НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційне забезпечення робототехнічних систем»»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Віртуальний конструктор робототехнічних пристроїв з елементами тестування»

Виконав (-ла):
студент (-ка) IV курсу, групи ІК-93
Патока Владислав Володимирович _____

Керівник:
Старший викладач кафедри ІСТ
Коваль Олександр Сергійович _____

Рецензент:
с.н.с., к. т. н., доцент
Писаренко Юлія Валеріївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Патоці Владиславу Володимировичу

1. Тема проєкту «Віртуальний конструктор робототехнічних пристроїв з елементами тестування», керівник проєкту Коваль Олександр Сергійович, старший викладач кафедри ІСТ, затверджені наказом по університету від «31» травня 2023 р. № 2101-с

2. Термін подання студентом проєкту: 12 червня 2023 року

3. Вихідні дані до проєкту: інтерактивний програмний застосунок для конструювання та тестування робототехнічних пристроїв, в якості формату зберігання даних обрано JSON, в якості основного інструменту для розробки обрано ігровий двигун Unity, для реалізації механік використано мову програмування C# та бібліотеки платформи .Net.

4. Зміст пояснювальної записки:
Розглянути процес проектування та тестування робототехнічних пристроїв, їх основні етапи. Проаналізувати інсуючі на ринку рішення, які дозволяють автоматизувати процес конструювання та тестування робототехнічних пристроїв. Розглянути інструменти для створення інтерактивних додатків. Обрати необхідні для створення програмного застосунку технології та обґрунтувати свій вибір. Розробити програмний застосунок. Описати його програмну реалізацію. Описати роботу користувача з програмним застосунком.

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо): Діаграма прецедентів (А3), Діаграма потоків даних (А3), Діаграма класів (А3), Діаграма бізнес-процесів (А3)

6. Дата видачі завдання: 1 грудня 2022 року

Календарний план

№ з/п	Назва етапів виконання проєкту	Термін виконання етапів проєкту	Примітка
1.	Затвердження теми роботи	11.02.23	
2.	Аналіз предметної області	20.04.23	
3.	Аналіз інструментів для створення інтерактивних програмних додатків	30.04.23	
4.	Проектування програмного забезпечення	10.05.23	
5.	Розробка програмного застосунку	29.05.23	
6.	Тестування програмного застосунку та виправлення помилок	04.06.23	
7.	Оформлення текстової документації	06.06.23	

Студент Владислав ПАТОКА

Керівник Олександр КОВАЛЬ

АНОТАЦІЯ

Патока В.В. Віртуальний конструктор робототехнічних пристроїв з елементами тестування. КПІ ім. Ігоря Сікорського, Київ, 2023.

Проєкт містить 70 с. тексту, 35 рисунків, 17 формул, посилання на 19 літературних джерел та 4 конструкторські документи.

Ключові слова: робототехніка, робототехнічний пристрій, проектування. конструювання, тестування, віртуальний конструктор, Unity.

Об’єктом розробки є віртуальний конструктор робототехнічних пристроїв з елементами тестування.

Метою роботи є автоматизація процесу конструювання та тестування робототехнічних пристроїв, для досягнення якої виконується завдання з реалізації віртуального конструктора робототехнічних пристроїв з елементами тестування.

Для реалізації системи було використано: двигун для створення інтерактивних додатків Unity, мова програмування C#, платформа Mono, бібліотеки платформи .Net, UnityEngine, UnityEngine.UI, UnityEngine, Newtonsoft.JsonJsonConvert, середовище розробки Visual Studio.

У дипломному проєкті було розроблено віртуальний конструктор робототехнічних пристроїв з елементами тестування, який містить функціонал зі сторення моделі робототехнічного пристрою, налаштування її окремих компонентів, її збереження, а також тестування.

Розроблений програмний застосунок в подальшому може бути використаний розробниками робототехнічних пристроїв для проектування та тестування їх розробок.

ANNOTATION

V.V. Patoka. Virtual Constructor of Robotics Devices with Testing Elements. Igor Sikorsky Kyiv Polytechnic Institute, Kyiv, 2023.

The project consists of 70 pages of text, 35 figures, 17 formulas, references to 19 literary sources, and 4 design documents.

Keywords: robotics, robotics device, design, construction, testing, virtual constructor, Unity.

The object of development is a virtual constructor of robotics devices with testing elements.

The goal of the work is to automate the process of designing and testing robotics devices. To achieve this, the task of implementing a virtual constructor of robotics devices with testing elements is performed.

For the implementation of the system, the following tools were used: Unity engine for creating interactive applications, C# programming language, Mono platform, .Net platform libraries, UnityEngine, UnityEngine.UI, UnityEngine, Newtonsoft.JsonJsonConvert, Visual Studio development environment.

The diploma project has developed a virtual constructor of robotics devices with testing elements, which includes functionality for creating a model of a robotics device, configuring its individual components, saving it, and testing.

The developed software application can be used by developers of robotics devices for designing and testing their developments.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІК93.120БАК.005 ПЗ	Пояснювальна записка	70		
6	А3	ІК93.120БАК.005 Д1	Діаграма прецедентів	1		
7	А3	ІК93.120БАК.005 Д2	Діаграма потоків даних	1		
8	А3	ІК93.120БАК.005 Д3	Діаграма класів	1		
9	А3	ІК93.120БАК.005 Д4	Діаграма бізнес-процесів	1		
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІК93.120БАК.005 ТП		
Розроб.		Патока В.В.			Віртуальний конструктор робототехнічних пристроїв з елементами тестування. Відомість проєкту	Літ.	Аркуш 1 / Аркушів 1
Керівн.		Коваль О.С.				Т	
						КПІ ім. Ігоря Сікорського	
						Група ІК-93	
Затв.							

Пояснювальна записка
до дипломного проєкту
на тему: «Віртуальний конструктор робототехнічних пристроїв з елементами тестування»

Київ – 2023 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

IT – Information Technology.

UI – User Interface.

UX – User Experience.

БД – База даних.

JSON – JavaScript Object Notation.

CLR – Common Language Runtime.

DFD – Data Flow Diagram.

BPMN – Business Process Model and Notation.

DTO – Data Transfer Object.

ПК – Персональний комп’ютер

ID – Identity Document.

SAT – Separating Axis Theorem.

API – Application Programming Interface.

ОС – Операційна система.

ВСТУП

Сьогодні ми стоїмо на порозі технічної революції, коли робототехніка та автоматизація набувають не бачених раніше обертів. Роботи починають використовуватися в досить різних сферах людського життя: вони є і в промисловості, і в медицині, і в побуті – вони є всюди, адже їх використання значно полегшує вирішення будь-яких завдань.

Разом з популяризацією робототехнічних пристроїв зростає також інтерес і до їх розробки. Однак, цей процес є досить складним та потребує багато часу, адже вимагає широкого спектру маніпуляцій. Для розробки реально ефективного пристроя необхідно провести цілу низку тестів над фізичним прототипом у реальних умовах. Такий підхід має значний недолік, який полягає в тому, що команда інженерів не має можливості створити новий прототип або модифікувати існуючий у короткі терміни. Що стає перешкодою для багатьох розробників, адже це потребує значних зусиль та ресурсів. Таким чином з'являється необхідність в створенні більш ефективних, а головне – доступних інструментів для побудови та тестування робототехнічних пристроїв.

Наразі вже існують програмні продукти, які націлені на спрощення етапу прототипування та тестування робототехнічних пристроїв. Але все ж вони не мають повноцінного функціоналу, який би дозволив розробити прототип, використовуючи всеможливі електронні компоненти, покази яких вкрай необхідні під час його тестування.

Мета і завдання дослідження. Метою даної дипломної роботи стала автоматизація процесу конструювання та тестування робототехнічних пристроїв шляхом створення віртуального конструктора.

Даний програмний продукт має надавати користувачу можливості з: конструювання прототипу майбутнього пристрою, налаштування характеристик його компонентів, збереження створеного та налаштованого прототипу, а також проведення тестування. Що дозволить більш широкому колу розробників реалізувати свої ідеї.

Для досягнення поставленої мети були поставлені наступні задачі:

– аналіз ринку на наявність готових рішень та інструментів для конструювання, а також тестування робототехнічних пристроїв;

– визначення базових вимог, щодо мінімального функціоналу застосунку та повного функціоналу в цілому;

– розробка архітектури додатку, його основних модулів, а також вибір необхідних інструменту, мови програмування, патернів проектування;

– проектування структури та користсувацького інтерфейсу додатку;

– розробка програмного продукту відповідно до базових вимог та визначеного функціоналу.

Об’єктом дослідження даного дипломного проекту є процес створення прототипів робототехнічних пристроїв та їх тестування.

Предметом дослідження є віртуальний конструктор робототехнічних пристроїв.

Практичне значення одержаних результатів полягає в тому, що застосунок за умови подальшої розробки стане корисним інструментом для розробників робототехнічних пристроїв, адже дозволить набагато швидше створювати нові механізми, а також суттєво зменшить витрати на фізичні компоненти та експериментальне тестування.

Положення, запропоновані автором для розв'язання поставлених задач, включають в себе розробку віртуального конструктора, функціонал якого дозволить створювати моделі пристроїв на основі інсуючої бази компонентів, які можливо індивідуально налаштувати, на відміну від існуючих на ринку рішень. Також невід’ємною частиною додатку є моливість протестувати сконструйовану модель, під час чого можливо переглянути показники наявних на ній компонентів, змінити їх налаштування, що є новизною у даній сфері.

Дипломний проект складається з наступних розділів: вступ, основні розділи, висновки, список використаних джерел із 19 найменувань. Графічна частина включає 4 кресленика формату А3 . Загальний обсяг 70 сторінок.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Робототехніка та її роль у сьогоденні

Робототехнікою називають галузь науки, яка вивчає проектування, програмування, конструювання та застосування роботів. Вона спирається на такі дисципліни, як інформатика, фізика, математика, кібернетика [1] (рисунок 1.1).

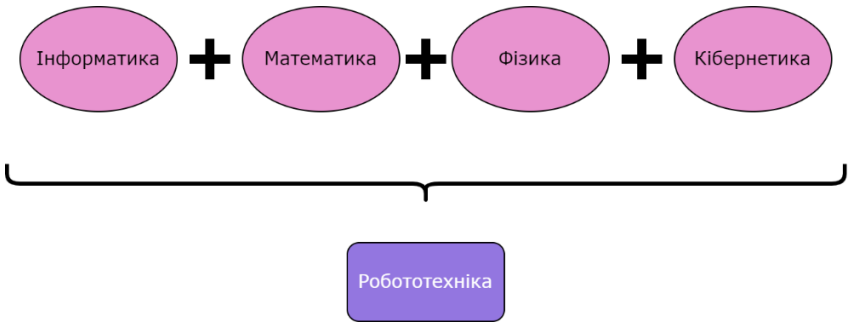

Рисунок 1.1 – Складові дисципліни робототехніки

Робот – це перш за все автоматизований механічний пристрій, який здатний виконувати різноманітні запрограмовані дії без участі наглядача. Вони можуть взаємодіяти з навколишнім середовищем, рухатися та приймати власні рішення, використовуючи для цього спеціальні електронні компоненти.

У наш час робототехніка активно використовується у багатьох галузях людської діяльності. Роботи створюються для вирішення майже будь яких задач різного рівня складності та націлені перш за все на спрощення та автоматизацію людської діяльності.

1.2 Сфери застосування робототехніки

Наразі роботів можна зустріти всюди, але все ж необхідно виділити їх важливу роль у таких сферах як: промисловість, медицина, наука, військова справа [2]. Також вони досить активно використовуються у сучасному побуті.

Промислові роботи надають можливість оптимізувати та автоматизувати виробничі процеси. Перш за все вони націлені на виконання складних завдань,

пов’язаних зі збиранням, монтажем, зварюванням та фарбування всеможливих елементів при виробництві. Такий підхід допомагає збільшити продуктивність виробничої лінії, знизити витрати, покращити якість готової продукції, а головне – вберегти людей від небезпечної роботи.

У галузі медицини роботи відкривають нові можливості для проведення хірургічних операцій, наданні медичної допомоги та визначення діагнозу пацієнта. Робототехніка допомагає лікарям проводити складні та точні операції, при цьому зменшуючи ризик нанести шкоди пацієнту.

В науці роботи відіграють не менш важливу роль, адже з їх допомогою проводяться наукові дослідження та експерименти, результати яких напряму впливають на розвиток людства в цілому. Існує безліч прикладів, де з науковою місією може впоратися лише робот.

Також не менший вклад роботи вносять і у військову справу. Їх використання рятує життя, та захищає населення. При цьому вони можуть використовуватися у всеможливих формах та напрямках, надаючи значну перевагу на полі бою.

1.3 Проектування робототехнічних пристроїв

1.3.1 Основні етапи проектування робототехнічних пристроїв

Проектування є одним з основних та головних процесів під час розробки робототехнічних пристроїв. Він включає в себе низку кроків, результат виконання кожного з яких напряму впливає на фінальний продукт. Можна виділити декілька основних етапів цього процесу [3]:

– визначення вимог до робототехнічної системи: під час цього етапу визначаються основні цілі та вимоги до пристрою. Розробляється перелік мінімального та повного фунціоналу.

– розробка концепту: впродовж цього етапу приймаються рішення, щодо структури, електроніки та програмного забезпечення робототехнічного пристрою. Виконується аналіз можливих варіантів та вибір оптимального концепту.

– проектування: на цьому етапі розробляється детальний проект пристрою. Звертається увага на всі можливі аспекти, включаючи механічну конструкцію, електричні схеми, сенсори, елементи живлення та зв’язку. Створюється програмне забезпечення. Основною частиною даного етапу є моделювання, під час якого відбувається перевірка працездатності системи та виконання поставлених перед нею вимог.

– виготовлення прототипу: створюється фізичний прототип, який налічує всі необхідні відповідно до документації компоненти.

– тестування: відбувається перевірка працездатності прототипу робототехнічного пристрою, його спроможності виконувати існуючі вимоги. Даний етап дозволяє виявити проблеми та вдосконалити конструкцію та програмне забезпечення пристрою, тим самим збільшити його ефективність.

1.3.2 Проблеми, які виникають при проектуванні

Проектування є досить складним процесом, так як кожний його етап є ключовим та вимагає досить багато уваги. Слід також пам’ятати і про значні як фінансові, так і часові витрати. Таким чином під час проектування розробники стикаються з рядом викликів та проблем, які криються перш за все в складності тестування, адже саме на створення прототипу та його оптимізації витрачається найбільше часу.

Також необхідно пам’ятати про ризики, які пов’язані з складністю системи в цілому, адже вона складається з великої кількості компонентів, кожен з яких повинен бути коректно налаштованим та правильно взаємодіяти з іншими. Програмне забезпечення повинно контролювати кожен процес та враховувати всі фактори, що стає неможливим при некоректній роботі хоча б одного елемента.

В процесі розробки не слід забувати і про енергоефективність системи, адже неправильне розподілення ресурсів на етапі проектування призведе до подальших проблем з прототипом.

1.4 Роль тестування у процесі розробки робототехнічних пристроїв

Тестування є одною з найважливіших складових процесу розробки робототехнічного пристрою. Воно дозволяє перевірити існуючий прототип зі всіх сторін: функціональність, надійність, безпека у використанні, відповідність до заданих вимог. Подальша експлуатації механізму та досвід майбутнього користувача напряму залежить від якості тестування.

Головною метою тестування є виявлення та усунення можливих недоліків ще на етапі розробки. Як результат це допомагає знизити ризик виникнення непердбачених проблем під час використання, покращити функціональність та забезпечити високу якість кінцевого продукту.

1.4.1 Основний методи тестування робототехніних пристроїв

До основних методів тестування робототехнічних пристроїв відноситься проведення різних експериментів та випробувань над прототипом механізму. Можна виділити декілька класичних технік тестування [4]:

– тестування функціональності: перевірка прототипу на виконання основних функцій та завдань, для яких він був розроблений. Подібний тест включає в себе перевірку основних рухомих елементів пристрою, елементів для взаємодії з навколишнім середовищем, а також елементів управління;

– тестування надійності: перевірка функціональності роботехнічного пристрою на стійкість та надійність у різних складних для нього умовах. Це може бути перевірка на механічну міцність, стійкість до пливу зовнішніх факторів, тривале використання та тому подібні дії;

– тестування безпеки: перевірка робототехнічного пристрою на предмет його загрози для оточуючих та навколишкього середовища. Даний спосіб може включати в себе перевірку систем безпеки, датчиків запобіжних засобів, виявлення колізій з оточуючим середовищем і тому подібні.

1.4.2 Недоліки існуючих методів тестування

Класичні методи тестування робототехнічних пристроїв мають низку недоліків, а саме:

– великі матеріальні витрати: тестування фізичної моделі вимагає наявності всіх необхідних для успішного функціонування компонентів, обладнання для проведення тестів, а також експериментального середовища;

– обмежена кількість спроб: неможливо в точності відтворити ті чи інші умови для тестування, що значно обмежує повторюваність результатів;

– обмеження в термінах: тестування фізичного прототипу потребує значних часових витрат, особливо, якщо необхідно провести велику кількість експериментів;

– можливість пошкодження пристрою: у результаті експериментів механізм може зазнати пошкоджень або нанести шкоди обладнанню. В такому разі набагато збільшуються часові та матеріальні витрати.

Одним зі варіантів вирішення проблем класичних способів тестування є використання сучасних методик, таких як коп’терне моделювання та симуляція. Такий підхід має велику кількість переваг над класичними способами, адже дозволяє проводити віртуальне тестування, відтворюючи реальні експерименти зі значно меншими матеріальними та часовими витратами та більшою гнучкістю.

1.5 Автоматизація процесу конструювання та тестування робототехнічних пристроїв

Основна ідея віртуального конструктора робототехнічних пристроїв полягає в створенні інтерактивного середовища, яке дозволить розробникам проектувати, моделювати та тестувати різні варіанти робототехнічних пристроїв до їх фізичної реалізації. Цифровий конструктор надає можливість візуалізувати та інтерактивно взаємодіяти з різними компонентами, налаштовувати їх, перевіряти їх взаємодію та функціональність.

Такий програмний застосунок має декілька вагомих переваг порівняно із загальноприйнятими способами проектування, а також тестування робототехнічних пристроїв. По-перше, такий підхід дозволяє значно пришвидшити процес розробки, а також вдосконалення моделей, шляхом інтерактивного тестування віртуальних прототипів.

