Санкт-Петербургский государственный университет

ЕГОРОВА Лада Владимировна

Выпускная квалификационная работа

Реализация трёхмерной визуализации для среды TRIK Studio

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *СВ.5006.2017 «Математическое обеспечение и администрирование информационных систем»*

Профиль *Системное программирование*

Научный руководитель:
доцент кафедры СП, к.т.н. Ю.В. Литвинов

Консультант:
старший преподаватель кафедры СП Я.А. Кириленко

Рецензент:
разработчик ПО ООО ”Яндекс Беспилотные Технологии” Г.А. Зимин

Санкт-Петербург
2022

Saint Petersburg State University

Egorova Lada Vladimirovna

Bachelor's Thesis

Three-dimensional modeling visualization for the TRIK Studio

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5006.2017 "Software and Administration of Information Systems"*

Profile: *Software Engineering*

Scientific supervisor:
C.Sc., docent Y.V. Litvinov

Consultant:
senior lecturer Y.A. Kirilenko

Reviewer:
software developer at "Yandex Self-Driving Group" G.A. Zimin

Saint Petersburg
2022

Содержание

1. Введение

TRIK Studio [8] — среда разработки, ориентированная на обучение программированию роботов, которая известна во многих странах. Более того, так как неотъемлемую часть обучения составляют соревнования, в мире ежегодно проводятся состязания школьников с использованием этой программы. Значительная их часть проходит онлайн.

Несмотря на удобство и популярность, TRIK Studio обладает рядом недостатков, которые во многом связаны с внешним видом продукта. В частности, сцена для тестирования выполнена в двухмерном режиме с использованием малого количества декораций и цветов. Из-за этого визуализация работы робота выглядит плоской и тусклой. Это создаёт противоречие в глазах детей, которые привыкли к тому, что даже самые простые современные игры объёмные и красочные. В связи с этим возникла задача расширить программу так, чтобы получить возможность создавать яркие трёхмерные сцены.

Уже сейчас существуют трёхмерные симуляторы и инструменты для программирования роботов [3, 2, 7]. Третье измерение расширяет спектр возможностей движка. Однако переход в движке к новому измерению влечёт за собой серьёзные затраты времени и ресурсов. Проще добиться эффекта псевдотрёхмерности, то есть визуализации, которая внешне выглядит как трёхмерная, но на самом деле таковой не является. Это решит задачу требуемого внешнего вида, не требуя при этом существенных изменений в движке TRIK Studio. Будем надеяться, что в будущей версии TRIK Studio появится возможность писать программы для настоящих трёх измерений.

В обновлённой версии TRIK Studio хотелось бы уметь работать с материалами и текстурами. Также необходим редактор сцен, который будет доступен пользователям. Это позволит каждому учителю оформлять домашние задания как сказки и истории не только на словах, но и наглядно. Такое оформление не будет отталкивающим, и, возможно, даже повысит интерес детей к обучению.

Таким образом возникла задача создать трёхмерную среду визуа-

лизации для TRIK Studio.

2. Постановка задачи

Целью работы является создание трёхмерной среды визуализации для TRIK Studio. В связи с этим были поставлены следующие задачи:

- Выполнить обзор существующих движков, выбрать наиболее подходящий;
- Разработать архитектуру для взаимодействия TRIK Studio и выбранного движка;
- Реализовать визуализацию сохранённой трассы из TRIK Studio;
- Реализовать визуализацию трассы из TRIK Studio в реальном времени;
- Провести апробацию проекта.

3. Обзор

TRIK Studio имеет встроенный симулятор сцен, но он обладает рядом недостатков. В частности, поддерживается только двухмерный режим симуляции.

В 2016 году была проведена интеграция TRIK Studio с трёхмерным симулятором V-REP [12]. В результате была создана трёхмерная модель робота, проведена её интеграция с TRIK Studio, добавлен трёхмерный режим и проведён тест простой программы: робот едет по прямой, пока не увидит препятствие, затем движется в обратном направлении в течение двух секунд. Однако сейчас среда V-REP обновилась под новым названием CoppeliaSim, и V-REP уже не поддерживается.

В 2017 году была реализована инфраструктура для студенческого проекта «Танчики роботов» [11]. Соответствующая модель робота-тележки была интегрирована в V-REP и в симуляторе была реализована недостающая функциональность: акселерометр, гироскоп и эмулятор энкодера. Однако V-REP специализируется на симуляции роботов и практически не уделяет внимания дизайну сцен, что для текущей задачи является ключевым недостатком.

3.1. Существующие движки

Для выбора движка необходимо понять, каким пожеланиям он должен удовлетворять. Основные пожелания связаны с тем, чтобы сделать использование движка доступным для широкого числа пользователей. К таким пожеланиям относятся:

1. Достаточное количество документации и других обучающих источников.

2. Возможность проектировать локации, используя инструменты дизайна.

3. Системные требования, позволяющие работать с использованием современных бюджетных моделей компьютеров (не все ученики и преподаватели могут позволить себе мощные машины).

4. Поддержка различных операционных систем (чтобы не ограничивать круг потенциальных пользователей).

В соответствии с этими пожеланиями будем выбирать подходящий движок среди популярных сред для симуляции роботов и платформ для создания игр.

