Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

Розробка 3d гри з використанням машинного навчання

Текстова частина до курсової роботи

за спеціальністю «Прикладна математика» 113

Керівник курсової роботи

Старший викладач

Борозений С.О.

_____

(підпис)

“____” _____ 2020 р.

Виконав студент 3-го курсу

Статейко А.О.

“____” _____ 2020 р.

Київ 2020

Календарний план виконання курсової роботи

Тема: Розробка ігор з використанням game engine

Календарний план виконання роботи:

Студент Статейко А.О. _____

Керівник Борозений С.О. _____

“_____”_____ 2020 р.

Зміст

Мета та план роботи

Наразі постає питання, щодо того, який треба використовувати game engine при написанні своєї гри. В цій роботі я хочу розібрати галузі, в яких кожний з двигунів має свою перевагу. Також буде обран один з них для написання своєї гри.

Гра - 2D платформер з бойовою системою. В грі планується реалізувати State Machine та динамічну генерацію рівней. Гравець буде мати змогу збирати зілля, тим самим зав’язуючи на цьому прогресс гри. Також буде реалізація ворожого AI, котрий буде вміти патрулювати деяку ділянку та переслідувати гравця на цій ділянці.

Порівняльний аналіз двох найпопулярніших game engine

Наразі є два найпопулярніших game engine, а саме Unity та Unreal Engine. Обидва ці двигуни мають деякі переваги перед іншим, але слід розібратися який саме треба використовувати при яких ситуаціях.

Давайте розберемо галузі, в котрих Unreal Engine має перевагу перед Unity:

- 3D ігри. Має велику перевагу з точки зору графічної реалізації, хоча Unity теж має широкий спектр можливостей для розробки
- Multiplayer. Unreal Engine має інтегровану підтримку багатокористувацького режиму навідміну від Unity

А тепер в котрих Unity має перевагу перед Unreal Engine:

- 2D ігри. Unity підходить краще, завдяки великій кількості розширень і вбудованих інструментів.
- Віртуальна реальність. Unity містить більшу кількість плагінів.
- Мобільні ігри. Unity - лідер в цій галузі.

Підбір game engine для роботи

Unity завжди вважався найкращим двигуном для незалежних розробників та галузі мобільних ігор. Але великих проектів розроблених за допомогою Unity було не так багато, наприклад Rust, Subnautica, Hearthstone чи Firewatch. В AAA-галузі, Unreal Engine користується більшим попитом.

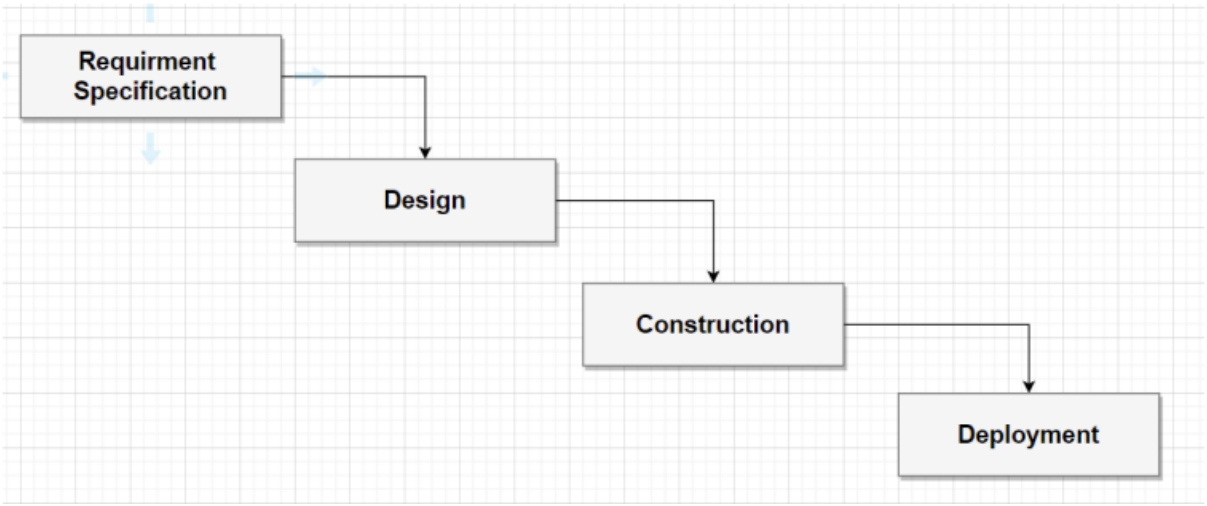
Найважливіших фактором в підборі двигуна є користувацька спільнота та наявність навчальних матеріалів, форумів, ресурсів тощо. І наразі Unity дуже багато може надати з цього, що спричиняє ще більший розвиток суспільства.

