КВАЛІФІКАЦІЙНА РОБОТА

Група ІПЗс-20
Штогрин В.Я

2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Штогрин Володимир Ярославович

УДК 004.4

Розробка агентно-орієнтованої системи для підвищення інтерактивності в комп'ютерних іграх

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації бакалавр

Нормоконтроль	Студент
_____Стисло О.В.	_____Штогрин В.Я.
(підпис, дата, розшифрування підпису)	(підпис, дата, розшифрування підпису)
Допускається до захисту	Керівник роботи
Завідувач кафедри	
	к.т.н., проф. каф. ІТ
_____к.т.н., доц. Ващишак С.П.	_____ Пашкевич О.П.
(підпис, дата, розшифрування підпису)	(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

«____» _____ 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Штогрин Володимиру Ярославовичу

(прізвище, ім’я, по батькові)

1. Тема кваліфікаційної роботи:

Розробка агентно-орієнтованої системи для підвищення інтерактивності в комп’ютерних іграх

керівник роботи:

Пашкевич Олег Петрович, к.т.н., проф. каф. ІТ

затверджена наказом вищого навчального закладу від « 12 » березня 2024 року № 19/1

2. Строк подання студентом роботи 05.06.2024

3. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)

1. Структура комп’ютерних ігор та агенти
2. Агентно-орієнтована методологія побудови комп’ютерних ігор для підвищення їх інтерактивності
3. Розробка діаграма розгортання та процесів компонування гри
4. Вибір методів оцінки методології підвищення інтерактивності гри

4. Дата видачі завдання 14.03.2024

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Структура комп'ютерних ігор та використання агентної технології	20.03.2024	Виконано
2.	Застосування агентно-орієнтованої методології побудови ігор	10.04.2024	Виконано
3.	Розробка діаграма розгортання та процесів компонування гри	30.04.2024	Виконано
4.	Вибір методів оцінки побудованої методології підвищення інтерактивності гри	12.04.2024	Виконано
5.	Формування висновків	01.05.2024	Виконано
6.	Оформлення пояснювальної записки	22.05.2024	Виконано
7.	Оформлення графічного матеріалу та підготовка до захисту роботи	05.06.2024	Виконано

Студент _____ Штогрин В.Я

(підпис) (прізвище та ініціали)

Керівник роботи _____ Пашкевич О.П

(підпис) (прізвище та ініціали)

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Сторінка	Опис графічного матеріалу	Сторінка	Опис графічного матеріалу
17	Класифікація ігрових додатків	60	Ієрархія цілей (спринт 3)
22	Зв'язок між характеристиками популярного агента та популярними іграми	61	Діаграма головної послідовності
26	Інтерфейс 3DUnity	61	Діаграма головної послідовності (спринт 1)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці агентно-орієнтованої системи для підвищення інтерактивності в комп’ютерних іграх шляхом використання моделей та засобів агентно-орієнтованого підходу, що дозволяє створювати продукти, які відзначаються високою ступенем інтелектуальної поведінки та динаміки, що забезпечує більш захоплюючий геймплей та задоволення для гравців.

В першому розділі здійснено аналіз предметної області використання штучного інтелекту для розробки комп'ютерних ігор. Наведено основні визначення та класифікацію ігор, описано використання штучного інтелекту в іграх, описи ігрового двигунів. Наведені аспекти використання агентного підходу в розробці комп’ютерних ігор.

В другому розділі описана агентно-орієнтована методологія побудови комп’ютерних ігор, здійснено огляд методологій розробки програмного забезпечення в ігрових додатках. Реалізована структура методології розробки ігор (GDMF), описано методику та оцінку тривалості розробки гри, наведено опис життєвого циклу методології створення комп’ютерної гри.

В третьому розділі виконано вибір методів оцінки побудованої агентно-орієнтованої методології для підвищення інтерактивності в комп’ютерних іграх. Здійснено оцінку методики, проведено аналіз фаз розробки комп’ютерної гри та розробку діаграми розгортання та процесів компонування гри.

КЛЮЧОВІ СЛОВА: КОМП’ЮТЕРНА ГРА, ШТУЧНИЙ ІНГТЕЛЕКТ, АГЕНТ, ПЛАТФОРМА, ПРРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ЖИТЄВИЙ ЦИКЛ, ТЕСТУВАННЯ, ДІАГРАМА РОЗГОРТАННЯ, АГЕНТНО-ОРІЄНТОВАНИЙ ПІДХІД

SUMMARY

The qualification work is devoted to the development of an agent-oriented system for increasing interactivity in computer games by using models and means of an agent-oriented approach, which allows creating products that are characterized by a high degree of intelligent behavior and dynamics, which provides more exciting gameplay and pleasure for players.

In the first section, an analysis of the subject area of the use of artificial intelligence for the development of computer games is carried out. The main definitions and classification of games are given, the use of artificial intelligence in games, descriptions of game engines are described. Aspects of using the agent approach in the development of computer games are given.

In the second chapter, the agent-oriented methodology of building computer games is described, and the methodologies of software development in game applications are reviewed. The framework of the game development methodology (GDMF) is implemented, the methodology and estimation of the game development duration are described, and the life cycle of the computer game creation methodology is described.

In the third section, the selection of evaluation methods of the constructed agent-oriented methodology for increasing interactivity in computer games is made. The evaluation of the methodology was carried out, the analysis of the development phases of the computer game and the development of the deployment diagram and the game layout processes were carried out.

KEY WORDS: COMPUTER GAME, ARTIFICIAL INTELLIGENCE, AGENT, PLATFORM, SOFTWARE, LIFE CYCLE, TESTING, DEPLOYMENT DIAGRAM, AGENT-ORIENTED APPROACH

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ABC – Agent Based Computing

ABLE – Agent Building and Learning Environment

ACL – Agent Communication Language

AEF – AOAB Evaluation Framework

AI – Articial Intelligence

AOAB – Agent Oriented Agile Based Game Development Methodology

AOD – Agent Oriented Development

AOM – Agent-Oriented Methodology

AOSE – Agent-Oriented Software Engineering

A-UML – Agent-UML

AV – Agent/Role View

BDI – Belief Desire Intention

BGE – Blender Game Engine

DV – Domain View

CIS – Computing Science

DV – Domain view

FIPA – Foundation for Intelligent Physical Agents

FPS – First Person Shooter

FSM – Finite State Machines

FuSM – Fuzzy State Machine

GA – Genetic Algorithm

GDD – Game Development Document

GDMF – Game Development Methodology Framework

GEHF – Game Evaluation Heuristics Framework

GOAP – Goal Oriented Action Plan

GTV – Goal/Task View

HEP – Heuristics for Evaluating Playability

HCI – Humen Computer Interaction

IE – Interactive Entertainment

IS – Information System

IV – Interaction View

JADE – Java Agent Development

JATLite – Java Agent Template

ВСТУП

Актуальність теми. Сьогодні неможливо уявити світ без комп'ютерних технологій. Вони використовуються в майже всіх галузях людської діяльності. Проте головною сферою використання залишаються освіта та розваги. Людина не може прожити життя без відпочинку, який деякі вбачають саме в комп'ютерних іграх. З моменту появи перших відеоігор минуло 50 років, але індустрію комп'ютерних ігор досі вважають молодою і незрілою. Перша комп'ютерна гра розроблялась за допомогою апаратних засобів, проте стрімкий розвиток мікропроцесорів змінив хід подій. У наш час відеоігри створюються як для універсальних ПК, так і для ігрових консолей. Розробка комп'ютерних ігор – це прибуткова галузь промисловості, що охоплює широкий спектр населення і розвивається щодня, створюючи нові жанри, використовуючи покращені технології. Тим не менш, ринок відеоігор не є стабільним, тому питання вибору грального рушія є актуальною темою для дослідження.

При розробці сучасних комп'ютерних ігор найбільш продуктивною є технологія грального рушіїв, яка покликана спростити процес розробки за рахунок уніфікації та систематизації внутрішньої структури гри. Ігровий рушій є комплексом програмних компонентів, які відповідають за реалізацію основних функціональних можливостей гри: візуалізацію ігрової сцени (рушій рендерингу), симуляцію фізичних законів реального світу у віртуальному (фізичний рушій), відтворення звуку (звуковий рушій), створення ілюзії інтелекту в поведінці ігрових персонажів (ігровий штучний інтелект), анімацію.

Ігровий рушій – це програмне забезпечення для розробки комп'ютерних ігор. Поняття «грального рушія» з'явилось в середині 1990-х років разом з шутерами від першої особи «first person shooter» (FPS), в побудові яких були явно розділені дві частини: програмне забезпечення компонентів (наприклад, аудіосистема) та художні засоби разом з частиною гри, у якій пояснювались правила та механізм здобуття гравцем ігрового досвіду. Різниця двох частин

явно проявилась через те, що автори почали ліцензувати свої продукти, після чого створювали нові ігри з додаванням нових інструменти (текстури, героїв, зброю, правила тощо) з мінімальними змінами.

Актуальність даної роботи полягає в тому, що розробка комп'ютерних ігор стає все складнішою та вимагає використання новітніх технологій, зокрема агентно-орієнтованого підходу. Вивчення цього підходу та його ефективного застосування в розробці ігор є актуальним завданням у сучасному інформаційному суспільстві. Результати роботи можуть бути корисні розробникам ігор, дослідникам у галузі інформаційних технологій та всім зацікавленим у вдосконаленні якості та інтелектуальної складності комп'ютерних ігор. Ріст популярності ігор - з кожним роком ігри стають все більш популярними та широко розповсюдженими як серед різних вікових груп, так і на різних платформах. Зростання ігрової індустрії створює потребу у вдосконаленні технологій розробки, агентно-орієнтований підхід вказує на новий напрямок у цьому відношенні. Розширення можливостей штучного інтелекту - застосування інтелектуальних агентів у розробці ігор відкриває нові можливості для створення глибших та більш інтерактивних ігор з високим рівнем адаптації до дій гравців. Підвищення складності ігор - сучасні гравці очікують від ігор не тільки захоплюючого сценарію та графіки, але й високого рівня інтелектуальної складності, де агентно-орієнтований підхід може виграти важливу роль.

Необхідність оптимізації процесу розробки - зростання складності ігор вимагає ефективних методологій розробки. Агентно-орієнтована методологія може стати ефективним інструментом для оптимізації робочого процесу та поліпшення кінцевого продукту.

Порівняння роботи з відомими розв’язаннями проблеми. Визначається конкретними характеристиками та підходами із застосуванням агентів, які використовуються у сучасній ігровій індустрії. Інтелектуальна поведінка агентів - даний дослідницький підхід зосереджений на розвитку інтелектуальної поведінки агентів у розробці ігор, що може призвести до більш динамічного та

цікавого геймплею. Відомі розв'язання - Деякі ігри вже використовують агентів, але підхід до їхньої інтелектуальної поведінки може бути обмеженим або не таким розширеним, як у вказаному дослідженні.

Методологія розробки - даний дослідницький підхід пропонує агентно-орієнтовану методологію (GDMF), яка може бути спрямована на оптимізацію процесу розробки та покращення командної співпраці. Відомі розв'язання: Зазвичай використовують традиційні методології, інколи не належним чином адаптовані для агентно-орієнтованої розробки. Проведено експеримент з дизайну академічної гри та оцінка його результатів, що вказує на практичну застосовність запропонованого підходу. Відомі розв'язання: Експерименти можуть бути менш широкими або зорієнтованими на конкретні технічні аспекти без глибшого аналізу організації методологій.

Аналіз життєвого циклу: проведено аналіз життєвого циклу методології, що дозволяє визначити етапи та оптимізувати їх. Відомі розв'язання: Життєві цикли можуть бути менше орієнтовані на агентно-орієнтований підхід, що може призводити до меншої ефективності.

Даний дослідницький підхід вирізняється розширеним фокусом на розробці інтелектуальної поведінки агентів та використанням агентно-орієнтованої методології з подальшим аналізом та експериментами. Його практичне значення полягає у можливості оптимізації розробки ігор та покращенні їхньої якості з використанням агентно-орієнтованого підходу.

Мета і задачі дослідження. **Метою** роботи визначення та аналіз використання моделей та засобів агентно-орієнтованого підходу при розробці комп'ютерних ігор, а також вивчення їхнього впливу на якість та інтерактивність розроблених ігрових продуктів.

Досягнення мети включало розв’язання таких **задач**:

– розгляд визначення та класифікації ігор, включаючи їхні структури та агентів;

– аналіз ролі штучного інтелекту та ігрових двигунів, зокрема 3D Unity, у розробці комп'ютерних ігор;

– вивчення зв'язку агентів та агентських платформ, а також роль мови моделювання агентів (AML) та конкретного інструменту - AgentTool 3;

– дослідження інтелектуального агента в контексті комп'ютерних ігор;

– огляд різних методологій розробки програмного забезпечення в іграх, включаючи агентно-орієнтовану методологію;

– визначення структури методології розробки ігор (GDMF) та аналіз її функцій, тривалості розробки та командної співпраці;

– опис життєвого циклу методології та критичний аналіз її ефективності.

Об’єктом дослідження є комп'ютерні ігри та їх розробка.

Предметом дослідження є моделі та методи застосування агентно-орієнтованого підходу при розробці комп'ютерних ігор з метою підвищення їх інтерактивності.

Методи дослідження. Для досягнення поставленої мети будуть використанні такі методи: літературний аналіз наукових публікацій та джерел, пов'язаних із розробкою комп'ютерних ігор та агентно-орієнтованим підходом; аналіз конкретних прикладів використання агентів у відомих іграх та їхніх ігрових двигунах; вивчення методологій розробки програмного забезпечення в іграх та їхнього впливу на ефективність процесу розробки. Використання емпіричних методів для проведення експерименту з дизайну академічної гри та оцінки його результатів.

Результати роботи. Результатом виконання кваліфікаційної роботи є розширення розуміння ролі агентів у структурі ігор - дослідження надає новий погляд на використання агентно-орієнтованого підходу у структурі комп'ютерних ігор, що може сприяти подальшому розвитку та вдосконаленню цього підходу. Аналіз методології GDMF - вперше проведений докладний аналіз агентно-орієнтованої методології GDMF, що може послужити основою для подальших досліджень та модифікацій у цьому напрямку. Проведено експеримент з дизайну академічної гри з використанням агентно-орієнтованої методології, що може стати важливим додатком до практики розробки ігор.

Отже, дана робота не тільки вносить важливий науковий внесок у галузь, але й надає практичні рекомендації для розробників ігор, що можуть бути використані для покращення процесу розробки та підвищення інтерактивності створених ігор. Розробки ігор можуть використати отримані результати для оптимізації процесу розробки, поліпшення якості гри та підвищення задоволення гравців. Підвищення рівня інтелектуальної складності гри - реалізація агентно-орієнтованого підходу може сприяти створенню ігор з вищим рівнем інтелектуальної складності та адаптацією до дій гравців.

Структура. Розділи – 3. Загальний обсяг основної частини - 63 сторінка. Список використаних джерел – 47.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ РОЗРОБКИ КОМП'ЮТЕРНИХ ІГОР

1.1 Визначення та класифікація ігор

Ми визначаємо гру загалом як «структуровану гру, яка зазвичай виконується для задоволення, а іноді використовується як навчальний інструмент». Багато інших авторів визначають гру як діяльність, яка повинна мати такі характеристики: весела, відокремлена, невизначена, непродуктивна, регламентована правилами, фіктивна» [14].

«Гра — це система, в якій гравці беруть участь у штучному конфлікті, визначеному правилами, що призводить до кількісно визначеного результату» або «Гра — це діяльність одного або кількох незалежних осіб, які приймають рішення, для досягти своїх цілей у певному обмеженому контексті» [16].

Однак визначення Кроуфорда є найбільш підходящим визначенням гри, оскільки він згадує про зв'язок між грою та її агентами. Згідно з його визначенням, гра – це «інтерактивна, цілеспрямована діяльність, проти якої грають активні агенти, в якій гравці, включаючи активних агентів і NPC, можуть заважати один одному» [45]. Ми помітили, що ігри можуть бути визначені різними способами. Крім того, ігри також можна класифікувати на багато типів. Зараз доступна величезна різноманітність ігор. Розробка єдиної класифікації гри є складним процесом, оскільки існує багато різних категорій класифікації, як показано на рисунку 1.1.

Рішення щодо вибору класифікації базуються на різних критеріях, а саме:

– залежить від ігрового середовища, як-от ігри на свіжому повітрі (ігрова діяльність) та ігри в приміщенні, які включають настільну гру, наприклад настільну гру та карткові ігри та/або одну з цих трьох підкатегорій;

– ігри для вечірок, які вимагають фізичних дій або трюків;

– ігри-вікторини, які вимагають запитань і відповідей;

– ігри на вміння та дії, які вимагають від гравців кидати, котитися, балансувати та будувати;

– залежить від аудиторії, на яку розрахована гра; тобто дитячі ігри, сімейні ігри або ігри для дорослих;

– залежить від кількості гравців, наприклад один гравець, наприклад Puzzle, або два гравці, або багато гравців.

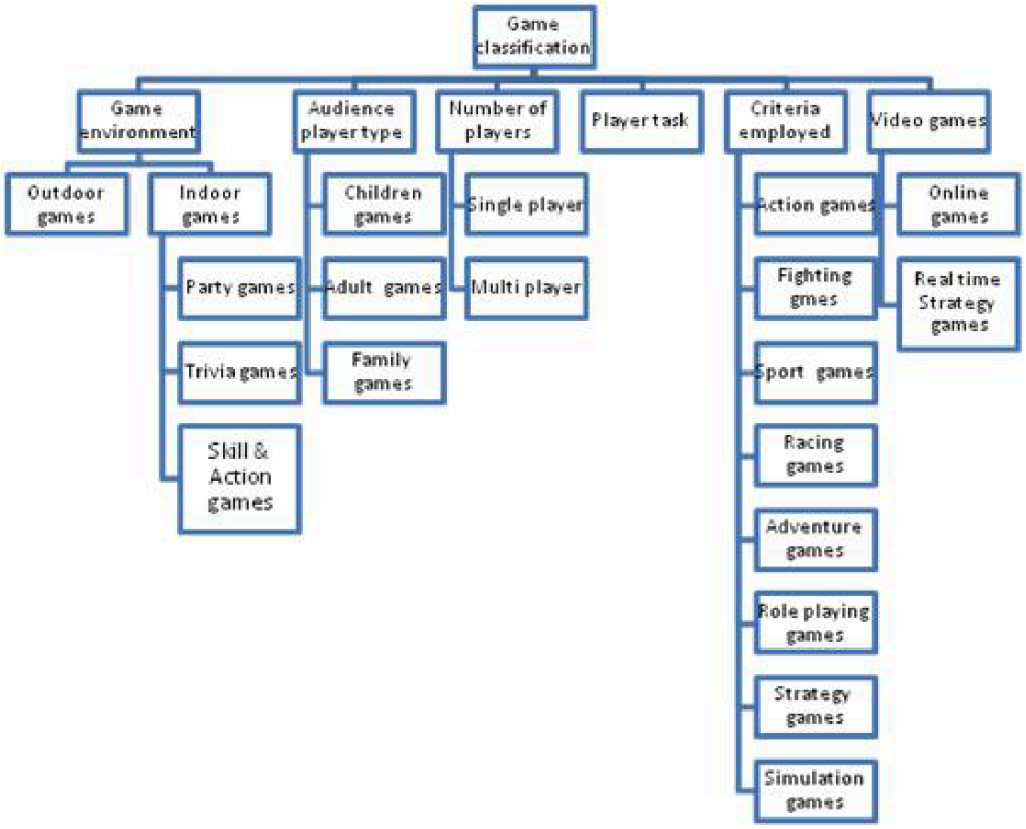


Рисунок 1.1 – Класифікація ігрових додатків

Екшн-ігри: ігри, у які проходять серії рівнів (у віртуальному світі) з різноманітними ворогами. У більшості екшенів гравець є шутером від першої особи (FPS), як у таких іграх, як Street fighter і Mortal Kombat.

Спортивні ігри: ігри, які зображують різні види спорту, створюючи віртуальні представлення спортивної діяльності. Деякі ігри-симулятори імітують спортивне середовище, наприклад Astro Race.

Пригодницькі ігри: сюжетні ігри. Пригодницькі ігри включають такі дії, як дослідження, збір інформації та вирішення проблем.. Головоломки також є прикладом пригодницьких ігор. Вирішуючи головоломки, можна відкрити доступ до нових областей ігрового світу.

Дизайн орієнтований не тільки на персонажів, а й на ресурси (будівля оборонних, наступальних підрозділів і військ) і необхідність організації та управління боями. Star Craft II: Wings of Liberty — популярна стратегічна гра.

