МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Забайкальский государственный университет»

(ФГБОУ ВО «ЗабГУ»)

Факультет энергетический

Кафедра информатики, вычислительной техники и прикладной математики

**КУРСОВАЯ РАБОТА**

По дисциплине: интерактивные графические системы

на тему: «Разработка интерактивной графической системы»

Выполнил ст. гр. ИВТ-18 Втулкин В. В.

Проверил ст. преподаватель кафедры ИВТ и ПМ Долгих Р.С.

Чита

2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Забайкальский государственный университет»

(ФГБОУ ВО «ЗабГУ»)

Факультет энергетический

Кафедра информатики, вычислительной техники и прикладной математики

ЗАДАНИЕ

на курсовую работу

По дисциплине: Интерактивные графические системы

Студенту: Втулкину Владиславу Викторовичу

Специальности (направления подготовки): Информатика и вычислительная техника

1. Тема курсовой работы: «Разработка интерактивной графической системы»
2. Срок подачи студентом законченной работы: 22.12.2021
3. Исходные данные к работе:
   1. Описание предметной области;
   2. «Общие требования к построению и оформлению учебной текстовой документации» (МИ 01-02-2018)».

Дата выдачи задания: 15.09.2021

Руководитель курсовой работы /Долгих Р.С./

(подпись, расшифровка подписи)

Задание принял к исполнению

«15» сентября 2021 г.

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Втулкин В. В./

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Забайкальский государственный университет»

(ФГБОУ ВО «ЗабГУ»)

Факультет энергетический

Кафедра информатики, вычислительной техники и прикладной математики

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

По дисциплине: Интерактивные графические системы

На тему: «Разработка интерактивной графической системы»

Выполнил студент группы ИВТ-18 Втулкин В. В.

Руководитель работы: старший преподаватель кафедры ИВТ и ПМ, Долгих Роман Сергеевич

Чита 2021

3

СОДЕРЖАНИЕ

[1 Описание предметной области 8](#_Toc91584094)

[2 Описание средств реализации 10](#_Toc91584095)

[2.1 Игровой движок Unity 11](#_Toc91584096)

[2.2 Преимущества Unity 12](#_Toc91584097)

[2.3 Недостатки Unity 13](#_Toc91584098)

[3 Программная реализация 14](#_Toc91584099)

[3.1 Описание «Blue Platformer» 14](#_Toc91584100)

[3.2 Описание модулей 15](#_Toc91584101)

[ЗАКЛЮЧЕНИЕ 17](#_Toc91584102)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 18](#_Toc91584103)

[ПРИЛОЖЕНИЕ 19](#_Toc91584104)

**Реферат**

Пояснительная записка 24 с., 4 рис., 7 источников, 1 прил.

Ключевые слова: ИГРА, UNITY, ЯЗЫК С#, ПЛАТФОРМЕР.

В данной работе рассматривается процесс создания компьютерной игры

«Blue Platformer» на игровом движке Unity, на языке программирования C#.

В работе определены методы разработки и описан процесс их применения при описании физики движения и управления.

**Введение**

Компьютерные игры — чудо двадцатого века. Для абсолютного большинства они оказались полной неожиданностью. Атомная бомба или полеты в космос в сознании обычного человека тоже не имели предыстории, но о них, по крайней мере, мечтали фантасты, хотя и относили время их появления на сотни лет вперед. У компьютерных игр в массовом сознании нет даже такой предыстории[2].

Компьютер раньше находился в специальном зале, с ним работали высококвалифицированные профессионалы. Они решали оборонные и другие исследовательские задачи. Вычислительная техника ассоциировалась с идеей математики, логики, рациональности. Предположение, что большинство людей станет использовать ее в абсолютно иррациональных целях (для игр), казалось абсурдным — как и словосочетание "компьютерная игра". Однако бесстрастного ученого за монитором сменил азартный игрок.

Сегодня компьютерные игры стали не только развлечением, но и носителем культуры. Они мало похожи на балетную сцену, галерейную стену или книжную страницу, но точно так же фиксируют современную мораль, этику, иллюзии, надежды и представления о прошлом и будущем большинства людей.