Зважаючи на сказане вище, було прийняте рішення використовувати для розробки саме Unity як інструмент для розробки.

Процесс розробки

Модель процесу

Оскільки даний проект розробляється більшою мірою в навчальному плані, це передбачає послідовну реалізацію та можливість переосмислення реалізації того чи іншого функціоналу. Тому доволі не раціонально використовувати будь-яку інкрементальну, або еволюційну модель. Попри те вимоги є чітко визначеними та стабільними, тому в процесі розробки практично відсутні непередбачувані витрати. Здійснено розробку реалістичного та безпечного аналізу вимог, що передбачає надійний початковий інтерфейс та відкидає можливість повернення до фази проектування. Враховуючи усі фактори вибір зупинився на використанні методології WaterFall. Усі модифікації котрі будуть стосуватися рефакторингу, тестування та налагодження інтеграції будуть реалізовані під час фази “Construction‘, як показано на малюнку.



Організація команди

В ході процесу розробки було вирішено, що структура команди не буде підпорядковуватися суворим вимогам. Спілкування в команді

здійснюється горизонтально. Модель взаємодії передбачає внесення посильного вкладу в розробку проекту, а також мотивує реалізовувати інноваційні та перспективні ідеї.

Користувацька інтерактивність та обмеження контролю

Даний продукт предназначенный для використання людьми, котрі мають базові навички використання комп'ютера, або мобільного дивайсу. Взаємодія користувача з інтерфейсом програми залежатиме від середовища користувача. Тому було реалізовано дві моделі взаємодії, перша з яких відбувається за допомогою клавіатури та мишки. Друга буде адаптована під мобільні застосунки, де взаємодія буде реалізована через екран смартфона.

Геймплей

В ході розробки гри, було прийнято рішення змоделювати середовище, котре б реалізовувало в собі реалістичну фізику, та природну взаємодію між компонентами. Користувач повинен зіштовхнутися з непередбачуваними ситуаціями та обрати оптимальне для себе рішення, правильно збалансовувати можливості свого персонажа та реалізовувати їх в необхідних ситуаціях. Відповідна реалізація потребує чіткої взаємодії програмної та графічної складової застосунку.

Сцени

Різноманітність та взаємодія 2d графіки, моделей, текстур, звукового супроводу та анімації, наближать ігрове середовище більш до реального, проте в свою чергу це відобразиться на системних вимогах до апаратного забезпечення

Клієнт-Сервер

Передбачено взаємодію між користувачем та сервером, котра буде оптимізовуватися для покращення швидкодії. Здійснено захищений

зв’язок між користувачем та сервером, задля безпеки і конфіденційності даних користувача, а також безпеки та цілісності системи.

Дослідження

В процесі планування розробки архітектури було проведено дослідження, внаслідок якого обрали оптимальні та найбільш гнучкі інструменти та середовища розробки.

