МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

СОЛОМАХА Тарас Тарасович

Програмне забезпечення візуалізації будови атома засобами Unity 3D/ Software for visualizing the Atom content using Unity 3D

напрям підготовки: 6.050103 - Програмна інженерія
фахове спрямування - Програмне забезпечення систем

Бакалаврська дипломна робота

Виконав студент групи ПЗС-41
Т. Т. Соломаха
_____

Науковий керівник:
к.т.н., доцент ПІГОВСЬКИЙ Ю.Р.
_____

Бакалаврську дипломну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2016

ВСТУП

У сучасних освітніх установах велика увага приділяється комп'ютерному супроводу професійної діяльності. У навчальному процесі використовуються навчальні і тестуючі програми з різних дисциплін.

Застосування мультимедійних засобів на заняттях дозволяє підвищити не тільки інтерес до майбутньої спеціальності, але і успішність по даній дисципліні. Навчальні комп’ютерні програми і електронні підручники дають можливість кожному студенту незалежно від рівня його підготовки брати активну участь у навчальному процесі, індивідуалізувати свій процес навчання, здійснювати самоконтроль.

Бути не пасивним спостерігачем, а активно отримувати знання і оцінювати свої можливості. Саме тому об’єктом даної дипломної роботи є проектування системи для відображення будови атома.

Акцентом розробки даного програмного забезпечення є те, що користувач отримує естетичне задоволення від процесу перегляду атома. Задоволення, що отримується в процесі сприяє формуванню стійкого інтересу до навчального процесу, який при грамотному впровадженню, призводить до накопичення знань в учнів. Основне завдання, яке ставиться при створенні такого продукту, — здійснити перетворення реального об´єкта вивчення у візуальну інформацію, яка засвоюється набагато краще. Тобто, засоби навчання описують об´єкт вивчення або створюють його модель, виділяють предмет вивчення і представляють його для засвоєння. В нашому випадку ця модель буде зображати будову атома.

Атом, найдрібніша частка речовини, яка може вступати в хімічні реакції. Фізична модель атома складається з щільного ядра з позитивно заряджених протонів і нейтральних нейтронів. Ядро оточене в рази більшою за розміром оболонкою з негативно заряджених електронів. Якщо кількість протонів дорівнює кількості електронів атом є електрично нейтральним в іншому випадку атом стає іоном.

Інструментом для розробки вибрано Unity 3D, оскільки цей ігровий движок володіє широким спектром інструментів, що дозволять красиво візуалізувати будову атома.

Розділ 1. Аналіз предметної області будова атома

1.1. Коротка характеристика об'єкту управління будова атома

Освітнє програмне забезпечення — вид програмного забезпечення, головним призначенням якого є навчання або розвиток деяких навиків. Принципи вчення, застосовані в таких програмах, можуть бути абсолютно різними. Це може бути: гра, тест, середовище програмування і так далі Вікова аудиторія користувачів такого програмного забезпечення займає щонайширший діапазон (від 3 років і вище).

Історія створення освітнього програмного забезпечення (ПЗ) бере свій початок з 40-х років XX століття, коли американські розробники симуляторів польоту використовували аналоговий комп'ютер для створення імітації показаних бортових систем літака.

До середини 70-х років освітнє ПЗ безпосередньо було пов'язано з мейнфреймами, на яких воно і виконувалося. У ці роки основоположниками освітніх комп'ютерних систем були: система PLATO(1960), розроблена в Університеті Ілінойса, і система TICCIT (1969). Ціна цих ранніх терміналів перевищувала 12.000 доларів, і інститути не могли собі дозволити придбати їх із-за високої ціни. Мови програмування того часу, такі як Logo і BASIC, теж можна розглядати як освітні системи, розраховані на студентів і недосвідчених користувачів комп'ютерів.

В залежності від поставленої задачі, складності програмної реалізації та інших факторів до електронних засобів навчання можна віднести:

Навчальні програми

електронні таблиці;

електронні бібліотеки;

презентації;

тестові завдання;

віртуальні лабораторні роботи;

операційні системи;

бази даних;

відео курси;

інше.

Програмне забезпечення візуалізації будови атома повинно могти відображати нейтрони, протони та електрони атома та надавати можливість динамічної взаємодії з інтерфейсом. Всі ці елементи повинні могти зацікавити користувача, тому вирішено реалізувати відображення будови атома в трьох-вимірному просторі.

Програмне забезпечення візуалізації будови атома засобами Unity 3D дозволяє:

- Відобразити будову атома в 3D просторі
- Показати рух електронів
- Керувати кутом обзору атома
- Вибирати різні атоми

1.2. Опис предметної області будова атома

Перед нами постає задача створення модуля користувача для системи перегляду атома. Метою розробки є спрощення процесу навчання . Для досягнення цієї мети потрібно визначити, які основні задачі повинен виконувати даний продукт. Для правильного проектування комплексу комп’ютерно-інформаційної системи необхідно чітко визначити функції які він повинен виконувати і в результаті декомпозиції проблеми обрати найбільш оптимальний варіант.

Для того, щоб повністю представити функціонал програмної системи, виділено такі основні бізнес-процеси:

- процес вибору атома;
- процес перегляду атома;

На рисунку 1.1 зображено діаграму бізнес-процесів.

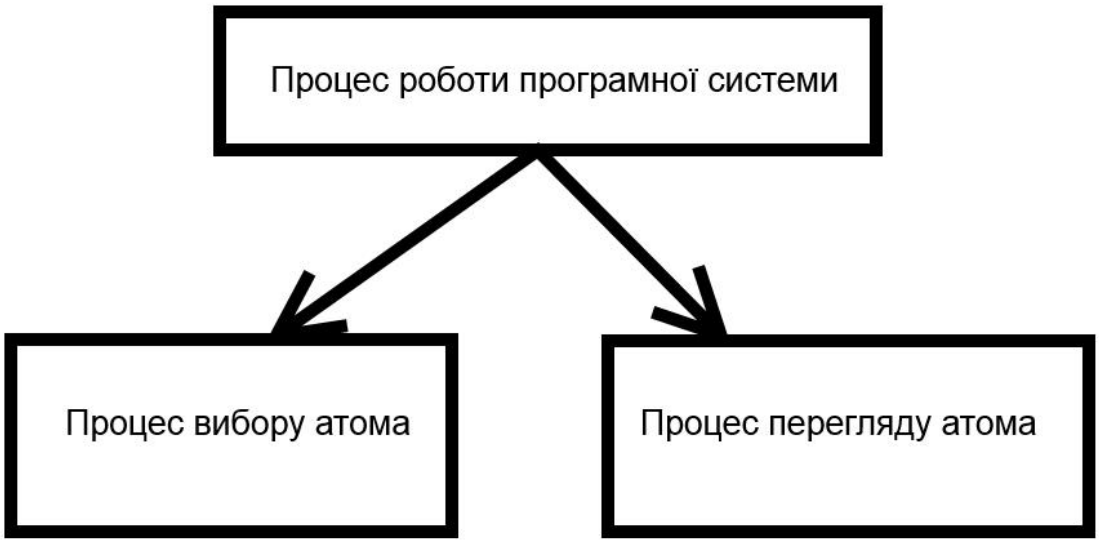

Рисунок 1.1 – Діаграма бізнес-процесів розроблюваного програмного продукту

Тепер детальніше розглянемо кожен з вище представлених бізнес-процесів.

