

УТВЕРЖДАЮ

Руководитель практики от Университета

Крутиков А. К. Преподаватель каф. ЭВМ *(должность, ФИО)*

МП \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(Подпись)*

Дата « \_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 год

**ОТЧЕТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ**

ПМ.05 Проектирование и разработка информационных систем

|  |  |
| --- | --- |
| Тюлькин Данила Алексеевич | |
| *(Ф.И.О. обучающегося)* | |
| 09.02.07 Информационные системы и программирование | |
| *(специальность)* | |
|  | |
| Учебная группа | ИСПк-201-51-00 |
|  |  |
| Место прохождения практики | ФГБОУ ВО «Вятский государственный университет», |
| Колледж ВятГУ | *(наименование организации, структурного подразделения организации)* |
|  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Итоговая оценка: |  | | | | |
| Руководитель  практики от университета |  |  |  |  |  |
|  | *(дата)* |  | *(подпись)* |  | *(Ф.И.О.)* |

Киров, 2021 г.



ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА ПРАКТИКУ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ф.И.О. обучающегося | | | Тюлькин Данила Алексеевич | | | | | | |
| Специальность | | | 09.02.07 Информационные системы и программирование | | | | | | |
| Учебная группа | | | ИСПк-201-51-00 | | | | | | |
| Вид практики | | | учебная практика | | | | | | |
| Сроки прохождения практики с | | | | 11.01.2021 | по | 16.06.2021 | | | |
| Место прохождения практики | | | | ФГБОУ ВО «Вятский государственный университет»,  Колледж ВятГУ | | | | | |
|  | | (наименование организации, структурного подразделения организации) | | | | | | |
| № | Виды работ, выполняемых обучающимися во время практики | | | | | | Объем работ (час) | Формируемые компетенции | | |
| 1 | Пройти инструктаж по ознакомлению с правилами внутреннего трудового распорядка, охраны труда, техники безопасности, противопожарной безопасности, санитарно-эпидемиологическими правилами и гигиеническими нормативами, а также вводный инструктаж и инструктаж на рабочем месте | | | | | | 2 | - | | |
| 2 | Постановка задачи | | | | | | 8 | ОК2, ОК6, ОК7, ОК9 | | |
| 3 | Настройка рабочего окружения | | | | | | 8 | ОК1, ОК8, ОК9 | | |
| 4 | Анализ предметной области, постановка расширенного технического задания | | | | | | 10 | ПК5.1, ОК3, ОК4 | | |
| 5 | Проектирование | | | | | | 35 | ПК5.2, ПК5.3, | | |
| 6 | Реализация программного кода | | | | | | 35 | ПК5.4, ПК5.8, | | |
| 7 | Тестирование и отладка полученного кода | | | | | | 22 | ПК5.5, ПК6.4, ОК2, ОК9 | | |
| 8 | Подготовка отчетной документации, включая детальное описание разработанных и использованных алгоритмов | | | | | | 10 | ПК5.6, ПК5.7, ОК9, ОК10, ОК11 | | |
| 9 | Подготовка к сдаче и сдача промежуточной аттестации | | | | | | 2 | ОК4, ОК5 | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Индивидуальное задание на практику разработано в соответствии с рабочей программой практики. | | | | | |
| Руководитель  практики от университета |  |  |  |  | Крутиков А. К |
|  | (дата) |  | (подпись) |  | (Ф.И.О.) |

|  |  |
| --- | --- |
| С индивидуальным заданием ознакомлен(а) |  |
|  | (дата, подпись обучающегося) |

**ОТЗЫВ РУКОВОДИТЕЛЯ ПРАКТИКИ**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ф.И.О. обучающегося | | Тюлькин Данила Алексеевич | | | |
| Специальность | | 09.02.07 Информационные системы и программирование | | | |
| Учебная группа | | ИСПк-201-51-00 | | | |
| Вид практики | | учебная практика | | | |
| Сроки прохождения практики с | | | 11.01.2021 | по | 16.06.2021 |
| Место прохождения практики | | | ФГБОУ ВО «Вятский государственный университет», Колледж ВятГУ | | |
|  | (наименование организации, структурного подразделения организации) | | | | |

