Міністерство освіти і науки України

Державний університет «Житомирська політехніка»

Факультет інформаційно-комп'ютерних технологій

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до випускної кваліфікаційної роботи бакалавра

на тему: «Гра-симулятор керування тролейбусом

(платформа Unity 3D)»

Виконав студент 4-го курсу, групи ПІ-53  
спеціальності 121 «Інженерія програмного забезпечення»

Іванов І.І.

Керівник Шатківський В. М.

Рецензент Єфремов Ю. М.

Житомир – 2020

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва прописними)

КАФЕДРА\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва прописними)

СПЕЦІАЛЬНІСТЬ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва прописними)

ЗАТВЕРДЖУЮ

Зав. кафедри \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

(назва кафедри)

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

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

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_\_ р.

**ЗАВДАННЯ**

на випускну кваліфікаційну роботу

Студента \_Іванова Івана Івановича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема роботи: \_«Гра-симулятор керування тролейбусом (платформа Unity 3D)»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Затверджена Наказом університету від «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 20\_\_\_ р. № \_\_\_\_\_\_\_\_

Термін здачі студентом закінченої роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Вихідні дані роботи (зазначається предмет і об’єкт дослідження) \_\_\_\_\_\_\_\_\_\_

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

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

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

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Консультант | Підпис, дата | |
| Завдання видав | Завдання прийняв |
| 1 | Шатківський В. М. | 2.03.18 | 2.03.18 |
| 2 | Шатківський В. М. | 6.04.18 | 6.04.18 |
| 3 | Шатківський В. М. | 4.05.18 | 4.05.18 |

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

**Календарний план**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів випускної кваліфікаційної роботи | Термін виконання етапів роботи | Примітка |
| 1 | Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел. | 2 березня 2018 – 5 квітня 2018 |  |
| 2 | Проектування структури системи | 6 квітня 2018 – 12 квітня 2018 |  |
| 3 | Написання програмного коду | 13 квітня 2018 – 26 квітня 2018 |  |
| 4 | Тестування | 27 квітня 2018 – 4 травня 2018 |  |
| 5 | Захист | 4 червня 2018 |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

РЕФЕРАТ

Зм.

Арк.

№ докум.

Підпис

Дата

Арк.

2

ІПЗ.КР.Б – 121 – 20 – ПЗ

Розроб.

В. Д. Ващенко

Керівник

.

Шатківський В. М.

н. контр.

О.П. Тичина

Зав. каф.

*А.В. Панішев*

**Гра-симулятор керування тролейбусом (платформа Unity 3D)**

**Пояснювальна записка**

Літ.

Аркушів

Житомирська політехніка,

група ПІ-50

Випускна робота бакалавра складається з програмного комплексу ігрової системи керування тролейбусом та пояснювальної записки. Пояснювальна записка до випускної роботи містить 58 сторінок, 37 ілюстрацій та 11 таблиць.

Метою роботи є розробка за допомогою рушія Unity гри-симулятора тролейбуса, що матиме можливість взаємодіяти з основними органами управління та рухатись по маршруту за графіком.

В роботі визначено основні завдання на розробку системи, проаналізовано аналоги розробленої системи. Наведено сценарії роботи програмного комплексу, загальна структура програмного комплексу, опис алгоритмів взаємодії окремих модулів системи, опис об’єктної структури системи (діаграма класів). Також наводиться результати тестування системи.

Ключові слова: ТРОЛЕЙБУС, СИМУЛЯТОР, ГРА, ТРАНСПОРТ, МАРШРУТ, ГРАФІК, ГРОМАДСЬКИЙ ТРАНСПОРТ.

ABSTRACT

Bachelor's graduate work consists of software complex of game system for simulating the trolleybus controlling and documentation. Documentation for bachelor’s graduate work contains 58 pages, 37 illustrations, and 11 tables.

The goal of the graduate work is to develop a game using Unity engine that simulates the trolleybus controlling process that includes interacting with main trolleybus controls and following the route’s schedule.

The work also contains main tasks of the system development, analogs of the developed system were analyzed. There are scenarios of the software package, the overall structure of the software system, description of module’s algorithms interaction, description of the system objects structure (class diagram). The results of the system testing are also present in the graduate work.

KEYWORDS: TROLLEYBUS, SIMULATOR, GAME, TRANSPORT, ROUTE, SCHEDULE, PUBLIC TRANSPORT.

ЗМІСТ

[РЕФЕРАТ 2](#_Toc514019495)

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 5](#_Toc514019497)

[ВСТУП 6](#_Toc514019498)

[РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СИМУЛЯЦІЇ КЕРУВАННЯ ЕЛЕКТРОТРАНСПОРТОМ 9](#_Toc514019499)

[1.1. ПОСТАНОВКА ЗАДАЧІ 9](#_Toc514019500)

[1.2. АНАЛІЗ АНАЛОГІВ ПРОГРАМНОГО ПРОДУКТУ 10](#_Toc514019501)

[1.3. ВИБІР АРХІТЕКТУРИ ІГРОВОГО СИМУЛЯТОРА 14](#_Toc514019502)

[1.4. ОБҐРУНТУВАННЯ ВИБОРУ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ТА ВИМОГИ ДО АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ 16](#_Toc514019503)

[ВИСНОВКИ ДО РОЗДІЛУ 1 19](#_Toc514019504)

[РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ СИМУЛЯЦІЇ КЕРУВАННЯ ЕЛЕКТРОТРАНСПОРТОМ 21](#_Toc514019505)

[2.1. ВИЗНАЧЕННЯ ВАРІАНТІВ ВИКОРИСТАННЯ ТА СТРУКТУРИ СИСТЕМИ. 21](#_Toc514019510)

[2.2. РОЗРОБКА СТРУКТУРИ ФАЙЛІВ СИСТЕМИ 33](#_Toc514019511)

[2.3. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ АЛГОРИТМІВ РОБОТИ СИСТЕМИ 37](#_Toc514019512)

[2.4. РЕАЛІЗАЦІЯ ФУНКЦІОНАЛУ СИСТЕМИ СИМУЛЯЦІЇ. 39](#_Toc514019513)

[ВИСНОВКИ ДО РОЗДІЛУ 2 44](#_Toc514019514)

[РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ СИМУЛЯЦІЇ 45](#_Toc514019515)

[3.1. ПОРЯДОК ВСТАНОВЛЕННЯ ТА НАЛАШТУВАННЯ СИСТЕМИ. 45](#_Toc514019516)

[3.2. СТРУКТУРА ІНТЕРФЕЙСУ. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ СИМУЛЯЦІЇ. 45](#_Toc514019517)

[3.3. ТЕСТУВАННЯ РОБОТИ СИСТЕМИ 53](#_Toc514019518)

[ВИСНОВКИ ДО РОЗДІЛУ 3 55](#_Toc514019519)

[ВИСНОВКИ 56](#_Toc514019520)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 58](#_Toc514019521)

[ДОДАТКИ 59](#_Toc514019522)

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

U3D – Unity3D.

ОС – Операційна система.

ЕТ – Електричний транспорт.

ТЕД – Тяговий електродвигун.

АХ – Автономний хід.

КМ – Контактна мережа (дроти живлення).

GUI – Graphic User Interface – графічний інтерфейс користувача.

ECS – Entity-Component-System – Сутність-Компонент-Система.

UE4 – Unreal Engine 4.

ВСТУП

Абсолютно кожен містянин хоча б раз користувався громадським транспортом. Автобус, трамвай, тролейбус чи метро присутні у повсякденному житті багатьох з нас, але значно менша кількість пасажирів замислюється як це керувати таким транспортним засобом. Без сумніву, керування автомобілями такого типу є складним процесом.

Навчання деяким важким та значущим професіям відбувається за допомогою симуляторів, тобто людина насправді не виконує певних дій, але тим самим все ж вивчає основні принципи роботи. Тож головним поставленим завданням було поповнити скарбничку таких симуляторів.

У багатьох країнах надають перевагу електричним видам транспорту перш за все через їх дружність до екології. Ще однією перевагою є саме електродвигун (надалі ТЕД), який в порівнянні із двигуном внутрішнього згоряння має набагато меншу кількість рухомих елементів і як наслідок – легше і дешевше обслуговування.

Електричні двигуни вже досить давно використовуються у транспортних засобах. Енергія, якою живляться ці двигуни, може надходити з найрізноманітніших джерел: від звичайних акумуляторів до екзотичних вітро- та фотогенераторів.

Одним із найочевидніших засобів живлення є пряме підключення до наземних генераторних станцій через контактну мережу (КМ), яка має вигляд звичайних дротів. Такий вид підключення використовується насамперед у тролейбусах, трамваях та електричних потягах і називається повітряною контактною мережею. Аналогічний за своєю ідеєю, але дещо видозмінений вид підключення використовується потягами метро, але роль дротів тут грає контактна рейка [1].

Ще одним варіантом отримання енергії для ТЕД є акумуляторні батареї, які стали досить популярні сьогодні. Однією з переваг такого живлення є незалежність від наявності КМ на дорозі, що активно використовують конструктори електробусів. Також акумулятори можна побачити і у сучасних тролейбусах, що надає їм можливість рухатись у куточки міста, де відсутня КМ, що значно розширює маршрутну сітку міста.

Крім найочевидніших джерел живлення електричних двигунів є менш очевидні, але не менш вживані, як от, наприклад, генерування електроенергії за допомогою двигуна внутрішнього згоряння (використовується у дизель-електричних локомотивах) або ж з використанням ядерної енергії, як на атомних підводних човнах.

Головним транспортним засобом для симуляції було обрано тролейбус, як типовий представник електротранспорту (надалі ЕТ). Симулятор дозволить вивчити базові навички керування тролейбусом не виходячи з дому, а виконання його у ігровій формі, можливо, стимулюватиме молоде покоління звертати увагу на дану професію як одну з опцій працевлаштування і підвищуватиме престиж робітничих професій.

Метою випускної роботи є проектування та розробка ігрових модулів симуляції керування тролейбусом на рушії Unity3D та об’єднання їх у програмний продукт.

Встановлена мета обумовлює наступні завдання:

* проведення аналізу задач симуляції керування транспортним засобом;
* визначення архітектури та узагальненої структури системи;
* обґрунтування та вибір засобів реалізації системи;
* розробка інформаційного та математичного забезпечення системи;
* проектування структурних складових та алгоритмів роботи системи;
* реалізація програмного комплексу симулятора.

Об’єктом дослідження є процес симуляції керування ЕТ, а саме тролейбусом.

Предметом дослідження є використання рушія Unity3D для розробки симулятора керування тролейбусом.

В роботі було використано методи об’єктно-орієнтованого проектування, об’єктно-орієнтованого програмування та дизайну інтерфейсів.

Розроблена система може бути використана для навчання базовим навичкам керування тролейбусом або ж просто для розваги.

Пояснювальна записка до випускної роботи містить 58 сторінок, 37 ілюстрацій та 11 таблиць, список літературних джерел містить 12 найменувань.

1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СИМУЛЯЦІЇ КЕРУВАННЯ ЕЛЕКТРОТРАНСПОРТОМ

## Постановка задачі

Основною метою провадження симулятора керування електротранспортом є: ознайомлення з принципами та методикою роботи з пристроями керування тролейбусом. Впровадження симулятора допоможе покращити навички управління транспортом молодим спеціалістам та зменшити кількість помилок та нештатних ситуацій при керуванні; прискорить процес опанування органами керування; скоротить час навчання та підвищить професійні навички. Система буде контролювати професійні компетенції водіїв.

