ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

(повне найменування назва факультету)

Кафедра програмування

(повна назва кафедри)

ДИПЛОМНА РОБОТА

на тему: «Дослідження технологій для масової розробки оцифрованих настільних ігор»

Студента IV курсу, групи ПМІ-41 ,

напряму підготовки *122 Комп'ютерні науки*

Левицького Ростислава Остаповича

(прізвище та ініціали)

Керівник *доцент Клакович Леся Миронівна*

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____

Кількість балів: _____ Оцінка: ECTS _____

Львів – 2023

Зміст

Вступ……..……..2
Огляд існуючих платформ для
реалізації……...……..……4
Деталі використаної технології…….…6
Приклад карткової гри……..10
Хід Реалізації……..……...14
Висновки……...…..21
Використані джерела……...…..23

Вступ

Історично склалось так, що найпопулярніша комп'ютерна гра - це розкладання пасьянсу “Solitaire”. Створена в 1990р. вона увійшла до списку стандартних ігор Windows 98, Millenium, 2000, XP, 7. Починаючи з Windows 8 гра доступна в списку безкоштовних додатків. Отож гра була на піку популярності з 1990 до 2012, 22 роки. Карткові ігри почали здавати свої позиції в списку популярних комп'ютерних ігор.

Сьогодні тенденції знову змінюються, настільні карткові ігри повертаються в цілком новій інтерпретації. Суть настільної гри тепер не тільки у наборі правил та можливостей, але й сюжет. Тобто настільні ігри беруть за основу казки, фільми, аніме, чи просто відтворюють побут певної країни у якомусь історичному проміжку часу. Таким чином асортимент ігор став настільки широким, що задовольнив навіть самі вишукані смаки. Є ігри для дітей шкільного віку, підлітків, дорослих та навіть для сімей. Це дозволило друзям чи родичам весело проводити своє дозвілля, спеціально для ігрових партій створено так звані анти-кафе, де клієнти орендують столик для гри, а не страви з меню. Родини змогли весело проводити вечори на кухні під веселу чи інтелектуальну настолку.

В 2019 все змінилось з появою корона-вірусу. З введенням пандемії кафе закрили, людей обмежили від прямого контакту, відповідно, настільні ігри попали в опалу. Наше життя стає віртуальним: віддалена робота, дистанційне навчання, побачення онлайн.

Основою до таких ігор став не стандартний набір гральних карт, а індивідуальний набір карт, кубиків, ігрових полів, тощо. Це ускладнило їх адаптації у віртуальному світі. Складність задачі нас лякає, проте не зупинить.

Тому актуальною є проблема оцифрування настільних ігор, на вирішення якої спрямована ця робота. Суть цієї роботи є дослідження існуючих

технологій та платформ для реалізації сучасних настільних карткових ігор на комп'ютері, а також розширити карткові настільні ігри багатокористувацьким режимом. Таким чином друзі зможуть “розкинути карти” і під пандемію у ліжку і під завивання сирен в укритті. Метою дипломної роботи було дослідити існуючі технології для розробки ігор, проаналізувати можливості різних технологій, як вивчених в університеті, так і самостійно опрацьованих в інтернеті. Після дослідження необхідно було реалізувати карткову гру Бенг для демонстрації методів та можливостей обраної технології, та визначити основні принципи для розробки настільної гри як комп’ютерної, щоб в майбутньому мати можливість масово створювати різноманітні ігри.

Огляд існуючих платформ

Для розробки карткових ігор існує низка популярних середовищ розробки, які можна використовувати:

1. **Unity**[7][8]: Unity є одним з найпопулярніших інструментів для розробки ігор, включаючи карткові ігри. Він надає потужні інструменти для створення, редагування та анімації карток, а також можливості програмування логіки гри. Unity підтримує кілька мов програмування, таких як C# і JavaScript, і має велику спільноту розробників, де можна знайти багато корисних ресурсів і плагінів.
   Недоліки: Складність реалізації, опис поведінки об'єктів реалізується кодом
2. **Unreal Engine**[1]: Unreal Engine є іншим потужним середовищем розробки, яке можна використовувати для створення карткових ігор. Він має великий перелік функціоналу і можливості для редагування карток, створення анімацій, програмування і логіки гри. Unreal Engine використовує мову програмування C++ і також має активну спільноту розробників.
   Недоліки[2]: 3D - орієнтована гра, не для логічних настільних ігор
3. **GameMaker**[3][4]: GameMaker є інструментом розробки ігор з простим інтерфейсом, що дозволяє швидко створювати карткові ігри без необхідності в програмуванні. Він має вбудовані функції для створення і редагування карток, а також можливості для програмування логіки гри на власній мові сценаріїв (GML). GameMaker також має активну спільноту розробників і велику кількість документації.
   Недоліки[4]: немає адаптації для карткових ігор, платний, лише для Windows і MacOs

