

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**   
**«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ»**

Факультет Физико-технический

Кафедра Компьютерных технологий (КТ)

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | КТ |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | Т. В. Ермоленко |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе бакалавра 2 курса

на тему:

Разработка приложения “Проектирование игр для мобильных устройств”

|  |  |  |  |
| --- | --- | --- | --- |
| Автор работы |  |  | Задорожний В.И. |

подпись

|  |  |  |
| --- | --- | --- |
| Направление | 09.03.01 | Информатика и вычислительная техника |

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преподаватель Маруга М.М.

подпись

Консультанты по разделам:

Техническое задание \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ доцент Т.В. Шарий

подпись

Нормоконтроль\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. лаборант В.Г. Медведева

подпись

Курсовая работа защищена \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата итоговая оценка комиссия

Подписи членов комиссии: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Донецк  
2020

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc38893484)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc38893485)

[1.1 Состояние вопроса 4](#_Toc38893486)

[1.2 Актуальность и цель работы 8](#_Toc38893487)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 10](#_Toc38893488)

[2.1 Описание области применения и исходных данных приложения 10](#_Toc38893489)

[2.2 Требования к пользовательским интерфейсам 11](#_Toc38893490)

[2.3 Требования к аппаратным, программным и коммуникационным интерфейсам 16](#_Toc38893491)

[2.4 Требования к пользователям продукта 17](#_Toc38893492)

[2.5 Требования к адаптации на месте 17](#_Toc38893493)

[2.6 Функции продукта 17](#_Toc38893494)

[2.7 Ограничения 18](#_Toc38893495)

[3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ 19](#_Toc38893496)

[4 РАЗРАБОТКА ПРИЛОЖЕНИЯ МОЙ КОШЕЛЁК 21](#_Toc38893497)

[4.1 Входные и выходные данные приложения 21](#_Toc38893498)

[4.2 Проектирование структуры приложения 23](#_Toc38893499)

[4.3 Описание объектов и их взаимодействия для ООП 26](#_Toc38893500)

[5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 30](#_Toc38893501)

[5.1 Аппаратные и программные средства создания и эксплуатации интернет-приложения 30](#_Toc38893502)

[5.2 Руководство пользователя 30](#_Toc38893503)

[5.3 Описание контрольных примеров 36](#_Toc38893504)

[ЗАКЛЮЧЕНИЕ 41](#_Toc38893505)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 43](#_Toc38893506)

[ПРИЛОЖЕНИЕ А Экранные формы 44](#_Toc38893507)

[ПРИЛОЖЕНИЕ Б Фрагменты листинга 49](#_Toc38893508)

# ВВЕДЕНИЕ

Курсовая работа посвящена написанию небольшой игры под Android и Desktop на основе популярной игры Flappy Bird.

В ходе написания работы подробно проанализировано состояние вопроса в указанной предметной области. Изучение различных материалов, и проэктов по написанию Android игр.

Целью работы является разработка оконного приложения с простой, линейной игрой. Для достижения данной цели решались следующие задачи:

1. Анализ предметной области и подготовка технического задания;

2. Изучение основных способов написания мини-игр;

3. Изучение анимации и прорисовки объектов;

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Состояние вопроса

В наше время сложно представить себе человека без сотового телефона, планшетного компьютера, смартфона или любого другого портативного мультимедийного устройства. Мы привыкли к тому, что всегда под рукой не только средство связи, но и множество полезных функций, таких как: калькулятор, органайзер, конвертер, календарь, часы. Смартфоны становятся новой мобильной игровой платформой, соревнуясь с классическими карманными игровыми системами вроде Nintendo DS или Playstation Portable.

1.2 Актуальность и цель работы

# В устройстве смартфона все довольно просто. Главным образом он состоит из нескольких отдельных блоков — памяти, процессора, который занимается вычислениями, хранилища данных, радиомодуля, который в свою очередь состоит из приемника и передатчика и отвечает за связь. Самое интересное здесь — операционная система, установленная на встроенную память. От операционной системы и ее версии зависят все основные возможности устройства. Смартфоны, как и персональные компьютеры, существуют в абсолютно разных комплектациях и под управлением разных операционных систем, разновидности которых мы рассмотрим далее.

# По мере роста продаж мобильных устройств во всем мире, растет и спрос на различные приложения для них. Каждая уважающая себя компания, стремится иметь хотя бы одно мобильное приложение, чтобы быть у своего клиента «всегда под рукой». А существование некоторых компаний и вовсе сложно представить без мобильных устройств и специализированных программ, при помощи которых можно, например, управлять базами данных или следить за состоянием своего продукта на рынке в любой момент времени.

# К сожалению, на сегодняшний день не существуют определенного стандарта средства разработки мобильных приложений. Каждый производитель пытается сделать операционную систему в своем устройстве более уникальной и запоминающейся пользователю, и как следствие возникают вопросы совместимости различных приложений на разных операционных системах.

Основной целью данной работы является разработка клиент-серверного игрового приложения на примере игры в шашки для мобильных устройств на базе операционной системы Android. И изучение среды разработки Netbeans IDE.

Наличие операционной системы (ОС) — главная особенность, отличающая смартфон от обычного мобильного телефона. При выборе конкретной модели смартфона или коммуникатора ОС часто становится определяющим фактором.