В випускній роботі необхідно розробити додаток у вигляді гри-симулятора керування електротранспортом (тролейбусом), якомога більше наблизити процес керування до реалістичного. Для реалізації поставленого завдання, основними етапами є:

1. Розробити дизайн користувацького інтерфейсу.
   1. Створити загальну концепцію дизайну користувацького інтерфейсу.
   2. За допомогою графічного редактора Adobe Photoshop створити графічні зображення основних елементів керування (кнопок, панелей та ін.)
   3. Зверстати та застосувати елементи інтерфейсу за допомогою UnityEditor:
      1. Головне меню
      2. Панель ігрових налаштувань
      3. Панель завантаження/видалення гри, що міститиме список слотів для збереження гри
      4. Панель створення нової гри (вибору ігрової карти)
      5. Панель меню паузи
      6. Панель ігрового користувацького інтерфейсу
2. Проектування та розробка основних ігрових модулів
   1. Модуль керування персонажем
   2. Модуль керування транспортним засобом
   3. Модуль маршруту
   4. Модуль погоди
      1. Керування погодою
      2. Завантаження погоди з сервера
   5. Модуль вуличного освітлення та світлофорів
3. Моделювання 3D об’єктів за допомогою Blender3D
   1. Транспортні засоби
   2. Вуличні об'єкти
   3. Вуличне освітлення
   4. Будівлі
4. Текстурування 3D об’єктів
   1. Створення матеріалів у Substance Designer
   2. Текстурування, використовуючи Substance Painter

Результатом реалізації поставленого завдання є ігрова система симуляції роботи електротранспорту, а саме тролейбусу.

## Аналіз аналогів програмного продукту

Взагалі, симуляція – це імітація певної речі, ситуації чи процесу, отже є досить велике різноманіття симуляторів і використовуються вони з різною метою:

* тренування;
* навчання персоналу;
* тестування певної технології;
* для розважальних цілей.

Зазвичай, симулятори застосовують тоді, коли взаємодія з реальним об’єктом зв’язана з недосяжністю, небезпекою чи високою вартістю такої взаємодії [2].

В основному, симулятори поділяють на два види:

* Навчальні або тренувальні симулятори;
* Розважальні (ігрові) симулятори.

Серед ігрових симуляторів великої популярності набули авто- та авіасимулятори, що надають можливість керувати літаком чи автомобілем не виходячи з кімнати. Для підсилення ефекту симуляції додатково можуть використовуватись як додаткові контролери, такі як кермо, геймпади або джойстики, так і цілі конструкції у вигляді гоночних сидінь із гідравлічним механізмом, шоломи віртуальної реальності та багато інших.

Якщо ж говорити саме про симулятори громадського транспорту, то зазвичай вони поділяються на ті, що імітують керування маршрутною сіткою певного міста, тим самим даючи настанови одночасно усім учасникам дорожнього руху і ті, що дають можливість безпосередньо керувати транспортним засобом. До першої групи належать серія Cities In Motion, Transport Fever. До останніх належать OMSI – Der Omnibussimulator, Trancity, серія Bus Simulator. Найбільший інтерес в рамках дипломного проекту являє друга група. Їх порівняльна характеристика наведена у табл. 1.1.

Таблиця 1.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва | Фізичний рушій | Платформа | Зручність GUI | Графічне оформлення | Підтримка 64-розрядних ОС |
| OMSI | Добре | Windows | Погано | Нормально | Ні |
| Trancity | Погано | Windows | Відсутній | Погано | Ні |
| Bus Simulator 2016 | Погано | Windows, MacOS, Linux, Xbox, PS | Добре | Добре | Так |

OMSI – Der Omnibussimulator. Програмний продукт від німецького розробника MR-Software GbR. Характеризується відмінною фізичною моделлю транспорту, що ставить його на перше місце серед йому подібних, адже поведінка автомобіля на дорозі є чи не найбільш важливим критерієм у оцінці симуляції. Водночас має відносно гіршу систему фізичної взаємодії транспортного засобу з оточуючим світом та практично відсутній графічний інтерфейс користувача, що значно сповільнює ознайомлення із продуктом. Присутня підтримка контролерів та інших засобів поліпшення ігрового досвіду. З представників громадського транспорту наявний автобус та трамвай. Відсутня підтримка 64-розрядних ОС.



Рис. 1.1. OMSI

Trancity. Присутні три види транспорту – автобус, трамвай та тролейбус. Симуляція здійснюється поверхнево, але основні функції присутні. Легко модифікується, але фреймрейт постійно нестабільний через погано оптимізований програмний код, що разом із повністю відсутніми елементами керування у користувацькому інтерфейсі псує відчуття від ігрової сесії. Також відсутня підтримка 64-розрядних ОС.



Рис. 1.2. Trancity

Серія Bus Simulator. Якісний програмний продукт від ще одного німецького розробника Stillalive Studios. У останніх версіях має ряд переваг, що досить суттєво виділяє його на фоні конкурентів: сучасне графічне оформлення, користувацьке прокладання маршрутів (присутнє у Bus Simulator 2016) та інші. Кросплатформений, можливий запуск як на різних ОС (Windows, MacOS, Linux), так і на ігрових консолях Play Station та Xbox. Наявний мультиплеєр, хоч і має свої недоліки. Серед недоліків також погана оптимізація, що майже унеможливлює запуск симулятора на середніх ПК.



Рис. 1.3. Bus Simulator 2016

Порівняємо аналоги за присутністю видів транспорту.

Таблиця 1.2.

Присутність видів транспорту

|  |  |  |  |
| --- | --- | --- | --- |
| Назва | Тролейбус | Трамвай | Автобус |
| OMSI | Ні | Так | Так |
| Trancity | Так | Так | Так |
| Bus Simulator 2016 | Ні | Ні | Так |

Розглянемо представлені аналоги в розрізі доступних внутрішньоігрових можливостей.

Таблиця 1.3.

Функціональні можливості

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Назва | Навчання | Взаємодія з окремими елементами транспорту | Економічна складова | Можливість вільно рухатись гравцем | Підтримка користувацьких модифікацій | Мультиплеєр |
| OMSI | Ні | Так | Ні | Ні | Так | Частково |
| Trancity | Ні | Ні | Ні | Ні | Так | Ні |
| Bus Simulator 2016 | Так | Так | Так | Так | Так | Частково |

Таблиця 1.4.

Мінімальні системні вимоги порівнюваних аналогів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Процесор | RAM | Відеокарта | Місце на диску |
| OMSI | 2.6 GHz | 2 GB | Мінімум 512 MB відеопам’яті | 3 GB |
| Trancity | 1.6 GHz | 1 GB | Мінімум 256 MB відеопам’яті | 150 MB |
| Bus Simulator 2016 | Intel Core i3 3.3 GHz / AMD Phenom II X4 96OT 3.0 GHz | 4 GB | Мінімум 1 GB відеопам’яті | 5 GB |

Переглянувши та проаналізувавши аналоги симулятора громадського транспорту, можна із впевненістю сказати, що кожен програмний продукт суттєво відрізняється від своїх конкурентів, що створює певну різноманітність та свободу вибору для користувача, хоча містить ряд недоліків.

Враховуючи дані протиріччя виникає потреба спроектувати та розробити ігрові модулі гри-симулятора керування тролейбусом, об’єднавши їх у програмний продукт.

Таким чином, основними рисами даного продукту мають бути: сучасний фізичний рушій, наявність зручного GUI, можливість вільно пересуватись гравцем ігровим світом, взаємодія із дрібними ігровими елементами. Також додатково можна реалізувати: навчальну методику, економічну складову, підтримку користувацьких модифікацій.

## Вибір архітектури ігрового симулятора

Для розробки даної системи симуляції є можливість використати наступні патерни:

* MVC
* ECS
* MVP
* MVVM

Адаптування паттернів типу MVC, MVVM чи MVP під розробку ігрового продукту в результаті могло б спричинити плутанину серед об’єктів GameObject та їх батьківських елементів, що є неприйнятним для розробки на ігровому рушії гри такого типу.

Оптимальним для реалізації симулятора буде використання шаблону проектування Сутність-Компонент-Cистема.

System

Entity

Component

Component

Component

Component

Entity

Component

Component

Component

Component

Entity

Component

Component

Component

Component

Entity

Component

Component

Component

Component

Рис. 1.4. Entity-Component-System

Сутність-Компонент-Система або Entity-Component-System(ECS) – це шаблон проектування, що, зазвичай, використовується при розробці ігор. ECS зебезпечує величезну гнучкість в проектуванні загальної архітектури програмного забезпечення. Великі компанії, як Unity Technologies, Epic Games чи Crytek використовують цей шаблон у якості основного у своїх фреймворках [3].

Сутності, компоненти та системи – три основних поняття цього шаблону, які зв’язані між собою досить вільно.

Сутності зазвичай використовуються для створення унікального ідентифікатора, надання середовищу інформації про існування окремого елемента і функції як свого роду кореневого об'єкту, що об'єднує безліч компонентів.

Компоненти - це об'єкти, що не володіють ніякою складною логікою. В ідеалі вони є об'єктами з простою структурою даних. Кожен тип компоненту можна прикріпити до сутності, надавши їй властивість. Припустимо, до сутності можна прикріпити «Vehicle-Component», що дозволить зробити сутність засобом для пересування, налаштувавши основні властивості компонента.

Що ж стосується систем, то думки щодо їх визначення відносно сильно різняться. Існує думка, що системам відомі тільки компоненти. Крім того, існує думка, що для кожного типу компонента повинна бути своя система, наприклад для «Vehicle-Component» повинна бути «Vehicle-System», для «Weather-Component» - «Weather-System» і т.д. Такий підхід досить суворий і не враховує взаємодію різних компонентів. Менш строгий підхід полягає в тому, що різні системи мають справу з компонентами, про які їм має бути відомо. Наприклад, «Physics-Systems» повинна знати про «Collision-Components» і «Rigidbody-Components», тому що обидві вони швидше за все містять необхідну для симуляції фізики інформацію. Якщо ж узагальнити, система – це певний об’єкт, що містить посилання чи просто у вигляді дочірніх елементів набір сутностей (або компонентів) які він контролює.

Взаємодія цих трьох елементів може здійснюватися різноманітними способами і повністю залежить від потреб та реалізації.

Таким чином, можна зробити висновок, що даний архітектурний шаблон забезпечує всі технологічні можливості для побудови системи, що дозволяє реалізувати поставлені завдання .

## Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Рушій. Найбільш популярними рушіями для розробки ігор сьогодні є Unity3D, UE4, Marmalade і Cocos2D. У кожного з них є свої переваги. Але оскільки необхідно розробити кроссплатформенну гру-симулятор з використанням фізичного рушія, то на першому місці за актуальними дослідженнями знаходиться Unity. Визначальним є те, що рушій дає можливість створення гри на різні платформи з відносно сучасним графічним оформленням та фізичним рушієм Nvidia PhysX. На даний час у Unity з'явився реальний конкурент – Unreal Engine 4 (UE4).