Першим що потрібно зробити користувачеві це обрати атом для подальшого перегляду.

Характеристику бізнес-процес вибору атома наведено в таблиці 1.1.

Таблиця 1.1.

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Вибір атома
Основні учасники	Користувач
Вхідна подія	Запит на завантаження атома
Вхідні документи	Таблиця
Вихідна подія	-
Вихідні документи	-
Клієнт бізнес-процесу	-

Процес перегляду атома є допоміжним процесом, адже він дозволяє переглянути атом під різним кутом. Він поділяється на такі частини:

- Scrolling;
- Rotation.

На рисунку 1.2 зображено процес перегляду атома.

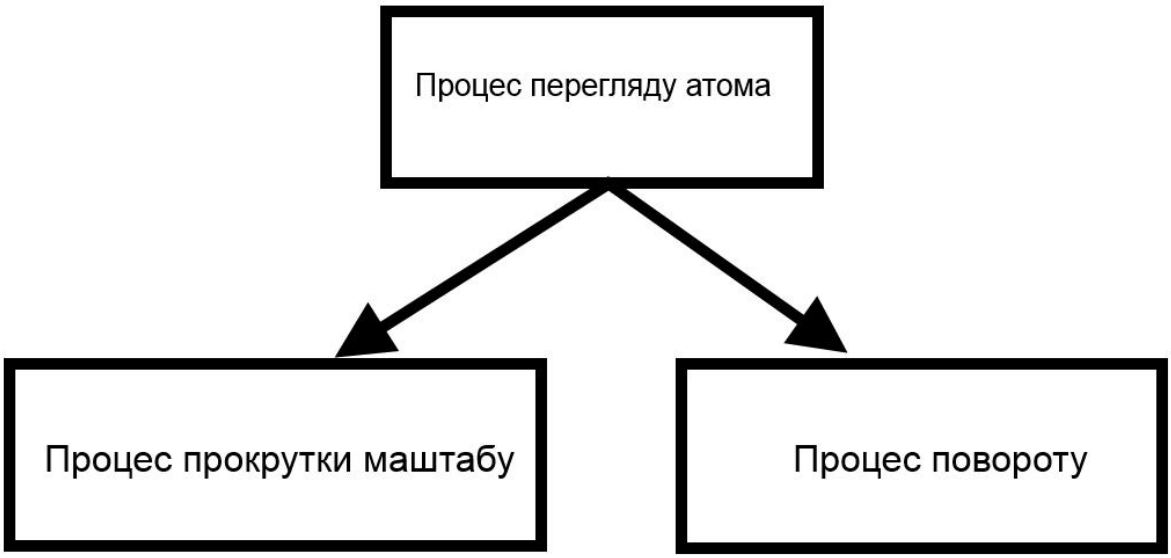

Рисунок 1.2 – Процес перегляду атома.

Характеристику бізнес-процесу перегляду атома наведено в таблиці 1.2.

Таблиця 1.2.

Назва характеристики	Значення характеристики
Ім'я бізнес-процесу	Перегляд атома
Основні учасники	Користувач
Вхідна подія	Запит на перегляд атома
Вхідні документи	-
Вихідна подія	Відображена атома
Вихідні документи	-
Клієнт бізнес-процесу	-

1.3. Огляд і аналіз існуючих аналогів

На сьогодні є достатньо програмних систем, за допомогою яких можна змоделювати будову атома. В даному пункті будуть розглядатися кілька з таких.

Розглянемо для початку програму “ Build an Atom ” (рис 1.3).

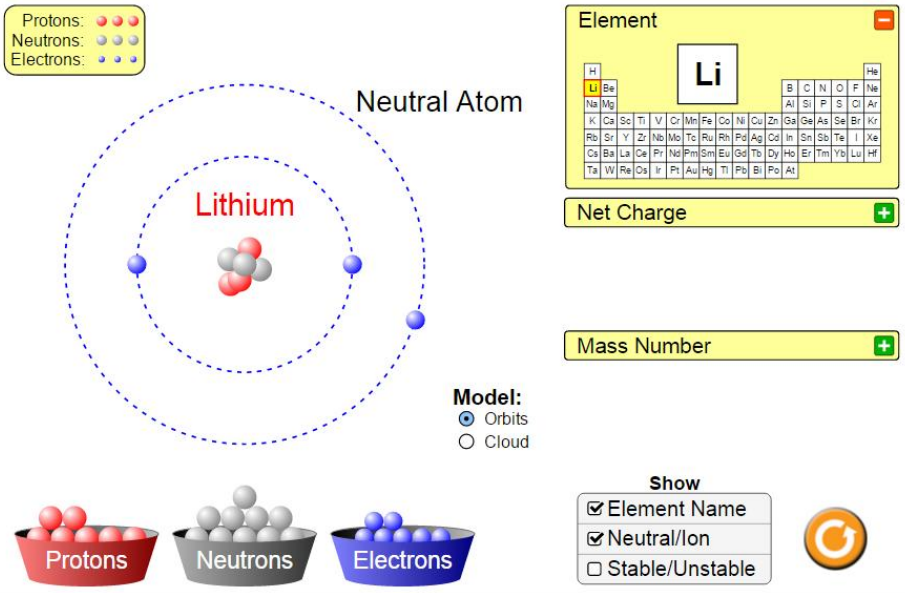

Рисунок 1.3 – Вікно створення атому у програмі «Build an Atom»

Дана програма має дуже простий але та доволі зручний інтерфейс, за допомогою якого користувач може досить легко зорієнтуватися в подальших діях.

Дана програмна система має наступні функції:

- Створення атома.
- Перегляд атома.
- Тестування.

Для порівняння розглянемо програмний продукт Atomic Basics: Interactive (рис 1.4).