Наиболее распространённые операционные системы и платформы для смартфонов:

1) Symbian OS — занимала большую часть рынка ОС для смартфонов до конца 2010 г. По состоянию на начало 2010 года на базе этой ОС осталась всего 1 платформа: Series 60, которая используется преимущественно в устройствах Nokia, а также некоторых моделях Samsung;

2) BlackBerry OS (RIM) — устройства на этой системе широко используются в основном в США, так как спецслужбы некоторых стран не заинтересованы в использовании этих смартфонов в своей стране из-за того, что все входящие и исходящие данные шифруются с помощью алгоритма шифрования AES;

3) Windows Mobile и Windows CE — компактная ОС компании Microsoft, выпускается с 1996 года и занимала крупный сегмент рынка ОС для смартфонов до 2010 г. В настоящее время переживает постепенный отказ от поддержки и разработки;

4) Windows Phone 7 — разработка компании Microsoft, кардинально отличающаяся от Windows Mobile; 8

5) Palm OS — некогда популярная платформа, в настоящее время аппараты на базе Palm OS малораспространены. Последний смартфон под управлением данной операционной системы был представлен в конце 2007 года (Palm Centro);

6) Linux — широкого распространения не получили, однако традиционно считаются перспективным направлением. Смартфоны на базе Linux распространены в основном в Азии;

7) Bada — новейшая мобильная платформа, разработанная компанией Samsung. Первым телефоном на новой платформе стал S8500 Wave;

8) Android — портативная (сетевая) операционная система для коммуникаторов, планшетных компьютеров, электронных книг, цифровых проигрывателей, наручных часов и нетбуков основанная на ядре Linux. Изначально разрабатывалась компанией Android Inc., которую затем купила Google. Впоследствии Google инициировала создание альянса Open Handset Alliance (OHA), который сейчас занимается поддержкой и дальнейшим развитием платформы. Android позволяет создавать Java-приложения, управляющие устройством через разработанные Google библиотеки. Android Native Development Kit позволяет портировать (но не отлаживать) библиотеки и компоненты приложений, написанные на Си и других языках;

9) iOS (до 24 июня 2010 года — iPhone OS) — мобильная операционная система разрабатываемая и выпускаемая американской компанией Apple. Была выпущена в 2007 году; первоначально — для iPhone и iPod touch, позже — для таких устройств, как iPad и Apple TV. В отличие от Windows Phone и Google Android, выпускается только для устройств, производимых фирмой Apple;

10) Windows Phone 8 — второе поколение операционной системы Windows Phone от Microsoft. Выход произошел 29 Октября, и является прототипом интерфейса известного как Metro (или Modern UI). В Windows Phone 8 используется новая архитектура Windows NT которая используется в операционных системах Microsoft. Устройства под управлением Windows Phone 7. x не могут обновиться до Windows Phone 8 и новые приложения созданные для Windows Phone 8 не могут запускаться на Windows Phone 7. x.

В настоящее время Android развивается экспоненциально: каждый год число пользователей данной ОС неуклонно растет. Согласно последнему отчету компании Canalys, ведущему аналитику хай-тек индустрии, операционная система Android занимает 69,2% мирового рынка мобильных устройств. Безусловно, этот факт привлекает множество разработчиков к созданию мобильных приложений именно для Android. Пожалуй, на сегодняшний день это самая популярная и интересная система. Разработчикам даны уникальные возможности. Установив набор бесплатного программного обеспечения, можно создавать программы под данную систему и продавать их в специализированном интернет-магазине.

Android — это основанная на Linux платформа для мобильных устройств, разработанная Open Handset Alliance (OHA), инициированным Google. Она позволяет создавать Java-приложения, управляющие устройством через разработанные Google библиотеки. Также есть возможность писать приложения на Си и других языках программирования с помощью Android Native Development Kit.1.5 (Cupcake) — выпущено 30 апреля 2009 года. Среди основных улучшений появилась поддержка записи и просмотра видео в режиме камеры; поддержка Bluetooth A2DP; возможность автоматически подключаться к Bluetooth-гарнитуре.

Первым устройством, работающим под управлением Android, стал разработанный компанией HTC смартфон T-Mobile G1, презентация которого состоялась 23 сентября 2008 года. Вскоре последовали многочисленные анонсы других производителей смартфонов о намерении выпустить устройства с Android.

В компании Google выделяют несколько основных преимуществ, отличающих устройства на базе платформы Android от аналогичных продуктов:

1. Открытость — Android позволяет получить доступ к основным функциям мобильных устройств с помощью стандартных вызовов API.

2. Разрушение границ — можно объединять информацию из интернета с данными телефона, например контактной информацией или данными о географическом положении, чтобы получить новые возможности.

3. Равноправие приложений — для Android нет разницы между основными приложениями телефона и сторонним программным обеспечением — можно изменить даже программу для набора номера или заставку экрана.

4. Быстрая и легкая разработка — в SDK есть все, что нужно для создания и запуска приложений Android, включая имитатор настоящего прибора и расширенные инструменты отладки.