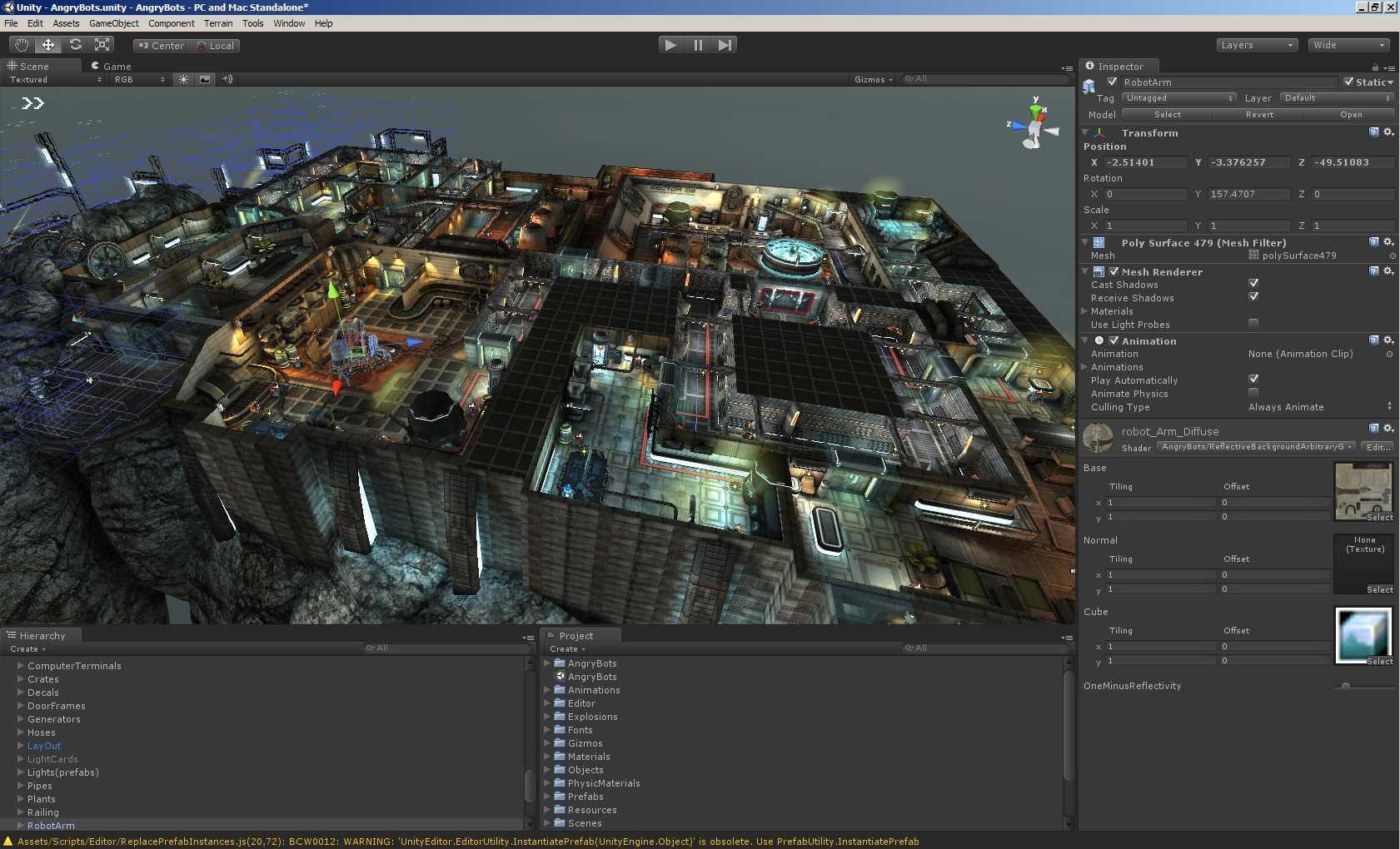


Рис. 1.5. Unity Editor

Вибір даного рушія обумовлений наступними перевагами.

Перша перевага – розробка ведеться мовою C#. Дана мова високорівнева і дозволяє програмісту легко увійти в розробку гри. Це важливий момент, тому що на відміну від інших рушіїв, де використовується мова C++, як от у UE4, в C# для Unity є багато елементів і прийомів, які вже реалізовані, і програмісту потрібно тільки скористатися ними [4].

Друга перевага - кроссплатформеність, тобто один і той же код, написаний на рушії Unity, з мінімальними змінами може бути перенесений на різні платформи (PC, Mac, Android, iOS, Web, ігрові консолі). Це суттєво скорочує час на розробку гри.

Третя перевага - гарне Community. Це означає, що у різних функцій рушія є чіткий опис з прикладами на сайті розробника, звернутися до якого можна в будь-який момент. Якщо щось все ж залишилося незрозумілим, служба підтримки обов'язково відповість на питання, що виникають.

3D-редактор. Blender3D зайвий раз доводить, що безкоштовне ще не означає погане. Проект Blender активно розвивається, не поступаючись ні в чому комерційним аналогам. Секрет успіху криється в тому, що до роботи над Blender може приєднатися будь-хто. Багато з інструментів, які з'явилися в цій програмі, були додані абсолютно різними людьми, які створювали ті чи інші функції для вирішення своїх завдань [5].



Рис. 1.6. Blender3D

У порівнянні з комерційними розробками розмір цього редактора версії 2.79b складає всього 86 мегабайт.

Однією з найголовніших переваг програми є кросплатформеність. Blender однаково добре і стабільно працює в Linux і Windows. Крім того, програма може функціонувати навіть на ПК з дуже слабкими конфігураціями. Мінімальні вимоги до системи: процесор з одним ядром, що працює на частоті 1 ГГц, оперативна пам'ять 512 Мбайт і відеокарта з підтримкою Open GL і обсягом пам'яті не нижче 64 Мбайт.

Програма включає в себе великий арсенал засобів для створення професійної тривимірної графіки:

* оперування системами частинок,
* контроль ваги окремих частинок при текстуруванні
* застосовування напрямних при анімації
* використання зовнішніх сил, наприклад вітру.

Редактор текстур.Текстурування створених моделей можна здійснювати і всередині 3D-редактора, але це дуже кропітка і довга справа. Щоб оптимізувати роботу доцільно використовувати спеціальні інструменти для текстурування, а саме Substance Painter. Дане програмне рішення спрощує та пришвидшує обробку моделі у декілька разів. Substance Painter має досить інтуїтивно зрозумілий інтерфейс та надзвичайно широкі можливості з кастомізації матеріалів [6]. Також є можливість створювати матеріали у окремому редакторі Substance Designer.

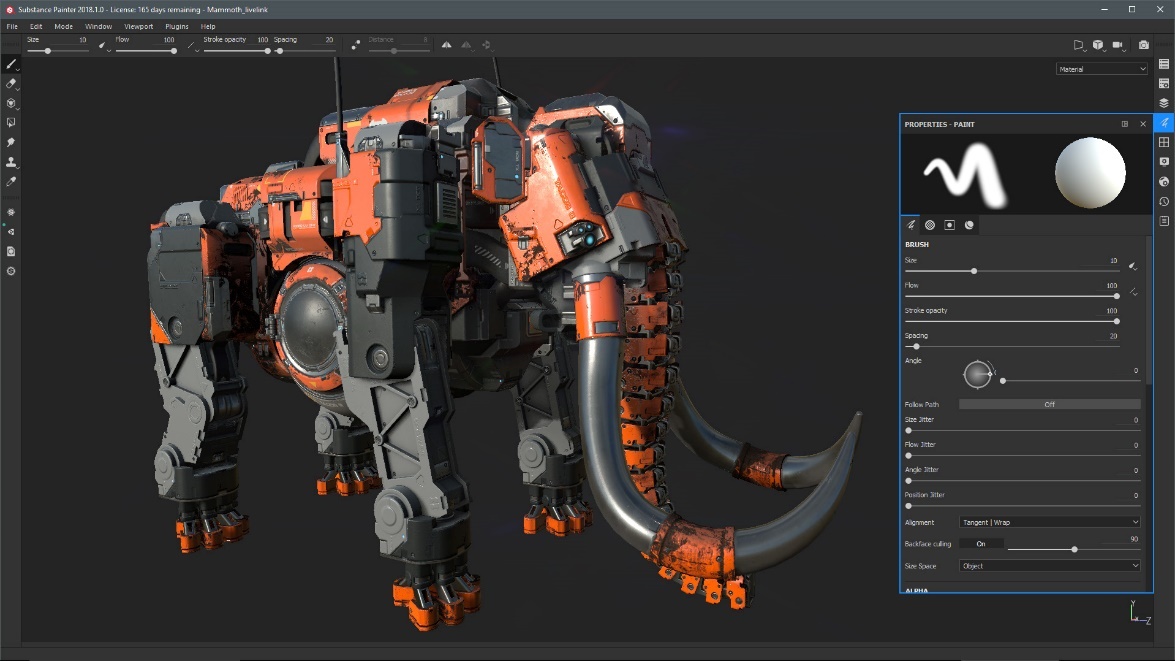


Рис. 1.7. Substance

Отже, використано такі програмні продукти та середовища розробки: рушій Unity 5.5.3f та Unity 2017.3.1; середовище розробки Visual Studio 15; графічний редактор Adobe Photoshop CC; 3D-редактор Blender3D 2.79; Substance Painter, Substance Designer 6.

## Висновки до розділу 1

Проведений аналіз предметної області дозволив визначити основні аспекти симуляції громадського транспорту. Огляд наявних аналогів та виявлення протиріч між наявними можливостями та потребами показав, що головними функціями майбутньої системи має бути: зручність GUI, наявність навчальних елементів, взаємодія з окремими елементами транспорту, вільне пересування по ігровій області. Наявність підтримки користувацьких модифікацій, економічна складова та мультиплеєрний режим є також бажаними функціями для таких програмних систем.

Визначившись з оптимальною архітектурою, обрано необхідний та достатній набір програмних продуктів для реалізації поставлених задач: рушій, 3D-редактор, редактор текстур.

1. [ПРОЕКТУВАННЯ](file:///C:\Users\Anna\AppData\Roaming\Skype\бакалаврат\fragment_bakalavr.docx#_Toc212894718) СИСТЕМИ СИМУЛЯЦІЇ КЕРУВАННЯ ЕЛЕКТРОТРАНСПОРТОМ

## [Визначення](file:///C:\Users\Anna\AppData\Roaming\Skype\бакалаврат\fragment_bakalavr.docx#_Toc212894719) варіантів використання та структури системи.

Система симуляції керування електротранспортом створюється з метою максимально реалістичного відтворення процесу керування тролейбусом у віртуальному світі.

*Вимоги користувачів.*

Користувачі мають доступ до наступних функцій:

1. Керування тролейбусом;
2. Вибір маршруту для руху;
3. Зберігання поточного ігрового прогресу;
4. Завантажування поточного ігрового прогресу;
5. Налаштування ігрових параметрів;
6. Налаштування графічних параметрів;
7. Налаштування звукових параметрів;
8. Зміна призначень клавіш для ігрових дій;



Рис. 2.1. Варіанти використання

*Функціональні вимоги:*

1. Геймплей: користувач матиме можливість керувати тролейбусним транспортом – у режимі вільної їзди та у режимі маршруту. При керуванні тролейбусом у режимі маршруту необхідно буде дотримуватись графіку руху.
2. Не потрібно перебувати онлайн: користувач може грати без підключення до Інтернету в довільний момент часу.

*Нефункціональні вимоги:*

1. Сприйняття:

* Час, потрібний для навчанння роботи з програмою для звичайних користувачів — 30 хвилин, для досвідчених — 10 хвилин.
* Час відповіді системи для звичайних запитів не повен перевищувати 0.016 секунди, а для більш складних запитів — 0.03 секунди.
* Інтерфейс симулятора повинен бути інтуїтивно зручним для користувача та не вимагати від нього додаткової підготовки.
* Надійність:
* Доступність - час, потрібний для обслуговування системи не повинен перевищувати 10 % від загального часу роботи.
* Середній час безперервної роботи — 5 годин.
* Максимальна норма помилок та дефектів в роботі програми - 1 помилка на 1000 запитів користувача.

1. Продуктивність. Гра повинна підтримувати безперебійну роботу користувача.
2. Можливість експлуатації:

* Масштабування – після оновлення все повинно працювати без збоїв, не мають виникати жодні помилки при розширенні гри.
* Оновлення версій – оновлення версій повинно здійснюватися за допомогою патчу.

Аналіз функціональних вимог дозволив виділити наступні сутності, що забезпечать реалізацію програмного комплексу системи. На рис.2.2. наведено діаграму класів, які використовуються для керування ігровою сценою (світом).

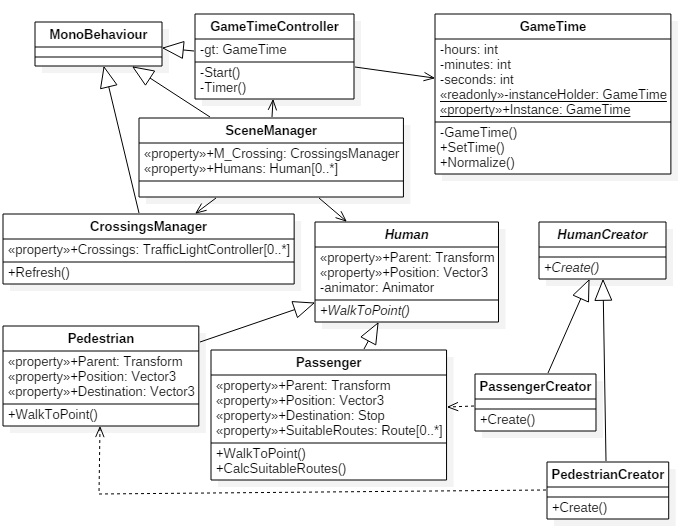


Рис. 2.2. Діаграма класів сцени

В даній діаграмі можна виділити класи, які спроектовано, використовуючи шаблони “Фабричний метод” та “Одинак”.