Как и любой носитель культуры, одни возможности для самовыражения они создают, другие ограничивают. Раньше писатель или сценарист делал один вариант развития сюжета — теперь сюжет может ветвиться. Раньше композитор записывал один вариант исполнения — сегодня игровой продюсер заказывает 5–6 разных по напряженности вариантов одной и той же темы. Раньше художник был ограничен неподвижным холстом — сейчас на его «полотне» могут плыть облака и течь реки. Технические ограничения постепенно уменьшаются, хотя процесс компьютеризации движущегося изображения пока полон компромиссов, таких как, например, сосуществование разных стандартов (Glide, OpenGL и D3D).

Компьютерные игры дают людям новую уникальную возможность перенестись в мир иллюзий и грез. Никогда раньше не было способа так глубоко погружаться в нереальный мир и иметь там настолько большую свободу поведения. Превосходство над кино, театром и книгами компьютерным играм дает их интерактивность: игры вовлекают в совместную деятельность, игрок перестает быть пассивным наблюдателем, он активно влияет на текущие события.

# 1 Описание предметной области

Платформер ([англ.](https://dic.academic.ru/dic.nsf/ruwiki/6161) platformer) – жанр [компьютерных игр](https://dic.academic.ru/dic.nsf/ruwiki/4846), в которых основной чертой игрового процесса является прыгание по платформам, лазанье по лестницам, собирание предметов, обычно необходимых для завершения уровня[2]. Некоторые предметы, называемые пауэр-апами ([англ.](https://dic.academic.ru/dic.nsf/ruwiki/6161) power-up), наделяют управляемого игроком персонажа особой силой, которая обычно иссякает со временем (к примеру: силовое поле, ускорение, увеличение высоты прыжков). Коллекционные предметы, оружие и "пауэер-ап" собираются обычно простым прикосновением персонажа, и для применения не требуют специальных действий со стороны игрока. Реже предметы собираются в «инвентарь» героя и применяются специальной командой (такое поведение более характерно для аркадных головоломок). Сходный жанр компьютерных игр SideScroller.

Противники (называемые «монстрами» независимо от внешнего вида), всегда многочисленные и разнородные, обладают примитивным [искусственным интеллектом](https://dic.academic.ru/dic.nsf/ruwiki/1423), стремясь максимально приблизиться к игроку, либо не обладают им вовсе, перемещаясь по круговой дистанции или совершая повторяющиеся действия. Соприкосновение с противником обычно отнимает жизненные силы у героя или вовсе убивает его. Иногда противник может быть нейтрализован либо прыжком ему на голову, либо из оружия, если им обладает герой. Смерть живых существ обычно изображается упрощённо или символически (существо исчезает или проваливается в низ экрана).

Уровни, как правило, изобилуют секретами (скрытые проходы в стенах, высокие или труднодоступные места), нахождение которых существенно облегчает прохождение и подогревает интерес игрока.

Игры подобного жанра характеризуются нереалистичностью, рисованной мультяшной графикой. Героями таких игр обычно бывают мифические существа (к примеру: [драконы](https://dic.academic.ru/dic.nsf/ruwiki/2179), [гоблины](https://dic.academic.ru/dic.nsf/ruwiki/94779)) или [антропоморфные](https://dic.academic.ru/dic.nsf/ruwiki/56537) животные.

Платформеры появились в начале [1980-х](https://dic.academic.ru/dic.nsf/ruwiki/8180) и стали [трёхмерными](https://dic.academic.ru/dic.nsf/ruwiki/12207) ближе к концу [1990-х](https://dic.academic.ru/dic.nsf/ruwiki/7539). Через некоторое время после образования жанра у него появилось данное название, отражающее тот факт, что в платформерах [геймплей](https://dic.academic.ru/dic.nsf/ruwiki/290183) сфокусирован на прыжках по платформам в противовес стрельбе. Правда, во многих платформерах присутствует стрелковое оружие, в таких, например, как [Blackthorne](https://dic.academic.ru/dic.nsf/ruwiki/132194) или [Castlevania](https://dic.academic.ru/dic.nsf/ruwiki/158530).

# Описание средств реализации

Для написания компьютерной игры был применён существующий игровой движок. Игровой движок - базовое [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [компьютерной игры](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)[7]. Разделение игры и игрового движка часто расплывчато, и не всегда студии проводят чёткую границу между ними. Но в общем случае термин «игровой движок» применяется для того программного обеспечения, которое пригодно для повторного использования и расширения, и тем самым может быть рассмотрено как основание для разработки множества различных игр без существенных изменений.