ВИДЫ И КАЧЕСТВО ВЫПОЛНЕНИЯ РАБОТ

|  |  |  |  |
| --- | --- | --- | --- |
| Вид работ | Критерий выполнения работ | | |
| Выполнены полностью самостоятельно | Выполнены с незначительной помощью наставника | Выполнены с помощью наставника |
| Постановка задачи | V |  |  |
| Настройка рабочего окружения | V |  |  |
| Анализ предметной области, постановка расширенного технического задания | V |  |  |
| Проектирование | V |  |  |
| Реализация программного кода | V |  |  |
| Тестирование и отладка полученного кода | V |  |  |
| Подготовка отчетной документации, включая детальное описание разработанных и использованных алгоритмов | V |  |  |
| Подготовка к сдаче и сдача промежуточной аттестации | V |  |  |

Обучающийся ознакомлен с правилами внутреннего трудового распорядка, охраны труда, техники безопасности, противопожарной безопасности, санитарно-эпидемиологическими правилами и гигиеническими нормативами, а также прошел вводный инструктаж и инструктаж на рабочем месте.

Во время прохождения учебной практики обучающимся освоены следующие профессиональные и общие компетенции:

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование компетенции | Показатели оценки | Оценка | |
| Освоена | Не освоена |
| ПК 5.1. Собирать исходные данные для разработки проектной документации на информационную систему. | Способен выполнять анализ предметной области, выявляя существенные элементы, оказывающие влияние на проектируемую систему | V |  |
| ПК 5.2. Разрабатывать проектную документацию на разработку информационной системы в соответствии с требованиями заказчика. | Способен выполнять декомпозицию сущностей с целью получения наиболее полной картины о целесообразной структуре разработки | V |  |
| ПК 5.3. Разрабатывать подсистемы безопасной информационной системы в соответствии с техническим заданием. | Способен прогнозировать потенциально некорректные действия пользователя и предусматривать соответствующие реакции со стороны системы | V |  |
| ПК 5.4. Производить разработку модулей информационной системы в соответствии с техническим заданием. | Способен создавать программный код, отвечающий предъявляемым требованиям | V |  |
| ПК 5.5. Осуществлять тестирование информационной системы на этапе опытной эксплуатации с фиксацией выявленных ошибок кодирования в разрабатываемых модулях информационной системы. | Способен выполнять оценку корректности функционирования системы | V |  |
| ПК 5.6. Разрабатывать техническую документацию на эксплуатацию информационной системы. | Способен разрабатывать проектную, техническую и пользовательскую документации | V |  |
| ПК 5.7. Производить оценку информационной системы для выявления возможности ее модернизации. | Способен оценивать перспективы дальнейшего развития программной системы | V |  |
| ПК 5.8. Осуществлять интеграцию информационной системы с другими программными продуктами. | Способен повторно использовать готовые шаблонные решения при разработке программного продукта | V |  |
| ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам. | Способен анализировать специализированную информацию и находить оптимальные пути решения задач профессиональной деятельности | V |  |
| ОК 02. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности. | Способен анализировать специализированную информацию и находить оптимальные пути решения задач профессиональной деятельности | V |  |
| ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие. | Способен находить пути улучшения имеющихся решений, позволяющих повысить их общий качественный уровень | V |  |
| ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами. | Способен конструктивно обмениваться информацией с коллегами, грамотно формулировать запросы в целях получения разъясняющей информации | V |  |
| ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста. | Способен конструктивно обмениваться информацией с коллегами, грамотно формулировать запросы в целях получения разъясняющей информации | V |  |
| ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей. | Способен конструктивно обмениваться информацией с коллегами, грамотно формулировать запросы в целях получения разъясняющей информации | V |  |
| ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях. | Способен эффективно использовать компьютерное время, а также материальные ресурсы, необходимые для решения поставленных задач | V |  |
| ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности. | Способен соблюдать требования внутреннего трудового распорядка организации, охраны труда и техники безопасности в целях сохранения собственного здоровья | V |  |
| ОК 09. Использовать информационные технологии в профессиональной деятельности. | Способен применять стек современных средств разработки ПО для решения задач профессиональной деятельности в заданном контексте | V |  |
| ОК 10. Пользоваться профессиональной документацией на государственном и иностранном языке. | Способен использовать в своей работе специализированную документацию | V |  |
| ОК 11. Планировать предпринимательскую деятельность в профессиональной сфере. | Способен проектировать алгоритмические решения, принимая во внимание имеющиеся ресурсные ограничения | V |  |

