МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук управ-
ління та адміністрування
Кафедра інформаційних технологій

Бакалаврська кваліфікаційна робота

на тему: Розробка ігрового додатку під мобільну операційну систему Андроїд

Виконала студентка 4 курсу групи К-42
Напрямок 6.05.01.01 Комп'ютерні науки

Царьова Аліса Олегівна

Керівник к.геогр.н., доцент
Кузніченко Світлана Дмитрівна

Консультант

Рецензент к.геогр.н., доцент
Гнатовська Ганна Арнольдівна

Одеса 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Комп'ютерних наук
Кафедра Інформаційних технологій
Рівень вищої освіти бакалавр
Напрям підготовки 6.050101 Комп'ютерні науки
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних технологій

_____
"_____" _____ 2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Царьової Аліси Олегівни
(прізвище, ім'я, по батькові)

1. Тема роботи «Розробка ігрового додатку під мобільну операційну систему Андроїд»

керівник проекту Кузніченко Світлана Дмитрівна,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "__"_____ 2019 р. № _____
2. Строк подання студентом проекту
3. Вихідні дані до проекту Ігровий додаток на ОС Android, СУБД Sqlite, , мова програмування C#, CASE-засоби Rational Rose.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз предметної області та постановка завдання. Вибір програмних засобів для реалізації системи. Моделювання ігрового додатку. Реалізація ігрового додатку.

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

Діаграма прецедентів, діаграма класів, інфологічна модель бази даних, даталогічна модель бази даних, UML діаграма класів.

6. Консультанти розділів проекту

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

7. Дата видачі завдання " " 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Термін виконання етапів проекту	Оцінка виконання етапу	
			у %	за 4-х бальною шкалою
1	Аналіз предметної області і постановка завдання дослідження			
2	Системний аналіз та проектування бази даних			
3	Вибір архітектури апаратної та програмної реалізації			
4	Проектування бази даних додатка			
5	Рубіжна атестація			
6	Розробка клієнтського застосування			
7	Програмна реалізація системи			
8	Тестування			
9	Оформлення пояснювальної записки та створення презентації			
	Інтегральна оцінка виконання етапів календарного плану (як середня по етапам)			

Студент _____ Царьова А.О.
(підпис) (прізвище та ініціали)

Керівник проекту _____ Кузніченко С.Д.
(підпис) (прізвище та ініціали)

Висновок про рівень оригінальності кваліфікаційної роботи

Назва роботи: «Розробка ігрового додатку під мобільну операційну систему Андроїд

Автор(и): *студентка гр.К -42 Царьва Аліса Олегівна*

Прізвище, ім'я та по батькові, науковий ступень, вчене звання керівника: *к.геогр.н., доц. Кузнічеенко Світлана Дмитрівна*

Обсяг роботи: 58 стор.

Програмно-технічні засоби перевірки на оригінальність роботи: *он-лайн система «TEXT.RU»*

Результати перевірки на оригінальність твору роботи

Назва структурного елементу твору (розділів)	Обсяг рукопису	Обсяг твору, який перевірено на оригінальність	Показник оригінальності (у відсотках)	Обгрунтування використання запозичень
частина 1	7277 символів	7277 символів	92.41%	_
частина 2	9701 символів	9701 символів	67.95%	_
частина 3	9366 символів	9366 символів	100%	_
частина 4	13421 символів	13421 символів	49.16%	_
частина 5	13329 символів	13329 символів	67.00%	_
частина 6	6922 символів	6922 символів	100%	-
Всього	60 016 символів	60 016 символів	**79.42%**	-

Загальний висновок:_____
_____
_____

Науковий керівник
к.геогр.н., доц. Кузнічеенко Світлана Дмитрівна. _____

в.о. зав. кафедри ІТ доц. Кузніченко С.Д. _____

Дата: 15.06.2019 р.

ЗМІСТ

ВСТУП

У сучасних умовах важко уявити собі людину без мобільного телефону, планшетного комп'ютера, смартфона або будь-якого іншого портативного мультимедійного пристрою. Ми звикли до того, що воно завжди під рукою, і це не тільки засіб спілкування, але має і багато корисних функцій, таких як калькулятор, органайзер, конвертер, календар, годинник. Смартфони з нової мобільної ігрової платформою можуть конкурувати з класичними портативними ігровими системами, такими як Nintendo DS або Playstation Portable.

Пристрій смартфона досить просто. В основному він складається з декількох окремих блоків - пам'ять, процесор, який займається організацією обчислень, пам'ять для зберігання даних, радіо-модуль, який, в свою чергу, складається з передавача і приймача і відповідає за зв'язок. Найцікавіше тут - операційна система, встановлена на внутрішній пам'яті. Від операційної системи і її версії залежать і всі основні можливості пристрою. Смартфони, а також персональні комп'ютери, існують з різними операційними системами, види яких будуть обговорюватися далі.

Ігрова індустрія на даний момент найбільш швидко розвивається, особливо це стосується ігрових додатків для мобільних платформ (консолей, смартфонів і планшетів). Зародилася в 70х роках вона зросла на величезну індустрію розваг з оборотами в мільярди доларів. Розробкою ігор зараз займаються як великі компанії (Blizzard Entertainment, Namco, Activision, Bethesda Softworks, Rockstar Games), так і невеликі фірми, спільноти і окремі инди- розробники. Прогрес не стоїть на місці, апаратна частина розвивається, у розробників з'являється більше свободи для творчості і самореалізації. З появою консолей і мобільних телефонів гри перестали бути прерогативою настільних комп'ютерів. Процесори в мобільних пристроях стають швидше, пам'яті більше,

зростає потужність мобільних графічних чіпів. Ринок мобільних ігрових додатків на даний момент є одним з найбільших і розвиваються.

Основними гравцями ринку мобільних платформ є корпорація Google зі своєю операційною системою Android, і Apple просуває мобільну ОС iOS. Обидві операційні системи, ще, будучи молодими, потребували популяризації і розвитку, потребували додатках для реалізації своїх можливостей, для чого компанія Google і Apple створили свої магазини додатків (Google play і Apple store). Магазин додатків дозволяв користувачам знаходити, завантажувати та встановлювати програми сторонніх розробників, які в свою чергу пропонували свої додатки за гроші або на безкоштовній основі. Розробка мобільних додатків стала надприбутковим бізнесом при мінімальних вкладеннях. Особливо швидко розвивався ринок саме мобільних ігор, величезна кількість ігор було перенесено на мобільні платформи з настільних комп'ютерів.

Згідно зі статистикою, більше половини мобільних пристроїв працюють під управлінням операційної системи Android. Очевидно, що розробка ігор під цю операційну систему зараз як ніколи актуальна. Ідея створення гри як дипломного проекту була продиктована моїм бажанням спробувати свої сили в розробці ігор (моя робота не пов'язана з цією областю), використовувати навички в розробці і використання інформаційних систем, отримані в процесі навчання у вузі.

На початку даної роботи ми розглянемо цілі і вимоги, що пред'являються до нашого ігрового додатку, і способи їх досягнення. Далі проведемо порівняльний аналіз і огляд ігрових движків с подальшим вибором кращого, і задовольняє нашим вимогам. У третьому розділі ми розробимо і розберемо структуру нашого застосування, розробимо меню, призначений для користувача інтерфейс, попутно вивчаючи можливості інструментарію і засобів розробки входять до складу обраного нами движка.

СПИСОК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

API (від Application Programming Interface) - набір методів, констант і структур надаються ОС, додатком або фреймворком для використання у зовнішніх програмних продуктах і використовуваних програмістами при написанні своїх додатків.

Android - операційна система для мобільних платформ (смартфонів, планшетів, електронних книг, ігрових приставок, ноутбуків), телевізорів, і навіть годин. Розробленої на базі Linux компанією Android, Inc., і згодом купленої компанією Google.

GUI (Graphical User Interface) - графічний інтерфейс користувача - забезпечує можливість управління поведінкою системи через візуальні елементи управління - вікна, списки, кнопки, гіперпосилання.

Скрипт - це програма або сценарій, написаний певною мовою, який автоматизує рішення деякої задачі.

TrueTypeFont (TTF) - формат зберігання шрифтів, розроблений компанією Apple в кінці 80-х років. Використовуються в багатьох сучасних операційних системах.

Ігровий движок (Engine) - це набір систем, інструментів які спрощують роботу програмісту при написанні гри, і складається з наступних підсистем:

- Графічна підсистема
- Підсистема введення-виведення
- Звукова підсистема
- Ядро

Движок зачіпає всі компоненти гри фізика, звукове оформлення, II, скриптинг, рендеринг, мережа.

Шейдер (Shader) - це програма для однієї з щаблі графічного конвеєра, яка використовується для завдання остаточних параметрів об'єктів і зображення. Шейдери пишуться на спеціалізованих мовах програмування і компілюються в байткод для виконання на GPU.

Ігровий Ассет (від англ. Game asset) - ресурс, що складається з однотипних даних, що є частиною ігрового контенту і використовуються, оброблювані комп'ютерною грою. Може зберігатися у вигляді файлу.

Інді-гра (від independent) - гра, створена окремим розробником або невеликою групою, без фінансової підтримки видавця.

Кросплатформеність (від англ. Cross-platform) - здатність додатки функціонувати на різних апаратних платформах і операційних системах без істотних доробок.

Фізичний движок - це комп'ютерна програма, яка виконує наближене моделювання певних фізичних систем (наприклад динаміка твердого тіла, виявлення зіткнень, динаміка рідин). Широко використовується в комп'ютерній графіці, ігрових додатках і кіно.

Візуалізація (rendering) - процес формування зображення з 2d або 3d моделі або декількох моделей за допомогою програми.

Сцена - сукупність графічних моделей розташованих в певному порядку.

Фреймворк (framework) - програмне забезпечення полегшує процес розробки, за рахунок використання єдиного API. Може включати в себе допоміжні бібліотеки коду, підтримку скриптів і інший інструментарій.

IDE (від Integrated development environment) - середовище розробки і набір інструментів використовуваних програмістами для розробки програмного забезпечення. У середу розробки зазвичай входить редактор коду, відладчик, компілятор і засоби збирання.

GPU (від Graphics processing unit) - графічний процесор прискорення розрахунків, виконання коду шейдеров і як підсумок створення зображення в буфері кадру і подальшого виведення його на екран. На даний момент широко використовується в відео картах персональних комп'ютерів, консолях і мобільних телефонах.

Текстура - растрове цифрове зображення, що відтворює візуальні властивості будь-яких поверхонь або об'єктів.

Спрайт - графічний об'єкт, який можна відобразити на екрані.

Атлас - набір спрайтів або текстур зібраних на одній або декількох великих структурах. Використовується для прискорення доступу до зображень і зменшення кількості перемикань між ними і як наслідок збільшення продуктивності графічної складової програми.

1 АНАЛІТИЧНА ЧАСТИНА

1.1 Аналіз предметної області

В даний час комп'ютерні технології стали активно застосовуватися в різних сферах нашого життя. Створення різних простих і складних комп'ютерних програм для різних областей знань постійно розвивається. Залежно від віку дитини комп'ютерна програма може виступати в якості опонента по грі, бути оповідачем, репетитором, екзаменатором. Вже існують комп'ютерні програми, спрямовані на розвиток різних психічних функцій дітей, таких, як візуальне і слухове сприйняття, увагу, пам'ять, словесні і логічні міркування і т.д, які можуть бути успішно використані дітьми старшого дошкільного і молодшого шкільного віку. Тепер на зміну настільним комп'ютерам приходять мобільні - планшетні ПК, смартфони, що дозволяє більш зручно працювати з усією інформацією.

Гра - це пізнавальна діяльність, вона є свого роду практичною формою мислення про природу навколишнього дитини соціальної реальності. Через особливості гри, дитина в грі вперше долучається до абстрактного мислення.