4. **Tabletop Simulator**[5][6]: Tabletop Simulator - це програмне забезпечення для симуляції настільних ігор, включаючи карткові ігри. Воно надає зручні інструменти для створення та редагування карток, а також можливість відтворювати їх в віртуальному середовищі.
   Недоліки: лише для ПК на базі OS Windows

Обрано було Unity з низки причин:

- безкоштовна версія включає достатньо функціоналу для реалізації настільної гри
- повна кросплатформність
- є певні наробки для карткових настільних ігор
- поведінкові скрипти реалізуються мовою C#

Деталі використаної технології

Unity Engine[9] (або просто Unity) - це ігровий двигун, розроблений компанією Unity Technologies, що використовується для створення ігор, візуальних ефектів, симуляцій та інших інтерактивних додатків. Unity надає розробникам потужні інструменти для створення багатоплатформних проектів, що працюють на різних операційних системах, таких як Windows, macOS, Android, iOS, Xbox, PlayStation та багато інших.

У Unity Prefab[10] (або **префаб**) є важливим поняттям. Він представляє собою шаблон або контейнер, який містить повторно використовувану колекцію об'єктів, компонентів та налаштувань. Prefab дозволяє створювати інстанси об'єктів на основі цього шаблону, що спрощує процес створення та керування багатьма екземплярами одного типу об'єкта.
Ось деякі основні характеристики та переваги використання Prefabs:

1. Перевикористання: Prefabs дозволяють створювати одну версію об'єкта та використовувати її в багатьох місцях у проекті. Якщо ви змінюєте Prefab, ці зміни автоматично відображаються у всіх його екземплярах, що зроблять оновлення та зміни в проекті більш зручними та ефективними.
2. Ієрархія об'єктів: Prefabs можуть містити ієрархію об'єктів. Це означає, що ви можете створити Prefab, що складається з кількох дочірніх об'єктів, які будуть утворювати єдину структуру. Це дозволяє легко керувати всією структурою та взаємодіяти з нею як з єдиною сутністю.
3. Зміна в реальному часі: Одним з переваг використання Prefabs є здатність до змін в реальному часі. Ви можете редагувати Prefab, а зміни автоматично застосовуватимуться до всіх екземплярів, що

базуються на цьому Prefab. Це дозволяє швидко експериментувати з дизайном та параметрами об'єктів.

4. Налаштування компонентів: Prefabs також дозволяють налаштовувати компоненти об'єктів. Ви можете визначити певні значення параметрів компонентів у Prefab, і ці значення будуть успадковуватися всіма екземплярами, що базуються на Prefab. Це дає зручну можливість зберігати та керувати настройками об'єктів в одному місці.

Використання Prefabs є потужним інструментом в Unity, що спрощує розробку, керування та підтримку багатьох об'єктів у вашому проекті. Вони допомагають забезпечити структурованість, перевикористання коду та зберігання налаштувань, що зробить вашу роботу більш продуктивною та ефективною.

У контексті Unity, сцена (**Scene**)[11] - це простір, в якому розміщуються та взаємодіють об'єкти вашої гри або додатку. Сцена визначає весь віртуальний світ вашої гри, включаючи об'єкти, їх положення, освітлення, камери, звуки тощо. Вона виступає як контейнер для розміщення та організації всіх елементів вашої гри.

Ось декілька ключових аспектів сцен в Unity:

1. Об'єкти сцени: Сцена містить набір об'єктів, таких як персонажі, об'єкти оточення, ефекти, інтерфейсні елементи тощо. Ви можете створювати, редагувати, переміщати та взаємодіяти з цими об'єктами в рамках сцени.
2. Компоненти об'єктів: Кожен об'єкт у сцені може мати різні компоненти, такі як графічні, фізичні, скриптові компоненти тощо.

Ці компоненти визначають поведінку, властивості та функціональність об'єктів у грі.

3. Камери: Сцена може містити одну або більше камер, які визначають видиму область гри. Камери встановлюють положення та орієнтацію з якої гравець або глядач спостерігають за сценою.
4. Освітлення: Сцена може містити різні джерела світла, такі як точкові, напрямлені або площинні світла, що визначають освітлення об'єктів у грі та створюють атмосферу.
5. Додаткові ресурси: У сцені можуть використовуватися і інші ресурси, такі як звукові ефекти, частинки, фонові зображення, музика та інші елементи, які допомагають створити бажану геймплейну атмосферу.

Сцени в Unity можна створювати, редагувати та організовувати засобами Unity Editor. Ви можете перемикатися між різними сценами в процесі гри, щоб створювати різні рівні, меню, екрани завантаження тощо. Керування сценами дозволяє вам структурувати та управляти контентом вашої гри відповідно до потреб вашого проекту.

