Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ
НА ТЕМУ:
«Классификация систем для создания трёхмерной среды для визуализации окружения робота-собеседника Ф-2»

Студент ИУ7-55Б _____ Бугаенко А. П.
(Группа) (Подпись, дата) (Фамилия И. О.)

Руководитель _____ Волкова Л. Л.
(Подпись, дата) (Фамилия И. О.)

2022 г.

Содержание

Введение

Ф-2 — это аффективный робот, разрабатываемый командой российских исследователей, обладающий умением общаться с людьми. С его помощью можно разрабатывать стратегии диалога, мимику и жесты, изменения направления взгляда и многое другое [2]. Робот сделан максимально простым, чтобы его можно было легко собрать. Использование данной платформы позволяет исследовать эмоциональный контакт, возникающий за счёт поведения робота, а не за счёт его внешнего вида [Volkova L., Ignatev A., Kotov N., Kotov A. 2021, 163-176]. Робот напоминает мультипликационных персонажей, которые совсем не похожи на человека, но при этом эмоциональны и симпатичны благодаря своим жестам и мимике, а не из-за внешнего сходства с человеком. Робот реагирует на слова: он принимает на вход текст на естественном языке, строит смысл этого текста, выбирает коммуникативную цель (многие из которых - выразить эмоцию) и выполняет мультимодальную реакцию (она включает речь, жесты и/или мимику), характерные для неё. Когнитивный компонент робота отвечает за мышление и за понимание текста. Для входящего высказывания когнитивный компонент должен построить некоторый набор умозаключений, выбрать эмоции, которые может вызывать текст, чтобы далее проявить эти эмоции в поведении робота посредством воспроизведения срисованных с человека эмоциональных реакций. Робот может читать книги в виде текстовых файлов, новости и блоги через подписку RSS, воспринимать устную речь через сторонний сервис преобразования в письменную форму. По письменному тексту для каждого предложения когнитивный компонент строит синтаксическое дерево, а затем конструирует семантическое представление — смысл текста. Смысл текста вызывает у робота различные умозаключения, выводы и ответные реакции. Эти процессы моделируются системой отношений типа «если-то» — сценариев. Робот сравнивает смысл поступившего текста с посылками всех сценариев и активизирует ближайшие сценарии. Смысл поступившего текста может ак-

тивизировать сценарии, ответственные за эмоциональную обработку — ”Меня никто не любит”, ”Я никому не нужен” или, наоборот, ”Приятно быть в центре внимания”. Когнитивный компонент может работать отдельно от робота: он будет прочитывать множество текстов, конструировать их смысл, приписывать каждому смыслу аффективную реакцию и сохранять результаты. При управлении роботом каждый сценарий может сформировать поведенческий пакет на языке BML и передать его роботу для выполнения. Таким образом, поведение робота составляется из BML-пакетов — реакций на окружающие события или просто движений в состоянии покоя: когда роботу нечего делать, он будет слегка двигаться [3]

Одной из проблем данного робота является невозможность проводить эксперименты без наличия физической копии робота. Решить данную проблему представляется возможным при помощи создания виртуальной среды, которая позволит эмулировать робота Ф-2 и набор входных данных (положение головы, ограниченная мимика и жестикуляция).

Создание подобной виртуальной среды будет иметь следующие применения в проекте Ф-2.

1. Будет предоставлена возможность воспроизведения реакций робота в тестовом режиме на этапе разработки элементов реакции или набора реакций, воспроизводимых совместно.
2. Станет возможной постановка экспериментов по взаимодействию робота с респондентами в виртуальном режиме. Это позволит проводить больше экспериментов по коммуникации и, следовательно, получать больше обратной связи от респондентов, в том числе позволит респондентам взаимодействовать с роботом удалённо вместо того или вместе с тем, чтобы приезжать к роботу.
3. Станет возможным проводить параллельные эксперименты по коммуникации человека и робота, что позволит объединить усилия множества респондентов, что приведет к более интенсивному развитию проекта на материале собранной обратной связи.

