МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА КАРТОЧНОЙ КОМПЬЮТЕРНОЙ ИГРЫ НА ДВИЖКЕ UNITY С ИСПОЛЬЗОВАНИЕМ СЕТЕВОГО API MIRROR

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Левитана Олега Олеговича

Научный руководитель
зав. каф. тех. прогр., к. ф.-м. н. _____ И. А. Батраева

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Видеоигры в современном обществе являются неотъемлемой частью жизни огромного числа людей по всему миру. Люди используют их как вид отдыха после длинного рабочего дня и даже как способ заработка. Каждый день миллионы человек включают свои персональные компьютеры для того, чтобы провести пару часов в другом измерении и отвлечься от своих жизненных проблем хотя бы ненадолго. В последнее время, большой популярностью пользуются коллекционные карточные игры, где игроки в разных режимах могут сразиться друг с другом, используя своё знание игровой теории, находчивость и внимательность.

Актуальность данной работы заключается в том, что жанр коллекционных карточных игр сейчас находится на своём пике, но ещё не все возможные варианты игры были реализованы и проверены. В рамках данной работы разработан проект, использующий нетривиальные механики, которые могут оказаться интересны пользователям.

Движок `Unity` для данной работы был выбран как одна из самых часто используемых систем в современной игровой индустрии. По данным сайта `Game Developer`, в 2021 году $49,48\%$ всех платных игр, вышедших в крупнейшем интернет-магазине компьютерных игр `Steam`, разработаны на `Unity`. А в сфере мобильных игр этот показатель уже давно превысил 50%. [1]

Целью данной работы было разработать коллекционную карточную компьютерную игру на движке `Unity` с использованием в качестве `API` для работы с сервером системы `Mirror`.

Для достижения заданной цели, были поставлены следующие задачи:

1. изучить основы работы с игровым движком `Unity`;
2. изучить основы работы с сетевым `API Mirror`;
3. разработать концепт для коллекционной карточной игры;
4. разработать программу, реализующую этот концепт.

В данной работе для реализации практической части используются следующие программные средства:

— игровой движок Unity;

— система для работы с сетью Mirror.

1 Основные принципы разработки игры

1.1 О движке Unity

Приложение Unity представляет собой профессиональный игровой движок, который используется в создании видеоигр для различных платформ. Это инструмент, которым ежедневно пользуются опытные разработчики, а также один из наиболее доступных инструментов для новичков. Для разработчиков Unity предлагает возможность добавить в свой проект обширный функционал, включающий в себя моделирование физических сред, карты нормалей, преграждение окружающего света в экранном пространстве, динамические тени. [2] Рабочий процесс в Unity чётко привязан к среде разработки, что позволяет программисту не разрываться между множеством окон и программ - через эту среду можно получить доступ к любой сцене, любому префабу и любому файлу с кодом программы.

1.2 О сетевом API Mirror

В Unity для разработки мультиплеерных проектов доступны множество различных фреймворков, платных и бесплатных. В рамках данной работы была использована бесплатная система Mirror, представляющая собой доработанную версию классической для движка технологии UNet, поддержка которой уже прекратилась. Mirror может работать как связка клиента и сервера, так и как NoGUI-сервер. [3]

1.3 Принципы разработки коллекционной карточной игры

1.3.1 Коллекционные карточные игры и самые известные их примеры

Коллекционные карточные игры – это не все игры, где игрок может собирать какие-то карты. Ими считаются только те, в которых игроки должны собирать из своих карт колоды и играть ими. Практически все ККИ - игры с неполной информацией: игроки держат свои карты на руке и не знают, какие карты находятся на руке оппонента, что избавляет от необходимости рассчитывать варианты на несколько ходов вперед, характерной для игр с полной информацией, вроде шахмат. Для ККИ характерно не давать игроку доступ ко всей колоде сразу, а только к её части, которая раскрывается по ходу партии случайным образом. Основной вид боя для такого вида игр - дуэль между двумя игроками. В качестве примеров таких проектов можно привести Magic: the Gathering и Hearthstone.

1.3.2 Метагейм и ротация

Метагейм – это, в принципе своём, игра над игрой, в которой игроки принимают участие при создании и выборе своих колод. Иногда метагейм определяют как то, как играют другие. Ни одна колода не может иметь хорошие шансы против всех прочих колод: это нарушило бы баланс игры и снизило качество игрового процесса. В продуманном метагейме всегда присутствуют нетранзитивные цепочки, вроде следующей: колода А побеждает Б, которая побеждает колоду В, а колода В побеждает колоду А, только различных типов колод очень много, и цепочки могут быть очень запутанными.