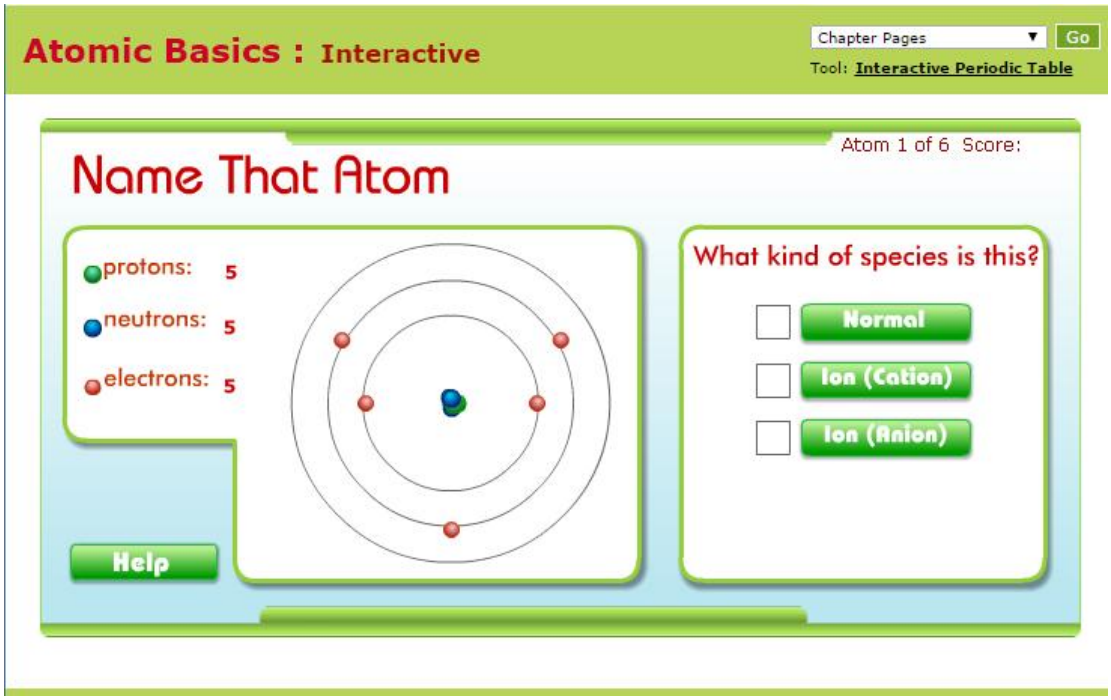

Рисунок 1.4 – Сторінка програми Atomic Basics: Interactive

Даний продукт дозволяє проходити тести та в процесі переглядати будову атома. Також цей продукт має два модуля: модуль тестів, де можна проходити завдання , та модуль з таблицею атомів.. Даний продукт має багато різноманітних функцій, таких як:

- Проходження тестів.
- Перегляд таблиці.
- Відображення даних про атом.

Оглянувши та зробивши аналіз існуючих аналогів можна зробити висновок, що вище перелічені програмні системи є хорошими для проведення різноманітних тестів але лише одна з них може відображати атом який обирає користувач а також вони є досить старі і зовнішній вигляд не розпалює інтерес. Тому, оптимальним рішенням даної проблеми є створення продукту.

Здійснивши аналіз альтернатив, створено таблицю, де відображається порівняльна характеристика аналогів.

Таблиця 1.3

Фірма-розробник	Kelly Lancaster (lead)	Annenberg Learner
Назва програмного продукту	Build an Atom	Atomic Basics: Interactive
Версії продукту	1.1.2	1.5
Функціональність	• Можливість створення атома • Можливість перегляду будови атома • Можливість проходження тестів.	• Можливість перегляду будови атома • Можливість проходження тестів. • Можливість перегляду даних з таблиці атомів.
Допомога користувачу	Присутня система довідки	Присутня система довідки

1.4. Специфікація вимог до модуля (системи)

Специфікація вимог для програмної системи - це повний опис поведінки системи що розробляється. Вона включає множину прецедентів які описують всі взаємодії, які користувачі мають з програмним забезпеченням. Прецеденти також відомі як функціональні вимоги. На додачу до прецедентів також включає нефункціональні вимоги. Нефункціональні вимоги є вимогами які накладають обмеження на проект, чи реалізацію. Специфікація вимог до системи включає: глосарій проекту, опис варіантів використання.

У таблиці 1.4 подано глосарій основних використовуваних термінів при створенні модуля «Користувач» системи перегляду атома.

Таблиця 1.4

Термін	Опис терміну
1. Основні поняття та категорії предметної області та проекту	
База даних	Впорядкований набір логічно взаємопов'язаних даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів. У технічному розумінні включно й система керування БД.
Таблиця бази даних	В реляційних базах даних і плоских базах даних, таблиця це набір елементів даних (значень), які організовані з використанням моделі вертикальних стовпців (з різними іменами) і горизонтальних рядків. Таблиця має визначену кількість стовпців, в той час як кількість рядків може різнитися в різні моменти. Кожен рядок ідентифікується особливим набором колонок який називається потенційним ключем.
Реляційна база даних	База даних, основана на реляційній моделі даних. Слово «реляційний» походить від англ. relation. Для роботи з реляційними БД застосовують реляційні СКБД. Інакше кажучи, реляційна база даних — це база даних, яка сприймається користувачем як набір нормалізованих відношень різного ступеню.
SQL (Structured query language)	Декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних і її модифікації, системи контролю за доступом до бази даних.
Модель «сутність-зв'язок» (ER-модель)	Модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. ER-модель — це мета-модель даних, тобто засіб опису моделей даних.

Система керування базами даних	Комп'ютерна програма чи комплекс програм, що забезпечує користувачам можливість створення, збереження, оновлення, пошук інформації та контролю доступу в базах даних.
Перегляд атома	Це процес відображення будови атома а саме його електронів нейтронів та протонів.
2. Користувачі системи	
Користувач	особа, що використовує модуль «Користувач» системи перегляду атома для навчання.
3. Вхідні та вихідні документи	
Запит	Це формулювання своєї інформаційної потреби користувачем деякої бази даних. Для складання запиту використовується мова пошукових запитів
Дані	Це інформація (найчастіше цифрова), подана у формалізованому вигляді, прийнятному для обробки автоматичними засобами за можливої участі людини. Дані - інформація, одержана в експерименті, взята з опублікованих праць чи отримана в результаті розрахунків. При цьому мають бути точно описані умови їх отримання та способи розрахунків. Представляються та організовуються у спосіб зручний для подальшої обробки та аналізу.

Далі, наведемо діаграму варіантів використання для акторів системи (користувач) на рисунку 1.5.

