ниже представлен базовый набросок класса: public class SnakeGame { private Snake snake; private Food food;

```
private int score; private int width;
private int height; private boolean isGameOver;
public SnakeGame(int width, int height)
{ this.width = width;
this.height = height;
this.snake = new Snake();
this.food = new Food();
this.score = 0;
this.isGameOver = false;
spawnFood();
} public void spawnFood()
{ food.spawn(width, height, snake.getBody());
} public void update(Direction direction)
{ if (!isGameOver) {
snake.move(direction);
checkCollision(); } }
private void checkCollision() {
```

```
if (snake.hasCollidedWithWall(width, height) || snake.hasCollidedWithItself())
{ isGameOver = true;
} else if (snake.headEquals(food.getPosition())) {
score++;
snake.grow();
spawnFood(); } } public boolean isGameOver() { return isGameOver; } public int getScore() {
return score; } }
```

В этом классе мы объявляем ряд переменных, которые будут использоваться для отслеживания состояния игры, включая ширину и высоту игрового поля, объект змейки, еду и текущий счет. Как только класс инициализируется, мы предоставляем ему размер игрового экрана и создаем начальные состояния, включая саму змейку и еду.

Метод spawnFood отвечает за генерацию еды на случайной позиции, которая не пересекается с телом змейки. Для этого мы можем использовать метод, определенный в классе Food, который принимает размеры игрового поля и текущее состояние змейки, чтобы избежать наложений. Метод update является сердцем игровой логики.

Он принимает направление изменения движения змейки и загружает новую позицию в каждом кадре игрового цикла. В методе checkCollision происходит проверка на столкновения со стенами поля или самим собой. Если змейка сталкивается с целью, увеличивается счет и вызывается генерация

новой еды.

Теперь давайте более подробно рассмотрим методы hasCollidedWithWall, hasCollidedWithItself и headEquals, которые смогут помочь в проверке столкновений.

В классе Snake, метод hasCollidedWithWall может выглядеть следующим образом:

```
public boolean hasCollidedWithWall(int width, int height) {
Point head = body.get(0);
return head.x < 0 || head.x >= width || head.y < 0 || head.y >= height; }
```

А метод hasCollidedWithItself будет проверять соприкосновение головы с телом змейки: public boolean hasCollidedWithItself()

```
{ Point head = body.get(0);
for (int i = 1; i < body.size(); i++) {
if (body.get(i).equals(head)) {
return true; } } return false; }
```

И последний метод headEquals в классе Food, который проверяет, совпадает ли положение головы змеи с позицией еды:

```
public boolean headEquals(Point foodPosition) {
Point head = body.get(0);
return head.equals(foodPosition); }
```

Эти методы помогают изолировать логику проверки состояния игры, обеспечивая простоту и читаемость кода. Такой подход к организации класса SnakeGame позволяет вы-

делить различные аспекты игрового процесса и минимизировать взаимосвязи между ними, что облегчает дальнейшее расширение проекта.

При добавлении новых функций, таких как дополнительные уровни сложности или новые типы еды, вы всегда можете безопасно модифицировать класс, не затрагивая другие его части. Теперь, когда мы завершили базовую реализацию класса SnakeGame, вы можете перейти к интеграции этого класса с другими элементами игры, такими как управление вводом пользователя и отрисовка графики.

Это даст вам возможность создать полноценную игровую логику, в которую будут вовлечены все аспекты игрового процесса. Такой продуманный подход к разработке логики игры помогут вам избежать многих проблем в будущем, делая код более стабильным и управляемым. Управление змейкой через ввод пользователя Управление змейкой является одним из самых важных аспектов игры "Змейка". Удобный и отзывчивый интерфейс управления позволяет игрокам легко направлять свою змейку, избегая препятствий и собирая еду.

В этой главе мы разберем, как реализовать управление движением змейки с помощью сенсорного ввода на устройствах Android. Существует несколько способов реализации управления для игры "Змейка". Один из самых распространенных методов – использование кнопок на экране. Однако в нашей реализации мы сосредоточим внимание на исполь-

зовании сенсорного ввода, который будет более интуитивным и современным. Мы будем отслеживать касания экрана и определять направление движения змейки на основе этих касаний.

Начнем с создания пользовательского интерфейса для управления. В классе GameView, отвечающем за отрисовку игры, мы можем добавить методы для обработки касаний. Первым делом необходимо переопределить метод onTouchEvent, который срабатывает при касании экрана. Вот пример, как это может выглядеть:

Конец ознакомительного фрагмента.