Термин «игровой движок» появился в середине 1990-х в контексте компьютерных игр жанра [шутер от первого лица](https://ru.wikipedia.org/wiki/%D0%A8%D1%83%D1%82%D0%B5%D1%80_%D0%BE%D1%82_%D0%BF%D0%B5%D1%80%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BB%D0%B8%D1%86%D0%B0), похожих на популярную в то время [Doom](https://ru.wikipedia.org/wiki/Doom_(%D0%B8%D0%B3%D1%80%D0%B0,_1993)). [Архитектура программного обеспечения](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) Doom была построена таким образом, что представляла собой разумное и хорошо выполненное разделение центральных компонентов игры (например, подсистемы трёхмерной графики, расчёта столкновений объектов, звуковой и других) и графических ресурсов, игровых миров, формирующие опыт игрока, игровые правила и другое. Как следствие, это получило определённую ценность за счёт того, что начали создаваться игры с минимальными изменениями, когда при наличии игрового движка компании создавали новую графику, оружие, персонажей, правила игры и тому подобное.

Разделение между игрой и игровым движком часто неопределённо. Некоторые движки имеют разумное и ясное разделение, в то же время другие практически невозможно отделить от игры. Например, в игре движок может «знать» о том, как рисовать [дугу](https://ru.wikipedia.org/wiki/%D0%94%D1%83%D0%B3%D0%B0_%D0%BE%D0%BA%D1%80%D1%83%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B8), в то же время другой движок может работать с другим уровнем абстракции, и в нём дуга будет частным случаем параметров вызываемых функций. Одним из признаков игрового движка является применение [архитектуры управления данными](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5,_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D0%B5%D0%BC%D0%BE%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%BC%D0%B8). Это определяется тем, что если игра содержит [жёстко фиксированные данные](https://en.wikipedia.org/wiki/Hard_coding) (англ.)[рус.](https://ru.wikipedia.org/w/index.php?title=%D0%A5%D0%B0%D1%80%D0%B4%D0%BA%D0%BE%D0%B4&action=edit&redlink=1), влияющие на логику, правила игры, [рисование объектов](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BD%D0%B4%D0%B5%D1%80%D0%B8%D0%BD%D0%B3) и тому подобное, то становится сложно применять данное программное обеспечение в разных играх.

## Игровой движок Unity

Для реализации был выбран один из самых популярных игровых движков – Unity. Как и многие другие движки Unity имеет свои плюсы и минусы, о них и будет данный раздел.

Для начала немного истории: Unity (unity в переводе с [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) — «единство») — [межплатформенная](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) [среда разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) [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), разработанная американской компанией [Unity Technologies](https://ru.wikipedia.org/wiki/Unity_Technologies)[3]. Unity позволяет создавать приложения, работающие на более чем 25 различных [платформах](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%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0), включающих [персональные компьютеры](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80), [игровые консоли](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BA%D0%BE%D0%BD%D1%81%D0%BE%D0%BB%D1%8C), [мобильные устройства](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%B0), [интернет-приложения](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) и другие. Выпуск Unity состоялся в [2005](https://ru.wikipedia.org/wiki/2005_%D0%B3%D0%BE%D0%B4_%D0%B2_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D1%85_%D0%B8%D0%B3%D1%80%D0%B0%D1%85) году и с того времени идёт постоянное развитие.

Редактор Unity имеет простой [Drag&Drop](https://ru.wikipedia.org/wiki/Drag-and-Drop) интерфейс, а также установкой плагинов KALI который легко настраивать, состоящий из различных окон, благодаря чему можно производить отладку игры прямо в [редакторе](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B9) (рис. 1). Движок использует для написания скриптов [C#](https://ru.wikipedia.org/wiki/C_Sharp). Ранее поддерживались также [Boo](https://ru.wikipedia.org/wiki/Boo) (диалект [Python](https://ru.wikipedia.org/wiki/Python), поддержку убрали в 5-й версии) и модификация [JavaScript](https://ru.wikipedia.org/wiki/JavaScript), известная как UnityScript (поддержка прекращена в версии 2017.1). Расчёты физики производит физический движок [PhysX](https://ru.wikipedia.org/wiki/PhysX) от [NVIDIA](https://ru.wikipedia.org/wiki/NVIDIA). Графический API — [DirectX](https://ru.wikipedia.org/wiki/DirectX) (на данный момент [DX 11](https://ru.wikipedia.org/wiki/DX11), поддерживается [DX 12](https://ru.wikipedia.org/wiki/Direct3D_12" \o "Direct3D 12))