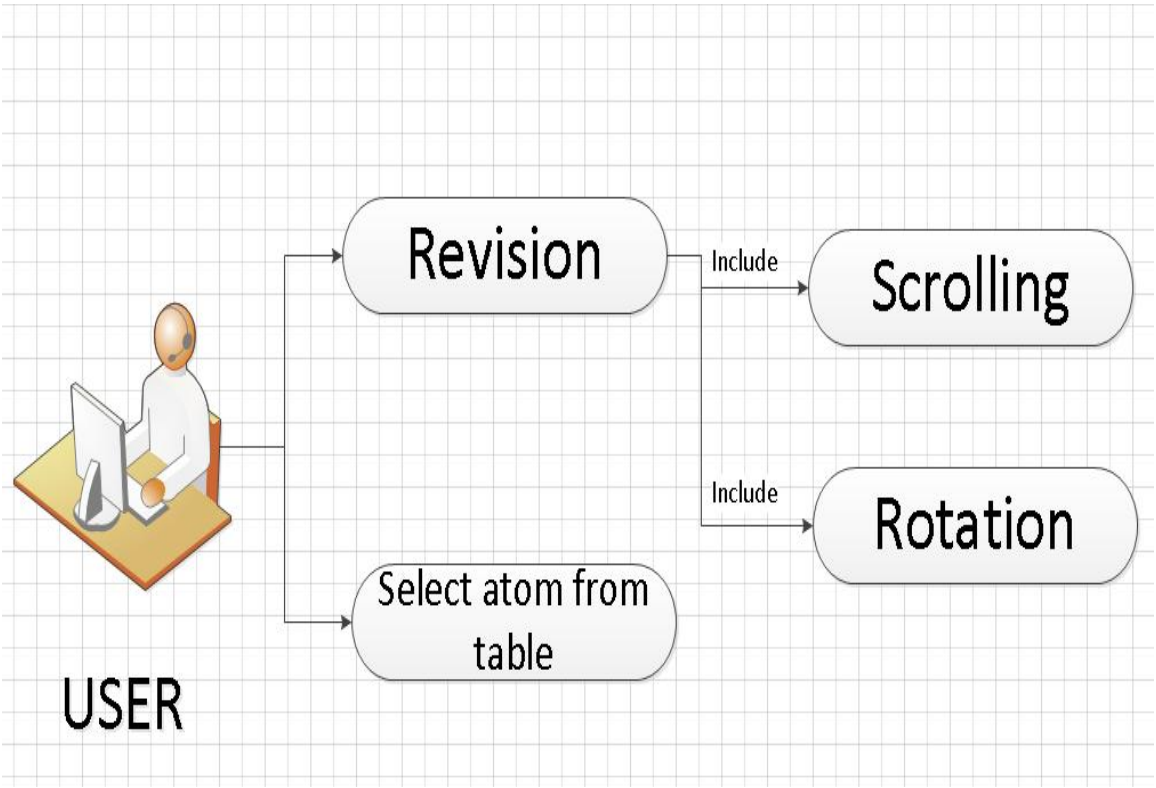

Рисунок 1.5 – Діаграма варіантів використання

Наступним кроком у визначенні специфікацій вимог до системи буде опис варіантів використання.

Варіанти використання представленні у таблицях 1.5-1.8

Варіант використання «Revision» подано у таблиці 1.5.

Таблиця 1.5

Контекст використання	Перегляд атома з метою отримання знань
Дійові особи	Користувач
Передумова	Користувач обрав атом
Триггер	-
Сценарій	Користувач переглядає атом
Післяумова	-

Варіант використання «Select atom from table» подано у таблиці 1.6.

Таблиця 1.6

Контекст використання	Вибір атома
Дійові особи	Користувач
Передумова	-
Триггер	Користувач обирає атом
Сценарій	-
Післяумова	-

Варіант використання «Scrolling» подано у таблиці 1.7.

Таблиця 1.7

Контекст використання	Прокрутка для збільшення масштабу
Дійові особи	Користувач
Передумова	Обрано атом
Триггер	Прокрутка колеса мишки
Сценарій	-
Післяумова	-

Варіант використання «Rotation» подано у таблиці 1.8.

Таблиця 1.8

Контекст використання	Зміна ракурсу перегляду
Дійові особи	Користувач
Передумова	Обрано атом
Триггер	Рух мишкою
Сценарій	-
Післяумова	-

Розкадровка варіантів використання

Прототип для функції «Select atom from table» зображено на рисунку 1.6

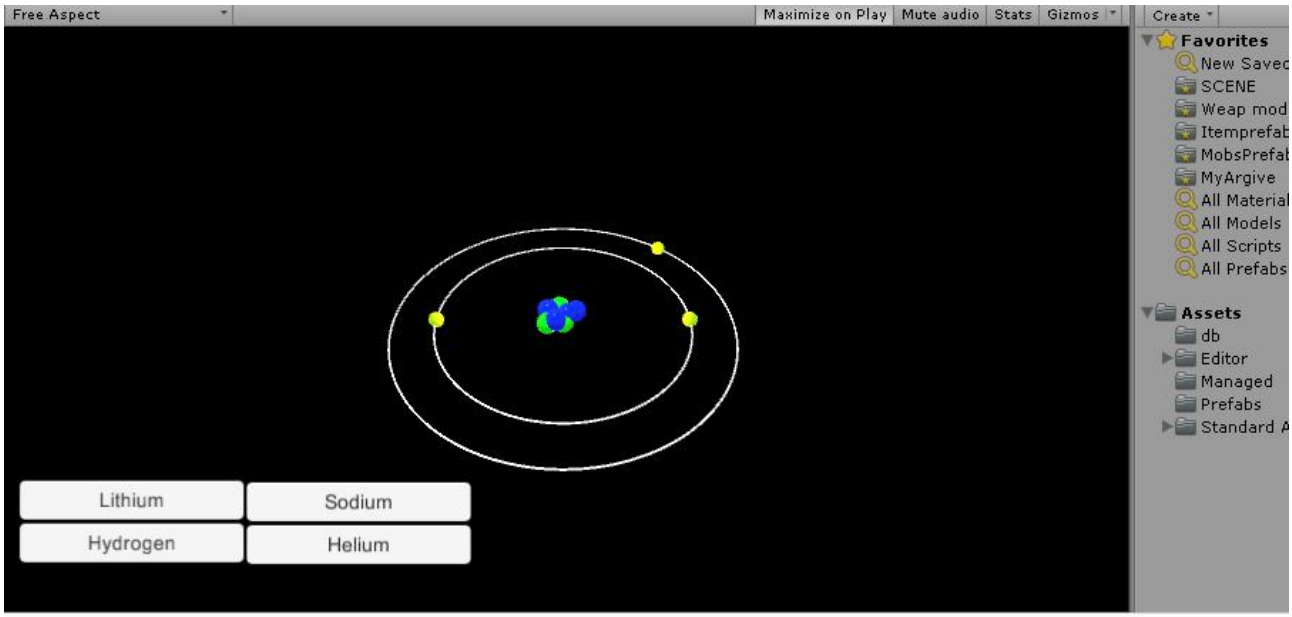

Рисунок 1.6 – прототип для функції «Select atom from table»

Специфікація функціональних вимог

Таблиця 1.9

Ідентифікатори вимоги	Назва вимоги	Атрибут вимог		
		Пріоритет	Складність	Контакт
1	Обрано атом	Обов’язкове	Низька	Користувач

Специфікація нефункціональних вимог

Таблиця 1.10

Ідентифікатор вимоги	Назва вимоги	Атрибути вимог		
		Пріоритет	Складність	Контакт
1. Застосовність				
1.1	Основні вимоги застосовності нової системи відносно інших систем, які знають користувачі	Рекомендована	Низька	Викладач

1.2	Вимоги по відповідальності стандартам графічного інтерфейсу користувача	Обов’язкова	Середня	Викладач
1.3	Час, необхідний для навчання звичайних і досвідчених користувачів	Рекомендована	Середня	Викладач
2. Надійність				
2.1	Доступність	Обов’язкова	Середня	Викладач
2.2	Середній час безвідмовної роботи	Обов’язкова	Середня	Викладач
2.3	Точність	Обов’язкова	Середня	Викладач
3. Робочі характеристики				
3.1	Використання ресурсів	Рекомендована	Середня	Викладач
4. Проектні обмеження				
4.1	Вимоги до технології програмування	Рекомендована	Середня	Викладач