Комп'ютерні ігри - це новий вид розвиваючого навчання. Комп'ютерні технології відкривають нові можливості для використання педагогічних методів:

- використання матеріалу різної складності. Індивідуально дитині завжди можна запропонувати саме те, що в даний момент відповідає його можливостям і цілям навчання;
- для того щоб "бачити" проблеми в розвитку дитини, які важко виявити в традиційному навчанні;
- для того щоб сформувати розуміння дитиною процесу отримання власних навичок;

- складне програмне забезпечення є надзвичайно простим у використанні;
- багато методи успішно використовувалися раніше, тепер поставлені на комп'ютерну базу і отримали як би друге розвиток. З точки зору фахівця, це можливість поглянути на свою роботу під новим кутом зору, переосмислити методичні прийоми, збагатити знання і навички, якими він володіє;
- робота на комп'ютері створює для дитини більш комфортні умови для успішного виконання вправ;
- комп'ютерні технології забезпечують захоплюючі заняття для дитини в формі експериментування, моделювання, порівняння;
- дитина навчиться правильно говорити, намагатися виправити ту помилку, що він побачив, шукаючи методи самоконтролю, зосередивши увагу на привабливою графіку;
- діти менше втомлюються, більше здатні працювати;
- дивлячись на екран, дитина сама бачить результат своєї роботи.

Таким чином, використання комп'ютерної програми підвищує мотивацію використання не тільки стратегії гри, на якій базується програма, а й того, що дитина отримує схвалення, похвалу, не тільки дорослих, а й комп'ютера.

Але слід зазначити, що до використання комп'ютерних ігор слід підходити розумно і вибірково, звичайно ж, творчо

1.2 Огляд існуючих мобільних платформ

Наявність операційної системи (ОС) - головна особливість, яка відрізняє смартфон від звичайного мобільного телефону. При виборі конкретної моделі

телефону або пристрою, операційна система часто є визначальним фактором[1][1)].

Найбільш поширені операційні системи для смартфонів і платформ:

- Symbian OS - ОС займала більшу частину ринку смартфонів до кінця 2010 р На початку 2010 року на базі ОС залишається тільки 1 Платформа: Series 60, яка використовується в основному в пристроях Nokia, а також деякі моделях Samsung;
- BlackBerry OS (RIM) - система широко використовується в пристроях в першу чергу в Сполучених Штатах, так як спецслужби деяких країн не зацікавлені у використанні смартфонів в країні через те, що всі вхідні і вихідні дані зашифровані з використанням алгоритму шифрування AES;
- Windows Mobile і Windows CE - компактна операційна система Microsoft, випущені з 1996 року і займала найбільший сегмент ринку ОС для смартфонів до 2010 року, в даний час проходить поетапну відмову від підтримки і розвитку;
- Windows Phone 7 - розробка від Microsoft, радикально відрізняється від Windows Mobile;
- Palm OS - одна з популярних платформ, хоча даний час мобільні телефони на базі Palm OS малораспространени. Останній смартфон під управлінням операційної системи була випущений в кінці 2007 року (Palm Centro);
- Linux - широкого поширення ця операційна система на мобільних пристроях не отримала, проте її розвиток традиційно вважається перспективним напрямком. Смартфони на базі Linux поширюються головним чином в Азії;

[1)] [1] DevGam сравнение игровых движков. URL: http://devgam.com/sravnenie-igrovyx-dvizhkov-kakoj-vybrat (дата звернення 12.01.2019).

- Bada - новітня мобільна платформа, розроблена компанією Samsung. Першим телефоном на новій платформі став S8500 Wave;
- Android - портативна (мережева) операційна система для смартфонів, планшетих ПК, електронні книг, цифрових плеєрів, годин і нетбуків на базі ядра Linux. Спочатку розроблена Android Inc., яку потім купив Google. Згодом Google ініціювала створення альянсу Open Handset Alliance (OHA), який зараз займається підтримкою і подальшим розвитком платформи. Android дозволяє створювати додатки на основі Java, який управляють пристроєм через розроблені Google бібліотеки. Android Native Development Kit дозволяє системі використовувати бібліотеки і компоненти додатків, написаних на C та іншими мовами;
- ОС IOS (до 24 червня 2010 року - iPhone OS) - це мобільна операційна система, розроблена і виготовлена американською компанією Apple. Вона була випущена в 2007 році; спочатку - для iPhone і IPod Touch, а пізніше - для таких пристроїв, як IPad і Apple TV. На відміну від Windows Phone і Google Android, доступна тільки для пристроїв, вироблених Apple;

1.3 Опис предметної області

Операційна система Android заснована на платформі Linux для мобільних пристроїв, розроблених Open Handset Alliance (OHA), ініційованої Google. Вона дозволяє створювати додатки на основі Java, які керують пристроєм через бібліотеки, розроблені Google. Крім того, можна писати програми на C та іншими мовами програмування за допомогою Android Native Development Kit.1.5 (Cupcake), випущеної 30 квітня 2009 г. Серед основних поліпшень з'явилася підтримка запису і перегляду відео в режимі камери; підтримка

Bluetooth A2DP; можливість автоматичного підключення до Bluetooth-гарнітури[3][1)].

Перший пристрій, яке працювало під управлінням Android, став розроблений HTC смартфон T-Mobile G1, який був запущений 23 вересня 2008 року Незабаром пішли численні анонси інших виробників смартфонів, які планують випустити пристрої з Android.

Є кілька основних переваг Google, які відрізняють пристроїв на базі платформи Android від аналогічних продуктів:

- відкритість - Android дозволяє отримати доступ до основних функцій мобільного пристрою, використовуючи стандартний API виклик;
- руйнування кордонів - можна об'єднувати інформацію з Інтернету з даними телефону, наприклад, контактну інформацію або географічних даних про місцезнаходження, щоб отримати нові можливості;
- рівність додатків - для Android немає ніякої різниці між основними телефонними додатками і програмним забезпеченням сторонніх виробників - ви можете навіть змінити програму для набору номера або заставку;
- швидко і легко розробляти програми - в SDK є все необхідне для створення і запуску Android-додатків, в тому числі імітатор справжнього приладу і передові інструменти налагодження.

Гнучкість Android має ціну: компанії, вважаючи за краще розробляти свої власні інтерфейси, постійно прагнуть до випусків нових версій ОС. Пристрої, що випустили кілька місяців тому, стають застарілими, оскільки оператори і виробники не хочуть створювати оновлення програмного забезпечення, так що користувачі можуть використовувати нові функції Android. Наприклад, багато експертів відзначають, що платформа заснована на Java, тому переваги і

[1)] [3] List of game engines. URL: https://80.lv/articles/the-most-comprehensive-list-of- game-engines/ (дата звернення 27.03.2019).

можливості операційної системи Linux на Android не використовуються повною мірою. Вона також не використовує в платформі будь-які з популярних графічних інструментів (Toolkit) і бібліотек (наприклад, Ot або GTK), що робить малоймовірною появу великої кількості додатків, перенесених на Linux з повною версією для домашнього комп'ютера на мобільну платформу через відсутність загального сервера і бібліотеці зображень. Крім того, повідомлялося, що Google буде на свій розсуд видаляти додаток на телефонах користувача, якщо порушуються умови їх використання.

Аналітики і експерти пророкують для IT-ринку Google Android хороші комерційні перспективи, в принципі, для продуктів на базі програмного забезпечення з відкритим вихідним кодом, що вже не є сенсацією. Вони поступово займають IT простір, витісняючи з нього визнаних лідерів, створюючи конкуренцію, що само по собі може тільки позитивно вплинути на відновлення ринку.

Велика частина коду ліцензується відповідно до ліцензії Apache 2, відкрита і необмежена ліцензія дозволяє вільне використання вихідного коду для створення своїх власних систем.

1.4 Огляд кроссплатформенного ігрового движка Unity

На веб-сайті Unity представлено кілька тарифних планів движка відрізняються вартістю і рівнем підтримки (рис. 1). Кожна з версій цього програмного забезпечення відкриває для користувача новий функціонал, який відсутній в базовій версії. Відмінності версій один від одного представлені на рисунку 1.1[4][1)].

[1)] [4] Unity store. URL: https://store.unity.com/ru?_ga=2.200322542.1765686190.1560285797-130111505.1547668569 (дата звернення 01.03.2019)

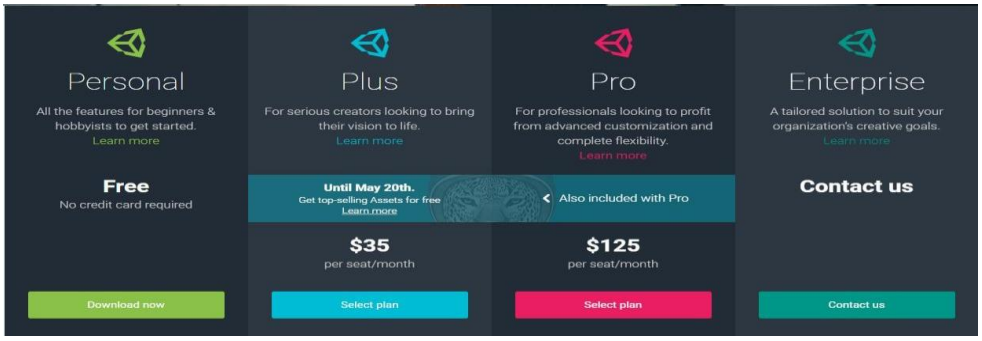


Рисунок 1.1 - Вартість передплати

Так само можна побачити основні відмінності версій, і їх вартість. У нашому випадку буде використана версія Personal, вона є безкоштовною, не має якихось суттєвих для проекту обмежень, а найголовніше має практично все необхідне для старту розробки з нуля.

Compare plans	Personal	Plus	Pro	Enterprise
Accelerator Pack		Free (a $190 value)	Free (a $190 value)	
All Engine Features	✓	✓	✓	✓
All Platforms	✓	✓	✓	✓
Continuous Updates	✓	✓	✓	✓
Royalty Free	✓	✓	✓	✓

Рисунок 1.2 Відмінності версій

1.4.1 Функціональні можливості Unity

Основним критерієм вибору того чи іншого кроссплатформенного ігрового движка є ставлення його функціональності до вартості. У даній роботі

необхідно створити двовимірний кроссплатформений ігровий додаток. Нижче будуть наведені основні критерії, за якими можна охарактеризувати Unity як сучасний ігровий движок з багатим потенціалом і широкими можливостями [5][1].

Особливості графіки:

- Система частинок. Спосіб представлення об'єктів, що не мають чітких геометричних кордонів (хмари, туманності, дим, пар, рідини і т.д.);
- Ефект рендеру в текстуру, тобто можливість приміщення зображення з камери нема на екран, а в текстуру. Потрібно для створення деяких ефектів;
- Автоматична нарізка і анімація спрайтів. вбудований редактор спрайтів здатний автоматично нарізати растр з спрайтами на кадри і створювати з них анімацію. Прив'язувати події до певних ключовим кадрам анімації;
- Пакувальник спрайтів, використання різноманітних видів стиснення текстур і упаковка використовуваних кадрів спрайтів в один ресурс;
- 2D фізика. Вбудована бібліотека Box2D, і підтримка NVIDIA PhysX (фізика реалізована за рахунок використання в розрахунках GPU);
- Альфавирез, обрізання частини зображення по масці, яку розміщують в альфакомпоненте текстури;
- Dynamic Batching, алгоритм оптимізації рендеринга дозволяє домогтися збільшення продуктивності.

Особливості роботи з анімацією:

- State machines. Візуальний редактор станів, які можна прив'язати до ігрових об'єктів і використовувати їх для створення і поліпшення анімації;

[1)] [5] Руководство Unity. URL: https://docs.unity3d.com/ru/current/Manual/index.html (дата звернення 18.02.2019).

– Інтегрований редактор анімації.

Загальні можливості програмного забеспечення:

– Створення скриптів за допомогою с #, JavaScript або Boo;
– Нативная інтеграція з Visual Studio;
– Функції II з передовою системою пошуку шляху;
– Реалістична анімація;
– Розміщення одним клацанням;
– Оптимізована графіка;
– Конвеєр імпорту;
– 64-розрядний розширюваний редактор.

1.5 Переваги і недоліки Unity

У порівнянні з іншими двигунами, Unity має такі переваги:

– Простота використання і продуктивність;
– Величезний число підтримуваних платформ: Windows, iOS, Android;
– Можливість запуску проекту в браузері як HTML5 додатки;
– Гарна продуктивність як в невеликих іграх на слабких мобільних платформах, так і в складних великих проектах на high end консолях;
– Підтримка як 2D так і 3D режимів без особливих зусиль;
– Потужна вбудована анімаційна система Mechanim;
– Мова програмування C # або JavaScript;
– Вбудований візуальний редактор, сильно спрощує і прискорює процес створення програми;
– Assets store, магазин для розробників, що відкриває доступ до величезного числа моделей, текстур, скриптів і доповнень для Unity.