Целью данной работы является обзор существующих решений для задачи создания трёхмерной среды для визуализации окружения робота-собеседника Ф-2. Для достижения поставленной цели необходимо выполнить следующие

задачи:

- рассмотреть существующие решения для создания трёхмерной виртуальной среды;
- провести анализ рассматриваемых решений;
- сформировать критерии для сравнения существующих решений;
- провести сравнительный анализ проанализированных решений по сформированным критериям.

1 Анализ предметной области

Реализация трёхмерной среды подразумевает создание приложения, с помощью которого можно проводить эксперименты. Так как создание программного обеспечения с нуля может оказаться более затратным, чем использование каких-либо готовых инструментов для создания приложений, необходимо рассмотреть данные инструменты и оценить, насколько выгодным является их использование в данном проекте.

Перед тем, как начать рассмотрение данных инструментов и технологий, необходимо ввести набор критериев, который будет является определяющим при объективном выборе того или иного инструмента для создания трёхмерной среды для визуализации окружения робота Ф-2. Критерии в свою очередь исходят от требований к разрабатываемому программному обеспечению. В набор данных критериев входят:

- кроссплатформенность — при поддержке трёхмерной визуальной средой различных платформ (персональный компьютер, смартфон) и различных операционных систем (Windows, Mac OS, Linux) проведение экспериментов становится более доступным, так как не требуется устанавливать виртуальные машины с нужной системой или покупать новое устройство, которое совместимо с трёхмерной средой;
- наличие инструментов для разработки трёхмерной среды — базовое требование необходимое для решения задачи создания трёхмерной среды.

1.1 Игровые движки

Игровой движок — это программная среда, предназначенная в первую очередь для разработки видеоигр и обычно включающая в себя соответствующие библиотеки и вспомогательные программы. [5] В нашем случае использование игрового движка может позволит снизить время разработки, и обеспе-

чить разработчиков набором уже написанных базовых инструментов, которые можно использовать в процессе разработке виртуальной среды. Использование игрового движка имеет следующие недостатки:

- зависимость от игрового движка — наиболее очевидная проблема, возникающая вследствие того, что в большей части случаев код игрового движка является закрытым, в результате чего внутренние проблемы движка не могут быть решены разработчиком приложения, так как доступа к внутреннему коду нет;
- стоимость использования игрового движка — большая часть игровых движков имеет бесплатную версию в которой недоступны многие инструменты, которые являются необходимыми для разработки и поддержки приложения.

На основе требований к разрабатываемой трёхмерной визуальной среде и особенностей использования игровых движков сформируем требования, с помощью которых будет произведен отбор рассматриваемых ниже движков. В набор входят следующие требования:

- наличие инструментов для разработки трёхмерной среды — движок должен поддерживать работу с трёхмерными объектами и иметь встроенный набор инструментов для работы с ними, данный критерий является обязательным;
- производительность — движок должен иметь либо встроенные средства оптимизации, либо возможность их имплементации для ускорения работы приложения;
- стоимость использования — вследствие особенностей разработки приложения игровой движок должен быть либо бесплатным, либо предоставлять возможность бесплатно использовать движок в некоммерческих целях;
- кроссплатформенность — движок должен позволять разрабатывать приложения под большинство наиболее популярных платформ и операционных систем (Windows, Mac OS, Linux);
- доступность — использование движком нестандартных языков программирования и систем, не являющихся общепринятым стандартом в своей области может сильно осложнить разработку трёхмерной визуальной среды, поэтому использование популярных языков программирования и до-

ступного интерфейса является преимущетсвом.