3.1.1. CoppeliaSim

CoppeliaSim (обновлённый V-REP) [1] с интегрированной средой разработки позиционирует себя как универсальный симулятор для моделирования, прототипирования и обучения работе с роботами. Он кроссплатформенный, поддерживает шесть языков программирования и широкий набор сенсоров для роботов. Но он не предоставляет такого широкого набора текстур и материалов, как игровые движки, что противоречит задаче.

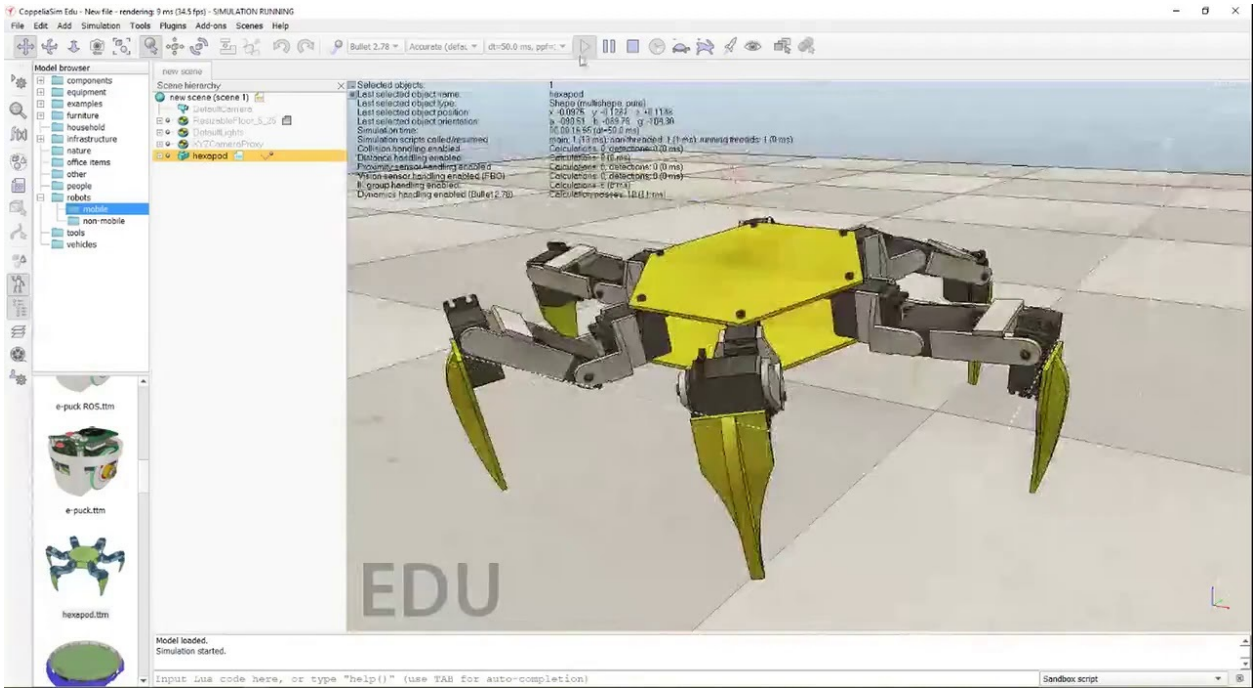

Рис. 1: CoppeliaSim.

3.1.2. Gazebo

Gazebo [5] — известный трёхмерный симулятор роботов с открытым исходным кодом. Он предоставляет возможности дизайна, однако его главным недостатком является то, что на нём можно работать только в семействе операционных систем Linux. Это существенно, так как хотелось бы, чтобы проводить визуализацию могли пользователи из разных школ по всему миру, а оборудование в них, и, в частности, операционные системы в компьютерных классах будут отличаться.