Статистику по додатках на Unity можна подивитися на малюнку 3.2[6][1].

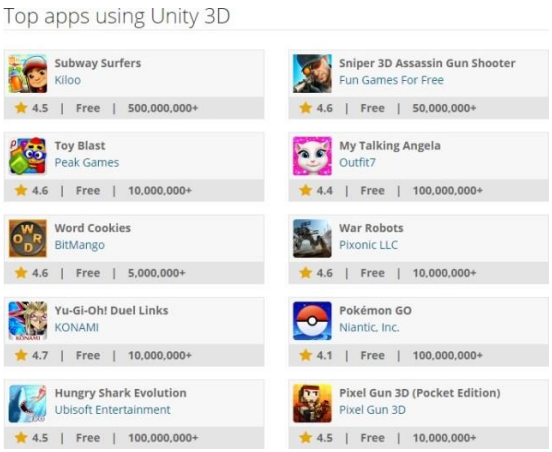


Рисунок 1.2 Статистика по додатках

– Можливість використання абсолютно безкоштовною середовища розробки MonoDevelop йде в комплекті з Unity, якої зовнішньої IDE (наприклад, Microsoft Visual Studio або IntelliJ IDEA);
– Все функції Unity абсолютно безкоштовні до досягнення доходу з продажу в 100 тисяч доларів;

До недоліків Unity можна віднести наступне:

– Ісходний код Unity закритий і доступний тільки в разі покупки Enterprise ліцензії. У разі будь-яких проблем, доведеться чекати поновлення і виправлень, що іноді займає досить багато часу;
– Відсутність такого звичного поняття як "game loop" або ігрового циклу. Немає єдиної точки входу, як в інших двигунах. Кожен ігровий об'єкт може мати скрипт або кілька скриптів, свій набір подій і свій ігровий цикл;

[1)] [6] Русскоязычный форум Unity3d. URL: http://unity3d.ru (дата звернення 28.01.2019).

– Не звичний і місцями складний в освоєнні візуальний редактор, який може виявитися проблемою для початківців розробників.

1.6 Установка Unity і короткий опис можливостей

Для того щоб почати установку Unity на свій комп'ютер слід зайти на вебсайт www.unity3d.com в розділ Store і вибрати потрібну підписку, в нашому випадку, це буде Personal Edition, це єдина представлена безкоштовна версія.

Після установки у нас є можливість відразу зайти в налаштування і вказати яку IDE ми будемо використовувати (рис. 3.3). із запропонованих варіантів можна вибрати Mono Developer, Microsoft Visual Studio, або будь-який інший IDE (наприклад: IntelliJ IDEA, для цього є спеціальна версія для Unity з підтримкою c# і javascript), який вам підходить. Нашим вибором буде Mono Developer.

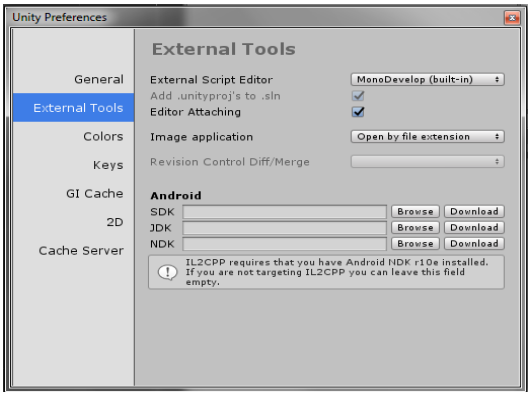


Рисунок 1.3 - Unity preferences

Після установки у нас є можливість відразу зайти в налаштування і вказати яку IDE ми будемо використовувати (рис. 1.3). із запропонованих варіантів можна вибрати Mono Developer, Microsoft Visual Studio, або будь-який

інший IDE (наприклад: IntelliJ IDEA, для цього є спеціальна версія для Unity з підтримкою c# і javascript), який вам підходить.

Так як збирати проект ми будемо не тільки під Windows, але і під Android, то нам необхідно буде завантажити Android SDK. Зробити це можна зайшовши на вебсайт developer.android.com в розділ downloads і вказавши шлях до нього після установки в Unity preferences.

При першому запуску Unity ми потрапляємо у вікно створення проектів, так як таких у нас поки що немає, Unity відразу пропонує нам створити новий проект (рис 1.4).

Рисунок 1.4 - Створення проекту

1. Ім'я за замовчуванням New Unity Project, ви можете поміняти його на що завгодно (рис. 1.5). Розташування вашого майбутнього проекту вказує на домашню папку вашого профілю в Windows. Розташування можна змінити тільки зараз, при створенні проекту.

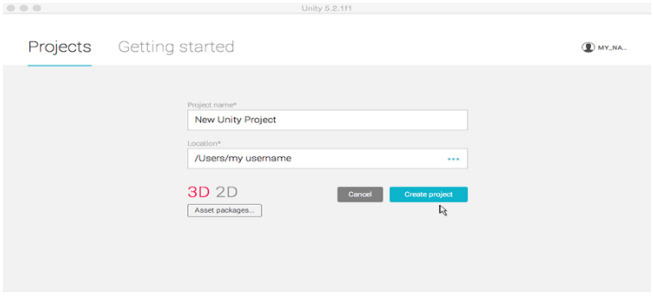


Рисунок 1.5 - Вибір назви і розташування проекту

3. Вибрати тип проекту, 3D або 2D. Це вплине на налаштування візуального редактора Unity. Їх завжди можна поміняти вже після створення проекту, так що можна залишити їх за замовчуванням.

4. Вибрати додаткові Ассет (рис. 1.6), які будуть включені в наш проект. Ассет представляють собою заздалегідь створені текстури, стилі, світлові ефекти, корисні додаткові інструменти та інше.

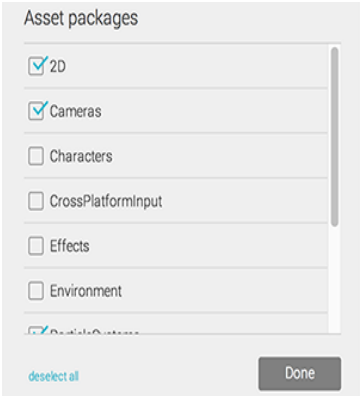


Рисунок 1.6 - Вибір Ассет пакетів за замовчуванням

Після чого можемо створити проект і тепер завжди при старті Unity ми будемо потрапляти в список створених нами проектів, вибравши один з них ми потрапляємо в візуальний редактор Unity.

1.7 Інтерфейс редактора Unity

Редактор складається з декількох основних частин, які можна перетягувати, групувати і розташовувати, так як нам зручно (рис. 1.7)[7][1)].

[1)] [7] Видео уроки по Unity3D. URL: https://www.youtube.com/watch?v=m0sKo81DZME (дата звернення 16.04.2019).

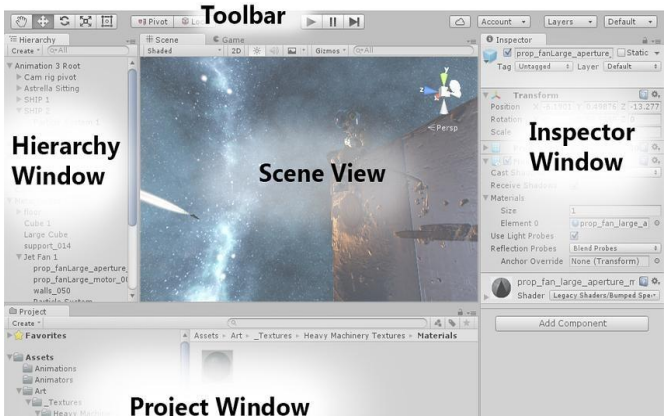


Рисунок 1.7 - Візуальний редактор

Project window (вікно проекту) відображає список встановлених Ассет, текстур, файлів анімацій, скриптів, матеріалів, спрайтів, сцен, в загальному, все того що ви додали і будете використовувати в своєму проекті. Додавання файлів відбувається простим перетягуванням файлів або їх груп в це вікно. У разі, якщо ми хочемо створити новий файл, то встановивши курсор на цьому вікні правою кнопкою миші, ми можемо викликати контекстне меню де вибираємо тип ресурсу, який хочемо створити (рис. 1.8).

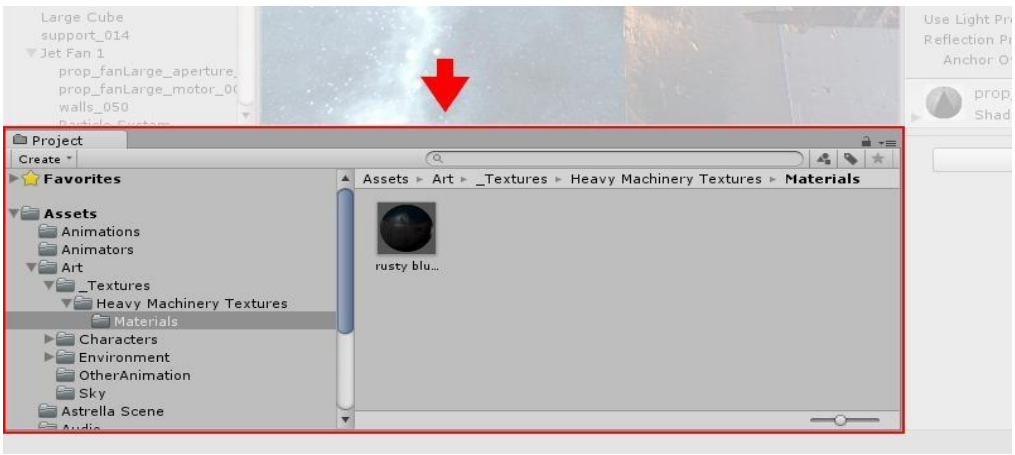


Рисунок 1.8 - Вікно Project

Unity автоматично імпортує додані файли в потрібний для неї формат. Надалі ми можемо експортувати всі додані сюди файли одним пакетом і використовувати їх в іншому проекті.

Вікно Scene, дозволяє нам управляти, редагувати сцену або кілька сцен проекту, тобто вікно, де будуть відображатися об'єкти нашої гри (рис.1.9).

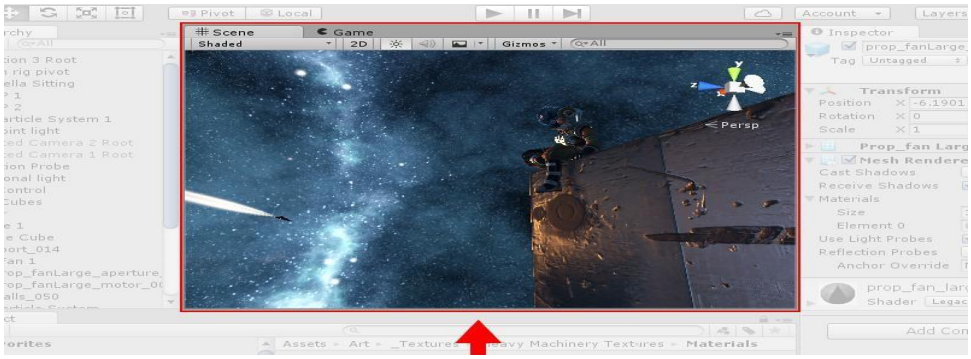


Рисунок 1.9 - Вікно Scene

Вікно має можливість перемикатися в 2D або 3D режими, в залежності від типу нашого проекту, вмикати і вимикати освітлення. Дозволяє вибирати, переміщати і змінювати позицію об'єктів знаходяться на сцені (наприклад: джерела світла, персонажа, камери та інші), тимчасово вимикати об'єкти.

Вікно Hierarchy відображає ієрархію всіх об'єктів знаходяться на сцені. Кожен елемент сцени має запис в цьому вікні. вікно Hierarchy відображає структуру сцени (рис.1.10) і як об'єкти прикріплені один до одного, воно нерозривно пов'язане з вікном Scene.

Вікно Inspector дає нам можливість переглядати і змінювати всі можливі параметри обраних об'єктів. Так як параметри різних об'єктів відрізняються, то розташування і вміст вікна інспектора може змінюватися в залежності від обраного об'єкта.