По-друге, віртуальний конструктор забезпечує гнучкість у розробці і надає можливість розробникам легко вносити зміни до проекту як на ранніх стадіях так і під час тестування. Інженери мають змогу експериментувати з різними конфігураціями та параметрами, без необхідності фізичного втручання до пристрою.

Головною перевагою цифрового конструктора є економічний аспект, адже такий підхід допомагає значно знизити матеріальні та часові витрати, а також зменшує ризик пошкодження обладнання, оскільки фізичні прототипи не є обов'язковими в процесі розробки.

Таким чином використання віртуального конструктора є гарною альтернативою класичним методам проектування та тестування, так як дозволяє ефективно та безпечно розробляти робототехнічні пристрої, надає можливість зекономити час та ресурси.

1.6 Огляд наявних рішень із автоматизації процесу конструювання та тестування робототехнічних пристроїв

1.6.1 Tinkercad

Tinkercad – це популярний онлайн додаток для 3D моделювання та проектування, який також має вбудовану систему симуляції та тестування. Застосунок надає користувачам можливість створювати віртуальні 3D-моделі та перевіряти їх функціональність та взаємодію віртуальних компонентів [5].

Система симуляції Tinkercad дозволяє створювати рухомі компоненти, такі як колеса, шестерні, санки тощо, та задавати їх параметри, наприклад, розміри,

швидкість обертання тощо. Ви можете розміщувати ці компоненти на сцені та налаштовувати їх взаємодію (рисунок 1.2).

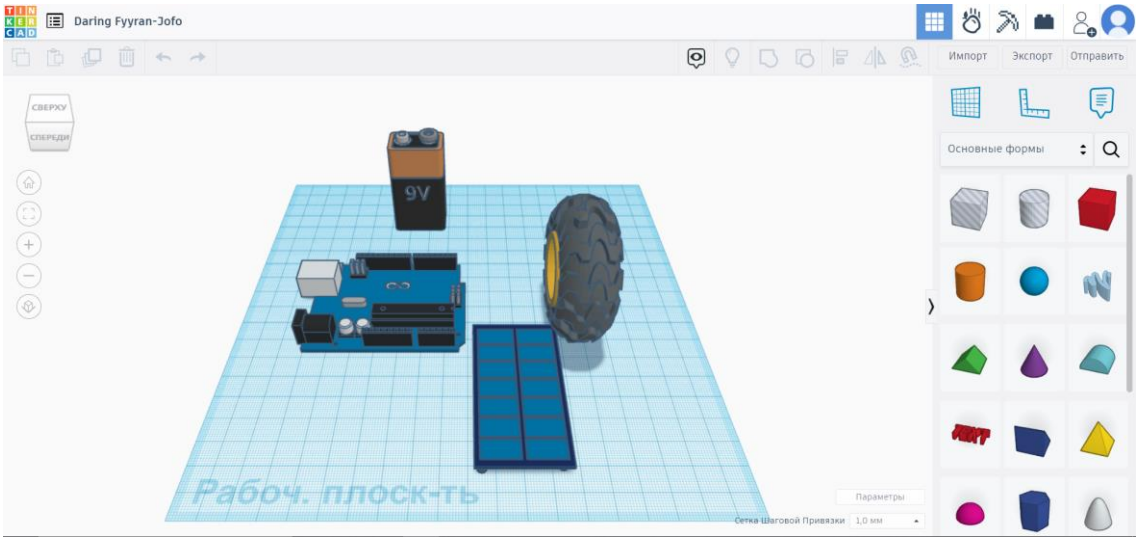

Рисунок 1.2 – Користувацький інтерфейс додатку

Також у Tinkercad можна створювати симуляції руху, що дозволяє перевіряти, як рухаються компоненти та як взаємодіють між собою. Наприклад, ви можете створити симуляцію робота, який переміщується по площадці або піднімає предмети. Система симуляції показує, як модель робота взаємодіє з навколишнім середовищем та перешкодами.

У Tinkercad також можна проводити тестування компонентів та моделей на стійкість, навантаження та інші фізичні властивості. Наприклад, ви можете перевіряти, чи може ваша модель витримати певне навантаження або як будуть змінюватися її параметри при різних умовах.

Загалом, система симуляції та тестування в Tinkercad дозволяє вам віртуально перевірити функціональність та взаємодію вашої 3D-моделі, що дозволяє економити час та ресурси, які були б необхідні для фізичного тестування.

До недоліків застосунку можна віднести:

– обмежена функціональність: Програма не дозволяє налаштовувати властивості компонентів в залежності від їх типу. Наявні лише візуальні налаштування (рисунок 1.3).

– обмежена можливість редагування: Після створення об'єкта в Tinkercad, внесені зміни можуть бути обмеженими, особливо у випадку потреби у докладному редагуванні моделі.

– залежність від Інтернет-підключення: Tinkercad потребує наявності Інтернет-підключення, що може бути незручним у випадку обмеженого або нестабільного зв'язку.

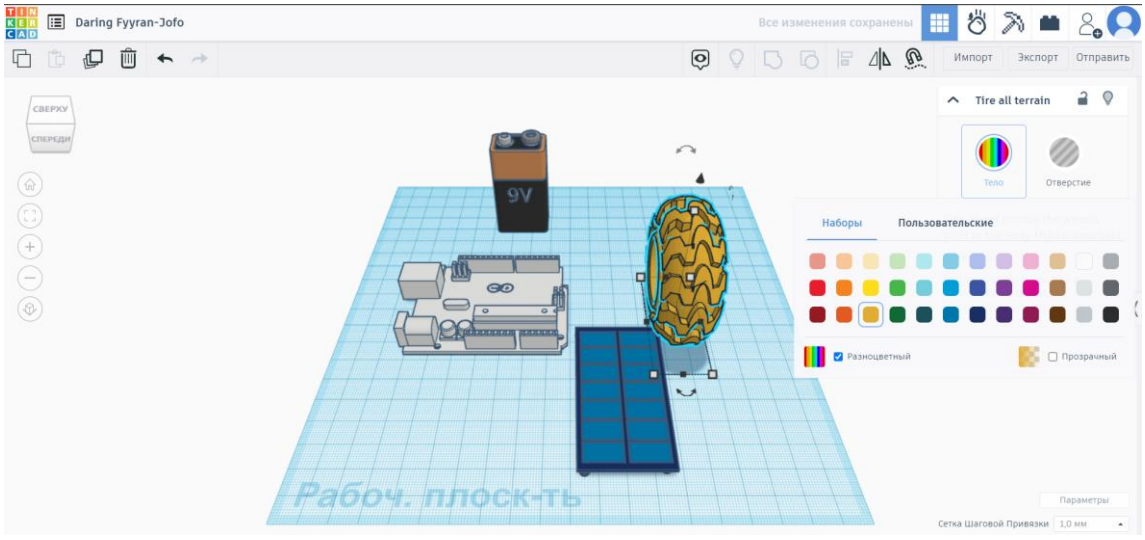

Рисунок 1.3 – Налаштування компоненту колеса

1.6.2 Robot Virtual Worlds

Програмний додаток Robot Virtual Worlds є інтерактивною віртуальною платформою для симуляції та тестування роботів (рисунок 1.4). Основна ідея цього додатку полягає в тому, щоб дозволити користувачам віртуально створювати та тестувати робототехнічні моделі без необхідності фізичного присутності робота.

Однією з переваг Robot Virtual Worlds є можливість тестування побудованих моделей. Користувачі можуть створювати віртуальних роботів, програмувати їх рух та поведінку, а потім тестувати їх у віртуальному середовищі. Це дозволяє перевірити правильність програмування, функціональність робота та взаємодію з навколишнім середовищем [6].

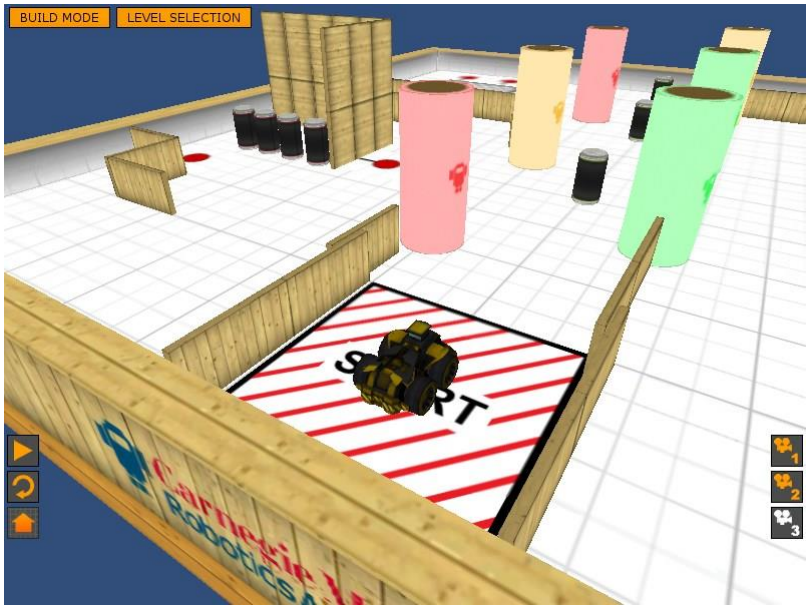

Рисунок 1.4 – Режим тестування у застосунку

Додаток також надає можливість налаштування окремих компонентів роботів. Користувачі можуть встановлювати параметри руху, налаштовувати сенсори та взаємодію з об'єктами в середовищі. Це дозволяє створювати різноманітні сценарії тестування та дослідження робототехнічних систем. Але при цьому не дає змоги отримати інформацію про поточний стан від конкретного елемента під час тестування.

Система симуляції та тестування в Robot Virtual Worlds дозволяє віртуально створювати реалістичні середовища для роботів. Вона включає в себе моделі різних об'єктів та перешкод, з якими робот може взаємодіяти. Користувачі можуть використовувати різні сенсори та актуатори, щоб програмувати поведінку роботів у цих віртуальних середовищах.

Щодо недоліків Robot Virtual Worlds, варто зазначити, що це платний додаток і деякі його функціональності можуть бути доступні лише за додаткову плату. Також, хоча віртуальне середовище дозволяє тестувати та валідувати моделі роботів, воно не може повністю замінити фізичні тестування, оскільки деякі аспекти, такі як фізичні обмеження чи взаємодія з реальними об'єктами, можуть бути складніше відтворити в віртуальному середовищі. Також слід зазначити, що додаток має доволі складний та застарілий інтерфейс

1.6.3 RoboDK

Програмний додаток RoboDK є потужним інструментом для симуляції, візуалізації та тестування робототехнічних систем (рисунок 1.5).

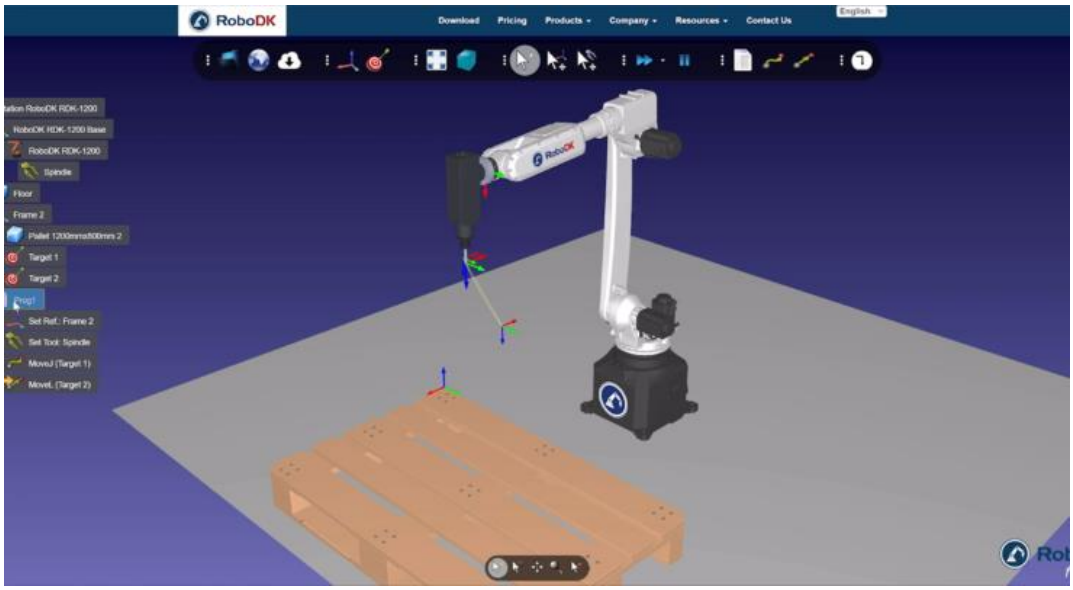

Рисунок 1.5 – Користувацький інтерфейс застосунку

Основні переваги та можливості додатку включають [7]:

– симуляція роботів: RoboDK дозволяє створювати віртуальні моделі роботів та виконувати їх симуляцію у віртуальному середовищі. Це дозволяє користувачам перевіряти та тестувати програми роботів без потреби фізичного наявності реального обладнання;

– візуалізація: Додаток надає візуалізацію роботів у 3D-середовищі. Користувачі можуть переглядати роботів та їх рухи в реальному часі, а також відстежувати їхню точність та колізії;

– програмування роботів: RoboDK має потужний редактор програм для роботів, що дозволяє користувачам створювати, редагувати та відлагоджувати програми роботів. Це дозволяє виконувати різноманітні завдання автоматизації та управління роботами;

– взаємодія з компонентами: RoboDK дозволяє налаштовувати окремі компоненти роботів, такі як інструменти, затискачі та додаткові пристрої. Це дозволяє виконувати детальну настройку та аналіз робототехнічних систем;

– система тестування: RoboDK надає функціональність для виконання тестування роботів. Користувачі можуть визначати вхідні параметри, запускати тестові скрипти та оцінювати результати. Це дозволяє виявляти помилки та вдосконалювати робототехнічні системи.

Щодо недоліків і складнощів, які можуть виникати при використанні RoboDK, вони можуть включати:

– вищий рівень складності: RoboDK є потужним інструментом, і для його ефективного використання може знадобитися певний час та навички. Користувачам може знадобитися деякий час для ознайомлення з функціями та інтерфейсом програми;

– обмежена безкоштовна версія: RoboDK надає безкоштовну версію, але вона має свої обмеження, такі як обмежена кількість моделей та обмеження на функціональність. Деякі додаткові функції можуть бути доступні лише у платній версії;

– інтеграція з реальним обладнанням: Використання RoboDK для симуляції робототехнічних систем може вимагати додаткових зусиль для інтеграції з реальним обладнанням. Підключення реального робота до програми може потребувати налаштування та узгодження параметрів.

1.7 Постановка задачі

Проаналізувавши існуючи рішення, можна прийти до висновку, що їх функціональність досить стисла та не дозволяє розробнику виконати повний процес моделювання, налаштування, тестування та зчитування експериментальних даних. Деякі додатки вже досить застарілі та не підтримуються розробниками.

Тож головною задачею дипломної роботи є розробка віртуального конструктора робототехнічних пристроїв з елементами тестування, який дозволить

розробникам проектувати, моделювати, налаштовувати та тестувати робототехнічні пристрої.

До програмного застосунку поставлені наступні вимоги:

– можливість вибору та розміщення різних компонентів робототехнічного пристрою, таких як сенсори, мотори, керуючі пристрої тощо;

– забезпечення можливості зміни параметрів компонентів, наприклад, швидкості руху моторів, чутливості сенсорів тощо;

– моделювання фізичної взаємодії між компонентами робототехнічного пристрою та їх впливу на роботу пристрою в цілому;

– симуляція поведінки робототехнічного пристрою у віртуальному середовищі з урахуванням фізичних законів.

Для реалізації додатку необхідно:

– розробити зручний користувацький інтерфейс;

– розробити алгоритм конструювання робототехнічних пристроїв;

– розробити алгоритм роботи та налаштування компонентів пристрою;

– розробити модуль тестування робототехнічних пристроїв, що дозволяють візуалізувати їх роботу та функціональність у віртуальному середовищі.

Очікується, що розроблений програмний застосунок буде мати наступні результати:

– інтуїтивно зрозумілий та зручний інтерфейс, який дозволить користувачам легко взаємодіяти з віртуальним конструктором та його компонентами;

– можливість ефективного та точного моделювання, а також симуляції роботи робототехнічних пристроїв у віртуальному середовищі;

– функціональність компонентів тестування, що дозволить виявляти потенційні проблеми та помилки в роботі робототехнічного пристрою.

Таким чином, на основі поставленої задачі та вимог до програмного застосунку можна побудувати діаграму прецедентів (рисунок 1.6).