Графіка

Розробка графічної частини, один з найвідповідальніших процесів, оскільки це можна вважати одним з ключових факторів того чи користувач зверне увагу на гру, і найважливіше продовжить її грати. Тому вибір зупинився на одному з кращих рішень Unity 2d. В даному середовищі графічні об’єкти відомі як спрайти. Спрайти - це, по суті, лише стандартні текстури, але існують спеціальні методи комбінування та управління текстурами спрайтів для ефективності та зручності під час розробки. Unity надає вбудований редактор спрайтів, який дозволяє витягувати спрайт-графіку з більшого зображення. Це дозволяє редагувати декілька компонентних зображень в одній текстурі у редакторі зображень.

UnityEngine

Unity можна використовувати для розробки практично будь-якої елементарної гри або інтерактивного контенту в реальному часі. Unity підтримує скрипти на C#, створені відповідно до одного з двох основних підходів: традиційним та широко розповсюдженим об'єктно-орієнтованим підходом та інформаційно-орієнтованим підходом, який тепер також підтримується в Unity у окремих випадках завдяки високопродуктивному багатопоточному стеку інформаційно-орієнтованої технології (DOTS).

Об'єкти GameObject і компоненти в Unity

Вся інтерактивність і ігровий процес в Unity будуються на основі трьох фундаментальних блоків: об'єкти GameObject, компоненти і змінні.

Будь-який об'єкт в грі є GameObject, будь то персонажі, джерела світла, спецефекти, декорації і все інше.

Компоненти

Ігрові об'єкти самі по собі не мають ніякого поведінки. Для того, щоб об'єкт почав працювати, ігровому об'єкту потрібні різні атрибути, що додаються за допомогою компонентів.

Компоненти (Component) визначають поведінку ігрових об'єктів, до яких вони прикріплені, і керують ними. Простий приклад - створення джерела світла, що включає прикріплення компонента Light до GameObject. Таким же прикладом може бути додавання компонента Rigidbody до об'єкта, щоб він міг падати.

Компоненти мають ряд властивостей або змінних, які можна налаштувати у вікні Inspector редактора Unity і / або за допомогою скрипта.

Інформаційно-орієнтований підхід з DOTS

Традиційна модель «ігровий об'єкт - компонент» добре працює і сьогодні, оскільки вона проста як для програмістів, так і інших користувачів, а також зручна для створення інтуїтивних інтерфейсів. Якщо додати компонент Rigidbody до об'єкта GameObject - він почне падати, додасте компонент Light - GameObject почне випромінювати світло. Все інше також підпорядковується цій простій логіці.

При цьому система компонентів створена на основі об'єктно-орієнтованої платформи, що створює складності для

розробників при роботі з кешем і пам'яттю розвивається обладнання.

Компоненти і ігрові об'єкти відносяться до «важких об'єктах C ++». Всі об'єкти GameObject мають ім'я. Їх компоненти являють собою оболонки для C # поверх компонентів на C ++. Це спрощує роботу з ними, але може впливати на продуктивність, якщо вони будуть зберігатися в пам'яті без явної структури. Об'єкт C # може перебувати на будь-якій ділянці пам'яті. Об'єкт C ++ також може знаходитися в будь-якій ділянці пам'яті. Угруповання і послідовне розміщення об'єктів в пам'яті відсутні.

При кожному завантаженні в центральний процесор для обробки об'єкт доводиться збирати по частинах з різних ділянок пам'яті. Це може сильно уповільнити завантаження, а оптимізація потребують багато зусиль. Для вирішення цих проблем, було вдосконалено базові системи Unity на основі високопродуктивного, многопоточного стека інформаційно-орієнтованих технологій або DOTS (в даний час в статусі попередньою версією).