**Краткая характеристика работы обучающегося**

|  |
| --- |
| Программа практики выполнена в полном объеме. Все виды работ выполнялись в срок, |
| без существенных замечаний. В достаточной степени была проявлена самостоятельность |
| и умение грамотно пользоваться источниками информации, находящимися в свободном |
| доступе. |

|  |  |  |
| --- | --- | --- |
|  |  | Руководитель практики  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись ФИО  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (должность)  Дата «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_ год |

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc74498144)

[**1. СВЕДЕНИЯ О РАБОТЕ, ВЫПОЛНЕННОЙ В ПЕРИОД ПРОХОЖДЕНИЯ УЧЕБНОЙ ПРАКТИКИ** 4](#_Toc74498145)

[**2. НАСТРОЙКА РАБОЧЕГО ОКРУЖЕНИЯ** 6](#_Toc74498146)

[**3. ФОРМУЛИРОВКА ИНДИВИДУАЛЬНОГО ЗАДАНИЯ** 8](#_Toc74498147)

[**4. ОПИСАНИЕ ВЫПОЛНЕНИЯ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ** 9](#_Toc74498148)

[**ЗАКЛЮЧЕНИЕ** 23](#_Toc74498149)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 24](#_Toc74498150)

[**ПРИЛОЖЕНИЕ А** 25](#_Toc74498151)

# **ВВЕДЕНИЕ**

Учебная практика ПМ.05 проходила на базе Колледжа ФГБОУ ВО «Вятский государственный университет» в период с 11.01.2021 г. по 16.06.2021 г. по понедельникам.

Цель практики: сформировать у обучающихся представление о процессе разработки прикладного программного обеспечения.

Задачи практики:

– дать представление о различных методах проектирования прикладного программного обеспечения;

– сформировать навыки написания документации, требуемой при разработке прикладного программного обеспечения;

– дать представление о различных инструментах, используемых при разработке прикладного программного обеспечения.

# **1. СВЕДЕНИЯ О РАБОТЕ, ВЫПОЛНЕННОЙ В ПЕРИОД ПРОХОЖДЕНИЯ УЧЕБНОЙ ПРАКТИКИ**

В период с 11.01.2021 по 16.06.2021 при прохождении учебной практики ПМ.05 на базе Колледжа ФГОБУ ВО «Вятский государственный университет» был выполнен следующий перечень работ, представленный в таблице 1.

Таблица 1 – Сведения о работе, выполненной в период практики

| Дата | Краткое содержание выполненных работ |
| --- | --- |
| 11.01.2021 | Ознакомление с правилами внутреннего трудового распорядка, охраны труда, техники безопасности, противопожарной безопасности, санитарно-эпидемиологическими правилами и гигиеническими нормативами, а также вводный инструктаж и инструктаж на рабочем месте |
| 12.01.2021-25.01.2021 | Постановка задачи |
| 26.01.2021-08.02.2021 | Настройка рабочего окружения |
| 09.02.2021-22.02.2021 | Анализ предметной области, постановка расширенного технического задания |
| 23.02.2021-29.03.2021 | Проектирование |
| 30.03.2021-03.05.2021 | Реализация программного кода |
| 04.05.2021-24.05.2021 | Тестирование и отладка полученного кода |
| 25.05.2021-13.06.2021 | Подготовка отчетной документации, включая детальное описание разработанных и использованных алгоритмов |
| 14.06.2021-16.06.2021 | Подготовка к сдаче и сдача промежуточной аттестации |

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

(дата) (подпись)

# **2. НАСТРОЙКА РАБОЧЕГО ОКРУЖЕНИЯ**

Для написания кода для фрагмента игрового приложения был использован язык программирования C#. Само приложение было сделано на игровом движке Unity. Выбор пал на это язык программирования, т.к. он достаточно прост в освоении и имеет большой потенциал для дальнейшего развития в направлении разработки игровых приложений.



Рисунок 1 – Среда разработки

Особенности Unity:

Unity позволяет разрабатывать игры без специальных навыков для этого. Разработчику нужно создать объект, а затем создать средства, в которых он будет существовать.  Программа дает возможность создавать и сразу расставлять объекты в реальном времени, при этом можно протестировать результат.

Второе преимущество движка – наличие огромной библиотеки ассетов и плагинов, которые не только бесплатные, но и с их помощью которых можно значительно ускорить процесс разработки игры.