Фабричний метод – породжуючий шаблон проектування, що надає підкласу можливість створення екземплярів деякого класу. В момент створення підкласи можуть визначити, який клас створювати. Іншими словами, даний шаблон надає право створення об'єктів батьківському класу. Це дозволяє використовувати в коді програм не специфічні класи, а маніпулювати абстрактними об'єктами на більш високому рівні [7]. Використано для реалізації створення пішоходів та пасажирів. Таким чином, маємо:

* загальний клас Human з відповідними йому властивостями і методами;
* дочірні класи Passenger та Pedestrian, що успадкували елементи батьківського класу та придбали нові власні;
* загальний клас для створення HumanCreator;
* дочірні класи PassengerCreator та PedestrianCreator, методи яких будуть використані для інстанціювання пасажирів та пішоходів.

Одинак (англ. Singleton) - породжуючий шаблон проектування, гарантує, що в однопроцессному додатку буде єдиний екземпляр деякого класу, і надає глобальну точку доступу до даного екземпляру. У класі є лише один екземпляр, і він надає йому глобальну точку доступу [8]. Використано у класі “GameTime” та у деяких інших модулях, наприклад у модулі головного меню присутній клас InputManager, що також має singleton у своїй основі.

Насправді, існує деяка вірогідність, що екземпляр класу у системі буде не єдиний через багатопотоковість. Саме тому використовувати екземпляр класу, спроектованого на основі Одинака необхідно обережно і зважати на те, що результат, отриманий після звертання до одинака у окремому потоці може бути зовсім іншим за той, що очікується.

Система спроектована на основі архітектурного патерну Entity-Component-System (ECS). На Рис. 2.3 представлено діаграму класів модулю, який використовується для керування маршрутами у ігровому світі.

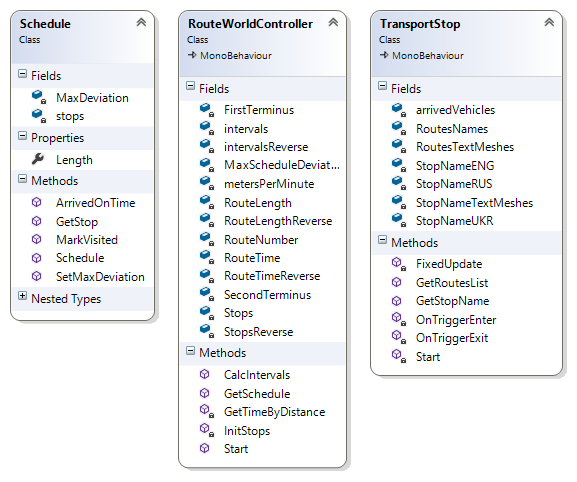


Рис. 2.3. Діаграма класів модулю «Маршрути»

Як відомо, патерн ECS містить три основних компоненти: Сутність, Компонент і Система. На даному рисунку RouteWorldController – клас-скрипт, який описує маршрут, яким рухатиметься гравець:

* Stops – масив типу TransportStop. Містить набір зупинок від першої кінцевої до другої;
* StopsReverse – масив типу TransportStop. Містить набір зупинок у зворотньому напрямку;
* Intervals та IntervalsReverse – масиви, що містять інтервали між зупинками. Можна задати вручну або ж розрахувати автоматично за допомогою методу CalcIntervals ();
* FirstTerminus та SecondTerminus – назва першої та другої кінцевої;
* RouteNumber – назва маршруту;
* RouteLength, RouteLengthReverse – приблизна довжина прямого та зворотнього рейсів у метрах. Розраховується при виконанні автоматичного призначення інтервалів між зупинками.
* RouteTime, RouteTimeReverse – приблизна довжина прямого та зворотнього рейсів у хвилинах. Розраховується при виконанні автоматичного призначення інтервалів між зупинками.
* MaxScheduleDeviation – максимальне відхилення від графіку; Використовується при перевірці на вчасність прибуття на зупинку;
* MetersPerMinute – швидкість руху по маршруту. Вимірюється у метрах за хвилину. Використовується в розрахунку інтервалів;
* CalcIntervals () – розрахунок прямих та зворотніх інтервалів;
* GetSchedule () – повертає новий графік, оснований на часі початку графіку та інтервалах;
* GetTimeByDistance () – перетворює відстань у час. Розрахунок оснований на швидкості руху по маршруту;
* InitStops () – первинна ініціалізація зупинок.

TransportStop – клас-скрипт, що контролює зупинку та її функції:

* ArrivedVehicles – містить список всіх транспортних засобів, які прибули на зупинку (потрапили в зону її дії);
* RoutesNames – список маршрутів, що проходять через дану зупинку;
* RoutesTextMeshes, StopNameTextMeshes – списки компонентів TextMesh, які використовуються для написів всередині ігрового світу. У даному випадку виводиться назва зупинки та маршрути, які через дану зупинку проходять;
* StopNameENG, StopNameRUS, StopNameUKR – назва зупинки у різних локалізаціях;
* GetRoutesList () – повертає список маршрутів, які проходять через дану зупинку;
* GetStopName () – повертає назву зупинки відносно поточної локалізації;

Schedule – клас, що описує графік руху:

* MaxDeviation – максимальне відхилення від графіку;
* Stops – масив зупинок, що необхідно відвідати за графіком;
* Length – довжина графіку (кількість зупинок);
* ArrivedOnTime () – повертає булеве значення на основі перевірки відвіданості зупинки.
* GetStop () – повертає зупинку по індексу;
* MarkVisited () – відмічає зупинку по індексу як відвідану;
* SetMaxDeviation () – встановлює максимальне відхилення від графіку;

Також є клас Stop, що використовується класом Schedule:

* ArrivedAt – фактичний час прибуття на зупинку;
* ExpectedAt – час прибуття на зупинку за графіком;
* Controller – скрипт TransportStop, що відповідає поточній зупинці.
* Visited – була відвідана зупинка чи ні;
* ArrivedOnTime – прибуття було в рамках відхилення чи ні.

Отже, при даній ієрархії та паттерні ECS, RouteWorldController виступає у ролі Системи, що контролює два набори зупинок – Сутностей, які виступають контейнерами для TransportStop та інших Компонентів.

На рис.2.4 зображено діаграму класів модулю, що контролює погодні умови у грі.

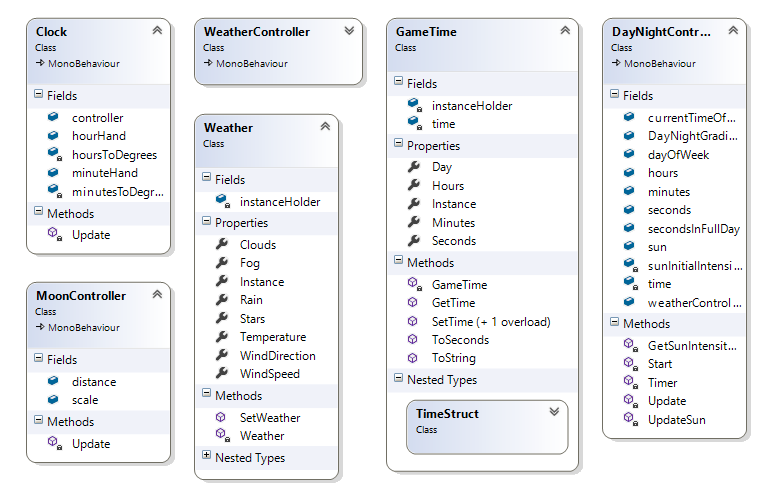


Рис. 2.4. Діаграма класів модулю «Погода»

Основним класом-скриптом у даному модулі є WeatherController, що містить велику кількість посилань на системи частинок, які емулюють погоду (хмари різних типів, дощі і тому подібне). Для видобування параметрів погоди використовується клас-одинак Weather, що містить наступні властивості:

* Clouds – властивість перелічувального типу, що описує тип хмар;
* Fog - властивість булевого типу, що відповідає наявності туману;
* Rain - властивість перелічувального типу, що описує тип дощу;
* Stars - властивість булевого типу, що відповідає наявності зірок;
* Temperature – температура повітря;
* WindDirection, WindSpeed – напрям і сила вітру.

Крім того, було створено клас-скрипт, що контролює поведінку сонця і місяця у ігровому світі відносно поточного часу доби – DayNightController, який містить посилання на Gameobject сонця і місяця, інтенсивність світіння, кількість секунд у повному ігровому дні (за замовчуванням рівне реальному значенню) та інші параметри, необхідні для роботи скрипта. Сам відлік часу реалізовано за допомогою корутіни.

Корутіна (Coroutine) – метод всередині класа-скрипта, який повертає особливий тип – IEnumerator. Особливість даного методу полягає в тому, що в тіло методу можна повернутись після досягнення певної умови, наприклад:

yield return new WaitForSeconds(10f);

sun.intensity = 1.5f;

Тут інтенсивність зміниться тільки після того, як пройдуть 10 секунд ігрового часу. Звичайно, під час очікування є можливість користуватись системою у звичайному режимі. Також можна ставити умови іншого типу, наприклад використання yield return new WaitUntil(()=>isDoor1Open) буде чекати доти, доки isDoor1Open не набуде значення true. Схожий, але протилежний ефект буде від використання методу WaitWhile(()=>isDoor1Open). При написанні таких складних умов використовуються лямбда-вирази.

Скрипт DayNightController використовує клас-одинак GameTime, який містить методи для задання, форматування часу та клас, що описує ігровий час – TimeStruct, який містить властивості для певних часових параметрів, день тижня, перевантаження операторів для зручних математичних операцій над часом та інші.

Окрім того DayNightController використовує посилання на скрипт MoonController, що містить базові параметри для місяця та його позиціонування.

Також у модулі наявний скрипт Clock, який можна використати для реалізації годинника із стрілками у ігровому світі. Скрипт містить всі необхідні параметри для того, щоб легко та швидко використати його у проекті.

Модуль головного меню включає в себе декілька підмодулів:

* Основні елементи GUI;
* збереження/завантаження гри;
* керування ігровими налаштуваннями;
* візуальні ефекти для елементів GUI та для сцен в цілому.



Рис. 2.5. Класи модулю головного меню

Для ігрового продукту однією з найважливіших складових є графічний користувацький інтерфейс (GUI). Саме основні скриптові елементи GUI входять у перший підмодуль (Рис. 2.6).



Рис. 2.6. Підмодуль GUI

В даний підмодуль входять десять класів, всі вони є скриптами і кожен з них контролює один конкретний елемент GUI, або ж повністю панель інтерфейсу. Під панеллю інтерфейсу необхідно розуміти тип елементу користувацького інтерфейсу Canvas у U3D, на якому знаходяться елементи керування. Наприклад, основний Canvas містить три кнопки – Грати, Налаштування і Вихід. Таким чином, головною панеллю керує скрипт MainMenuUI. До аналогічних скриптів відносяться:

* PauseUI;
* SettingsUI;
* NewGameUI;
* KeyBindingUI.

Всі вони мають посилання на окремі елементи керування (кнопки, слайдери, випадаючі списки) та методи, що контролюють їх поведінку.

Окремо можна виділити скрипт LoadingScreen, який також відноситься до поточного підмодулю, але має дещо інший підхід до реалізації. Основний процес, що відбувається у скрипті – завантажування наступного рівня – відбувається всередині корутіни (Рис 2.7). Саме завантаження відбувається асинхронно.