Скрипти в Unity - це фрагменти програмного коду, написані на мовах програмування, таких як C# або JavaScript (UnityScript), які виконуються в середовищі Unity. Скрипти використовуються для керування поведінкою об'єктів, реалізації логіки гри, взаємодії зі звуком, графікою та іншими аспектами гри. Скрипти дозволяють вам розширювати функціональність Unity та створювати інтерактивні ігри та додатки.

Ось кілька прикладів того, що можна зробити за допомогою скриптів в Unity:

1. Керування рухом об'єктів: Скрипти можуть керувати рухом об'єктів у просторі, змінювати їх положення, поворот та масштабування, використовуючи компонент Transform.
2. Реалізація логіки гри: Скрипти можуть включати логіку гри, таку як керування персонажами, обробка колізій та зіткнень, розрахунок очків або здоров'я гравця, штучний інтелект ворогів та багато іншого.
3. Створення користувацького інтерфейсу: Скрипти дозволяють створювати інтерфейсні елементи, такі як кнопки, текстові поля, слайдери та інші, і реагувати на дії користувача.
4. Обробка введення: Скрипти можуть обробляти введення від користувача, таке як клавіатура, миша або сенсорні пристрої, і реагувати на нього, наприклад, керувати рухом персонажа або змінювати камеру.
5. Анімація: Скрипти можуть керувати анімацією об'єктів, відтворювати, зупиняти або змінювати параметри анімаційних кліпів.
6. Мережеву взаємодію: Скрипти можуть дозволяти багатокористувацьку гру через мережу, обмінюватись даними між гравцями та організовувати комунікацію.

Приклад карткової гри

Бенг (Bang!)[13] - це популярна настільна карткова гра з вестерн-тематикою. У грі гравці виконують ролі шерифа, помічника шерифа, безжалісних бандитів та хитрих ренегатів. Метою гри є вижити та виконати свої рольові цілі.

Гра Бенг здобула популярність завдяки своєму захоплюючому геймплею та відмінному відтворенню вестерн-атмосфери. Вона підходить для групи від 4 до 7 гравців і зазвичай триває близько 30-45 хвилин. Крім базової версії, гра має також додаткові розширення та варіанти правил, що додають більше можливостей і різноманітності до гри.

В моїй електронній версії гри я реалізую версію для 4-ох гравців, оскільки така кількість на даний момент є найоптимальнішою і найпопулярнішою серед реальних гравців.

У грі Бенг використовуються різні типи карт, такі як картки персонажів, картки зброї, картки дій та картки рольових цілей. Гравці виконують ходи, розігруючи картки та взаємодіючи з іншими гравцями. Взаємодія може включати атаки, захисти, лікування та інші дії. Гра пропонує багато стратегічних можливостей та може залежати від ролей та взаємодії гравців.

Картки ролей[14]**:**

- Картки ролей(див. Рис. 1) розподіляються серед гравців випадковим чином в початковому етапі гри. Кожен гравець знає свою роль, але інші гравці можуть не знати, хто вони є. Ролі можуть бути такими, як шериф, бандит, ренегат і т.д. Кожна роль має свої особливі правила і цілі.

Рис. 1

Картки зброї[14]**:**

- Картки зброї(див. Рис. 2) використовуються лише для атаки, має властивість зміни дальності для нанесення удару.

Рис. 2

Картки дії[14]**:**

- Картки взаємодії(див. Рис. 3) використовуються гравцями для атаки, захисту або спеціальних дій проти інших гравців. Ці картки можуть дозволяти вистрілити в іншого гравця, вилікувати себе, отримати додаткові і т.д.

Рис. 3

Комбінації карток та їх взаємодія можуть створювати різноманітні ситуації та стратегії гри.