Согласно статистике, собранной платформой itch.io [6], в состав наиболее популярных игровых движков (имеющих наибольшее количество проектов) входят следующие движки:

- Unity — 82 тысячи проектов, прирост за неделю — 349;
- Construct — 18.3 тысячи проектов, прирост за неделю — 156;
- GameMaker:Studio — 12.1 тысячи проектов, прирост за неделю — 56;
- Godot — 8.640 тысяч проектов, прирост за неделю — 82;
- Twine — 7.872 тысячи проектов, прирост за неделю — 51;
- Unreal Engine — 4.819 тысяч проектов, прирост за неделю — 27;
- Bitsy — 4.814 тысяч проектов, прирост за неделю — 27;
- RPG Maker — 4.531 тысяч проектов, прирост за неделю — 17;
- PICO-8 — 4.234 projects тысяч проектов, прирост за неделю — 19.

Остальные движки имеют менее четырёх тысяч проектов и гораздо меньший прирост по количеству, поэтому рассматриваться не будут. Также необходимо изначально учесть первый критерий — наличие инструментов для работы с трёхмерными объектами.

- Unity — данный движок имеет встроенную поддержку трёхмерной графики [7].
- Construct — данный движок имеет только инструменты для 2D разработки, поэтому он не может быть использован при разработке трёхмерной визуальной среды [8].
- GameMaker:Studio — данный движок имеет ограниченную поддержку трёхмерной разработки, включающую основные трансформации трёхмерных объектов и возможность создания нескольких примитивов. Этого достаточно для того, чтобы разработать прототип проекта, однако возможности развития проекта при использовании данного движка органичены [9].
- Godot — данный движок имеет встроенную поддержку трёхмерной графики [10].
- Twine — данный движок используется для разработки интерактивных текстовых игр и поэтому для нашей задачи не подходит [11].
- Unreal Engine — данный движок имеет встроенную поддержку трёхмер-

ной графики [12].

- Bitsy — не имеет поддержки трёхмерной графики [13].
- RPG Maker — не имеет поддержки трёхмерной графики [14].

В результате из рассматриваемых движков остаются следующие: Unity, Godot, Unreal Engine. Рассмотрим их подробнее.

1.1.1 Технологии

Unity

Unity поддерживает графические API DirectX, Metal, OpenGL и Vulkan в зависимости от доступности API на конкретной платформе. Unity использует встроенный набор графических API или графические API, которые выбираются в редакторе [7].

Unreal Engine

Система рендеринга в Unreal Engine использует конвейеры, основанные на технологиях DirectX 11 и DirectX 12. Она включает в себя отложенное затенение, глобальное освещение, освещение прозрачных объектов и пост-обработку, также как и симуляциию частиц на графическом процессоре с использованием векторных полей [12].

Godot

Godot использует OpenGL ES 3.0 для высококачественного рендеринга (и OpenGL 3.3 на персональных компьютерах). Это обеспечивает совместимость со всеми настольными ПК, мобильными устройствами и WebGL 2 [10].

1.1.2 Стоимость использования движка

Unity

Лицензирование Unity делится на 4 уровня.

- Personal — если доход от использования приложения не превышает 100 000 долларов, то для разработчика доступна поддержка 20 пользователей,

одновременно использующих приложения, стандартная сборка в облачном хранилище. Стоимость — бесплатно.

- Plus — если доход от использования приложения не превышает 100 000 долларов, то для разработчика доступна поддержка 50 пользователей, одновременно использующих приложения, приоритетная сборка в облачном хранилище, отчёты по производительности. Стоимость — 399 долларов в год.
- Pro — если доход от использования приложения не превышает 100 000 долларов, то для разработчика доступна поддержка 200 пользователей, одновременно использующих приложения, одновременная сборка в облачном хранилище, отчёты по производительности, премиум поддержка. Стоимость — 1800 долларов в год.
- Enterprise — если доход от использования приложения не превышает 100 000 долларов, то для разработчика доступна поддержка пользовательского мультиплеера, выделенные ресурсы в облачном хранилище, отчёты по производительности, премиум поддержка и доступ к исходному коду. Стоимость - 200 долларов в месяц [7].