Значення нефункціональних вимог:

- час, необхідний для навчання звичайних користувачів – 30 хвилин;
- час, необхідний для навчання досвідчених користувачів – 15 хвилин;
- основні вимоги застосовності нової системи відносно інших систем, які знають користувачі – всі функції системи є легкими у виконанні, а структура програми не відрізняється від існуючих аналогів;
- вимоги по відповідності загальним стандартам застосовності та стандартам графічного інтерфейсу користувача – програма повинна працювати в операційній системі Windows XP і вище;
- доступність – час, що витрачається на обслуговування системи не повинен перевищувати 3% від загального часу роботи;
- середній час безвідмовної роботи – 3години;
- використання ресурсів

– мінімальні системні вимоги: Об’єм HDD: > 100 Мб, Об’єм RAM: > 512 Мб , Відео пам'ять: > 128 Мб, Процесор: > 1 ГГц, Клавіатура, маніпулятор миша.

Розділ 2. Розробка проекту програмної системи

2.1. Розроблення архітектури програмної системи

Для розроблення системи було вибрано централізовану архітектуру. При використанні цієї архітектури все програмне забезпечення автоматизованої системи виконується централізовано на одному комп'ютері, який виконує одночасно багато завдань. На цьому комп'ютері повністю здійснюється процес вводу/виводу інформації, а також її прикладна обробка. В якості центрального комп'ютера для такої системи може застосовуватися або велика ЕОМ (названа також мейнфреймом) або так звана міні-ЕОМ. До подібного комплексу підключаються периферійні пристрої для введення / виведення інформації від кожного користувача.

Схему централізованої програмної системи зображено на рисунку 2.1.

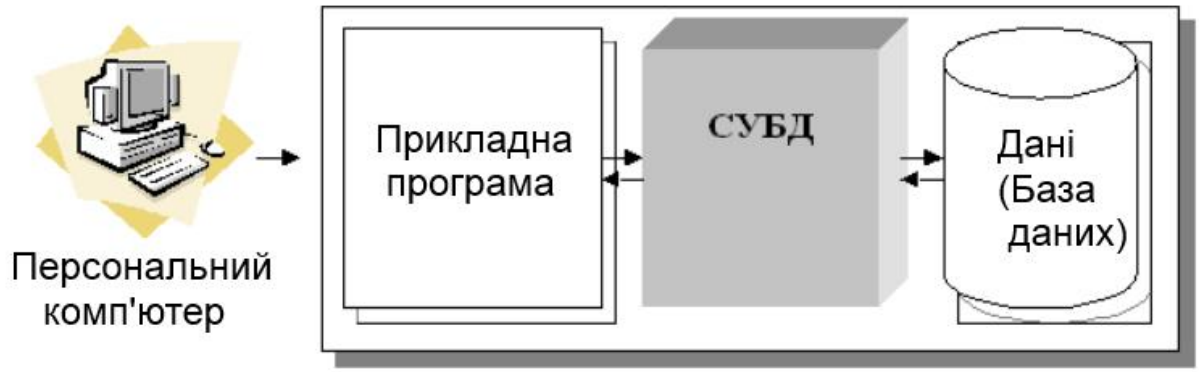

Рисунок 2.1. Схема централізованої програмної системи.

Для того, щоб краще зрозуміти процес функціонування системи додатка побудовано діаграми станів.

Діаграма станів процесу вибору атома зображено на рисунку 2.2

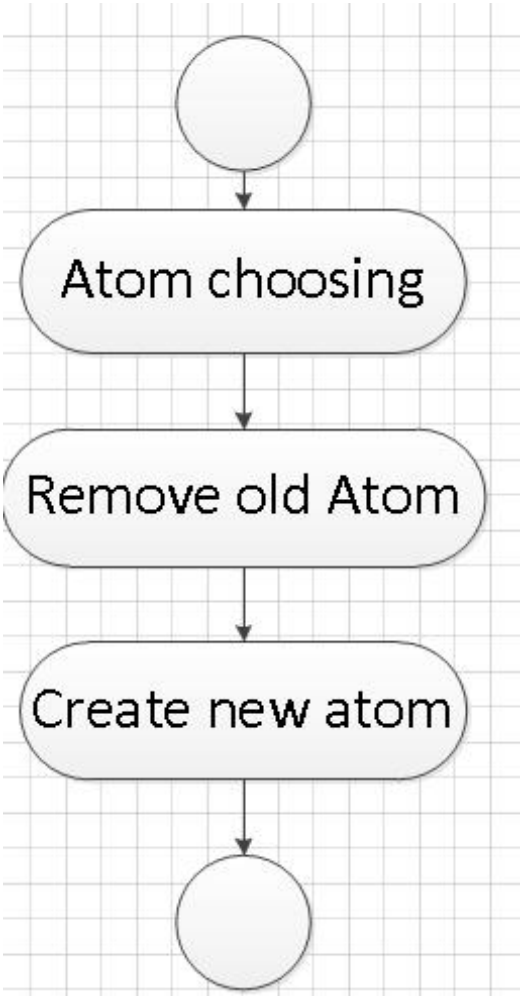

Рисунок 2.2. Діаграма станів процесу вибору атома.

Діаграму станів для зміни позиції камери зображено на рисунку 2.3

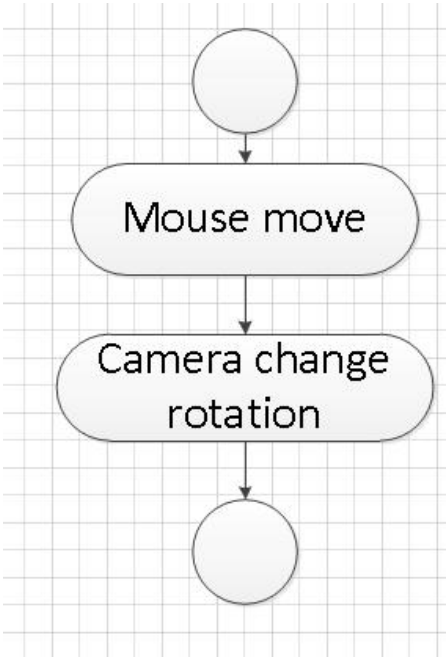

Рисунок 2.3. Діаграма станів зміни позиції камери.