Хід реалізації

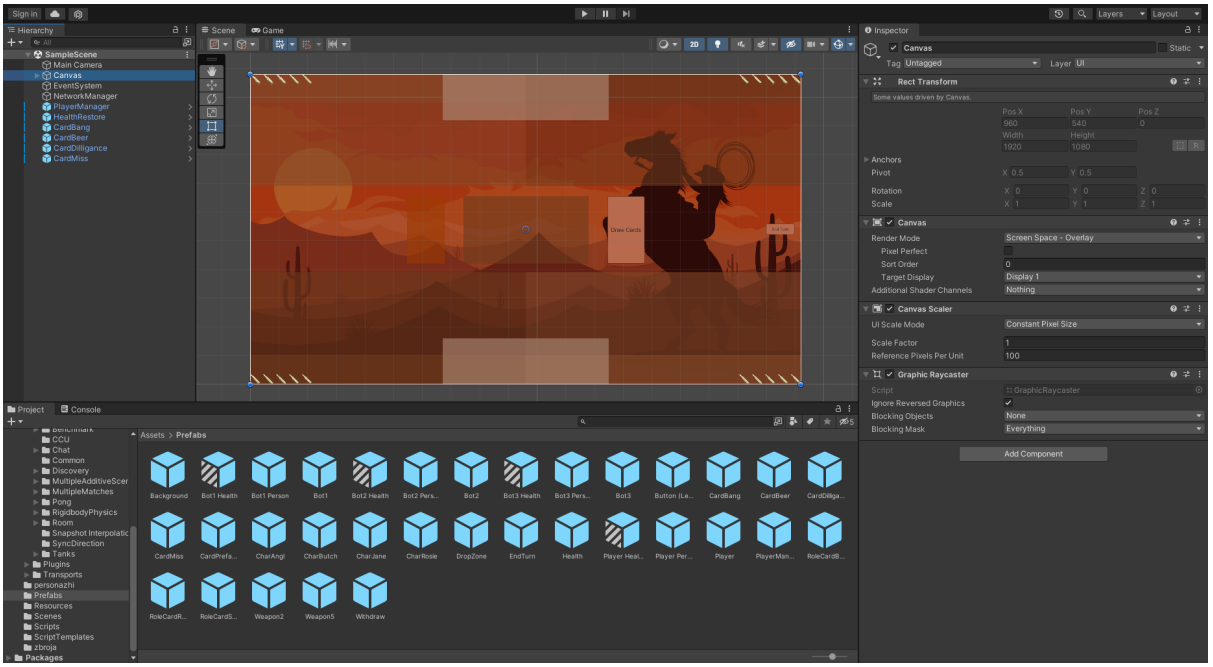


рис.4

На Рис. 4 видно реалізоване ігрове поле, на якому є великі слоти для гравців по кутах екрану, місце для розіграшу карт, кнопки для отримання карт та скидання розіграних у відбій. Але все по порядку.

Поле гравця:

Рис. 5

Посередині Рис. 5 є динамічне місце для розміщення карт, зліва знизу є масив куль, як маркерів життя гравця, справа знизу є поле для розміщення ролі, персонажу та зброї. Сама панель містить розбиття сіткою з орієнтацією на верхній центр. На Рис. 6 продемонстровано вигляд усіх компонент для поля гравця.

.

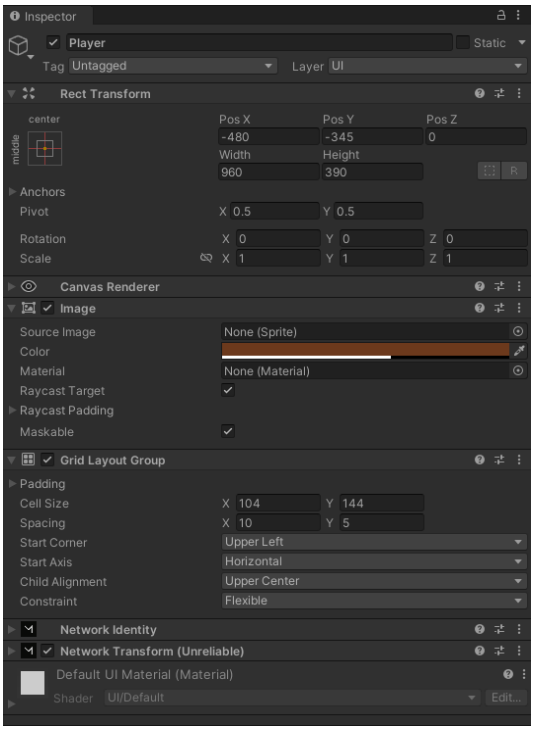


Рис. 6

Всі карти роздаються гравцям випадковим чином, а саме в поля гравців надсилаються ініціалізовані префаби карт.