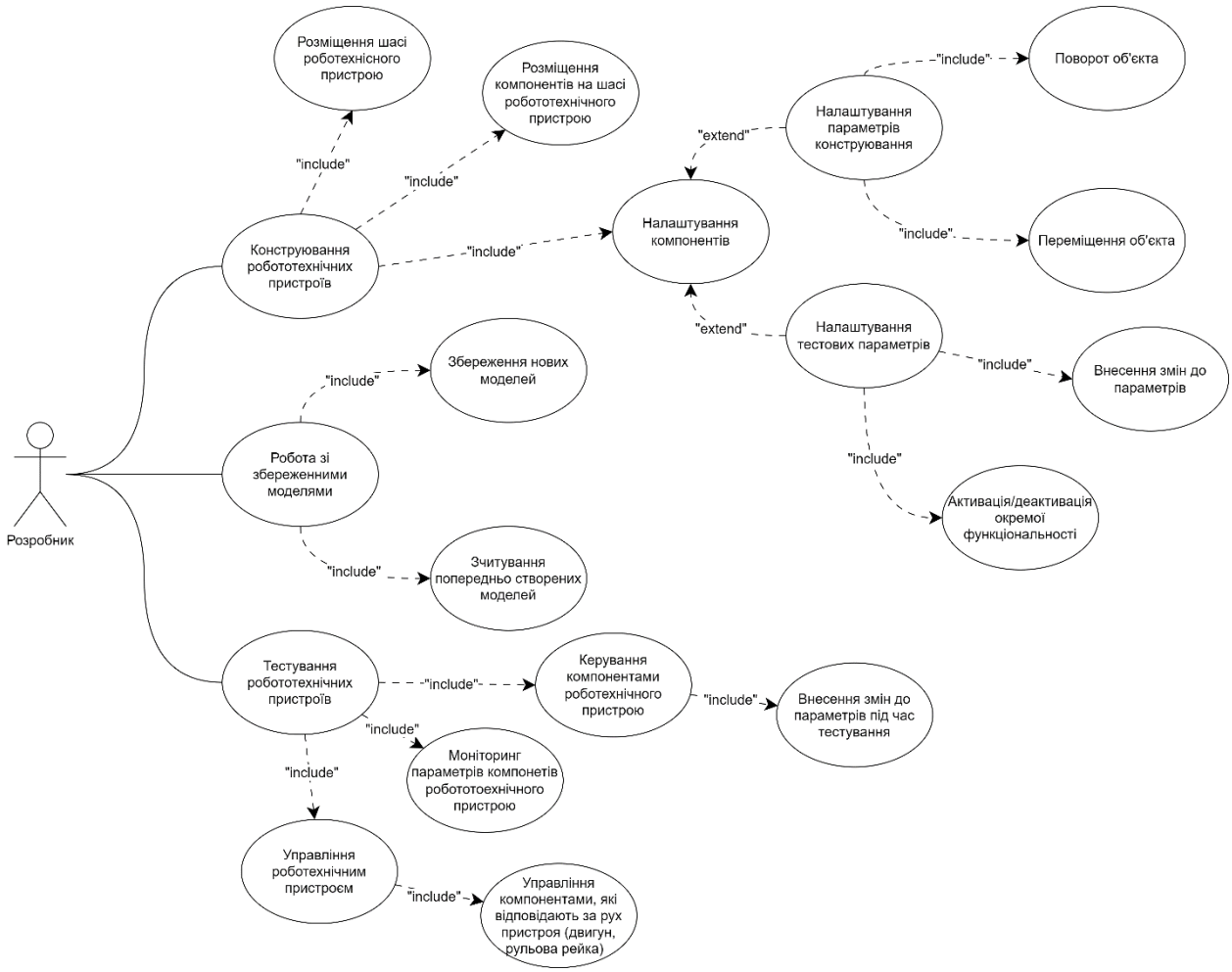

Рисунок 1.6 – Діаграма прецедентів

Дана прецедентів або ж діаграма варіантів використання (use-case diagram) описує функціонал програмної системи, яка розробляється. Вона представляє відношення між акторами, які присутні в системі та відповідними прецедентвами.

Висновки по розділу

У першому розділі було розглянуто поняття робототехніки в сучасному світі, а також сфери її використання. Було детально досліджено питання проектування робототехнічних пристроїв – розглянуті етапи розробки моделей, а також вивчені їх проблеми та недоліки. Вивчена роль тестування у процесі проектування робототехнічних пристроїв, недоліки та проблеми класичних методів проведення екперементів над пристроями.

Також була приділена увага до автоматизації процесу конструювання та тестування робототехнічних пристроїв, запропоноване рішення, яке полягає у створенні віртуального конструктора робототехнічних пристроїв, а також наведені його переваги відносно класичних методів розробки.

У результаті досліджень та аналізу наявних на ринку рішень було виявлено, що вони не мають важливого для даної галузі функціоналу та в більшості застарілі.

Для вирішення проблеми, яка існує на даний момент в сфері рішень з автоматизації процесу конструювання та тестування робототехнічних пристроїв у даній роботі була поставлена задача, яка полягає в розробці віртуального конструктора робототехнічних пристроїв з елементами тестування, який дозволить розробникам проектувати, моделювати, налаштовувати та тестувати робототехнічні пристрої.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Unity як інструмент для створення інтерактивних додатків

У процесі розробки програмного додатку було прийнято рішення використовувати Unity як головний інструмент. Вибір Unity був обґрунтований кількома факторами.

По-перше, Unity є однією з найпопулярніших платформ для розробки ігрових додатків. Вона має велику спільноту розробників, розширену документацію та підтримку.

По-друге, дана платформа є універсальною. Вона дозволяє розробляти як ігри, так і інтерактивні додатки. Ця універсальність дозволяє зосередитися на реалізації функціональності самого додтку, не витрачаючи час на створення власного двигуна або інших складових частин.

2.1.1 Універсальність та переваги Unity

Unity відрізняється від своїх конкурентів кількома ключовими особливостями, які зробили його популярним [8]:

– мультиплатформеність. Це дозволяє створювати кросплатформені додатки, які можуть працювати на різних пристроях без значних змін у коді;

– unity має потужні інструменти для візуалізації і моделювання 3D об'єктів;

– розширена бібліотека компонентів та пакетів, які допомагають розробникам вирішувати різноманітні завдання. Це включає функціонал для реалістичної фізики, створення користувацького інтерфейсу, анімації, звуку та іншого.

Універсальність та переваги Unity в поєднанні з його широкими можливостями створення віртуальних середовищ роблять його ідеальним вибором для розробки інтерактивних додатків. Таким чином він виділяється на фоні його конкурентів.

2.1.2 Архітектура Unity додатку

Основними компонентами архітектури Unity-проекту є сцени, об'єкти, компоненти та скрипти (рисунок 2.1).

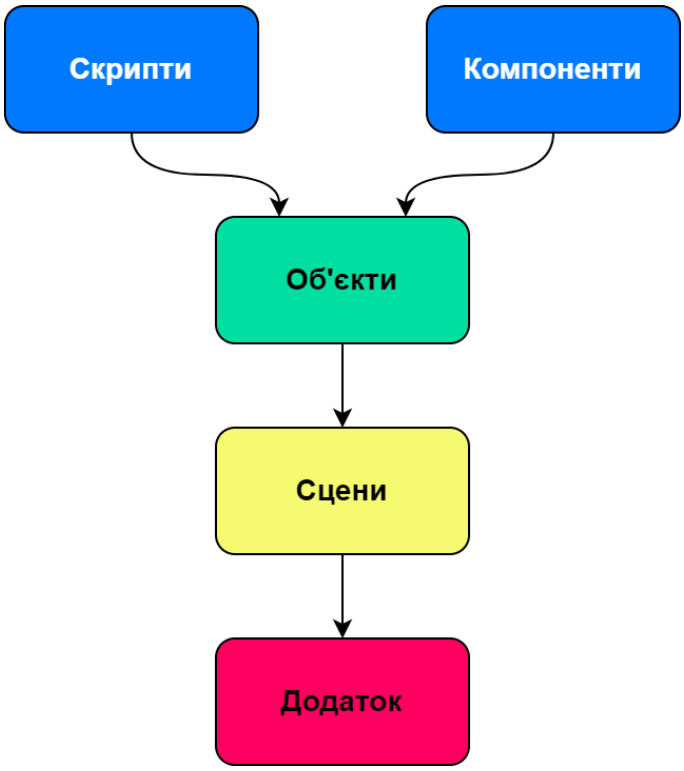

Рисунок 2.1 – Архітектура Unity додатку

Загалом можна виділити кожен окремий елемент [9]:

– сцени: Сцена в Unity представляє собою віртуальне середовище, в якому розгортається додаток. Великі проекти можуть містити кілька сцен, що дозволяє розділити гру на окремі рівні або локації. Сцени використовуються для розміщення та організації об'єктів у грі;

– об'єкти (GameObject): Об'єкти є основними елементами в Unity. Вони можуть бути реалістичними моделями, колізійними тілами, світловими джерелами, частинками та іншими сутностями. Об'єкти можуть бути створені в редакторі Unity або додані програмно за допомогою скриптів;

– компоненти: Компоненти - це розширення функціональності об'єктів в Unity. Кожен об'єкт може мати один або кілька компонентів, які контролюють його

поведінку та властивості. Наприклад, компоненти можуть керувати рухом, графікою, звуком, фізикою, колізіями тощо;

– скрипти: Скрипти використовуються для програмування поведінки об'єктів та управління грою в Unity. Скрипти пишуться на мові програмування C# і дозволяють створювати власні функціональність, обробляти введення користувача, керувати фізикою, анімацією та взаємодіяти з компонентами об'єктів.

2.1.3 Технології та інструменти Unity

Unity має багатий набір технологій та компонентів, які можуть бути використані при розробці програмного додатку [9]:

– користувацький інтерфейс: Unity має потужні засоби для створення інтуїтивно зрозумілого UI. Розробники можуть створювати кнопки, панелі, текстові поля, списки та інші елементи інтерфейсу, які дозволяють користувачам взаємодіяти з інтерактивним додатком. Це дозволяє забезпечити зручний та ефективний спосіб управління та налаштування взаємодіїз додатком;

– фізика: Програмний інструмент також надає потужні можливості для симуляції фізики у віртуальних середовищах. За допомогою вбудованої системи фізики в Unity, розробники можуть створювати реалістичну поведінку об'єктів, враховуючи силу тяжіння, колізії, динаміку руху та інші фізичні властивості;

– анімація: Unity надає засоби для створення різноманітних анімацій, а також інструменти для маніпуляції над ними під час роботи додатку;

– штучний інтелект: Unity має функціональність для реалізації штучного інтелект. Розробники можуть програмувати алгоритми поведінки, прийняття рішень та взаємодії окремих моделей.

2.1.4 Симуляція фізики в Unity

Симуляція фізики в Unity базується на технології, відомій як фізичний двигун Unity (Unity Physics Engine). Фізичний двигун Unity є вбудованим компонентом

Unity, який надає реалістичну симуляцію фізичного руху об'єктів у віртуальному середовищі.

Unity Physics Engine використовує розширений варіант двигуна PhysX, розробленого компанією NVIDIA. PhysX є одним з найпопулярніших фізичних двигунів у галузі відеоігр та симуляційних програм. Він забезпечує точну та швидку симуляцію фізичного руху, враховуючи закони механіки, колізії та взаємодію об'єктів [10].

Фізичний двигун Unity використовує об'єктно-орієнтований підхід до моделювання фізики. Об'єкти, які мають фізичні властивості, такі як маса, форма, матеріал та колізійні області, можуть бути створені та налаштовані у редакторі Unity або за допомогою скриптів.

Під час симуляції фізичного руху, фізичний двигун Unity взаємодіє з об'єктами, враховуючи колізії, тертя, гравітацію та інші фізичні властивості. Він обчислює рух об'єктів на основі прикладених сил, маси та законів механіки.

Unity також надає можливість використовувати фізичні симуляції для створення реалістичних ефектів, таких як руйнування, рідини, симуляція тканин та інше. Unity забезпечує широкий набір інструментів та налаштувань, що дозволяє розробникам створювати різноманітні фізичні ефекти і взаємодію об'єктів у своїх додатках.

В основі фізичного двигуна PhysX лежать базові фізичні рівняння та закони, такі як [11]:

– Закони Ньютона: Фізичний двигун використовує другий закон Ньютона, який говорить, що сила, що діє на об'єкт, дорівнює масі об'єкта помноженій на прискорення. Формула:

$$F = m * a, \tag{2.1}$$

де F - сила, m - маса об'єкта, a - прискорення.

– Закон збереження імпульсу: Фізичний двигун використовує закон збереження імпульсу, який говорить, що сумарний імпульс системи об'єктів залишається постійним, якщо на систему не діють зовнішні сили. Формула:

$$m_1 * v_1 + m_2 * v_2 = m_1' * v_1' + m_2' * v_2', \tag{2.2}$$

де *m* - маса об'єкта, *v* - швидкість об'єкта.

– Рівняння руху: Фізичний двигун використовує рівняння руху, такі як рівняння прямолінійного руху або рівняння руху з постійним прискоренням, для обчислення шляху, швидкості та прискорення об'єктів. Рівняння прямолінійного руху:

$$s = u * t + \frac{1}{2} * a * t^2; \quad (2.3)$$

Рівняння руху з постійним прискоренням:

$$v = u + a * t, \quad (2.4)$$

де *s* - шлях, *u* - початкова швидкість, *a* - прискорення, *t* - час.

– Рівняння колізій: Фізичний двигун використовує рівняння колізій для визначення зіткнень та реакції на них. Рівняння колізій включають розрахунок імпульсу, сили та моменту обертання, що виникають при зіткненні об'єктів.

Фізичний двигун також використовуює різні алгоритми обчислення, такі як метод Ейлера або метод Верле, для симуляції руху та взаємодії об'єктів. Основна ідея цих методів полягає в тому, що вони розбивають час на малий кроки і обчислюють нове становище та швидкість об'єкта на кожному кроці.

Метод Ейлера:

– Обчислення швидкості: Швидкість об'єкта на наступному кроці обчислюється шляхом додавання до поточної швидкості продукту прискорення на поточному кроці помноженого на часовий крок. Формула:

$$v(t + \Delta t) = v(t) + a(t) * \Delta t, \quad (2.5)$$

де *v(t)* - поточна швидкість, *a(t)* - поточне прискорення, *Δt* - часовий крок.

– Обчислення становища: Становище об'єкта на наступному кроці обчислюється шляхом додавання до поточного становища швидкості на поточному кроці помноженої на часовий крок. Формула:

$$s(t + \Delta t) = s(t) + v(t) * \Delta t, \quad (2.6)$$

де *s(t)* - поточне становище, *v(t)* - поточна швидкість, *Δt* - часовий крок.

Метод Верле (також відомий як метод залучення та викидання):

– Обчислення напівкроку швидкості: На початку кожного кроку обчислюється напівкрок швидкості, де швидкість залучається до поточного прискорення. Формула:

$$v(t + \Delta t/2) = v(t) + a(t) * (\Delta t/2), \quad (2.7)$$

де *v(t)* - поточна швидкість, *a(t)* - поточне прискорення, *Δt* - часовий крок.

– Обчислення становища: Становище об'єкта на наступному кроці обчислюється шляхом додавання до поточного становища напівкроку швидкості помноженого на часовий крок. Формула:

$$s(t + \Delta t) = s(t) + v(t + \Delta t/2) * \Delta t, \quad (2.8)$$

де *s(t)* - поточне становище, *v(t+Δt/2)* - швидкість на напівкроці, *Δt* - часовий крок.

– Обчислення кінцевої швидкості: Після обчислення нового становища на кроці *s(t+Δt)*, обчислюється кінцева швидкість на поточному кроці. Формула:

$$v(t + \Delta t) = v(t + \Delta t/2) + a(t + \Delta t) * (\Delta t/2), \quad (2.9)$$

де *v(t+Δt)* - кінцева швидкість, *v(t+Δt/2)* - швидкість на напівкроці, *a(t+Δt)* - прискорення на наступному кроці.

Ці методи дозволяють апроксимувати рух та взаємодію об'єктів у фізичному двигуні. Вони є лише двома з багатьох алгоритмів, які можуть бути використані для симуляції.

В цілому, симуляція фізики в Unity є потужним інструментом, що дозволяє створювати реалістичні та динамічні віртуальні середовища, де об'єкти ведуть себе відповідно до фізичних законів. Це дозволяє розробникам створювати захоплюючі інтерактивні дослідження, симуляції та ігри, які максимально наближені до реального світу.

2.2 Мова програмування C# та її можливості

C# є однією з основних мов програмування, яку використовують для створення ігор та інтерактивних додатків у середовищі Unity.

Мова програмування має декілька важливих можливостей, які можуть бути використані для ефективного розв'язання конкретної задачі зі створення інтерактивного програмного додатку [12]:

– об'єктно-орієнтований підхід: C# є об'єктно-орієнтованою мовою програмування, що дозволяє створювати класи, об'єкти, успадкування та поліморфізм. Цей підхід дозволяє структурувати код, розділяти його на модулі та забезпечує легку розширюваність і підтримку коду;

– широкі можливості бібліотек: C# має багату екосистему бібліотек, які спрощують розробку різноманітних функціональностей;

– інтеграція з Unity: C# є офіційною мовою програмування для розробки в Unity. Це означає, що C# має широку підтримку та інтеграцію з функціональністю, інструментами та середовищем розробки Unity.

2.2.1 Платформа Mono Runtime

Mono Runtime - це реалізація платформи .NET, яка використовується Unity для виконання коду, написаного мовою програмування C#. Він є частиною середовища Unity і відповідає за виконання C#-коду, що контролює поведінку ігрових об'єктів, логіку гри та інші аспекти розробки ігор.

Mono Runtime є кросплатформовим середовищем виконання, що підтримує багато операційних систем, включаючи Windows, macOS, Linux, iOS та Android. Він забезпечує сумісність між Unity та різними платформами, що дозволяє розробникам створювати ігри, які можуть бути запущені на різних пристроях [13].

Виконання коду для цієї платформи відбувається аналогічно до платформи .NET [14] (рисунок 2.2):

Ви пишете код на мові програмування C#.

Код на C# компілюється за допомогою вбудованого компілятора C# і генерує проміжний байт-код або проміжну мову. Це є збіркою (build).

Збірка запускається на цільових пристроях з використанням упакованого середовища виконання Mono (або Common Language Runtime). Mono runtime відповідає за збір сміття, безпеку, обробку винятків тощо.

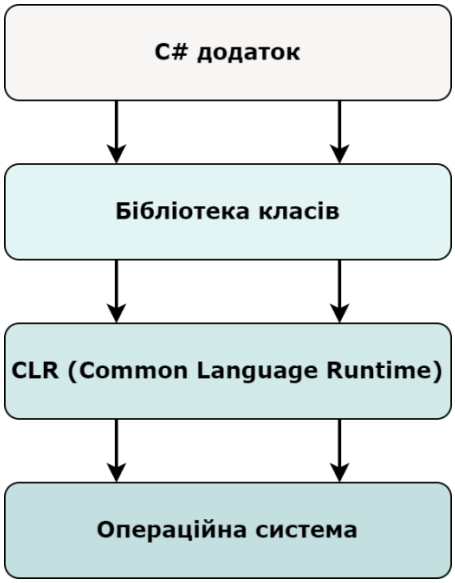

Рисунок 2.2 – Архітектура Mono

2.3 Формат JSON як інструмент збереження

Формат JSON (JavaScript Object Notation) є легким, текстовим форматом, який широко використовується для зберігання та обміну структурованих даних. Він базується на синтаксисі мови JavaScript і здатний представляти різні типи даних, такі як рядки, числа, масиви, об'єкти і булеві значення.

Основна структура JSON-файлу полягає в парах "ключ: значення", де ключі є рядками, а значення можуть бути будь-якого типу даних. Формат JSON дозволяє вкладати дані, створювати вкладені об'єкти та масиви, що робить його дуже гнучким для зберігання складних структур даних [15].

2.3.1 Переваги використання JSON для створення інтерактивних додатків

Формат JSON є одним з найпоширеніших форматів для зберігання та обміну даними у веб-розробці та програмуванні загалом. Використання JSON у

програмному застосунку має переваги, особливо при інтеграції з додатком Бази Даних.