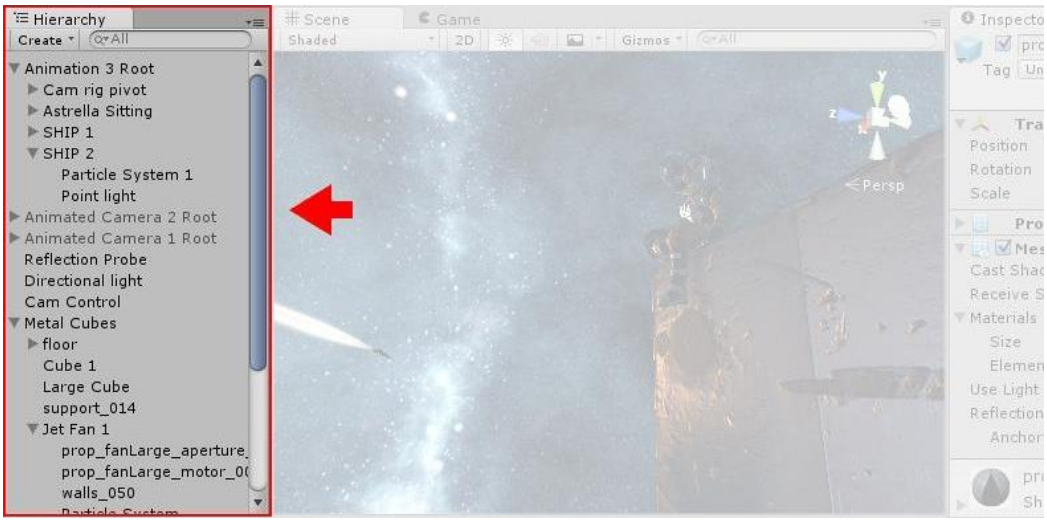


Рисунок 1.10 - Вікно Hierarchy

Інспектор (рис. 1.11) ми будемо використовувати для перегляду та зміни параметрів і налаштувань майже все, що є в редакторі Unity, включаючи ігрові об'єкти, матеріали, спрайт, Ассет і так далі.

Якщо до об'єкта доданий скрипт, то Інспектор (рис. 1.12) буде відображати всі публічні змінні і посилання на інші об'єкти, присутні в ньому. Вміст цих змінних можна змінювати прямо в цьому вікні, не залазячи в скрипт і не змінюючи код безпосередньо.

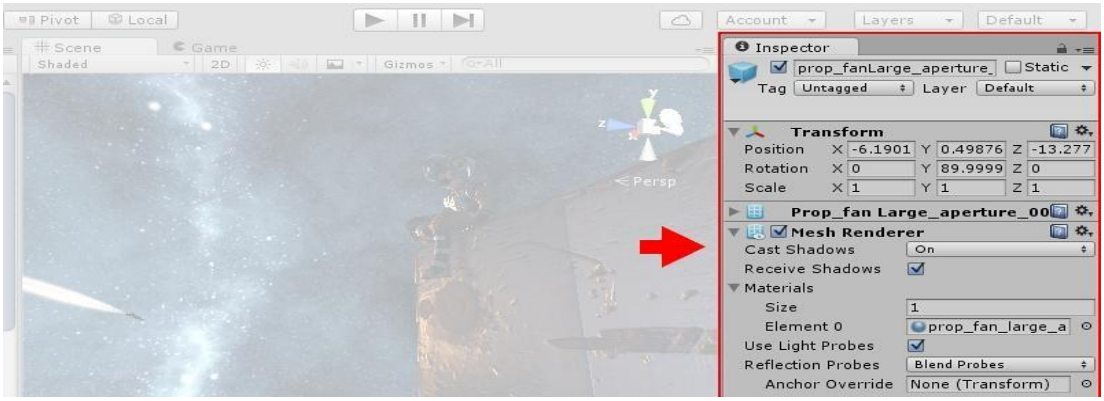


Рисунок 1.11 - Вікно Inspector

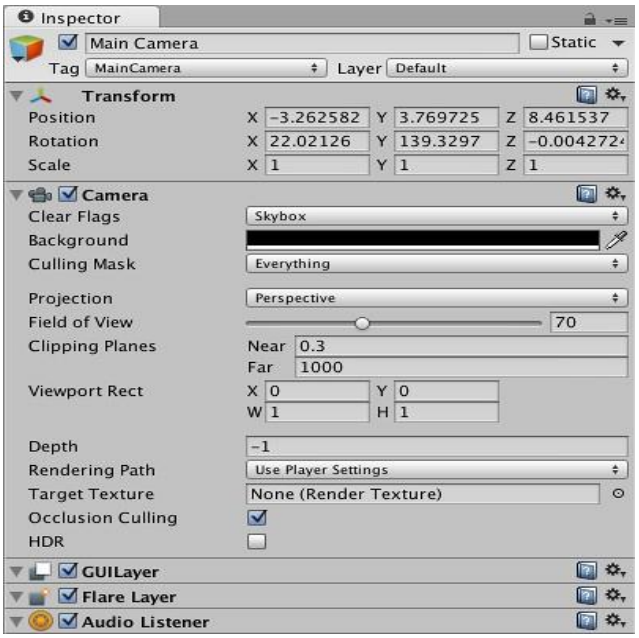


Рисунок 1.12 - Вікно Inspector – Camera

Toolbar надає доступ до найбільш часто використовуваних функцій редактора (рис. 1.13). Справа знаходяться кнопки доступу до хмарних сервісів і аккаунту Unity. Layers відповідає за видимість шарів на сцені, дозволяє включати і вимикати їх у процесі налагодження. У середній частині панелі знаходяться кнопки управління роботою додатка, що дозволяють запускати, ставити на паузу і покроково його виконувати. Ліва частина містить стандартні інструменти управління сценою і об'єктами на ній.

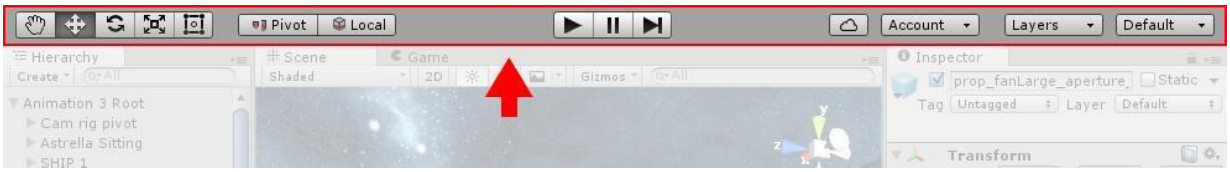


Рисунок 1.13 – Toolbar

2 РОЗРОБКА АРХІТЕКТУРИ ІГРОВОГО ДОДАТКУ

Створення ігрового додатку дуже трудомісткий і складний процес. В ході цього процесу величезна кількість елементів об'єднується в програму призначену для того, щоб розважати величезну масу людей.

2.1 Основні модулі ігрового додатку

Наше ігрове додаток буде мати будову класичної гри, яка складається з декількох основних частин або модулів, кожен з яких виконує свою унікальну функцію[8][1)]:

- Модуль управління. Відповідає за опитування зовнішніх пристроїв управління, наприклад, таких як миша, клавіатура;
- Модуль, який відповідає за рендеринг об'єктів сцени.

Цей модуль займається тим, що відображає на екрані гравця ігрові об'єкти, елементи меню, текст, вже з урахуванням використовуваних текстур, матеріалів, джерел світла, шейдерів. Саме цей модуль по засобом API (наприклад: DirectX, OpenGL, в залежності від обраної платформи) спілкується з апаратною частиною комп'ютера, і зокрема GPU:

- Модуль, який відповідає за звук;
- Модуль штучного інтелекту.

В нашій майбутню гру буде використовуватися тільки частина з цих модулів, це модуль управління, звуку і рендеринга. Всі ці модулі вже реалізовані в ігровому движку, тому нам немає необхідності самим займатися їх розробкою.

[1)] [8] Хокинг Д. Unity in Action: Multiplatform Game Development in C# with Unity 5 : Manning Publications Company. 2015 – 352с (дата звернення 17.04.2019)

Для гри нам залишається тільки реалізувати свій модуль, який відповідає за ігрову логіку і фізику, який буде управляти взаємодією ігрових об'єктів на сцені, підраховувати очки, перемикати сцени в залежності від обраного пункту меню, завантажувати карти рівнів і так далі.

Розгорнутий опис функціональних вимог здійснюється на етапі проектування програми. Для того щоб деталізувати вимоги, необхідно виділити процеси, що відбуваються в заданій предметній області. Опис таких процесів на UML виконується у вигляді прецедентів (рис. 2.1). Прецеденти є сценарієм або варіантом використання ігрового програми при взаємодії із зовнішнім середовищем. За допомогою прецедентів описується функціонування гри з точки зору користувача, який називається в UML актором (actor). Актор є будь-яку зовнішню по відношенню до модельованої системі сутність (людина, програмна система, пристрій), яка взаємодіє з системою і використовує її функціональні можливості для досягнення певної мети або вирішення приватних завдань.

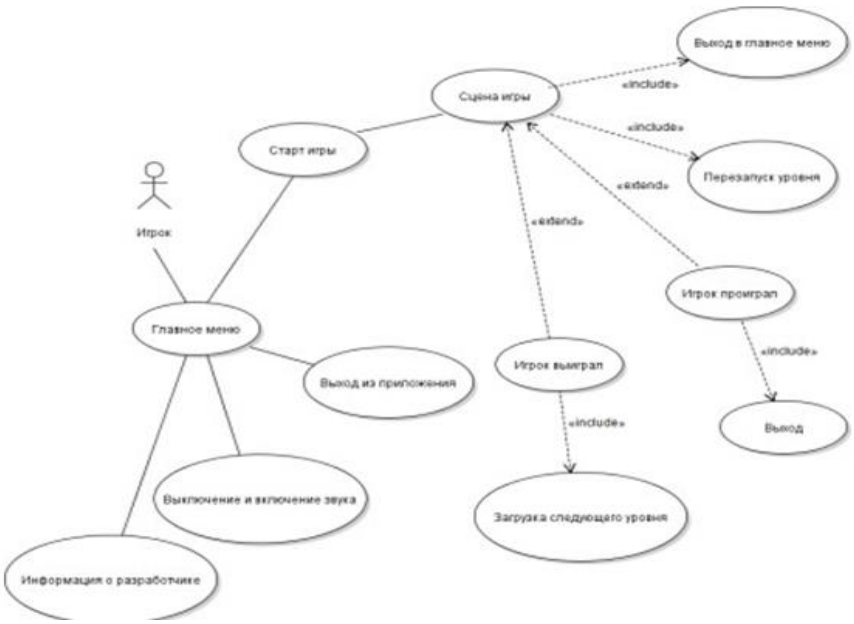


Рисунок 2.1 Діаграма прецедентів

2.2 Ігровий цикл

Серцем гри є ігровий цикл (рис. 2.2), який здійснює послідовний виклик всіх цих модулів в правильній послідовності і синхронізацію їх між собою. Наш ігровий цикл буде реалізований в скрипті і підключений до камери сцени.

Рисунок 2.2 - Діаграма найпростішого ігрового циклу

Ігровий цикл нашої гри буде складатися з:

– опитування стану зовнішніх пристроїв управління;
– зміна положень ігрових об'єктів;
– підрахунок очок;
– лічильник часу відведеного на проходження рівня;
– оновлення координат фізичних об'єктів сцени;
– оновлення елементів меню і інтерфейсу.