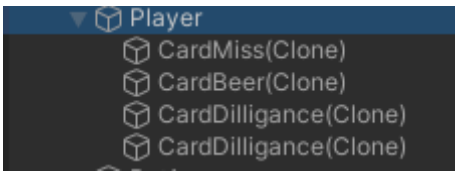


Рис. 7 приклад вигляду гравця з картами в дереві об'єктів

Ігрова карта

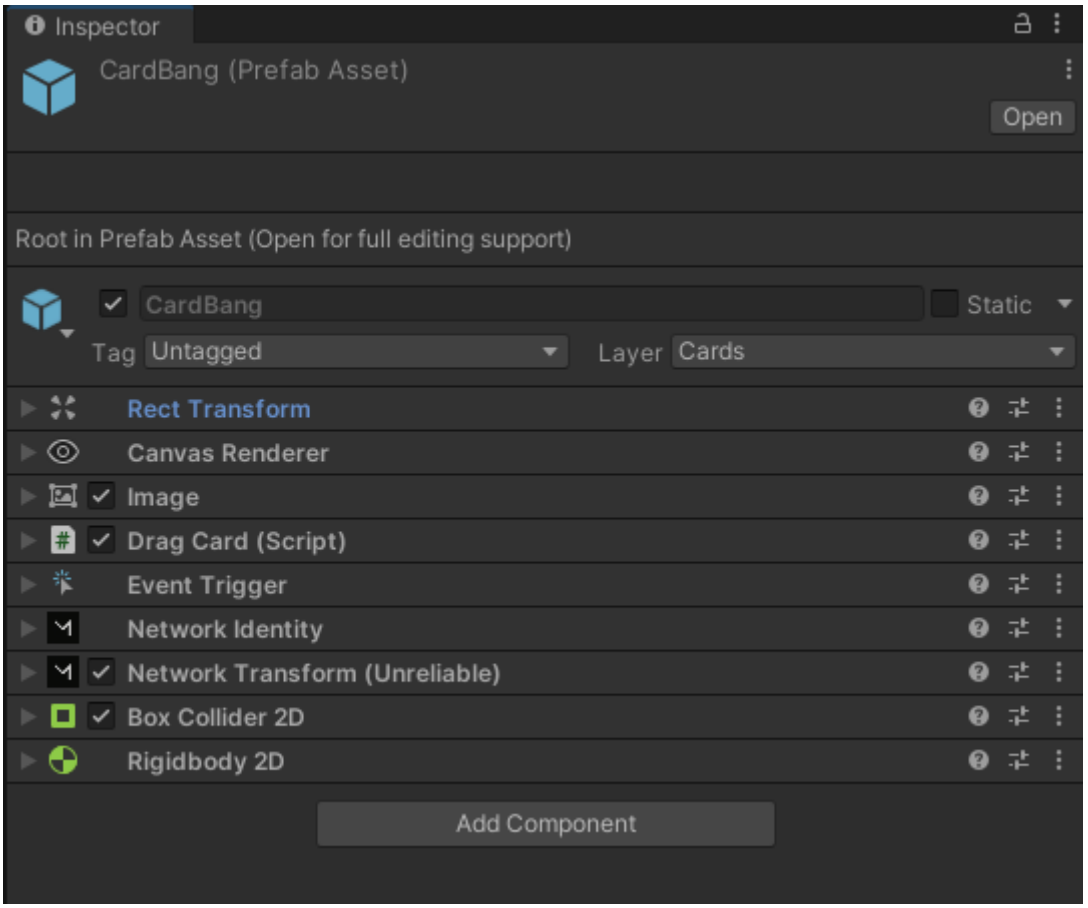


Рис. 8

Для кожної карти реалізовані такі компоненти як Image (див. Рис. 8) для зображення відсканованих і оброблених карт, скрипт в якому описані переміщення карти та їх імплементація, Event Trigger для реакції на певні події, такі як початок та кінець переміщення, відпускання, наведення мишею та інші. BoxCollider 2D та Rigidbody 2D (див. Рис. 8) для створення фізичних показників

об’єкту та можливості оцінити зіткнення з іншими об’єктами. Та Network Identity з Network Transform для реалізації мережевої гри.