Гибкость Android имеет свою цену: компании, предпочитающие разрабатывать собственные пользовательские интерфейсы, вынуждены постоянно гнаться за релизами новых версий ОС. Устройства, выпущенные всего несколько месяцев назад, становятся устаревшими, поскольку операторы и производители не хотят создавать обновления ПО, чтобы пользователи могли применять новые возможности Android. Так, например, многие эксперты отмечают, что платформа базируется на Java, поэтому преимущества и возможности операционной системы Linux на Android используются не в полной мере. Также в платформе не используется ни один из популярных графических инструментов (toolkit) и библиотек (например, Ot или GTK), что делает маловероятным появление большого числа приложений, портированных с полноценного варианта Linux для домашнего компьютера на данную платформу из-за отсутствия по умолчанию X-сервера и распространенных графических библиотек. Кроме того, появилась информация о том, что Google будет по своему усмотрению удалять приложения на телефонах пользователей, если нарушаются условия их использования.

К недостаткам платформы можно также отнести и невозможность установки некоторых приложений на карту памяти. Данный пробел разработчиков является существенным, в особенности, если у телефона небольшой объем встроенной памяти (например, у T-Mobile G1 он составляет всего 70 Мб).

Аналитики и эксперты ИТ-рынка пророчат Google Android хорошие коммерческие перспективы, что в принципе для продуктов на базе ПО с открытым кодом уже не является сенсацией. Они постепенно захватывают ИТ-пространство, вытесняя с него общепризнанных лидеров, порождая конкуренцию, что само по себе может только положительно сказаться на оздоровлении рынка.

Большая часть кода лицензирована под Apache License 2 открытой и неограниченной лицензией Допускается свободное использование исходного кода для создания собственных систем. Однако системы, объявленные как совместимые с Android, должны для начала пройти Android Compatibility Program — процесс, удостоверяющий базовую совместимость со сторонними приложениями, которые созданы сторонними разработчиками. Совместимые системы могут вливаться в экосистему Android, включающую в себя Android Market.

**2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

2.1 Описание области применения и исходных данных приложения

Данное приложение предназначено для использования на персональных компьютерах а также Android устройствах.

Исходными данными для приложения являются:

1) Управление персонажем с помощью кликов мышки по игровой области, либо нажатия на игровую область с телефона.

2.2 Требования к пользовательским интерфейсам

Для установки и работы приложения необходимо иметь вычислительную систему следующей минимальной аппаратной конфигурации:

1. Наличие основных средств ввода-вывода (мышь, клавиатура, монитор, либо Android устройсво).
2. Установленную Java 8 и выше версии (Для запуска на ПК ).

Для работы программы не требуется подключение к интернету.

2.4 Требования к пользователям продукта

Системой должны иметь возможность пользоваться следующие категории пользователей: все пользователи.

Пользователь должен:

* иметь базовые навыки работы за компьютером;
* понимать принцип работы приложения.
  1. Требования к адаптации на месте

Необходима программа-инсталлятор приложения, а также Java 8 и выше версии для запуска приложения на Персональном Компьютере. Для Android устройств, необходим только .apk файл для установки приложения.

2.6 Функции продукта

Основные функции системы:

1. Отображение меню (кнопка Play);
2. Управление Анимационной птицой с помощью нажатия на экран, либо клика мышей.
3. При соприкосновении птицы с трубой ( игровой объект ) появляется окно Game Over
4. За основным персонажем, гонится Анимационный Bed-персонаж.
5. В игре присутствует Audio.

2.7 Ограничения

* Продукт будет корректно работать только под операционными системами Windows и Android

# 3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ

# libGDX – кроссплатформенный фреймворк для разработки игр и визуализации, основанный на языке программирования Java с некоторыми компонентами, написанными на C и C++ для повышения производительности определенного кода. В настоящее время поддерживает Windows, Linux, Mac OS X, Android, iOS и HTML5 как целевые платформы.

libGDX позволяет вам написать код однажды и затем развертывать игру или приложение на нескольких платформах без модификации. Вы можете разрабатывать приложение на основном компьютере и получать огромную выгоду быстрой разработки, вместо того, чтобы ждать, когда последние изменения буду внедрены и установлены на устройство и будут скомпилированы в HTML5. Вы можете использовать все инструменты Java, чтобы быть продуктивным, насколько это возможно.

Android Studio - На начальном этапе Eclipse с плагином ADT (Android Development Tools) широко использовался в качестве среды разработки приложений под Android. Второй средой разработки под Android стал IDE от JetBrains, для которой также был создан соответствующий плагин. Но после появления Android Studio на базе IDEA, программисты начали использовать новую среду разработки приложений под Android. Android Studio — это официальная среда разработки приложения для Android. По сути Android Studio — это известная Java IDE IntelliJ IDEA с плагинами, отличающая от IDEA мелочами, но эти мелочи сильно облегчают труд разработчиков приложений под Android.Программы на Jαvα транслируются в байт-код Jαvα, выполняемый виртуальной машиной Jαvα (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор.

Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Jαvα-приложения на любом устройстве, для которого существует соответствующая виртуальная машина.