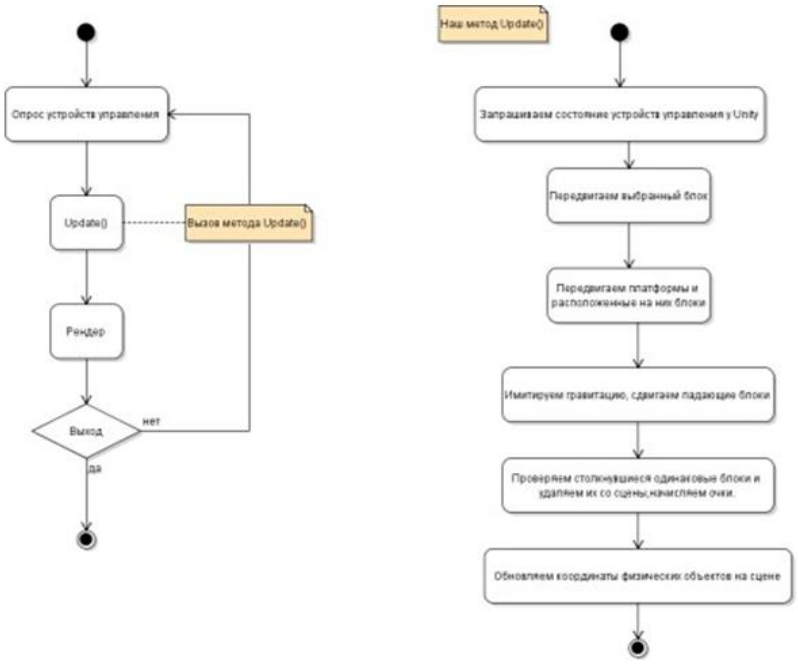


Рисунок 2.3 Реалізація ігрового циклу в нашій грі

Використання класичного ігрового циклу в Unity неможливо через особливості архітектури движка. Кожен об'єкт в Unity має в скрипті два обов'язкових методу. Перший метод Start() для ініціалізації об'єкта і використовується один раз при додаванні об'єкта на сцену. Другий метод називається Update(), Unity викликає його кожен раз і у кожного об'єкта, перед тим як оновити всю сцену (рис. 2.2)[9][1)].

2.3 Система меню

Для додатка потрібно реалізувати систему меню, яка дозволить гравцеві запускати гру, перезапускати рівень, виходити з програми, включати і вимикати звук. Для її проектування зручно використовувати діаграму станів.

[1)] [9] Руководство по программированию на C#. https://docs.microsoft.com/ru-ru/dotnet/articles/csharp/programming-guide/index (дата звернення 19.04.2019).

Головне меню буде включати наступні пункти:

– Старт гри, вибір цього пункту викликає завантаження сцени з останнім пройденим рівнем (рис. 2.4);
– вихід з програми (рис. 2.5);
– кнопка Звук, повністю виключає або включає звукові ефекти в ігровому додатку (рис. 2.6);
– інформація про творця гри, показує вікно з короткою інформацією про розробника (рис. 2.7).

Рисунок 2.4 Діаграма станів - Старт гри

Рисунок 2.5 Діаграма станів - Вихід з гри

Рисунок 2.6 Діаграма станів – Звук

Рисунок 2.7 Діаграма станів – Інформація

Додаткове меню, яке можна буде використовувати в процесі гри:

– Рестарт рівня, перезапускає сцену з рівнем, скидає лічильник часу;

– вихід в головне меню.

На стадії розробки гри вкрай необхідно приділити увагу графічної, візуальної складової гри, від її якості залежить популярність гри, а значить і кількість її продажів.

Рисунок 2.8 Діаграма станів - Рестарт рівня

На стадії розробки гри вкрай необхідно приділити увагу графічної, візуальної складової гри, від її якості залежить популярність гри, а значить і кількість її продажів.

Але й не варто сильно захоплюватися цим процесом, так як це може позначитися на підсумковій продуктивності ігрового програми і відповідно погіршити такі показники як фреймрейт або fps (frames per second), відгук управління і плавність ігрового процесу, обсяг займаного іграми платформи простору і часом завантажити програму з мережі.

3. ОСНОВНІ ЕТАПИ РЕАЛІЗАЦІЇ ГРИ

3.1 Проектування інфологічної модель бази даних

Інфологія ІС – складене слово від двох коренів слів «інформація» і «логіка», тобто інформаційна логіка (функціонування) інформаційної системи. Відповідно, інфологічне (інформаційно-логічне) моделювання та проектування представляють процеси, пов'язані з поданням семантики у вигляді моделі, якій властиві морфізм і заходи. Основними конструктивними елементами інфологічних моделей є сутності, зв'язки між ними і їх властивості (атрибути). Інфологічна модель бази даних являє собою опис об'єктів (сутностей), з набором атрибутів і зв'язків між ними та призначена для структурного утворення предметної області. Вона повинна бути стабільної та незмінною.

Загальновідомо, що при побудові інфологічних моделей баз даних, зокрема, часто користуються мовою ER-діаграм (від англ. entity-relationship, тобто «сутність-зв'язок»). Її компонентами є сутності і зв'язки. Сутністю може бути будь-який інформаційний об’єкт, що необхідно зберігати у базі даних. Це може бути формалізований опис конкретного галузі предметної області інформаційної системи, що розроблюється. Дуже важливо розрізняти поняття сутності та екземпляру сутності, адже екземпляром сутності є лише конкретний прояв дії або об’єкту галузі, що описується [9].

Атрибутом сутності виступає та чи інша характеристика присутня сутності. Вона може бути як кількісною, так і якісною. Назва атрибута повинна унікальною у межах однієї сутності, але атрибути різних сутностей можуть співпадати. Атрибут вказує на ту інформацію, яку необхідно зібрати про сутність. Атрибути також має екземпляри, але в даному випадку кожному атрибуту привласнюється тільки одне значення до відповідної сутності.

Ключем сутності називають набір атрибутів або один атрибут, що однозначно ідентифікують сутність у межах моделі.

Зв'язок – асоціювання двох або більше сутностей. Якби призначенням бази даних було тільки збереження окремих, не пов'язаних між собою даних, то її структура могла б бути дуже простою. Проте одна з основних вимог до організації бази даних – це забезпечення можливості відшукання одних сутностей за значеннями інших, для чого необхідно встановити між ними певні зв'язки. А так як в реальних базах даних нерідко містяться сотні або навіть тисячі сутностей, то теоретично між ними може бути встановлено більше мільйона зв'язків. Наявність такої безлічі зв'язків і визначає складність інфологічних моделей.

При побудові інфологічних моделей можна використовувати мову ER-діаграм (від англ. Entity-Relationship, тобто сутність-зв'язок). У них сутності зображуються позначеними прямокутниками, асоціації – поміченими ромбами або шестикутниками, атрибути – поміченими овалами, а зв'язки між ними – ненаправленими ребрами, над якими може проставлятися ступінь зв'язку (1 або буква, що заміняє слово "багато") і необхідне пояснення.

Між двома сутностями, можливі чотири види простих зв'язків: «один до одного», «один до багатьох», «багато до одного», «багато до багатьох» і більш складні зв'язки, наприклад, семантичні зв'язки, що визначають якість функціонування семантичних інформаційних систем (інформаційних порталів, бібліотек, навчальних систем і т.д.).

Перший тип – зв'язок ОДИН-ДО-ОДНОГО (1: 1): в кожен момент часу кожному представнику (екземпляру) сутності А відповідає 1 чи 0 представників сутності В.

Другий тип – зв'язок ОДИН-ДО-БАГАТЬОХ (1: М): одному представнику сутності А відповідають 0, 1 або кілька представників сутності В.

Так як між двома сутностями можливі зв'язки в обох напрямках, то існує ще два типи зв'язку БАГАТО-ДО-ОДНОГО (M: 1) і БАГАТО-ДО-БАГАТЬОХ (M: N).

Крістофер Дейт – один з найбільших фахівців в області баз даних – визначає три основні класи сутностей: стрижневі, асоціативні і характеристичні, а також підклас асоціативних сутностей – позначення.

Стрижнева сутність (стрижень) – це незалежна сутність.

Асоціативна сутність (асоціація) – це зв'язок виду "багато-до-багатьох" між двома або більше сутностями або екземплярами сутності.

Асоціації розглядаються як повноправні суті:

- вони можуть брати участь в інших асоціаціях і позначеннях точно так же, як стрижневі сутності;
- можуть мати властивості, тобто мати не тільки набір ключових атрибутів, необхідних для вказівки зв'язків, а й будь-яке число інших атрибутів, що характеризують зв'язок.

Характеристична сутність (характеристика) – це зв'язок виду "багато-до-одного" або "один-до-одного" між двома сутностями (окремий випадок асоціації). Єдина мета характеристики в рамках розглянутої предметної області складається в описі чи уточнення деякої іншої сутності. Необхідність в них виникає в зв'язку з тим, що сутності реального світу мають іноді багатозначні властивості.

Позначаюча сутність або позначення – це зв'язок виду "багато-до-одного" або "один-до-одного" між двома сутностями і відрізняється від характеристики тим, що не залежить від позначаємої сутності.

Як правило, позначення не розглядаються як повноправні сутності, хоча це не привело б до будь-якої помилки.

Позначення і характеристики не є повністю незалежними сутностями, оскільки вони припускають наявність деякої іншої сутності, яка буде "позначатися" або "характеризуватися". Однак вони все ж є окремі випадки сутності і можуть, звичайно, мати властивості, можуть брати участь в асоціаціях, позначеннях і мати свої власні (нижчого рівня) характеристики. Підкреслимо також, що всі екземпляри характеристики повинні бути обов'язково пов'язані з будь-яким екземпляром характеризуємої сутності. Однак допускається, щоб деякі екземпляри характеризуємої сутності не мали зв'язків.

Ключ або можливий ключ – це мінімальний набір атрибутів, за значеннями яких можна однозначно знайти необхідний екземпляр сутності. Мінімальність означає, що виключення з набору будь-якого атрибута не дозволяє ідентифікувати сутність по тим атрибутам, що залишилися. Кожна сутність має хоча б один можливий ключ. Один з них приймається за первинний ключ. При виборі первинного ключа слід віддавати перевагу простим ключам або ключам, складеним з мінімального числа атрибутів. Недоцільно також використовувати ключі з довгими текстовими значеннями.

Не допускається, щоб первинний ключ стрижневий сутності (будь-який атрибут, який бере участь в первинному ключі) брав невизначений значення. Інакше виникне суперечлива ситуація: з'явиться екземпляр стрижневий сутності, який не володіє індивідуальністю, і, отже не існуючий. З тих же причин необхідно забезпечити унікальність первинного ключа.

Тепер про зовнішні ключі:

- якщо сутність С пов'язує сутності А і В, то вона повинна включати зовнішні ключі, відповідні первинним ключам сутностей А і В
- якщо сутність У позначає сутність А, то вона повинна включати зовнішній ключ, відповідний первинному ключу сутності А

Цілісність (від англ. Integrity – недоторканність, збереженість, цілісність) – розуміється як правильність даних в будь-який момент часу. Але ця мета може бути досягнута лише в певних межах: СУБД не може контролювати правильність кожного окремого значення, що вводиться в базу даних (хоча кожне значення можна перевірити на правдоподібність).

Підтримка цілісності бази даних може розглядатися як захист даних від невірних змін або руйнувань (не плутати з незаконними змінами і руйнуваннями, які є проблемою безпеки). Сучасні СУБД мають ряд засобів для забезпечення підтримки цілісності (так само, як і засобів забезпечення підтримки безпеки).

Виділяють три групи правил цілісності:

- цілісність по сутностей
- цілісність по посиланнях
- цілісність, що визначається користувачем

Не допускається, щоб будь-який атрибут, який бере участь в первинному ключі, приймав невизначене значення.

Значення зовнішнього ключа повинно або:

- бути рівним значенню первинного ключа мети;
- бути повністю невизначеним, тобто кожне значення атрибуту, який бере участь в зовнішньому ключі має бути невизначеним. [13][1)]

Інфологічна модель бази даних цього проекту зображена на додатку А.

3.4 Проектування даталогічної моделі бази даних

Даталогічне проектування – створення схеми бази даних на основі конкретної моделі даних, наприклад, реляційної моделі даних. Для реляційної

[1)] [13] Реферат «Инфологическая модель баз данных сущность-связь» URL: https://www.kazedu.kz/referat/24093 (дата звернення 25.05.2019).

моделі даних даталогічна модель – набір схем відносин, зазвичай із зазначенням первинних ключів, а також «зв'язків» між відносинами, що представляють собою зовнішні ключі.

Будь-яка СУБД оперує з допустимими для неї логічними одиницями даних, а також допускає використання певних правил композиції логічних структур більш високого рівня зі складових інформаційних одиниць нижчого рівня. Крім того, багато СУБД накладають кількісні та інші обмеження на структуру бази даних. Тому перш ніж приступити до побудови даталогіческой моделі, необхідно детально вивчити особливості СУБД, визначити чинники, що впливають на вибір проектного рішення, ознайомитися з існуючими методиками проектування, а також провести аналіз наявних засобів автоматизації проектування, можливості і доцільності їх використання.