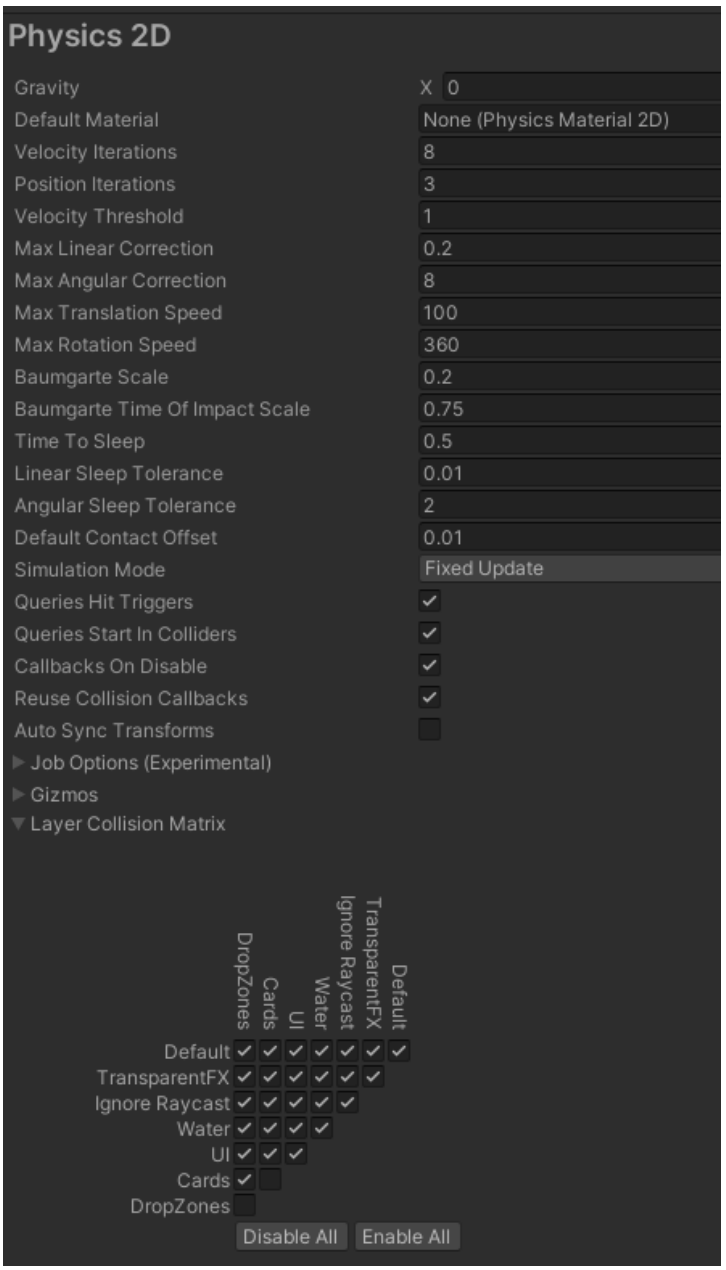


Рис. 9

Для правильної роботи зіткнення карт з полем для скиду налаштував слої для зіткнення карт, а саме не даю можливості картам стикатись одне з одною(див. Рис. 9).

Player Manager

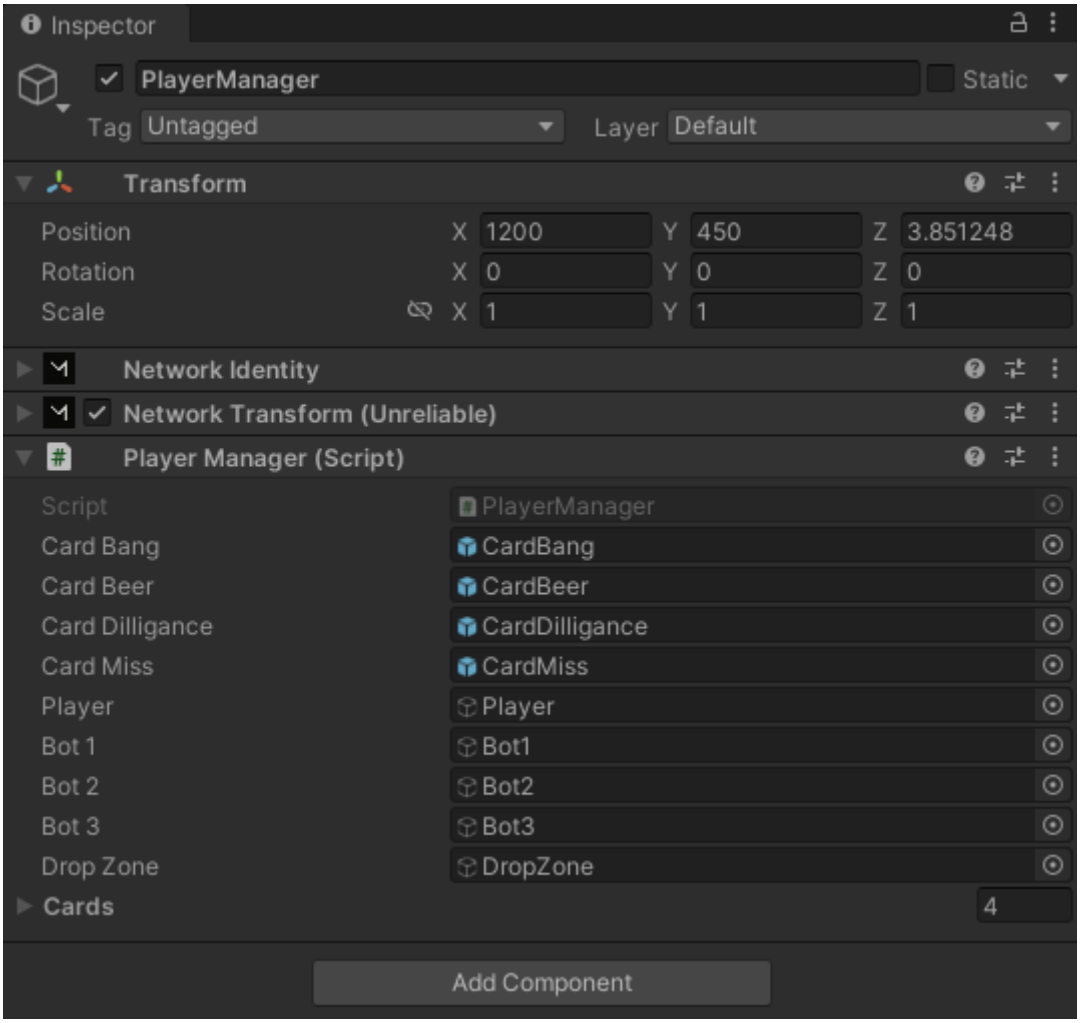
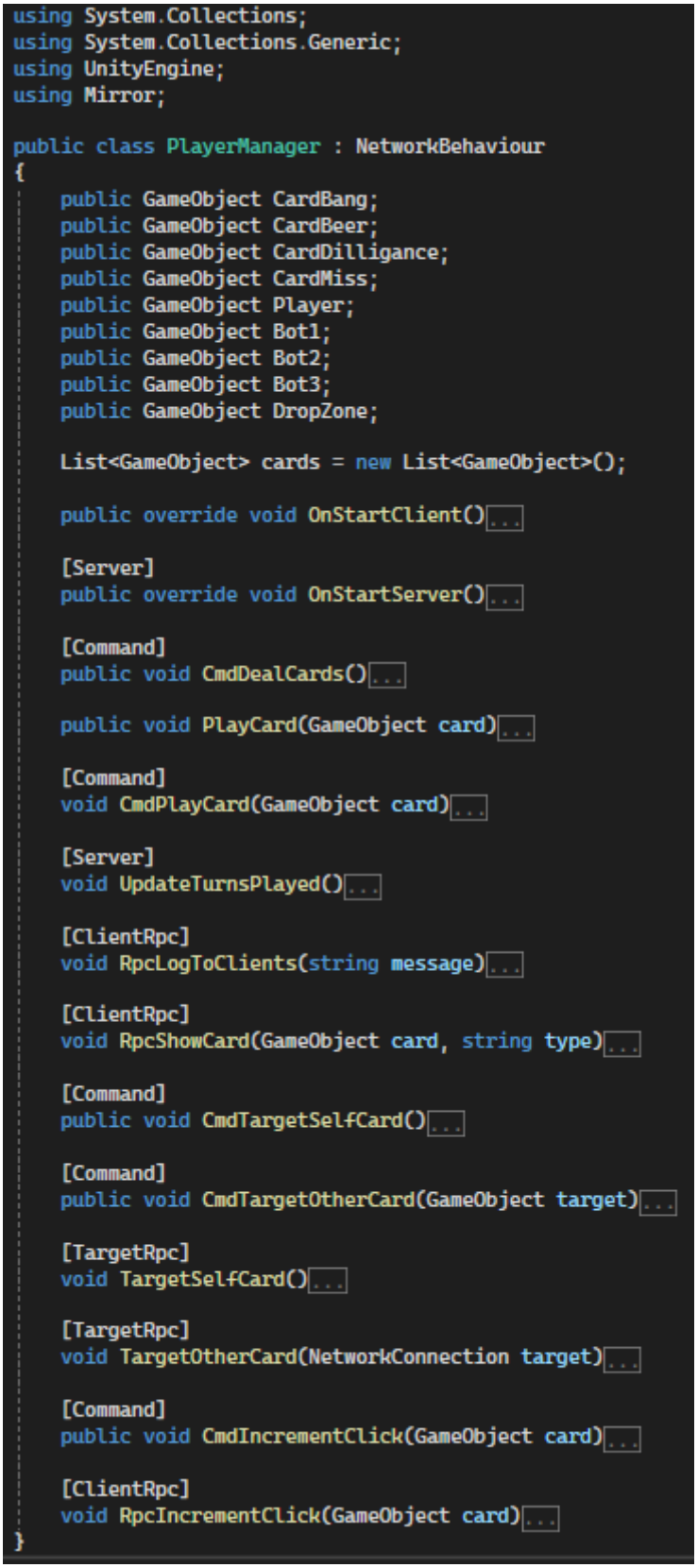