Часто к недостаткам концепции виртуальной машины относят снижение производительности. Ряд усовершенствований несколько увеличил скорость выполнения программ на Jαvα:

# 

# 4 РАЗРАБОТКА ПРИЛОЖЕНИЯ Flappy Game

4.1 Входные и выходные данные приложения

Входными данными являются клики мышкой либо по экранну благоданя методу:

@Override  
**protected void** handleInput() {  
 **if** (Gdx.*input*.justTouched())  
 **bird**.jump();  
}

И хранятся в файле PlayState.java

4.2 Проектирование структуры приложения

Структурно приложение состоит из:

1. Начальное окно ( Меню ):
   1. Надпись “Avtor: Zadoroznii Vladislav”
   2. Фоновое изображение “back.png”
   3. Кнопка Play “play2.png”
   4. Фоновая музыка
2. Игровое пространство:
   1. Фоновое изображение “back.png”
   2. Анимированный персонаж
   3. Фнимированный Bed-персонаж
   4. Надпись “Avtor: Zadoroznii Vladislav”
   5. Игровой объект “Труба”
   6. Игровой объект “Земля”
3. Окно проигрыша:
   1. Фоновое изображение back.png
   2. Надпись “Avtor: Zadoroznii Vladislav”
   3. Кнопка Game Over “GameOver.png”

4.3 Описание объектов и их взаимодействия для ООП

В программе созданы следующие класс

1. Animation – Класс Анимации персонажей
2. BedBird – Класс персонажа BedBird
3. Bird – Класс Персонажа Bird
4. FlappyGame – Класс интерфейс игры
5. GameOver – Класс кнопки GameOver
6. GameState – Вспомогательный класс
7. MenuState – Класс меню + кнопка Play
8. PlayState – Основной класс с функционалом игры, объеденяющий остальные классы
9. State - Вспомогательный класс
10. Tube – Класс игрового объекта ‘Труба’

# 5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

5.1 Аппаратные и программные средства создания и эксплуатации интернет-приложения

Аппаратные требования для работы интернет-приложения:

* процессор с тактовой частотой 1.5 ГГц.
* оперативная память 256 Мб и более;
* видеокарта с объёмом памяти 128 Мб и выше;
* монитор 1280х1024 или с более высоким разрешением.

Программные требования к приложению.

Установленные на компьютере:

* + - операционная система —Windows 7 или новее.

# ЗАКЛЮЧЕНИЕ

В результате написания курсовой работы разработано приложение,

На подобии популярной игры Flappy Bird.

В результате проделанной работы было разработано игровое клиент-серверное приложение, которое возможно установить на любое мультимедийное устройство под управление операционной системы Android версии не ниже 2.0.

В процессе разработки были исследованы различные операционные системы мобильных устройств, платформы и способы разработки игровых приложений.

Работа приложения была протестирована на двух устройствах, различного типа:

Смартфон Xiaomi Readmi Note 7

Ноутбук MSI GL63 8RC

В ходе тестирования были выявлены и исправлены ошибки несовместимости приложения с различными типами экранов и устройствами ввода.

Таким образом, можно говорить о том, что поставленные задачи были выполнены в полной мере.

Для решения задачи была использована среда разработки Android Studio и язык программирования Java.

Приложение «Flappy Game» может использоваться любым заинтересованным пользователем для убийства времени.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

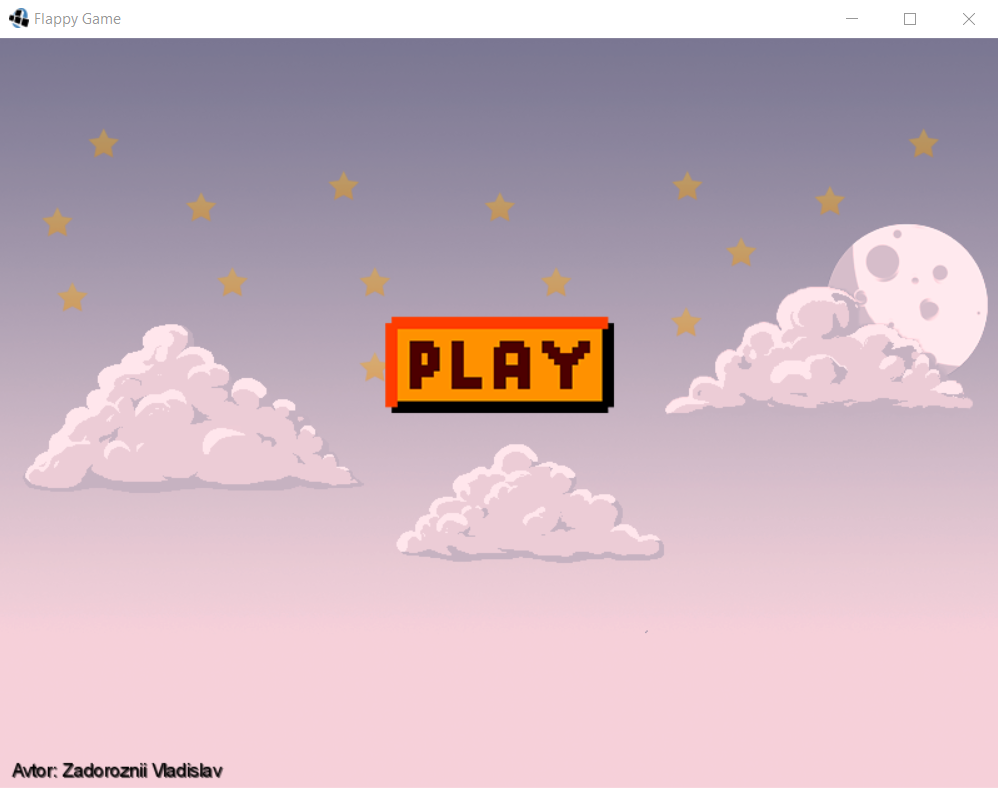
1. https://www.youtube.com/wαtch?v=plbIWp4F\_g0&list=PL0lO\_mIqDDFUzG5WOCUVmqx4CBW2qIulV