Основна структура JSON-файлу полягає в парах "ключ: значення", де ключі є рядками, а значення можуть бути будь-якого типу даних. Формат JSON дозволяє вкладати дані, створювати вкладені об'єкти та масиви, що робить його дуже гнучким для зберігання складних структур даних.

У контексті віртуального конструктора робототехнічних пристроїв, формат JSON може бути використаний для збереження моделей роботів та їх компонентів. Моделі роботів можуть бути представлені у вигляді об'єктів JSON, де кожен об'єкт містить необхідні атрибути, властивості та параметри робота.

Зберігання моделей роботів в форматі JSON має декілька переваг. По-перше, це дозволяє легко зберігати та передавати дані про робота в одному компактному файлі. Крім того, JSON-файли можуть бути легко зчитані та оброблені з використанням різних мов програмування, оскільки існують багато бібліотек, які підтримують роботу з JSON.

Завдяки простоті та стандартизованості формату JSON, його легко інтегрувати з додатком Бази Даних. Розробники можуть використовувати функції та бібліотеки, доступні у багатьох мовах програмування, для роботи з JSON-даними. Це дозволяє зручно зчитувати, записувати, оновлювати та видаляти дані в форматі JSON у БД.

2.3.2 Використання формату JSON для збереження параметрів компонентів

Параметри кожного компоненту робототехнічного пристрою можна розділити на два основних типи: розташування та налаштування. Таким чином для збереження одного елементу необхідно зазначити його розташування на моделі, а також його тестові параметри, відповідно до його типу. Також слід пам’ятати і про те, що на один елемент можуть бути закріплені і інші, тобто існує деякий перелік компонентів. Виходячи з поставлених вимог документ збереження має включати

(рисунок 2.3): розташування (позиція в просторі), налаштування (відповідно до типу компонента) та прикріплені компоненти.

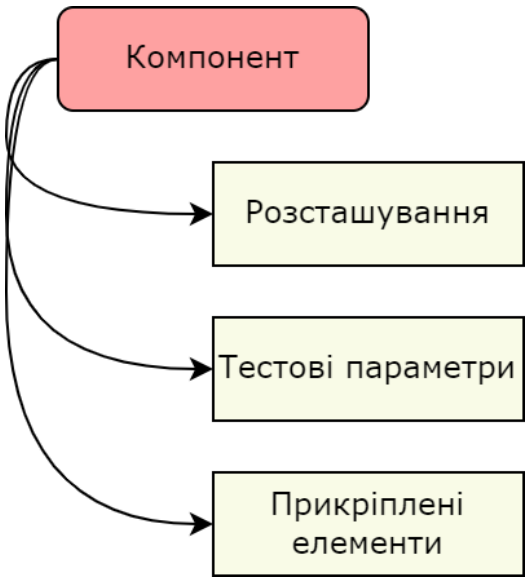

Рисунок 2.3 – Структура документа збереження

Висновки по розділу

Отже, опираючись на поставлене в першому розділі завдання, у другому розділі було проаналізовано технології, які допоможуть для вирішення даної задачі.

Перш за все, було розглянуто інструмент для створення інтерактивних додатків, а саме – Unity. Наведено його основні переваги порівняно з конкурентами, чим обгруновано вибір саме цього інструменту. Також було розглянуто основні технології Unity, такі як: Unity Physics Engine, UI, штучний інтелект, анімація, – які можна використати при створенні програмного застосунку.

Виходячи з вибору основного інструменту, було обрано і мову програмування. Unity на даний момент підтримує лише мову C#, виконання якої відбувається на платформі Mono. Дана платформа є основною для створення інтерактивних додатків на Unity. Розглянуто основні особливості даної мови програмування, а також платформи Mono.

Також у розділі було розглянуто спосіб збереження даних базуючись на тому, що у випадку віртуального конструктора головною задачею збереження є подальше відтворення та використання моделей та їх компонентів окремо. Таким чином формат JSON став найкращим варіантом для використання, адже за допомогою своєї структури – "ключ: значення", де ключі є рядками, а значення можуть бути будь-якого типу даних, він дозволяє вкладати дані, створювати вкладені об'єкти та масиви, що робить його дуже гнучким для зберігання складних структур даних.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Архітектура програмного застосунку

Програмний застосунок складається з трьох основних модулів: конструювання, тестування та збереження даних. Кожен модуль є незалежним та може працювати окремо від інших. При цьому вони знаходяться в постійній взаємодії для повноцінної роботи додатку.

Модуль конструювання відповідає за створення моделей робототехнічних пристроїв. Він містить в собі функціонал, який дозволяє з існуючих в бібліотеці додатку компонентів створювати моделі, надаючи можливість розташовувати кожен елемент так, як це потрібно розробнику. Інженер може обирати як місце розташування компоненту на шасі приладу, так і його поворот навколо власної осі.

Модуль тестування надає розробнику можливість проводити експерименти зі створеною ним моделлю робототехнічного пристрою. При цьому кожен окремий компонент приладу має власні параметри, які користувач може змінювати відповідно до його потреб. Таким чином розробник може задати необхідну чутливість конкретному сенсору, задати потужність електродвигуна і тому подібне. Невід'ємною частиною тестування є симуляція фізики, яка реалізовується за допомогою Nvidia PhysX. Всі елементи моделі підпадають під дію фізичних властивостей та напряму впливають на експерименти.

Модуль збереження надає можливість зберігати та зчитувати існуючі моделі, а також налаштування кожного окремого компоненту. Він також забезпечує зберігання даних під час роботи застосунку використовуючи для цього DTOs, які в тому числі використовуються для зв'язку окремих модулів системи між собою. Створені моделі зберігаються та поширюються у форматі JSON, що дозволяє розробникам відкривати їх на різних пристроях, а також змінювати відповідні фрагменти документу, змінюючи при цьому саму модель.

3.1.1 Процес обробки даних в системі

Загалом взаємодія модулів системи відбувається за допомогою DTOs модулю збереження, таким чином для взаємодії модулю тестування та конструювання завжди використовується модуль збереження (рисунок 3.1).

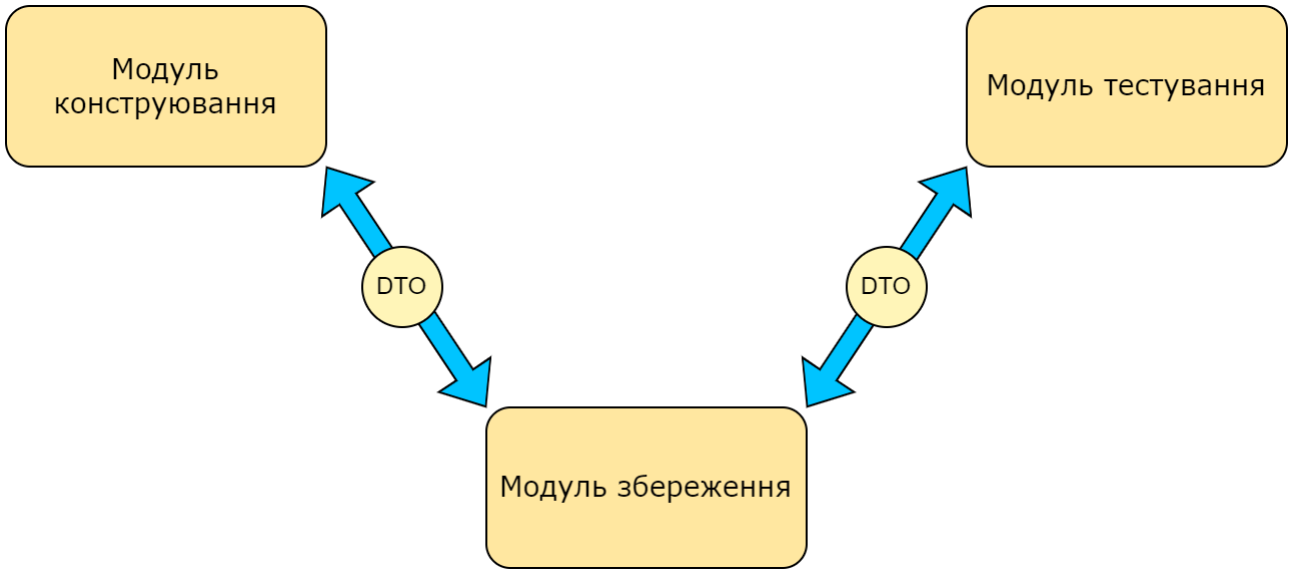

Рисунок 3.1 – Схематичне зображення взаємодії модулів системи

Для опису руху та обробки інформації в системі використовується діаграма потоків даних (DFD). Вона дозволяє візуально продемонструвати всі процеси з точки зору даних. DFD відображає функціональні залежності значень, що обчислюються в системі, включаючи вхідні та вихідні, а також внутрішні сховища даних. Дана діаграма налічує декілька базових елементів: зовнішні сутності – об’єкти, які активно взаємодіють з даними, виробляють і споживають їх; процеси, які обробляють дані, перетворюючи їх; сховища даних – основна ідея яких полягає в тому, що вони пасивно зберігають інформацію; потоки даних, які переносять дані. Таким чином за допомогою DFD можна відобразити повний процес обробки даних у програмному застосунку [16]. Діаграма потоків даних для віртуального конструктора робототехнічних пристрої зображена на рисунок 3.2.

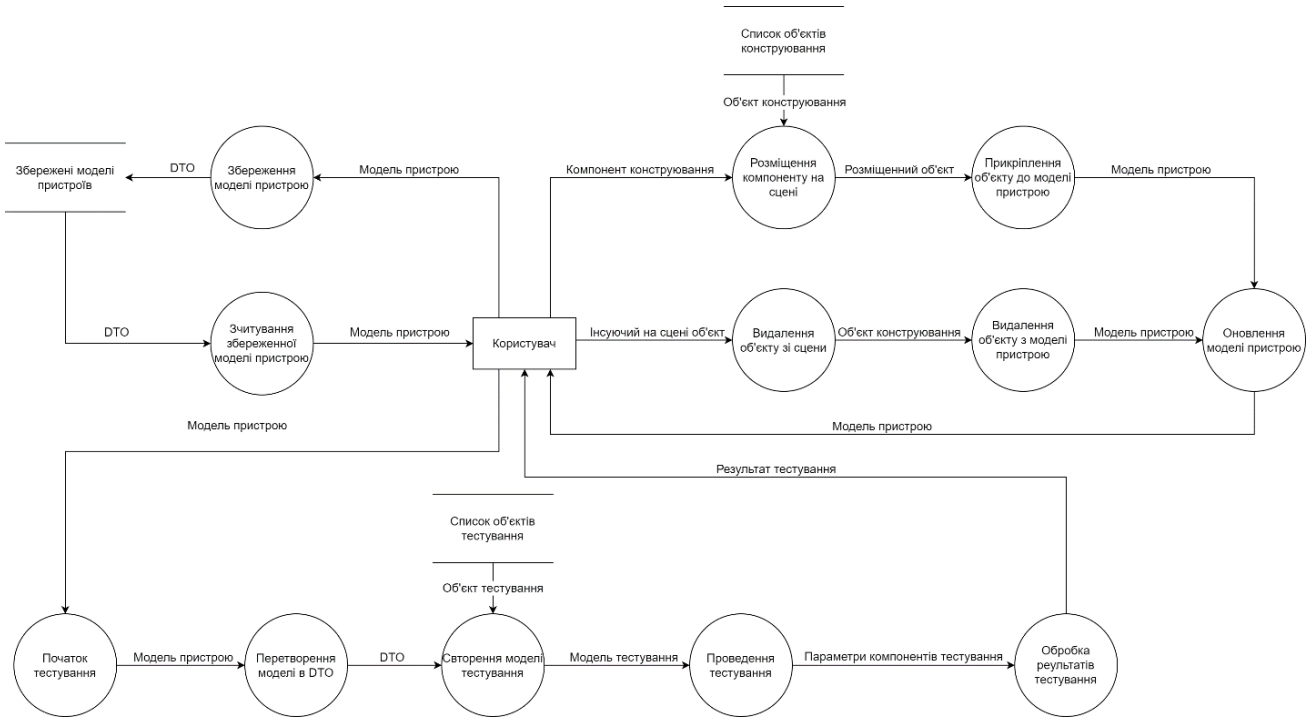

Рисунок 3.2 – Діаграма потоків даних

3.1.2 Опис базових класів системи

В системі можна виділити декілька базових класів, які є основними для кожного модулю: конструювання – BuildingController, тестування – TestingController, збереження – SaveHandler. Вони відповідають за роботу цільових механік та взаємодіють з іншими класами (рисунок 3.3). BuildingController працює з об’єктами класу BuildingObject, який описує базові параметри елементів конструювання та є абстрактним. Саме від нього наслідуються інші типи, які містять необхідний функціонал для розміщення та оперування компонентами на сцені конструювання. Подібним чином TestingController оперує об’єктами типу TestingObject, який в свою чергу також є абстрактним. Його потомки описують роботу окремих елементів тестування та містять поля, на які опираються компоненти під час процесу симуляції. SaveHandler в свою чергу напряму працює з DTO та має повний набір методів для їх обробки та конвертації в формат JSON.

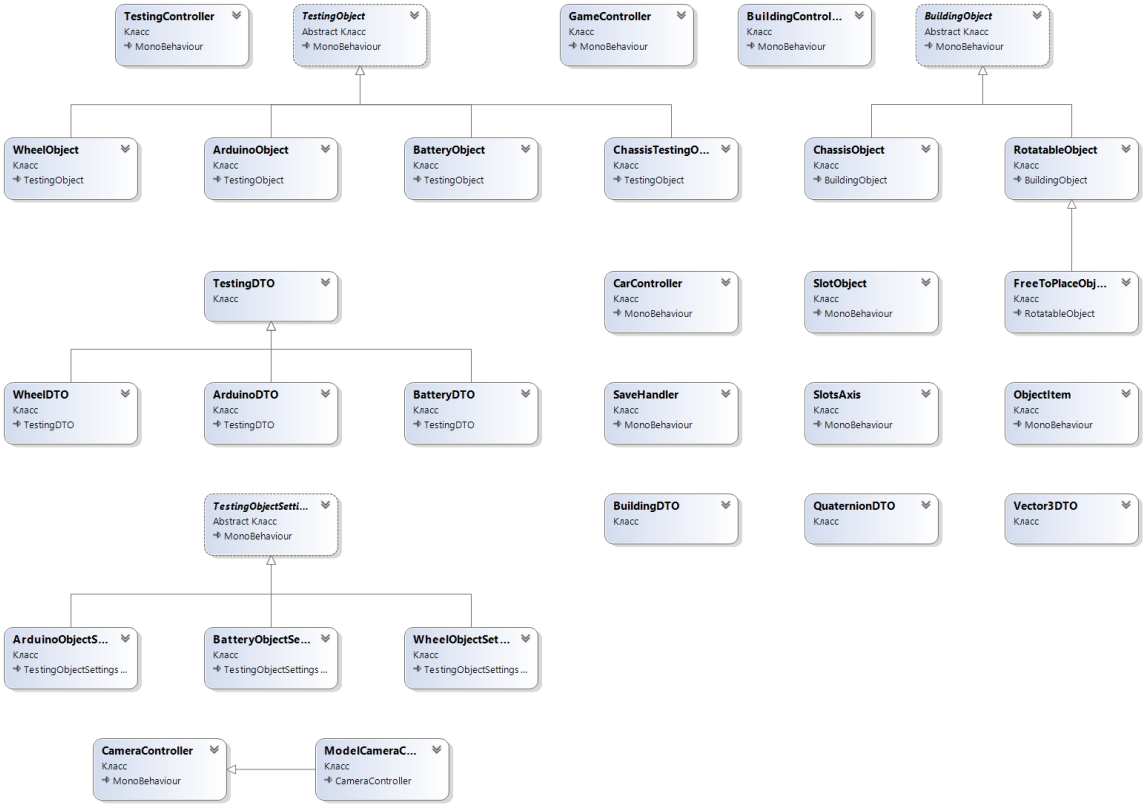

Рисунок 3.3 – Діаграма класів

Unity проект налічує в собі дві основні сцени, а саме: конструювання та тестування. Кожна з них надає можливість користувачу взаємодіяти з відповідним модулем та використовувати його функціонал для виконання поставленої задачі. Для переходу між ними та обміном даних застосовується клас GameController, який напряму взаємодіє з DTO для зберігання інформації під час виконання програми.

3.1.3 Бізнес-процеси програмного застосунку

Для відображення бізнес-процесів загалом використовують стандарт BPMN, який дозволяє описати роботу системи за допомогою діаграми. Дани стандарт налічує базовий набір інтуїтивно зрозумілих елементів, які дозволяють визначити складні семантичні конструкції. Моделювання BPMN здійснюється з використанням базових діаграм, які складаються з невеликої кількості графічних елементів та описують загальну логіку окремого процесу. Будь-який процес завжди починається з якоїсь стартової події та закінчується кінцевими. Таким чином,

BPMN є універсальним стандартом, який надає можливість як технічним спеціалістам, так і звичайним користувачам швидко зрозуміти логіку процесу [17]. BPMN діаграма для віртуального конструктора робототехнічних пристроїв зображена на рисунку 3.4.