Общая сила колод постоянно растёт. Разумеется, нельзя делать карты из каждого нового выпуска сильнее прежних: при этом старые карты быстро теряют актуальность, игрокам это постепенно надоедает, и они уходят из игры. Но даже в том случае, если средняя сила карт новых выпусков не превосходит силу старых, общая сила колод постепенно растет, поскольку у игроков появляется больше вариантов.

1.3.3 Проблемы разработки коллекционной карточной игры

Дизайнеры всех сетевых ККИ вынуждены решать одни и те же задачи, и качество их решения полностью определяет, насколько хорошей получится игра, и будет ли она приносить серьёзный доход. Рассмотрим основные проблемы, на которые стоит обратить внимание при разработке:

— Высокий уровень абстракции.
— Элитарность.
— Спортивность и казуальность.

1.3.4 Принципы создания карт

При создании карт необходимо руководствоваться следующими принципами:

— Картам необходима синергия – удачное сочетание карт.
— Необходимо обращать внимание на комбинацию карт.

2 Разработка игры

2.1 Описание идеи проекта

Рабочее название проекта - `Family Affairs`. Игра представляет собой противостояние трёх сторон - мафии, полиции и обычных людей. В игре доступен режим игры вдвоём, где каждому игроку на выбор предлагается на выбор одна из трёх заранее заготовленных колод из небольшого числа карт и героев, привязанных к этим колодам. Сама игра проходит в привычном режиме - на первом ходу каждому игроку выдаётся 5 карт и 1 монета, которая позволяет сыграть одну из полученных карт. Затем, каждый ход игрок получает в руку ещё 1 карту, а количество доступных для использования монет увеличивается на 1 по сравнению с предыдущим ходом. Каждая карта способна нанести урон персонажу противника. Партия считается оконченной, когда здоровье одного из игроков достигает 0. [4]

2.2 Префаб карт и связанные с ним процессы

2.2.1 Структура префаба

Формируется структура игрового объекта - карты. Происходит объяснение каждого входящего в префаб объекта, создаются структуры, необходимые для корректной работы коллизии.

2.2.2 Класс Card.cs

Здесь объясняется формирование базового для карт класса. [5] Дано описание каждому полю и каждой входящей в класс функции, которые формируют объект. [6]

2.2.3 Класс CardDatabase.cs

Происходит создание класса, выполняющего роль базы данных используемых в игре карт, в качестве объектов, которые содержат в себе данные об этих картах используются списки.

2.2.4 Класс CardFlipper.cs

Формируется класс, отвечающий за переворачивание карты для того, чтобы противник не мог видеть набор карт игрока во время партии.

2.2.5 Класс CardZoom.cs

Для того, чтобы игрок мог прочитать описание дополнительных эффектов розыгрыша карты и другие её характеристики проводится реализация класса, который позволяет при наведении курсора на карту создавать её копию, размеры которой в 2 раза больше оригинальной. [7]

2.2.6 Класс DragDrop.cs

Описывается составление класса, позволяющего игроку перемещать карту по экрану с помощью мыши.

2.2.7 Класс ThisCard.cs

Формируется класс, чьё назначение - определять визуальные характеристики карты. Здесь находятся процедуры для выставления цвета карты, её здоровья и атаки, цены её розыгрыша, её изображения и так далее.

2.3 Префаб героя и связанные с ним процессы

2.3.1 Структура префаба

Формируется структура игрового объекта - героя. Составляющие этого объекта почти полностью повторяют префаб карты, отличие лишь в том, что у героя нет атаки и поля с ценой розыгрыша, здесь вместо них есть поле для активации специальной способности персонажа.

2.3.2 Класс Hero.cs

Описывается базовый класс для героев, повторяющий по своим принципам аналогичный класс для карт.

2.3.3 Класс HeroDatabase.cs

Происходит описание класса, повторяющего принципы `CardDatabase.cs`, но для героев.

2.3.4 Класс HeroZoom.cs

Логика этого класса полностью повторяет аналогичный класс для карт `CardZoom.cs`. Единственные различия - позиция копии героя и используемая функция для установки данных героя.