Уроки Jαvα для начинающих | #1 - Программирование на Jαvα

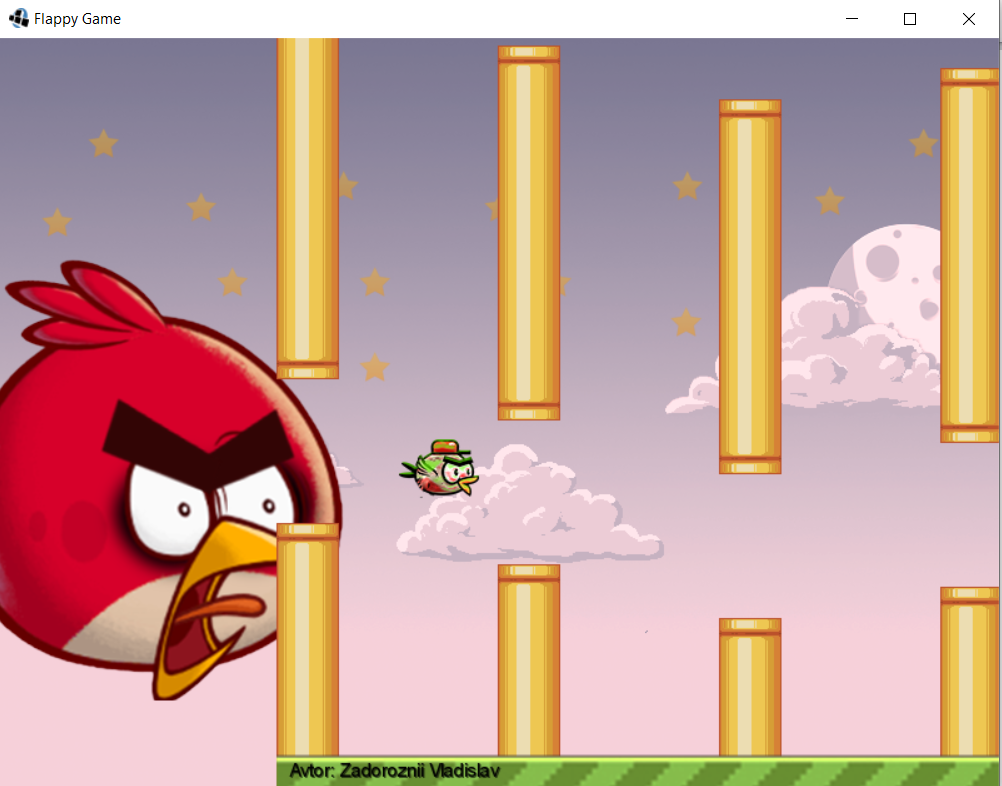
1. [https://www.youtube.com/wαtch?v=plbIWp4F\_g0&list=PL0lO\_mIqDDFUzG5WOCUVmqx4CBW2qIulV](https://www.youtube.com/watch?v=plbIWp4F_g0&list=PL0lO_mIqDDFUzG5WOCUVmqx4CBW2qIulV)

Уроки Jαvα для профессионалов | #1 - Программирование на Jαvα

1. **ПРИЛОЖЕНИЕ**
2. . Начальное окно ( Меню ):



1. Игровое пространство:



3. Окно проигрыша:



ПРИЛОЖЕНИЕ Б  
  
**public class** Animation {  
 **private** Array<TextureRegion> **frames**;  
 **private float maxFrameTime**;  
 **private float currentFrameTime**;  
 **private int frameCount**;  
 **private int frame**;  
  
 **public** Animation(TextureRegion region, **int** frameCount, **float** cycleTime) {  
 **frames** = **new** Array<TextureRegion>();  
 **int** frameWidth = region.getRegionWidth() / frameCount;  
 **for** (**int** i = 0; i < frameCount; i++) {  
 **frames**.add(**new** TextureRegion(region, i \* frameWidth, 0, frameWidth, region.getRegionHeight()));  
 }  
 **this**.**frameCount** = frameCount;  
 **maxFrameTime** = cycleTime / frameCount;  
 **frame** = 0;  
 }  
  
 **public void** update(**float** dt) {  
 **currentFrameTime** += dt;  
 **if** (**currentFrameTime** > **maxFrameTime**) {  
 **frame**++;  
 **currentFrameTime** = 0;  
 }  
 **if** (**frame** >= **frameCount**)  
 **frame** = 0;  
 }  
  
 **public** TextureRegion getFrame() {  
 **return frames**.get(**frame**);  
 }  
}

**public class** BedBird {  
  
 **private** Vector3 **positionB**;  
 **private** Vector3 **velosityB**;  
 **private** Rectangle **boundsB**;  
 **private** Animation **BbirdAnimation**;  
 **private** Texture **textureB**;  
  