Третья сильная сторона Unity 3D – поддержка огромного количества платформ, технологий, API. Созданные на движке игры можно легко портировать между ОС Windows, Linux, OS X, Android, iOS, на консоли семейств PlayStation, Xbox, Nintendo, на VR- и AR-устройства.

Недостатки:

1. Если разработчики занимаются разработкой сложных программных продуктов, то скорее всего придется высокопрофессионального специалиста по C#. Его задачей станет написание скриптов, внедрение их в продукт.
2. Следующая слабая сторона — неспешность. Формирование сложных игровых сцен, может оказать отрицательное влияние на эффективность продукта. В итоге разработчики должны понести расходы и времен, и ресурсов для усовершенствования отдельных элементов, а возможно и их удаления.
3. Продукты разработанные на UNITY3D отличаются большим «весом», даже не сложная игра может занимать несколько сот мегабайт места. Если для настольного компьютера это не столь и важно, но, если продукт готовится для мобильных устройств, но разработчики должны уделить внимание оптимизации его объема.

# **3. ФОРМУЛИРОВКА ИНДИВИДУАЛЬНОГО ЗАДАНИЯ**

В соответствии с индивидуальным заданием на учебную практику необходимо:

Создать фрагмент игрового приложения и написать исходный код. В приложении должны быть основные механики (Передвижение вправо и влево, прыжок). А также создать общий визуальный стиль для игрового поля и персонажа.

Кроме этого, требуется создать документацию: технического задания, руководства, анализа предметной области, обзора аналогов, структуры приложения, алгоритмов функционирования и само программное обеспечение.

# **4. ОПИСАНИЕ ВЫПОЛНЕНИЯ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ**

**4.1 Анализ предметной области и обзор аналогов**

Благодаря развитию рынка инди-игр, многие разработчики компьютерных игр получили возможность работать над своими игровыми проектами без финансовых и юридических обязательств перед компаниями-издателями. Инди-игры (англ. Indie games, от англ. independent video games -- «независимые компьютерные игры») -- это компьютерные игры, созданные отдельными разработчиками или небольшими коллективами без финансовой поддержки издателя компьютерных игр. Распространение осуществляется посредством каналов цифровой дистрибуции. Масштаб явлений, связанных с инди-играми, ощутимо возрастает со второй половины 2000-х годов, в основном ввиду развития новых способов онлайн-дистрибуции и средств разработки.

Данный проект относится к инди-разработке и развивается только за счет средств его разработчиков. Благодаря развитию ПО разработки компьютерных игр, команде разработчиков не требуется тратить несколько лет на разработку игрового движка. Это позволяет сразу же приступать к непосредственной работе над игровым проектом и значительно сокращает время его разработки.

Платформер — жанр [компьютерных игр](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0), в которых основу игрового процесса составляют прыжки по платформам, лазанье по лестницам, сбор предметов, необходимых для победы над врагами или завершения уровня.

Многие игры подобного жанра характеризуются нереалистичностью, рисованной мультяшной графикой. Персонажами таких игр часто бывают вымышленные существа

Обзор аналогов:

Одной из первых игр рассмотрим Crash Bandicoot - видеоигра в жанре платформер, созданная компанией Naughty Dog в 1996 году на PlayStation. В игре присутствует три острова, каждый из которых разделён на множество уровней. Нашей задачей становится пройти их, управляя рыжим бандикутом по имени Крэш. Игроку даётся определённое количество жизней, теряющихся при смерти от врага или падения в яму или реку (Бандикут не умеет плавать). Когда все жизни истрачены, появляется окно с надписью "Game Over". Нам даётся выбор между продолжением игры с последнего проигранного уровня или её окончанием с последующим выходом в главное меню. Расклад событий зависит от того, что мы выбирем: "да" или "нет".

Прыжок и верчение, кроме того, могут послужить и для разрушения многочисленных ящиков, разбросанных по всем уровням. Большинство таких ящиков содержат вампа-фрукты, именуемые в народе просто яблочками. Если собрать 100 таких яблочек, то будет получена одна дополнительная жизнь.

Достоинства:

1)Разнообразные уровни игры

2)Красивая графика

3)Разнообразный (в рамках жанра) игровой процесс

Недостатки:

1)Сложность в производстве

2) Более высокие, в сравнении с 2д платформерами, системные требования

3)Отсутствие повышение навыков главного героя

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Интерфейс Crash bandicoot.