Unreal Engine

Unreal Engine 4 является бесплатным, однако разработчики должны передавать 5% от выручки с продаж приложения компании Epic Games, если выручка превышает 1 000 000 долларов [12].

Godot

Godot Engine — это бесплатное программное обеспечение с открытым исходным кодом, выпущенное под разрешительной лицензией MIT (также называемой лицензией Expat). Эта лицензия предоставляет пользователям ряд свобод:

- можно использовать Godot Engine для любых целей;
- можно изучить, как работает Godot Engine, и изменить его;
- можно распространять немодифицированные и измененные версии Godot Engine даже на коммерческой основе и под другой лицензией (включая проприетарную).

Единственное ограничение этой третьей свободы заключается в том, что вам необходимо распространять уведомление об авторских правах и заявление о лицензии Godot Engine всякий раз, когда вы его распространяете. Таким образом, ваш производный продукт может иметь другую лицензию, но в документации должно быть указано, что он является производным от движка Godot Engine, лицензированного MIT [10].

1.1.3 Кроссплатформенность

Unity

Unity поддерживает следующие ОС:

- мобильные ОС — iOS, Android;
- не мобильные ОС — Windows, Mac, Linux [15].

Unreal Engine

Unity поддерживает следующие платформы:

- мобильные ОС — iOS, Android;
- не мобильные ОС — Windows, Mac, Linux [16].

Godot

Unity поддерживает следующие платформы:

- мобильные ОС — iOS, Android;
- не мобильные ОС — Windows, Mac, Linux [17].

Стоит заметить, что все три рассматриваемых движка поддерживают серию различных консольных платформ и различные VR-платформы, однако в нашем случае мы это не учитываем, поскольку они не относятся к целевым платформам разрабатываемого приложения.

1.1.4 Язык программирования для разработки

Unity

При разработке приложений на Unity используется код, написанный на языке C# [7].

Unreal Engine

При разработке приложений на Unreal Engine используется код написанный либо на C++, либо на Blueprint — язык разработанный специально для Unreal Engine [12].

Godot

При разработке приложений на Godot используется код, написанный на GDScript, C#, GDVisual [10].

1.2 Программные интерфейсы

API (программный интерфейс приложения) — описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой. Обычно входит в описание какого-либо интернет-протокола, программного каркаса (фреймворка) или стандарта вызовов функций операционной системы [18].

1.2.1 Графические API

На сегодняшний день существует множество графических API, которые могут быть использованы для реализации графической части проекта. Рассмотрим наиболее популярные из них на сегодняшний день: OpenGL, Vulkan, WebGL, DirectX, OpenGL ES, WebGPU, Metal, Mantle [19].

Составим список требований, которым должно удовлетворять графическое API:

- кроссплатформенность — графический API должен поддерживать целевые платформы, для которых разрабатывается приложение;
- производительность — графический API должен обеспечивать достаточную скорость рендеринга изображения для обеспечения работы приложения в реальном времени;
- доступность — использование графическим API нестандартных языков программирования и систем, не являющихся общепринятым стандартом

в своей области, может сильно осложнить разработку трёхмерной визуальной среды, поэтому использование популярных языков программирования и доступного интерфейса является преимущетсвом;
- стоимость использования — вследствие особенностей разработки приложения игровой движок должен быть либо бесплатным, либо предоставлять возможность бесплатно использовать движок в некоммерческих целях.

Кросплатформенность

Рассмотрим платформы, поддерживаемые представленными выше графическими API:

- OpenGL — Windows, Linux, Mac, FreeBSD;
- Vulkan — Windows, Linux, Max, FreeBSD;
- WebGL — Web platform;
- DirectX — Windows;
- OpenGL ES — Linux, Windows;
- Metal — iOS, Mac, tvOS.