Рольові ігри (RPG): ці ігри беруть на себе роль персонажів, які діють у вигаданому місці. В основну рольову гру грає невелика кількість учасників, переважно віч-на-віч. Рольові ігри зазвичай вимагають від гравця виконання квесту. Dragon Quest і Pokemon є прикладами RPG.

Стратегічні ігри в реальному часі (RTS): це гра, в якій гравці зосереджуються на логістиці та виробництві ресурсів, а також керують боями та війною. Ці ігри зазвичай включають швидке прийняття рішень і швидкі рефлекси. Гравцям потрібно одночасно командувати арміями та збирати ресурси. Це може виявитися надзвичайно складним завданням, і більшість ігор RTS розробили багато ігрових інструментів, щоб допомогти гравцям впоратися з цим завданням. Типовими прикладами ігор у реальному часі є Age of Empires, Age of Empires 3 та Empire Earth.

1.2 Використання штучного інтелекту в іграх. Опис ігрового двигунів

В іграх використовується багато методів і концепцій ШІ. Найпопулярніші методи включають кінцевий автомат, агент нечіткої логіки сценаріїв тощо, а деякі розробники ігор використовують дерева рішень, нейронні мережі та генетичні алгоритми. Нам потрібно представити найважливіші критерії ШІ та пояснити, як вони пов’язані з вимогами гри. Загалом, ігровий ШІ зосереджується на створенні видимості інтелекту багатьма способами. Таблиця 1.1 показує основні критерії ШІ для ігор і пояснює, як вони пов’язані з технікою ШІ [24].

Таблиця 1.1

Критерії вимог ШІ

AI requirement	AI Criteria
Manage game world	FSM, FuSM, Script
Optimal solution	GA
Path finding	A*, Streeing
Decision making	NN, Agent, Fuzzy logic
Learning	NN, GA
Developing game strategy	GA

У цьому розділі розглядається використання ШІ в ігрових додатках:

1. Кінцевий автомат (FSM) в іграх.

FSM — це набір кінцевої кількості станів, кожен набір для входу, виходу та функції переходу станів. FSM в іграх розділяє поведінку ігрового об'єкта на логічні стани. У цьому випадку об’єкт має лише один стан для кожного різного типу поведінки. FSM реалізується простими операторами if-then, він використовує графічні представлення, які є частиною дев’яти діаграм, визначених Уніфікованою мовою моделювання (UML).

В ігровому ШІ можливі способи використання FSM нескінченні. FSM можна використовувати для визначення поведінки модулів у RTS. Крім того, його можна використовувати для аналізу вхідних даних від людини-гравця або навіть для імітації емоцій NPC. FSM є природним вибором для розробників ігор при проектуванні NPC. Він використовується в більшості комерційних комп’ютерних і відеоігор, таких як Age of Empires, а також у FPS-відеоіграх, таких як Quake, Quake 2, Doom і Half Life, або в RTS-грі, як-от Enemy Nations. Ігри, які зараз використовують FSM, також можуть використовувати інші параметри, такі як нечіткий стан і нейронна мережа.

2. Нечітка логіка в іграх.

Логічна логіка містить лише істинні або хибні твердження, тоді як нечітка логіка допускає проміжні значення, такі як «досить гаряче» або «дуже швидко», для опису безперервних або накладених станів, які можуть виникнути в математиці. Нечітка логіка може представити концепцію за допомогою найменшої кількості нечітких значень. Нечітка логіка, яка використовується в рішеннях, може базуватися на неповних або помилкових даних, які не можуть бути використані в булевій логіці. Це корисно для прийняття рішень, вибору поведінки та фільтрації введення/виведення.

Нечітка логіка використовується, щоб визначити, наскільки щось лякає гравця, щоб NPC вирішив, скільки вони розкривають «гравцю». Найбільш поширеним використанням цієї технології в комерційних іграх є відеоігри, такі як Platoon Leader; тактичні ігри в реальному часі, такі як SWAT 2; і стратегічні ігри, такі як CCTP. Якщо ми накладаємо нелінійну задачу або не маємо простого рішення, або якщо NN незастосовна, тоді підходить нечітка логіка.

3. Нейронна мережа в іграх.

Нейронні мережі (NN) містять низку відносних компонентів, пов’язаних із системою, які можуть виробляти результати на основі ідентифікації шаблонів у даних. Багато ігор, у яких використовуються нейронні мережі, можуть «навчатися» за допомогою досвіду або навчання та включають здатність приймати рішення. Нейронну мережу також можна постійно вдосконалювати, а це означає, що гравцеві доведеться змінити стиль стратегії гри, тобто він не повинен повторно використовувати ту саму стратегію. Нейронна мережа використовувалася в пригодницьких іграх, таких як Battle-cruiser 3000AD; гоночні ігри, такі як Dirt Track Racing, і стратегічні ігри, такі як Fields of Battle. Найцікавішим застосуванням нейронної мережі в ШІ є «Бійовий крейсер 3000AD». У цій грі NPC керує нейронна мережа.

4. Генетичний алгоритм в іграх.

Генетичний алгоритм (GA) використовується для пошуку оптимального рішення та може ефективно використовуватися в машинному навчанні для оцінки та пошуку вирішення конкретних проблем. GA починається з невеликої

кількості початкових стратегій, створюючи цілу сукупність, шукаючи рішення та оцінюючи кожну сукупність для вирішення проблеми. GA є відповідним рішенням, коли проблема або гра містить досить великий домен пошуку. Це також корисно для вирішення нелінійних задач.

GA використовується в RTS для адаптації комп'ютерної стратегії, використання слабких місць гравця та визначення поведінки окремих підрозділів, а не групи. «Плащ, кинджал, ДНК» (CDD) є прикладами використання GA в іграх RTS. У CDD гравці мають спільну «ДНК», яка відповідає за моніторинг і збереження продуктивності. GA використовується в RPG або FTS для розвитку поведінки, пов'язаної з персонажами та подіями.

5. Інтелектуальний агент в іграх.

Це програмний агент, який бачать у середовищі та діють у цьому середовищі для досягнення цілей за допомогою функції оцінки, яка також називається евристичною функцією, щоб допомогти агенту визначити найкраще значення. Крім того, агенти вивчили багато проблем в ШІ. Ігри є ідеальним середовищем для агентів, оскільки вони використовують реалістичне середовище, але є обмеження в доступній інформації, оскільки рішення повинні прийматися в умовах обмежень часу та тиску. Як правило, агенти в іграх використовують набори FSM і будь-які інші методи або будь-яку комбінацію деяких або всіх методів, які вирішують певні проблеми та дозволяють надсилати повідомлення.

Агент має здатність приймати рішення та виконувати завдання для досягнення своїх цілей, як це роблять люди. Можна сказати, що кожна гра, яка включає штучний інтелект, використовує певну форму агента. У FTS або RPG монстр буде активним агентом і більше підходить для простої реакції на те, що відбувається в грі. Стратегічні ігри потрібно ретельно планувати, особливо коли плануєте рух, який відбудеться пізніше в грі. Хороша архітектура агента RTS необхідна для забезпечення успіху. В Empire Earth RTS містить кілька компонентів, які називаються менеджерами. Є менеджери для цивілізацій, будівель, підрозділів, ресурсів, досліджень і боїв. Менеджер цивілізації — це

менеджер найвищого рівня, який відповідає за розвиток гравця та координацію між іншими менеджерами. Інші керівники нижчого рівня відповідають за надсилання запитів і звітів один одному. На рисунку 1.2 показано зв'язки між найпопулярнішими характеристиками агента та найпопулярнішими іграми.

	Aspects of autonomy								
	Reactivity	Supervision by higher level agents	Visible inter-agent communication	Complexity of decision making	Variety of action repertoire	Learning	Situatedness	Temporally continuous	Goal-oriented
World of Warcraft	Low	None	Low	Low	Low	No	Yes	Low	Yes
Half Life 2 (Single player mode	High	None	High	Low	High	No	Yes	Low	Yes
Supreme Commander	Low	High	Low	Low	High	No	Yes	High	Yes
Warcraft 3	Low	High	Low	Low	High	No	Yes	High	Yes
Black & White 2	Low	Low	Low	High	High	Yes	Yes	High	Yes
Tomb Raider: Anniversary	Low	None	High	Low	Low	No	Yes	Low	Yes
Unreal Tournament (2004bots)	High	None	High	High	High	No	Yes	High	Yes

Рисунок 1.2 – Зв'язок між характеристиками популярного агента та популярними іграми

6. Машинне навчання в іграх.

Це процес навчання в іграх, який загалом передбачає адаптацію поведінки гравців-супротивників з метою покращення продуктивності. Машинне навчання може допомогти охопити простір пошуку в комп'ютерних іграх і ефективно шукати вдалі комбінації параметрів. Машинне навчання також використовується в NPC. Машинне навчання можна використовувати для автоматизації створення інтелектуальної поведінки NPC. Машинне навчання з'являється в старих іграх, таких як Tic-Tac-Toe, Backgammon, Go, Othello та Checkers. Останнім часом методи машинного навчання почали з'являтися у відеоіграх, а також у боях, шутерах від першої та третьої особи, а також у стратегічних іграх.

7. Машина нечітких станів (FuSM) в іграх.

Існує суміш між нечіткою логікою та FSM. Замість того, щоб використовувати лише стан «увімкнено» або «вимкнено», ми додаємо «майже увімкнено» або «трохи увімкнено». FuSM все частіше використовуються в іграх, оскільки вони додають цікаві та різноманітні відповіді на NPC. Таким чином, гравець може взаємодіяти з NPC, який може зображувати різні ступені «божевільного» або «пораненого», а також гравець може випробувати різний досвід. FuSM використовувалися в FPS, щоб зробити ворога розумним. Базуючись на елементах у бойовій ситуації, нечітка логіка використовується, щоб дозволити ворожим персонажам втекти, якщо вони програють битву. Крім того, вони використовуються в CCTP, яка є RTS-грі. FuSM використовується в RPG для націлювання на точки NPC і агентів і ідеально підходить для контролю поведінки ігрових персонажів і для наповнення їх змінними діями та реакціями. В таблиці 1.2 показано зв’язки між популярним ШІ в іграх.

Таблиця 1.2

ШІ в ігрових додатках

AI	Game type	Game name
FSM, FuSM	FPS- RTS	Doom, Quake, unreal, CCTP
GA	RTS	CDD
Agent	Action game RTS	S.W.A.T2, CCTP, Empire Earth
NN	Racing, Action, Adventure, Strategy games	Dirt Track Racing, Battle-cruiser 3000AD, Heavy Gear
GA	RTS	CDD
Fuzzy logic	Adventure, Strategy games	SWAT, CCTP

1.3 Дослідження та опис ігрових двигунів

Ігровий движок — це програмна система, призначена для створення та розробки відеоігор. Основні функції, які зазвичай надаються ігровим механізмом, включають механізм візуалізації («рендерер») для 2-D або 3-D

графіки, фізичний механізм для виявлення зіткнень і реагування на них, а також граф сцени для керування як статичними, так і анімаційні моделі, звук, сценарії, штучний інтелект, мережі, потокове передавання, керування пам’яттю, потоки тощо. Існує багато доступних ігрових движків, і більшість з них дружні до користувача. Сучасні ігри часто розробляються з використанням ігрових движків, які можуть бути розгорнуті на персональних комп’ютерах, ігрових консолях, кишенькових комп’ютерах і мобільних пристроях, і поєднують кілька технологій з області інформатики, таких як: графіка, штучний інтелект, мережеве програмування. , мови та алгоритми. Ігрові механізми розроблені для керування багатьма рівнями досвіду програмування. Вони поділяються на власні версії (найнижчий рівень), здебільшого готові ігрові движки (середній рівень) і движки «вказуй і клацай» (найвищий рівень). Далі ми розповімо про найпопулярніший ігровий движок.

Механізми «наведіть і клацніть» стають все більш поширеними, оскільки вони включають повний ланцюжок інструментів, який дозволяє користувачеві вказати та клацнути для створення гри. 3DUinty є прикладом цього ігрового движка. Це безкоштовний інструмент розробки ігрового движка для індивідуального та професійного використання, який є одним із найкращих ігрових движків на ринку. Розробникам дуже легко користуватися, і вони можуть легко до нього дістатися. Завдяки дружньому інтерфейсу відносно легко зрозуміти майже всі розділи за кілька хвилин використання. В Інтернеті є багато посібників, щоб дізнатися про це. Він має дуже високу графічну якість і постійно вдосконалюється. Графіка виглядає майже справжньою та має освітлення та тіні в реальному часі тощо. 3DUnity має чудову реалістичну фізику, наприклад тверде тіло. Коллайдери та м’які тіла допомагають зробити гру більш реальною, що дозволяє легко приєднати їх до гри. З плагінами в магазині активів стає набагато легше зробити гру більш реалістичною. Це легко для більшості користувачів

3DUnity, оскільки будь-яка додаткова допомога надається рішеннями 3DUnity у сховищі ресурсів. За допомогою 3DUnity одного кліка достатньо,

щоб опублікувати гру на кількох платформах, таких як Xbox 360, PS3, Wii, Mac, ПК, Linux, Web, Adobe Flash, Android та IOS. З усіма підтримуваними платформами гру легко поширювати [9]. Сценарії/програмування двигуна Unity побудовано на Mono, реалізації .NET з відкритим кодом. Існує три варіанти мови програмування: C Sharp, Unity Script (яка в основному є Javascript) і Boo (мова, схожа на Python).

Для створення нашої гри ми вибрали 3DUnity як ігровий двигун, оскільки він працює на високому рівні та може бути легко опублікований на багатьох платформах.

3DUnity. Механізми «наведіть і клацніть» стають все більш поширеними, оскільки вони включають повний ланцюжок інструментів, який дозволяє користувачеві вказати та клацнути для створення гри. 3DUinty є прикладом цього ігрового движка. Це безкоштовний інструмент розробки ігрового движка для індивідуального та професійного використання, який є одним із найкращих ігрових движків на ринку. Розробникам дуже легко користуватися, і вони можуть легко до нього дістатися.

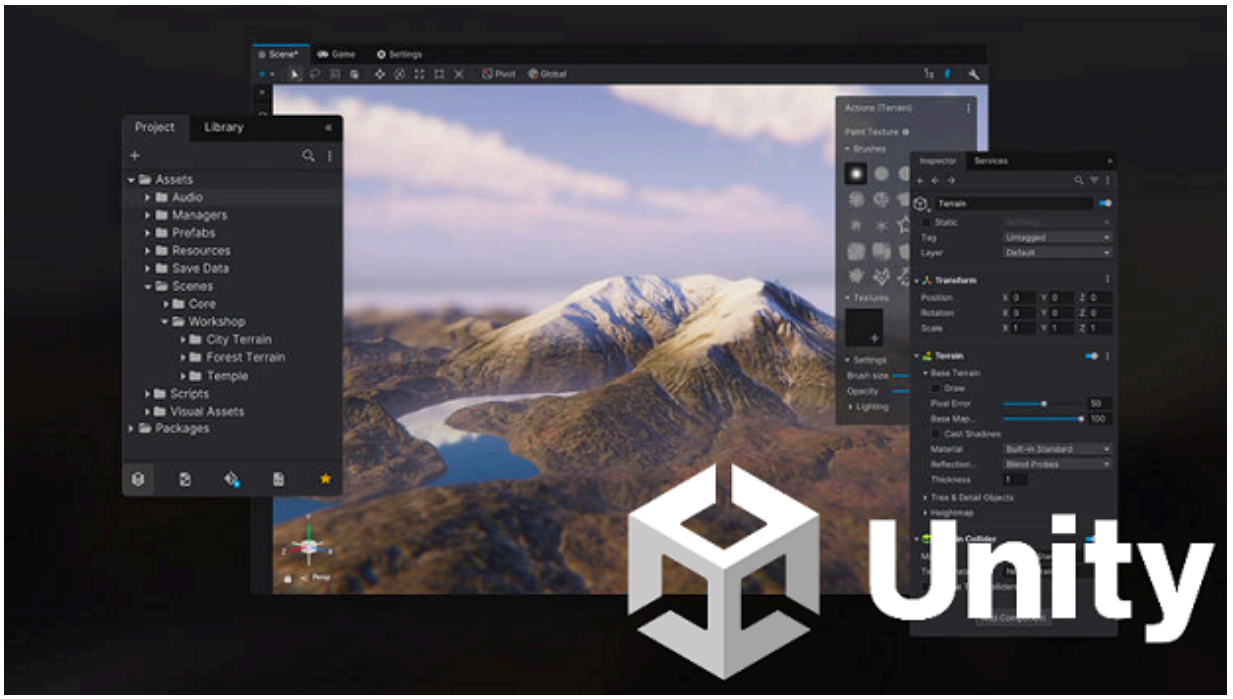


Рисунок 1.3 – Інтерфейс 3DUnity

Завдяки дружньому інтерфейсу відносно легко зрозуміти майже всі розділи за кілька хвилин використання. В Інтернеті є багато посібників, щоб дізнатися про це. Він має дуже високу графічну якість і постійно вдосконалюється. Графіка виглядає майже справжньою та має освітлення та тіні в реальному часі тощо. 3DUnity має чудову реалістичну фізику, наприклад тверде тіло. Коллайдери та м'які тіла допомагають зробити гру більш реальною, що дозволяє легко приєднати їх до гри. З плагінами в магазині активів стає набагато легше зробити гру більш реалістичною. Це легко для більшості користувачів3DUnity, оскільки будь-яка додаткова допомога надається рішеннями 3DUnity у сховищі ресурсів. За допомогою 3DUnity одного кліка достатньо, щоб опублікувати гру на кількох платформах, таких як Xbox 360, PS3, Wii, Mac, ПК, Linux, Web, Adobe Flash, Android та IOS. З усіма підтримуваними платформами гру легко поширювати [9]. Сценарії/програмування двигуна Unity побудовано на Mono, реалізації .NET з відкритим кодом. Існує три варіанти мови програмування: C Sharp, Unity Script (яка в основному є Javascript) і Boo (мова, схожа на Python) [1].

Для створення нашої гри ми вибрали 3DUnity як ігровий двигун, оскільки він працює на високому рівні та може бути легко опублікований на багатьох платформах.

Unreal Development Kit (UDK). UDK є прикладом майже готового ігрового движка, який миттєво готовий до роботи в прайм-тайм, із коробки, із рендерингом, введенням, графічним інтерфейсом користувача та фізикою. Двигун UDK використовувався для штучного інтелекту, графіки та всіх інших функцій гри, окрім фізики. Це було передано іншому двигуну під назвою «Havok» для забезпечення фізики в грі; це приклад використання проміжного посуду. UDK має дві версії: одна безкоштовна для некомерційного використання та для комерційного використання. UDK є дещо складним для людей, які новачки займаються розробкою ігор, хоча навчитися ним користуватися не неможливо, але це вимагає додаткового заробітку.

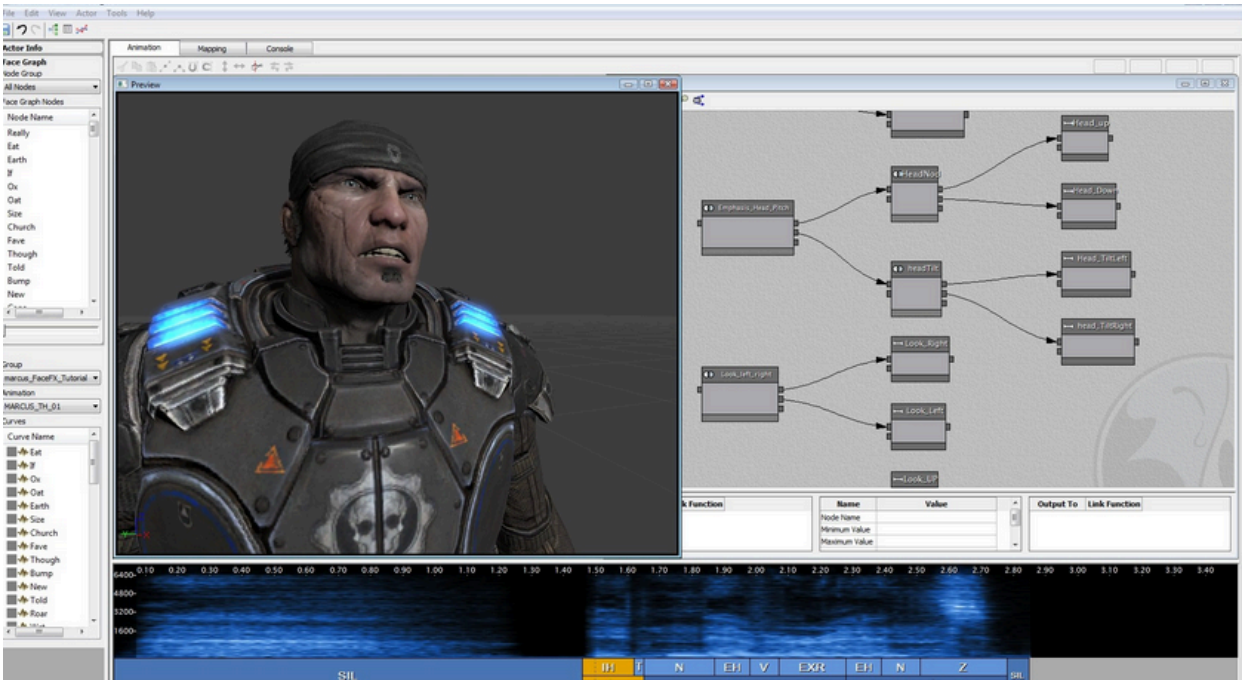


Рисунок 1.4 – Unreal Development Kit

Очікується, що в новій версії UDK інтерфейс буде набагато кращим і інтуїтивно зрозумілішим. Графіка надзвичайно приголомшлива, освітлення вдосконалено в ігровому движку та може динамічно відтворювати карти освітлення та підсвічування під час виконання та більшість іншого персоналу. Це дає змогу використовувати сотні об’єктів і текстур з гри для створення нових рівнів, але також можна імпортувати власні дані. UDK підтримує дві різні мови програмування: Kis-met, що використовується для базових сценаріїв, і сценарій для інших сценаріїв. Розробники можуть працювати тільки на ПК, а їх роботи можна публікувати на iOS, Android, Xbox, PS3 і Windows.