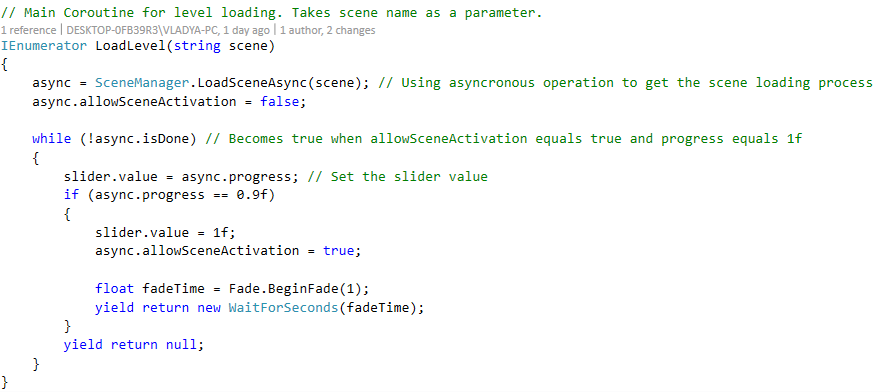


Рис. 2.7. Головна корутіна скрипту LoadingScreen

Таким чином, поки відбувається завантаження сцени відбувається зміна значення слайдеру, який використовується для індикації прогресу завантаження. Після того, як значення прогресу буде рівним 0.9, завантаження завершується, та вмикається вицвітання і перехід до наступної сцени.

Також у модулі наявні скрипти, що контролюють списки та елементи списків. До таких належать MapItemController, KeyListItemController, SaveItemController та SavesListController. Насамперед, вони містять властивості певних елементів, наприклад, SaveItemController має у своєму складі властивості збереження: слот для зберігання, деякі дані ігрової статистики, назву карти та інше.

Підмодуль збереження та завантаження є дещо складнішим у функціональному плані (Рис. 2.9).

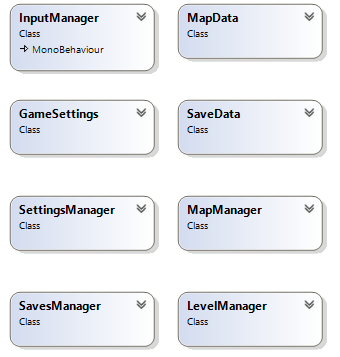


Рис. 2.8. Класи підмодулю збереження і завантаження.

Для збереження та завантаження даних використовується файли .xml, що є стандартним рішенням під час розробки ігрового продукту. Задля реалізації процесу контролювання файлів було створено основні класи SaveData та MapData. Кожен з них має відповідні менеджери, які побудовані на основі Singleton та дещо модифікованого патерну Репозиторій. Наприклад, SavesManager має всі необхідні методи для керування файлами збережень. Аналогічно і MapManager містить необхідні інструменти управління ігровими картами.

Клас LevelManager має схожу, і водночас відмінну реалізацію. Архітектура також побудована на основі Singleton, але містить дещо іншу систему керування – він шукає у певній папці нові карти і додає їх у проект за необхідності. Такий функціонал реалізовано задля майбутньої підтримки користувацьких модифікацій.

Для реалізації переприв'язки клавіш керування було написано скрипт InputManager, так як стандартний клас Input мав незадовільну реалізацію перепризначень. Скрипт було виконано на основі патерну Singleton.

Реалізація патерну «Одинак» для скриптів дещо відрізняється від її реалізації для звичайних класів [9]. Так як скрипт є компонентом і його можна прив’язати до будь-якої об’єкту на сцені є імовірність, що скрипт вже існує. Тому, необхідно переконатись, що об’єкт дійсно один на сцені і видалити інші, якщо вони присутні. Окрім того, для доступності об’єкту на всіх сценах використовується метод DontDestroyOnLoad (). Приклад реалізації на основі скрипту InputManager:

public static InputManager instance = null; // Object instance

void Awake()

{

// Check object exsistance (Singleton)

if (instance == null)

instance = this;

else if (instance == this)

Destroy(gameObject); // If there is another object -> delete it

// Set the object not to destroy when changing scenes

DontDestroyOnLoad(gameObject);

ManagerInitialization();

}

До складової візуальних ефектів (Рис. 2.8) належать два скрипти FadeUI та FadeScene.

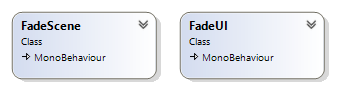


Рис. 2.9. Класи підмодулю візуальних ефектів.

FadeUI надає сутності компонента властивості вицвітання. За допомогою відповідного метода можна запустити процес вицвітання від нормального до прозорого та навпаки. Аналогічний ефект відбувається під час активації сцени, в якій є FadeScene: при завантаженні відображається вицвітання від чорного до нормального, а при вимкненні сцени від нормального до чорного.

## [Розробка структури](file:///C:\Users\Anna\AppData\Roaming\Skype\бакалаврат\fragment_bakalavr.docx#_Toc212894723) файлів системи

Для збереження інформації у системі використовуються файли типу XML. Вони мають подібну до HTML розмітку у вигляді тегів, дані у які можна записати двома способами: через атрибут та всередину парного тегу. У реалізації симулятора використовуються обидва способи.

У системі наявні такі види файлів:

* файл збереження;
* файл опису ігрової карти;
* файл налаштувань.

У системі використовується система бандлів (Bundles) – спеціальних архівів з об’єктами, що можуть бути підвантажені у систему протягом ігрового процесу. Під час збірки архіву відбувається пошук та пакування всіх необхідних файлів для роботи сцени, після чого архів можна перемістити у певну папку, з якої проект зможе її підвантажити за необхідності. Зазвичай, це папка StreamingAssets. Після виконання збірки все, що міститься в даній папці перенесеться в підпапку білду з ідентичною назвою.

Файл збереження (Рис. 2.10) реалізовано як файл, який описує ігровий слот з інформацією про збереження. Загалом слотів 10, гравець може розпочати нову гру на будь-якому з них. Всі слоти розміщуються в папці StreamingAssets проекту.



Рис. 2.10. Файл збереження

Файл організований у вигляді тегу-контейнеру SaveData, у який вкладено одиночний тег save, де атрибутами виступають дані, що зберігаються. Детальний опис параметрів наведено у таблиці 2.1.

Таблиця 2.1

Інформація про параметри тегу SaveData

|  |  |
| --- | --- |
| Назва атрибуту | Призначення |
| SaveIndex | Індекс слоту, у якому міститься збереження. |
| IsEmpty | Вказує на порожність слоту. |
| ImagePath | Назва та шлях до зображення. Зображення до ігрової карти, на якій відбувалась гра. По замовчуванню зображення шукається в папці Images поруч із файлом опису карти. |
| SceneName | Назва сцени для завантаження. Після вибору слоту відбувається завантаження сцени з цим іменем, якщо її було підвантажено через бандл. Якщо ж ні – перед цим шукається бандл з таким іменем і підвантажується. |
| MapName | Назва ігрової карти, на якій відбувалась гра. Лише для виводу у головному меню. |
| TimePlayed | Загальний час, проведений у грі. |
| XP | Ігровий досвід, набутий в процесі. |
| Money | Кількість внутрішньоігрової валюти у гравця. |

Файл опису ігрової карти створено для того, щоб мати певну мінімальну інформацію про сцену, яку є можливість завантажити. Він, як і збереження, міститься в папці StreamingAssets, але у випадку з підвантажуваними елементами необхідна додаткова ієрархія, яку показано на Рис. 2.11.

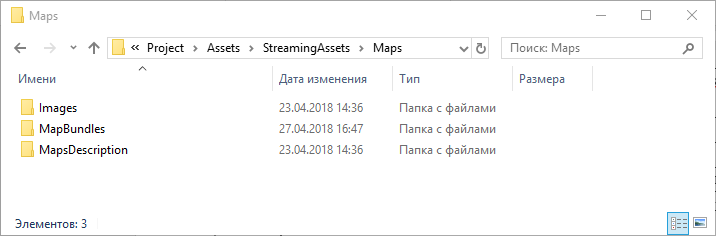


Рис. 2.11 Ієрархія папки StreamingAssets/Maps

Опис папок наведений у табл. 2.2.

Таблиця 2.2

Інформація про призначення елементів в папці Maps

|  |  |
| --- | --- |
| Назва папки | Призначення |
| Images | Зображення-ілюстрації до карт. |
| MapBundles | Папка з бандлами. |
| MapsDescription | Опис певної карти у вигляді файлів xml. |

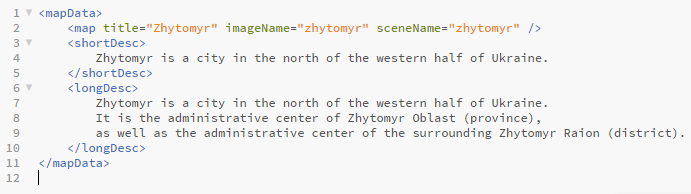


Рис. 2.12 Файл з описом карти

Детальний опис кожного тегу та атрибуту наведено у таблиці 2.3.

Таблиця 2.3

Інформація про призначення тегів та атрибутів файлів опису карти

|  |  |  |
| --- | --- | --- |
| Назва | Тип | Призначення |
| mapData | Тег | Основний тег-контейнер для інформації. |
| map | Тег | Тег для основних даних про карту. |
| title | Атрибут тегу map | Назва, яку буде показано в меню. |
| imageName | Атрибут тегу map | Назва зображення карти. Для читання необхідно, щоб зображення знаходилось в папці Images. |
| sceneName | Атрибут тегу map | Назва сцени, яка буде використовуватись для завантаження. |
| shortDesc | Тег | Короткий опис карти (Одне-два коротких речення). Буде показано на екрані початку нової гри, в елементі списку певної карти. |
| longDesc | Тег | Довгий опис карти. Показується після натиску на детальну інформацію про карту. |

Дані про призначення клавіш зберігаються іншим чином – за допомогою PlayerPrefs (Параметри гравця), який являє собою внутрішньорушійний інструмент для зберігання та доступу до параметрів, що були задані гравцем, між ігровими сесіями. Прямого доступу до файлу немає, адже значення знаходяться в реєстрі.

Дані про інші налаштування – графічні, звукові та ігрові зберігаються у json-файлі gameSettings.json в папці PersistentDataPath.



Рис. 2.13 Папка PersistentDataPath.



Рис. 2.14. Вміст файлу gameSettings.json

Так як використовується json-серіалізація відповідний клас також має бути наявний у системі.Структура та призначення елементів класу та, відповідно, файлу налаштувань наведена у таблиці 2.4.

Таблиця 2.4

Інформація про призначення елементів файлу ігрових налаштувань

|  |  |  |
| --- | --- | --- |
| Назва | Тип | Призначення |
| fullscreen | bool | Повноекранний режим. |
| resolutionWidth | int | Розширення екрану. Ширина. |
| resolutionHeight | int | Розширення екрану. Висота |
| textureQuality | int | Якість текстур:   * 0 – Стандартне розширення; * 1 – Половинне розширення; * 2 – Чверть розширення. |
| antialiasing | int | Режим згладжування. Значення відповідає кількості ітерацій для кожного пікселя. Максимальне значення залежить від специфікації обладнання. |

Продовження таблиці 2.4

|  |  |  |
| --- | --- | --- |
| vSync | int | Режим вертикальної синхронізації:   * 0 – Вимкнути синхронізацію; * 1 – Сихронізація відбувається для кожного кадру; * 2 – Синхронізація відбувається для кожного другого кадру. |
| masterVolume | float | Головна гучність. Регулює одночасно всі джерела звуку у грі. |
| musicVolume | float | Гучність музичного супроводу. |
| vehicleVolume | float | Гучність звуків від транспортних засобів. |
| voiceVolume | float | Гучність голосу. |
| surroundingsVolume | float | Гучність звуків оточення (ембіенту) |
| difficultyLevel | enum (int) | Рівень складності:   * 0 – Симуляторна складність; * 1 – Аркадна складність. |
| fieldOfView | int | Поле зору гравця у градусах. В діапазоні від 50 до 110. |
| throttleIntensity | float | Інтенсивність ходової педалі. |
| brakeIntensity | float | Інтенсивність гальмування. |
| inputSensitivity | float | Чутливість контролера огляду камерою. |

## Проектування та реалізація алгоритмів роботи системи

Основним модулем системи є керування електротранспортом. Даний модуль складається з багатьох скриптів, що контролюють:

* рух транспортного засобу;
* елементи приборної панелі та інтерактивні елементи кабіни;
* інтерактивні елементи кузова;
* вхід та вихід гравця у транспортний засіб;
* штанги тролейбуса та інші допоміжні елементи;
* автономний хід тролейбуса;
* клімат всередині автомобіля.