DOTS дозволяє грі ефективно використовувати всі можливості новітніх багатоядерних процесорів. Стек складається з наступних компонентів:

- система завдань C # для ефективного виконання коду на багатопотокових системах;
- Entity Component System (ECS) для розробки високопродуктивного коду за замовчуванням;
- компілятор Burst для компіляції скриптів в оптимізований нативний код.
- ECS - це нова система компонентів в складі DOTS;

Звуки

Для роботи зі звуком використовувалося розширення від Unity, AudioSource. Система може відтворювати аудіокліп будь-якого типу і може бути налаштований на відтворення у форматі 2D, 3D або як суміш (SpatialBlend). Аудіо можна розподілити між динаміками і перетворити між 3D та 2D. Це можна контролювати на відстані за допомогою кривих спаду. Крім того, якщо слухач знаходиться в одній або декількох зонах реверберації, до джерела застосовується реверберація. Індивідуальні фільтри можна застосувати до кожного джерела звуку для ще більш насиченого звуку.

PlayFab

PlayFab - це повна серверна платформа з керованими ігровими сервісами, аналітикою в режимі реального часу та LiveOps. Бекенд-сервіси PlayFab зменшують перешкоди для розробників ігор, пропонуючи як великим, так і малим студіям економічно ефективні рішення для розробки, які масштабуються відповідно до їх ігор та допомагають їм залучати, утримувати та монетизувати гравців. PlayFab дозволяє розробникам використовувати інтелектуальну хмару для створення та управління іграми, аналізу ігрових даних та покращення загального ігрового досвіду. Платформа PlayFab є природним доповненням до Azure для ігор. Платформа підтримується в 42 регіонах по всьому світу, забезпечує серверну інфраструктуру світового класу

Міжмережева ідентифікація та дані

Здійснюється за допомогою платформи PlayFab. Відбувається забезпечення ігрового процесу у режимі реального часу з низькою затримкою для будь-якої платформи. Також дана платформа реалізує цілодобовий моніторинг та захист від DDoS-атак. Відбувається контроль витрат: динамічне масштабування серверних ядер у відповідь на запит.

Специфікація вимог

Функціональні вимоги: Меню

Головне меню відображатиметься користувачеві під час входу в гру. Деталі меню перелічені нижче.

Головне меню:

1. Компанія
    1.1. Розпочати нову компанію
    1.2. Завантажити збережений процес
    1.3. Вийти в головне меню
2. Ранер
    2.1 Завантаження сцени з локацією та персонажем
    2.2 Вийти в головне меню
3. Налаштування
    3.1 Налаштування мови
    3.2 Налаштування звуку
    3.3 Налаштування графіки
    3.4 Налаштування контролю персонажа
4. Статистика
    4.1 Відображення статистики гравця
    4.2 Відображення таблиці лідерів
5. Вийти з гри

Меню паузи

Меню «Пауза» відображатиметься користувачеві, коли користувач натисне кнопку “Пауза” протягом гри.

Меню паузи:

1. Повернутися до гри
2. Опції
    2.1 Налаштування звуку

2.2 Налаштування графіки

2.3 Налаштування контролю персонажа

3. Повернутися в головне меню

4. Вийти з гри

Функціональні вимоги: Персонаж

При завантаженні локації персонаж автоматичного генерується і приземляється на платформу

Персонаж має змогу:

- бігати
- стрибати
- переходити від бігу до ходьби
- присідати
- робити підкат
- здійснювати ривок
- здійснювати подвійний стрибок при дворазовому натисканні кнопки, яка відповідає за стрибок
- здійснювати атаку зброєю
- здійснювати атаку магією
- автоматично відновлювати кількість магічної сили
- атакувати зброєю в стрибку
- атакувати магією в стрибку
- лазити по драбині
- знищувати певні елементи світу
- атакувати ворожі AI
- ховатися від ворожих AI
- отримувати пошкодження від ворожих AI
- регенерувати здоров’я, підбираючи спеціалі предмети
- отримувати пошкодження від пасток
- помирати, коли рівень його здоров’я стає рівним нулю

- помирати при падінні з платформи
- збирати внутрішньо-ігрову валюту
- користуватися спеціальними предметами
- носити в рюкзаку, фіксовану кількість предметів
- обирати нові уміння
- отримувати бали для розвитку характеристик
- використовувати бали для характеристик