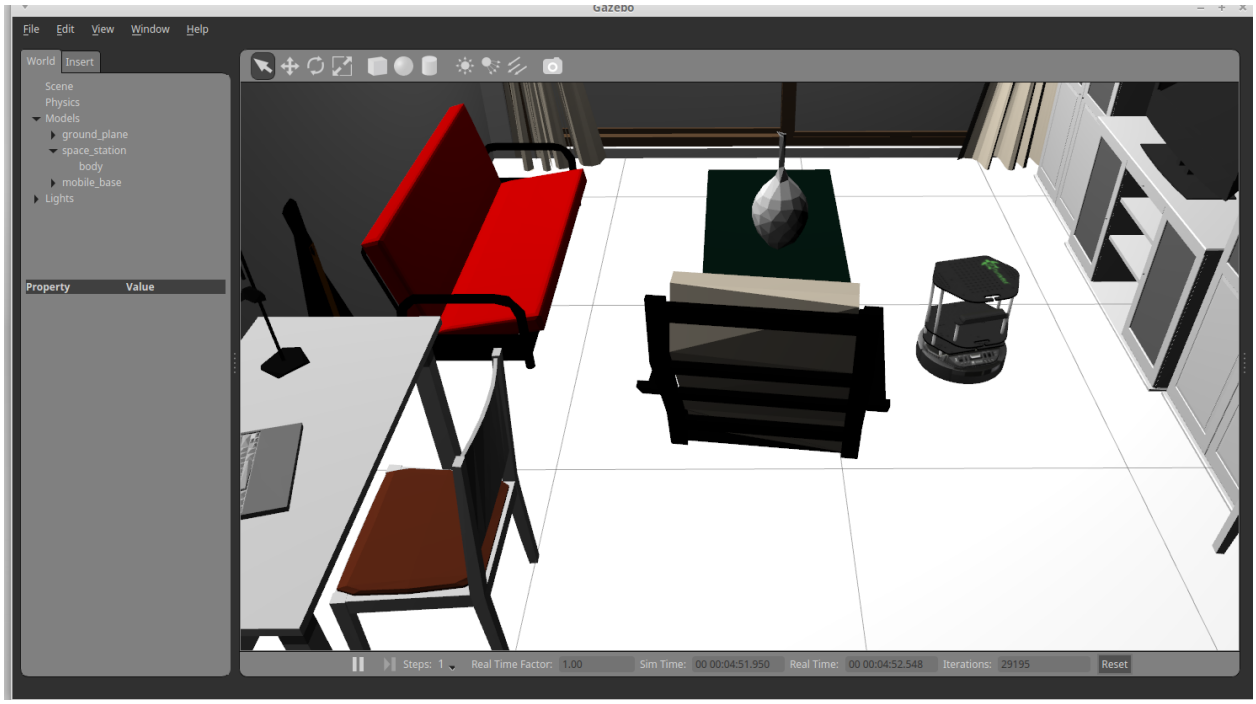

Рис. 2: Gazebo.

3.1.3. Unity

Интерес к игровым движкам вызван тем, что они предоставляют широкие возможности дизайна сцен, что напрямую связано с поставленной задачей. Unity [10] — один из наиболее известных игровых движков, но, помимо этого, он также используется для симулирования физических явлений в разных областях науки. Отличается кроссплатформенностью и большим объёмом документации. С простотой освоения Unity связан значительный объём игр, написанных любителями со всего мира. Так как это игровой движок, то в нём поддерживается продвинутая возможность дизайна сцен, текстур и освещения.

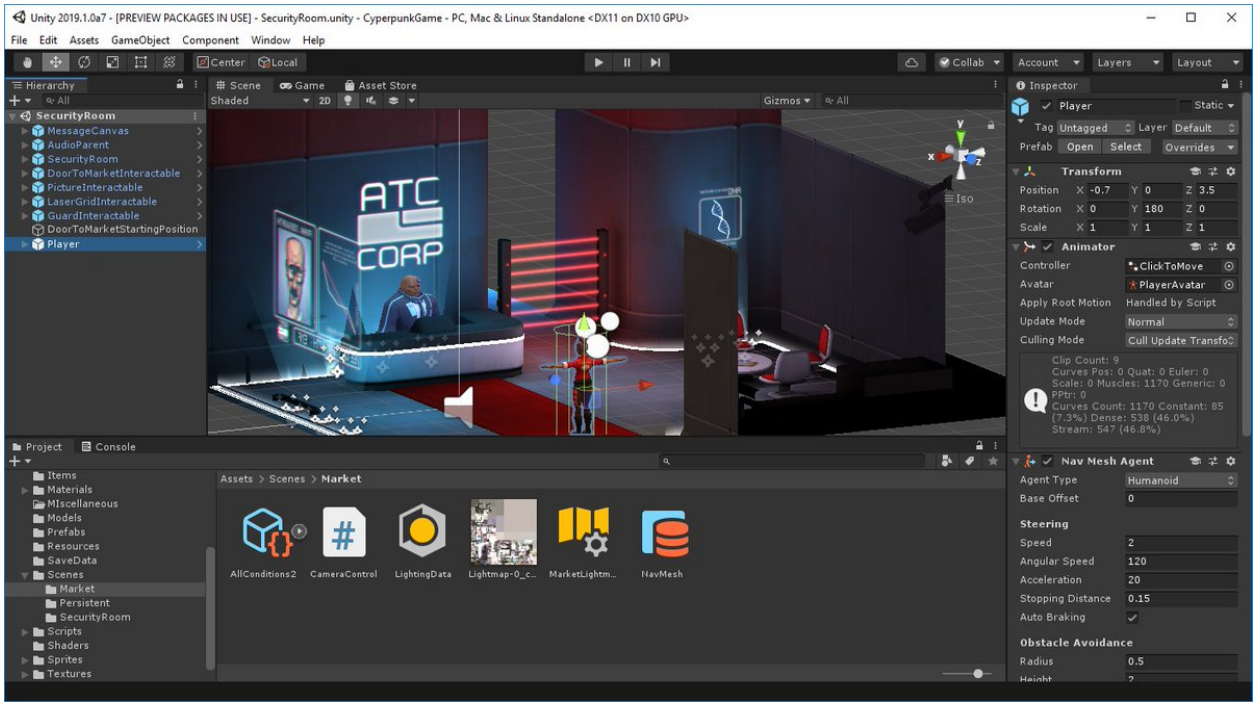

Рис. 3: Unity.

3.1.4. Unreal Engine 4

Unreal Engine 4 (UE 4) [4] — также известный кроссплатформенный игровой движок. Как и Unity, он обладает высокими системными требованиями и взамен предлагает широкие возможности проектирования игровых локаций. Но по сравнению с Unity Unreal Engine обладает меньшим объёмом документации и других обучающих источников в Интернете.