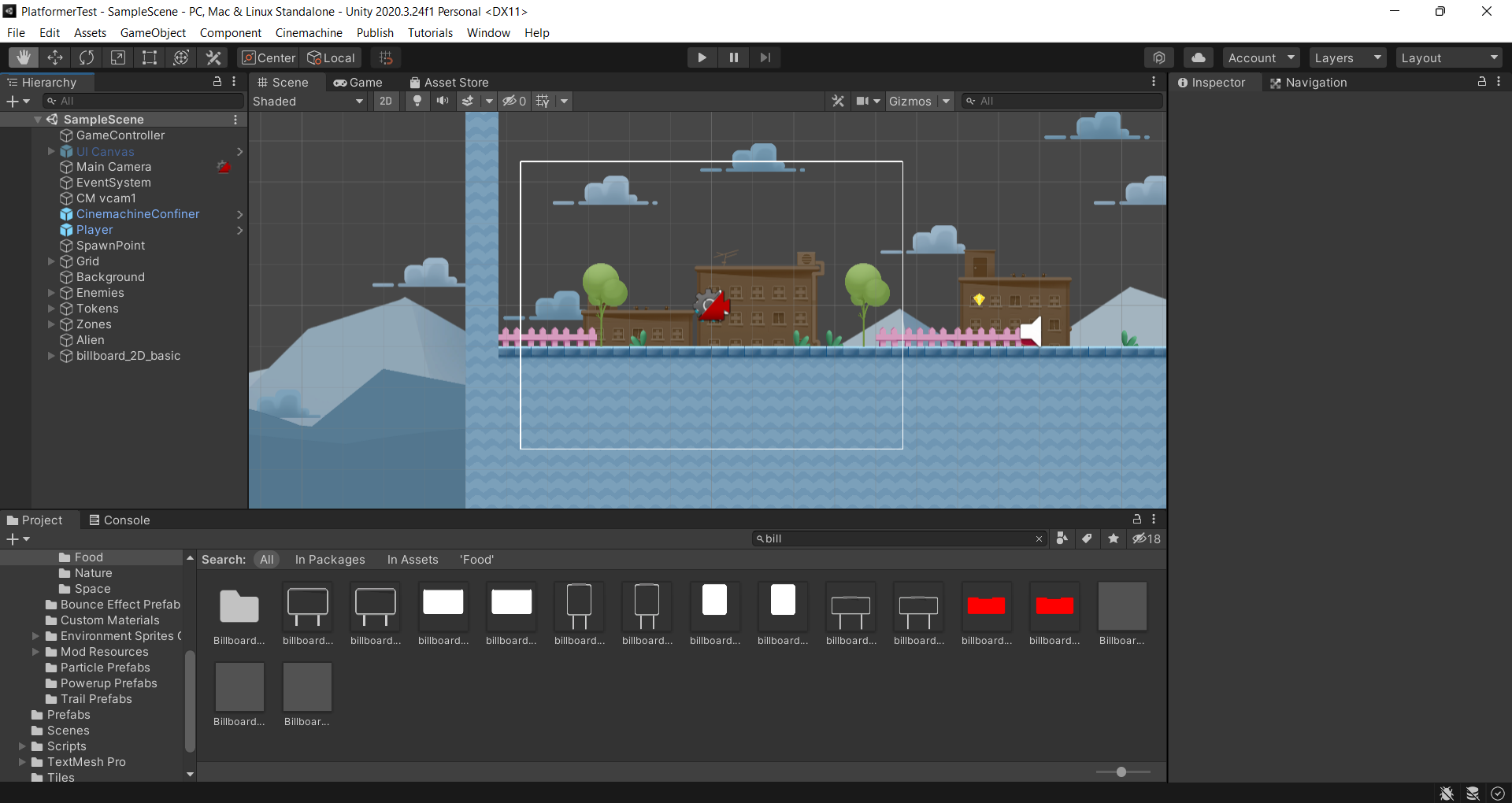


Рисунок 1. Интерфейс игрового движка Unity

Проект в Unity делится на сцены ([уровни](https://ru.wikipedia.org/w/index.php?title=%D0%98%D0%B3%D1%80%D0%BE%D0%B2%D1%8B%D0%B5_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B8&action=edit&redlink=1)) — отдельные файлы, содержащие свои игровые миры со своим набором объектов, сценариев, и настроек. Сцены могут содержать в себе как, собственно, объекты (модели), так и пустые игровые объекты — объекты, которые не имеют модели («пустышки»). Объекты, в свою очередь содержат наборы компонентов, с которыми и взаимодействуют скрипты. Также у объектов есть название (в Unity допускается наличие двух и более объектов с одинаковыми названиями), может быть тег (метка) и слой, на котором он должен отображаться. Так, у любого объекта на сцене обязательно присутствует компонент Transform — он хранит в себе координаты местоположения, поворота и размеров объекта по всем трём осям. У объектов с видимой геометрией также по умолчанию присутствует компонент [Mesh Renderer](https://ru.wikipedia.org/w/index.php?title=Mesh_Renderer&action=edit&redlink=1), делающий модель объекта видимой.

К объектам можно применять коллизии (в Unity т. н. коллайдеры — collider), которых существует несколько типов.

Также Unity поддерживает физику твёрдых тел и ткани, а также физику типа [Ragdoll](https://ru.wikipedia.org/wiki/Ragdoll-%D1%84%D0%B8%D0%B7%D0%B8%D0%BA%D0%B0) (тряпичная кукла). В редакторе имеется система наследования объектов; дочерние объекты будут повторять все изменения позиции, поворота и масштаба родительского объекта. Скрипты в редакторе прикрепляются к объектам в виде отдельных компонентов.

## 2.2 Преимущества Unity

Как правило, игровой движок предоставляет множество функциональных возможностей, позволяющих их задействовать в различных играх, в которые входят моделирование физических сред, карты нормалей, динамические тени и многое другое. В отличие от многих игровых движков, у Unity имеется два основных преимущества: наличие [визуальной среды разработки](https://ru.wikipedia.org/wiki/RAD_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и [межплатформенная](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) поддержка. Первый фактор включает не только инструментарий визуального моделирования, но и [интегрированную среду](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8), [цепочку