Функціональні вимоги: Вороги

Після завантаження світу, вороги мають змогу:

- патрулювати територію
- переслідувати персонажа, яка він перебуває в радіусі переслідування(залежить від типу ворога)
- атакувати персонажа ближніми атаками
- атакувати персонажа дальними атаками
- отримувати пошкодження від персонажа
- отримувати пошкодження від пасток
- помирати після падіння з платформи
- регенерувати здоров’я(залежить від типу ворога)
- помирати після того як рівень здоров’я буде дорівнювати нулю
- використовувати спеціальні атаки(залежить від типу ворога)
- повертатися в місце дисклокації, після переслідування персонажа
- наносити персонажу пошкодження після смерті(залежить від типу ворога)
- розрізняти і оминати перешкоди

Інші функціональні вимоги

Тут представлені такі функціональні вимоги:

- користувач може змінити свій профіль
- користувач може авторизуватися та створити акаунт
- меню з достатньою кількістю функціональних можливостей, що надають змогу персонажу змінювати конфігурації профілю, здійснювати вхід в гру під своїм акаунтом, та виходити з неї.
- персонаж може бути телепортованим до іншої зони
- існує проста економічна модель для управління потоком валюти в грі
- об’єкти світу взаємодіють з персонажем
- здійснюється зміщення рухомих платформ
- пастки реагують на персонажа і наносять йому пошкодження при контакті
- коректне відображення різноманітних анімацій
- коректне відображення графічних елементів
- вчасне знищення об’єктів
- з ворога може випасти якийсь випадковий предмет

Non-Functional Requirements

Юзабіліті

Гра є орієнтованою на доволі широку аудиторію, включаючи як досвідчених, так і не досвідчених гравців. Беручи це до уваги, було проведено грунтовну роботу над взаємодією користувача та інтерфейсу. Гра спрямована на те, щоб бути простою у вивченні та приємною у використанні.

Якість

Якість дизайну та якість відповідності цій конструкції є однією з основних вимог розробки проекту. В процесі розробки ігрових механік, було одразу проведено їхнє тестування як незалежно одні від одних, так і в сукупності для зменшення вірогідності виникнення помилок на стадії тестування гри.

Сумісність з платформами

Платформа Unity реалізує мультиплатформеність, що дозволяє адаптувати гру практично під усі існуючі ігрові платформи. Широка сумісність з різними системами є критично важливою для того, щоб розширити аудиторію. В процесі розробки, було прийнято рішення, надати користувачу свободу вибору між мобільними операційними платформами IOS, Android та операційною системою Windows.