Еще один аналог – это Spyro. Spyro — что-то среднее между Crash Bandicoot и Mario 64: здесь тоже есть центральный мир-хаб, из которого игрок попадает на разные уровни. Но левелдизайн не такой прямолинейный, как в платформере про бандикута: в Spyro, особенно по меркам 98-го года, были довольно большие уровни, поощрявшие исследование. Это ещё не открытый мир, но уже и не тоннели с препятствиями.

Способность дракона летать заметно повлияла на геймдизайн: большое количество возвышенностей делало уровни вертикальными, отчего геймплей в Spyro никогда не протекал в одной плоскости.

У каждого героя платформера должен быть свой уникальный способ расправляться с врагами: Крэш умел быстро крутиться, Марио прыгал противникам на головы. А у Спайро таких способностей было аж две: он мог разгоняться, бодая противников рогами, и дышать огнём.

Всё это делало Spyro the Dragon уникальным 3D-платформером, в котором хороший по меркам 98-го геймдизайн сочетался с яркой картинкой и запоминающимся саундтреком.

Достоинства:

1)Разнообразные противники

2)Открытый мир

3) Использование в программе средства из 3D графики, что делает интерфейс игры более привлекательным, по сравнению с ранними играми, в данном жанре, таких как серия Mario.

Недостатки:

1)Недоработанный дизайн уровней и врагов, что может вызывать сбои в работе игры, а также портить впечатление играющих

2) Неудобное управление

Изображение выглядит как трава, внешний, цветной

Автоматически созданное описание

# Рисунок 2 – Интерфейс программы Spyro Reignited Trilogy.

**4.2 Разработка структуры приложения и алгоритмов функционирования**

Приложение состоит из одного интерфейса, что позволит гораздо быстрее освоиться новому пользователю.

Функции:

1. Передвижение вправо:

При нажатии кнопки которая специализируется за передвижение игрового персонажа вправо, будет выполнено данное действие, если на пути игрока не будет преград.

1. Передвижение влево:

При нажатии кнопки которая специализируется за передвижение игрового персонажа влево, будет выполнено данное действие, если на пути игрока не будет преград.

1. Передвижение вперёд:

При нажатии кнопки которая специализируется за передвижение игрового персонажа вперёд, будет выполнено данное действие, если на пути игрока не будет преград.

1. Передвижение назад:

При нажатии кнопки которая специализируется за передвижение игрового персонажа назад, будет выполнено данное действие, если на пути игрока не будет преград.

1. Функция прыжок:

При нажатии клавиши, отвечающей за прыжок, происходит действие, если на пути игрока не находиться преград

Структура приложения в виде блок-схем

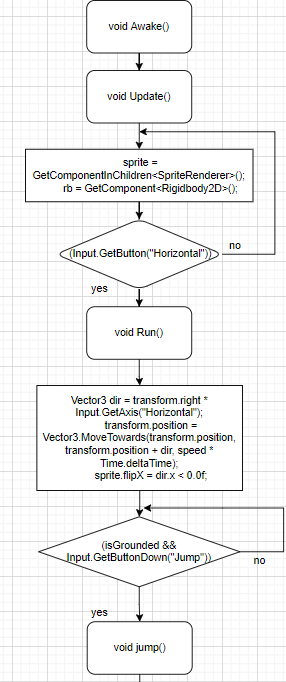


Рисунок 3 – Алгоритм программы игрового персонажа

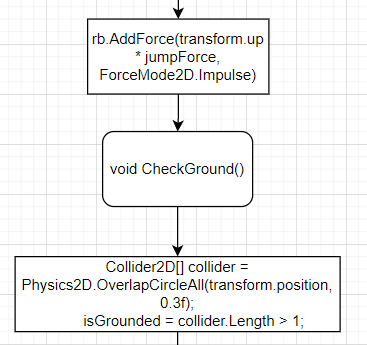


Рисунок 4 - Алгоритм программы игрового персонажа

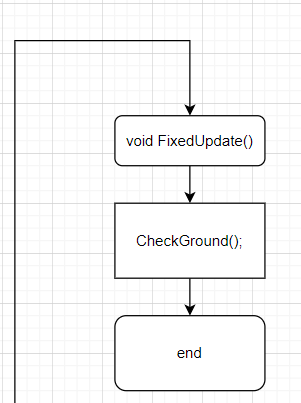


Рисунок 5 - Алгоритм программы игрового персонажа

**4.3 Проектирование программного обеспечения**

При нажатии на кнопку «W» персонаж начнет двигаться вперёд.