Целевыми платформами данного проекта являются Windows и Linux, поэтому из всех предложенных графических API выбраны OpenGL, Vulkan, DirectX, OpenGL ES, Metal, Mantle. Остальные рассмотрены не будут. Также стоит заметить, что согласно официальному сайте OpenGL, развитие данного графического API прекращено в пользу развития Vulkan, поэтому использование OpenGL и OpenGL ES подразумевает риск для возможности дальнейшей поддержки разработанного приложения. Вследствие чего они не будут включаться в рассматриваемые графические API.

Стоимость

Рассмотрим стоимость использования графических API:

- Vulkan — распространяется бесплатно [20];
- DirectX — распространяется бесплатно[21];
- Metal — распространяется бесплатно [22].

Доступность

Рассмотрим доступность графических API:

- Vulkan — основным способом работы с Vulkan является использование официальной библиотеки под C++, однако помимо неё существует серия неофициальных библиотек, позволяющих использовать такие языки, как Python и C# [20];
- DirectX — основным способом работы с DirectX является использование официальной библиотеки от Microsoft под C++ [21];
- Metal — для работы с Metal API используется официальный интерфейс от Apple, написанный на C++ [22].

Производительность

Согласно данным, полученным в ходе экспериментов на серии различных видеокарт, Vulkan работает быстрее DirectX в среднем на 5 % [23]. Если сравнивать Vulkan и Metal, то согласно данным, полученным в ходе экспериментов на серии различных видеокарт, Metal работает быстрее Vulkan на графических процессорах AMD, достигая преимущества от 8 до 40 процентов в производительности. Однако при тестировании на картах от NVIDIA Vulkan выигрывает от 5 до 30 процентов в производительности.

1.3 Вывод из анализа предметной области

В результате проведённого анализа были рассмотрены решения, позволяющие решить задачу создания трёхмерной виртуальной среды.

2 Сравнительный анализ существующих решений

Полученное множество систем, позволяющих решить задачу создания трёхмерной среды можно классифицировать по типу системы, по стоимости использования системы, доступности и кроссплатфоменности.

Классификация решений по типу подразумевает разбиение по виду реализации предложенных систем. В данной работе рассматриваются два типа систем — игровой движок и графический API. Данные решения таже разбиваюся согласно требованиям, которым должно удовлетворять решение, использующееся в разработе трёхмерной виртуальной среды. В частности, этими требованиями являются стоимость использования — движок или графическое API должен находиться в свободном доступе, либо позволять использовать основные функции в некой бесплатной версии, кроссплатформенность — чем большее количество платформ покрывает данная система, тем меньше времени требуется на разработку кроссплатформенной версии приложения, доступность — использование системой нестандартных языков программирования и систем, не являющихся общепринятым стандартом в своей области, может сильно осложнить разработку трёхмерной визуальной среды, поэтому использование популярных языков программирования и доступного интерфейса является преимуществом.

Стоит заметить, что производительность не будет рассматриваться как критерий, поскольку все рассматриваемые варианты предоставляют функционал достаточный, чтобы реализовать динамическое взаимодействие с объектами в трёхмерной среде.

2.1 Анализ возможностей рассмотренных решений

Ниже в таблице 2.1 представлены сведения о кроссплатформенности и доступности рассмотренных решений, при этом решения разбиты на два класса — игровые движки и графические API. Базовым средством решения задачи создания трёхмерной среды в данном случае является решение, функционал которого позволяет в полной мере реализовать трёхмерную виртуальную среду.