Діаграму станів для зближення камери зображено на рисунку 2.4

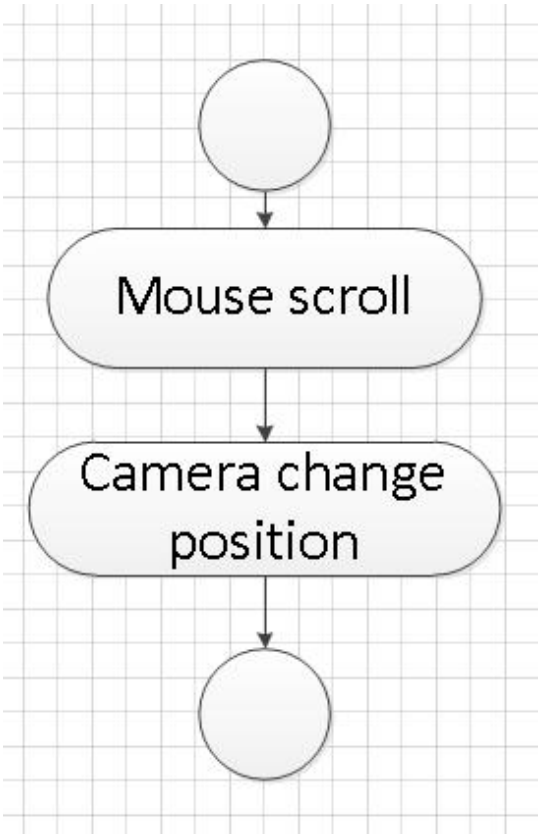

Рисунок 2.4. Діаграма станів зближення камери

2.2. Проектування структури бази даних

Етап проектування бази даних (БД) вважається одним із самих складних етапів створення БД, який не має явно вираженого початку й закінчення.

Порівняно з аналізом вимог до БД або розробкою додатків, проектування БД, на думку багатьох провідних фахівців, є невдало структурованим завданням. Якщо всі етапи створення БД перекриваються один з одним у своїй послідовності, то етап проектування перекривається з усіма іншими етапами.

Проектування починається з моменту прийняття стратегічних рішень і триває на етапах реалізації й тестування.

Процес проектування БД охоплює кілька основних сфер:

- проектування об'єктів БД (таблиці, подання, індекси, тригери, збережені процедури, функції, пакети) для подання даних ПО в БД;

- проектування інтерфейсу взаємодії з БД (форми, звіти й т.д.), тобто проектування додатків, які будуть супроводжувати дані в БД і реалізовувати питально-відповідні відношення на цих даних;
- проектування БД під конкретне обчислювальне середовище або інформаційну технологію (архітектура "клієнт-сервер", паралельні архітектури, розподілене обчислювальне середовище);
- проектування БД під призначення системи (інтелектуальний аналіз даних, OLAP, OLTP і т.д.).

Проектування бази даних слід розпочинати зі створення DFD-моделі предметної області (рис. 2.5).

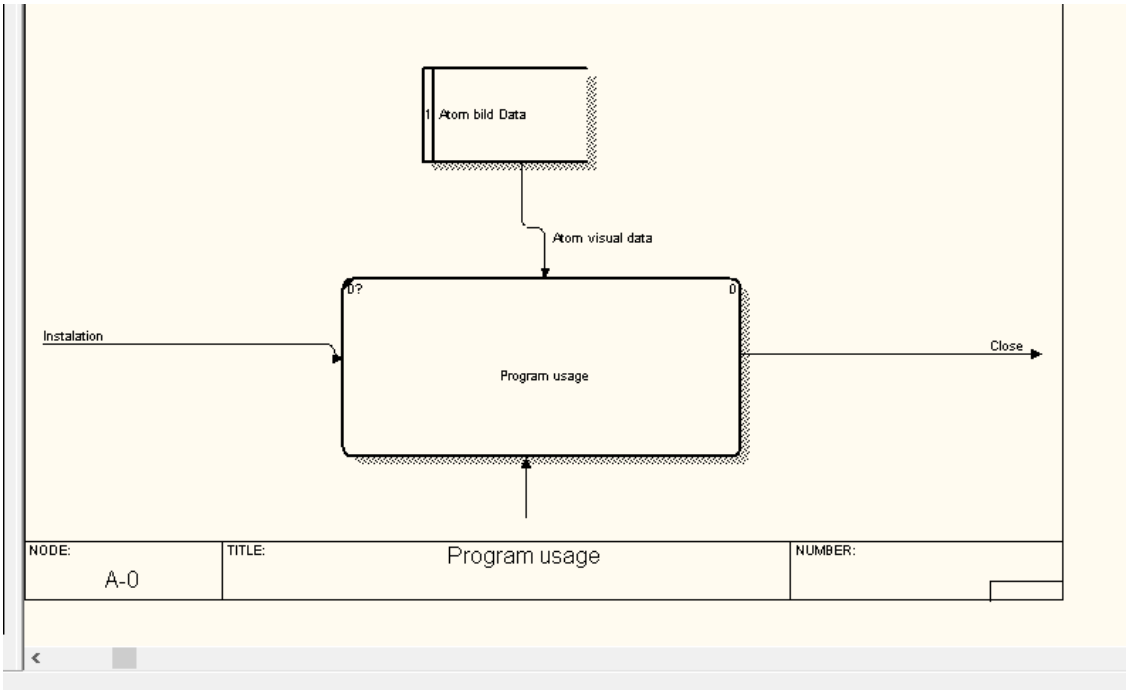

Рисунок 2.5 – Діаграма потоків даних

Діаграма декомпозиції зображена на рисунку 2.5.