Microsoft XNA є одним з найкращих варіантів в ігровій індустрії на даний момент. Це пакет програмування .NET, який дозволяє користувачеві підготувати будь-яку невелику та середню гру, яку він здатний написати, а потім він дозволяє користувачеві ділитися своїм творінням зі світом на Xbox Live [15].

Brains — це бібліотека ШІ для використання в іграх XNA. Він складається з кількох будівельних блоків, щоб користувач міг швидко почати роботу з прототипом ШІ, який легко впровадити в гру. Наразі це лише 2D, але не потребуватиме надто багато модифікацій для підтримки 3D. Brains AI також

міститиме список типів агентів. Цей тип використовується для забезпечення автономної поведінки гри. Агент зберігає деякі прості властивості позиціонування, такі як положення, радіус і комірки, в яких зараз знаходиться агент. Він також зберігає бажану орієнтацію та бажане положення агента для використання з контролером Locomotion. Агент може зберігати набір щупів, які можна використовувати для перегляду даних по всьому світу.

Blender 3D — це безкоштовна програма з відкритим кодом, яка підтримується Blender Foundation.

Вивчити Blender нелегко, але він має безмежні можливості та дає розуміння складності комп'ютерної анімації. Blender розвивався з часом, і його розробка призвела до появи нових випусків (версій) програми. Програма досягла такого етапу, коли розробники вимагали повного перегляду графічного інтерфейсу користувача (GUI) [2]. Blender використовується для створення інтерактивного вмісту в реальному часі та написаний з нуля на C++ як переважно незалежний компонент; він включає підтримку таких функцій, як сценарії Python і звук OpenAL 3D.

Симуляції реалістичної фізики в блендері можливі завдяки модулю Blender Game Engine (BGE). Цей модуль використовує систему графічних логічних блоків (так звані логічні блоки) для керування поведінкою та візуалізації об'єктів, розташованих у 3D-сцені. Усі атрибути об'єктів і методи (функції) BGE доступні для користувачів через Blender Python API, і всі вони можуть бути використані в настроюваних контролерах, реалізованих мовою програмування Python.

Game maker — це ігровий движок, створений компанією YoYo, який використовує платформи Android, Browser, iOS, Mac, PC і Windows Phone. Game Maker в основному використовується для створення двовимірних ігор із гнучким дизайном і може бути поганим або хорошим залежно від вимог користувача. Його інтерфейс приховує більшу частину кодування та дозволяє користувачам вводити дані, а не писати рядки коду, але ті, хто хоче зануритися в кодування, можуть зробити це в розширених областях програми. Якщо

користувач готовий займатися програмуванням, тоді можна налаштувати створення в багатьох жанрах, але якщо користувач стурбований програмуванням, тоді будуть обмеження на бічну прокрутку та перспективні ігри зверху вниз. Одним із важливих моментів у дослідженні ігор є архітектура гри, яку використовуватиме розробник. Ми взяли загальну архітектуру гри з [27], як показано на рисунку 1.5. Ця логіка гри містить історію гри. Аудіо та графіка – це модулі, які допомагають автору розповісти історію гравцеві. Обробник подій і модулі введення забезпечують логіку гри наступною дією гравця. Модуль рівня є модулем зберігання деталей про статичну поведінку, а модуль динаміки налаштовує динамічну поведінку персонажів гри [27].

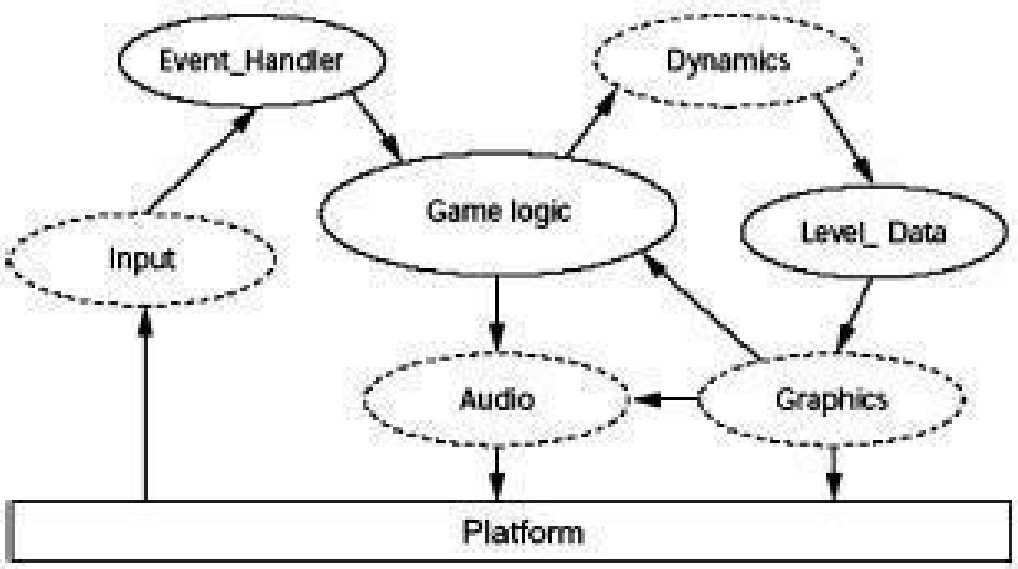


Рисунок 1.5 – Загальна архітектура ігрової платформи

Створення сучасних ігор — це неймовірно складне завдання, набагато складніше, ніж хтось може собі уявити. Це пов’язано з підвищеною складністю в поєднанні з мультидисциплінарним характером процесу розробки ігор (арт, звук, ігровий процес, системи керування, штучний інтелект, людський фактор, серед багатьох інших). Взаємодія з традиційною розробкою програмного забезпечення створює сценарій, який також збільшує цю складність. Щоб встановити цей зв’язок, нам потрібна методологія, яка б враховувала знання

програмної інженерії у сфері ігор. Як ми знаємо, індустрія ігор є дуже потужним сектором в індустрії розваг, що заробляє мільярди доларів прибутку та створює трильйони годин розваг.

Існує тісний зв’язок між агентом і іграми, як пояснювалося раніше в цьому розділі. Запропонований AOAB поєднає концепцію агента зі створенням загальної методології розробки ігор. У наступних двох розділах ми пояснимо агента загалом, а пізніше детально пояснимо методології AOSE.

1.4 Використання агентного підходу в розробці комп’ютерних ігор

Агенти мають можливість спілкуватися з іншими агентами, програмами та людьми. Ця комунікація включає надсилання та отримання повідомлень для досягнення своїх цілей або цілей суспільства/системи, в якій вони працюють. Комунікація може дозволити агентам координувати свої дії та поведінку для створення більш узгоджених систем. Ступінь координації - це ступінь, до якого можна уникнути непотрібних дій. Співпраця — це координація між неантагоністичними агентами, тоді як переговори — це координація між конкурентними агентами. Основа для інтелектуальних фізичних агентів (FIPA) — це ACL і набір одного або кількох параметрів повідомлення, які включають, наприклад, відправника, одержувача, виконавську роль і вміст, виражений мовою вмісту.

Платформи агентів використовуються для забезпечення більш реалістичного моделювання агентів у реальному світі. Платформа стає необхідною для того, щоб агенти могли спілкуватися один з одним за допомогою відповідних протоколів, повідомляти про присутність агента на платформі та представляти загальні стандарти для спільної роботи агентів. Платформи відрізняються за своїми функціями, можливостями та загальними стандартами. Існують деякі інструменти та середовища, які підтримують розробників агентів, які здебільшого базуються на мові програмування Java. Серед них інструментарій ZEUS, Java Agent Development (JADE), JACK

Intelligent Agents, Java Agent Template (JATLite), відкритого агента (OAA), Foundation for Intelligent Physical Agents-Open , Source (FIPA-OS), IBM Agent Building and Learning Environment (ABLE) і Agent Builder.

Мова моделювання для MAS є важливою підгалуззю в дослідженнях технології MAS. Дослідники та організації запропонували деякі методології розробки MAS, щоб допомогти розробникам полегшити процес розробки та вирішити внутрішньо складні проблеми MAS. У сфері розробки програмного забезпечення уніфікована мова моделювання (UML) пропонує стандартний спосіб специфікації, візуалізації, модифікації, конструювання та документування артефактів об'єктно-орієнтованої системи програмного забезпечення, яка розробляється. Найважливішою мотивацією, що спонукала до розробки AML, була існуюча потреба в готовій до використання, комплексній, універсальній та високовиразній мові моделювання, придатній для розробки комерційних програмних рішень на основі багатоагентних технологій.

AgentTool підтримує аналіз методології та етапи проектування. За допомогою цього інструменту можна представити безпеку, автентифікацію та повідомити про проблему за допомогою ролей, одночасних завдань і діаграм класів агентів. Це надає аналітику відповідальність за моніторинг прогресу проекту та оновлення діаграм протягом усіх етапів проекту.

Корисним інструментом агента, який використовувався з методологією MaSE, є AgentTool 3 (aT3); цей інструмент є продуктом лабораторії багатоагентної та кооперативної робототехніки (MACR) Університету штату Канзас. AgentTool — це графічне середовище розробки на основі Java, розроблене, щоб допомогти користувачам аналізувати, проектувати та впроваджувати MAS. Початкова версія aT3 — це плагін Eclipse, який надасть розробнику агентської системи безпрецедентну гнучкість, але все ще збереже можливості верифікації, надані раніше. У нашій практичній роботі ми використовували AgentTool для створення діаграми цілей, діаграми ролей і діаграми класу агента. Веб-сторінка AgentTool3 містить останню версію для завантаження, а також навчальні посібники та приклади. Ми використали

програмне забезпечення QSEE, яке є програмним забезпеченням моделі UML для створення послідовності та діаграми розгортання.

Залишається питання: «Що таке інтелектуальний агент?» Якості інтелектуального агента відповідають чотирьом властивостям волі, як пояснюється у визначенні агента. Інтелектуальні агенти використовувалися в широкому діапазоні програм. З цієї причини важливо визначити області застосування, які мають найбільше значення для роботи над розробкою агентів для інтерактивних розваг (IE). Для агентів IE середовище є віртуальним світом, у якому відбувається взаємодія та стає все складнішим. Було зроблено ряд спроб створити NPC для комп'ютерних ігор з використанням методів, заснованих на агентах. Ця робота визначає потенційну архітектуру інтелектуального агента для використання в програмах IE. Агенти в програмах IE повинні мати можливість залучати користувача (або гравця) до цікавої та, очевидно, розважальної взаємодії. Крім того, NPC повинні виглядати так, щоб залучати інших NPC до цікавих соціальних взаємодій.

Висновки до розділу 1

В даному розділі виконано аналіз предметної області використання штучного інтелекту для розробки комп'ютерних ігор. Здійснено визначення та класифікацію ігор, описано процеси використання штучного інтелекту в іграх. Також представлено використання агентного підходу в розробці комп'ютерних ігор.

РОЗДІЛ 2. АГЕНТНО-ОРІЄНТОВАНА МЕТОДОЛОГІЯ ПОБУДОВИ КОМП'ЮТЕРНИХ ІГОР

2.1 Огляд методологій розробки програмного забезпечення в ігрових додатках

Розробка ігор еволюціонувала до великих проектів, у яких працюють сотні людей, а час розробки тепер вимірюється роками. На відміну від більшості інших областей застосування програмного забезпечення, розробка ігор представляє унікальні проблеми, пов'язані з кількома дисциплінами, які сприяють іграм. У традиційних системах і розробці програмного забезпечення існує багато методологій. Деякі з цих методологій включають водоспад, інкрементний, прототипування та спіраль. Кожен структурований як лінійний або ітеративний, хоча іноді є гібридом обох, і зазвичай використовується в методології розробки ігор. Більшість лінійних методологій класифікуються як прогнозні, навіть якщо вони містять певну ітерацію; хоча вони зазвичай слідують за послідовними фазами, як-от методологія Waterfall. Методологія прототипування передбачає розбиття системи на невеликі сегменти та залучення користувача до процесу. Спіральна методологія поєднує лінійну та ітераційну структуру. Спіральний розвиток розбиває проекти на кілька циклів, усі з яких потім слідують за набором все більшими кроками.

Більшість методологій, обраних для використання розробниками ігор, можна охарактеризувати як прогнозні, тобто всебічне планування окремого завдання до фактичної розробки; або адаптивний, тобто використання кількох ітерацій і прототипів для формування гри та її дизайну на основі зворотного зв'язку та аналізу. У наступному розділі ми зосередимося на найпопулярнішій адаптивній методології розробки програмного забезпечення «Agile», яка в основному використовується в методології розробки ігор.

2.2 Структура методології розробки ігор (GDMF)

Структура, представлена нижче, базується на двох критеріях: по-перше, деякі адаптації існуючих якісних структур; по-друге, нові системи кількісного оцінювання. Крім того, результати, отримані з цих конкретних фреймворків, скеровують нас у виборі найбільш підходящої методології для домену розробки ігор. Ми провели порівняння відомих методологій, відібраних за такими критеріями:

– вони були детально описані та мали повний життєвий цикл. Більшість існуючих AOSE зосереджується лише на аналізі та проектуванні; тоді як MaSE, Prometheus і Tropos мають повнужиттєвий цикл;

– на них впливає корінь програмної інженерії;

– спільнота агентів сприймає їх як значимі.

Відповідно до цих критеріїв ми вирішили взяти до уваги лише MaSE та Tropos. Кількісна оцінка є важливою складовою процесу оцінювання, оскільки вона ґрунтується на фіксованих результатах для цілей порівняння. Більшість попередніх досліджень зосереджувалися на якісному підході, що дозволяло проводити порівняння методологій. Під час огляду літератури щодо кількісної оцінки виникли деякі труднощі:

Більшість оцінок, порівнювали дві методології, використовуючи конкретне прикладне дослідження для визначення результатів.

Для оцінки не використовувалися стандартні атрибути.

Щоб оцінити нашу структуру, були використані такі критерії:

1. Виберіть загальні критерії. Що стосується якісної оцінки, ми вирішили спочатку прийняти методологію з [42], оскільки його точна оцінка охоплювала якісні критерії та використовувала методи аналізу на основі ознак, а по-друге, з [37], оскільки його оцінка базувалася на методах опитування. Що стосується кількісних критеріїв оцінки, критерії були розділені на три підкритерії: шляхом передачі існуючих значень якісних атрибутів кількісних

чисел, роботи з метриками метамоделі та методами оцінки варіантів використання [25].

2. Переведіть якісні атрибути в кількісні значення, а потім перетворіть ці значення за допомогою запропонованої загальної шкали для кожного показника, як показано нижче:

- Yes-No to 0-1 and the common scale 0-10
- None-Low- Medium- High To 0-1-2-3 and the common scale 0-3-7-10

2.3 Опис методики та оцінка тривалості розробки гри

AOAB має багато функцій, основна увага яких полягає в тому, щоб відповідати галузевим вимогам. Двома ключовими особливостями є чітко визначена тривалість і політика командної співпраці. У наступних двох розділах детально описано ці функції.

AOAB — це ітеративна методологія, яка зосереджена на наданні функцій. Діяльності з планування, документування та розробки повторюються на кожній ітерації, а кількість ітерацій залежить від обсягу проекту. Метою ітерації є надання багатьох робочих версій гри після короткого періоду ітерації. Відгуки, отримані від клієнтів щодо кожної ітерації, допоможуть вирішити багато проблем на ранній стадії, і, як наслідок, час розробки гри буде скорочено. AOAB має можливість почати роботу над новими функціями до завершення поточної функції. У цьому випадку тривалість періоду розробки гри буде скорочена, оскільки час не витрачається на очікування. На етапі планування замовник і розробник, як правило, співпрацюють, щоб вибрати нові функції, які потім додаються до журналу Sprint Backlog, щоб обговорити, чи є вони пріоритетними. Основні зусилля при використанні лише AOSE в основному витрачаються на підготовку документації, як показано на рисунку 2.1. AOAB скорочує кількість документації, створюючи GDD і розділяючи їх спринт.

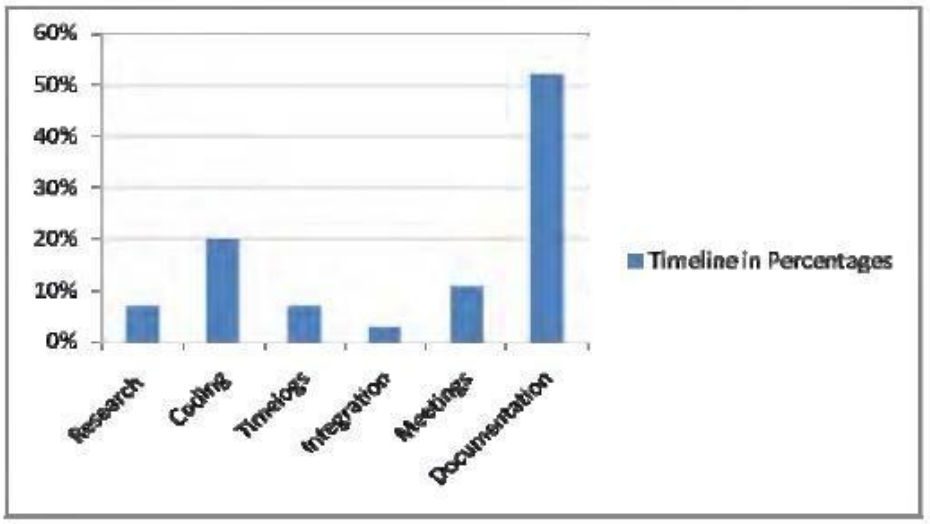


Рисунок 2.1 – Тривалість для AOSE

Документація важлива та потрібна для етапів аналізу та проектування, оскільки ці деталі потрібні для підтримки ігор або створення нових версій гри. Накладні витрати, пов’язані зі спілкуванням із великими командами, і вартість тривалих зусиль щодо розробки призвели до того, що зацікавлені сторони вимагають певності. Великі детальні проектні документи намагаються створити таку впевненість.

Команда розробників ігор відрізняється від традиційної команди розробників програмного забезпечення. Основна відмінність між ними полягає в тому, що для ігор групи розробки складаються з людей з різними сферами знань. На початковому етапі розробки гри потрібен сценарист, який представляє документи, зазвичай відомі як «концептуальний документ». [27]. Концептуальний документ буде перетворено на GDD дизайнером ігор, і GDD керуватиме грою процес розробки.

Ігрові компанії зазвичай наймають групу програмістів, включаючи, наприклад, програміста двигуна, програміста графіки, програміста ШІ, програміста звуку та програміста інструментів. Крім того, ігрова компанія може співпрацювати з музикантом і техніком звукових ефектів. Крім того, художнє оформлення для гри буде створено художниками 2D або 3D графіки. Перед випуском гри ігрова компанія найме тестерів, щоб грати та оцінювати ігри,

знаходити помилки та пропонувати вирішення проблем або рекомендувати зміни в грі. Мультидисциплінарні команди мають переваги, наприклад, вони створюють середовище, яке заохочує творчість; однак існує також потреба забезпечити співпрацю між командами.