Таблица 2.1: Анализ возможностей рассмотренных решений

Типы систем	Игровой движок			Графический API		
	Unity	Godot	Unreal Engine	Vulkan	DirectX	Metal
Кроссплатформенность						
Windows	Да	Да	Да	Да	Да	Нет
Mac OS	Да	Да	Да	Да	Нет	Да
Linux	Да	Да	Да	Да	Нет	Нет
Стоимость						
Бесплатно без ограничений	Нет	Да	Нет	Да	Да	Да
Бесплатно с ограничениями	Да	Нет	Да	Нет	Нет	Нет
Доступность						
Используется C++	Нет	Нет	Да	Да	Да	Да
Используется C#	Да	Да	Нет	Нет	Нет	Нет
Используются собственные языки	Нет	Да	Да	Нет	Нет	Нет
Необходимость виртуальной среды выполнения						
Требуется виртуальная среда	Да	Да	Нет	Нет	Нет	Нет

Необходимость включения требования к наличичию виртуальной среды объясняется тем, что некоторые языки (в частности C#) разрабатываются под специфические программные платформы, являющиеся по сути виртуальной средой выполнения кода. Наличие данной среды позволяет программам, напи-

санных на языках, построенных на основе программной платформы, использовать пространство имён System. Минусом данного подходя является замедление компиляции и работы программы вследствие того, что код, написанный на таком языке, сначала должен быть переведён в байт-код, а затем native-код, который уже может быть выполнен [26].

2.2 Анализ дополнительных возможностей реализации программного обеспечения

Одним из преимуществ игровых движков является то, что они изначально созданы как комплексная система, содержащая в себе множество уже реализованных встроенных подсистем, которые могут быть использованы при реализации трёхмерной виртуальной среды. В частности такими системами являются поддержка многопользовательского режима. Многопользовательский режим является одним из запланированных расширений разработанной трёхмерной среды, и в случае реализации трёхмерной среды при помощи графического API всю систему, которая отвечает за многопользовательскую работу, необходимо будет реализовать с нуля, используя соответствующие библиотеки. Игровые движки с другой стороны предоставляют высокоуровневые API, позволяющие быстро и эффективно реализовать многопользовательский режим работы приложения.

2.3 Вывод из сравнительного анализа существующих решений

На основе полученного в результате анализа сравнения существующих решений сделаны выводы о том, что в случае, когда в трёхмерной среде необходимо реализовать специфические функции на низком уровне, наилучшим решением будет использование графического API Vulkan, поскольку данный API является кроссплатформенным, распространяется бесплатно, использует C++ как основной язык и не имеет виртуальной среды выполнения. Если необходимо разработать трёхмерную виртуальную среду, затрачивая наименьшее количество человеко-часов, то лучшим решением будет использование Unity, посколь-

ку данный игровой движок является кроссплатформенным, распространяется бесплатно в случае, если прибыль проекта не превышает установленного порога, и использует C#, являющийся языком программирования более высокого уровня.

Заключение

Цель данной работы достигнута: проведён обзор существующих решений для задачи создания трёхмерной среды для визуализации окружения робота-собеседника Ф-2. Были выполнены следующие задачи:

- рассмотрены существующие решения для создания трёхмерной виртуальной среды;
- проведён анализ рассматриваемых решений;
- сформированы критерии для сравнения существующих решений;
- проведён сравнительный анализ проанализированных решений по сформированным критериям.

На основе полученного в результате анализа сравнения существующих решений сделаны выводы о том, что в случае, когда в трёхмерной среде необходимо реализовать специфические функции на низком уровне, наилучшим решением будет использование графического API Vulkan, поскольку данный API является кроссплатформенным, распространяется бесплатно, использует C++ как основной язык и не имеет виртуальной среды выполнения. Если необходимо разработать трёхмерную виртуальную среду, затрачивая наименьшее количество человеко-часов, то лучшим решением будет использование Unity, поскольку данный игровой движок является кроссплатформенным, распространяется бесплатно в случае, если прибыль проекта не превышает установленного порога, и использует C#, являющийся языком программирования более высокого уровня.

Список использованных источников

[1] Volkova L., Ignatev A., Kotov N., Kotov A. New Communicative Strategies for the Affective Robot: F-2 Going Tactile and Complimenting // Creativity in Intelligent Technologies and Data Science, CCIS, vol. 1448. — Springer, Cham, 2021. — С. 163-176.