Хоча даталогічне проектування є проектуванням логічної структури бази даних, на нього впливають можливості фізичної організації даних, що надаються конкретною СУБД. Тому знання особливостей фізичної організації даних є корисним при проектуванні логічної структури.

Логічна структура бази даних, а також сама заповнена даними база даних є відображенням реальної предметної області. Тому на вибір проектних рішень найбезпосередніший вплив надає специфіка відображається предметної області, відображена в інфологічній моделі.

Кінцевим результатом даталогічного проектування є опис логічної структури бази даних на мові опису даних. Однак якщо проектування виконується «вручну», то для більшої наочності спочатку будується схематичне графічне зображення структури бази даних. При цьому повинно бути забезпечено однозначна відповідність між конструкціями мови опису даних і графічними позначеннями інформаційних одиниць і зв'язків між ними. Графічне представлення використовується і при автоматизованому проектуванні

структури бази даних як частиною інтерфейсу засіб спілкування з проектувальником, і при документуванні проекту.

Спроектувати логічну структуру бази даних означає визначити всі інформаційні одиниці і зв'язку між ними, задати їх імена; якщо для інформаційних одиниць можливе використання різних типів, то необхідно визначити їх тип. Слід також задати деякі кількісні характеристики, наприклад довжину поля.

Кожному типу моделі даних і кожного різновиду моделі, яку підтримує конкретної СУБД, притаманні свої специфічні особливості. Разом з тим є багато спільного у всіх структурованих моделях даних і принципах проектування БД в їх середовищі. Все це дає можливість використовувати єдиний методологічний підхід до проектування структури бази даних.

В БД відображається певна предметна область. Тому процес проектування БД передбачає попередню класифікацію об'єктів предметної області, систематизоване представлення інформації про об'єкти і зв'язки між ними.

На проектні рішення впливають особливості необхідної обробки даних. Тому відповідна інформація повинна бути певним чином представлена і проаналізована на початкових етапах проектування БД.

Дані про предметну область і особливості обробки інформації в ній фіксуються в інфологічної моделі. У такій моделі відображена вся інформація, що циркулює в інформаційній системі, але це зовсім не означає, що вся вона повинна зберігатися в базі даних. У зв'язку з цим одним з перших кроків проектування є визначення складу БД, тобто переліку тих показників, які доцільно зберігати в БД.

При проектуванні логічної структури БД здійснюються перетворення вихідної інфологічної моделі в модель даних, підтримувану конкретної СУБД, і

перевірка адекватності отриманої даталогіченої моделі відображається предметної області.

Для будь-якої предметної області існує безліч варіантів проектних рішень її відображення в даталогіченої моделі. Методика проектування повинна забезпечувати вибір найбільш підходящого проектного рішення.

Мінімальна логічна одиниця даних (незважаючи на їх різні назви) семантично для всіх СУБД однакова і відповідає або ідентифікатором об'єкта, або властивості об'єкта або процесу.

Зв'язки між сутностями предметної області, відображені в інфологічної моделі, можуть відображатися в даталогічній моделі або за допомогою спільного розташування відповідних їм інформаційних елементів, або шляхом оголошення зв'язку між ними.

Не всі види зв'язків, що існують в предметній області, можуть бути безпосередньо відображені в конкретній даталогічній моделі. Так, багато СУБД не підтримують безпосередньо відношення М: М між елементами. У цьому випадку в даталогічну модель вводиться додатковий допоміжний елемент, що відображає цей зв'язок (таким чином, ставлення М: М хіба що розбивається на два відносини 1: М між цим знову введеним елементом і вихідними елементами).

Слід звернути увагу на те, що відносини, які мають місце в предметній області, можуть бути передані не тільки за допомогою структури бази даних, але і програмним шляхом (тобто завжди існує альтернатива між декларативним і процедурним способом опису явища). Наприклад, при відображенні узагальнених об'єктів можна не виділяти підкласи на рівні логічної структури бази даних. В цьому випадку підкласи будуть виділятися програмним шляхом при обробці даних, що зберігаються.

Рішення про те, який із способів відображення (структурний / декларативний або програмний / процедурний) слід використовувати в кожному конкретному випадку, буде залежати від багатьох факторів, таких, як стабільність сутності, що відображається обсяг номенклатури, особливості СУБД, характер обробки даних та ін. Так, якщо в предметній області використовується класифікація співробітників по статі, в базі даних не слід створювати класифікатор статей, оскільки він буде містити всього дві позиції і ніколи не змінюється. Як правило, не слід виділяти за цією ознакою і відповідні підкласи для об'єктів предметної області, тому що в більшості випадків вони обробляються спільно і в основному мають однаковий набір властивостей, що характеризує їх. Однак в деяких предметних областях поділ на такі підкласи може бути доцільним. Якщо ж відображається сутність не стабільна, то її краще передавати за допомогою даних, так як в противному випадку буде часто турбуватися перетворення програми, що зазвичай забезпечити важче, ніж зміна даних.

При відображенні узагальнених об'єктів в БД можливі різні варіанти: зберігати всю інформацію про всі узагальнені об'єкті в одному файлі / таблиці, кожному підкласу об'єктів нижчого рівня виділяти окремі самостійні файли / таблиці. І перший, і другий варіанти широко використовуються у різноманітних СУБД. У першому випадку підкреслюється спільність об'єктів різних підкласів, що входять в узагальнений об'єкт. У другому випадку, навпаки, узагальнений об'єкт як єдине ціле не відображається в структурі бази даних.

Інші способи відображення пов'язані з явним або неявним виділенням підкласів в логічній структурі БД. Неявне виділення підкласу полягає в тому, що в запису відводяться поля для фіксації значень властивостей, загальних для об'єктів різних підкласів, і значення ознаки підкласу, а замість полів, наявність яких залежить від підкласу, використовується одне поле зі змінним складом,

зміст якого буде залежати від того, до якого підкласу відноситься описуваний об'єкт. Реалізація принципу явного виділення підкласів в структурі БД істотно залежить від специфіки СУБД.

При проектуванні логічної структури БД основне значення має специфіка предметної області, що відображається. Однак, як зазначалося вище, і характер обробки інформації впливає на прийняте проектне рішення. Наприклад, рекомендується зберігати разом інформацію, часто оброблювану спільно, і, навпаки, розділяти по різних файлах інформацію, не що використовується одночасно. Інформацію, яку використовують часто, і інформацію, частота звернення до якої мала, також слід зберігати в різних файлах, причому останню може виявитися вигідним винести в архівні файли, а не підтримувати в складі БД.

Як зазначалося вище, описується не окремий об'єкт, а клас об'єктів. Але в окремих випадках буває, що клас включає в себе тільки один екземпляр. Наприклад, якщо предметною областю є якийсь конкретний інститут, то клас об'єктів буде містити тільки один екземпляр об'єкта. Таким «виродженим» класами об'єктів зазвичай не ставиться у відповідність окремий файл бази даних[12][1)].

На основі інфологічної моделі з попереднього пункту була спроектована даталогічна модель.

У якості системи керування базою даних була обрана SQLite – компактна вбудовувана реляційна база даних, що поставляється з вихідними кодами. Вперше випущена в 2000 році, призначена для надання звичних можливостей реляційних баз даних без властивих їм накладних витрат. За час експлуатації встигла заслужити репутацію як переносима, легка у використанні, продуктивна і надійна база даних.

[1)] [12] Даталогическое проектирование URL: http://wiki.mvtom.ru/index.php/Даталогическое_проектирование (дата звернення 22.05.2019).

Слово «вбудована» (embedded) означає, що база даних існує не як процес, окремий від обслуговується процесу, а є його частиною – частиною деякого прикладного застосування. SQLite не використовує парадигму клієнт-сервер, вона являє собою бібліотеку, з якої програма компонується, і SQLite стає складовою частиною програми. Таким чином, в якості протоколу обміну використовуються виклики функцій (API) бібліотеки SQLite. І клієнт, і сервер працюють в одному процесі – це позбавляє від проблем конфігурації. Все, чого потребує програміст, вже скомпільовано в його додатку. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому пристрої, на якому виповнюється програма. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції запису весь файл, який зберігає базу даних, блокується; ACID-функції досягаються в тому числі за рахунок створення файлу журналу.

Кілька процесів або потоків можуть одночасно без будь-яких проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, якщо ніякі інші запити в даний момент не обслуговуються; в іншому випадку спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

У комплекті поставки йде також функціональна клієнтська частина у вигляді виконуваного файлу sqlite3, за допомогою якого демонструється реалізація функцій основної бібліотеки. Клієнтська частина є кросплатформною утилітою командного рядка.

Завдяки архітектурі SQLite її можливо використовувати як на вбудовуваних системах, так і на виділених машинах з великими масивами даних.

SQLite підтримується динамічна типізація даних. Підтримуються такі типи даних:

- NULL – NULL-значення;
- INTEGER – цілочисельне значення зі знаком;
- REAL: число з плаваючою комою;
- TEXT: текстовий рядок з кодуванням UTF-8, UTF-16BE або UTF-16LE;
- BLOB: тип даних, що зберігається точно в такому ж вигляді, в якому і був отриманий.

На поточному ринку вбудовуваних баз даних представлено багато продуктів від різних виробників, але тільки один з них поставляється з відкритим кодом, не вимагає ліцензійних зборів і спроектований виключно як вбудовувана БД – це SQLite.

SQLite має модульну архітектуру, яка відображає унікальні підходи до управління реляційними базами даних. Вісім окремих модулів згруповані в три головних підсистеми (рис. 3.1). Вони поділяють обробку запиту на окремі завдання, які працюють подібно конвеєру. Верхні модулі компілюють запити, середні виконують їх, а нижні працюють з диском і взаємодіють з операційною системою.

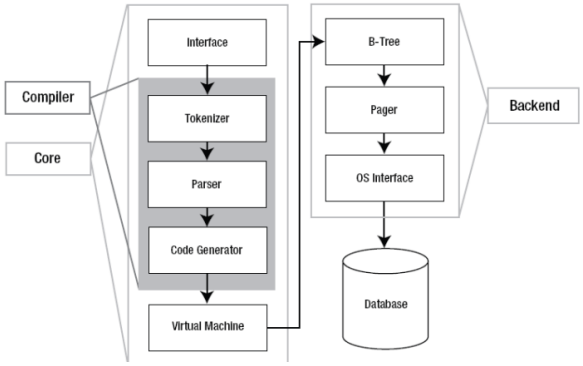


Рисунок 3.1 – Архітектура SQLite

Незважаючи на маленький розмір, SQLite надає великий спектр особливостей і можливостей. Він підтримує вельми повний набір стандарту ANSI SQL92 для особливостей мови SQL, а також такі особливості як тригери, індекси, стовпці з автоінкрементом та LIMIT / OFFSET особливості. Так само підтримуються такі рідкісні властивості, як динамічна типізація і вирішення конфліктів. Даталогічна модель наведна у додатку Б.

Використовуючи цю систему керування базами даних, була створена база даних, яка має таблиці, наведені на рис. 3.2 – 3.9.

Таблиця «Hero» містить у собі інформацію про головного героя. Структура даної таблиці наведена на рис. 3.2.

Рисунок 3.2 – Таблиця «Hero»

Таблиця «Enemy» містить у собі інформацію про ворогів. Структура даної таблиці наведена на рис. 3.3.

Рисунок 3.3 – Таблиця «Enemy»

Таблиця «Level» містить у собі інформацію про рівень, на якому знаходиться герой (рис. 3.4).

Рисунок 3.4 – Таблиця «Level»

Таблиця «Weapon» містить у собі інформацію про зброю (рис. 3.5).

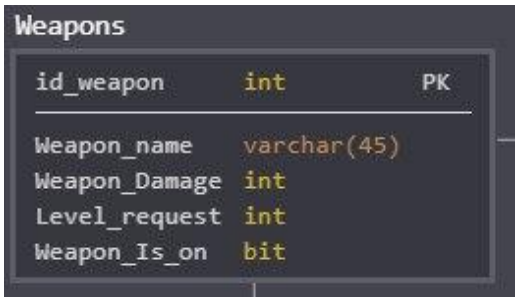


Рисунок 3.5 – Таблиця «Weapon»

Таблиця «Armor» містить у собі інформацію про броню. Структура даної таблиці наведена на рис. 3.6.