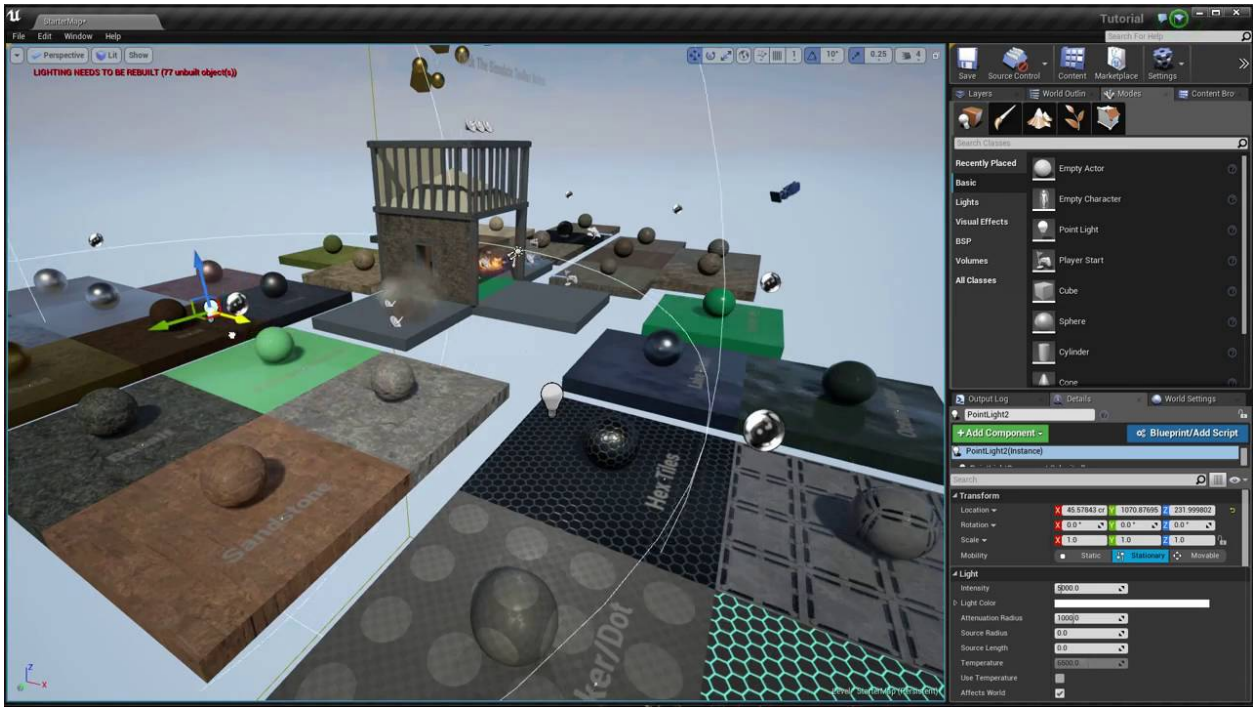

Рис. 4: Unreal Engine.

3.1.5. Сравнение движков

Наглядно сравнить движки в соответствии с требованиями можно с помощью следующей таблицы:

Движок	CoppeliaSim	Gazebo	Unity	UE 4
Количество документации	много	средне	много	средне
Инструменты дизайна	практически нет	есть	есть	есть
Системные требования	низкие	низкие	высокие	высокие
Кроссплатформенность	да	нет	да	да

Таблица 1: Сравнение движков

Таким образом для выполнения поставленной задачи был выбран движок Unity. Несмотря на высокие системные требования, он обладает рядом достоинств, благодаря которым можно работать с ним, несмотря на этот недостаток.

4. Реализация

Так как планируется, что проектом будут пользоваться учителя, чтобы давать домашние задания, то можно предположить, что они сами будут создавать сцены и потом на них показывать работу программ учеников. Следовательно, для визуализации сначала нужно научиться создавать сцену в Unity и переносить её в TRIK Studio. Затем ученик напишет программу, которая позволит роботу выполнить определённую задачу. После этого нужно научиться переносить трассу робота в Unity и визуализировать её.