Основна мета створення гри – забезпечити гравців задоволенням і насолодою від гри. Взаємодія між замовниками та розробниками призводить до більшої задоволеності замовника та розуміння динамічних змін у вимогах замовника. При використанні Agile спілкування між розробником і замовником зазвичай відбувається щодня, тоді як в AOAB спілкування між замовником і розробником відбувається лише за потребами, оскільки непросто проводити щоденні зустрічі з усіма команди.

2.4 Опис життєвого циклу методології створення комп'ютерної гри

Першим кроком етапу вимог є підготовка GDD на основі концептуального документа, який розділений на Sprint Backlog. З Backlog спринту можна оцінити кількість ітерацій, які охопить фаза спринту.

Документ дизайну гри (GDD).

На цьому етапі, перш ніж створювати GDD, розробник повинен визначити мету гри, визначити «фактор веселощів» і вибрати типи людей, які будуть грати в гру. Створення GDD є важливим кроком, який необхідно виконати на першому етапі розробки гри, оскільки він відповідає за визначення обсягу проекту, а отже також впливає на етапи розробки та тестування гри. Поганий GDD може призвести до розповзання функцій і, як наслідок, до затримок і, можливо, пропуску етапів.

Не існує стандартного способу побудови GDD, хоча він вимагає вичерпного опису всіх аспектів гри. Він також повинен описувати об'єкти та персонажів гри, їхні ефекти, те, як вони взаємодіють, а також їх роль і поведінку в грі. GDD зазнає багатьох змін, а також матиме додаткові вимоги. Слід також оцінити ризики будь-яких змін, а також те, чи можна дотриматися

термінів. Пізніше на етапі вимог GDD буде переведено в Беклог спринту для невеликих ігор, оскільки це може бути необов’язковим при перекладі будь-яких вимог безпосередньо як Беклог продукту. Це може заощадити час, але також може збільшити ризик повзучої функції або призвести до того, що гра буде недостатньо цікавою [23].

Якщо GDD розроблено ретельно, керівник проекту зможе спланувати ітерацію Sprint Backlog так, щоб у гру можна було грати наприкінці кожної ітерації. Це має кілька переваг; по-перше, тестери можуть перевірити гру на наявність помилок у ігровому стані, який імітує те, з чим зіткнеться кінцевий користувач. Початок гри, у яку можна грати, якомога раніше допомагає команді побачити потенціал кінцевого продукту, а отже, це корисно для публікації гри до остаточного випуску. Основна мета роботи з шаблоном GDD — це надати стандартний загальний GDD, який можна використовувати для різних жанрів ігор, як і мета AOAB.

Беклог спринту.

Під час етапу планування спринту всім командам доручається розділити великий GDD, якщо він занадто великий, на Backlog продукту. Однак, якщо GDD не надто великий, це завдання не вимагається. Команда витратила значну кількість часу на аналіз вимог і наслідків підтримки мережевих ігор. Наприкінці спринту вони висловили бажання замінити цю функцію, оскільки її впровадження в достатньо збалансованій якості загрожує загальній якості гри.

Наприкінці кожного спринту команда може оновити резервну інформацію спринту та виконати необхідні дії для наступного спринту. Для фінальної презентації команда мала успішно представити повністю придатний для гри прототип із повним набором функцій. Вони могли легко визначати проблемні фази чи вирішальні моменти й охоче аналізували свій прогрес. Підсумовуючи, вони сприйняли Scrum як дуже підтримку на всіх етапах проекту, за єдиним винятком. На цьому етапі, після завершення GDD, ідея гри, іншими словами, про що насправді гра, обсяг проекту та те, що насправді потрібно зробити, має

стати зрозумілою для розробників і клієнтів. Після цього GDD має бути переведено в Backlog Sprint.

У кожній ітерації життєвого циклу гри найважливіші невиконані завдання слід спочатку вирішити та розділити на менші завдання. AOAB використовує ті ж методи, що й Agile, і показано на рисунку 2.2. Це означає, що наприкінці фази спринту деякі роботи все ще можуть бути на стадії розробки. Нову ітерацію можна розпочати, використовуючи новий Backlog Sprint. Мета полягає в тому, щоб досягти безперервного потоку створення контенту.

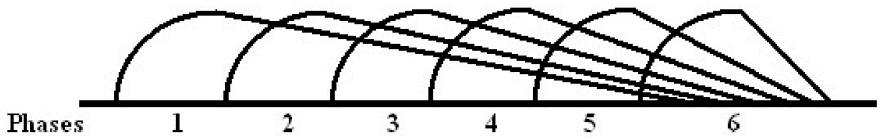


Рисунок 2.2 – Представлення фаз розробки

Фаза спринту.

Команда розробників зазвичай розглядає фазу спринту як міні-водоспад, з невеликими проектними документами, написаними на початку, і робочим кодом, що не об’єднується до завершення. Це краще, ніж традиційна модель Waterfall, хоча навчання команди, щоб підтримувати їх зв’язок між собою та переконатися, що будівництво триває, а не написання документів, ще більше покращить продуктивність. Одна з головних відмінностей між методами Scrum і Waterfall полягає в тому, що продукт зберігається в стані майже завершеного в кожному спринті, і що функції, додані в кожному спринті, мають рівень повноти, який підвищує цінність фінального результату. гра. Мета полягає в тому, щоб довести цінність функції в кожному спринті. Дизайн, кодування, налагодження, тестування та активи беруться до уваги.

В AOAB методологія MaSE використовується для охоплення підетапів спринту, таких як аналіз, проектування, реалізація та оцінка. Мета роботи таким чином полягає в тому, щоб кожні два-чотири тижні показувати клієнтам цінність функції, показувати, як вона покращує Sprint за Sprint, і в той же час

отримати документацію, яка буде корисною для оцінки або створення нового версія гри.

Етап аналізу.

Фаза аналізу включає три етапи: визначення цілей, застосування варіантів використання та уточнення ролей. В AOAB перша частина аналізу, яка фіксує цілі, уже визначена та додана до GDD. За допомогою GDD можна визначити випадки використання, а початковий набір ролей можна вдосконалити та розширити.

Метою моделі є перенесення GDD із системних вимог до набору стандартних цілей для гри. Цільова модель використовується в багатьох агентно-орієнтованих методологіях. Є два кроки для досягнення цілей: визначення та структурування цілей. Аналітик визначає цілі, аналізуючи всі доступні вимоги. На діаграмі ієрархії цілей цілі впорядковані за важливістю. Ціль моделі базується на декомпозиції І/АБО, яка перетворює загальну мету гри на набір підцілей. AND визначається тим, чи всі проміжні цілі повинні бути досягнуті для досягнення батьківської цілі. У той час як АБО визначається тим, чи є підціль альтернативним способом досягнення основної мети. Однак мета діаграми ієрархії цілей, яка використовується в MaSE, полягає у визначенні основних цілей рівня системи, а не цілей окремих агентів. Цільова модель створює високорівневу специфікацію щодо того, що повинна робити система.

Застосування випадків використання.

Крок варіантів використання важливий для перетворення цілей у ролі та пов’язані завдання. Варіанти використання витягуються з системних вимог і є наративними описами послідовності подій, які визначають бажану поведінку системи. Щоб допомогти визначити фактичні комунікації, необхідні в MAS, варіанти використання перетворюються на діаграми послідовності. Діаграми послідовності MaSE схожі на стандартні діаграми послідовності UML, за винятком того, що вони використовуються для зображення послідовності подій між ролями, щоб пояснити комунікацію, яка має відбутися. Ролі, визначені на цьому кроці, утворюють початковий набір ролей, які використовуються для

повного визначення системних ролей на наступному кроці, а ідентифіковані події також використовуються пізніше для визначення завдань.

Модель ролі та завдання.

Завдання модельних ролей визначає всі ролі в організації, а також їхню взаємодію одна з одною та зовнішніми акторами. Результатом виконання завдання «Рольові моделі» є модель для наслідування. Метою рольового моделювання є призначення кожної підцілі моделі цілей організації певній ролі. Ролі визначаються з варіантів використання, а також цілей системи. Усі системні цілі враховуються шляхом пов’язування кожної цілі з певною роллю, яку зрештою виконує принаймні один агент у остаточному проекті. Кожна ціль, як правило, співвідноситься з однією роллю; однак є багато ситуацій, у яких корисно поєднати кілька цілей в одній ролі для зручності чи ефективності. Визначення ролей зафіксовано в стандартній рольовій моделі. Після визначення ролей визначаються детальні завдання, які визначають, як роль досягає своїх цілей, і призначаються для конкретних ролей. Набір одночасних завдань забезпечує високорівневий опис того, що має передбачати роль, щоб задовольнити свої цілі; включаючи те, як він повинен взаємодіяти з іншими ролями. Цей крок задокументовано в розширеній рольовій моделі.

Фаза проектування.

На етапі проектування моделі аналізу перетворюються на конструкції, корисні для створення гри. Фаза проектування складається з трьох етапів: створення класів агентів, побудова розмов і складання класів агентів. В AOAB основна увага зосереджена на створенні діаграми класів агентів, яка є найважливішою частиною етапу проектування. Конструювання розмов і складання класів агентів номіновано через необхідність зосередитися на діаграмах, які будуть корисні або на наступних етапах створення гри, або після виходу гри.

Етап впровадження.

Генерація коду досягається на етапі впровадження. Гра реалізована поетапно та є результатом спринту. База може бути реалізована на одній або

кількох мовах програмування. Метою завдання створення коду є взяти всі моделі дизайну, створені на етапі розробки, і перетворити їх у код, який правильно реалізує моделі. Існує багато підходів до генерації коду, залежно від обраної платформи виконання та мови реалізації.

Схема розгортання.

Діаграма, що стосується фази впровадження, є схемою розгортання. Діаграми розгортання визначають конфігурацію фактичної системи, яка буде реалізована; крім того, вони визначають загальну архітектуру системи, щоб показати кількість, тип і розташування агентів у системі.

Етап оцінювання.

Компанії, які розробляють ігри, мають сильну конкуренцію, і ігровий досвід став важливим фактором у диференціації подібних типів ігор. Якщо ігровий процес не є оптимальним, гравці можуть легко перейти до іншої гри. Ігровий досвід можна оцінити, коли прототип реалізований і готовий до бета-тестування. На цьому етапі виправлення будь-яких проблем буде занадто дорогим, або графік проекту не допускатиме жодних затримок з маркетингових причин. Як наслідок, існує потреба в методі оцінки, який може ідентифікувати ці проблеми до початку бета-тестування, таким чином забезпечуючи достатній час для внесення виправлень.

Процес оцінювання, відомий як формуюче оцінювання, проводиться для виявлення проблем. Зазвичай вони визначаються на ранній стадії процесів розробки гри та можуть бути використані для покращення та вдосконалення гри на кожній ітерації, перш ніж вона буде готова для звільнення.

У методології розробки ігор етап оцінки виконується двома типами оцінювачів: експертами-оцінювачами та реальними користувачами. Експерти-оцінювачі — це особи, які володіють як знаннями, так і досвідом проведення оцінок із застосуванням різних методів експертної оцінки. При цьому реальні користувачі - це група цільових користувачів, для яких розробляється гра. Ці користувачі будуть респондентами в оцінюванні з використанням різних методів оцінювання користувачів. Крім того, процес буде

повторено, а результати поточної ітерації порівнюватимуться з результатами попередніх ітерацій до випуску гри. Щоб вирішити проблему оцінювання, пропонуються різні набори протоколів, засновані на різних точках зору.

Структура протоколу оцінки гри (GEPF).

Особливу увагу потрібно приділити визначенню методів оцінки ігор. Оцінка гри все ще триває, і набори протоколів досить різні, але містять деякі загальні проблеми. Доступно декілька наборів протоколів, і важливо ретельно вибирати змінні для вимірювання, а також правильні методи збору даних [30]. Деякі запропоновані протоколи не перевірені, тоді як інші націлені на певні ігрові жанри або не охоплюють усіх потреб оцінювання. Це призводить до важливого питання: «Які аспекти ігор можна оцінювати?»

Це може бути складним завданням визначити протокол, який здатний охопити основні аспекти оцінки гри. Тому важливо вибрати набори протоколів високорівневого набору для різних ігрових жанрів, не втрачаючи можливості орієнтуватися для оцінювачів на етапі оцінювання. У той же час слід розглянути всі аспекти оцінки, щоб отримати вимірювання, яке підходить для вдосконалення наступної ітерації методології розробки ігор. Крім того, необхідно використовувати правильні методи збору та порівняння результатів.

Ця робота спрямована на досягнення ясності, загальності та корисності наборів протоколів, щоб оцінити більшість ігрового жанру. Запропонована структура для оцінки ігор містить стандартні вимоги для полегшення роботи в компаніях-розробниках ігор. Замість того, щоб використовувати різні набори оцінок для кожного типу гри, набори протоколів, використані в цьому дослідженні, охоплюють усі важливі аспекти протоколу гри високого рівня. Завдання оцінювання перевіряє гру на наявність будь-яких проблем у будь-який час. Метою наборів протоколів є керівництво етапом оцінювання та нагадування оцінювачам звернути увагу на певні важливі аспекти, які будуть пояснені детально. Буде представлено початкову структуру протоколу високого рівня для ігор, які використовуватимуться в AOAB.

З наведених вище причин запропоновані набори для оцінювання містять протоколи, упорядкованих у чотири категорії; по-перше, можливість відтворення гри, яка стосується трьох основних питань: історії гри, процесу гри та ігрової механіки. По-друге, зручність використання гри, яка стосується двох основних питань: інтерфейс гри та керування грою. По-третє, якість гри, що складається з трьох ключових понять: функціональність гри, ефективність гри та адаптивність гри. Нарешті, задоволення від гри, яке охоплює елементи задоволення, які не містяться в попередніх наборах протоколів. Водночас методи оцінювання користувача охоплюватимуть набори протоколів, подібні до тих, які використовують експертні оцінювачі, за винятком ігрової механіки, яка стосується технік [21]. Оцінка користувачів зазвичай проводиться за допомогою анкети, інтерв'ю або сценарію, як показано на рисунку 2.3.

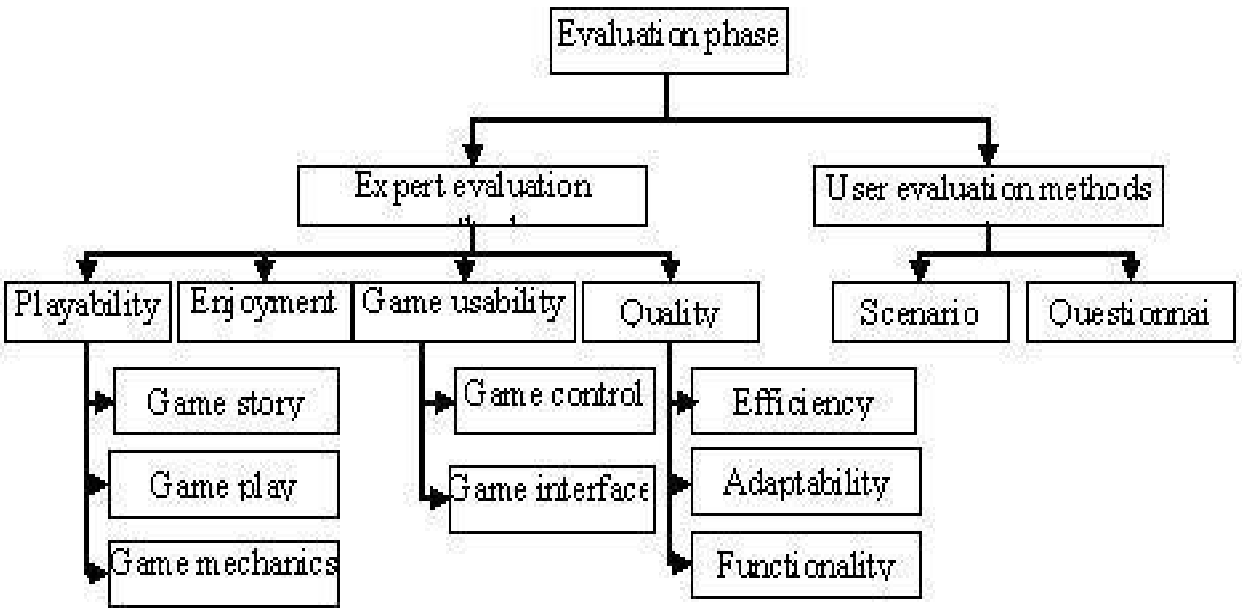


Рисунок 2.3 – Критерії оцінки гри

Процес оцінювання гри.

Початковим кроком у процесі оцінювання гри є пояснення загального процесу. Рисунок 2.4 ілюструє загальний процес, який необхідно дотримуватися в кожній ітерації методології розробки гри. Результати кожної ітерації порівнюються з результатами попередньої ітерації, щоб переконатися,

що всі виниклі проблеми вирішено. У кожній з оцінюваних ігор вимірюється значна кількість критеріїв і встановлюється кореляція між ними. Ці критерії розбиті на ряд більш вимірюваних факторів, визначених на основі поточної ігрової літератури.

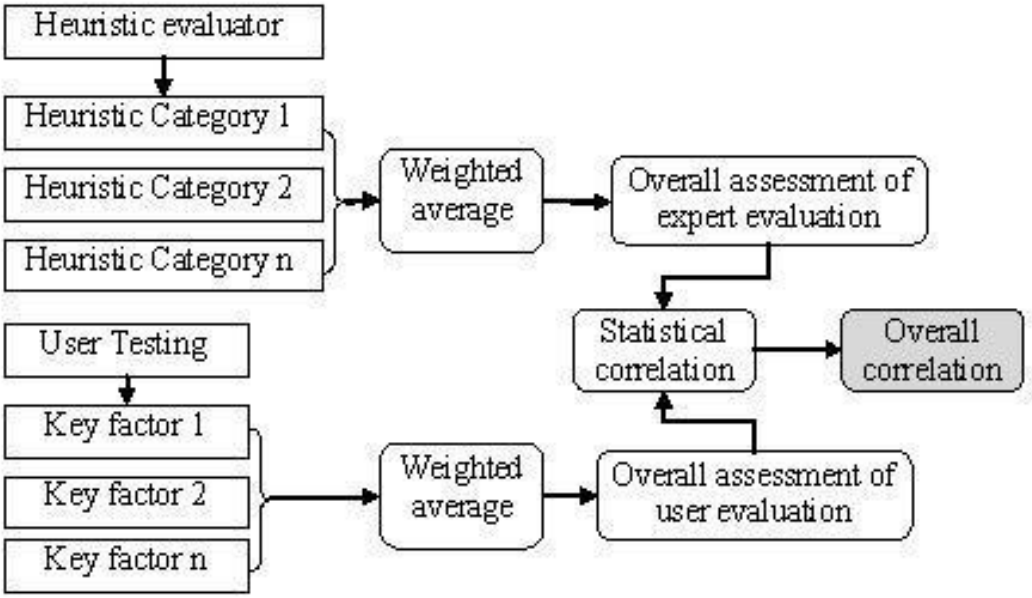


Рисунок 2.4 – Процес оцінювання гри

Дані загальної кореляції висвітлюють погляди користувачів, експертні оцінки та коментарі для розробки прототипу, а також ідентифікують конкретні елементи, які необхідно покращити.

Методи експертної оцінки. Сприйняття експертних методів оцінювання розробниками ігор є цікавим, оскільки вони можуть вимірювати різні аспекти, які допомагають в оцінці та тестуванні ігор. Використання експертних критеріїв оцінки дозволяє визначити та поглиблено оцінити багато проблем гри. Набори протоколів, які використовуються в цьому дослідженні, є загальними, що означає, що вони застосовні для оцінки більшості ігрових жанрів, охоплюючи можливість відтворення, якість, мобільність, задоволення та зручність використання. Ці критерії обрано тому, що вони є спільними для всіх ігор; вони охоплюють більшість аспектів, необхідних для оцінки ігор, і прості у використанні.

Зручність гри. Зручність використання гри охоплює аспекти керування грою та ігровий інтерфейс, за допомогою якого гравець взаємодіє з грою, як показано в таблиці 2.1.

Таблиця 2.1.

Розділ оцінки якості

Criteria	No.	Sub-Criteria	Description	Score (1-5)	Priority (1-3)
	4		The game allows for activities that keep the curiosity and the interest of the player in the content.		
	5		The game allows players to take decisions.		
	6	Efficiency	Is there no extra information?		
	7		The game has a good program structure that allows easy access to content and activities.		
	8		The speed of communication between the program and the user is adequate.		
	9		The program execution is efficient and with no operational errors.		
	10		The system is developed with originality.		
	11	Functionality	The information are well structured and does it adequately distinguish the objectives, context, results, multimedia resources.		
	12		The game checks all the alert message.		