Рисунок 3.6 – Таблиця «Weapon»

Таблиця «Potions» містить у собі інформацію про зілля. Структура даної таблиці наведена на рис. 3.7.

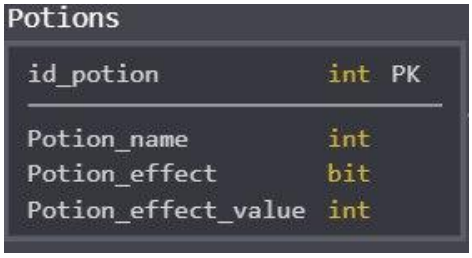


Рисунок 3.7 – Таблиця «Weapon»

Таблиця «Hero_Level» містить у собі інформацію про рівень, який на даний момент є у героя. Структура даної таблиці наведена на рис. 3.8.

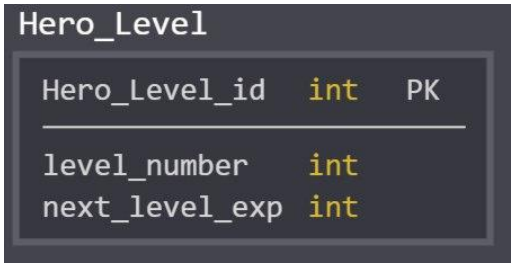


Рисунок 3.8 – Таблиця «Hero_Level»

Таблиця «Loot» містить у собі інформацію про речі, знайдені героєм. Структура даної таблиці наведена на рис. 3.9.

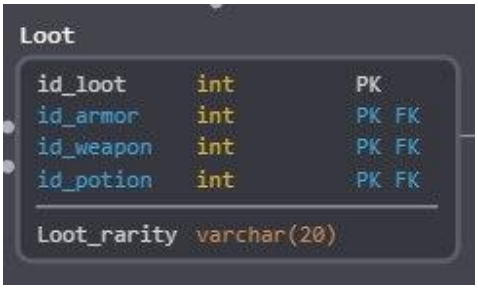


Рисунок 3.9 – Таблиця «Loot»

3.2 Створення головного меню ігрового програми

Для того, щоб створити меню в Unity не буде потрібно багато часу. Для початку створимо ще одну сцену і назвемо її Menu. Для створення елементів інтерфейсу в Unity будемо використовувати такий компонент як Canvas, він є головним на сцені елементом і всі інші елементи, такі як кнопки, панелі і так далі створюються як дочірні до нього. Canvas відображається на сцені як великий прямокутник, він дозволяє легко керувати позиціонуванням елементів на сцені і має свою Event System. Елементи призначеного для користувача інтерфейсу відображаються на сцені в тій же послідовності, в якій вони представлені у вікні ієрархії.У головному меню на даному етапі розробки нам потрібно тільки два елементи, це кнопка Play, яка дозволить нам запускати тестовий рівень ігрового програми і кнопка Quit для виходу. Додати кнопку можна натиснувши правою кнопкою миші на сцені і вибравши в контекстному меню розділ UI Button (рис. 3.10).

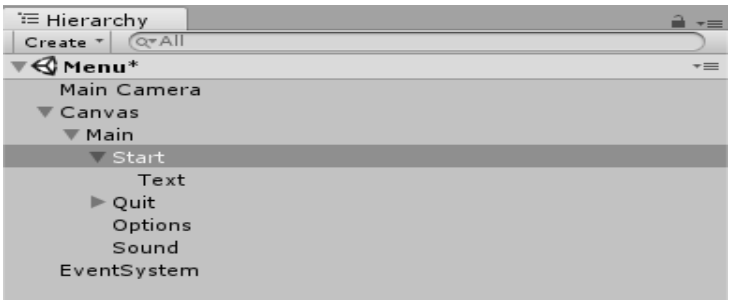


Рисунок 3.10 Створення елементів інтерфейсу

Після того як кнопки додані, ми можемо вибрати одну з них і подивитися які елементи управління вона має у вікні інспектора. У першій верхній частині вікна знаходяться елементи керування положенням кнопки, управління її розмірами, зумом, так само є можливість змінювати шар в якому ця кнопка буде

відображатися (в нашому випадку всі елементи меню будуть лежати в шарі UI) і кут повороту. Далі йдуть елементи управління кольором кнопки, її текстурою і матеріалом якщо такі є і способом їх відображення[10][1)].

Натиснувши на опцію Source Image, виберемо текстуру кнопки, яка буде відображатися на ній за замовчуванням. Ця текстура буде відповідати стану кнопки, коли над нею немає курсору (тобто вона не в фокусі) і вона не в натиснутому стані.

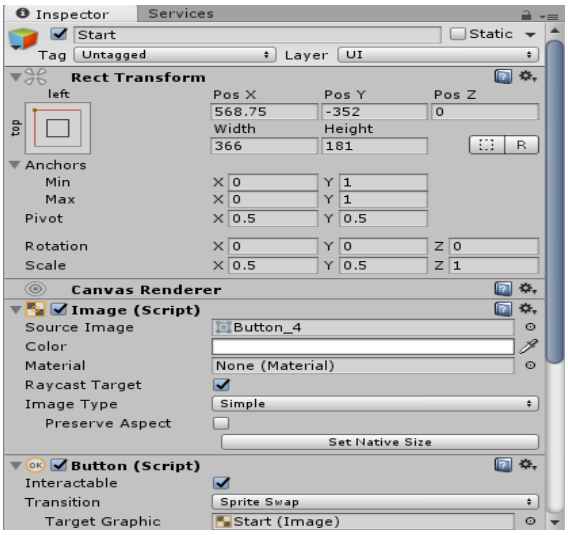


Рисунок 3.11 Кнопка – Інспектор

Після того як ми вибрали основну для кнопки текстуру, необхідно перейти до наступної частини вікна інспектора (рис. 3.11). У ній нас цікавить спосіб зміни текстур кнопки в залежності від її стану. Ця опція в Unity називається Transition і вона має кілька станів:

– None означає, що ніякої візуальної реакції на зміну станів кнопки не буде;

[1)] [10] Free winter platformer game tileset.URL: http://www.gameart2d.com/winter-platformer-game-tileset.html (дата звернення 24.04.2019).

– Color Tint - в залежності від стану буде змінюватися колір кнопки;
– Sprite swa - кожномустаном буде відповідати своя текстура;
– Animation - стану кнопки будуть анімованими.

У нашому випадку найбільше підходить варіант Sprite swap, так як у нас вже є заздалегідь заготовлені текстури для всіх станів кнопки. Після того як ми виберемо цей режим з'являться додаткові опції з яких нас цікавлять (рис. 3.12):

– Highlighted Sprite - текстура, яка буде завантажуватися кнопкою, в разі якщо вона у фокусі, тобто курсор знаходиться над нею;
– Pressed Sprite - текстура відповідна кнопці;
– Disabled Sprite - якщо кнопка має статус "виключена", то буде завантажена ця текстура;

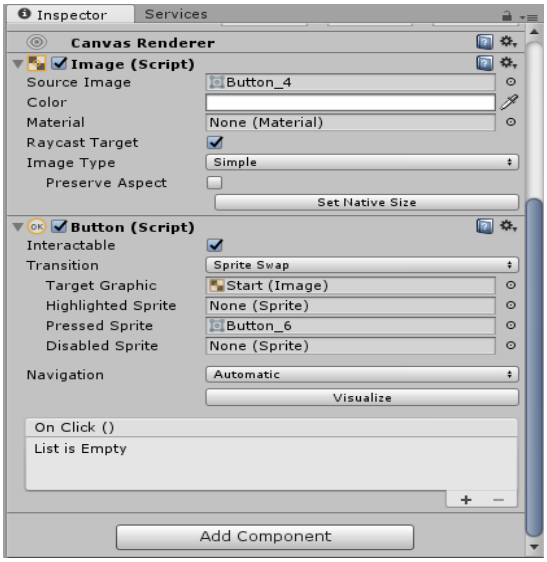


Рисунок 3.12 Властивості кнопки в інспектора

Так само в цьому вікні можна вказати метод, в який буде передаватися управління, в разі якщо кнопку натиснули.

У об'єкта Button автоматично створюється нащадок Text, який ми будемо використовувати для створення написів на поверхні текстур кнопок. Для того

щоб скористатися широкими можливостями цього об'єкта, ми повинні вибрати його в вікні ієрархії, у кожної кнопки буде свій індивідуальний об'єкт Text. Перейшовши у вікно інспектора, ми зможемо вказати використовуваний на наших кнопках шрифт (рис. 3.13) (ми будемо використовувати Carbon block), відрегулювати його розмір, налаштувати вирівнювання та колір[11][1)].

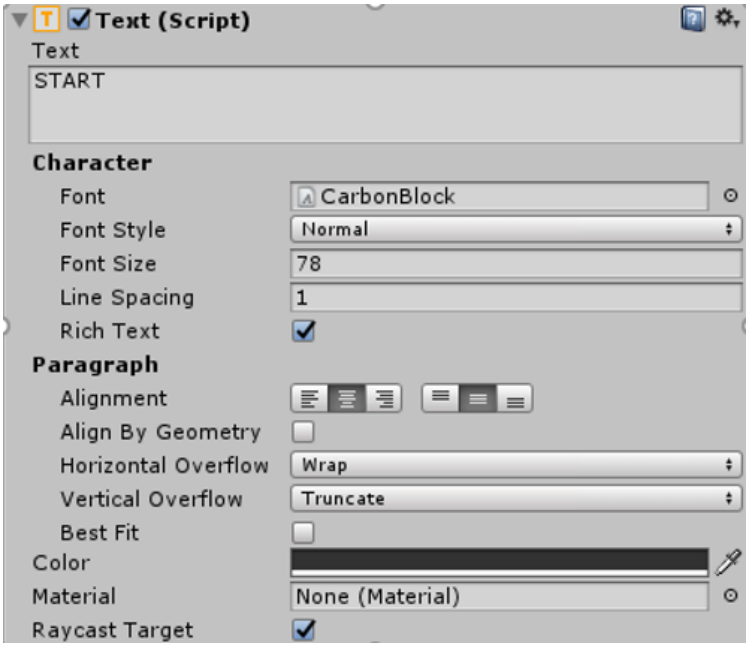


Рис.3.13 Текст на кнопці

Після того як ми створили першу кнопку, необов'язково проробляти всю процедуру створення і налаштування для інших кнопок. можна просто скопіювати вже створений об'єкт, задати йому нове ім'я, інший текст і текстуру (рис. 3.14).

Використання копіювання, у разі коли потрібно створити кілька схожих за властивостями об'єктів, сильно спрощує процес створення програми та скорочує тимчасові витрати на розробку.

[1)] [11] Форум для разработчиков игр. URL: http://www.gamedev.ru/forum/ (дата звернення 25.04.2019).

Рисунок 3.14 – Сцена за меню

Отже, ми створили меню. Тепер до кнопок цього меню потрібно прив'язати дії. Для це ми створимо новий скрипт і назвемо його NewGame. Додамо туди функції для старту гри і виходу. Для завантаження сцен в Unity служить Scene Manager, його ми будемо використовувати для завантаження головної сцени гри. На початковому етапі у нас буде всього один рівень, тому більше нічого нам не знадобиться.

```
public static void NewGame()
{
    Application.LoadLevel(“GameScene”)
}
```

Далі ми створюємо порожній ігровий об'єкт, назвемо його mainMenuObj і призначаємо йому наш скрипт (рис. 3.15).