сборки](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D1%81%D0%B1%D0%BE%D1%80%D0%BA%D0%B8), что направлено на повышение производительности разработчиков, в частности, этапов создания прототипов и тестирования. Под межплатформенной поддержкой предоставляется не только места развертывания (установка на персональном компьютере, на мобильном устройстве, консоли и т. д.), но и наличие инструментария разработки (интегрированная среда может использоваться под Windows и Mac OS).

Третьим преимуществом называется модульная система компонентов Unity, с помощью которой происходит конструирование игровых объектов, когда последние представляют собой комбинируемые пакеты функциональных элементов. В отличие от механизмов [наследования](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), объекты в Unity создаются посредством [объединения функциональных блоков](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BC%D0%B5%D1%81%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), а не помещения в узлы дерева наследования. Такой подход облегчает создание прототипов, что актуально при разработке игр.

## 2.3 Недостатки Unity

В качестве недостатков приводятся ограничение визуального редактора при работе с многокомпонентными схемами, когда в сложных сценах визуальная работа затрудняется. Вторым недостатком называется отсутствие поддержки Unity ссылок на внешние [библиотеки](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), работу с которыми программистам приходится настраивать самостоятельно, и это также затрудняет командную работу. Ещё один недостаток связан с использованием шаблонов экземпляров ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) prefabs). С одной стороны, эта концепция Unity предлагает гибкий подход визуального редактирования объектов, но с другой стороны, редактирование таких шаблонов является сложным. Также, [WebGL](https://ru.wikipedia.org/wiki/WebGL)-версия движка, в силу специфики своей архитектуры (трансляция кода из C# в С++ и далее в JavaScript), имеет ряд нерешённых проблем с производительностью, потреблением памяти и работоспособностью на мобильных устройствах.

# Программная реализация

## Описание «Blue Platformer»

«Blue Platformer» - видеоигра в жанре платформер, в которой игрок управляет игровым аватаром – синим человечком. В начале игры аватар появляется в начальной локации (рис 2.).