При реалізації системи симуляції було найбільш точно реалізовано процес керування тролейбусом. Аналогічні дії виконуються і для реальних об’єктів.

Перед запуском тролейбуса необхідно виконати певні дії для увімкнення усіх систем, але основний алгоритм, за яким відбувається сам процес, наведено на рис. 2.16.

Перш за все перевіряється наявність напруги у системі (присутня тільки за умови піднятих струмоприймачів). Якщо струм є, тоді запускається електродвигун та допоміжні елементи, після чого відбувається накачування повітря у пневмосистему за допомогою двигуна-генератора, якщо тиск нижче за критичний. У разі наявності достатньої кількості повітря тролейбус чекає інших дій водія.

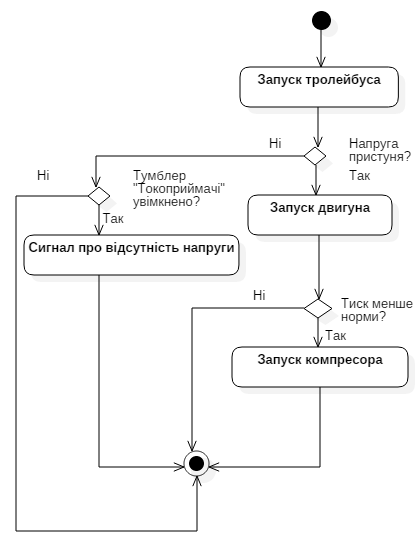


Рис. 2.16. Алгоритм запуску електродвигуна.

За умови відсутності струму відбувається увімкнення зуммеру, що сигналізуватиме про опущені штанги (лише якщо увімкнено тумблер «Струмоприймачі»), після чого тролейбус також чекає інших дій водія [10].

Унікальним з точки зору алгоритмізації системи є алгоритм роботи автономного ходу, що представлено на рис. 2.17.

Автономний хід (АХ) – можливість роботи електротранспорту при відсутності безпосереднього підключення до контактної мережі [11].

Процеси, що стосуються АХ відбуваються постійно під час руху. Наприклад, весь час, поки тумблер є вимкненим відбувається заряджання акумуляторних батарей (якщо вони не є повністю зарядженими). Після увімкнення тумблеру при достатній кількості збереженої енергії вмикається живлення від батарей, що триватиме доти, доки тумблер не буде відключено. Після вимкнення батареї починають зарядку, якщо транспортний засіб під’єднано до контактної мережі [12].



Рис. 2.17. Алгоритм роботи автономного ходу.

Таким чином, було розглянуто алгоритми основних процесів керування системи симуляції.

## Реалізація функціоналу системи симуляції.

Система симуляції повинна мати модель поведінки, що є якнайбільш наближеною до реалістичної, отже головний контролер транспортних засобів CTSVehicleController містить як елементи, що стосуються саме системи, так і різноманітні фізичні аспекти. Список основних методів скрипту наведено у таблиці 2.5.

Таблиця 2.5.

Основні методи CTSVehicleController

|  |  |
| --- | --- |
| Назва | Опис та призначення |
| *1* | *2* |
| Awake() | Передпервинна ініціалізація значень. |
| Start() | Первинна ініціалізація. Містить присвоєння значень за замовчуванням для змінних та виклики інших методів ініціалізації. |
| Update() | Оновлення. Викликається рушієм на кожному кадрі. Містить дії, що не стосуються фізичного рушія, такі як: зчитування натиснутих клавіш, оновлення звукових ефектів, джерел світла, параметри тиску та ін. |
| FixedUpdate() | Оновлення. Викликається рушієм через певні проміжки часу. Містить дії, що стосуються фізичного рушія: оновлення параметрів двигуна, гальм, диференціалу та стабілізатора поперечної стійкості. |
| SoundsInitialize() | Первинна ініціалізація звуків. Містить динамічне створення джерел звуку для кожного із заданих. |
| KillOrStartEngine() | Запуск або зупинка двигуна, залежно від поточних параметрів, наприклад, якщо двигун запущено, то виклик методу його зупинить. |
| Contactors() | Симуляція звуків контакторної панелі. |
| Handbrake() | Оновлення параметрів ручного гальма. |
| WheelTransforms() | Оновлення позиції та обертання коліс. |

Продовження таблиці 2.5

|  |  |
| --- | --- |
| *1* | *2* |
| Engine() | Оновлення параметрів двигуна. Містить підрахунок швидкості транспортного засобу, прискорення, надання крутного моменту колесам. |
| Sounds() | Оновлення звуків транспортного засобу. |
| ApplyMotorTorque() | Надання крутного моменту певному колесу (розгін). |
| ApplyBrakeTorque() | Надання крутного моменту певному колесу (гальмування). |
| ApplySteering() | Надання обертання переднім колесам. |
| KeyboardControlling() | Оновлення параметрів вводу дій користувача. |
| SteeringWheelRotation() | Оновлення параметрів рульового колеса. |
| WheelCamber() | Оновлення параметрів розвалу коліс. |
| Lights() | Оновлення джерел світла. |

На основі функціоналу даного скрипта можна зробити висновок, що в процесі роботи системи відбувається симуляція та обробка всіх фізичних параметрів, які є необхідні для максимально реалістичної поведінки транспортного засобу та отримання максимальної кількості гарних відчуттів та емоцій від використання продукту.

Елементами, без яких неможливо уявити тролейбус є його штанги (струмоприймачі). Саме вони відповідають за передачу струму від контактної мережі до електродвигуна тролейбуса, тому необхідно, аби симуляція їх роботи в системі була ідентичною до реальної моделі. Основні методи скрипту PolesController, що відповідає за поведінку штанг та їх опис наведено у таблиці 2.6.

Таблиця 2.6.

Основні методи PolesController

|  |  |
| --- | --- |
| Назва | Опис та призначення |
| *1* | *2* |
| Start () | Первинна ініціалізація. Містить присвоєння значень за замовчуванням для змінних та виклики інших методів ініціалізації. |

Продовження таблиці 2.6

|  |  |
| --- | --- |
| *1* | *2* |
| OnTriggerEnter (), OnTriggerExit () | Викликається при заході та виході з тріггера, якщо штанги опущені. Використовується для оновлення списку найближчих дротів. |
| Update () | Оновлення. Викликається рушієм кожного кадру. Містить відслідковування натискань клавіш користувачем. |
| GetClosestWireLeft (),  GetClosestWireRight () | Повертає найближчий до тролейбуса лівий або правий дріт відповідно. |
| CalcNewLeftBindingPointPosition (),  CalcNewRightBindingPointPosition () | Підрахунок нової точки прив'язки для лівої та правої штанги відповідно. Детальний опис наведено нижче. |
| Power () | Оновлення параметрів живлення та зарядки батарей для автономного ходу. |
| UpdatePolesRotation () | Оновлення обертання штанги відносно точки прив'язки. |
| UpdateBindingPoints () | Оновлення точок прив'язки. |

Точка прив'язки штанги – це точка на дроті, яка є кінцем для піднятого струмоприймача. Для її розрахунку необхідно використати певні геометричні операції.

Під час оновлення кожного кадру відомо:

* точки початку та кінця дроту;
* точку початку штанги;
* довжину штанги.

Необхідно знайти точку кінця штанги, якщо відомо, що вона лежить на відрізку, що представлений дротом.

Виходячи з даних відомостей, можна уявити дану систему точок як трикутник, у якому відомо дві сторони та кут навпроти однієї з них (рис. 2.18).

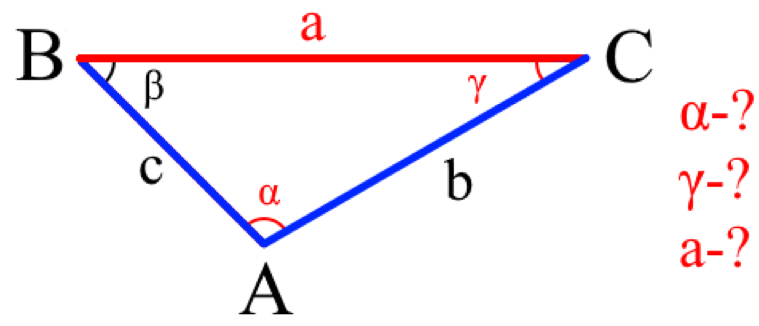


Рис. 2.18. Трикутник для розрахунку точки прив'язки

З цього випливає, що:

* A та С – точки початку та кінця штанги відповідно;
* b – довжина штанги;
* B – точка кінця дроту;
* с – довжина відрізку від початку штанги до кінця дроту;
* a – шукана відстань від кінця штанги до кінця дроту;

Таким чином, можна розрахувати кут β між двома векторами – BC (який можна знайти, адже відомо точку кінця та початку дроту) та AB.

За теоремою синусів можна знайти синус кута γ:

У випадку розрахунку позиції штанг обов’язково sin γ < 1 (кут, що отримується, не може бути більший за 90°) та b > c ( якщо ж ні, то сторона, що відповідає штанзі занадто коротка і не дістає до дроту). Якщо всі умови, накладені на значення синусу виконуються, то знаходиться значення кута:

Виходячи із значення γ можна розрахувати значення α:

Третя сторона шукається за теоремою синусів виходячи з вже відомих значень:

Тепер, коли відомо відстань від кінця штанги до кінця дроту є можливість розрахувати шукані координати точки C. Для цього у вигляді вектора пересуваємо нову точку, яка відповідає кінцю дроту на *a* одиниць в бік початку дроту.

Програмний код, що написано на основі даного алгоритму наведено у додатку Б.

## Висновки до розділу 2

Система симуляції реалізує функціонал управління тролейбусом та моделювання реальних подій, що зв’язані з керуванням даним транспортним засобом. Симуляція забезпечується наборами скриптів.

Для збереження користувацьких даних було реалізовано систему управління файлами у відповідності до потреб. До них входять: файл збереження ігрової сесії, файл опису карти, що завантажуватиметься, файл збереження налаштувань та призначення клавіш.

Було розглянуто алгоритми основних процесів системи та проаналізовано взаємодію класів системи під час процесів симуляції.

В останньому пункті наводиться детальний розгляд деяких аспектів реалізації: описано методи головного скрипта-контролера, методи скрипта для контролю струмоприймачів, детально описано процедуру розрахунку точки прив’язки для штанг.

1. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ СИМУЛЯЦІЇ

## [Порядок](file:///C:\планшет\Магістерська%20робота1.docx#_Toc423399390) встановлення та налаштування системи.

Система симуляції керування тролейбусом це додаток, що було створено на рушії U3D. Після побудови релізної версії систему можна переносити на будь-який девайс, що відповідає цільовій операційній системі збірки, адже рушій має підтримку багатьох платформ. Наприклад, якщо збірку було виконано під Windows, симулятор комфортно запуститься на будь-якій системі, конфігурація якої задовольняє наступним мінімальним вимогам:

* процесор з тактовою частотою 2.4 ГГц або вище;
* 4 ОЗУ;
* 1 ГБ вільного місця на жорсткому диску;
* Графічний пристрій з підтримкою DirectX 9 (бажано DirectX 11).

Таким чином, розроблена система матиме достатню частоту кадрів (30 fps) на системах, конфігурація яких не старіше за 4-5 років та комфортну (60fps) на більшості сучасних систем.

## [Структура](file:///C:\\планшет\\Магістерська%20робота1.docx" \l "_Toc423399391) інтерфейсу. Інтерфейс та порядок роботи з системою симуляції.

Після завантаження користувач опиняється у головному меню (рис. 3.1).



Рис. 3.1. Головне меню.

Тут наявні три елементи керування, що представлені у вигляді кнопок: «Грати», «Налаштування» та «Вихід». Перша з них переносить гравця у підменю завантаження гри, кнопка «Налаштування» - у підменю налаштувань, кнопка «Вихід» виконує вихід з гри, якщо користувач підтвердить свої наміри (рис. 3.2).