2.3.5 Классы HeroPower.cs и HeroPowerDatabase.cs

Происходит описание классов, аналогичных подобным для героев и карт.

2.3.6 Класс ThisHero.cs

Формируется класс, отвечающие за регулирование визуальных аспектов отображения героев на экране игрока, повторяет собой `ThisCard.cs`.

2.4 Первая страница меню

2.4.1 Описание структуры сцены

Описываются объекты, формирующие первую страницу меню.

2.4.2 Описание кода работы страницы

Происходит описание содержимого файлов, содержащих код, осуществляющий работу основных элементов страницы - кнопок начала игры, вызова меню опций и завершения игры.

2.4.3 Описание кода работы с меню опций

Объясняются принципы работы меню опций. Описываются объекты, участвующие в регулировании звука. [8] Объясняется процесс возвращения к работе с основными элементами первой страницы меню. [9] Также здесь описывается принципы работы аудио в проекте. [10] Отдельное внимание уделяется процессу сохранения настроек пользователя. [11]

2.5 Вторая страница меню

2.5.1 Описание структуры сцены

Описываются объекты, формирующие вторую страницу меню.

2.5.2 Описание кода работы страницы

Происходит объяснение принципов работы двух кнопок страницы - перехода к странице для ввода адреса сервера и возврата к первой странице меню. [12]

2.6 Игровое лобби

2.6.1 Описание структуры сцены

Описываются объекты, формирующие игровое лобби.

2.6.2 Описание кода работы страницы

Происходит объяснение принципов работы страницы, включающих в себя сохранение указанного пользователем адреса сервера, и перехода на вторую страницу меню. [13]

2.7 Основная сцена игры

2.7.1 Описание структуры сцены

Описываются объекты, формирующие основную игровую сцену. [14] Это включает в себя процесс загрузки необходимых объектов при подключении игрока. [15]

2.7.2 Описание кода работы страницы

Объясняются процессы, связанные с стартом игры и работой основных кнопок на странице. [16]

2.8 Класс PlayerManager.cs и работа основных процессов игры

2.8.1 Переменные и OnStartClient()

Происходит подробное описание используемых в классе переменных и функции, работа которой осуществляется при присоединении клиента к серверу.

2.8.2 Вспомогательные функции

Описывается работа неосновных функций класса, включающих в себя обновление числа отыгранных ходов, отображение сообщений от сервера и обновления состояния игрока. [17]

2.8.3 Функции выбора персонажа

Объясняется процесс выбора персонажа и перехода к самой игре. [18]

2.8.4 Функции выдачи карт игроку

Объясняются процессы, отвечающие за выдачу карту игроку в начале каждого его хода.

2.8.5 Функции розыгрыша карт

Происходит описание функций, отвечающих за розыгрыш карт и вызова их специальных эффектов. [19]

2.8.6 Функции, отвечающие за завершение хода игрока

Описываются функции, отвечающие за переход хода, включающие в себя отключение кнопки перехода хода, обновление количества доступных игроку и его противнику монет и автоматический процесс боя.

2.9 Функции, отвечающие за использование игроком способности его персонажа

В этой части описываются функции, реализованные для того, чтобы игрок мог использовать специальные способности его персонажа.

2.10 Работа чата

Описание создания интерфейса чата и работы с чатом.

2.11 Пример работы проекта

Для тестирования работы проекта был использован `ParrelSync`. `ParrelSync` - плагин, позволяющий создавать копии проекта в `Unity` и открывать его одновременно в нескольких окнах. [20] Также, для тестирование к объекту `NetworkManager` в сцене `GameScene2Players` был добавлен `NetworkManagerHUD` - интерфейс, позволяющий вручную запустить сервер и клиентов. [21] В качестве второго игрока выступит дополнительное окно редактора `Unity`.

Далее происходит описание нескольких примеров корректной работы проекта.

ЗАКЛЮЧЕНИЕ

Таким образом, в рамках выполненной работы была разработан режим для коллекционной карточной игры, доступный для двух игроков. В ходе работы были выполнены следующие задачи:

1. изучены основы работы с игровым движком `Unity`;
2. изучены основы работы с сетевым `API Mirror`;
3. разработан концепт для коллекционной карточной игры;
4. разработана программа, реализующая этот концепт.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Разработка игр на Unity: почему этот движок так популярен, кто работает с ним и сколько зарабатывает [Электронный ресурс]. — URL: https://netology.ru/blog/01-2022-unity-development (Дата обращения 21.05.2022). Загл. с экр. Яз. рус.