Хороша зручність гри гарантує, що гравець проведе веселу та приємну сесію. Більшість існуючих наборів протоколів юзабіліті гри базуються на десяти протоколах юзабіліті які використовуються для виконання оцінок протоколів розробки програмного забезпечення та веб-сайтів. Інтерфейс гри повинен дозволяти гравцеві вільно керувати грою та відображати всю необхідну інформацію щодо статусу гри та будь-яких можливих дій.

Якість гри. Одним із важливих показників протягом життєвого циклу розробки гри є якість. Процес оцінювання якості починається з етапу ретельного планування, який включає мету оцінювання, терміни оцінювання та людей, залучених до проведення процесу оцінювання. Протокол якості використовувався для оцінки не тільки кінцевих версій ігор, але й якості протягом усього життєвого циклу розробки ігор, що дозволяє розробникам запобігати більшості збоїв ігор.

2.5 Критичний аналіз методології

Для AOAB пропонується не проводити щоденні зустрічі, щоб заощадити час. Зустрічі мають відбуватися лише щодня, коли виникають важливі питання в міждисциплінарних командах, таких як художники, музиканти, розробники та клієнти. Ці групи можуть містити підгрупи, такі як команда AI або команда текстур, оскільки AOAB створює функціональні комбінації спеціальностей. Щодо проблеми обсягу проекту, AOAB використовується для методів прототипу, а GDD – для охоплення важливого аспекту ігор. Ітерація в AOAB дає змогу розробнику розвивати функції та зменшувати кількість функцій у грі.

Щодо управління проектами та організації команди, AOAB пропонує суворі процеси, що лежать в основі зворотного зв’язку щодо проекту та спілкування між командами. Що стосується обсягу проекту та розширення функцій, багато ситуацій в індустрії розваг відображають численні особливості, виявлені під час розробки гри, які мають відношення до успіху гри.

AOAB — це не лінійний, а ітераційний процес. Таким чином, якщо виявлено цікаву особливість, її необхідно проаналізувати з точки зору її ризику і, якщо вона життєздатна, її слід додати до графіка проекту. Зазначається, що вартість змін у традиційному програмному забезпеченні зросла наприкінці проекту для вирішення будь-яких проблем, а пізня зміна системи призведе до додаткового часу та витрат. У той час як у випадку Agile вони також збільшаться, але зазвичай наприкінці проекту, оскільки природа Agile така, що

він може приймати додаткові вимоги та оновлення на наступних етапах проекту. Припускають, що наприкінці Спринту деякі роботи ще можуть бути в стадії розробки. Метою є досягнення безперервного потоку створення вмісту, що є основною концепцією AOAB.

Коли AOAB використовує концепції Agile, це покращує якість і ефективність великих складних ігрових проектів. Крім того, це зміцнює зв'язок між розробником і кінцевим користувачем. Дані з протоколу оцінювання та коментарі до розробленого прототипу визначають елементи, які необхідно вдосконалити в наступній ітерації методології. Для отримання кількісних результатів було додано два додаткові стовпці: один з оцінками протоколу, а інші – для встановлення пріоритетів для протоколу на основі жанру гри та точок зору оцінювачів.

Оскільки в протоколі цього дослідження використовується критичний огляд ігор для виявлення проблем і розробки набору вимог до дизайну ігор, стверджується, що ця загальна методологія є новим підходом, який можуть використовувати дослідники та дизайнери для розуміння проблем дизайну у більшості ігрових жанрів.

Висновки до розділу 2

В даному розділі виконано дослідження різних аспектів наборів протоколів та їх корисність на етапі оцінки розробки комп'ютерної гри. Більшість наборів протоколів можна використовувати на ранній стадії розробки; у цьому випадку розв'язання задач не потребуватиме додаткових витрат або часу, а покращить процес розробки ігри.

РОЗДІЛ 3. ВИБІР МЕТОДІВ ОЦІНКИ АГЕНТНО-ОРІЄНТОВАНОЇ МЕТОДОЛОГІЇ ДЛЯ ПІДВИЩЕННЯ ІНТЕРАКТИВНОСТІ В КОМП'ЮТЕРНИХ ІГРАХ

3.1 Мета та контекст оцінювання. Процес оцінки методики

Метою оцінювання методології AOAB є, по-перше, розробка методології на основі використання різних методів оцінювання, а по-друге, перевірка методології за допомогою відгуків професіоналів. Система оцінювання AOAB (AEF) надасть комплексний інструмент оцінки за багатьма критеріями. Ієрархічний критерій оцінки покращує зручність використання фреймворку та надає достатньо деталей для оцінки різних методологій розробки ігор.

Критичний аналіз представлений разом із чітким описом та обґрунтуванням найкращого методу оцінки для оцінки методології AOAB. Це продемонструє, що процедури збору даних були ретельно та систематично сплановані, дозволяючи читачеві оцінити якість процедури збору даних. Крім того, він документує методи дослідження, надаючи структуру для оцінки, яку можна легко використати в іншій оцінці методології. AEF буде зосереджено на трьох основних методах оцінки; по-перше, опитування сприятиме порівнянню різних методологій розробки ігор, а також буде проведено семінар. Ці семінари допоможуть досліднику отримати відгуки від експертів, які заповнять анкету. По-друге, це академічний експеримент. По-третє, буде представлено практичне дослідження компанії, що займається розробкою ігор. Наприкінці цієї глави представлено рамкову процедуру оцінювання AOAB та часові шкали оцінювання.

Перед початком оцінки необхідно прийняти певні рішення: «який метод оцінки використовувати?» І «яка мета оцінки AOAB?» Серед спільноти методів оцінювання [31] існує згода, що першим і найважливішим кроком будь-якого процесу оцінювання є визначення його мети, оскільки жодне раціональне

порівняння неможливе без визначення мети вправи . Залежно від мети спосіб проведення оцінювання та результати можуть суттєво відрізнятися. Оскільки розробка ігрових методологій все ще знаходиться на ранніх стадіях , цілі та завдання оцінювання AOAB є:

1. Краще зрозуміти природу методологій AOAB, включаючи їхню філософію, цілі, особливості тощо.

2. Визначте їхні сильні та слабкі сторони, а також спільні риси та відмінності, щоб виконати класифікацію та вдосконалити методологію майбутньої розробки ігор.

3. Проведіть оцінку з метою прийняття нової методології для існуючого процесу розробки.

4. Обрана методологія має найкращим чином відповідати потребам розробки гри та не вимагати суттєвих змін у поточному процесі практики.

Необхідно розуміти один важливий факт: різні методології підходять для різних ситуацій; таким чином, методологію слід обирати, розглядаючи різні питання. Ці фактори впливу можуть включати контекст проблеми, що розглядається, домен, організацію та її культуру. Проте очікується, що оцінка також допоможе в практичних питаннях, таких як визначення сфер застосування для кожної оцінюваної методології.

Структура оцінки залежить від контексту, що означає, що AOAB не очікується, щоб вона була найкращою за всіх обставин. Цілком можливо, що один із розробників гри може визначити AOAB як кращий, тоді як інший розробник прийде до іншого висновку. Отже , відмінності в результатах оцінювання можуть бути пов’язані з властивостями розробника гри, який виконує оцінювання, а не з самими методологіями. З цієї причини наша система оцінювання буде максимально загальною та намагатиметься охопити найважливіший критерій, щоб допомогти отримати результати. Крім того, AEF співпрацює з розробниками і дизайнерами ігор з різним досвідом, такими як академічні експерти, експерти галузі та представники Індії.

Структура оцінки AOAB (AEF) — це багатоетапний метод для визначення кожної частини структури за допомогою наборів критеріїв. Кожен набір критеріїв визначає характеристики та основні елементи AOAB. AEF потребує вдосконалення, щоб усунути невідповідності, конфлікти, збіги та додати критерії, які охоплюють більше одного аспекту. У випадку особливо складної системи чи гри «вибір із очевидно дуже широкого спектру доступних методів та інструментів сам по собі може бути складним і дорогим процесом» необхідно визначити систему систематичного оцінювання, щоб ідентифікувати відповідний метод оцінки. Існує три основні методи, які використовуються для оцінки нових технік, методологій та інструментів, а саме: опитування, тематичні дослідження та експерименти. Коротке визначення та опис кожного з них наведено на рисунку 3.1.

Empirical method	Description
Experiment	A set of subjects is asked to perform a task in a highly controlled environment. The results are derived from observing of the subjects during the experiment, from inspecting the task outcome or from questioning the subjects at the end of the procedure
Survey	A set of subjects is asked to fill-in questionnaires either directly, or via internet. The results are derived from the valid answers to the questionnaire
Case study	A project, an activity or an assignment is monitored with respect to the methodology under study. Results are directly derived from project measurements

Рисунок 3.1 – Методи оцінювання

Рисунок 3.2 описує порівняння між методами оцінки, що використовуються в ігровій інженерії. Області порівняння вибираються наступним чином:

1. Розробка ігрового програмного забезпечення є підкатегорією звичайної розробки програмного забезпечення.

2. традиційна інженерія програмного забезпечення є зрілим науковим напрямом.

3. Гнучка розробка програмного забезпечення обрана тому, що це молодий домен і, отже, доповнює традиційну розробку програмного забезпечення з огляду на зрілість області.

Результати дослідження показують, що тематичні дослідження найчастіше використовуються в дослідженнях розробки програмного забезпечення Agile.

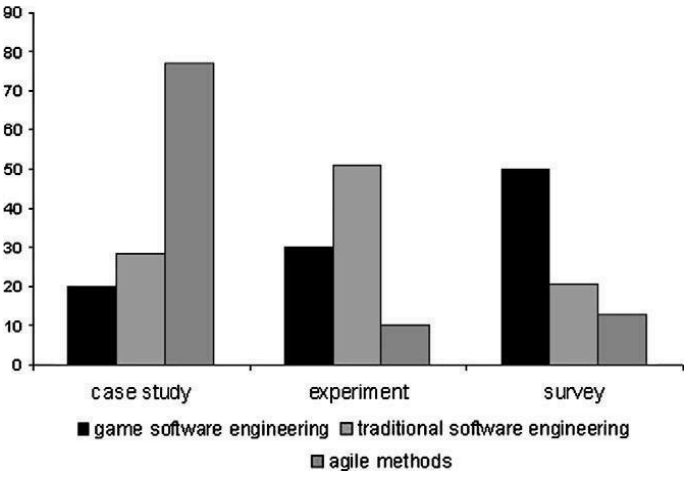


Рисунок 3.2 – Порівняння між методами оцінки

Крім того, опитування найчастіше використовуються в дослідженнях розробки ігор і, нарешті, експерименти частіше проводяться в дослідженнях інженерії програмного забезпечення. Процедура AEF включатиме опитування, тематичні дослідження та експерименти, як показано на Рисунок 3.3.

Більшість популярних методологій розробки програмного забезпечення створено як загальні та придатні для багатьох видів проектів. Методи оцінки AEF поділяються на два типи: якісна та кількісна оцінка.

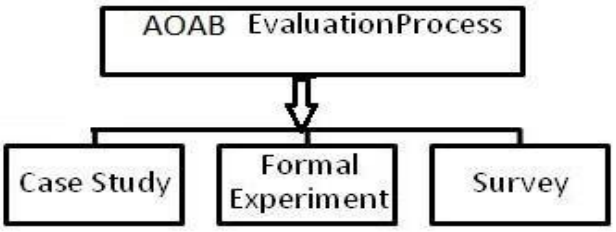


Рисунок 3.3 – Процес оцінювання AOAB

Підхід до опитування. Такий підхід не передбачає практичного використання оцінки. Опитування можна використовувати різними способами; іноді організацію або особу з досвідом просять використати методологію, а потім надати інформацію про методологію. Потім ця інформація може бути проаналізована за допомогою стандартних статистичних методів.

Ця методика використовується в поточній оцінці AOAB шляхом проведення семінарів в академічному та промисловому секторах. Ця частина дослідження буде детально описана у восьмому та дев'ятому розділах. Іншою формою оціночного опитування є порівняння нової методології з аналогічними існуючими методологіями. Оцінка методології є складним питанням, яке є предметом різних точок зору. Результати AOAB вже порівнювалися з точки зору часу розробки, управління проектами та переваг адаптації двох методологій для створення нової гібридної методології. У процесі методології оцінки для порівняння з методологією AOAB вибираються такі методології:

Розглянемо покращену версію MaSE, яка називається методологією багатоагентної розробки програмного забезпечення на основі організації (O-MaSE), щоб усунути брак промислових методів та інструментів для підтримки мультиагентної розробки. Питання керування та розгортання спочатку висвітлюються в O-MaSE, але не таким чином, щоб залучити клієнта до процесу або прийняти будь-які зміни в часі розробки, як це використовується з AOAB. Крім того, фаза тестування та оцінки не включена в життєвий цикл O-MaSE. Фреймворк методології O-MaSE базується на двох метамоделях: SPEM 2.0 і метамоделі O-MaSE. Метамодель SPEM визначає концепції, пов’язані з методологією, тоді як метамодель O-MaSE описує концепції, пов’язані з продуктом.

Відповідно до DESMET є методом для оцінки методів розробки програмного забезпечення, а інструменти називають переваги та недоліки опитування наступним чином:

Переваги Опитування

- вони використовують наявний досвід (тобто наявні дані);

- вони можуть підтвердити, що ефект поширюється на багато проектів/організацій;
- вони використовують стандартні методи статистичного аналізу;
- вони потребують менше часу та зусиль, ніж формальний експериментальний підхід.

Недоліки опитування

- вони покладаються на різні проекти/організації, які зберігають порівняльні дані;
- вони лише підтверджують зв’язок, а не причинно-наслідковий зв’язок;
- вони можуть бути упередженими через різницю між тими, хто відповідає, і тими, хто не відповідає.

З цим опитуванням пов’язано декілька завдань, як-от вибір типу опитування, наприклад веб-опитування чи особисте інтерв’ю, створення документації опитування, наприклад анкети, та визначення людей для участі в опитуванні. Нарешті, оцінювачі проведуть опитування, зберуть і проаналізують відповіді відповідно до дизайну опитування [46].

Формальний експеримент у реальному світі передбачає прохання реальних компаній виконати завдання або зіграти в гру за новою методологією. Результати будуть проаналізовані за допомогою стандартних статистичних методів. Для методології AOAB був проведений офіційний експеримент зі студентами бакалаврату, які виконували завдання з розробки та впровадження гри. Ця частина роботи буде детально розглянута у восьмому та дев’ятому розділах. Формальні експерименти підходять для дослідження стосунків; цей підхід, ймовірно, дасть найбільш надійні результати, оскільки він, здається, зменшує вплив відмінностей одного оцінювача. Однак це найбільш дорогий і трудомісткий підхід. Рамкова процедура оцінювання – це спосіб організації оцінювання. Добре розроблена структура системи оцінювання дозволяє розглядати результати з точки зору того, що методологія AOAB додає до існуючих методологій. Більшість систем оцінювання можуть бути кількісними,

якісними або гібридними. Запропонована система оцінювання для цього дослідження складається з трьох основних кроків, які є такими:

1. Визначте параметри, необхідні для оцінювання: у цьому розділі повинні бути визначені основні критерії, необхідні для оцінювання. Ці критерії базуються на наступних трьох коренях: game, Agile та agent основа.

2. Визначте методи, які використовуються для оцінювання: оцінка має на меті пояснити, наскільки добре AOAB відповідає потребам і культурі організації. Найпопулярнішими методами оцінювання є: тематичні дослідження, опитування, формальний експеримент, аналіз ознак і оцінювання.

3. Критичний аналіз результатів і пропозицій: ця частина структури оцінювання критично аналізуватиме результати попередніх кроків і вноситиме відповідні пропозиції щодо AOAB.

Оскільки існує багато різних методів оцінки, методи ранжуються в порядку вірогідного масштабу часу, необхідного для виконання оцінки. Відносні шкали часу поділяються на такі види:

1. **д**овгий, для проектів понад три місяці. Методи оцінювання довгострокових проектів мають форму академічних експериментів, оскільки процес підготовки та подання завдань займає 14 тижнів;

2. середній, кілька місяців. Зазвичай використовуються галузеві тематичні дослідження;

3. **к**оротший, кілька тижнів; презентація семінарів займає лише один день і кілька тижнів підготовки.

Важливо переконатися, що вибір методу оцінки відповідає зовнішнім часовим обмеженням. AEF охоплює всі розглянуті типи шкал часу.

3.2 Аналіз фаз розробки комп’ютерної гри

Фаза аналізу є дуже плідним періодом; це залежить головним чином від того, що збирається на етапі специфікації вимог. Першим кроком на етапі аналізу є визначення цілей, які зазвичай залежать від етапу специфікації вимог і

перетворюються на структурований набір системних цілей, зображених за допомогою діаграми ієрархії цілей для тестової гри. На діаграмі ієрархії цілей визначається головна ціль, яка менш імовірно зміниться, ніж детальні кроки та дії. Цілі впорядковано за важливістю. Підцілі призначаються для конкретних батьківських цілей і визначають, що потрібно зробити для досягнення батьківської цілі. Коротко, цільова модель для цієї роботи розділена на чотири.

Таблиця 3.1

Поділ розробки гри на спринти

Phase	Activity	Deliverables	Week
Analysis phase (Sprint 1)	Use Case, Sequence Diagram, Define tasks and role	Apply Use Case and role diagram	4
Design phase (Sprint 1)	Agent classes, Agent Architecture	Creating Agent class diagram	5
Implementation phase (Sprint 1)	Deployment Diagram	Create prototype 1	6
Analysis phase (Sprint 2)	use case, Sequence Diagram, Define tasks and role	Apply use case and role diagram	7
Design phase (Sprint 2)	Agent classes, Agent Architecture	Creating Agent class diagram	8
Implementation phase (Sprint 2)	Deployment Diagram	Create prototype 2	9
Analysis phase (Sprint 3)	use case, Sequence Diagram, Define tasks and role	Apply use case and role diagram	10
Design phase (Sprint 3)	Agent classes, Agent Architecture	Creating Agent class diagram	11
Implementation phase (Sprint 3)	Deployment Diagram	Create prototype 3	12
Testing and evaluation phase (Sprint 4)	Integrate game iterations, user evaluation, expert evaluation	Code review, final game release	13-15

Перша модель, як показано на рисунку 3.2, є моделлю головної цілі, яка складається з трьох підцілей. Кожна підціль представлена окремою схемою. Робочий план поділено на три основні спринти, як показано в таблиці 3.1. Sprint four зазвичай об’єднує три прототипи, створені після кожної ітерації.

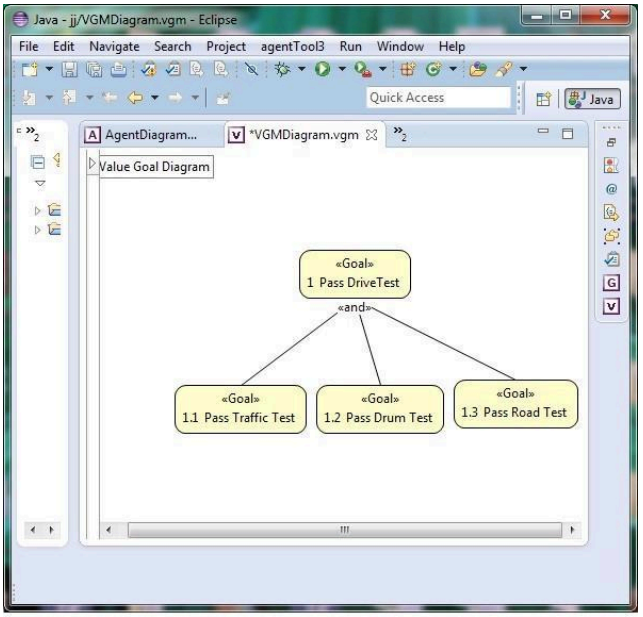


Рисунок 3.2 – Ієрархія головних цілей

Спринт 1. На рисунку 3.3 показано ієрархію цілей для першої частини іспиту з водіння. Перша частина, як правило, є перевіркою дорожніх знаків. Головна мета — пройти тест із варіантами відповідей, у якому гравець повинен читати та розпізнавати різні дорожні знаки.