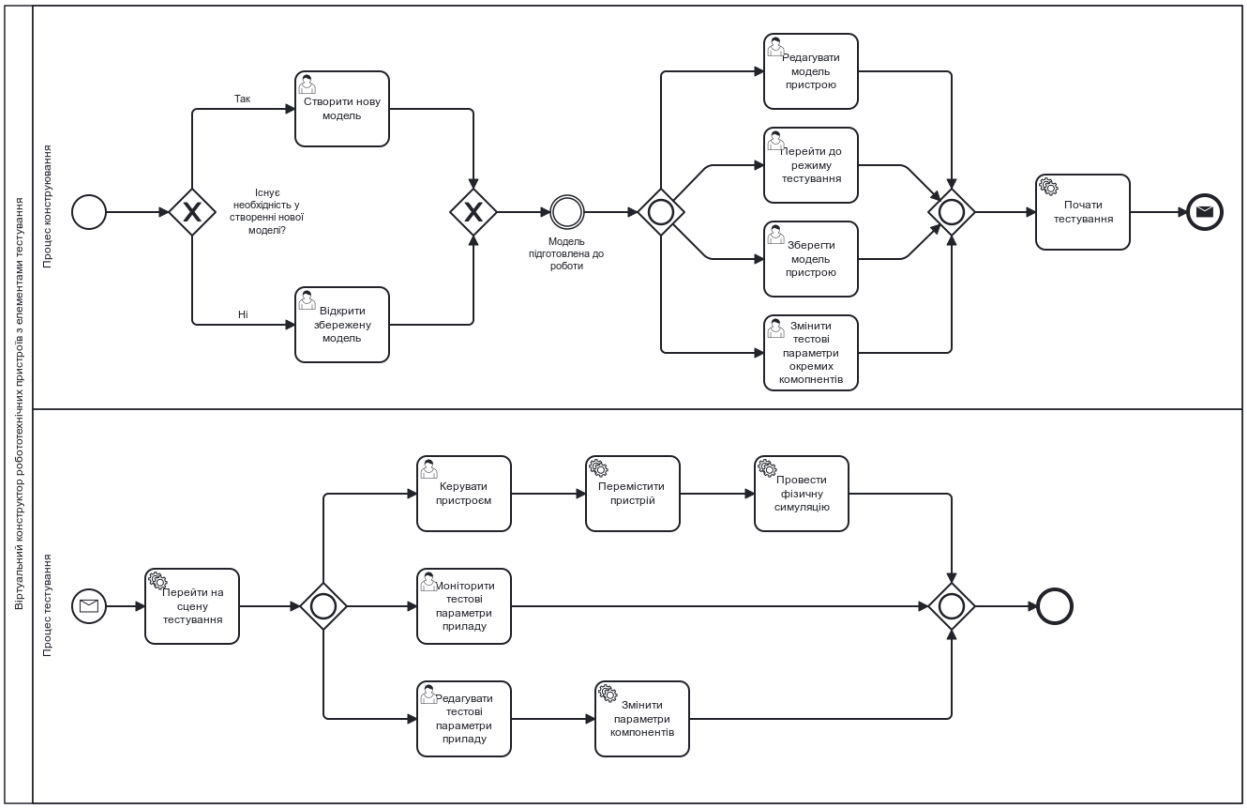

Рисунок 3.4 – BPMN діаграма

3.2 Програмний модуль конструювання

Головною задачею модуля конструювання є надання користувачеві функціоналу для створення моделі робототехнічного пристрою, використовуючи для цього існуючі в системі компоненти. Загалом, дана система налічує декілька основних класів, які виконують базові запити, а також декілька допоміжних, що займаються менш значимими процесами. Можна виділити наступні функціональні можливості даного модулю:

– вибір необхідного компоненту зі списку існуючих;

– розташування компоненту у будь-якому необхідному положенні на шасі пристрою;

– редагування положення та повороту компоненту на сцені;

– видалення компоненту зі сцени.

3.2.1 Елементи конструювання

Клас BuildingController є основним, з яким взаємодіє користувач, саме він приймає вхідні запити на розташування або видалення об’єкту зі сцени. Він контролює весь процес, зберігаючи повну інформацію про дії розробника, що пов’язані з конструюванням. Але слід зазначити, що він містить функціонал, який дозволяє лише керувати елементами тестування, та не приймає участі в самому процесі.

Для взаємодії з користувачем в процесі розташування елементів на сцені використовується клас BuildingObject. Він є абстрактним та містить в собі опис основних параметрів будь-якого елементу іншого елементу конструювання (рисунок 3.5).

Рисунок 3.5 – Опис класу BuildingObject

На основі класу BuildingObject будуються три інших, які є основою всього процесу конструювання, а саме: ChassisObject, RotatableObject та FreeToPlaceObject. Розглянемо кожен з них окремо:

– ChassisObject – об’єкт, що описує шасі пристрою, тобто його основний початковий елемент.

– RotatableObject – елемент з яким користувач може власноруч взаємодіяти, обертаючи його навколо власної осі, використовуючи для цього клавіатуру його пристрою ПК.

– FreeToPlaceObject – це об’єкт, який наслідує можливості класу RotatableObject та при цьому містить власну механіку. Його функціонал дозволяє користувачу розташовувати елемент на сцені, використовуючи для цього маніпулятор його ПК.

FreeToPlaceObject є широкопоширеним у программному застосунку та використовується майже для всіх існуючих компонентів, доступних для конструювання.

Не менш значущим класом у системі конструювання є SlotObject (рисунок 3.6). Він реалізовує іншу механіку прикріплення та розташування елементів на сцені, дозволяючи розташувати на собі елемент лише в одній заданій позиції.

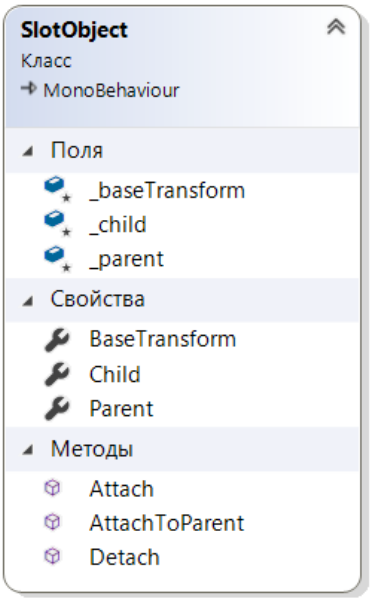

Рисунок 3.6 – Опис класу SlotObject

SlotObject не наслідується від BuildingObject, а лише тісно взаємодіє з ним. Така реалізація обумовлена тим, що один BuildingObject може мати в собі деякий масив SlotObject, тобто один об’єкт конструювання, який є деякою 3D моделлю, дозволяє розташувати на собі інший елемент як на будь-якій своїй площині, так і на спеціально створеному для цього місці – слоті. Даний підхід дозволяє спростити створення моделі, так як розробник не повинен турбуватися про симетричне розміщення таких елементів як наприклад коліс, адже вони можуть бути прикріплені у спеціально створені для цього слоти.

Слід наділити увагою і клас ObjectItem (рисунок 3.7), який є невід’ємною частиною будь-якого BuildingObject. Він відіграє важливу роль в процесі конструювання, так як надає користувачу можливість взаємодіяти з розміщеними ним елементами на сцені. ObjectItem активно слідкує за діями розробника і у випадку коли користувач за допомогою маніпулятора свого ПК, обере якийсь елемент на сцені, то саме даний клас подасть подальший сигнал про те з яким саме об’єктом відбулася взаємодія так яка саме до відповідного BuildingObject.

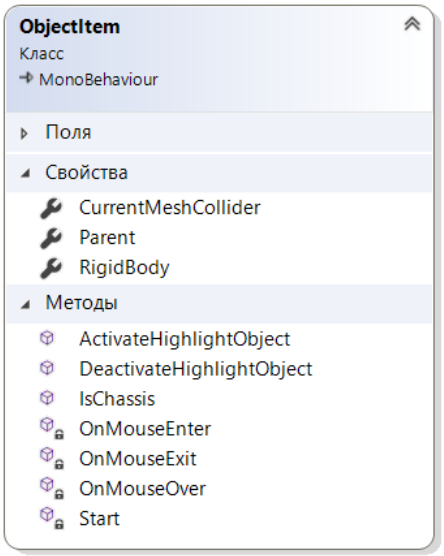

Рисунок 3.7 – Опис класу ObjectItem

Описані класи є основними для модулю конструювання, саме на них базується весь функціонал даної системи. Також використовуються і інші класи,

але вони створені для виконання менш глобальних задач, які не впливають на функціонування системи в цілому.

3.2.2 Параметри елементів конструювання

Як вже було сказано основним класом, який взаємодіє з користувачем під час конструювання моделі є BuildingObject, саме він налічує основні параметри будь-якого іншого подібного об’єкту. Саме вони визначають положення елементів моделі як під час тестування та і для збереження. Загалом, можна виділити наступні параметри:

– розташування: визначається як координати об’єкту в 3D просторі;

– поворот: визначається як кватерніон – чотиривимірне гіперкомплексне число, що складається з чотирьох основних компонентів: перші три – вісь повороту, а останній – його кут;

– батьківський об’єкт: це елемент на площині 3D моделі якого був розташований даний;

– список прикріплених об’єктів: масив елементів, які розташовані на площині 3D моделі даного;

– слот: якщо об’єкт прикріплений до якогось зі слотів батьківського елементу, то вказується його ID в рамках даного компоненту.

3.2.3 Механізм конструювання

Як відомо основним класом, який описує механіку розташування об’єктів на сцені є клас FreeToPlaceObject. Він містить в собі метод PlaceObject, який в свою чергу включає в себе алгоритм прикріплення однієї 3D моделі до іншої (рисунок 3.8). Основною ідеєю даної механіки є: виявлення бажаного користувачем місця розташування компоненту на основі положення його маніпулятора та розміщення його в даному положенні. Для виконання поставленої задачі алгоритм виконує наступні дії:

– розраховує точку на яку вказує маніпулятор користувача;

– розраховує якому об’єкту належить дана точка;

– розраховує нормаль поверхності на якій знаходиться дана точка;

– на основі знайденої нормалі обраховується зміна положення та повороту об’єкта, шо розміщується.

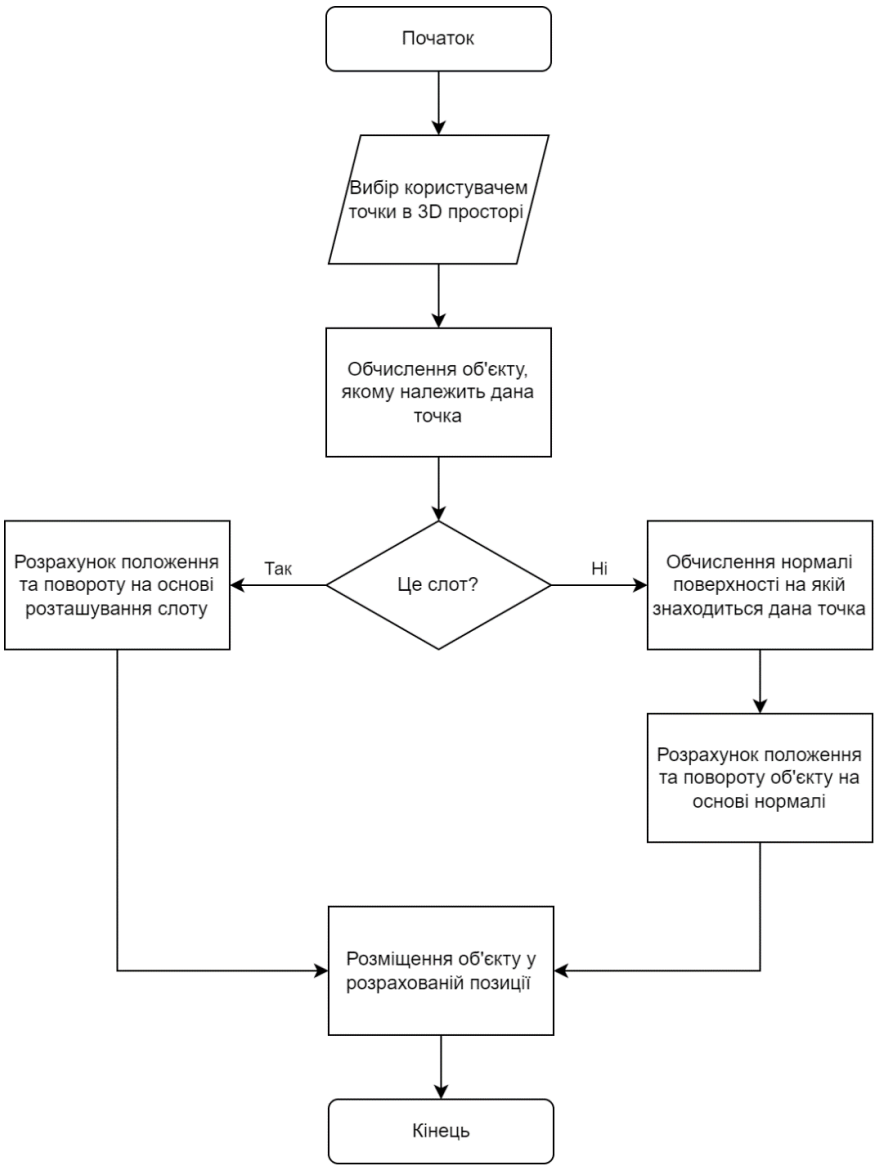

Рисунок 3.8 – Блок-схема роботи алгоритму розміщення об’єкта конструювання на сцені

Також слід зазначити, що у випадку коли користувачем був обраний об’єкт типу SlotObject, елемент конструювання буде розміщений у відповідному слоті.

Для знаходження точки на яку вказує маніпулятор користувача використовується модуль Physics Unity, а саме метод Raycast, який використовує промінь для виявлення перетину з фізичними об'єктами в сцені. Після знаходження перетину, функція повертає інформацію про точку перетину, нормаль до поверхні, з якою перетинається, та інші параметри.

Для розрахунку нормалі до поверхні при використанні Unity Raycast в 3D просторі використовується методика відома як "нормалізація нормалі". Основна ідея полягає в тому, що Unity обчислює нормалі для поверхні в точці перетину, використовуючи інформацію про геометрію цієї поверхні. Для цього використовуються вектори нормалі, які перпендикулярні до поверхні в кожній точці. Формула для розрахунку нормалі у 3D просторі зазвичай базується на векторному добутку векторів, які лежать на поверхні. Припустимо, що ми маємо три точки - A, B і C. Ці точки визначають дві сторони трикутника на поверхні. Для обчислення нормалі, ми можемо використати наступний алгоритм:

– обчислити вектори AB і AC:

$$AB = B - A, \tag{3.1}$$

$$AC = C - A, \tag{3.2}$$

– виконати векторний добуток між векторами AB і AC:

$$normal = AB \times AC, \tag{3.3}$$

Векторний добуток визначається як:

$$normal.x = AB.y * AC.z - AB.z * AC.y, \tag{3.4}$$

$$normal.y = AB.z * AC.x - AB.x * AC.z, \tag{3.5}$$

$$normal.z = AB.x * AC.y - AB.y * AC.x, \tag{3.6}$$

Отримана нормаль буде вказувати у напрямку перпендикулярно до поверхні.

– нормалізувати отриману нормаль. Нормалізація полягає у зміні довжини вектора на одиницю, за допомогою наступної формули:

$$normalized = \frac{vector}{length}, \tag{3.7}$$

Довжина вектора розраховується як:

$$length = \sqrt{(vector.x^2 + vector.y^2 + vector.z^2)}, \tag{3.8}$$

Даний алгоритм дозволяє отримати нормаль до поверхні в точці перетину з нею променя. Unity внутрішньо виконує ці розрахунки, коли використовується Physics.Raycast.

3.3 Програмний модуль тестування

Модуль тестування дозволяє розробнику проводити експерименти зі створеною ним моделлю робототехнічного пристрою. Дана система складається з двох підсистем. Перша з них надає користувачу можливості прямої взаємодії, тобто надає інструмент керування компонентами, що можуть рухатися та піддаються впливу оператора, наприклад мотори та рульові рейки. Друга підсистема дозволяє розробнику втручатися в процес фізичних обчислень шляхом зміни параметрів тестових компонентів, які напряму впливають на симуляцію. Це може бути як зміна радіусу дії якогось із сенсорів, так і налаштування жорсткості пружин підвіски моделі. Невід'ємною частиною даної системи також є фізичний двигун, адже саме з ним взаємодіють тестові компоненти.

Можна виділити наступний функціонал модулю тестування:

– керування моделлю за допомогою клавіатури;

– перегляд та редагування тестових параметрів кожного окремого компоненту;

– симуляція фізики.

Такий набір можливостей дозволяє провести всі необхідні експерименти, надаючи при цьому можливість прямого втручання у процес тестування.

3.3.1 Компоненти тестування

Основним класом даної системи є TestingController, який ставить перед собою основну задачу – керувати процесом тестування. Він ні яким чином не взаємодіє з фізичним двигуном, а лише з об'єктами типу TestingObject. З цього виходить, що TestingController має доступ до параметрів кожного окремого

компоненту, які він може вільно редагувати, але при цьому не може ніяк напряму вплинути на процес симуляції. Даний клас використовується саме для надання користувачу можливості змінювати тестові параметри необхідного об‘єкту і TestingObject допомагає йому в цьому.

Клас TestingObject описує те, як повинен виглядати кожен тестовий компонент та надає методи для взаємодії з його параметрами (рисунок 3.9). При цьому він є абстрактним, тобто лиш створює шаблон для подальших класів.

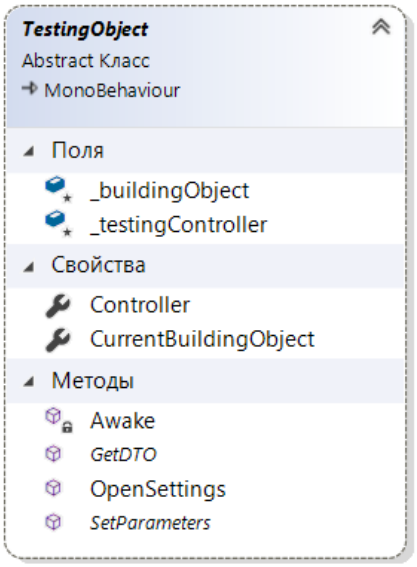

Рисунок 3.9 – Опис класу TestingObject

Від TestingObject наслідуються вже типи, які напряму описують кожен окремий тип компонентів, наприклад: колесо, мотор, датчик і тому подібні. Такі класи можна легко створювати, описуючи конкретний тип та його параметри.

Розглянемо для прикладу детальніше класс WheelObject (рисунок 3.10), який описує загальні параметри колеса. Загалом, він містить в собі декілька основних інструментів:

– створення DTO, за допомогою якого в подальшому легко ділитися даними про конкретний тестовий компонент;

– задання параметрів, при якому за допомогою DTO отримується ряд значень, які в подальшому привласнюються компоненту фізичного двигуна або також можуть бути застосовані і для інших механік компонента.

```
public class WheelObject : TestingObject
{
    [SerializeField]
    protected WheelCollider _wheelCollider;

    public WheelCollider CurrentWheelCollider { get { return _wheelCollider; } }

    public override object GetDTO() {
        return new WheelDTO(this);
    }

    public override void SetParameters(object DTO) {
        WheelDTO wheelDTO = DTO as WheelDTO;
        if (wheelDTO != null)
        {
            if (CurrentWheelCollider != null)
            {
                CurrentWheelCollider.wheelDampingRate = wheelDTO.wheelDampingRate;
                CurrentWheelCollider.suspensionDistance = wheelDTO.suspensionDistance;
                CurrentWheelCollider.forceAppPointDistance = wheelDTO.forceAppPointDistance;

                JointSpring suspensionSpring = CurrentWheelCollider.suspensionSpring;
                suspensionSpring.spring = wheelDTO.spring;
                suspensionSpring.damper = wheelDTO.damper;
                suspensionSpring.targetPosition = wheelDTO.targetPosition;
                CurrentWheelCollider.suspensionSpring = suspensionSpring;
            }
        }
    }
}
```

Рисунок 3.10 – Лістинг класу WheelObject

Таким чином за домогою TestingObject можна створювати об’єкти, для зміни тестових параметрів яких не потрібно буде заглиблюватися до фізичного двигуна.