Рис. 3.2. Підтвердження виходу.

Після переходу у підменю налаштувань (рис. 3.3) користувачеві будуть представлені налаштування чотирьох видів: ігрові, графічні, звукові та налаштування клавіш. Всі чотири типи організовані у вигляді списків, кожен пункт в яких складаються з назви налаштування та елементу керування, що надає гравцеві управління цим налаштуванням. Детальний опис ігрових налаштувань представлено у таблиці 3.1.

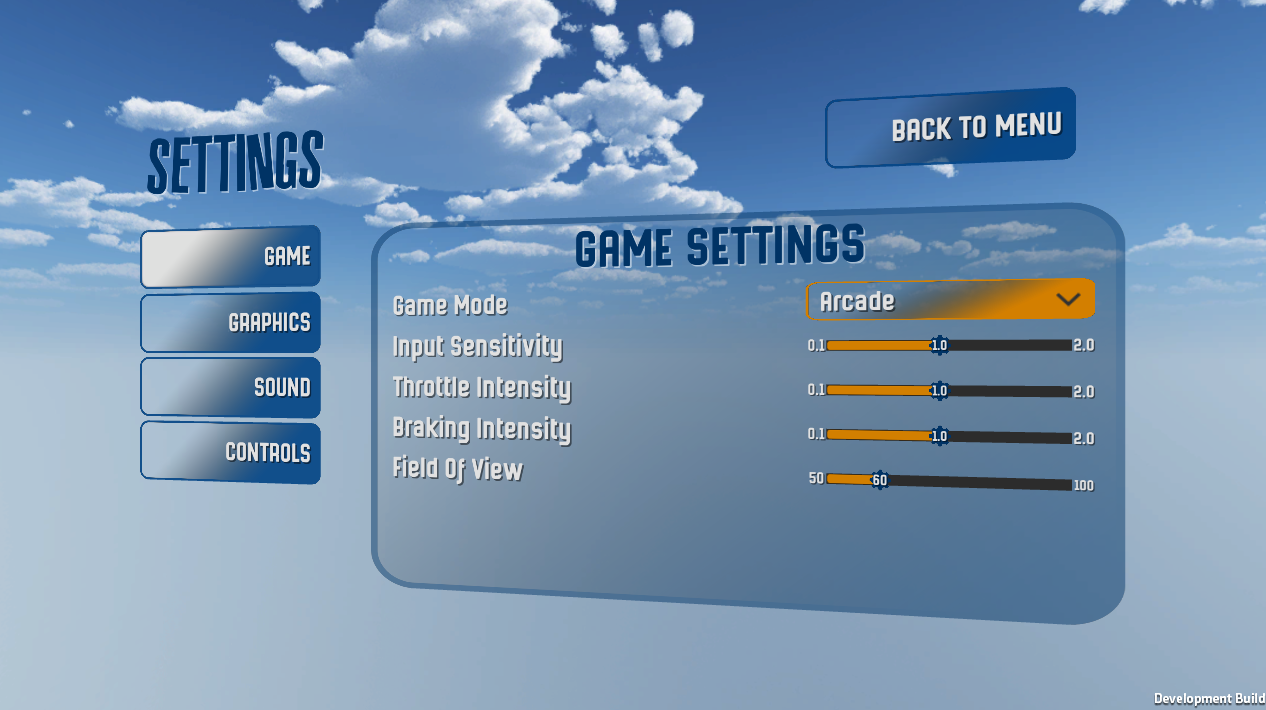


Рис. 3.3 Підменю налаштувань. Ігрові налаштування.

Таблиця 3.1

Опис та призначення ігрових налаштувань

|  |  |
| --- | --- |
| Назва | Опис |
| Режим гри (Game Mode) | Режим гри представлений у двох видах:   * Симулятор. Жорстка симуляція всіх аспектів керування; * Аркада. Полегшене керування, наприклад, запуск двигуна з однієї кнопки. |
| Чутливість керування (Input Sensitivity) | Чутливість огляду мишою чи іншим контролером. |
| Інтенсивність педалі газу (Throttle Intensity) | Чутливість педалі газу до натисків. |
| Інтенсивність гальмування (Braking Intensity) | Чутливість гальма до натисків. |
| Поле зору (Field Of View) | Поле зору гравця. Працює як в режимі керування транспортом, так і при переміщенні пішки. |

Графічні налаштування містять різні властивості, такі як: повноекранний режим, розширення екрану, режими згладжування та інші.



Рис. 3.4. Підменю налаштувань. Графічні налаштування.

Звукові налаштування відповідають за гучність різних типів звуків під час гри. Елементи керування представлені у вигляді слайдерів.



Рис. 3.5. Підменю налаштувань. Звукові налаштування.

Налаштування клавіш представлені у вигляді списку назв дії та клавіш дії, що до них прив’язані (рис. 3.6). Для перепризначення клавіші необхідно натиснути кнопку, яку є необхідність перепризначити та натиснути клавішу на заміну.



Рис. 3.6. Підменю налаштувань. Керування.

Після натискання на кнопку «Назад» у правому верхньому куті всі зміни, що було внесено користувачем, збережуться і гравця буде повернено у головне меню.

Підменю завантаження гри (рис. 3.7) складається зі списку з десяти слотів для збереження, на кожному з яких можна розпочати нову гру або продовжити раніше почату. За необхідності присутня можливість видалення всіх даних, що стосуються виділеного слоту.

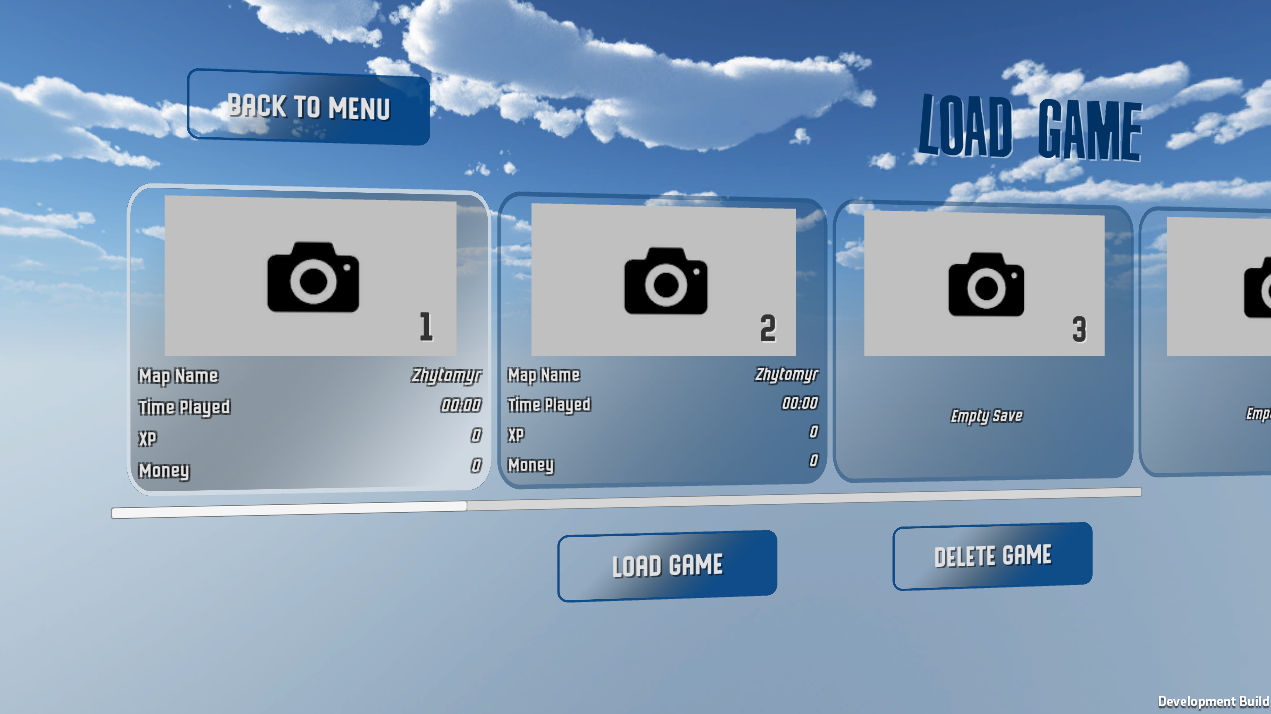


Рис. 3.7. Підменю завантаження гри.

Для того, аби розпочати нову гру необхідно обрати порожній слот та натиснути на кнопку «Нова гра», що з’явиться. Далі відкриється список ігрових карт, на яких можна розпочати гру (рис. 3.8). Тут можна почитати інформацію про ту чи іншу ігрову карту та обрати ту, яка до вподоби. Для перегляду детальної інформації необхідно натиснути на кнопку «і» поруч з іконкою карти.



Рис. 3.8. Список карт для нової гри.

Після того, як гравцем було обрано карту для гри буде відображено прогрес завантаження (рис. 3.9) та після того, воно буде завершене гравець потрапляє безпосередньо у режим симуляції.

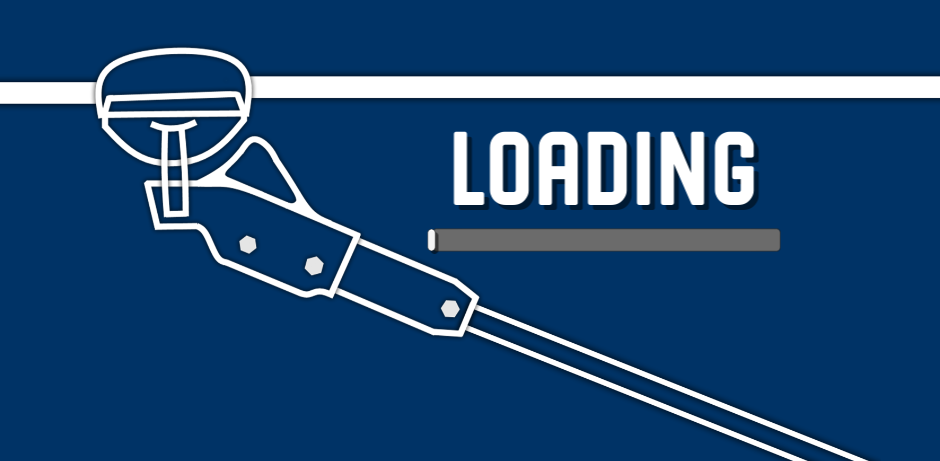


Рис. 3.9. Відображення прогресу завантаження.

Загальне керування. Щоб розпочати керування тролейбусом спочатку необхідно сісти за кермо. Для цього потрібно підійти до зовнішніх водійських дверей та відчинити їх, натиснувши клавішу взаємодії (за замовчуванням ліва кнопка миші), після чого відчинити розсувні двері в кабіну водія та сісти на сидіння, виконавши аналогічні дії.



Рис. 3.10. Взаємодія з інтерактивним елементом

Після посадки на робоче місце водія для початку руху необхідно підняти струмоприймачі (за замовчуванням «Q») та увімкнути:

* головний автоматичний вимикач;
* акумулятор;
* гідропідсилювач керма;
* тумблер керування;
* тумблер контакторів;
* тумблер струмоприймачів.

Після виконання вищевказаних дій або після натискання на кнопку «Двигун» (за замовчуванням «I»), яка вмикає все необхіне, електродвигун вмикається і тролейбус готовий до руху. Якщо ж після виконання всіх дій двигун не працює, це означає, що щось не було виконано. Наприклад, якщо чутно звук зуммера та горить червоний індикатор на приборній панелі, то відсутнє живлення.

Для вибору напрямку руху необхідно перевести реверсор у позицію «вперед» (за замовчуванням «LeftShift» - перемикання на позицію вище, «LeftCtrl» - на позицію нижче). Загалом позицій три: «вперед», «нейтраль», «назад».