4.1. Разработка архитектуры проекта

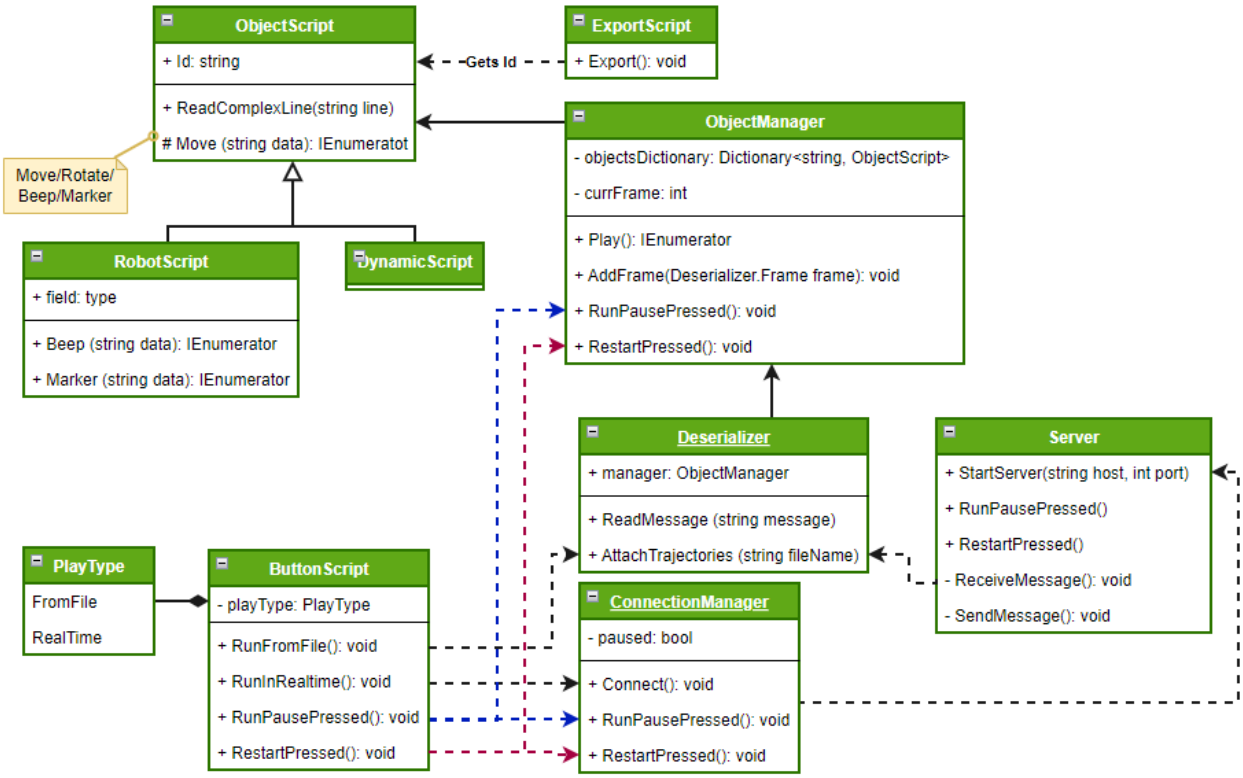

Рис. 5: Архитектура проекта (Unity).

В соответствии с пользовательскими сценариями, а также с тем, что TRIK Studio и Unity — обособленные самостоятельные движки, архитектура проекта разбита на две части:

1. Unity (Рис. 5). За сохранение сцены отвечает класс ExportScript. Для корректности ему необходимо обращаться к объектам на сцене, чтобы присвоить им уникальный Id, который затем попадёт в файл сцены.

В Unity к каждому динамическому объекту прикреплён скрипт, унаследованный от класса ObjectScript: у робота это RobotScript, у других объектов (в TRIK Studio к таким относятся мячи и кегли) это DynamicScript. RobotScript отличается тем, что в нём реализованы методы рисования линии и включения звукового сигнала.

Класс Deserializer считывает файл траектории и обращается к объектам, чтобы присвоить им траектории.

Пользователь использует кнопки RunFromFile (запускает визуали-

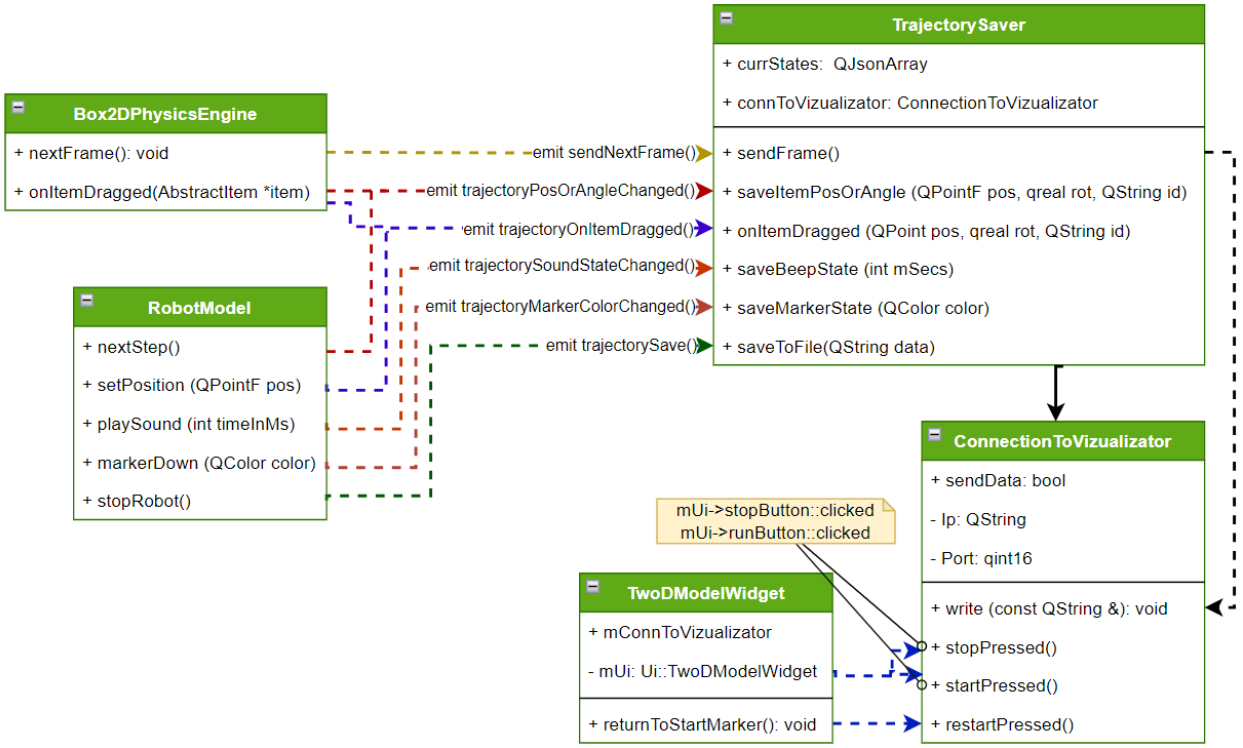

Рис. 6: Архитектура проекта (TRIK Studio).