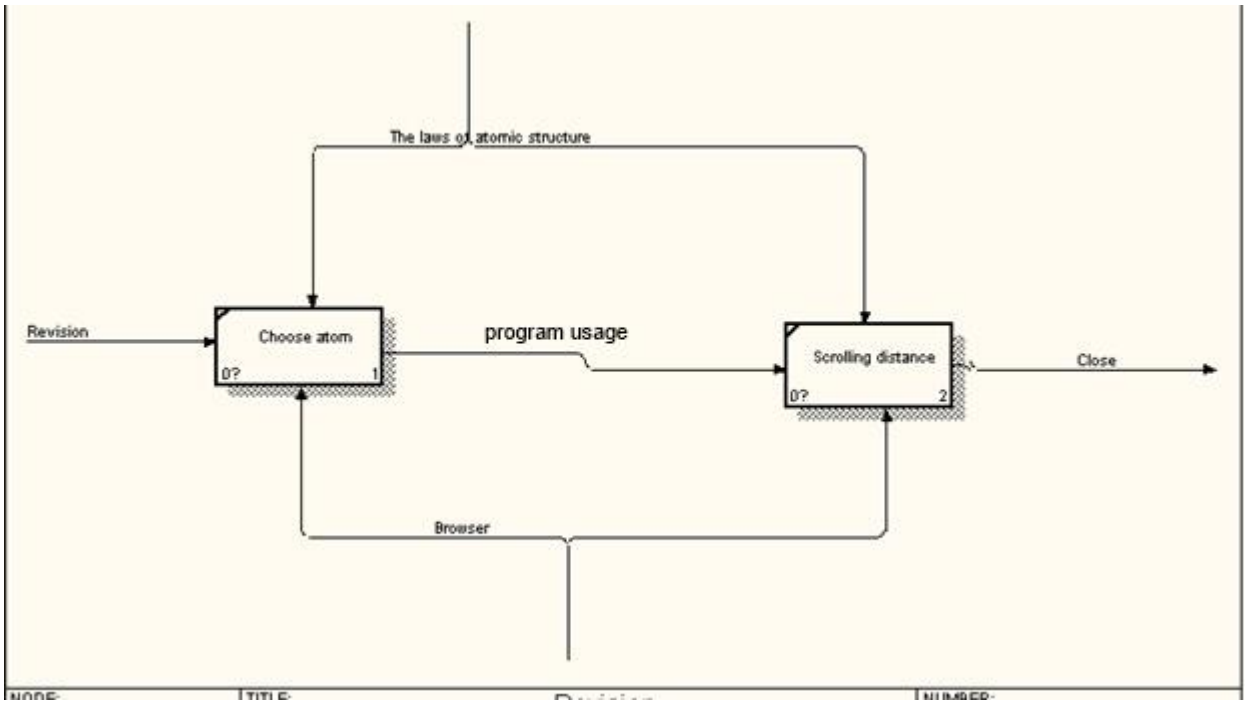

Рисунок 2.6 – Діаграма декомпозиції

Як показано на Рис. 2.6 робота програмної системи здійснюється наступним чином :

1. Входи (input), потоки, які поступають у систему і переробляються нею у вихідні величини, на діаграмі зображені ліворуч:

- ✓ The laws of atomic structure(закони будови атома);
- ✓ Revision(перегляд);

2. Виходи (output), продукти діяльності системи, тобто результати перетворення вхідних величин тощо, на діаграмі зображені праворуч:

- ✓ Close(завершення роботи

Наступним кроком є виявлення основних сутностей та зв’язків між ними. Для цього створено діаграму елементів та зв’язків (рис. 2.7).

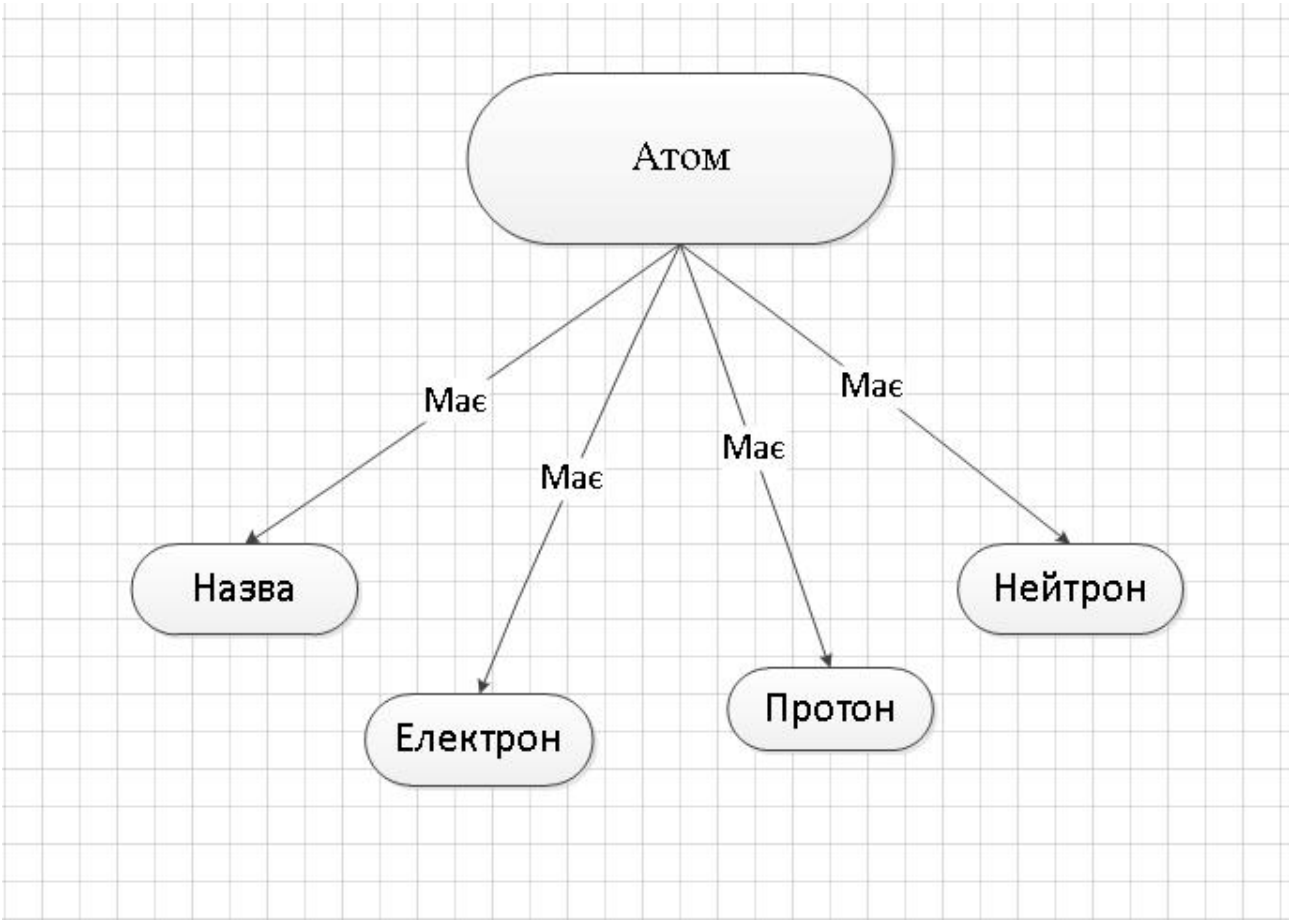

Рисунок 2.7 – Діаграма елементів та зв’язків

Після цього було створено таблицю ідентифікаторів, яку подано у таблиці 2.1.

Таблиця 2.1

Об’єкт	Властивість	Тип	Розмірність	Ідентифікатор
Атом	Ім’я Протон Нейтрон Електрон	string integer integer integer	15 10 10	Name proton neutron electron

Наступним кроком є створення діаграми класів UML (рис. 2.8).

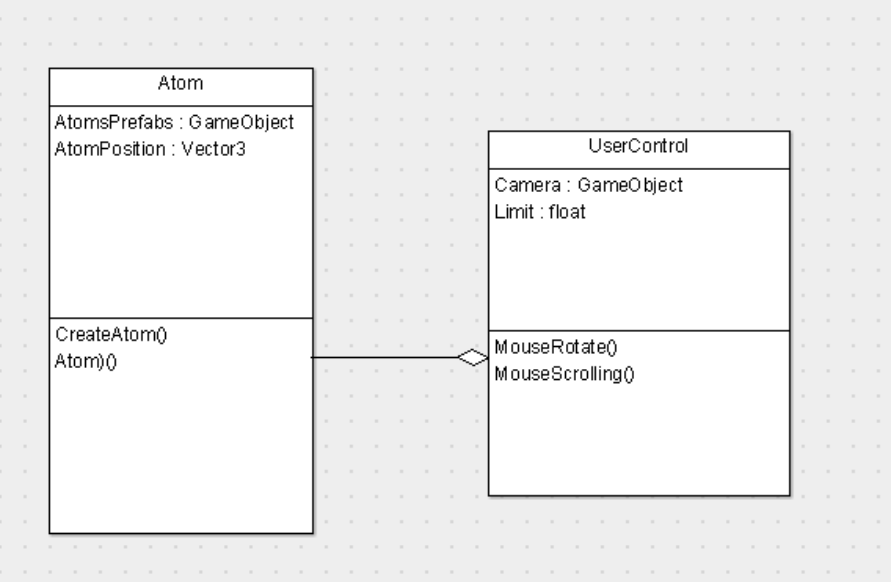

Рисунок 2.8 – Діаграма класів UML

Найбільше поширеним засобом моделювання даних є діаґрами “сутність-взаємозв'язок” (ERD). З їхньою допомогою визначаються важливі для предметної області об'єкти (сутності), їх властивості (атрибути) і відношення один з одним (зв'язки). ERD безпосередньо використовуються для проектування реляційних баз даних.