 **public** BedBird(**int** x, **int** y){  
 **positionB** = **new** Vector3(x, y, 0);  
 **velosityB** = **new** Vector3(-150, 0, -150);  
 **textureB** = **new** Texture(**"BedBird3.png"**);  
 **BbirdAnimation** = **new** Animation(**new** TextureRegion(**textureB**), 3, 0.5f);  
 **boundsB** = **new** Rectangle(x, y, **textureB**.getWidth() /3, **textureB**.getHeight());  
 }  
  
 **public** Vector3 getPositionB() {  
 **return positionB**;  
 }  
  
 **public** TextureRegion getBedBird() {  
 **return BbirdAnimation**.getFrame();  
 }  
  
 **public void** update(**float** dt){  
 **BbirdAnimation**.update(dt);  
 **velosityB**.scl(1 / dt);  
 **boundsB**.setPosition(**positionB**.**x**, **positionB**.**y**);  
 }  
  
  
 **public** Rectangle getBounds(){  
 **return boundsB**;  
 }  
  
  
 **public void** dispose() {  
 **textureB**.dispose();  
 }  
}

**public class** Bird {  
 **private static final int *MOVEMENT*** = 100;  
 **private static final int *GRAVITY*** = -15;  
 **private** Vector3 **position**;  
 **private** Vector3 **velosity**;  
 **private** Rectangle **bounds**;  
 **private** Animation **birdAnimation**;  
 **private** Texture **texture**;  
  
 **public** Bird(**int** x, **int** y){  
 **position** = **new** Vector3(x, y, 0);  
 **velosity** = **new** Vector3(0, 0, 0);  
 **texture** = **new** Texture(**"BirdB.png"**);  
 **birdAnimation** = **new** Animation(**new** TextureRegion(**texture**), 3, 0.5f);  
 **bounds** = **new** Rectangle(x, y, **texture**.getWidth() /3, **texture**.getHeight());  
 }  
  
 **public** Vector3 getPosition() {  
 **return position**;  
 }  
  
 **public** TextureRegion getBird() {  
 **return birdAnimation**.getFrame();  
 }  
  
 **public void** update(**float** dt){  
 **birdAnimation**.update(dt);  
 **if** (**position**.**y** > 0)  
 **velosity**.add(0, ***GRAVITY***, 0);  
 **velosity**.scl(dt);  
 **position**.add(***MOVEMENT*** \* dt, **velosity**.**y**, 0);  
 **if** (**position**.**y** < 0)  
 **position**.**y** = 0;  
  
 **velosity**.scl(1 / dt);  
 **bounds**.setPosition(**position**.**x**, **position**.**y**);  
 }  
  
 **public void** jump(){  
 **velosity**.**y** = 250;  
 }  
  
 **public** Rectangle getBounds(){  
 **return bounds**;  
 }  
  
  
 **public void** dispose() {  
 **texture**.dispose();  
 }  
}

**public class** FlappyGame **extends** ApplicationAdapter {  
 **public static final int *WIDTH*** = 800;  
 **public static final int *HEIGHT*** = 600;  
  
 **public static final** String ***TITLE*** = **"Flappy Game"**;  
  
 **private** GameState **gsm**;  
 **private** SpriteBatch **batch**;  
 **private** Music **music**;  
 **private** BitmapFont **FontBlack1**;  
  
  
 @Override  
 **public void** create () {  
 **batch** = **new** SpriteBatch();  
 **gsm** = **new** GameState();  
 **music** = Gdx.*audio*.newMusic(Gdx.*files*.internal(**"music2.mp3"**));  
 **music**.setLooping(**true**);  
 **music**.setVolume(0.6f);  
 **music**.play();  
 Gdx.*gl*.glClearColor(1, 0, 0, 1);  
 **gsm**.push(**new** MenuState(**gsm**));  
  
 **FontBlack1** = **new** BitmapFont();  
 **FontBlack1**.setColor(Color.***BLACK***);  
  
 }  
  
  
 @Override  
 **public void** render () {  
 Gdx.*gl*.glClear(GL20.***GL\_COLOR\_BUFFER\_BIT***);  
 **gsm**.update(Gdx.*graphics*.getDeltaTime());  
 **gsm**.render(**batch**);  
 **batch**.begin();  
 **FontBlack1**.draw(**batch**, **"Avtor: Zadoroznii Vladislav"**, 10, 20);  
 **batch**.end();  
 }  
  
 @Override  
 **public void** dispose() {  
 **super**.dispose();  
 **music**.dispose();  
 }  
}

**public class** GameOver **extends** State {  
  
 **private** Texture **background**;  
 **private** Texture **gameover**;  
  
 **public** GameOver(GameState gsm) {  
 **super**(gsm);  
 **camera**.setToOrtho(**false**, FlappyGame.***WIDTH*** / 2, FlappyGame.***HEIGHT*** / 2);  
 **background** = **new** Texture(**"back3.png"**);  
 **gameover** = **new** Texture(**"GameOver2.png"**);  
 }  
  
 @Override  
 **public void** handleInput() {  
 **if** (Gdx.*input*.justTouched()) {  
 **gsm**.set(**new** PlayState(**gsm**));  
 }  
  
 }  
  
 @Override  
 **public void** update(**float** dt) {  
 handleInput();  
  