2 *Хокинг, Д.* Unity в действии. Мультиплатформенная разработка на C♯ / Д. Хокинг. — 2-е изд. — С.: ООО Издательство «Питер», 2019. — С. 16–33.

3 Mirror [Электронный ресурс]. — URL: https://mirror-networking.gitbook.io/docs/ (Дата обращения 21.05.2022). Загл. с экр. Яз. англ.

4 *Бонд, Д.* Unity и CC♯. Геймдев от идеи до реализации / Д. Бонд. — 2-е изд. — С.: ООО Издательство «Питер», 2019. — С. 170–197.

5 Serializable [Электронный ресурс]. — URL: https://docs.unity3d.com/ScriptReference/Serializable.html (Дата обращения 22.05.2022). Загл. с экр. Яз. англ.

6 ScriptableObject [Электронный ресурс]. — URL: https://docs.unity3d.com/Manual/class-ScriptableObject.html (Дата обращения 22.05.2022). Загл. с экр. Яз. англ.

7 EventTrigger [Электронный ресурс]. — URL: https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/script-EventTrigger.html (Дата обращения 22.05.2022). Загл. с экр. Яз. англ.

8 Slider [Электронный ресурс]. — URL: https://docs.unity3d.com/2018.2/Documentation/ScriptReference/UI.Slider.html (Дата обращения 22.05.2022). Загл. с экр. Яз. англ.

9 GameObject [Электронный ресурс]. — URL: https://docs.unity3d.com/ScriptReference/GameObject.html (Дата обращения 22.05.2022). Загл. с экр. Яз. англ.

10 AudioMixer [Электронный ресурс]. — URL: https://docs.unity3d.com/ScriptReference/Audio.AudioMixer.html (Дата обращения 22.05.2022). Загл. с экр. Яз. англ.

11 PlayerPrefs [Электронный ресурс]. — URL: https://docs.unity3d.com/ScriptReference/PlayerPrefs.html (Дата обращения 22.05.2022). Загл. с экр. Яз. англ.

12 SceneManager.LoadScene [Электронный ресурс]. — URL: https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.LoadScene.html (Дата обращения 22.05.2022). Загл. с экр. Яз. англ.

13 Implement data persistence between scenes [Электронный ресурс]. — URL: https://learn.unity.com/tutorial/implement-data-persistence-between-scenes# (Дата обращения 22.05.2022). Загл. с экр. Яз. англ.

14 Префабы (Prefabs) [Электронный ресурс]. — URL: https://docs.unity3d.com/ru/530/Manual/Prefabs.html (Дата обращения 22.05.2022). Загл. с экр. Яз. рус.

15 NetworkManager [Электронный ресурс]. — URL: https://mirror-networking.gitbook.io/docs/components/network-manager (Дата обращения 22.05.2022). Загл. с экр. Яз. англ.

16 NetworkBehaviour Callbacks [Электронный ресурс]. — URL: https://mirror-networking.gitbook.io/docs/guides/communications/networkbehaviour-callbacks?q=OnStartServer%28%29 (Дата обращения 22.05.2022). Загл. с экр. Яз. англ.

17 Remote Actions [Электронный ресурс]. — URL: https://mirror-networking.gitbook.io/docs/guides/communications/remote-actions#clientrpc-calls (Дата обращения 22.05.2022). Загл. с экр. Яз. англ.

18 Deprecations [Электронный ресурс]. — URL: https://mirror-networking.gitbook.io/docs/general/deprecations#networkclient (Дата обращения 22.05.2022). Загл. с экр. Яз. англ.

19 SyncVars [Электронный ресурс]. — URL: https://mirror-networking.gitbook.io/docs/guides/synchronization/syncvars (Дата обращения 22.05.2022). Загл. с экр. Яз. англ.

20 ParrelSync [Электронный ресурс]. — URL: https://github.com/VeriorPies/ParrelSync (Дата обращения 23.05.2022). Загл. с экр. Яз. англ.

21 Network Manager HUD [Электронный ресурс]. — URL: https://mirror-networking.gitbook.io/docs/components/network-manager-hud (Дата обращения 23.05.2022). Загл. с экр. Яз. англ.