зацию, используя файл с готовой траекторией), RunInRealTime (запускает сервер, который получает данные о траектории в реальном времени), Pause и Restart.

2. TRIK Studio (Рис. 6). Загрузка сцены из файла уже поддерживается, поэтому остаётся реализовать только сохранение траектории. За это отвечает класс TrajectorySaver, который сохраняет траекторию в виде набора фреймов (кадров). Каждый кадр — состояние сцены в определённый момент воспроизведения. TrajectorySaver записывает в потоки данные во время реализации методов saveItemPosition() и saveItemRotation() (сохраняет позицию и/или поворот объекта), saveBeepState() (сохраняет продолжительность воспроизведения звука) и так далее. Эти методы вызываются из классов, которые непосредственно имеют доступ к объектам.

4.2. Перенос сцены из Unity в TRIK Studio

TRIK Studio уже поддерживает функцию сохранения и загрузки сцен в формате XML, поэтому этот же формат был выбран для записи сцены из Unity. Создадим тестовый пример сцены — бильярдный стол (Рис. 7) Для корректного сохранения объекты на сцене должны быть помечены тегами. Тег — строка из определённого набора, по которым их можно находить и разбивать на группы. Примерами тегов могут быть «robot», «ball» и так далее. Таким образом, если несколько объектов (как в примере) помечены тегом «ball», то с помощью метода Unity FindGameObjectsWithTag() мы сможем найти их все. Некоторые объекты могут служить только для украшения, поэтому на них может не быть тегов вообще и они записаны не будут.

Рис. 7: Сцена в Unity.

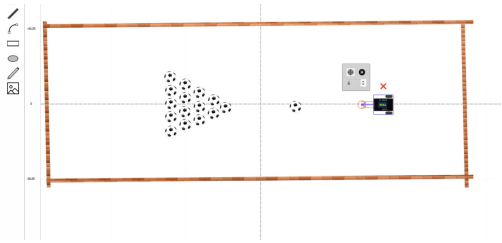

Рис. 8: Бильярдный стол в TRIK Studio.

Стоит сказать, что TRIK Studio обладает немного изменённой системой координат (ось Y направлена не вверх, а вниз), поэтому это нужно было учитывать при сохранении. Так как нас интересуют координаты X и Z (при переносе ось Z из Unity должна совпасть с осью Y в TRIK Studio), то координаты по Z следует взять с противоположным знаком. Если посмотреть на сцену в Unity сверху, то мы получим двухмерную картинку, как в TRIK Studio.

В проекте Unity хранится файл пустой сцены TRIK Studio в качестве образца. При вызове метода Export происходит поиск всех объектов, которые можно отобразить в TRIK Studio по тегам. По очереди записываются стены, мячи, кегли и робот. В первую очередь нас интересуют атрибуты объектов: их позиции и вращения.

Сохранённый файл легко открыть в TRIK Studio (Рис. 8).

4.2.1. Сохранение стен

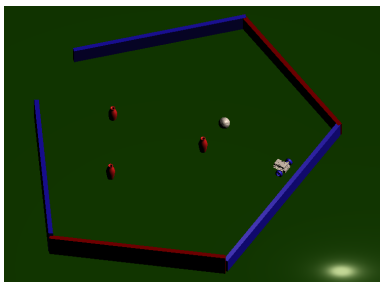

Рис. 9

Стены сохраняются особым образом. Допустим, у нас есть следующая сцена (Рис. 9)

Здесь находится несколько стен, которые повёрнуты под различными углами. Unity хранит позицию объекта как позицию его центра, но такой способ нам не подходит, так как TRIK Studio сохраняет и загружает стены, используя две точки: начало и конец. Однако у объектов на сцене есть свойство bounds — границы. Для любого объекта оно возвращает координаты параллелепипеда, описанного вокруг объекта. Для каждой стены находим это свойство и извлекаем только координаты по осям X и Z.

Из четырёх полученных точек нужно выбрать две противолежащие. Чтобы не ошибиться и сохранить стену, повёрнутую в нужную сторону, посмотрим на её угол поворота.

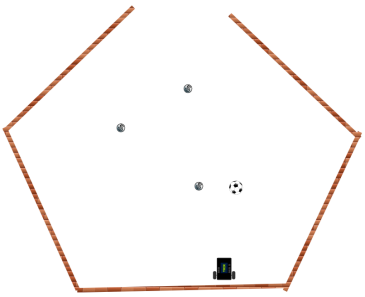

Рис. 10

- Получим угол в градусах (это важно, так как по умолчанию Unity использует кватернионы);
- Возьмём угол по модулю 180 (чтобы избежать лишних кругов, они будут мешать при сравнении с углом в 90 градусов);
- Если угол больше 90 градусов, берём левый нижний и правый верхний угол, если меньше — два других.

Видим, что стены были перенесены с учётом их поворотов (Рис. 10).