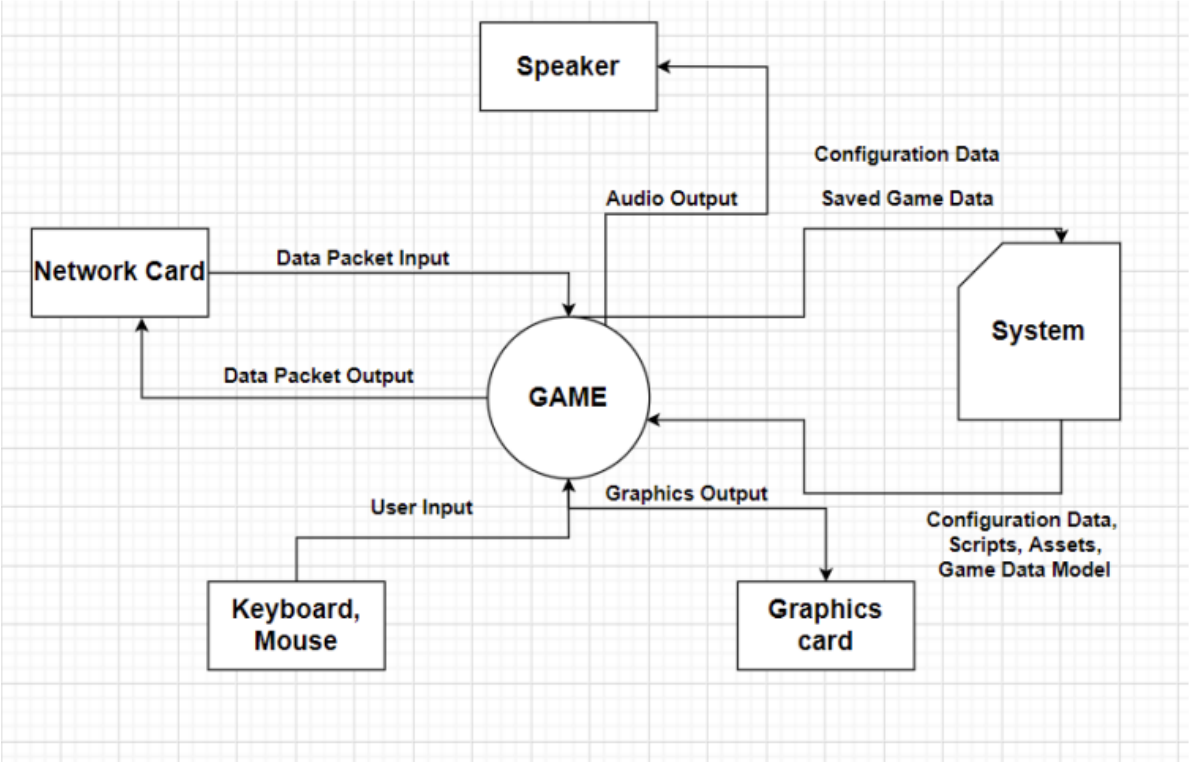
Надійність і міцність

Більшість тестів на залежність від відеоігор мають слабку конструктивність та обмежені можливості правильно ідентифікувати людей, яким заперечують. Метою цього дослідження було дослідити надійність та обґрунтованість нового тесту на залежність від відеоігор (Behavioral Addiction Measure-Video Gaming [BAM-VG]), який був частково розроблений для усунення цих недоліків. Регулярних дорослих відеогеймерів (n = 506) було набрано з

канадської онлайн-панелі та пройшло опитування, що включало три показники надмірної кількості відеоігор (BAM-VG; DSM-5 критерії для розладу в Інтернеті ігор [IGD]; і IGD-20) , а також питання щодо масштабності участі у відеоіграх та самозвітування про проблеми, пов’язані з відеоіграми. Через місяць вони були переоцінені з метою встановлення надійності тестування та повторного тестування. BAM-VG продемонстрував хорошу внутрішню узгодженість, а також надійність тестування та повторного тестування за 1 місяць. Обґрунтованість критеріїв була продемонстрована значними кореляціями з наступним: час, проведений за грою, самоідентифікація проблем відеоігор та оцінки на інших інструментах, призначених для оцінки залежності від відеоігор (DSM-5 IGD, IGD-20). Відповідно до теорії, аналіз основних компонентів виявив два компоненти, що лежать в основі BAM-VG, які приблизно відповідають порушенню контролю та значним негативним наслідкам, які виникають внаслідок цього порушення контролю. Разом із чудовою конструктивною валідністю та іншими технічними характеристиками BAM-VG являє собою надійний і дійсний тест на залежність від відеоігор.