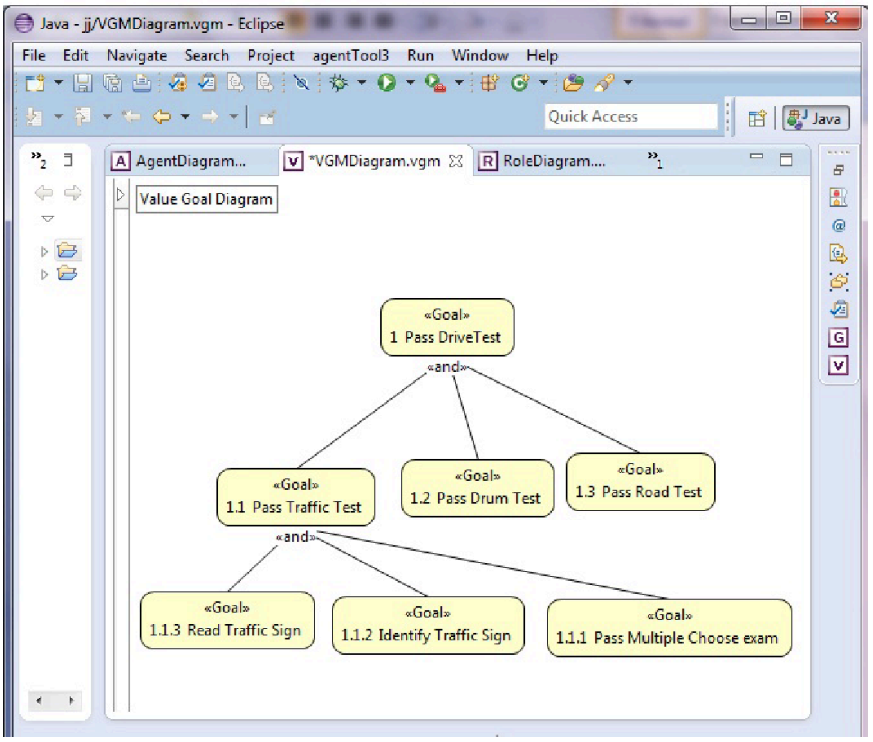


Рисунок 3.3 – Ієрархія цілей (спринт 1)

Спринт 2. На рисунку 3.4 представлена модель цілі для другої частини іспиту з водіння, яка стосується тесту на барабан.

Щоб пройти тест на барабан або паркування, гравець повинен вибрати тип передач, необхідний для тесту, який може бути ручним або автоматичним. Крім того, гравець повинен досягти мети контролювати автомобіль і мати можливість їздити вперед і назад.

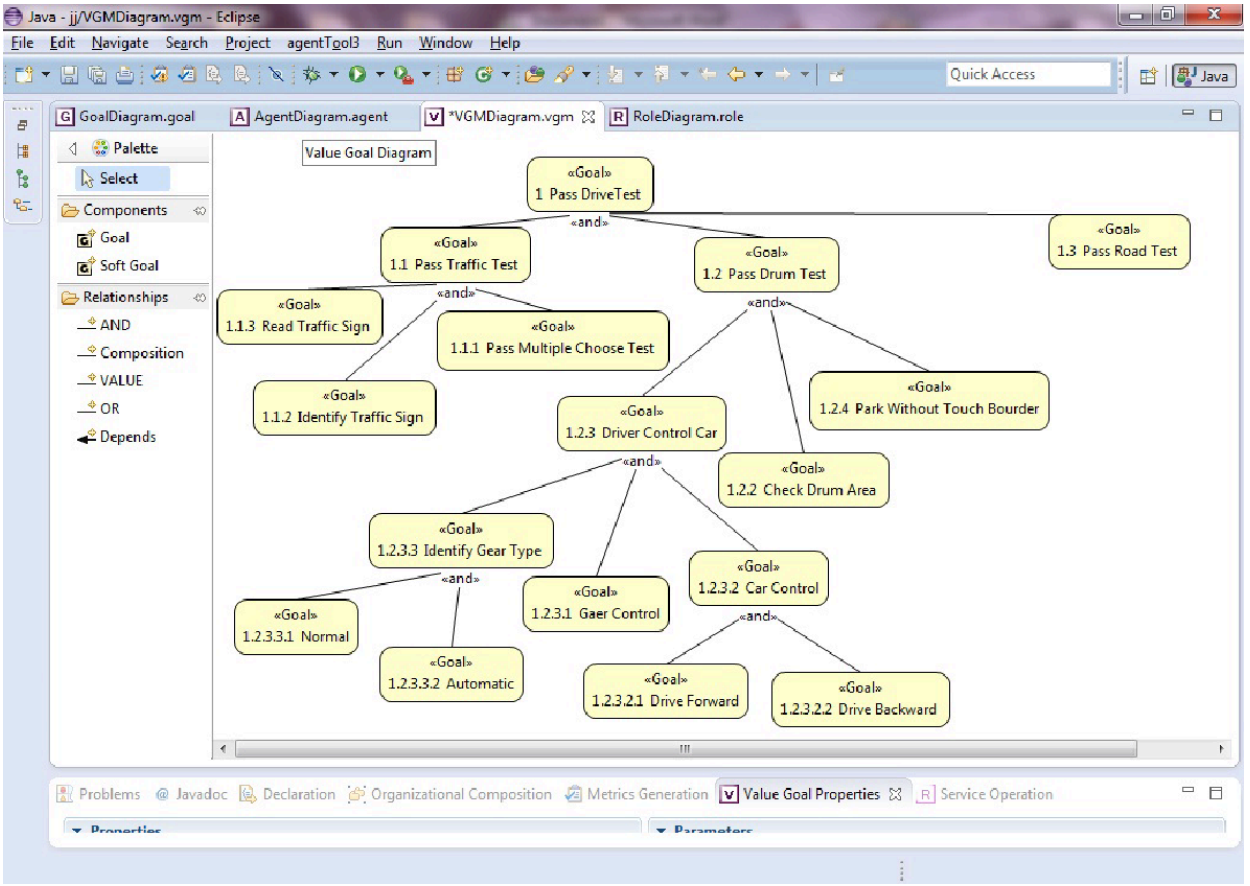


Рисунок 3.4 – Ієрархія цілей (спринт 2)

Спринт 3. На рисунку 3.5 показана цільова модель третьої частини іспиту з водіння, дорожніх іспитів. Це найскладніша частина тесту в реальному світі, тому що метою є повний контроль над автомобілем і здатність аналізувати накази екзаменатора. Гравець повинен ігнорувати неправильні накази і виконувати лише правильні. Крім того, гравець повинен знати правила дорожнього руху та інструкції.

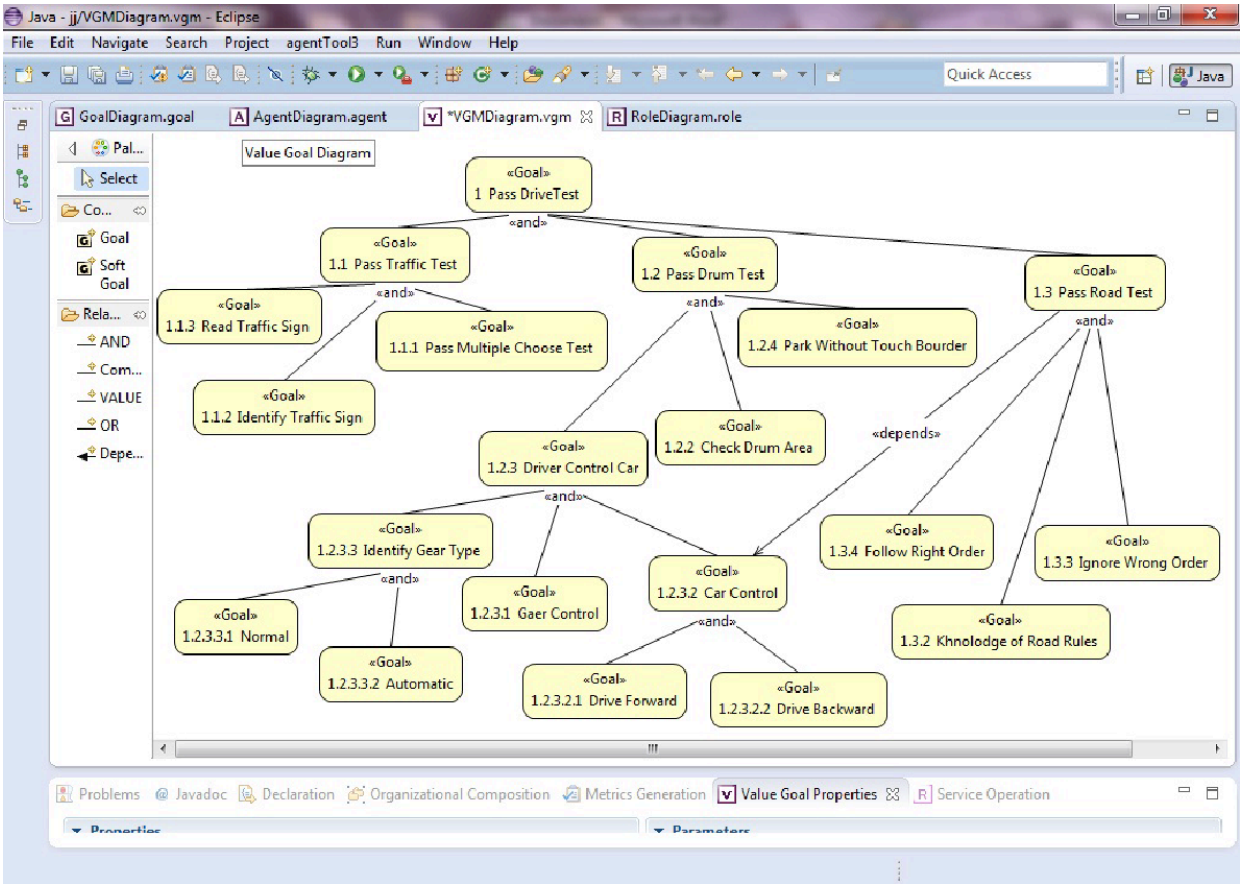


Рисунок 3.5 – Ієрархія цілей (спринт 3)

Застосування схеми використання та послідовності. Діаграма варіантів використання визначає фактичні комунікації, необхідні в іграх. Діаграма послідовності використовується для зображення послідовності подій у кількох ролях і визначає мінімальний зв’язок між ролями.

Діаграма послідовності показує події, які відбуваються, коли агент грає в гру. Кожна ціль, як правило, прив’язана до однієї ролі з пов’язаними завданнями. Для створення діаграми послідовності використовується програмне забезпечення QSEE.

Створено чотири діаграми послідовності; головна діаграма послідовності, рисунок 3.6 ілюструє основну частину роботи, яка поділена на три діаграми підпослідовності, що охоплюють три спринти. Нижче подано ще три діаграми підпослідовності по спринтах.

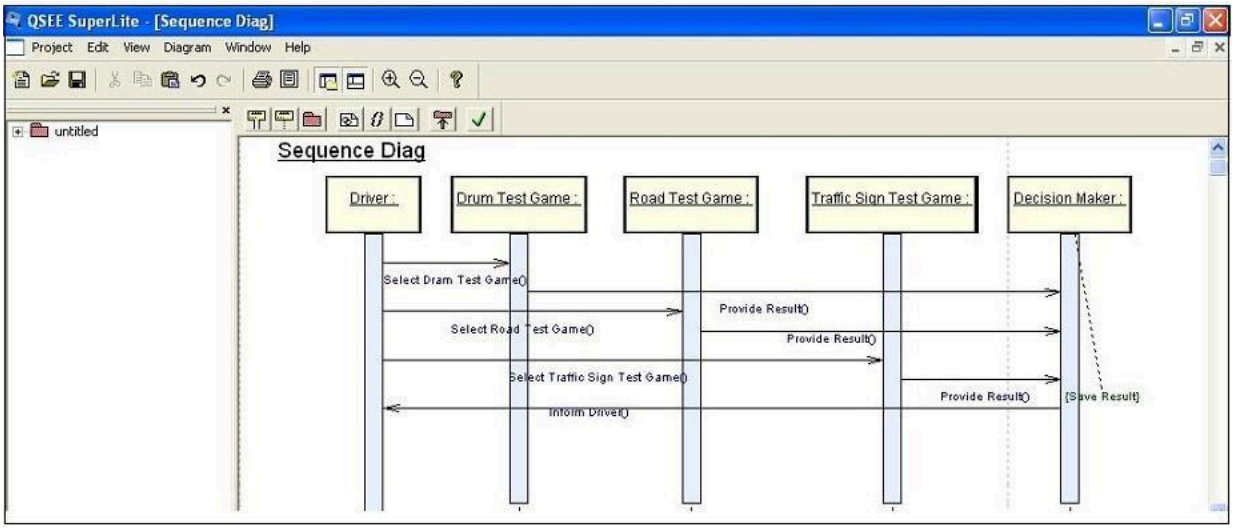


Рисунок 3.6 – Діаграма головної послідовності

Спринт 1На рисунку 3.7 показана діаграма послідовності першого спринту, яка охоплює етапи перевірки дорожніх знаків від початку перевірки до інформування водія про кінцевий результат.

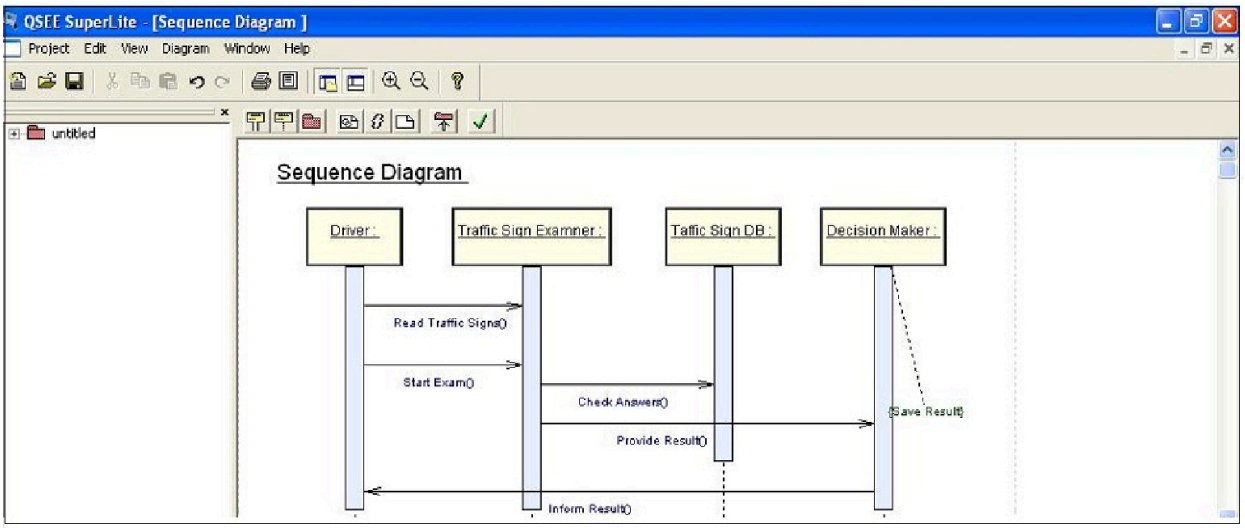


Рисунок 3.7 – Діаграма головної послідовності (спринт 1)

Спринт 2. На рисунку 3.8 представлена діаграма послідовності другого спринту, яка охоплює етапи випробування барабана або паркування від початку випробування до інформування водія про результат.

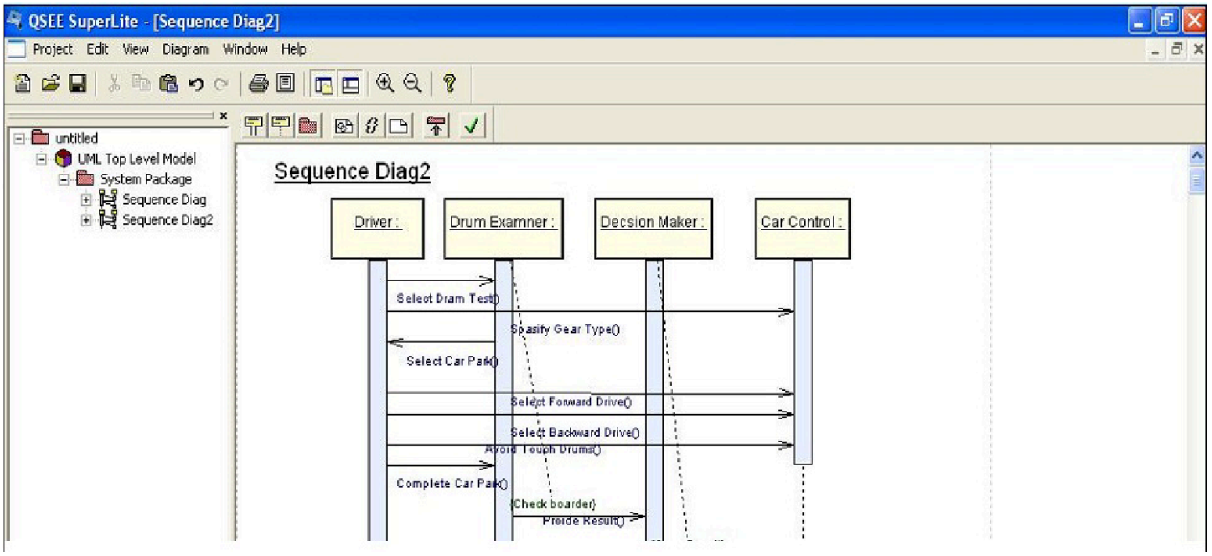


Рисунок 3.8 – Діаграма головної послідовності (спринт 2)

Спринт 3. На рисунку 3.9 показана діаграма послідовності третього спринту, яка охоплює етапи дорожнього випробування від початку випробування до інформування водія про результат.

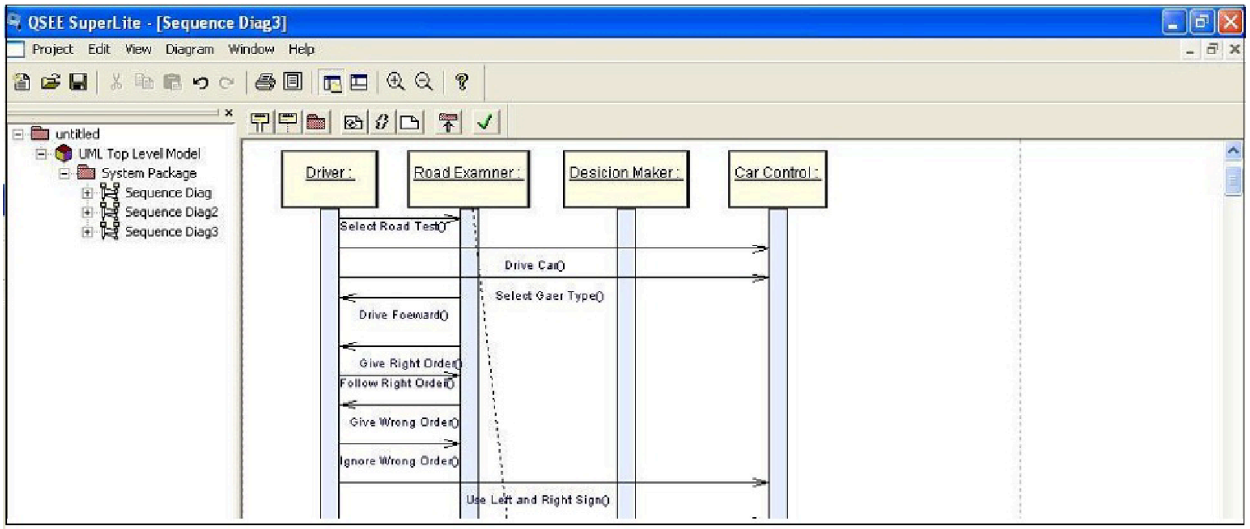


Рисунок 3.9 – Діаграма головної послідовності (спринт 3)

На рисунках 3.8 і 3.9 додано нові ролі, пов’язані з керуванням автомобілем, щоб переконатися, що водій має можливість керувати автомобілем і контролювати його. Крім того, ще однією додатковою опцією є вибір типу передач, автоматичного або ручного. Це пов’язано з тим, що в багатьох країнах,

включаючи Оман, є два типи водійських прав: одне лише для автомобілів з автоматичною коробкою передач, інше – для автомобілів з автоматичною та механічною коробкою передач.

Третім видом діяльності на етапі аналізу є модель ролі та завдання. Розробник повинен забезпечити визначення всіх необхідних ролей і розробити завдання, які визначають рольову поведінку та спілкування. Ця ж класифікація використовується для поділу роботи на три спринти. На рисунку 3.10 показана діаграма головних ролей.