3.3.2 Налаштування компонентів тестування

TestingController поміж іншим надає користувачу можливість взаємодіяти з параметрами кожного окремого TestingObject. Для цього використовуються його два основні методи: GetDTO та SetParameters. З їх допомогою клас тестового об’єкту легко відтворити як UI елемент, а після задання значень користувачем – застосувати їх.

Особливістю даної системи в порівнянні з іншими подібними – це відсутність логічних операторів у самому алгоритмі, замість них використовуються узагальнені типи. Такий підхід значно полегшує подальшу розробку, а головне покращує роботу з кодом.

Загалом алгоритм роботи даної системи можна описати так (рисунок 3.11):

– користувач надає запит на редагування одного із компонентів;

– параметри компоненту переводяться у DTO типу object;

– DTO направляється до UI компоненту, який має власний масив вікон для кожного типу;

– виконується порівняння типів і таким знаходиться необхідний для відкриття UI елемент.

Рисунок 3.11 – Блок-схема роботи алгоритму виведення параметрів тестового компоненту

Для задання цих параметрів вже не потрібно виконувати такий перелік дій, так як до UI вікна прикріпляється TestingObject об’єкт, якому вони належать. Таким чином можна легко їх застосувати.

3.3.3 Симуляція тестування

Для симуляції використовують базові класи для кожного окремого типу шасі робототехнічного пристрою, які напряму взаємодіють лише з об’єктами типу TestingObject. Дані класи оперують вводом користувача для керування створеними для цього компонентами, такими як: двигуни, колеса, рульові рейки і тому подібні.

Розглянемо їх детальніше на прикладі класу CarController, який надає функціонал для управління пристроями, які побудовані на шасі транспортних механізмів та складаються в основному з парної кількості коліс. CarController дозволяє користувачу керувати моторами та рульовою рейкою пристрою для переміщення його по тестовій сцені, для цього він взаємодіє з класом типу WheelsAxis (рисунок 3.12). Цей клас відображає в собі одну колесну вісь та надає функціонал для оперування об’єктами типу WheelObject, а саме його тестовим компонентом WheelCollider, який є частиною фізичного двигуна Unity.

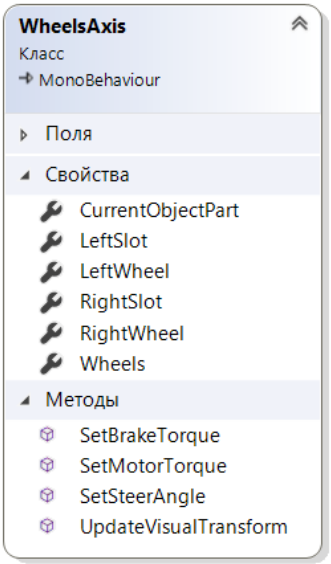

Рисунок 3.12 – Опис класу WheelsAxis

Важливо зазначити, що основним компонентом фізичного двигуна Unity є Rigidbody, який використовується для симуляції реалістичної фізики. Він відповідає за моделювання поведінки об'єктів під впливом фізичних сил, таких як: гравітація; сили, що діють на об'єкт; колізії та інші фактори. Rigidbody займається основними розрахунками та тісно взаємодіє з іншими компонентами фізичного двигуна для цього [18].

Також не менш важливим компонентом у процесі симуляції є об’єкт типу Collider, який прикріплюється до кожного GameObject, для якого необхідно обчислювати зіткнення. У даному випадку Collider прикріплений до об’єкту шасі пристрою. Об'єкт типу Collider у Unity є компонентом, який використовується для визначення геометрії та взаємодії об'єкту з фізичною системою. Він визначає область або форму об'єкту, з якою інші об'єкти можуть взаємодіяти. Колайдери використовуються в фізичній системі Unity для обчислення зіткнень та взаємодій між об'єктами. При виявленні зіткнення фізична система обробляє геометрію колайдерів та виконує розрахунки, щоб визначити, як об'єкти повинні взаємодіяти між собою. У Unity для обчислення колайдерів в 3D просторі використовується алгоритм "Separating Axis Theorem" (SAT) [19]. Цей алгоритм є широко використовуваним і ефективним способом виявлення зіткнень між простими геометричними формами, такими як прямокутники, сфери, капсули та інші.

У загальному випадку, алгоритм SAT працює наступним чином:

– вхідні дані: Два об'єкти, для яких потрібно перевірити зіткнення, представлені своїми колайдерами (Collider-ами) у вигляді простих геометричних форм;

– перевірка осей розділення: Для кожної осі, що проходить через об'єкти, обчислюються проекції колайдерів на цю ось. Потім перевіряється, чи перетинаються проекції цих колайдерів на цій осі. Якщо проекції не перетинаються ніде, це означає, що об'єкти не зіткнулися;

– повторення для всіх осей: Попередній крок повторюється для кожної осі, що є ортогональною до поверхні колайдерів. Це включає осі, паралельні граням колайдерів;

– виявлення зіткнення: Якщо на будь-якій осі виявлено перетин проекцій колайдерів, це означає, що об'єкти зіткнулися. В іншому випадку, якщо перетини не виявлено на жодній осі, об'єкти не зіткнулися.

Слід зазначити, що формули обчислень для алгоритму SAT в залежності від типу колайдерів можуть змінюватись.

WheelCollider наслідується від класу Collider та містить в собі функціонал, який дозволяє керувати поточним станом деякого фізичного тіла, що уособлює колесо, змінюючи для цього параметри потужності його двигуна, гальм, а також повороту рульової рейки. Зміна цих значень напряму впливає на фізичний двигун, примушуючи даний об’єкт діяти відповідно до заданих параметрів та штовхати його Rigidbody. У загальному, Unity WheelCollider використовує комплексний підхід до моделювання фізики коліс, який базується на фізичних принципах і обчислювальних методах, щоб створити симуляцію реалістичної роботи колеса.

3.4 Модуль збереження

Модуль збереження відіграє важливу роль у процесі роботи всього додатку, адже саме за допомогою його DTO спілкуються інші модулі. При цьому його основною задачею, все ж необхідно назвати саме збереження моделей робототехнічних пристроїв в формат JSON та їх зчитування.

Основним класом, який займається даним процесом є SaveHandler, саме він описує взаємодію програмного застосунку з документами типу JSON, а також роботу з іншими модулями. На відміну від BuildingController та TestingController він не просто керує процесом, а й виконує основні механіки: збереження та зчитування. Для виконання поставлених перед ним задач SaveHandler використовує DTO, які можна роділити на два типи:

– BuildingDTO – це об’єкт, який зберігає в собі параметри конструювання окремого компоненту, які описані в прикріпленому до нього типі BuildingObject;

– TestingDTO – зберігає інформацію про тестові параметри окремого компоненту TestingObject, які є досить індивідуальними для кожного окремого типу.

Важливо зазначити, що TestingDTO завжди є частиною BuildingDTO і вони виступають як одне ціле, адже відносяться до одного об’єкту та описують саме його.

3.4.1 Data Transfer Objects (опис об’єктів)

Розглянемо детальніше DTO кожного типу, їх використання та особливості. Спершу необхідно почати з класу BuildingDTO (рисунок 3.13), так як він є основним у програмному застосунку.

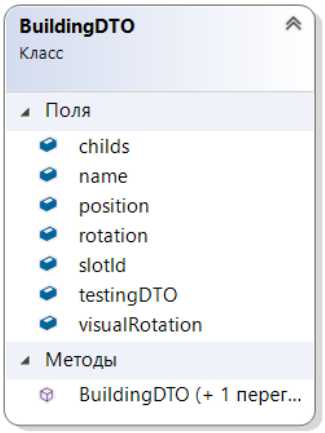

Рисунок 3.13 – Опис класу BuildingDTO

Даний тип складається з усіх основних параметрів класу BuildingObject, а також об'єкту типу TestingDTO. Такий набір значень дозволяє легко створити подібний до зберігаємого компонент у будь який момент виконання програми.

Слід зазначити, що позиція та поворот зберігаються як об’єкти типу VectorDTO та QuaternionDTO відповідно (рисунок 3.14). Вони не є частиною Unity та створені по причині того, що базові класи Vector та Quaternion є досить складними і зберігання їх в такому виді лиш ускладнює розробку та роботу з

даними. Дані DTO є досить простими та складаються лише з кількох числових полів. У випадку VectorDTO – це положення об’єкту по кожній з осей кординат, а для QuaternionDTO – осі повороту по кожній з осей та його кут. Вони не мають базового класу, але мають подіні методи, що дозволяють привести до їх типу стандартні Vector та Quaternion і навпаки

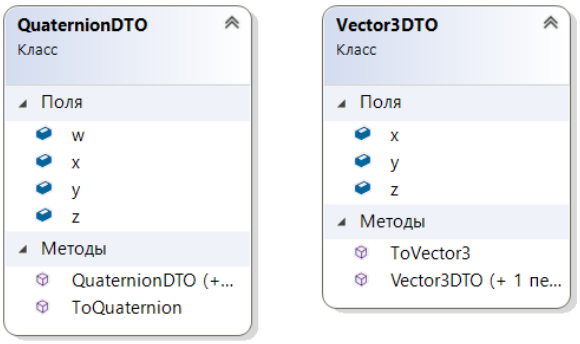

Рисунок 3.14 – Опис класів VectorDTO та QuaternionDTO

TestingDTO у свою чергу є базовим класом та не має в своїй основі якихось параметрів. Він є батьківським класом для DTO всіх типів TestingObject. Розглянемо для прикладу клас WheelDTO (рисунок 3.15).

```
[Serializable]
public class WheelDTO : TestingDTO
{
    public float wheelDampingRate;
    public float suspensionDistance;
    public float forceAppPointDistance;
    public float spring;
    public float damper;
    public float targetPosition;

    public WheelDTO() { }

    public WheelDTO(WheelObject wheelObject) {
        if (wheelObject.CurrentWheelCollider != null)
        {
            wheelDampingRate = wheelObject.CurrentWheelCollider.wheelDampingRate;
            suspensionDistance = wheelObject.CurrentWheelCollider.suspensionDistance;
            forceAppPointDistance = wheelObject.CurrentWheelCollider.forceAppPointDistance;
            spring = wheelObject.CurrentWheelCollider.suspensionSpring.spring;
            damper = wheelObject.CurrentWheelCollider.suspensionSpring.damper;
            targetPosition = wheelObject.CurrentWheelCollider.suspensionSpring.targetPosition;
        }
    }
}
```

Рисунок 3.15 – Лістинг класу WheelDTO

Даний клас складається з подібних, що і в WheelObject параметрів та має лиш один метод – конструктор, який дозволяє перетворити WheelObject в WheelDTO.

3.4.2 Зчитування та запис даних

Для парсингу DTO в JSON використовується утиліта JsonConvert, яка є частиною бібліотеки Newtonsoft.Json. JsonConvert надає основну функціональність для серіалізації та десеріалізації об'єктів в формат JSON. Можна виділити основні можливості даної утиліти:

– серіалізація: JsonConvert дозволяє перетворити об'єкти C# на рядки JSON. Це дозволяє зберігати дані в форматі, який може бути зручно поширений та збережений;

– десеріалізація: JsonConvert дозволяє перетворити рядки JSON в об'єкти C#. Це дозволяє отримати дані з формату JSON та працювати з ними в рамках C#-додатку.

В рамках класу SaveHandler використовуються саме ці дві його основні можливості. Їх застосування описане в методах Write та Read, які відповідають за запис та зчитування даних відповідно.

Розглянемо детальніше метод Write (рисунок 3.16). Він надає можливість запису моделі робототехнічного пристрою в формат JSON. Для цього виконуються наступні кроки:

– відкривається або створюється директорія;

– модель конвертується до DTO;

– відбувається запис до файлу та його збереження.

Метод Read в цілому виконує подібні дії але в іншому порядку (рисунок 3.17):

– відкривається директорія;

– зчитується JSON файл;

– JSON конвертується до DTO;

– на основі DTO будується модель.

Таким чином SaveHandler надає повний функціонал для зберігання та зчитування відповідних файлів.

Рисунок 3.16 – Блок-схема процесу збереження моделі робототехнічного пристрою

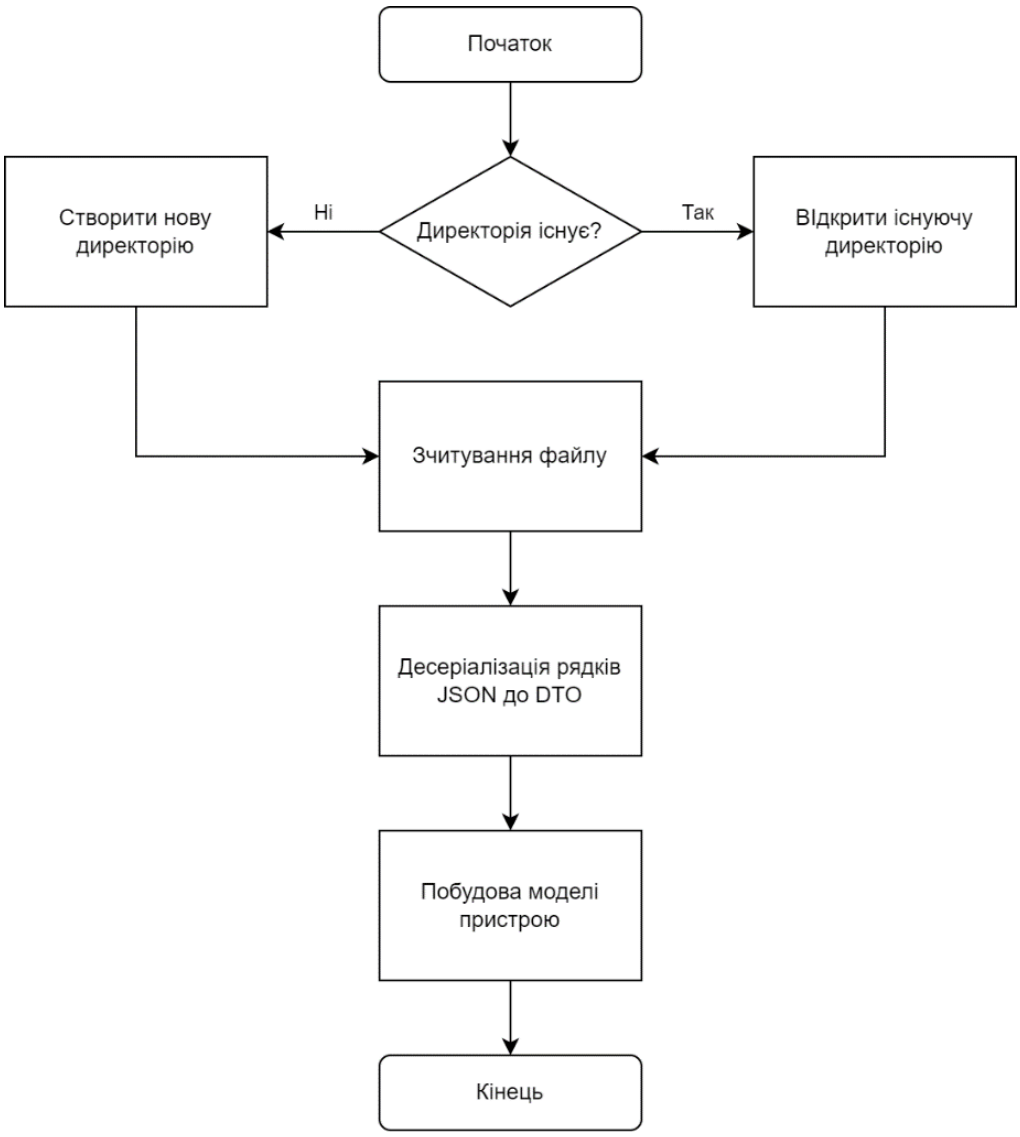

Рисунок 3.17 – Блок-схема процесу зчитування JSON

3.5 Користувацький інтерфейс

Для створення користувацького інтерфейсу було використано модуль Unity UI, який містить всі необхідні для цього інструменти. Unity має велику кількість компонентів, які допомагають забезпечити позитивний UX.

Базовим компонентом Unity UI є Canvas. Він є основою для розміщення та відображення елементів користувацького інтерфейсу. Canvas визначає область, на якій буде розміщуватися UI, та надає функціонал для її налаштування. Таким чином розробник має можливість створювати адаптивний динамічний інтерфейс.

Одним із основних елементів користувацького інтерфейсу в Unity є компонент Image, який використовується для відображення графічних зображень в інтерфейс. Властивості Image дозволяють налаштовувати розмір, кольори, прозорість, вирівнювання та інші параметри зображення. При цьому як і для будь-якого іншого компоненту Unity існує можливість роботи через API, що дозволяє розробнику взаємодіяти з UI через скрипти для створення динамічного інтерфейсу: зображення можна замінити, змінити його розміри і тому подібне.

Невід'ємною частиною UI є текст, в Unity розробник може взаємодіяти з даним механізмом за допомогою компоненту Text. Даний елемент використовується для відображення текстової інформації в інтерфейсі. Розробник може налаштовувати шрифт, розмір, колір, стиль та інші атрибути тексту. Текст може містити різні ефекти, такі як тінь або обведення.

Більш функціональним компонентом Unity UI можна назвати Button. Button - це інтерактивний елемент, який дозволяє користувачам взаємодіяти з інтерфейсом шляхом натискання на кнопку. Розробник може призначити функцію-обробник подій, яка виконується при натисканні на кнопку.

Також слід зазначити і компонент ScrollView, так як його функціонал, який дозволяє створювати область, яку можна перелистувати для відображення більшої кількості вмісту, який не поміщається на екран. Даний елемент значно спрощує створення інтерфейсу в цілому.