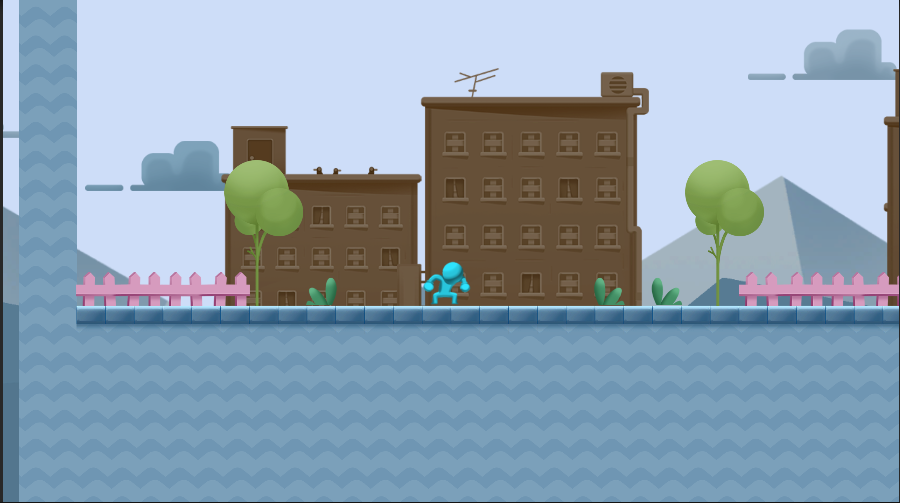


Рисунок 2 Начало уровня.

Целью игрока является прохождение уровня и сбор токенов (рис. 3).

Рисунок 3 Токен для сбора и враг.

Прохождению мешают враги – красные капли. Их можно не только оббегать, но и убивать спрыгнув на них ровно сверху (рис. 3). Также для прохождения необходимо перепрыгивать ямы (рис. 4).



Рисунок 4 Пример пролома.

## 3.2 Описание модулей

Для создания видеоигры «Blue Platformer» были написаны следующие модули:

- AnimationController – объединяет физику и анимацию. Обычно используется для анимации врага;

- DeathZone – отетка для запуска события, связанного со смертью врага;

- EnemyController - простой контроллер для врагов. Обеспечивает управление движением по пути патрулирования;

- GameController – класс для представления игровой модели в инспекторе;

- Health – представляет текущую жизненную статистику некоторого игрового объекта;

- KinematicObject – реализует игровую физику для некоторых объектов в игре;

- PatrolPath – используется для создания пути патрулирования: двух точек, между которыми будут перемещаться враги;

- PlayAudioClip – позволяет воспроизводить звуки в определенных состояниях;

- PlayerController – основной класс, используемый для управления игровым персонажем;

- TokenController – класс для анимации всех экземпляров токенов в сцене. Позволяет обновить сотни спрайтов анимации. Автоматически загружает все экземпляры токенов в сцене (см. приложение А);

- TokenIncstance – класс механизма сбора токенов;

- VictoryZone – тригер на прохождения уровня.

# ЗАКЛЮЧЕНИЕ

В результате проделанной работы была разработана игра «Blue Platformer» на движке Unity.

Была изучена работа игрового движка, его механизмы и свойства. Скрипт игры был написан на объектно- ориентированном языке высокого уровня C#.

Был реализован весь вышеописанный функционал игры, проведено тестирование работоспособности, а также проверено взаимодействие с пользователем.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Википедия – Свободная энциклопедия [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Платформер (Дата обращения: 02.12.2021).
2. Платформер [Электронный ресурс] – URL: <https://dic.academic.ru/dic.nsf/ruwiki/106975> (Дата обращения: 10.12.2021).
3. Википедия – Свободная энциклопедия [Электронный ресурс] – URL: [https://ru.wikipedia.org/wiki/Unity\_(игровой\_движок)](https://ru.wikipedia.org/wiki/Unity_(%D0%B8%D0%B3%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA)) (Дата обращения: 22.11.2021).
4. Хабр [Электронный ресурс] – URL: <https://habr.com/ru/post/459184/> (Дата обращения: 01.12.2021).
5. PSY PRACTICE [Электронный ресурс] – URL <https://psy-practice.com/publications/prochee/rol-kompyuternoy-igry-v-zhizni-cheloveka/> (Дата обращения: 01.12.2021).
6. ВикиЧтение [Электронный ресурс] – URL: <https://psy.wikireading.ru/2242> (Дата обращения: 03.12.2021).
7. Википедия – Свободная энциклопедия [Электронный ресурс] – URL: <https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%BE%D0%B2%D0%BE%D0%B9_%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA> (Дата обращения: 2.12.2021).