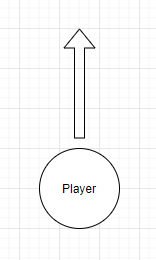


Рисунок 6 – Передвижение персонажа вправо

При нажатии на кнопку «A» персонаж начнет двигаться влево.

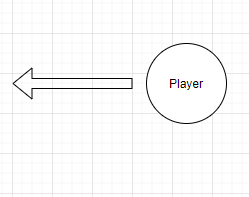


Рисунок 7 – Передвижение персонажа влево

При нажатии на кнопку «S» персонаж начнет двигаться назад.

Изображение выглядит как седзи

Автоматически созданное описание

Рисунок 8 – Передвижение персонажа вправо

При нажатии на кнопку «D» персонаж начнет двигаться вправо.

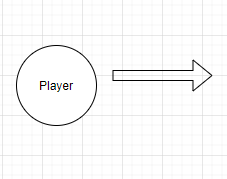


Рисунок 9 – Передвижение персонажа вправо

При нажатии на кнопку «Space» персонаж прыгнет.

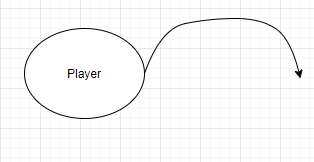


Рисунок 10 – Прыжок персонажа

Разработка программного обеспечения

Программное обеспечение разработано на языке C# с использованием библиотек System.Collections, System.Collections.Generic, UnityEngine.

Установка переменной для проверки контакта с поверхностью и тело объекта

Character control

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

[\\Подключение](file:///\\Подключение) библиотек Unity для работы с интерфейсом, объектами, сценами, внутриигровой физикой.

public class CharacterControl : MonoBehaviour

MonoBehaviour это **это** **базовый** **класс, из** **которого** **извлекается** **каждый** **сценарий** **Unity.**

{

public float speed = 4.0f;// скорость персонажа

public float jumpSpeed = 10.0f;// скорость прыжка персонажа

public float gravitation = 10.0f;// гравитация для персонажа

private Vector3 moveDir = Vector3.zero;// движение персонажа

private CharacterController controller;// компонент CharacterController

public int coins;

public Text TextCoins;

public int res;

Объявление переменных

// Start is called before the first frame update

void Start()

{

controller = GetComponent<CharacterController>();

}

Функция void Start выполняет заданный внутри себя код при загрузке сцены

// Update is called once per frame

void FixedUpdate()

{

if (controller.isGrounded)

{

moveDir = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));

moveDir = transform.TransformDirection(moveDir);

moveDir \*= speed;

}

if (Input.GetKeyDown(KeyCode.Space) && controller.isGrounded)

{

moveDir.y = jumpSpeed;

}

moveDir.y -= gravitation \* Time.deltaTime;

controller.Move(moveDir \* Time.deltaTime);

}

private void OnTriggerEnter(Collider col)

{

if (col.gameObject.tag == "Money")

{

coins++;

col.gameObject.SetActive(false);

TextCoins.text = coins.ToString();

}

if (TextCoins.text == "10")

{

SceneManager.LoadScene(2);

}

}

}

Функция FixedUpdate() выполняет код с каждым обновлением кадра

Controller.isGrounded – булевая функция, проверяющая, коснулся ли персонаж земли

Возвращает значение по axisName виртуальной оси.

Для ввода с клавиатуры или джойстика значение будет лежать в диапазоне -1...1.

transform.TransformDirection() – эта операция изменяет параметр x y z объекта с локальных на заданные мировые координаты

Input.GetKeyDown() – булевая функция, проверяющая нажатие клавиши на физической клавиатуре

Col.gameObject.tag – обращение к игровому объекту по тегу

Col.gameObjects.SetActive() – определяет загружен игровой объект на сцену или нет

TextCoins.text = coins.ToString() – выводит значение переменной coins в текстовую строку

Scenes

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Scenes : MonoBehaviour

{

public void ChangeScenes(int numberScenes)

{

SceneManager.LoadScene(numberScenes);

}

public void Exit()

{

Application.Quit();

}

}

Respawn

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Respawn : MonoBehaviour

{

[SerializeField] private Transform Player;

[SerializeField] private Transform respawnPoint;

void OnTriggerEnter(Collider other)

{

Player.transform.position = respawnPoint.transform.position;

}

}

Menu paused

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class MenuPaused : MonoBehaviour