За допомогою функціоналу Unity можна розробляти різні ієрархічні структури об'єктів, створюючи при цьому абсолютно будь-який необхідний UI.

Компоненти модулю Unity UI широко використовуються у програмній реалізації віртуального конструктора. Розроблена велика кількість класів (рисунок 3.18), які взаємодіють з тим чи іншим компонентом, додаючи до вже існуючих можливостей нові. Дані класи дозволяють тісно зв'язати UI з різними механіками додатку. Також розроблені компоненти, які відповідають за візуальний аспект – це ефекти затемнення, закриття та відкриття додаткових вікон, анімації переміщення елементів користувацького інтерфейсу.

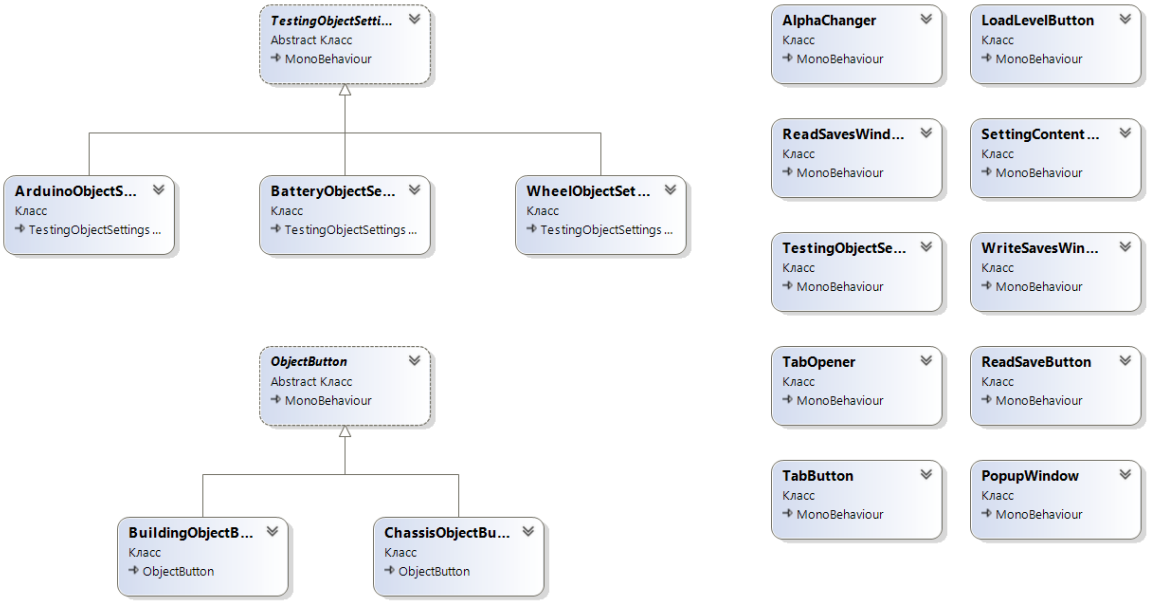

Рисунок 3.18 – Діаграма класів для роботи з UI

Висновки по розділу

У даному розділі було розглянуто основні особливості реалізації віртуального конструктора робототехнічних пристроїв з елементами тестування. В результаті розробки програмного застосунку створено систему, яка складається з декількох програмних модулів, кожен з яких має свою конкретну задачу, а саме:

– модуль конструювання: реалізовує функціонал створення моделей робототехнічних пристроїв. Даний модуль містить декілька базових класів: BuildingController, який керує процесом конструювання, шляхом контролю над основними операціями; BuildingObject – абстрактний клас, який, в цілому, описує кожен окремий компонент будівництва та є абстрактним, так як визначає шаблон для більш деталізованих класів;

– модуль тестування: надає інструменти для проведення експериментів над моделями робототехнічних пристроїв. Має дві основні задачі: оперування тестовими компонентами, тобто зміна їх параметрів в процесі тестування, а також симуляція фізики, що дозволяє відтворювати реалістичні ситуації. Налічує декілька основних класів: TestingController – об’єкт, що керує компонентами типу TestingObject, які в свою чергу описують базові параметри кожного окремого

компоненту; MovementController, який описує способи управління моделлю пристрою, враховуючи фізичні властивості;

– модуль збереження: дозволяє зберігати створені у додатку моделі пристроїв у форматі JSON та їх. Для серіалізації та десеріалізації JSON документів використовується утиліта JsonConvert бібліотеки Newtonsoft.Json, яка містить широкий спектр функціональних можливостей для роботи з даним форматом. Також даний модуль описує взаємодію з DTO.

Також у розділі приділена увага UI частині додатку. Розглянуто основні компоненти Unity, які реалізують необхідний для кожного інтерфейсу функціонал: Canvas, Image, Text, Button, ScrollView. Описано деякі програмні особливості при роботі з даними класами.

В цілому, реалізований програмний застосунок є повноцінною системою, яка відповідає всім поставленим до неї вимогам. Особливості реалізації дозволяють легко розширювати кожен окремий модуль, що надає можливість не тільки додавати новий функціонал у додаток, а й контент. Наразі віртуальний конструктор робототехнічних пристроїв дозволяє створювати лише подібні до автомобільних транспортних засобів моделі, але саме розширюваність кожного із модулів системи робить створення нових типів досить простим та швидким. При необхідності додаток завжди можна наповнити новими компонентами, швидко та легко створивши для них механіки, або використавши готові Unity елементи.

4 РОБОТА КОРИСТУВАЧА З ДОДАТКОМ

4.1 Інструкція користувача

Для початку необхідно запустити програмний застосунок використавши для цього наявні можливості ОС. Після чого відкривається головне вікно додатку (рисунок 4.1).

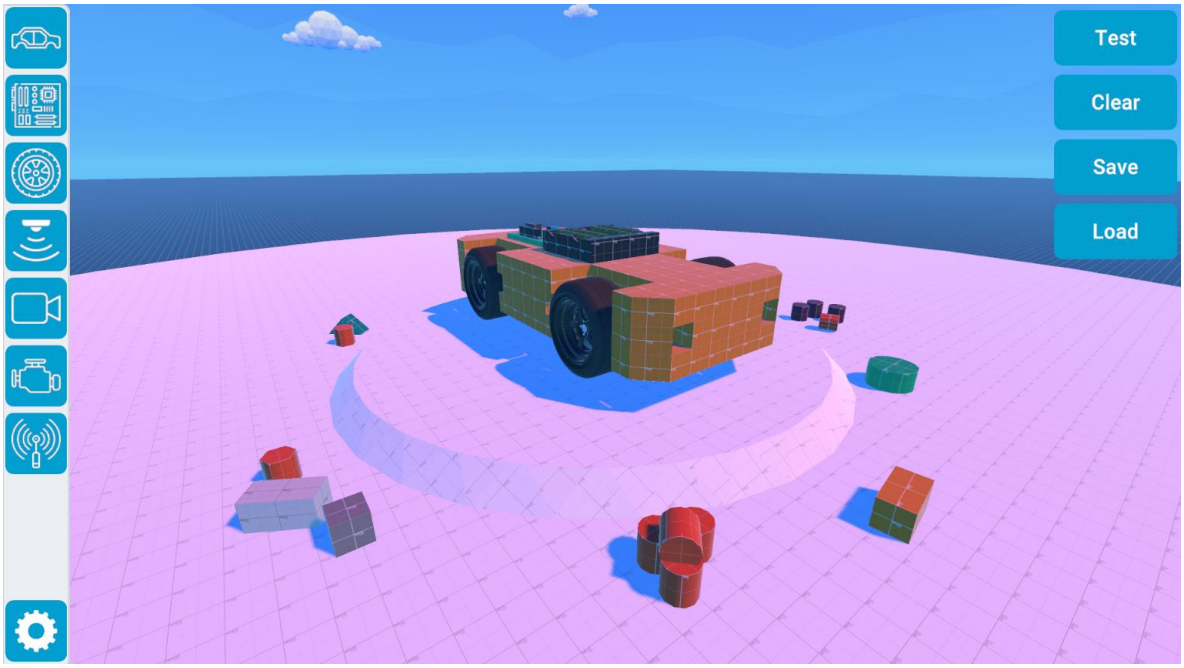

Рисунок 4.1 – Головний екран додатку

В даному вікні можна отримати доступ до всіх функціональних можливостей застосунку. Візуально дане вікно можна поділити на дві зони відповідальності: ліву та праву.

Спершу розглянемо ліву частину користувацького інтерфейсу. Тут знаходиться декілька клавіш, а саме: чотири, які відповідають певному типу елемента конструювання та налаштування. Якщо натиснути на одну з цих чотирьох кнопок, то відкриється список компонентів конструювання даного типу (рисунок 4.2). Якщо після цього обрати інший тип, то даний список оновиться та буде відображати вже нові елементи.

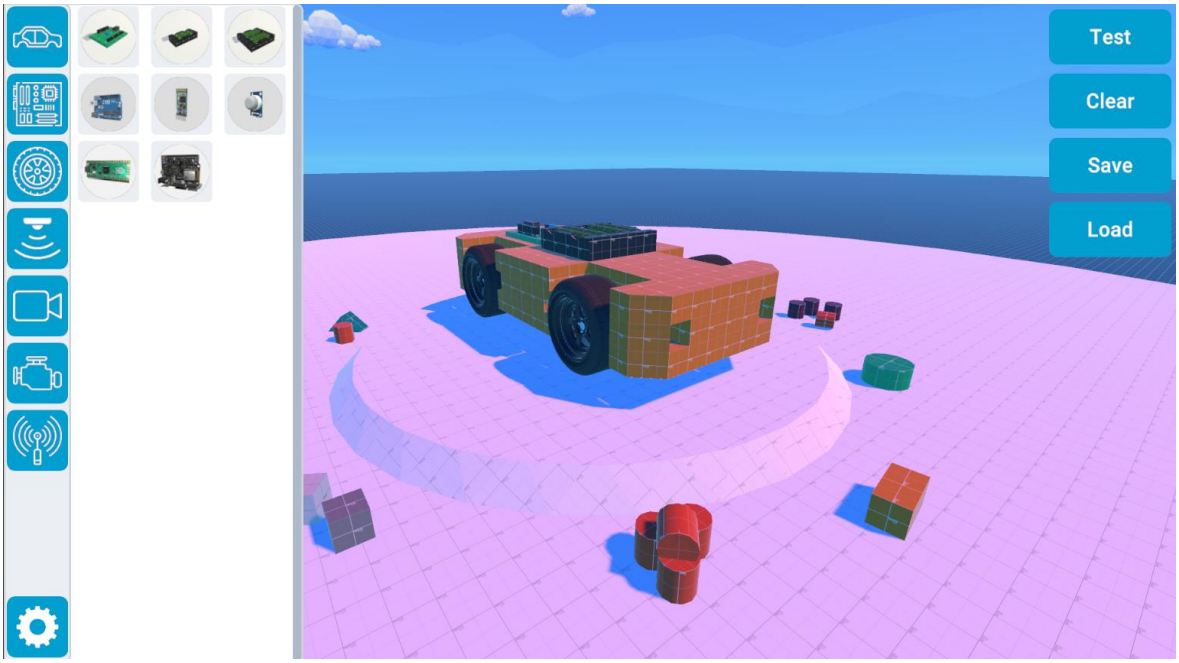

Рисунок 4.2 – Вікно зі списком компонентів конструювання певного типу

Натиснувши на кнопку налаштувань, дане вікно знову оновиться та буде представляти собою список графічних налаштувань програмного застосунку (рисунок 4.3).

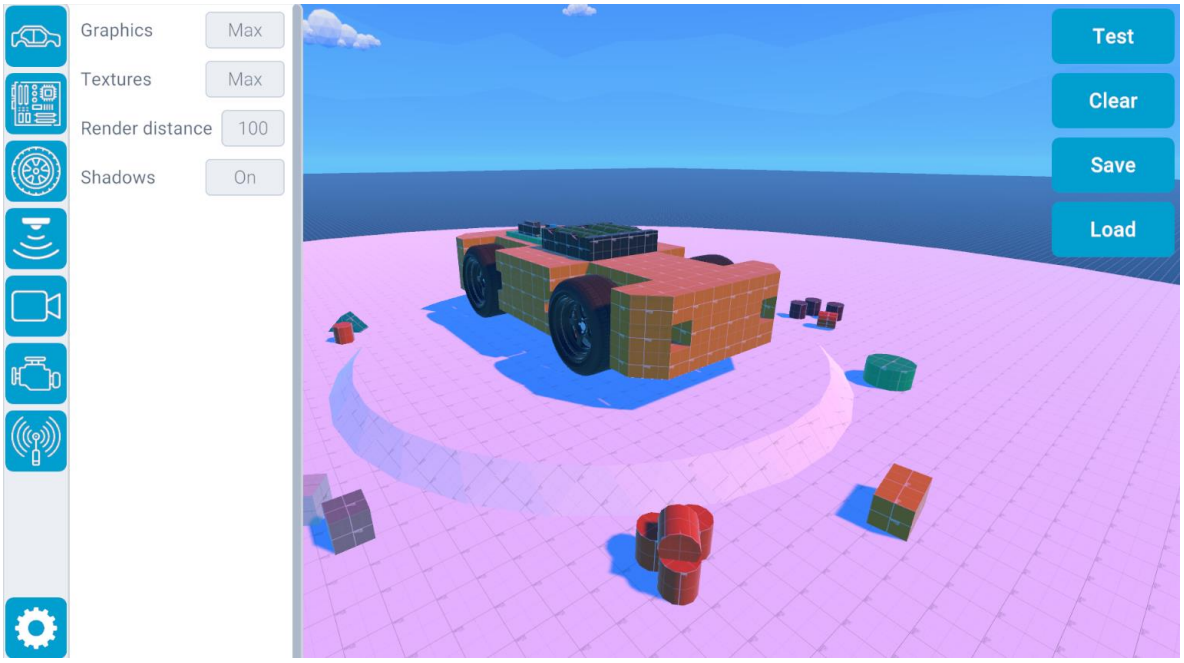

Рисунок 4.3 – Вікно графічних налаштувань програмного застосунку

Повернемося до списку компонентів конструювання, обравши необхідний тип. Тепер можна обрати один із елементів, просто натиснувши на нього, та почати процес створення моделі. Слід зауважити, що першим розташованим об’єктом має бути шасі, яке розміщується рівно по центру сцени автоматично. Після цього вже можна обрати будь-який інший тип та розмістити його в необхідному положенні. Для розміщення використовується маніпулятор ПК: об’єкт буде слідувати за курсором по сцені допоки не буде прикріплений або дія не буде скасована. Для того щоб підтвердити операцію з розташування необхідно натиснути на ліву клавішу маніпулятора, щоб скасувати дію – на праву. Також перед тим як закріпити об’єкт можна задати йому необхідний поворот, натиснвши для цього клавішу “R” на клавіатурі.

Слід також приділити увагу тому, як можна взаємодіяти зі сценою, крім розташування компонентів. Користувач може обертати камеру навколо центру сцени, натиснувши середню клавішу маніпулятора та тримаючи її, а також наближати або віддалювати зображення за допомогою колесика маніпулятора.

Після розміщення компоненту конструювання відкриється вікно налаштування його тестових параметрів (рисунок 4.4).

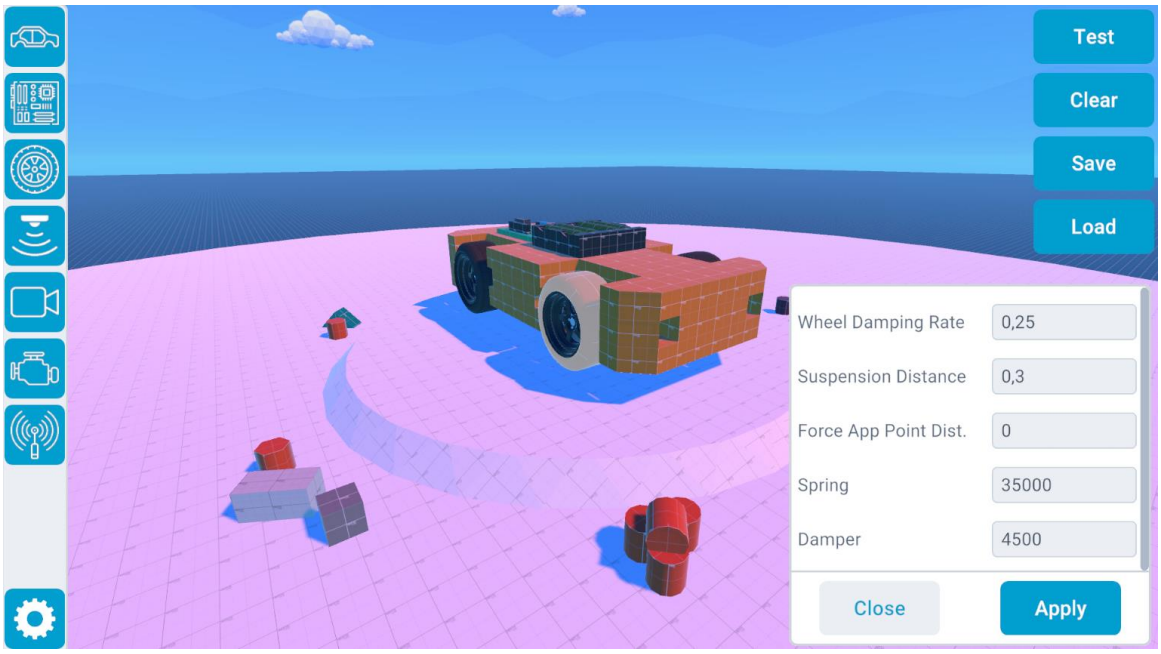

Рисунок 4.4 – Вікно налаштувань тестових параметрів компоненту на сцені конструювання

Також це вікно можна відкрити, натиснувши лівою клавішею маніпулятора на необхідний об’єкт на сцені. При цьому слід зауважити, що при наведені курсору на елементи на сцені, вони будуть підсвічуватись, у разі якщо ними можна маніпулювати.

Саме вікно є індивідуальним для кожного компоненту та може налічувати різні параметри відповідно до типу самого елементу. Загалом, воно може складатися з текстових полів, в які можна вписати необхідні значення, та чекбоксів, які дозволяють увімкнути або вимкнути той чи інший функціонал. Дане вікно має дві клавіши “Close” та “Apply”. Натиснувши на “Close” вікно закриється без збереження заданих параметрів. Для збереження налаштувань необхідно спершу натиснути “Apply” і вже після цього закрити вікно.