# Приложение А

(обязательное)

Примеры модулей видеоигры.

using Platformer.Gameplay;

using UnityEngine;

using static Platformer.Core.Simulation;

namespace Platformer.Mechanics

{

/// <summary>

/// Тригер для прохождения уровня игры

/// </summary>

public class VictoryZone : MonoBehaviour

{

[SerializeField]

// появление текста после окончания игры на билборде

public GameObject text;

void OnTriggerEnter2D(Collider2D collider)

{

var p = collider.gameObject.GetComponent<PlayerController>();

if (p != null)

{

text.SetActive(true);

var ev = Schedule<PlayerEnteredVictoryZone>();

ev.victoryZone = this;

}

}

}

}

using UnityEngine;

namespace Platformer.Mechanics

{

/// <summary>

/// Класс для анимации всех экземпляров токенов в сцене.

/// Позволяет обновить сотни спрайтов анимации. Автоматически загружает все экземпляры токенов в сцене.

/// </summary>

public class TokenController : MonoBehaviour

Продолжение приложения А

{

[Tooltip("Frames per second at which tokens are animated.")]

public float frameRate = 12;

[Tooltip("Instances of tokens which are animated. If empty, token instances are found and loaded at runtime.")]

public TokenInstance[] tokens;

float nextFrameTime = 0;

[ContextMenu("Find All Tokens")]

void FindAllTokensInScene()

{

tokens = UnityEngine.Object.FindObjectsOfType<TokenInstance>();

}

void Awake()

{

// Если токены пустые - найти все экземпляры

if (tokens.Length == 0)

FindAllTokensInScene();

// Регистрируем все токены для работы с данным контроллером

for (var i = 0; i < tokens.Length; i++)

{

tokens[i].tokenIndex = i;

tokens[i].controller = this;

}

}

void Update()

{

// если время для следующего фрейма

if (Time.time - nextFrameTime > (1f / frameRate))

{

// обновить все токены

for (var i = 0; i < tokens.Length; i++)

{

var token = tokens[i];

// не нужно анимировать

if (token != null)

{

Продолжение приложения А

token.\_renderer.sprite = token.sprites[token.frame];

if (token.collected && token.frame == token.sprites.Length - 1)

{

token.gameObject.SetActive(false);

tokens[i] = null;

}

else

{

token.frame = (token.frame + 1) % token.sprites.Length;

}

}

}

// вычислить время следующего фрейма

nextFrameTime += 1f / frameRate;

}

}

}

}

using System.Collections;

using System.Collections.Generic;

using Platformer.Core;

using Platformer.Model;

using UnityEngine;

namespace Platformer.Mechanics

{

/// <summary>

/// AnimationController объединяет физику и анимацию. Обычно используется для простой анимации врага.

/// </summary>

[RequireComponent(typeof(SpriteRenderer), typeof(Animator))]

public class AnimationController : KinematicObject

{

/// <summary>

/// горизонтальная скорость

/// </summary>

public float maxSpeed = 7;

Продолжение приложения А

/// <summary>

/// Максимальная скорость прыжка

/// </summary>

public float jumpTakeOffSpeed = 7;

/// <summary>

/// Used to indicated desired direction of travel.

/// </summary>

public Vector2 move;

/// <summary>

/// Используется для указания желаемого направления движения

/// </summary>

public bool jump;

/// <summary>

/// true для остановки прыжка

/// </summary>

public bool stopJump;

SpriteRenderer spriteRenderer;

Animator animator;

PlatformerModel model = Simulation.GetModel<PlatformerModel>();

protected virtual void Awake()

{

spriteRenderer = GetComponent<SpriteRenderer>();

animator = GetComponent<Animator>();

}

protected override void ComputeVelocity()

{

if (jump && IsGrounded)

{

velocity.y = jumpTakeOffSpeed \* model.jumpModifier;

jump = false;

}

else if (stopJump)

{

stopJump = false;

if (velocity.y > 0)

{

velocity.y = velocity.y \* model.jumpDeceleration;

Продолжение приложения А

}

}

if (move.x > 0.01f)

spriteRenderer.flipX = false;

else if (move.x < -0.01f)

spriteRenderer.flipX = true;

animator.SetBool("grounded", IsGrounded);

animator.SetFloat("velocityX", Mathf.Abs(velocity.x) / maxSpeed);

targetVelocity = move \* maxSpeed;

}

}

}