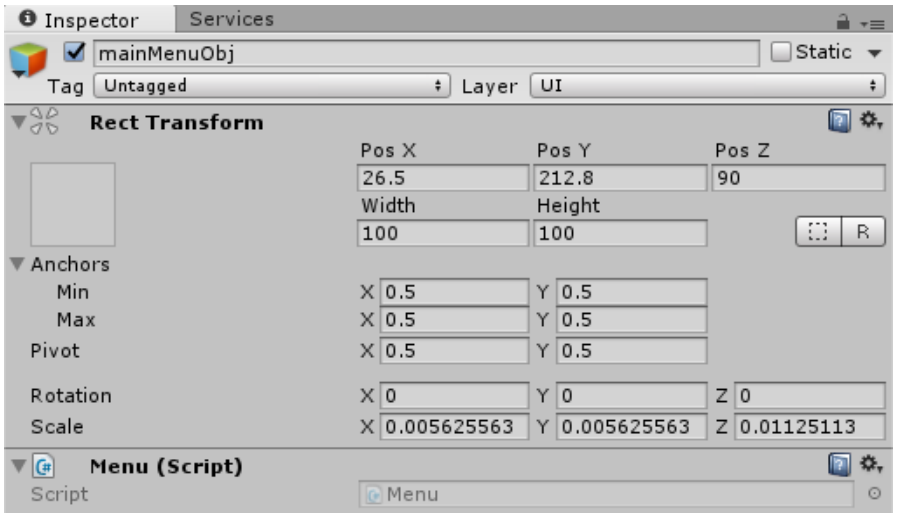


Рисунок 3.15 - Порожній об'єкт

У властивостях створеної нами кнопки в інспектора можна задати дію, яке буде відбуватися при її натисканні, додаємо туди наш порожній об'єкт і функцію, яка буде викликатися при натисканні, в даному випадку, це playButton (рис. 3.16).

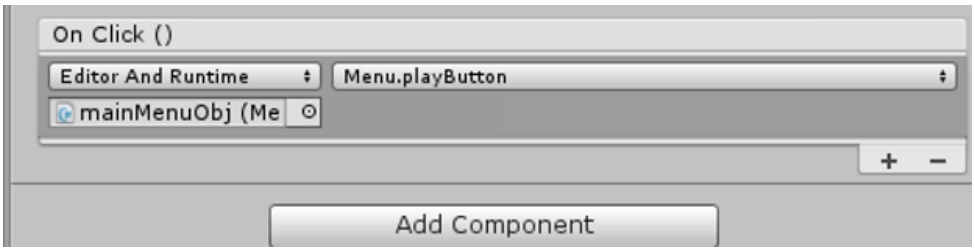


Рис.3.16 - Створення події

Тепер при натисканні кнопки Start буде завантажуватися сцена з першим рівнем гри. Дії для інших кнопок меню аналогічні цим, створюємо функцію в скрипті Menu, відповідну кнопці, і призначаємо її в інспектора у властивостях цієї кнопки.

3.3 Створення карти

Ландшафт буде виконаний у вигляді карти, тобто реєстрового зображення з розставленими на ньому коллайдера. У ролі коллайдеров будем використовувати вбудований компонент 2D box collider.

Для карти будемо використовувати asset, який можна взяти на вебсайті вільного ігрового арту[12][1)]. Він представляється собою набір тайлів або елементів карти (рис. 3.17).

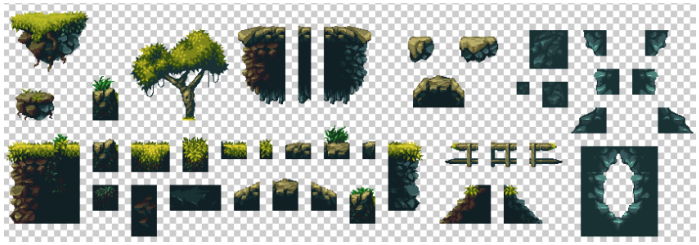


Рисунок 3.17 – Asset для створення карти

Саму карту можна підготувати в будь-якому доступному графічному редакторі (наприклад: paint.net) з довільно виконаним ландшафту (рис. 3.18).

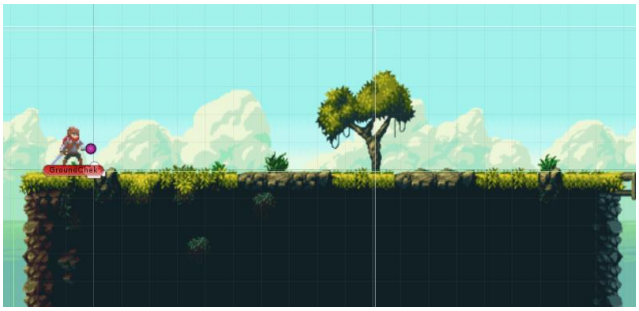


Рисунок 3.18 – Підготовленна карта ландшафту

[1)] [12] Itch.io. URL: https://ansimuz.itch.io/magic-cliffs-environment (дата звернення 28.04.2019).

Після створення карти поміщаємо її на сцену. Тепер за допомогою коллайдеров покажемо які місця в лабіринті будуть стінами і підлогою. Для цього додамо компонент 2D box collider, так як таких непрохідних поверхонь в лабіринті буде багато, то, відповідно, і компонентів теж треба буде додати стільки ж (рис. 3.19).

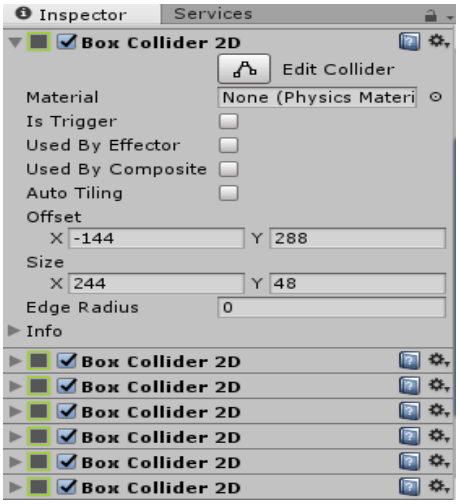


Рисунок 3.19 Налаштування коллайдера

Після того як вони додані, потрібно зайти в кожен компонент, натиснути кнопку Edit bounding volume і виділити області, які будуть займати наші поверхні. Колайдери, використовуючи виділені інформуватимуть Unity про попадання в них об'єкта. Тепер при завантаженні рівня ми можемо запросити у Unity список всіх коллайдеров і використовувати їх у своїй функції перевірки зіткнень.

ВИСНОВКИ

При виконанні даної випускної роботи було спроектовано і розроблено ігрове додаток з використанням кроссплатформенного движка Unity. Цей додаток не є закінченим продуктом, так як вимагає наповнення професійно виконаним графічним контентом, але при відповідному доопрацюванні може зайняти свою нішу на ринку ігор.

У даній роботі були розглянуті основні аспекти розробки ігрових додатків. Движок Unity показав себе як зручний, простий в освоєнні і професійно виконаний інструмент для створення ігор.

Крім того були розробленні шнфологічна та даталогічна моделі бази даних.

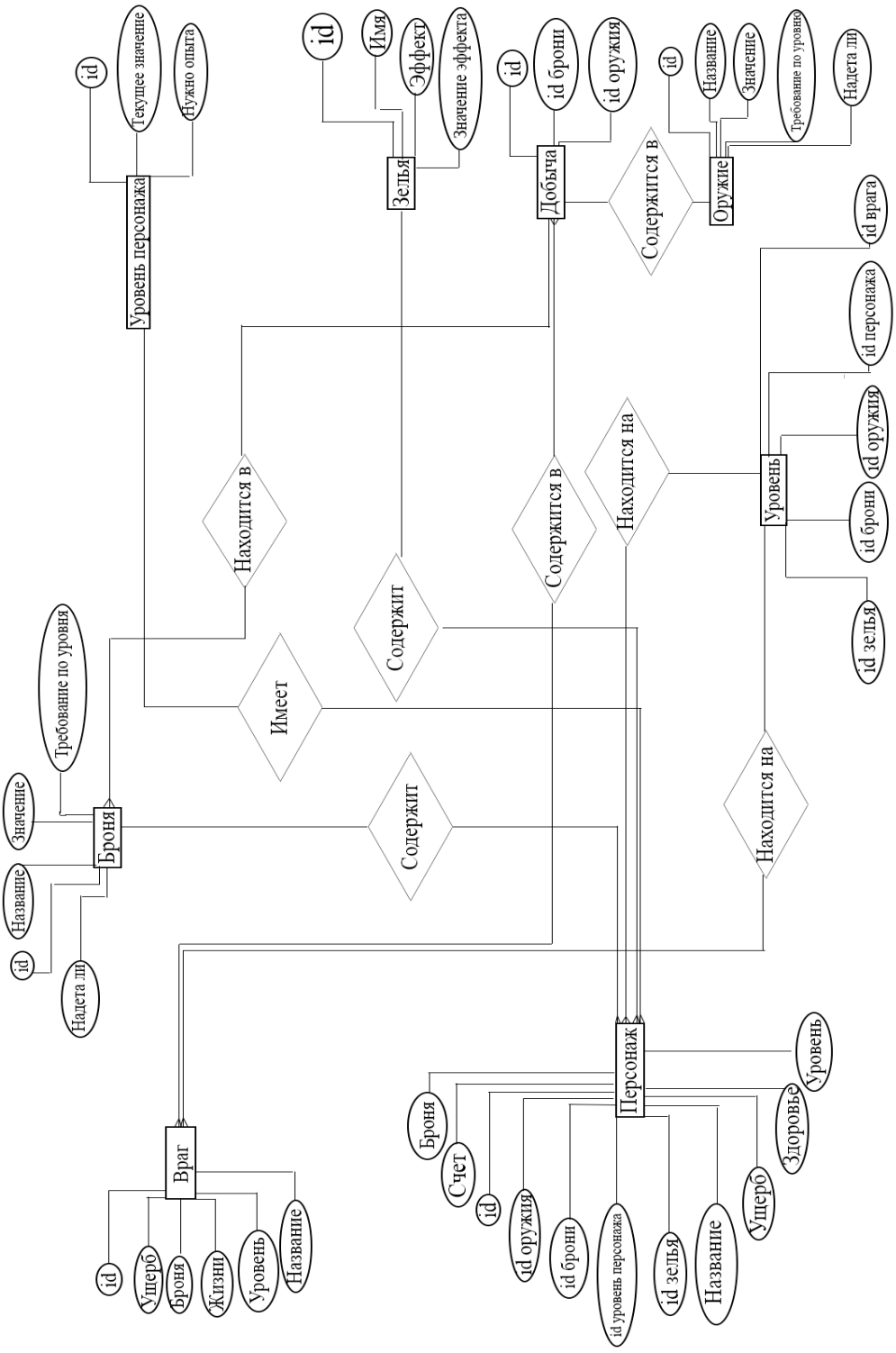
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. DevGam сравнение игровых движков URL: http://devgam.com/sravnenie-igrovyx-dvizhkov-kakoj-vybrat (дата звернення 12.01.2019).
2. List of game engines and frameworks URL: https://en.wikipedia.org/wiki/List_of_game_engines (дата звернення 21.03.2019).
3. List of game engines URL: https://80.lv/articles/the-most-comprehensive-list-of- game-engines/ (дата звернення 27.03.2019).
4. Unity store URL: https://store.unity.com/ru?_ga=2.200322542.1765686190.1560285797-130111505.1547668569 (дата звернення 01.03.2019).
5. Руководство Unity URL: https://docs.unity3d.com/ru/current/Manual/index.html (дата звернення 18.02.2019).
6. Русскоязычный форум Unity3d URL: http://unity3d.ru (дата звернення 28.01.2019).
7. Видео уроки по Unity3D URL: https://www.youtube.com/watch?v=m0sKo81DZME (дата звернення 16.04.2019).
8. Хокинг Д. Unity in Action: Multiplatform Game Development in C# with Unity 5 : Manning Publications Company. 2015 – 352с (дата звернення 17.04.2019).
9. Руководство по программированию на C# URL: https://docs.microsoft.com/ru- ru/dotnet/articles/csharp/programming-guide/index (дата звернення 19.04.2019).

10. Free winter platformer game tileset.URL: http://www.gameart2d.com/winter-platformer-game-tileset.html (дата звернення 24.04.2019).
11. Форум для разработчиков игр. URL: http://www.gamedev.ru/forum/ (дата звернення 25.04.2019).
12. Даталогическое проектирование URL: http://wiki.mvtom.ru/index.php/Даталогическое_проектирование (дата звернення 22.05.2019)
13. Реферат «Инфологическая модель баз данных сущность-связь» URL: https://www.kazedu.kz/referat/24093 (дата звернення 25.05.2019).

ДОДАТОК А

Інфологічна модель бази даних



ДОДАТОК Б

Даталогічна модель бази даних