Повернемося до правої частини основного екрану. Тут знаходиться чотири клавіши, які непов’язані між собою та виконують різні задачі. Які в цілому можна розділити на три типи обов’язків: початок тестування, очищення сцени конструювання від моделі, робота зі збереженням.

Для початку, розглянемо клавішу “Load”. Вона відповідає за відкриття вікна зі збереженими моделями (рисунок 4.5).

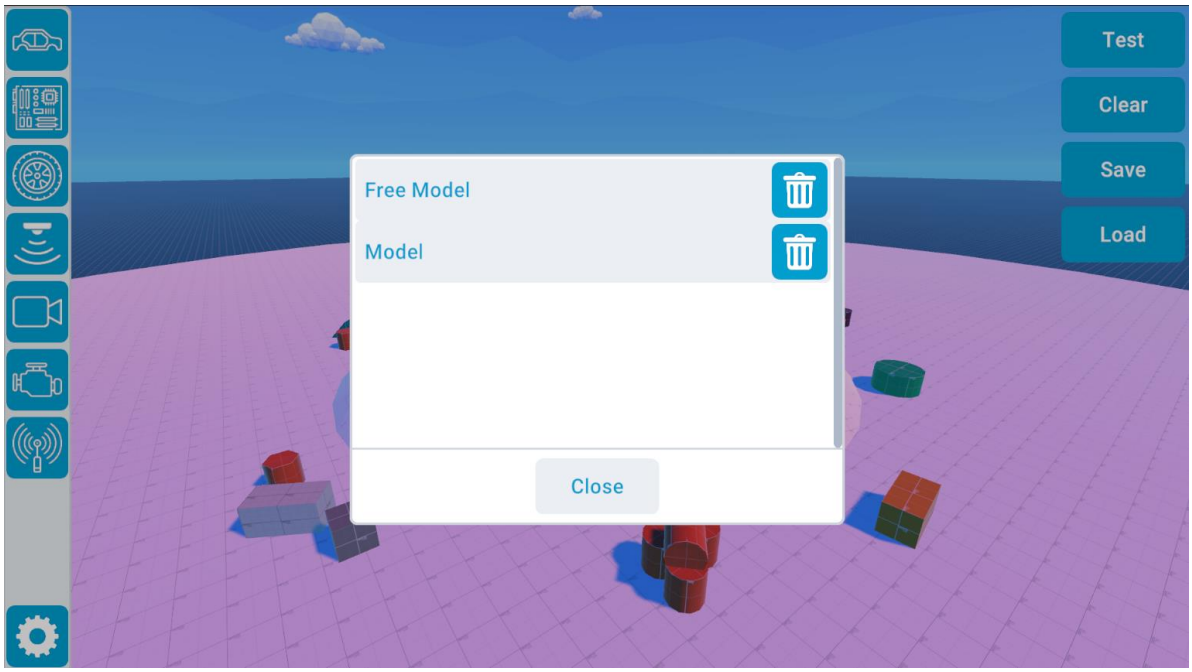

Рисунок 4.5 – Вікно зі збереженими моделями

У вікні присутній список збережених моделей, які знаходяться у папці “Saves” програмного додатку та мають формат JSON. Користувач може видалити необхідне збереження, натиснувши на діаграму з кошиком для сміття біля його назви. Якщо ж натиснути на назву збереження, то воно відкриється, при умові, що воно відповідає параметрам збереження системи. Важливо зауважити, що в результаті відкриття моделі, поточна модель на сцені буде видалена без збереження. Для виходу з даного вікна треба натиснути клавішу “Close”.

Кнопка “Save” відкриває вікно збереження моделі (рисунок 4.6), в якому можна записати поточну модель до формату JSON. Користувач може задати назву збереження, після чого натиснувши клавішу “Save” зберегти модель. JSON файл буде записаний в директорії програмного застосунку, а саме в папці “Saves”. Вікно закриється автоматично. Для того щоб закрити вікно власноруч необхідно натиснути на клавішу “Close”.

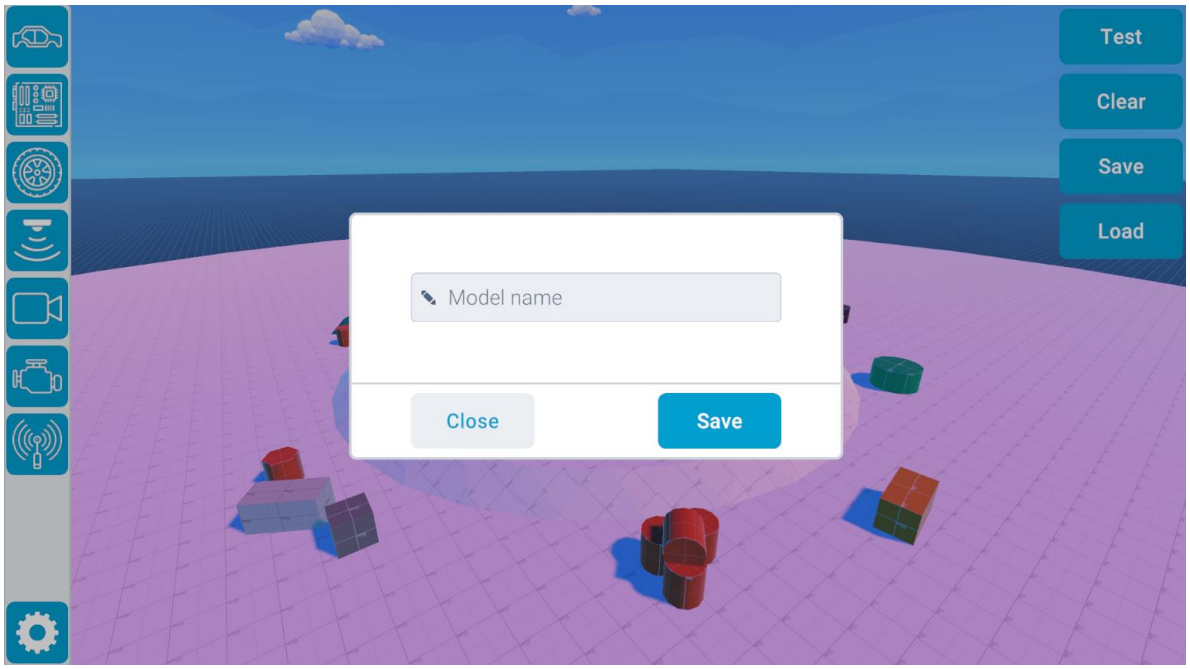

Рисунок 4.6 – Вікно збереження моделі

Наступна кнопка – “Clean” відповідає за очищення сцени конструювання від поточної моделі. Всі об’єкти які були розміщені будуть видалені безповоротно, якщо модель треба зберегти, то це потрібно зробити заздалегідь.

Кнопка “Test” відповідає за перехід на сцену тестування, при цьому ніяке додаткове вікно для вибору моделі не відкривається і переноситься поточна.

Після переходу на сцену тестування відкривається нове вікно (рисунок 4.7), яке дозволяє провести експерименти на робототехнічни пристроєм.

Рисунок 4.7 – Екран тестування

Управління рухом моделі на даній сцені відбувається за допомогою клавіатури: це можуть бути або стрілочки, кожна з яких відображає відповідний напрямок руху пристрою, або ж клавіши “W”, “S”, “A” та “D”, де “W” і “S” – рух вперед на назад відповідно, а “A” та “D” – вліво і вправо.

Користувацький інтерфейс на даній сцені має декілька основних задач:

– відображення основних поточних параметрів пристрою: заряд батареї; швидкість руху; енергія, яка використовується;

– налаштування компонентів тестування.

Вікно з поточнимим параметрами знаходиться зверху по центру екрану, та відображає відповідні параметри.

Налаштування компонентів тестування відбувається подібно до того, як і на сцені конструювання: необхідно натиснути на необхідний об’єкт, після чого

відкриється вже знайоме вікно (рисунок 4.8). Головною відмінністю в даному випадку є те, що деякі з параметрів можуть бути захищені від зміни і користувач має можливість лише моніторити їх поточні значення. Такими елементами можуть бути сенсори, або активні значення, зміна яких неможлива в процесі симуляції. Для того, щоб повернутися до сцени конструювання потрібно натиснути на клавішу з піктограмою виходу, яка знаходиться в правому верхньому куті екрану.

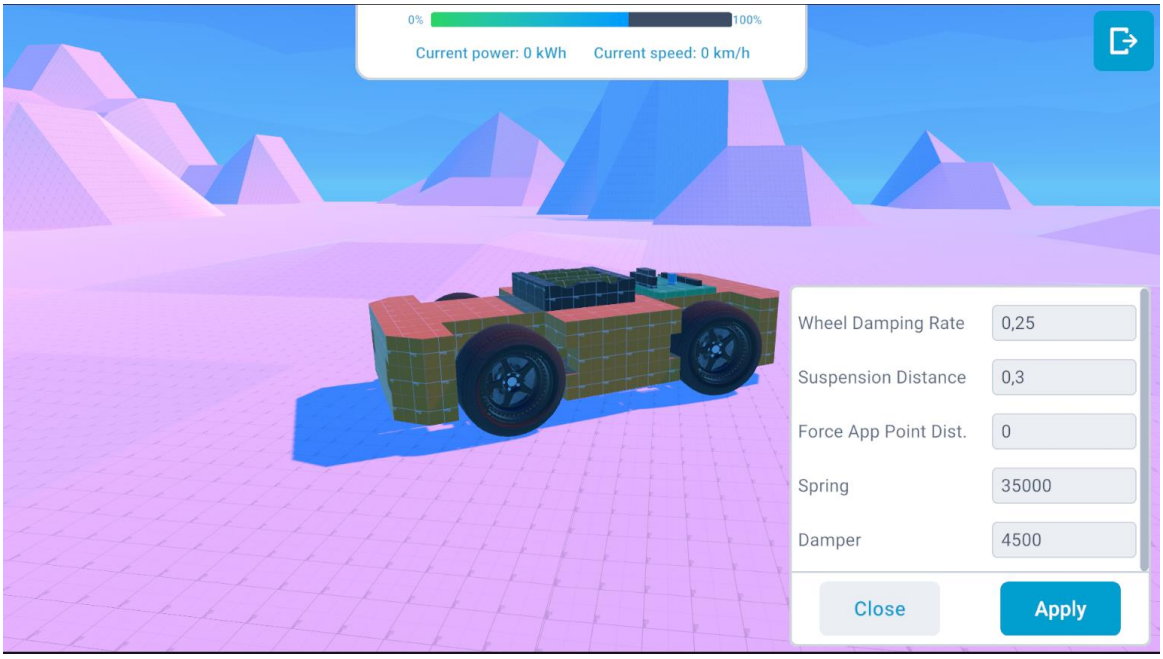

Рисунок 4.8 – Вікно налаштувань тестових параметрів компоненту на сцені тестування

4.2 Компоненти тестування

Програмний застосунок надає можливість використати сім основних типів компонентів тестування. В цей перелік входять найбільш необхідні для створення робототехнічного пристрою компоненти. Розглянемо кожен із цих типів окремо:

– мікроконтролер: є основним елементом в конструкції пристрою, так як виступає в ролі головного органу управління, який взаємодіє зі всіма іншими компонентами на шасі. Дозволяє керувати: двигунам, сенсорами, камерами і тому подібними елементами. Користувач має можливість налаштувати його під

використання з конкретною шасі, тобто налаштувати активність кожного окремого компоненту;

– плата зв’язку: дозволяє відправляти та приймати сигнали від оператора. Даний компонент надає можливість налаштувати керування пристроєм та його окремими елементами. Користувач зможе надавати вказівки мікроконтролеру, а також отримувати від нього інформацію про поточний стан системи: заряд батареї; швидкість руху; енергію, яка використовується на даний момент. За допомогою його компоненту можна, наприклад керувати двигунами пристрою. При чому можна налаштувати його таким чином, щоб в процесі тестування можна було переключити потік управління на будь-який необхідний елемент;

– шасі: є основою робототехнічного пристрою. Саме на базі шасі розташовуються всі інші компоненти. Даний елемент визначає тип самого пристрою: це може бути, як автомобільний транспортний засіб, так і літальний або наприклад – плаваючий. Шасі може визначати і загальні місця розташування окремих елементів, таких як двигуни чи батареї;

– колеса: використовуються в основному для автомобільних шасі. Даний компонент є також і елементом підвіски, тому його налаштування дозволяють змінити окремі значення в даній системі: жорсткість пружин, висоту підвіски і тому подібне;

– датчики: функціонують відповідно до їх призначення та дозволяють налаштувати лиш свою точність та чутливість до змін. При цьому користувач має можливість переглянути поточні показники в будь-який момент;

– камера: дозволяє отримати зображення та вивести його в окреме вікно. Даний компонент тісно взаємодіє з мікроконтролером для обробки інформації;

– двигуни: даний компонент має декілька характеристик: потужність та споживання енергії. Можна налаштовувати його режими роботи – економний або продуктивний. З його допомогою можна створювати рухому елементи та керувати ними. В автомобільному шасі зазвичай використовується лише разом з колесами.

Висновки по розділу

В даному розділі було розглянуто взаємодію з користувацьким інтерфейсом програмного застосунку. Наведені інструкції того як слід виконувати кожну окремо дію. Загалом, було описано способи роботи з інтерфейсом та механіками додатку на двох його сценах: конструювання та тестування.

На сцені конструювання розглянуто можливості для створення моделі робототехнічного пристрою, а саме: вибір компоненту конструювання зі списку існуючих, розміщення його на сцені, поворот навколо своєї осі. Також наведено інструкцію зі збереження моделі, її зчитування, а також взаємодію з уже існуючими файлами. Описано особливості при роботі з даною системою. Показано як потрібно налаштовувати компоненти тестування та як правильно застосувати зміни.

Для сцени тестування наведено перелік клавіш, за допомогою яких можна керувати моделлю пристрою. Показано як можна переглядати поточний стан системи в цілому, а також її окремих компонентів.

Також приділена увага наявним у системі типам компонентів тестування, а саме: двигун, мікроконтролер, плата зв’язку, шасі, колесо, камера, датчик. Розглянуто особливості використання та налаштування кожного з них.

ВИСНОВКИ

У даній роботі було розглянуто основні етапи проектування та тестування робототехнічних пристроїв, виділені основні недоліки класичного підходу до розробки. В результаті чого було запропоновано створення віртуального конструктора робототехнічних пристроїв, який дозволить автоматизувати даний процес. Здійснено аналіз існуючих рішень на ринку, їх переваг та недоліків. На основі цієї інформації було сформовано вимоги та задача до проекту. Проаналізовано існуючі технології, які можуть бути використані для виконання поставлених задач. Після чого була спроектована архітектура програмної системи, що, загалом, складається з трьох модулів: конструювання, тестування та збереження. Реалізовано програмне рішення.

Розроблений в даній дипломній роботі програмний застосунок дозволяє розробникам робототехнічних пристроїв значно полегшити кожен з етапів його конструювання та тестування, зменшити матеріальні та часові витрати. Такий підхід до розробки дозволяє більш широкому колу людей прийняти в цьому участь, що допомагає популяризувати робототехніку в цілому.

Програмна реалізація віртуального конструктора дозволяє з легкістю розширювати його функціональні можливості, додавати нові моделі компонентів та механіки, що є вагомою перевагою.

В цілому, розроблений віртуальний конструктор робототехнічних пристроїв відповідає всім поставленим до нього вимогам. Таким чином, можна стверджувати, що мета дипломної роботи – автоматизація процесу конструювання та тестування робототехнічних пристроїв, досягнута в повній мірі.

В подальшому, можна розширити список існуючих у системі компонентів, додавши нові типи шасі – це може бути, як автомобільний транспортний засіб, так і літальний або наприклад – плаваючий, що дозволить значно розширити функціональні можливості тестування робототехнічних пристроїв, датчиків, двигунів і тому подібних елементів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Робототехніка: від глини до нано-матеріалів [Електронний ресурс] – Режим доступу до ресурсу: https://phm.cuspu.edu.ua/nauka/naukovo-populiarni-publikatsii/2130-robototekhnika-vid-hlyny-do-nano-materialiv.html
2. Застосування робототехніки в сучасному та майбутньому (2021) [Електронний ресурс] – Режим доступу до ресурсу: https://uk.warbletoncouncil.org/aplicaciones-robots-9435
3. Robot Design Process [Електронний ресурс] – Режим доступу до ресурсу:https://penfieldrobotics.com/robot-design-process/
4. Bruno Siciliano, Lorenzo Sciavicco. Robotics: Modelling, Planning and Control, 2008.
5. What Is Tinkercad? – Simply Explained (2022) [Електронний ресурс] – Режим доступу до ресурсу: https://all3dp.com/2/what-is-tinkercad-simply-explained/
6. Getting started with Robot Virtual Worlds [Електронний ресурс] – Режим доступу до ресурсу: https://www.robotc.net/getting-started-lego-rvw.html
7. RoboDK [Електронний ресурс] – Режим доступу до ресурсу: https://robodk.com/
8. Why Choose Unity 3D for Your Next Game Development Project? [Електронний ресурс] – Режим доступу до ресурсу: https://www.mindinventory.com/blog/unity-3d-game-development/
9. Unity User Manual [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/
10. Physics in Unity 5.0 [Електронний ресурс] – Режим доступу до ресурсу: https://docs.unity3d.com/Manual/UpgradeGuide5-Physics.html
11. NVIDIA PhysX SDK Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/3.3.4/Index.html
12. Joseph Albahari, Ben Albahari. C# 8.0 in a Nutshell: The Definitive Reference, 2020.

13.How Unity Supports Cross Platform Feature [Електронний ресурс] – Режим доступу до ресурсу: https://niraj-vishwakarma.medium.com/how-unity-supports-cross-platform-feature-ae722321cfa
14.Jeffrey Richter. CLR via C#, 2013.
15.Ray Rischpater. JavaScript JSON Cookbook, 2015.
16.Joseph S. Valacich, Joey F. George, Jeffrey A. Hoffer. Modern Systems Analysis and Design, 2015.
17.Layna Fischer, Frank Chong. BPMN 2.0 Handbook, 2010.
18.Joe Hocking. Unity in Action: Multiplatform Game Development in C#, 2015.
19.Separating Axis Theorem [Електронний ресурс] – Режим доступу до ресурсу: http://programmerart.weebly.com/separating-axis-theorem.html