{

public GameObject menuPaused;

[SerializeField] KeyCode keyMenuPaused;

bool isMenuPaused = false;

private void Update()

{

ActiveMenu();

}

void ActiveMenu() {

if (Input.GetKeyDown(keyMenuPaused))

{

isMenuPaused = !isMenuPaused;

}

if (isMenuPaused)

{

menuPaused.SetActive(true);

Time.timeScale = 0f;

}

else

{

menuPaused.SetActive(false);

Time.timeScale = 1f;

}

}

}

Mouse look

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class MouseLook : MonoBehaviour

{

public enum RorationAxes

{

XandY,

X,

Y

}

public RorationAxes \_axes = RorationAxes.XandY;

public float \_rotationSpeedHor = 5.0f;

public float \_rotationSpeedVer = 5.0f;

public float maxVert = 46.0f;

public float minVert = -45.0f;

private float \_rotationX = 0;

private bool isMoving;

private void Update()

{

//Проверим ось движения

if (\_axes == RorationAxes.XandY)

{

\_rotationX -= Input.GetAxis("Mouse Y") \* \_rotationSpeedVer;

\_rotationX = Mathf.Clamp(\_rotationX, minVert, maxVert);

float delta = Input.GetAxis("Mouse X") \* \_rotationSpeedHor;

float \_rotationY = transform.localEulerAngles.y + delta;

transform.localEulerAngles = new Vector3(\_rotationX, \_rotationY, 0);

}

else if (\_axes == RorationAxes.X)

{

transform.Rotate(0, Input.GetAxis("Mouse X") \* \_rotationSpeedHor, 0);

}

else if (\_axes == RorationAxes.Y)

{

\_rotationX -= Input.GetAxis("Mouse Y") \* \_rotationSpeedVer;

\_rotationX = Mathf.Clamp(\_rotationX, minVert, maxVert);

float \_rotationY = transform.localEulerAngles.y;

transform.localEulerAngles = new Vector3(\_rotationX, \_rotationY, 0);

}

}

public void SetIsMovimg(bool new\_state)

{

isMoving = new\_state;

}

}

Рисунок 11 – Исходный код функции Start

Установка функций на корректную работу с физикой

Рисунок 12 – Исходный код функции FixedUpdate

Установка направлений движения

Рисунок 13 – Исходный код функции MovementLogic()

Установка прыжка

Рисунок 14 – Исходный код функции JumpLogic

Рисунок 15 – Исходный код обработки столкновений

**4.4 Тестирование программного обеспечения**

В процессе проведения приёмочных испытаний должны быть протестированы следующие подсистемы “ChillJump”:

* Движение главного героя
* Подсистема исполнения управления;

Все подсистемы испытываются одновременно на корректность взаимодействия подсистем, влияние подсистем друг на друга, т.е. испытания проводятся комплексно.

Приемочные испытания включают проверку:

* полноты и качества реализации функций, указанных в ТЗ;
* выполнения каждого требования, относящегося к интерфейсу Системы;
* работы пользователей в диалоговом режиме;
* полноты действий, доступных пользователю, и их достаточность для функционирования фрагмента игрового приложения;
* сложности процедур диалога, возможности работы пользователей без специальной подготовки;
* реакции системы на ошибки пользователя;
* практической выполнимости рекомендованных процедур.

Испытания подсистем управления и исполнения направлены на проверку корректного отзыва персонажа на действия, которые совершает пользователь.

Требования к Подсистеме управления приведены в Техническом задании.

Проверяются процедуры и параметры:

* Корректное перемещения персонажа
* Правильная работа кнопок

**Таблица 1 – Общие сведения**

|  |  |  |
| --- | --- | --- |
| № | Действие | Результат |
| 1 | **Игровой процесс приложения «ChillJump»** | |
| 1а | Вход в программу | Вход в программу |
| 1б | Нажатие на кнопку “W” | Персонаж движется вперёд |
| 1в | Нажатие на кнопку “A” | Персонаж движется влево |
| 1г | Нажатие на кнопку “S” | Персонаж движется назад |
| 1д | Нажатие на кнопку “D” | Персонаж движется вправо |
| 1е | Нажатие на кнопку «Space» | Персонаж прыгает вверх |
| **2** | **Проверка работоспособности конца фрагмента игрового приложения “ChillJump”** |  |
| 2а | Персонаж падает в воду | Конец игры |

**ПРОТОКОЛ**

**Предварительных и приемочных испытаний** “ **ChillJump**”

В соответствии с требованиями индивидуального задания были проведены испытания АИС ГКУ в соответствии с утвержденной «Программой и методикой испытаний».