Ігровий інтерфейс. Під час безпосереднього керування елементи користувацького інтерфейсу включають:

* годинник і дату;
* список наступних зупинок (з’являється лише при затисненні клавіші «Показати графік»);

віртуальна приборна панель.

Інтерфейс графіку складається із списку зупинок, які необхідно відвідати та часу, коли це необхідно зробити за графіком.

Інтерфейс керування складається з двох приладів та набору індикаторів.



Рис. 3.11. Інтерфейс під час керування.

Перший прилад – спідометр – включає в себе цифровий та аналоговий прилад для вимірювання швидкості та індикатор реверсору. Другий прилад – манометр – вимірює тиск повітря у системі та індикатор заряду, що лишився в акумуляторах автономного ходу. Між приладами знаходиться індикатор живлення. Також деякі індикатори продубльовані на приборній панелі.

Панель індикаторів складається з трьох рівнів:

* Індикатори на першому рівні: низький тиск у системі, сигнали повороту, ручне гальмо, аварійна сигналізація, опалення;
* Двері;
* Габаритні вогні, ближнє світло, світло в салоні, два склоочищувачі.

Автономний хід (АХ). Використовується коли підключення до контактної мережі відсутнє у межах дії струмоприймачів. Активується натисканням клавіші «Автономний хід» (за замовчуванням «CapsLock»). Використання АХ обмежене у часі, який на інтерфейс виводиться лише приблизно, у межах ±25% від реального значення. Зарядження акумуляторів здійснюється після підключення до контактної мережі.

## Тестування роботи системи

Перевірка на працездатність – невід’ємний процес під час розробки програмного коду системи. Ігрове тестування в певних аспектах відрізняється від тестування інтернет-ресурсів, мобільних додатків та інших продуктів.

Тестування ігрових систем містить такі основні види:

* тестування функціональності;
* тестування контенту;
* тестування геймплею;
* користувацьке тестування.

Тестування функціональності. Нижче наведено тест-кейси роботи головного меню.

Кнопка «Settings»:

*передумови:* гравець повинен знаходитись у головному меню;

*кроки:* натискання кнопки «Settings»;

*очікуваний результат:* анімований перехід до вікна з налаштуваннями;

*фактичний результат:* анімований перехід до вікна з налаштуваннями.

Кнопка «Play»:

*Передумови:* Гравець повинен знаходитись у головному меню;

*Кроки:* Натискання кнопки «Play»;

*Очікуваний результат:* Анімований перехід до списку слотів збереження;

*Фактичний результат:* Анімований перехід до списку слотів збереження;

Кнопка «Delete» у списку слотів для збереження:

*передумови:* гравець повинен знаходитись у головному меню, підменю «Play»;

*кроки:* натискання кнопки «Delete» для виділеного непорожнього слоту збереження;

*очікуваний результат:* слот збереження буде очищено, при повторному вході в дане підменю слот буде порожнім;

*фактичний результат:* слот збереження очищено, при повторному вході в дане підменю слот є порожнім.

Кнопка «New Game» у списку слотів для збереження:

*передумови:* гравець повинен знаходитись у головному меню, підменю «Play»;

*кроки:* натискання кнопки «New Game» для виділеного пустого слоту збереження;

*очікуваний результат:* анімований перехід до списку встановлених ігрових карт;

*фактичний результат:* анімований перехід до списку встановлених карт.

Кнопка «Load Game» у списку слотів для збереження:

*передумови:* гравець повинен знаходитись у головному меню, підменю «play»;

*кроки:* натискання кнопки «Load Game» для виділеного непорожнього слоту збереження;

*очікуваний результат:* анімований перехід у екран завантаження;

*фактичний результат:* анімований перехід у екран завантаження.

Кнопка «New Game» у списку карт для нової гри:

*передумови:* гравець повинен знаходитись у головному меню, підменю «New Game»;

*кроки:* натискання кнопки «New Game» для виділеної ігрової карти;

*очікуваний результат:* анімована заміна кнопки «New Game» на кнопки підтвердження «Yes» та «No», при натисненні кнопки «Yes» - перехід до екрану завантаження, «No» - анімована заміна кнопок підтвердження на «New Game»;

*фактичний результат:* анімована заміна кнопки «New Game» на кнопки підтвердження «Yes» та «No», при натисненні кнопки «Yes» - перехід до екрану завантаження, «No» - анімована заміна кнопок підтвердження на «New Game».

Тестування геймплею. Було проведено тестування геймплейних особливостей, включаючи основну - керування тролейбусом та рух по маршруту, що складаєьтся із віздвідування зупинок у певній послідовності. Крім того, присутнє пасивне навчання. Під час тестування доступного функціоналу у нього не було виявлено жодних недоліків чи помилок у роботі.

## Висновки до розділу 3

В даному розділі було наведено порядок встановлення та налаштування додатку та визначено мінімальну конфігурацію системи для комфортної гри.

Реалізована система симуляції містить інтерфейси головного меню, меню паузи та основний ігровий інтерфейс, робота та функції елементів якого були наведені.

Також було проведено функціональне та геймплейне тестування додатку. Знайдені помилки було виправлено, система працює коректно та виконує поставлені задачі.

# ВИСНОВКИ

Під час виконання випускної роботи бакалавра було спроектовано та реалізовано систему симуляції керування тролейбусом для цього було вирішено наступні завдання:

1. Проведений аналіз предметної області дозволив визначити основні аспекти симуляції громадського транспорту. Огляд наявних аналогів та виявлення протиріч між наявними можливостями та потребами показав, що головними функціями майбутньої системи має бути: зручність GUI, наявність навчальних елементів, взаємодія з окремими елементами транспорту, вільне пересування по ігровій області. Наявність підтримки користувацьких модифікацій, економічна складова та мультиплеєрний режим є також бажаними функціями для таких програмних систем. Визначившись з оптимальною архітектурою Entity-Component-System, обрано необхідний та достатній набір програмних продуктів для реалізації поставлених задач: рушій Unity3D, 3D-редактор Blender3d, редактор текстур Substance Painter.
2. Було проаналізовано функціональні та нефункціональні вимоги до системи симуляції, для чого побудовано діаграму варіантів використання та на її основі розроблено діаграму класів для аналізу структури системи.
3. Для збереження користувацьких даних було реалізовано систему управління файлами у відповідності до потреб. До них входять: файл збереження ігрової сесії, файл опису карти, що завантажуватиметься, файл збереження налаштувань та призначення клавіш.
4. Було розглянуто алгоритми основних процесів системи та проаналізовано взаємодію класів системи під час процесів симуляції.
5. Було наведено детальний розгляд деяких аспектів реалізації: описано методи головного скрипта-контролера, методи скрипта для контролю струмоприймачів, детально описано процедуру розрахунку точки прив’язки для штанг.
6. Реалізована система симуляції містить інтерфейси головного меню, меню паузи та основний ігровий інтерфейс, робота та функції елементів якого були наведені.
7. Також було проведено функціональне та геймплейне тестування додатку. Знайдені помилки було виправлено, система працює коректно та виконує поставлені задачі.

Реалізований програмний комплекс готовий до використання. Система може бути використана для навчання базовим навичкам керування тролейбусом або ж просто для розваги.

В подальшому може дана система може бути розширена шляхом впровадження нових функціональностей та та розширення ігрового світу. Завдяки обраній архітектурі розширення системи відбуватиметься значно швидше.

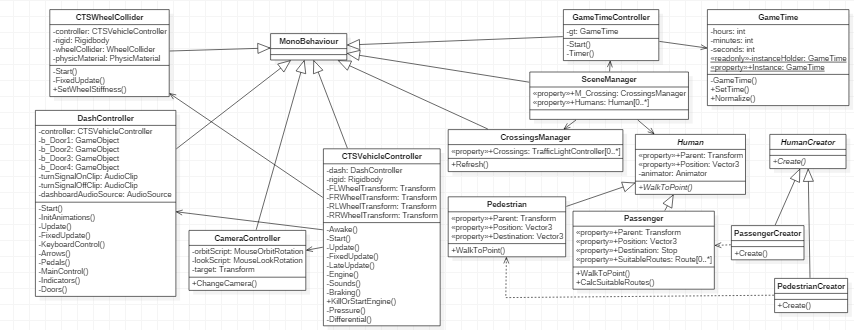
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ефремов И. С. Технические средства горэлектротранспорта / И. С. Ефремов, В. М. Кобозев, В. В. Шевченко. – М.: "Высшая школа", 1985. – 448 с.
2. Симуляція [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D1%83%D0%BB%D1%8F%D1%86%D1%96%D1%8F>.
3. Білас С. "A Data-Driven Game Object System" / Скот Білас., 2013. – 251 с.
4. Торн А. Mastering Unity Scripting / Алан Торн. – Бірмінгем: Packt Publishing, 2015. – 380 с.
5. Прахов А. Blender. 3D-моделирование и анимация. / Андрей Прахов. – СПб.: БХВ-Петербург, 2009. – 266 с.
6. Substance Painter [Електронний ресурс] – Режим доступу до ресурсу: <https://support.allegorithmic.com/documentation/display/SPDOC/Substance+Painter>.
7. Шаллоуей А. Шаблоны проектирования. Новый подход к объектно-ориентированному анализу и проектированию / Алан Шаллоуей. – М.: "Вильямс", 2002. – 288 с.
8. Фрімен Е. Паттерни проектування / Е. Фрімен, Е. Фрімен. – СПб: Питер, 2011. – 656 с.
9. Использование Singleton в Unity3D [Електронний ресурс] – Режим доступу до ресурсу: <https://habr.com/post/341830/>.
10. Троллейбус пассажирский ЗиУ-682Г-01 – Енгельс: АО ТРОЛЗА, 1999. – 291 с.
11. Троллейбус пассажирский ЗиУ-682Б – М.: ТРАНСПОРТ, 1977. – 208 с.
12. Слепцов М. А. Основы электрического транспорта / М. А. Слепцов. – М.: "Издательский центр "Академия"", 2006. – 464 с.

# ДОДАТКИ

# ДОДАТОК А

Діаграма класів основних елементів системи



ДОДАТОК Б

Метод, що розраховує позицію точки прив’язки для штанги

private Vector3 CalcNewBindingPointPosition(Wire wire, Pole pole, bool lowerPolesIfNotFound)

{

Vector3 newPosition = new Vector3(float.PositiveInfinity, float.PositiveInfinity, float.PositiveInfinity);

Vector3 AB = wire.End - pole.EquipmentPoint.position;//вектор від точки обладнання до початку провода

Vector3 CB = wire.End - wire.Start;//вектор провода

float c = Vector3.Distance(wire.End, pole.EquipmentPoint.position);

float b = Vector3.Distance(pole.BashmakPoint.position, pole.EquipmentPoint.transform.position);

float a = 0f;

float angleAlpha = 0f;

float angleBeta = Vector3.Angle(AB, CB);

float angleGamma = 0f;

float sinGamma = Mathf.Sin(angleBeta \* Mathf.Deg2Rad) \* (c / b);

Debug.DrawLine(wire.End, pole.EquipmentPoint.position, Color.green);

Debug.DrawLine(pole.BashmakPoint.position, pole.EquipmentPoint.position, Color.magenta);

Debug.DrawLine(wire.Start, wire.End, Color.red);

if (sinGamma >= 1f || (angleBeta >= 90f && b <= c))

{

//ініціювати зліт штанг

if (lowerPolesIfNotFound)

{

Debug.Log("Штанги злетіли. Довжина штанги (" + b + ") менше, ніж відстань до КС(" + c + ") або синус кута гамма(" + sinGamma + ") > 1");

InitPolesLowering();

}

return new Vector3(float.PositiveInfinity, float.PositiveInfinity, float.PositiveInfinity);

}

else

angleGamma = Mathf.Asin(sinGamma) \* Mathf.Rad2Deg;

angleAlpha = 180f - angleGamma - angleBeta;

a = b \* (Mathf.Sin(angleAlpha \* Mathf.Deg2Rad) / Mathf.Sin(angleBeta \* Mathf.Deg2Rad));

newPosition = wire.Start;

newPosition = Vector3.MoveTowards(newPosition, wire.End, wire.Controller.length - a);

return newPosition;

# }