 }  
  
 @Override  
 **public void** render(SpriteBatch sb) {  
 sb.setProjectionMatrix(**camera**.**combined**);  
 sb.begin();  
 sb.draw(**background**, 0, 0);  
 sb.draw(**gameover**, **camera**.**position**.**x** - **gameover**.getWidth() / 2, **camera**.**position**.**y**);  
 sb.end();  
  
 }  
  
 @Override  
 **public void** dispose() {  
 **background**.dispose();  
 **gameover**.dispose();  
  
 System.***out***.println(**"GameOver"**);  
  
 }  
}

**public class** GameState {  
  
 **private** Stack<State> **states**;  
  
 **public** GameState(){  
 **states** = **new** Stack<State>();  
 }  
  
 **public void** push(State state){  
 **states**.push(state);  
 }  
  
 **public void** pop(){  
 **states**.pop().dispose();  
 }  
  
 **public void** set(State state){  
 **states**.pop().dispose();  
 **states**.push(state);  
 }  
  
 **public void** update(**float** dt){  
 **states**.peek().update(dt);  
 }  
  
 **public void** render(SpriteBatch sb){  
 **states**.peek().render(sb);  
 }  
}

**public class** MenuState **extends** State {  
  
 **private** Texture **background**;  
 **private** Texture **playBtn**;  
  
 **public** MenuState(GameState gsm) {  
 **super**(gsm);  
 **background** = **new** Texture(**"back3.png"**);  
 **playBtn** = **new** Texture(**"Play2.png"**);  
 }  
  
 @Override  
 **public void** handleInput() {  
 **if** (Gdx.*input*.justTouched()) {  
 **gsm**.set(**new** PlayState(**gsm**));  
 }  
  
 }  
  
 @Override  
 **public void** update(**float** dt) {  
 handleInput();  
  
 }  
  
 @Override  
 **public void** render(SpriteBatch sb) {  
 sb.begin();  
 sb.draw(**background**, 0, 0, FlappyGame.***WIDTH***, FlappyGame.***HEIGHT***);  
 sb.draw(**playBtn**, (FlappyGame.***WIDTH*** / 2) - (**playBtn**.getWidth() / 2), FlappyGame.***HEIGHT*** / 2);  
 sb.end();  
  
 }  
  
 @Override  
 **public void** dispose() {  
 **background**.dispose();  
 **playBtn**.dispose();  
  
 System.***out***.println(**"MenuState Disposed"**);  
 }  
}

**public class** PlayState **extends** State {  
  
 **private static final int *TUBE\_SPACING*** = 125;  
 **private static final int *TUBE\_COUNT*** = 6;  
 **private static final int *GROUND\_Y\_OFFSET*** = -30;  
  
 **private** Bird **bird**;  
 **private** BedBird **birdBed**;  
 **private** Texture **bg**;  
 **private** Texture **ground**;  
 **private** Vector2 **groundPos1**, **groundPos2**;  
 **private** Array<Tube> **tubes**;  
 **private** BitmapFont **FontWhite2**;  
  
  
  
 **public** PlayState(GameState gsm) {  
 **super**(gsm);  
 **bird** = **new** Bird(-200, 200);  
 **birdBed** = **new** BedBird(-700, 400);  
 **camera**.setToOrtho(**false**, FlappyGame.***WIDTH***, FlappyGame.***HEIGHT***);  
 **bg** = **new** Texture(**"back3.png"**);  
 **ground** = **new** Texture(**"ground.png"**);  
 **groundPos1** = **new** Vector2(**camera**.**position**.**x** - **camera**.**viewportWidth** / 2, ***GROUND\_Y\_OFFSET***);  
 **groundPos2** = **new** Vector2((**camera**.**position**.**x** - **camera**.**viewportWidth** / 2) + **ground**.getWidth(), ***GROUND\_Y\_OFFSET***);  
 **tubes** = **new** Array<Tube>();  
  
  
 **FontWhite2** = **new** BitmapFont();  
 **FontWhite2**.setColor(Color.***WHITE***);  
  
 **for** (**int** i = 0; i < ***TUBE\_COUNT***; i++) {  
 **tubes**.add(**new** Tube(i \* (***TUBE\_SPACING*** + Tube.***TUBE\_WIDTH***)));  
 }  
 }  
  
 @Override  
 **protected void** handleInput() {  
 **if** (Gdx.*input*.justTouched())  
 **bird**.jump();  
 }  
  
  
  
 @Override  
 **public void** update(**float** dt) {  
 handleInput();  
 updateGround();  
 **bird**.update(dt);  
 **birdBed**.update(dt);  
 **camera**.**position**.**x** = **bird**.getPosition().**x** + 80;  
 **for** (**int** i = 0; i < **tubes**.**size**; i++) {  
  
 Tube tube = **tubes**.get(i);  
  
 **if** (**camera**.**position**.**x** - (**camera**.**viewportWidth** / 2) > tube.getPosTopTube().**x** + tube.getTopTube().getWidth()) {  
 tube.reposition(tube.getPosTopTube().**x** + ((Tube.***TUBE\_WIDTH*** + ***TUBE\_SPACING***) \* ***TUBE\_COUNT***));  
 }  
  