4.3. Перенос трассы из TRIK Studio в Unity

Допустим, программа для робота на сцене уже написана. Теперь нужно визуализировать её работу в Unity. Для точного воспроизведения необходимо каждый фрейм (кадр) сохранять текущее состояние робота и таким образом получать его траекторию. Кроме того, вспомним, что на сцене помимо робота могут присутствовать и другие движущиеся объекты (мячи и кегли). Их состояния тоже необходимо запоминать.


```
{"frame":[
    {"id":"trikKitRobot","state":"pos=-42.8151 36.2626"},
    {"id":"trikKitRobot","state":"rot=184.434"},
    {"id":"ball12","state":"pos=-111.004 -59.473"},
    {"id":"ball12","state":"rot=0"},
    {"id":"ball16","state":"pos=-119.295 -17.4914"},
    {"id":"ball16","state":"rot=0"}
]}
```


Рис. 11: Пример кадра, записанного в файле траектории.

Сохранять траекторию помогает система «сигналов и слотов», по сути реализующая паттерн «Наблюдатель». Она помогает связать между собой два класса, один из которых хранит «сигнал», а второй — «слот», то есть метод, который выполняется при вызове сигнала.

Класс Box2DPhysicsEngine (хранящий в себе ссылки на динамические объекты) и класс RobotModel (отвечающий за поведение робота) каждый кадр обновляют сцену и посылают сигналы классу TrajectorySaver. Эти сигналы могут содержать в себе параметры — в случае с отправкой сигнала об изменении позиции этими параметрами будут новые координаты объекта и его id. Вспомнив, что при переносе сцены из Unity в TRIK Studio пришлось изменить координаты по оси Z на противоположные, проделываем эту манипуляцию и в обратную сторону.

TrajectorySaver записывает полученные состояния в новый кадр, который добавляется к общему списку. Таким образом этот список можно создавать частями, добавляя кадры по мере их исполнения. Кадры сохраняются в формате JSON. (Рис. 11)

4.3.1. Запись линий и звука

Робот в TRIK Studio, в отличие от других объектов, умеет рисовать за собой линии разного цвета и издавать звуковые сигналы. Робот начинает чертить линию при вызове функции markerDown(). Эта функция посылает сигнал markerStateChanged(), который содержит в себе параметр — цвет маркера. При вызове markerUp() рисование линии прекращается, и отправляется такой же сигнал, параметром которого выступает прозрачный цвет. В файле траектории такое состояние записывается с помощью ключевого слова «markerState».

Состояние воспроизведения звука имеет ключевое слово «beepState».

4.4. Воспроизведение траектории из файла

Когда робот останавливается, он посылает сигнал, тем самым запуская метод записи списка кадров в файл, который можно воспроизвести в Unity, даже если TRIK Studio закрыта. За чтение и десериализацию файла из формата JSON в список кадров отвечает класс Deserializer. Он передаёт этот список классу ObjectManager. Это класс, принадлежащий специальному пустому объекту на сцене. Он хранит в себе ссылки на все динамические объекты аналогично Box2DPhysicsEngine в TRIK Studio.

Для воспроизведения используется система корутин [9], — это инструмент для работы с потоками в движке. В начале воспроизведения в классе ObjectManager запускается корутина Play(), номер текущего кадра currFrame равен нулю. В Play() происходит сравнение currFrame с числом кадров, которые хрянятся в ObjectManager. Если кадров больше, то у каждого объекта одновременно вызывается метод ReadLine(), передавая ему параметры из кадра под номером currFrame. Таким образом воспроизводится состояние сцены в один конкретный момент, а currFrame увеличивается на единицу.

Корутины работают таким образом, что им можно назначить выполнение после других корутин с помощью фразы yield return. Здесь это используется, и воспроизведение нового кадра начинается только после окончания предыдущего (yield return StartCoroutine (Play()).

4.5. Передача данных в реальном времени

Траектория может быть довольно сложной, и проезд робота может занимать достаточно долго времени. Утомительно каждый раз ждать, когда закончится проезд робота в TRIK Studio и сохранится траектория, поэтому хотелось бы научиться передавать данные в реальном времени. Это можно сделать, подключившись по сети к Unity из TRIK Studio. Если передача данных по сети активна, то после формирования каждого нового кадра происходит его отправка. Для этого используется класс ConnectionToVizualizator. Кадры можно накапливать и отправлять не один, а несколько за раз, если сеть достаточно загружена. При получении новых кадров Unity так же, как и при воспроизведении из файла, сначала передаёт их классу Deserializer, а затем классу ObjectManager.

4.6. Пауза и рестарт

При нажатии кнопки паузы воспроизведение программы останавливается. Кнопка рестарта возвращает все объекты на исходные позиции, если программа ещё не выполнилась до конца, то она продолжает свою работу с текущего места.

Работа паузы и рестарта отличается при разных способах получения траектории:

При воспроизведении из файла для паузы достаточно приостановить выполнение корутины с помощью метода StopCoroutine(), а затем возобновить его, используя команду StartCoroutine().

Для рестарта необходимо начать чтение файла с кадрами сначала, то есть приравнять номер текущего кадра к нулю.

Для передачи по сети приостановить корутину недостаточно, так как если в Unity проигрывание приостановится, то в TRIK Studio оно продолжится и новые кадры будут отправляться так же, как и до этого. Поэтому необходимо отправлять сигнал о паузе или рестарте, причём не только если пауза/рестарт нажаты в Unity, но и в обратную сторону. Последнее полезно, например, при рисовании линий, когда нажимается кнопка рестарта, а робот продолжает рисовать линию, или при воспроизведении звука, потому что, даже если приостановить корутину, внутренний таймер Unity продолжит работу.