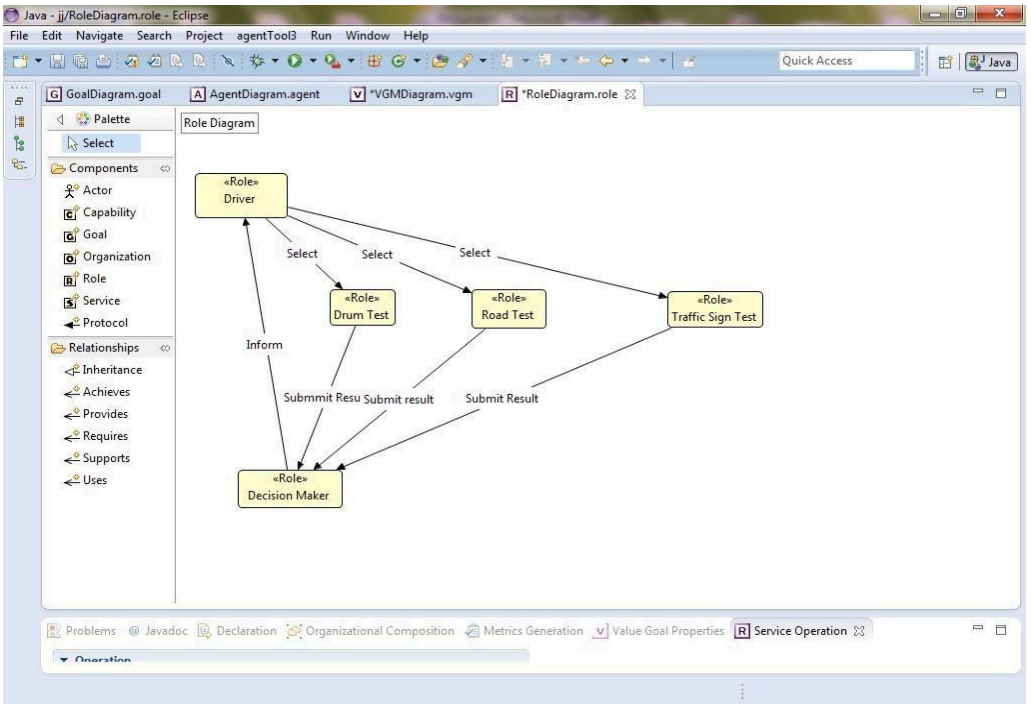


Рисунок 3.10 – Діаграма основних ролей

Спринт 1. Стосується перевірки дорожніх знаків; він містить більше деталей, показує результат, оцінює бали та додає базу даних, яка містить більшість дорожніх знаків із їхнім описом та використанням.

Спринт 2. Охоплює випробування барабана або паркування. У цій частині водій повинен керувати автомобілем, контролювати автомобіль, вибрати місце для паркування і, нарешті, вдало припаркувати автомобіль. Робота екзаменатора

полягає в тому, щоб перевірити паркувальні сходинки та вирішити, чи пройде водій іспит чи не пройде.

Спринт 3. Охоплює дорожнє випробування. Ця частина є найскладнішою для водія в реальному іспиті, а також у грі, оскільки екзаменатор буде давати водієві як правильні, так і неправильні накази. У цьому випадку водій повинен проаналізувати накази відповідно до свого досвіду, а потім виконувати правильні, ігноруючи неправильні накази. Після закінчення іспиту екзаменатор повідомить водієві остаточний результат. Особливу роль відіграє діаграма класів агентів. У кожній діаграмі агента визначаються можливості агента. У цьому дослідженні представлено три діаграми агентів, по одній для кожної з трьох фаз спринту. Кожен спринт охоплює один рівень гри.

Спринт 1. На рисунку 3.11 зображено перший рівень гри, який є перевіркою дорожніх знаків. Для цього етапу створюються три агенти: екзаменатор, гравець і особа, яка приймає рішення.

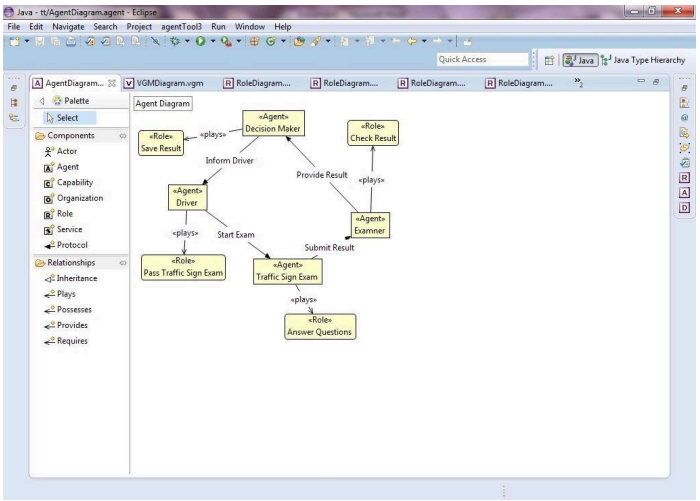


Рисунок 3.11 – Діаграма агента (Спринт 1)

Спринт 2. У цій частині дослідження використовуються попередні агенти з Sprint one. Також створюються додаткові агенти, такі як контроль автомобіля та паркування автомобіля.

Спринт 3. Знову використовуються ті самі агенти з попередніх спринтів, з деякими додатковими ролями.

3.3 Розробка діаграми розгортання та процесів компонування гри

На етапі впровадження використовується діаграма розгортання, як показано на рисунках. Рисунок 3.12 ілюструє схему розгортання для цієї роботи, яка містить перевірку барабана або парку, перевірку дорожніх знаків, перевірку доріг; водій буде вибирати одну частину за раз, послідовність не потрібна.

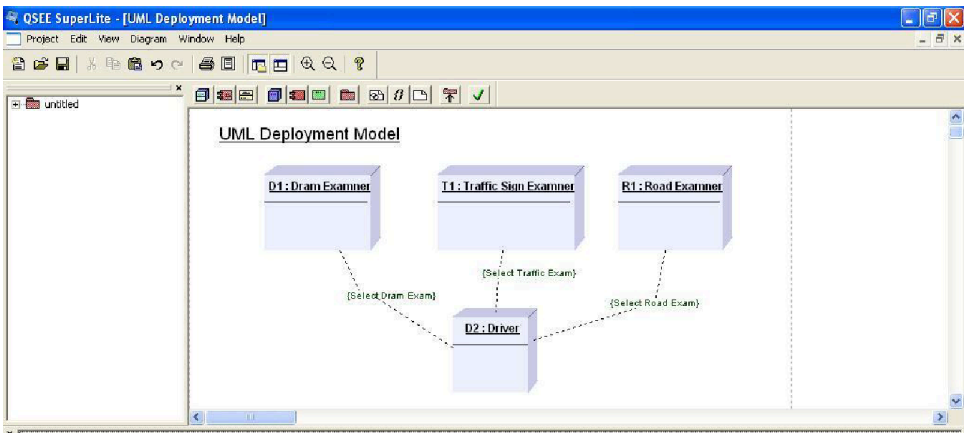


Рисунок 3.12 – Діаграма розгортання

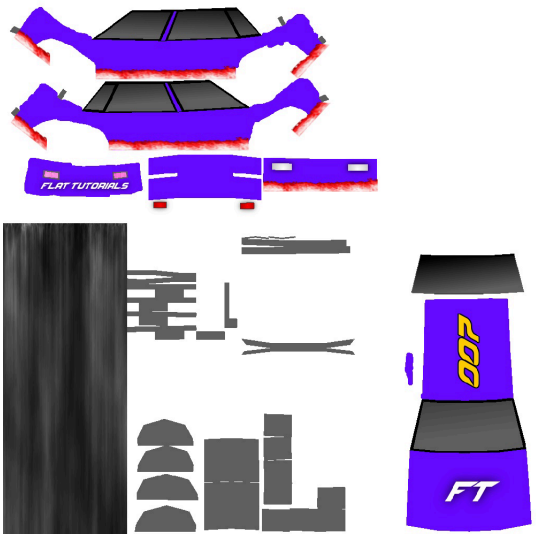


Рисунок 3.13 – Модель об’єкта гри (автомобіля)

Дальше розглянемо фактичну реалізацію з використанням 3Dunity як ігрового двигуна. Початковий крок полягає в імпорті моделі автомобіля в

робочу область 3DUnity, як показано на рисунку 3.13, яка з’явиться в 3Dunity, як на рисунку 3.14.

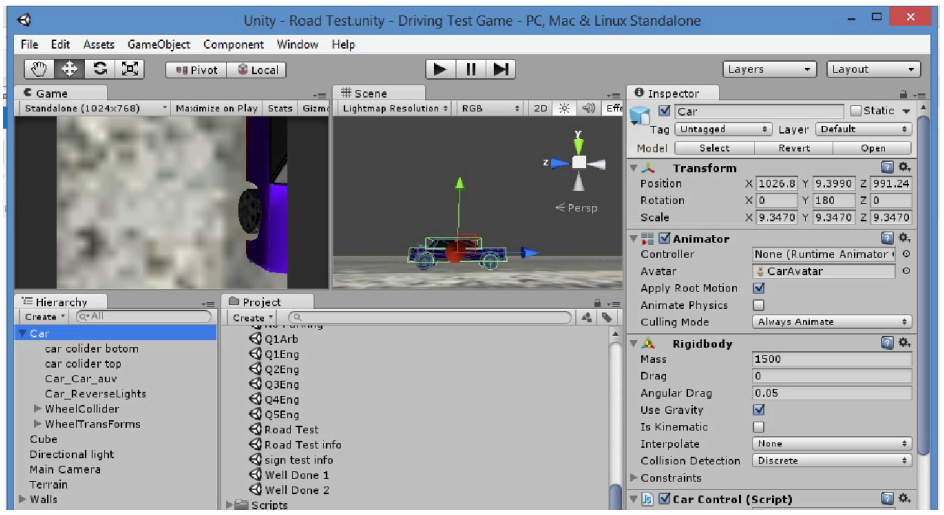


Рисунок 3.14 – Модель об’єкта в Unity

Модель автомобіля є важливою частиною дизайну гри. У цій роботі деякі зображення дорожніх знаків імпортуються для формування частини тесту дорожніх знаків. Кілька сцен обміну повідомленнями використовуються для спілкування з гравцем і пояснення того, як скласти іспит з водіння, що можна побачити на наступних рисунках 3.15 і 3.16.

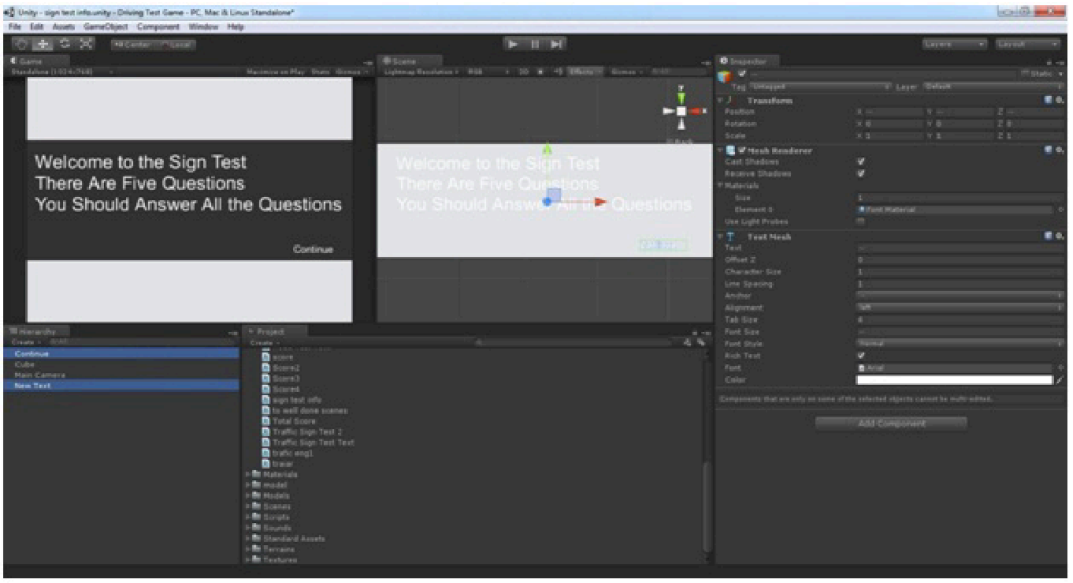


Рисунок 3.15 – Повідомлення 1 з гравцем

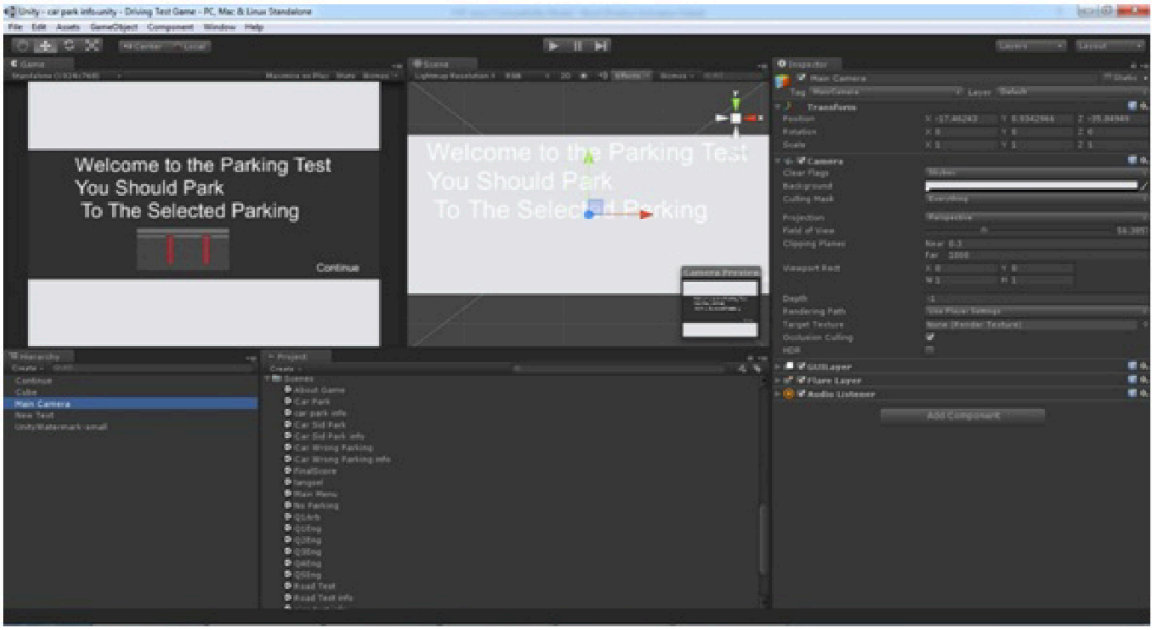


Рисунок 3.16 – Повідомлення 2 з гравцем

На рисунках 3.17 і 3.18 показано результати певних дій, якщо вони правильні чи неправильні.

Ігрова анімація. Щоб досягти відчутного зображення динамічного руху автомобіля, до моделі автомобіля додається Java-скрипт керування автомобілем, який забезпечує рух автомобіля.

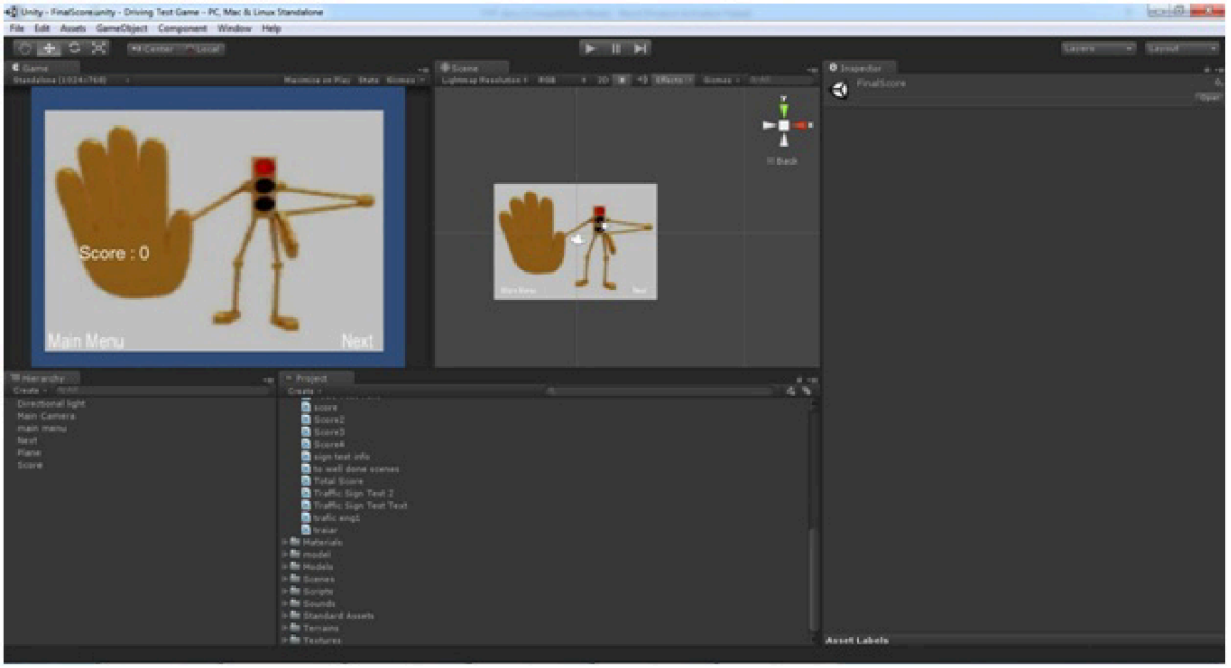


Рисунок 3.17 – Показ результату

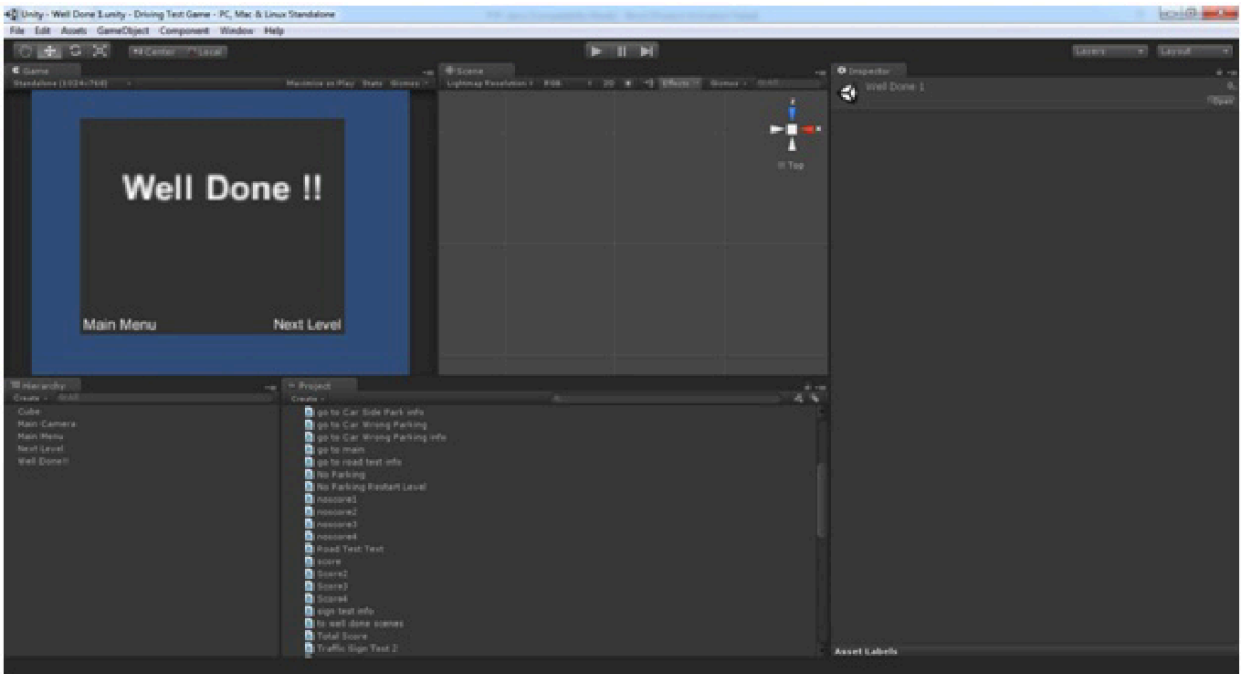


Рисунок 3.18 – Успішність виконаних дій

Для автомобіля використовуються два коробкових коллайдера, а також колісний коллайдер до чотирьох коліс автомобіля. Простий спосіб перевірити зіткнення за допомогою unity – це додати компонент Rigidbody. Тверді тіла — це фізично змодельовані об’єкти, які можна використовувати як позначки. Спосіб використання коллайдерів — позначити їх як тригер. Для цього дослідження це корисно для запуску певної події в грі. Крім того, додано другу камеру для спостереження за рухом автомобіля. Як перший крок, ще одна камера створюється та імпортується за допомогою пакета під назвою Script для використання файлу Smooth follow Java і зв’язування його з нашим автомобілем. Щоб отримати кращий досвід перегляду для гравця, відстань між камерою та автомобілем мінімізовано.