Общие сведения об испытаниях приведены в таблице 1.

Результаты испытаний приведены в таблице 2.

**Таблица 1 - Общие сведения**

|  |  |  |
| --- | --- | --- |
| **Испытываемый образец:** | **Специализированная информационная системы «Информационный портал малого предпринимательства»** | |
| Дата проведения испытаний: | «12» июня                       2021г. | |
| Место проведения испытаний | Колледж ФГБОУ ВО «Вятский государственный университет» | |
| Испытания проводили: | Фамилия, И.О. | Должность |
| От Исполнителя | Тюлькин Данила Алексеевич | Студент |
| От Заказчика | Крутиков Александр Константинович | Руководитель учебной практики |
|  |  |  |

**Таблица 2 - Результаты испытаний**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Шаг испытаний (проверок)** | **№ пункта**  **методики** | **Отметка о прохождении (да/нет)** | **Примечания** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 1 | 5 | 1а | Да |  |
| 2 | 5 | 1б | Да |  |
| 3 | 5 | 1в | Да |  |
| 4 | 5 | 1г | Да |  |
| 5 | 5 | 1д | Да |  |
| 5 | 5 | 1е | Да |  |
| 5 | 5 | 2а | Да |  |

# **ЗАКЛЮЧЕНИЕ**

В данной учебной практике получилось реализовать всё что задумывалось. Получен большой опыт при работе на Unity, при создании на нем игрового приложений. Были изучены основы языка программирования C#. Данная наработка не исправляет недостатки аналогов, но с помощью её можно создать игру, в которой будут учтены все недочеты аналогичных игр, а также будут присутствовать функции, которых нет у конкурентов.

В работе над приложениями появились некоторые проблемы, такие как установка среды разработки, т.к. чтобы начать разработку пришлось потратить немного времени на изучении как Unity, так и Microsoft Visual Studio, но благодаря видео в интернете, данный процесс не занял много времени.

Достоинства:

1. Простата
2. Надёжность
3. Возможность улучшения приложения

Недостатки:

1. Однообразность
2. Довольно примитивный игровой процесс

Таким образом, программа учебной практики ПМ.05 в период с 11.01.2021 по 16.06.2021 была выполнена мной в полном объеме.

При выполнении работ, предусмотренных заданием на практику, в организации ФГБОУ ВО «Вятский государственный университет» мной, Тюлькиным Данилой Алексеевичем, соблюдались правила внутреннего трудового распорядка, охраны труда, техники безопасности, противопожарной безопасности, санитарно-эпидемиологические правила и гигиенические нормативы. Пройден вводный инструктаж и инструктаж на рабочем месте. Замечаний не имел.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Официальная документация от Unity.
2. Joseph Hocking. Unity in Action. – 2015 – 352 с.
3. А. Torn. Animation basics in Unity – 2015 – 175 c.
4. Unity в действии. Мульти платформенная разработка на C#. - М.: Питер, 2018. - **608** c.
5. W. Murray C# Game Programming Cookbook for Unity 3D – 2014 – 458 с

# **ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД**

using UnityEngine;

[RequireComponent(typeof(Rigidbody))]

public class Movement : MonoBehaviour

{

public float Speed = 10f;

public float JumpForce = 300f;

private bool \_isGrounded;

private Rigidbody \_rb;

void Start()

{

\_rb = GetComponent<Rigidbody>();

}

void FixedUpdate()

{

MovementLogic();

JumpLogic();

}

private void MovementLogic()

{

float moveHorizontal = Input.GetAxis("Horizontal");

float moveVertical = Input.GetAxis("Vertical");

Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);

\_rb.AddForce(movement \* Speed);

}

private void JumpLogic()

{

if (Input.GetAxis("Jump") > 0)

{

if (\_isGrounded)

{

\_rb.AddForce(Vector3.up \* JumpForce);

}

}

}

void OnCollisionEnter(Collision collision)

{

IsGroundedUpate(collision, true);

}

void OnCollisionExit(Collision collision)

{

IsGroundedUpate(collision, false);

}

private void IsGroundedUpate(Collision collision, bool value)

{

if (collision.gameObject.tag == ("Ground"))

{

\_isGrounded = value;

}

}

}