Рис. 10

Цей об’єкт є дуже важливим для реалізації мультиплеера (див. Рис. 10). В ньому є компонента зі скриптом Player Manager, в якому реалізовані основні команди для гравця. Також тут є Network Identity та Network Transform, які будуть ідентифікувати кожного гравця під час гри, при чому як для гравця-сервера, так і для гравця-клієнта цей об’єкт працює однаково, тому достатньо розробити префаб одного гравця для гри, де всі гравці рівноправні.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mirror;

public class PlayerManager : NetworkBehaviour
{
    public GameObject CardBang;
    public GameObject CardBeer;
    public GameObject CardDilligance;
    public GameObject CardMiss;
    public GameObject Player;
    public GameObject Bot1;
    public GameObject Bot2;
    public GameObject Bot3;
    public GameObject DropZone;

    List<GameObject> cards = new List<GameObject>();

    public override void OnStartClient()...

    [Server]
    public override void OnStartServer()...

    [Command]
    public void CmdDealCards()...

    public void PlayCard(GameObject card)...

    [Command]
    void CmdPlayCard(GameObject card)...

    [Server]
    void UpdateTurnsPlayed()...

    [ClientRpc]
    void RpcLogToClients(string message)...

    [ClientRpc]
    void RpcShowCard(GameObject card, string type)...

    [Command]
    public void CmdTargetSelfCard()...

    [Command]
    public void CmdTargetOtherCard(GameObject target)...