Як описано раніше, 3DUnity має можливість працювати з мовами програмування Java Script, C Sharp і Boo в рамках одного проекту.

Проілюструємо код Java Script. Java Script, як показано на рисунку 3.19, використовується для переходу від однієї сцени до іншої в рамках проекту 3DUnity.

trafic eng1.js

trafic eng1 ▸ IsQuitButtom

```
1  var IsQuitButtom=false;
2
3  function OnMouseEnter () {
4  renderer.material.color=Color.red;
5  }
6
7  function OnMouseExit () {
8  renderer.material.color=Color.white;
9  }
10 function OnMouseUp () {
11 if( IsQuitButtom)
12 {Application.Quit();
13 }
14 else
15 Application.LoadLevel("Q1Eng");
16
17 }
```

Рисунок 3.19 – Код Javascript для переміщення до сцени "Q1Eng"

Щоразу, коли гра переходить між сценами, назва сцени з’являється в останньому рядку рисунка 3.19. На рисунку 3.20 зображено сценарій Java, імпортований із пакета сценаріїв. Файл Java Smooth Follow дозволяє другій камері стежити за автомобілем і забезпечує чіткий огляд для гравця. Сценарій Java показує випадок, коли гравець вибирає правильний варіант під час перевірки дорожніх знаків; його/її остаточний бал збільшиться на одиницю.

SmoothFollow.js

SmoothFollow ▸ LateUpdate

```
1  var target : Transform;
2  // The distance in the x-z plane to the target
3  var distance = 10.0;
4  // the height we want the camera to be above the target
5  var height = 5.0;
6  // How much we
7  var heightDamping = 2.0;
8  var rotationDamping = 3.0;
9  // Place the script in the Camera-Control group in the component menu
10 @script AddComponentMenu("Camera-Control/Smooth Follow")
11 function LateUpdate () {
12         if (!target)    return;
13     // Calculate the current rotation angles
14     wantedRotationAngle = target.eulerAngles.y;
15     wantedHeight = target.position.y + height;
16     currentRotationAngle = transform.eulerAngles.y;
17     currentHeight = transform.position.y;
18     // Damp the rotation around the y-axis
19     currentRotationAngle = Mathf.LerpAngle (currentRotationAngle,
20     wantedRotationAngle, rotationDamping * Time.deltaTime);
21     currentHeight = Mathf.Lerp (currentHeight, wantedHeight, heightDamping * Time.deltaTime);
22     // Convert the angle into a rotation
23     currentRotation = Quaternion.Euler (0, currentRotationAngle, 0);
24         // Set the position of the camera on the x-z plane to:
25     // distance meters behind the target
26     transform.position = target.position;
27     transform.position -= currentRotation * Vector3.forward * distance;
28     // Set the height of the camera
29     transform.position.y = currentHeight;
30         // Always look at the target
31     transform.LookAt (target);
32 }
```

Рисунок 3.20 – Код Javascript для другого наступного руху камери

Сценарій Java імпортується зі сценаріїв з назвою car control, щоб додати керування до автомобіля. Кожен із попередніх сценаріїв Java повинен бути пов’язаний зі сценою або компонентом сцени, щоб працювати ідеально. Зазвичай, якщо вибрано деталі компонента, можна знайти назву пов’язаного файлу Java. У наступному розділі обговорюватимуться фактичні кроки для впровадження гри на основі попереднього аналізу та розділу проектування.

Щоб досягти відчутного зображення динамічного руху автомобіля, до моделі автомобіля додається Java-скрипт керування автомобілем, який забезпечує рух автомобіля. Для автомобіля використовуються два коробкових коллайдера, а також колісний коллайдер до чотирьох коліс автомобіля. Простий спосіб перевірити зіткнення за допомогою unity – це додати компонент Rigidbody. Тверді тіла — це фізично змодельовані об’єкти, які можна використовувати як позначки. Спосіб використання коллайдерів — позначити їх як тригер. Для цього дослідження це корисно для запуску певної події в грі. Крім того, додано другу камеру для спостереження за рухом автомобіля. Як перший крок, ще одна камера створюється та імпортується за допомогою пакета під назвою Script для використання файлу Smooth follow Java і зв’язування його з нашим автомобілем. Щоб отримати кращий досвід перегляду для гравця, відстань між камерою та автомобілем мінімізовано.

Висновки до розділу 3

В даному розділі здійснено вибір методів оцінки побудованої агентно-орієнтованої методології при розробці комп’ютерних ігор, наведені мета та контекст оцінювання. Проведено аналіз фаз розробки комп’ютерної гри та здійснено розробка діаграма розгортання та процесів компонування гри.

ВИСНОВКИ

В кваліфікаційній роботі виконано дослідження моделей та засобів агентно-орієнтованого підходу при розробці комп'ютерних ігор для підвищення їх інтеравктивності, що дозволило зрозуміти важливість використання інтелектуальних агентів у створенні ігрових продуктів.

Визначено роль ігор у визначенні структури і показано, що ігри можна класифікувати та визначити їхню структуру, розглянуто роль штучного інтелекту та ігрових двигунів, зокрема 3D Unity. Застосування агентного підходу дало можливість дослідити зв'язок агентів та агентських платформ, визначено важливість використання мови моделювання агентів (AML) та конкретного інструменту - AgentTool 3.

В роботі використана агентно-орієнтована методологія, оглянуті різні методології розробки програмного забезпечення в іграх, а зокрема агентно-орієнтована методологія (GDMF), з'ясовано її функції, тривалість розробки та важливість командної співпраці. Вивчено методи оцінки побудованої методології, включаючи етапи впровадження, експеримент з дизайну академічної гри та ретельний аналіз отриманих результатів.

Визначено актуальність та перспективи застосування, тобто показано актуальність використання агентно-орієнтованого підходу в розробці комп'ютерних ігор. Результати можуть служити орієнтиром для розробників та дослідників, сприяючи покращенню якості та інтелектуальної складності ігрових продукті.

Отже, можна зробити висновок, що агентно-орієнтований підхід є перспективним інструментом у розробці комп'ютерних ігор та підвищенні їх інтерактивності, дозволяючи створювати продукти, які відзначаються високою ступенем інтелектуальної поведінки та динаміки, що забезпечує більш захоплюючий геймплей та задоволення для гравців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Atomic-Object LLC. URL:http://spin.atomicobject.com/2012/12/29/game-programming-boo-unity-engine-part-1/(дата звернення: 19.03.2024).

2. Blender orgnazation.URL: http://download.blender.org/documentation/pdf(дата звернення: 21.03.2024).

3. Create 3D Gasmes. URL:http://create3dgames.wordpress.com/2012/07/03/unity-vs-udk/(дата звернення: 27.03.2024).

4. GameLab. URL :http:gaminglab.jo/jo/about-2(дата звернення: 29.03.2024).

5. GAMEON'2014.URL: http://www.eurosis.org/cms/index.php?q=node/2844(дата звернення: 10.05.2024).

6. Conkerjo. URL: http://conkerjo.wordpress.com/category/xna/(дата звернення: 14.04.2024).

7. Rpgmaker. URL: http://rpgmaker.net/engines/gm/(дата звернення: 19.04.2024).

8. IBM Innov8.URL: http: //www-01.ibm.com/software/solutions/soa/innov8(дата звернення: 22.04.2024).

9. Intent Media Ltd. URL:http://www.develop-online.net/tools-and-tech/the-top-10-game-engines-no-4-unity-3d/0116475(дата звернення: 24.04.2024).

10. Kansas State University. URL: http://agenttool.cis.ksu.edu/(дата звернення: 24.04.2024).

11. Leed Metropolitan University.URL:http://www.leedsmet.ac.uk/qsee/(дата звернення: 24.04.2024).

12. Oman Driving License.URL:http://omandrivinglicense.blogspot.com/2013/10/blog-post.html?m=1(дата звернення: 30.04.2024).

13. Video Game Industry Statics.URL:http://www.esrb.org(дата звернення: 30.04.2024).

14. Wikipedia.URL: http://en.wikipedia.org/wiki/Game(дата звернення: 30.04.2024).

15. wikipedia. URL:http://en.wikipedia.org/wiki/MicrosoftXNA(дата зверн ення: 30.04.2024).

16. C Abt. Serious games. University Press of America, 1987.

17. P. S. Adler. The evolving object of software development. In Organization, volume 12, 2005.

18. Z Akbari. A Survey Of Agent-oriented Software Engineering Paradigm: Towards Its Industrial Acceptance. Journal of Computer Engineering Research, 1:14{28, 2010.

19. Zohreh Akbari and Ahmad Faraahi. Evaluation Framework for Agent-Oriented Methodolo-gies. In Proceedings of World Academy of Science (WCSET), volume 35, pages 419{424, Paris, France, 2008.

20. Rula Al-Azawi and Aladdin Ayesh. Comparing Agent -Oriented Programming Versus Object-Oriented Programming. In ICIT 2013 The 6th International Conference on Information Technology, pages 24{29, Jordan, 8-10 May 2013. IEEE Jordan Chapter.

21. Rula Al-Azawi, Aladdin Ayesh, and Mohaned Al-Obaidi. Generic evaluation framework for games development methodology. In Third International Conference on Communications and Information Technology ICCIT 2013, pages 55{60, Lebanon, 19-21 June 2013. IEEE Computer Society.

22. Rula Al-Azawi, Aladdin Ayesh, and Mohaned Al-Obaidi. Towards agent-based agile approach for game development methodology. In WCCAIS'2014 World Congress On Computer Applications and Information Systems,, Hammamet, Tunisia, January 2014. IEEE Computer Society.

23. Rula Al-Azawi, Aladdin Ayesh, Ian Kenny, and Khalfan AL-Masrur. Multi agent software engineering (mase) and agile methodology for game development. In 14th Middle Eastern Simulation and Modelling Multiconference, MESM 2014 - 4th GAMEON-ARABIA Conference, GAMEON-ARABIA 2014., In

The 14th Middle Eastern Simulation and Modelling Multiconference (MESM) - The 4th GAMEON-ARABIA Conference., pages 116{122, Muscat, Oman, 2014.

24. Rula Al-Azawi, Aladdin Ayesh, Ian Kenny, and Khalfan AL-Masruria. Analysis of using intelligent technique in games. In 3rd GAMEON-ARABIA'2012 Conference, pages 83{87, Oman,Muscat, December 2012. EUROSIS.

25. Rula Al-Azawi, Aladdin Ayesh, Ian Kenny, and Khalfan AL-Masruria. Towards an aose: Game development methodology. In Distributed Computing and Articial Intelligence, 10th International Conference. Advances in Intelligent and Soft-Computing series of Springer, volume 217, pages 493{501, Selemenca, Spain, May 2013.

26. EF Al-Hashel. A Novel Development Methodology for Cooperative, Distributed Multi-agent Systems. Master's thesis, Faculty of information science and engineering, University of Canberra, Australia, 2010.

27. Apostolos Ampatzoglou and Alexander Chatzigeorgiou. Evaluation Of Object-oriented Design Patterns In Game Development. Information and Software Technology, 49(5):445{ 454, May 2007.

28. Apostolos Ampatzoglou and Ioannis Stamelos. Software Engineering Research For Computer Games: A Systematic Review. Information and Software Technology, 52(9):888{901, 2010.

29. EF Anderson. Playing Smart-articial Intelligence In Computer Games. Proceedings of CON03 Conference on Game Development, ZFX - 3D Entertainment, 2003.

30. Gustavo Andrade, Geber Ramalho, AS Gomes, and Vincent Corruble. Dynamic Game Balancing: An Evaluation Of User Satisfaction. In the 2nd Articial Intelligence and Interactive Digital Entertainment Conference (AIIDE06)., pages 3{8. AAAI Press, 2006.

31. D. Avison and G. Fitzgerald. Information Systems Development: Methodologies, Techniques and Tools. McGraw-Hill, New York, 2nd edition edition, 1995.

32. R. Basseda, F. Taghiyareh, T. Alinaghi, C. Ghoroghi, and A. Moallem. A Framework For Estimation Of Complexity In Agent Oriented Methodologies. In Conference on Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International, pages 645{652, May 2009.

33. Reza Basseda and Tannaz Alinaghi. A Dependency Based Framework For The Evaluation Of Agent Oriented Methodologies. In IEEE International Conference on System of Systems Engineering, SoSE 2009., 2009.

34. E Bethke. Game Development And Production. Wordware Publishing, Inc., 2003.

35. Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems, 8(3):203{236, may 2004.

36. W. Brown Alan and C. Wallnau Kurt. A Framework for Systematic Evaluation of Software Technologies. IEEE Software, 15213(September), 1996.

37. Longbing Cao, Chengqi Z, and Ni Jiarui. Agent services-oriented architectural design of open complex agent systems. In Intelligent Agent Technology, IEEE/WIC/ACM International Conference on. IEEE,, 2005.

38. L Cernuzzi. On The Evaluation Of Agent Oriented Modeling Methods. In the OOPSLA (2002) Workshop on Agent-Oriented Methodologies., Seattle, 2002.

39. Luca Cernuzzi. Prole Based Comparative Analysis For Aose Methodologies Evaluation. Proceedings of the 2008 ACM symposium, pages 60{65, 2008.

40. A. Chella, M. Cossentino, L. Sabatucci, and V. Seidita. Agile PASSI: An Agile Process for Designing Agents. International journal of computer systems science & engineering. Special issue on software, (i):1{6, 2004.

41. Antonio Chella, Viale Scienze, Massimo Cossentino, Luca Sabatucci, Valeria Seidita, Alte Prestazioni, and Consiglio Nazionale. From Passi To Agile Passi : Tailoring A Design Process To Meet New Needs.

42. Chia-En Lin, Krishna M. Kavi and Frederick T. Sheldon and Thomas E. Potok. A Methodology To Evaluate Agent Oriented Software Engineering

Techniques. In 40th Annual Hawaii International Conference on System Sciences, HICSS 2007, pages 1{20, Island of Hawaii,USA, 2007. IEEE Computer Society.

43. Mark Claypool and Kajal Claypool. Latency Can Kill : Precision and Deadline in Online Games. pages 215{222, 2010.

44. G. Costikyan. I have no words & i must design. In In Interactive Fantasy., number 2, 1994.

45. Chris Crawford. Chris Crawford on game design. New Riders, 2003.

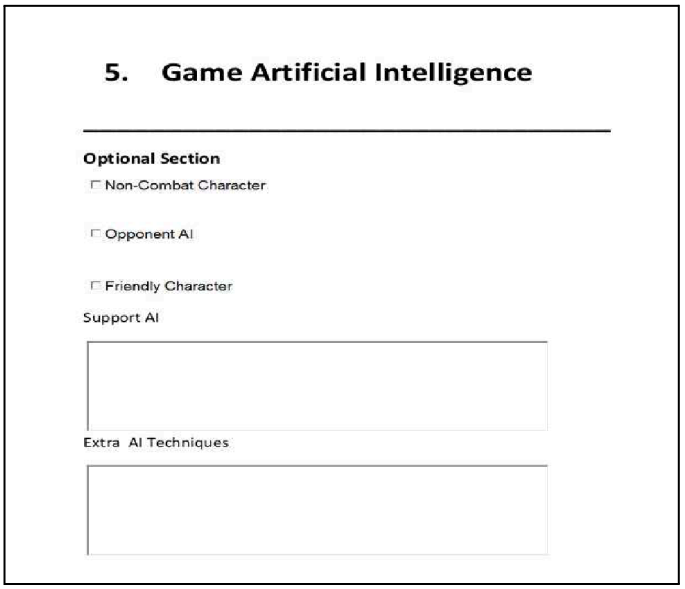
46. Khanh Dam. Evaluating And Comparing Agent-oriented Software Engineering Method-ologies. PhD thesis, School of Computer Science and Information Technology, RMIT University, Australia., 2003.

47. Michael Dam, Khanh and Winiko. Comparing Agent-oriented Methodologies. In Pro-ceedings of the Fifth International Bi-Conference Workshop on Agent-Oriented Information Systems (AAMAS). Lecture Notes in Computer Science; Springer, volume 3030, pages 78{93, Melbourn, Australia., 2004. Springer Berlin / Heidelberg.

ДОДАТКИ

Додаток А

GDD Template



5. Game Artificial Intelligence

Optional Section

☐ Non-Combat Character

☐ Opponent AI

☐ Friendly Character

Support AI

Extra AI Techniques

score.js

score ▸ Score

```
1 static var Score : int ;
2 Score=0;
3  var IsQuitButtom=false;
4  var guiScore : GUIText;
5 function Start () {
6 //guiScore.text = "Score: 0";
7 }
8 function OnMouseEnter () {
9 renderer.material.color=Color.red;
10 }
11
12 function OnMouseExit () {
13 renderer.material.color=Color.white;
14 }
15 function OnMouseUp () {
16 if( IsQuitButtom)
17 {Application.Quit();
18 }
19 else{
20 Score+=1;
21 //guiScore.text = "Score:" + Score;
22 //Debug.Log(Score);
23 Application.LoadLevel("Q2Eng");
24
25
26 }
27 }
```

Лістинг А.1 – Код Java Script для правильного вибору Traffic Sign Test

noscore2.js

oscore2 ▸ F IsQuitButtom

```
1 var IsQuitButtom=false;
2
3 function OnMouseEnter () {
4 renderer.material.color=Color.red;
5 }
6
7 function OnMouseExit () {
8 renderer.material.color=Color.white;
9 }
10 function OnMouseUp () {
11 if( IsQuitButtom)
12 {Application.Quit();
13 }
14 else
15 Application.LoadLevel("Q3Eng");
16
17 }
```

Лістинг А.2 - Код Java Script для неправильного вибору Traffic Sign Test

Total Score.js

otal Score ▸ M Awake

```
1
2
3 public var guiScore : GUIText;
4 function Start () {
5 if (score.Score<0)
6 guiScore.text = "Score: 0";
7 else
8 guiScore.text = "Score:" + score.Score;
9 }
```

Лістинг А.3 - Код Java Script для остаточної оцінки Traffic Sign Test

CarControl.js

CarControl ▸ Awake

```
#pragma strict

var wheelFL : WheelCollider;
var wheelFR : WheelCollider;
var wheelRL : WheelCollider;
var wheelRR : WheelCollider;

var maxTorque : float = 20;

function Start () {
rigidbody.centerOfMass.y = -10.0;
}

function FixedUpdate () {
rigidbody.AddForce(-transform.up * rigidbody.velocity.magnitude);
wheelRR.motorTorque = maxTorque * Input.GetAxis("Vertical");
wheelRL.motorTorque = maxTorque * Input.GetAxis("Vertical");

wheelFL.steerAngle = 30 * Input.GetAxis ("Horizontal");
wheelFR.steerAngle = 30 * Input.GetAxis ("Horizontal");
}
```

Лістинг А.4 - Код Javascript для керування автомобілем

метадані

Заголовок

Розробка агентно-орієнтованої системи для підвищення інтерактивності в комп'ютерних іграх

Автор
Штогрин Володимир

Науковий керівник / Експерт
кандидат технічних наук Олег Пашкевич

підрозділ
King Danylo University

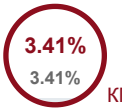
Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

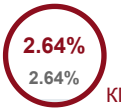
Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	12
Парафрази (SmartMarks)	a	9

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КП 2

25
Довжина фрази для коефіцієнта подібності 2

13785
Кількість слів

103446
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	https://core.ac.uk/download/pdf/83535137.pdf	158	1.15 %
2	https://core.ac.uk/download/pdf/83535137.pdf	127	0.92 %
3	http://repository.ukd.edu.ua/bitstream/handle/123456789/386/%D0%9A%D0%92%D0%90%D0%9B%D0%86%D0%A4%D0%86%D0%9A%D0%90%D0%A6%D0%86%D0%98%CC%86%D0%9D%D0%90%20%D0%A0%D0%9E%D0%91%D0%9E%D0%A2%D0%90.pdf?sequence=1	43	0.31 %
4	http://repository.ukd.edu.ua/bitstream/handle/123456789/386/%D0%9A%D0%92%D0%90%D0%9B%D0%86%D0%A4%D0%86%D0%9A%D0%90%D0%A6%D0%86%D0%98%CC%86%D0%9D%D0%90%20%D0%A0%D0%9E%D0%91%D0%9E%D0%A2%D0%90.pdf?sequence=1	36	0.26 %