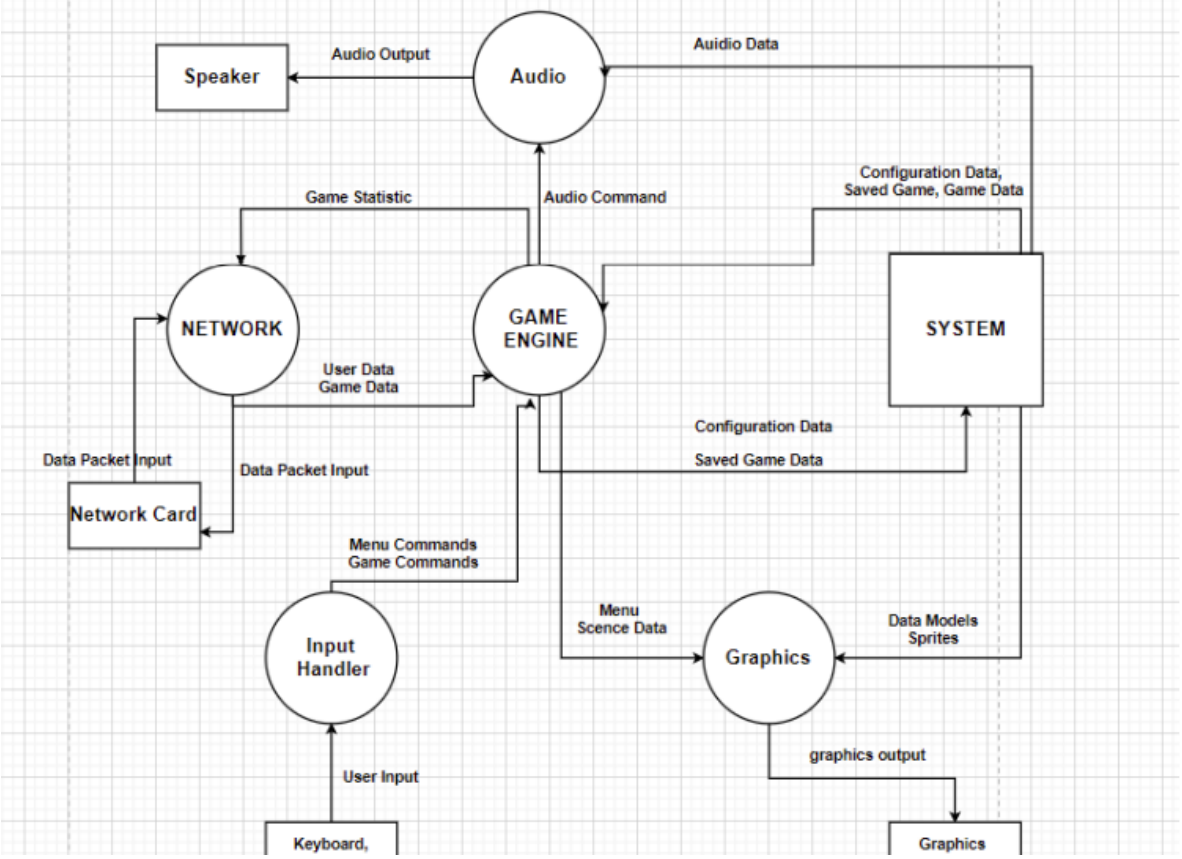
Системний аналіз та моделювання

Рівень 0 діаграми потоку даних



Програма приймає команди вводу через клавіатуру та мишу користувача. Одночасно програма передає звуковий вихід на динамік. Крім того, програма бере дані конфігурації, сценарій, аудіо, відео, моделі та текстури зі сховища (жорсткий диск) та зберігає в систему дані про конфігурацію, збережену гру сховища. Програма надсилає графічні результати до графічної карти, яка буде відображати інформацію на моніторі користувача. Крім того, програма передає статистику на сервер.

Рівень 1 діаграми потоку даних (на стороні клієнта)



Висновок

В процессі написання курсової роботи було розроблено прототип 2D гри, використовуючи Unity для розробки. Був реалізований зазначений функціонал. Переваги Unity підтвердилися: користувацька спільнота дійсно є дуже великою і більшість проблем, з якими я зіткнувся під час розробки, були вирішені дуже швидко завдяки достатньої кількості тематичних форумів та навчальних матеріалів.