    [TargetRpc]
    void TargetSelfCard()...

    [TargetRpc]
    void TargetOtherCard(NetworkConnection target)...

    [Command]
    public void CmdIncrementClick(GameObject card)...

    [ClientRpc]
    void RpcIncrementClick(GameObject card)...
}
```

Рис. 11

Ось згорнутий скрипт для Player Manager (див. Рис. 11). Тег Server ініціалізується при запуску сервера, Command відповідає за команди, які виконує сервер на вимогу гравців, та ClientRpc як реакції сервера клієнтам. Наявність такого Player Manager спрощує створення та написання інших скриптів, для прикладу скрипт (див. Рис. 12) для роздачі карт.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mirror;


public class drawCard : NetworkBehaviour
{
    public PlayerManager PlayerManager;


    public void OnClick()
    {
        NetworkIdentity networkIdentity = NetworkClient.connection.identity;
        PlayerManager = networkIdentity.GetComponent<PlayerManager>();
        PlayerManager.CmdDealCards();
    }
}
```

Рис. 12

Також реалізував метод CardFlipper (див. Рис. 13) для того, щоб приховати карти гравця від інших гравців.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class CardFlipper : MonoBehaviour
{
    public Sprite CardFront;
    public Sprite CardBack;

    public void Flip()
    {
        Sprite currentSprite = gameObject.GetComponent<Image>().sprite;

        if (currentSprite == CardFront)
        {
            gameObject.GetComponent<Image>().sprite = CardBack;
        }
        else
        {
            gameObject.GetComponent<Image>().sprite = CardFront;
        }
    }
}
```

Рис. 13

Висновок

У дипломній роботі було розглянуто задачу оцифрування настільних ігор, визначення основних об'єктів, таких як карти, поле гравця, ігрове поле, умов і правил гри для реалізації подібного проекту, проаналізовано низку технологій для вирішення поставленої задачі, і вибрано Unity як найкращий технологія для розробки настільної гри “Бенг” через наступні переваги:

1. Кросплатформність: Unity Engine дозволяє створювати ігри, які можуть працювати на різних платформах, включаючи ПК, консолі, мобільні пристрої і навіть веб.
2. Легка розробка: Unity має дружній інтерфейс користувача і потужні інструменти для створення ігор. Ви можете використовувати графічний редактор для створення власних карт, анімацій, ефектів інтерфейсу та інших важливих елементів карткової гри.
3. Гнучкість: Unity дозволяє створювати карткові ігри з різними техніками та стилями. Ви можете налаштувати правила гри, механіку переміщення карт, бойову систему та інші аспекти гри відповідно до ваших уявлень.
4. Спільнота розробників: Unity має велику спільноту розробників, яка активно ділиться знаннями, підказками та ресурсами. Ви можете знайти безліч уроків, документації, форумів і груп, де можна задати питання та отримати допомогу.

Відповідно завдяки цій технології та використанню фреймоворку Mirror для налаштування мережевої гри було реалізовано оцифрований варіант настільної гри “Бенг” з базовими можливостями отримання карт, їх розіграш і скидання, приховування від противника, тощо. В проекті метод реалізації надає можливість для простого розширення проекту, наприклад для розширення гри

до версії з доповненням, чи для обмеження певних правил гри для спрощення чи ускладнення ігрового процесу.

Список використаної літератури

1. https://www.unrealengine.com/en-US/unreal-engine-5
2. https://www.udemy.com/course/complete-guide-to-unreal-engine-5/?utm_source=adwords&utm_medium=udemyads&utm_campaign=DSA_Catchall_la.EN_cc.ROW&utm_content=deal4584&utm_term=_._ag_88010211481_._ad_535397282061_._kw__._de_c_._dm__._pl__._ti_dsa-45211625058_._li_1012859_._pd__._&matchtype=&gclid=Cj0KCQjw7PCjBhDwARIsANo7CgmcyeBlfa_eJwVpqV0V9M0cCSnmPEanaiokUcU1HlpTSqpWHMno6AYaAtCaEALw_wcB
3. https://manual.yoyogames.com/#t=Content.htm
4. https://gamemaker.io/en/tutorials
5. https://steamcommunity.com/sharedfiles/filedetails/?id=2217838904
6. https://www.youtube.com/watch?v=xcPK9EiyGWQ&list=PL24Zj3DBZCt_vVZRaGzXz5ng7ldpCxOuo
7. https://learn.unity.com/
8. https://assetstore.unity.com/packages/tools/network/mirror-129321
9. https://en.wikipedia.org/wiki/Unity_(game_engine)
10. https://docs.unity3d.com/Manual/Prefabs.html
11. https://docs.unity3d.com/Manual/CreatingScenes.html
12. https://docs.unity3d.com/Manual/Unity2D.html
13. https://en.wikipedia.org/wiki/Bang!_(card_game)
14. https://www.ultraboardgames.com/bang/game-rules.php