Следовательно, ConnectionToVizualizator должен быть настроен не только на отправку, но и на получение данных. Он соединён системой сигналов и слотов с классом TwoDModelWidget, который частично обрабатывает пользовательский ввод, в частности, кнопки паузы и рестарта. При получении сигналов от Unity этот класс совершает требуемое действие, имитируя пользовательский ввод.

4.7. Перемещение объектов

Помимо визуализации траектории, хотелось бы переносить изменения сцены из TRIK Studio в Unity. Если, например, объект на сцене в Unity был перенесён пользователем, то, чтобы получить такое же перемещение в TRIK Studio, нужно произвести повторный экспорт сцены и её загрузку. В обратную же сторону изменения перенести сложнее.

Когда пользователь перемещает объект, ему присваиваются новые координаты, которые можно собрать и отправить в Unity.

Метод onItemDragged() из класса Box2DPhysicsEngine обрабатывает различные перемещение мячей и кегель пользователем. В RobotModel же в такой ситуации происходит присваивание координат с помощью методов setPosition() и setRotation(). Эти три метода связываются сигналами с методом onItemDragged() класса TrajectorySaver, который вызывается и для роботов, и для кегель одинаково.

Отличие onItemDragged() от setItemPosition() и setItemRotation() в том, что отправка кадра может происходить после получения каждого состояния, соответственно, в таком кадре будет всего одно состояние. Действительно, если программа не запущена, но происходит перемещение объекта, то nextFrame() не вызывается, и набор кадров отправить нельзя.

5. Апробация

5.1. Проведение апробации

Апробация проводилась в виде демонстрации решения задачи: был собран плеер на основе Unity с тестовой сценой — макетом дома. Сцена была загружена в TRIK Studio, а затем написана программа для задачи проезда по всем комнатам. Таким же образом дизайнер может передавать плеер на основе созданной сцены людям, у которых этот движок не установлен.

Визуализатор и траектория были показаны преподавателю Академии Робототехники RoboNest, руководителю лаборатории НТИ "Робототехнические системы", преподавателю курса "Мобильная робототехника и интеллектуальные робототехнические системы", организатору и судье олимпиад на базе платформы TRIK и среды программирования TRIK Studio. Клячину Алексею Михайловичу [6]. Он оценил работу так:

> *Результат работы полезен и интересен, особенно на соревнованиях по программированию роботов в TRIK Studio, где значительная роль уделяется демонстрации решений участников. Трёхмерный режим более наглядный и красочный, и не создаётся ощущения, что робот «движется по листу бумаги».*
> *Надеюсь, что в скором будущем программа будет массово применяться на соревнованиях и в школах, и мы увидим яркие сцены, сделанные дизайнерами из разных уголков мира.*

6. Заключение

Были выполнены следующие задачи:

- Был выполнен обзор существующих движков, выбран движок Unity;
- Разработана архитектура проекта;
- Реализована визуализация сохранённой трассы из TRIK Studio;
- Реализована визуализация трассы из TRIK Studio в реальном времени.
- Проведена апробация.

Ссылки на код:
https://github.com/ladaegorova18/TRIK_Studio_3D_Visualization
https://github.com/ladaegorova18/trik-studio

Список литературы

[1] Coppelia Robotics. CoppeliaSim. — 2021. — URL: https://www.coppeliarobotics.com/ (online; accessed: 2021-11-15).

[2] Design Inc. Logic. Robologix Home Page. — 2021. — URL: https://robologix.com/ (online; accessed: 2021-12-13).

[3] Dyn-Soft. Dyn-Soft RobSim 5. — 2018. — URL: http://robsim.dynsoft.ru/ (online; accessed: 2021-11-21).

[4] Epic Games. Unreal Engine. — 2021. — URL: https://www.unrealengine.com/en-US/ (online; accessed: 2021-12-15).

[5] Foundation Open Source Robotics. Gazebo Simulator. — 2021. — URL: https://www.gazebosim.org/ (online; accessed: 2021-11-20).

[6] RoboNest Академия робототехники. Преподаватели Академии робототехники. — 2022. — URL: http://www.robonest.ru/teachers (online; accessed: 2022-05-23).

[7] Scratch Foundation. Scratch. — 2021. — URL: https://scratch2.ru/ (online; accessed: 2021-12-13).

[8] TRIK Studio. TRIK Studio. — 2021. — URL: https://trikset.com/products/trik-studio (online; accessed: 2021-10-03).

[9] Unity. Coroutines Unity. — 2022. — URL: https://docs.unity3d.com/Manual/Coroutines.html (online; accessed: 2022-03-04).

[10] Unity Technologies. Unity. — 2021. — URL: https://unity.com/ru (online; accessed: 2021-11-21).

[11] Иванова Марина. Инфраструктура для разработки и отладки алгоритмов движения роботов. — 2017. — URL: https://oops.math.spbu.ru/SE/YearlyProjects/spring-2017/YearlyProjects/spring-2017/344/344-Ivanova-report.pdf (online; accessed: 2021-11-12).

[12] Приходько Станислав. Интеграция среды программирования роботов TRIK Studio со средой трехмерного моделирования V-REP. — 2016. — URL: https://oops.math.spbu.ru/SE/YearlyProjects/spring-2016/344/344-Prikhodko-report.pdf (online; accessed: 2021-11-10).