 **if** (tube.collides(**bird**.getBounds()))  
 **gsm**.set(**new** GameOver(**gsm**));  
 }  
 **camera**.update();  
  
 }  
  
 @Override  
 **public void** render(SpriteBatch sb) {  
 sb.setProjectionMatrix(**camera**.**combined**);  
 sb.begin();  
 Gdx.*graphics*.getDeltaTime();  
 sb.draw(**bg**, **camera**.**position**.**x** - (**camera**.**viewportWidth** / 2), 0);  
 sb.draw(**bird**.getBird(), **bird**.getPosition().**x**, **bird**.getPosition().**y**);  
 sb.draw(**birdBed**.getBedBird(),(**camera**.**position**.**x** - 120) - (**camera**.**viewportWidth** / 2), 70);  
 **for** (Tube tube : **tubes**) {  
 sb.draw(tube.getTopTube(), tube.getPosBotTube().**x**, tube.getPosTopTube().**y**);  
 sb.draw(tube.getBottomTube(), tube.getPosBotTube().**x**, tube.getPosBotTube().**y**);  
 }  
 sb.draw(**ground**, **groundPos1**.**x**, **groundPos1**.**y**);  
 sb.draw(**ground**, **groundPos2**.**x**, **groundPos2**.**y**);  
  
 sb.end();  
  
 }  
  
 @Override  
 **public void** dispose() {  
 **bg**.dispose();  
 **bird**.dispose();  
 **birdBed**.dispose();  
 **ground**.dispose();  
 **for** (Tube tube : **tubes**)  
 tube.dispose();  
 System.***out***.println(**"PlayState Disposed"**);  
  
 }  
  
 **private void** updateGround() {  
 **if** (**camera**.**position**.**x** - (**camera**.**viewportWidth** / 2) > **groundPos1**.**x** + **ground**.getWidth())  
 **groundPos1**.add(**ground**.getWidth() \* 2, 0);  
 **if** (**camera**.**position**.**x** - (**camera**.**viewportWidth** / 2) > **groundPos2**.**x** + **ground**.getWidth())  
 **groundPos2**.add(**ground**.getWidth() \* 2, 0);  
 }  
}

**public abstract class** State {  
  
 **protected** OrthographicCamera **camera**;  
 **protected** Vector3 **mouse**;  
 **protected** GameState **gsm**;  
  
 **public** State(GameState gsm){  
 **this**.**gsm** = gsm;  
 **camera** = **new** OrthographicCamera();  
 **mouse** = **new** Vector3();  
 }  
  
  
 **protected abstract void** handleInput();  
 **public abstract void** update(**float** dt);  
 **public abstract void** render(SpriteBatch sb);  
 **public abstract void** dispose();  
}

**public class** Tube {  
  
 **public static final int *TUBE\_WIDTH*** = 52;  
  
 **private static final int *FLUCTUATION*** = 130;  
 **private static final int *TUBE\_GAP*** = 115;  
 **private static final int *LOWEST\_OPENING*** = 120;  
  
 **private** Texture **topTube**, **bottomTube**;  
 **private** Vector2 **posTopTube**, **posBotTube**;  
 **private** Random **rand**;  
 **private** Rectangle **boundsTop**, **boundsBot**;  
  
  
 **public** Texture getTopTube() {  
 **return topTube**;  
 }  
  
 **public** Texture getBottomTube() {  
 **return bottomTube**;  
 }  
  
 **public** Vector2 getPosTopTube() {  
 **return posTopTube**;  
 }  
  
 **public** Vector2 getPosBotTube() {  
 **return posBotTube**;  
 }  
  
 **public** Tube(**float** x) {  
 **topTube** = **new** Texture(**"Err.png"**);  
 **bottomTube** = **new** Texture(**"Err.png"**);  
 **rand** = **new** Random();  
  
 **posTopTube** = **new** Vector2(x, **rand**.nextInt(***FLUCTUATION***) + ***TUBE\_GAP*** + ***LOWEST\_OPENING***);  
 **posBotTube** = **new** Vector2(x, **posTopTube**.**y** - ***TUBE\_GAP*** - **bottomTube**.getHeight());  
  
 **boundsTop** = **new** Rectangle(**posTopTube**.**x**, **posTopTube**.**y**, **topTube**.getWidth(), **topTube**.getHeight());  
 **boundsBot** = **new** Rectangle(**posBotTube**.**x**, **posBotTube**.**y**, **bottomTube**.getWidth(), **bottomTube**.getHeight());  
  
 }  
  
 **public void** reposition(**float** x) {  
 **posTopTube**.set(x, **rand**.nextInt(***FLUCTUATION***) + ***TUBE\_GAP*** + ***LOWEST\_OPENING***);  
 **posBotTube**.set(x, **posTopTube**.**y** - ***TUBE\_GAP*** - **bottomTube**.getHeight());  
 **boundsTop**.setPosition(**posTopTube**.**x**, **posTopTube**.**y**);  
 **boundsBot**.setPosition(**posBotTube**.**x**, **posBotTube**.**y**);  
 }  
  
 **public boolean** collides(Rectangle player) {  
 **return** player.overlaps(**boundsTop**) || player.overlaps(**boundsBot**);  
 }  
  
 **public void** dispose() {  
 **topTube**.dispose();  
 **bottomTube**.dispose();  
 }  
}