[2] Volkova L., Ignatev A., Kotov N., Kotov A. A Robot Commenting Texts in an Emotional Way // Creativity in Intelligent Technologies and Data Science, CCIS, vol. 754. — Springer, Cham, 2017. — С. 256-266.

[3] Робот Ф-2 [Электронный ресурс]. — Электрон. данные. — 2021. — Режим доступа: http://f2robot.com/robot/ (Дата обращения: 29.11.2021).

[4] Кибрик А.Е. Язык // Языкознание. Большой энциклопедический словарь. — Москва, Большая Российская энциклопедия, 1998. — С. 605.

[5] Valencia-Garcia, Rafael, Technologies and Innovation: Second International Conference. — Guayaquil, Ecuador, 2016 — С. 146.

[6] itch.io [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://itch.io/game-development/engines/most-projects (Дата обращения 25.01.2022).

[7] Unity Documentation [Электронный ресурс]. — Электрон. текстовые данные. —- Сан-Франциско, Unity Technologies, 2022. — Режим доступа: https://docs.unity3d.com/Manual/UnityManual.html (Дата обращения 25.01.2022).

[8] Construct Documentation [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://www.construct.net/en/make-games/manuals/construct-3 (Дата обращения 25.01.2022).

[9] GameMaker Studio 2 Manual [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://ru.wikipedia.org/wiki/GameMaker:_Studio (Дата обращения 25.01.2022).

[10] Godot [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://godotengine.org/ (Дата обращения 26.01.2022).

[11] Twine [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://twinery.org/ (Дата обращения 26.01.2022).

[12] Unreal Engine 4 Documentation [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://docs.unrealengine.com/4.27/en-US/ (Дата обращения 26.01.2022).

[13] Bisty [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://www.bitsy.org/#0,0. —

[14] RPG Maker [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://www.rpgmakerweb.com/ (Дата обращения 27.01.2022).

[15] Официальный сайт компании Unity Technologies [Электронный ресурс]. — Электрон. данные. — Сан-Франциско, Unity Technologies, 2022. — Режим доступа: https://unity.com/our-company (Дата обращения 1.2.2022).

[16] Официальный сайт Unreal Engine [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://www.unrealengine.com/en-US/ (Дата обращения 1.02.2022).

[17] Официальный сайт Godot [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://godotengine.org/news (Дата обращения 1.02.2022).

[18] The Linux kernel user-space API guide [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://www.kernel.org/doc/html/latest/userspace-api/index.html (Дата обращения 2.02.2022).

[19] What are the best cross-platform 3D graphics APIs? [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа:

https://www.slant.co/topics/5346/ cross-platform-3d-graphics-apis (Дата обращения 2.02.2022).

[20] Официальный сайт Vulkan [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://vulkan.org/ (Дата обращения 8.02.2022).

[21] Официальный сайт DirectX [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://docs.microsoft.com/en-us/windows/win32/getting-started-with-direct3d (Дата обращения 8.02.2022).

[22] Официальный сайт Metal [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://developer.apple.com/metal/ (Дата обращения 8.02.2022).

[23] GeForce RTX 3090 Founder review — API Performance: Vulkan vs OpenGL vs DirectX12 [Электронный ресурс]. — Электрон. данные. — 2022 — Режим доступа: https://www.guru3d.com/articles_pages/geforce_rtx_3090_founder_review,29.html (Дата обращения 8.02.2022).

[24] Metal Benchmarks [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://browser.geekbench.com/metal-benchmarks (Дата обращения 8.02.2022).

[25] Vulkan Benchmarks [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://browser.geekbench.com/vulkan-benchmarks. — (Дата обращения 8.02.2022).

[26] Краткий обзор языка C# [Электронный ресурс]. — Электрон. данные. — 2022. — Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/ (Дата обращения 8.02.2022).