Тому на підставі аналізу предметної області у роботі спроектовано структуру бази даних у вигляді ERD діаграми (рис 2.9).

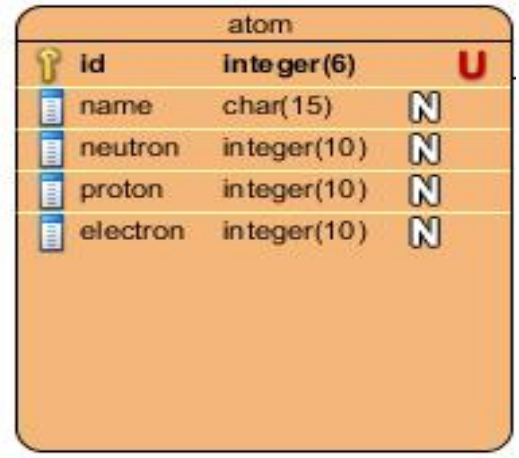

Рисунок 2.9 – ERD-діаграма

Проектування структури бази даних дозволяє визначити та повністю описати всі відношення та поля і є завершальною ланкою перед етапом програмної реалізації бази даних.

Розділ 3. Програмна реалізація

3.1. Обґрунтування вибору мови програмування і середовища.

Для розробки програмного забезпечення візуалізації будови атома вирішено обрати движок Unity3D. На рис 1.1 зображую логотип движка

Рисунок 3.1 – Логотип Unity

Unity3D – це інструмента для розробки двовимірних та тривимірних застосунків та ігор що, працює на системах Windows та OS X. Створені за допомогою цього інструменту додатки працюють на системах Windows, OS X Android, Linux а також на гральних консолях Xbox, PlayStation Wii. Також Unity надає можливість створювати інтернет-застосунки.

Однією з переваг Unity є те довідка доступна як в онлайн, так і в оффлайн варіанті. У довідці по скриптів працює пошук, в ній міститься безліч наочних прикладів і вся документація.

Движок unity підтримує такі мови програмування як C#, JavaScrit. За мову програмування обрано C#.

Для створення інтерфейсу використовується система UI. UI- система користувальницького інтерфейсу, що дозволяє створювати призначені для користувача інтерфейси швидко і інтуїтивно.

Для запуску програмного забезпечення, апаратна частина повинна підтримувати наступні вимоги

- Графічна карта з підтримкою directx X9.
- Центральний процесор з підтримкою набору інструкцій sse2

3.2. Розробка додатку

За допомогою UI створюю форми та кнопки, вивожу в інтерфейс необхідні тексти кнопки та зображення. Результат демонструю на рисунку 3.2

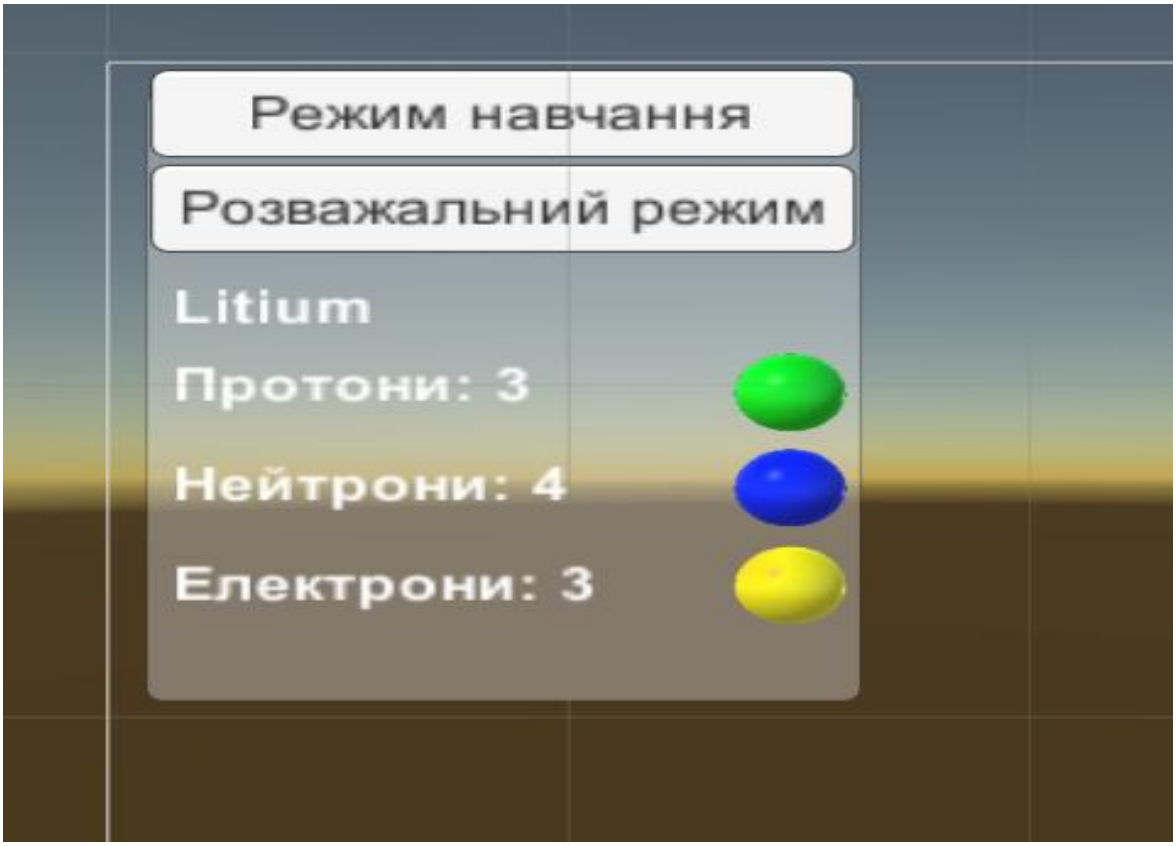

Рисунок 3.2 UI інтерфейс

Далі налаштовую дії кнопок які запускатимуть виконання різних операцій в скріптах. Приклад налаштування кнопки зображено на рисунку 3.3.

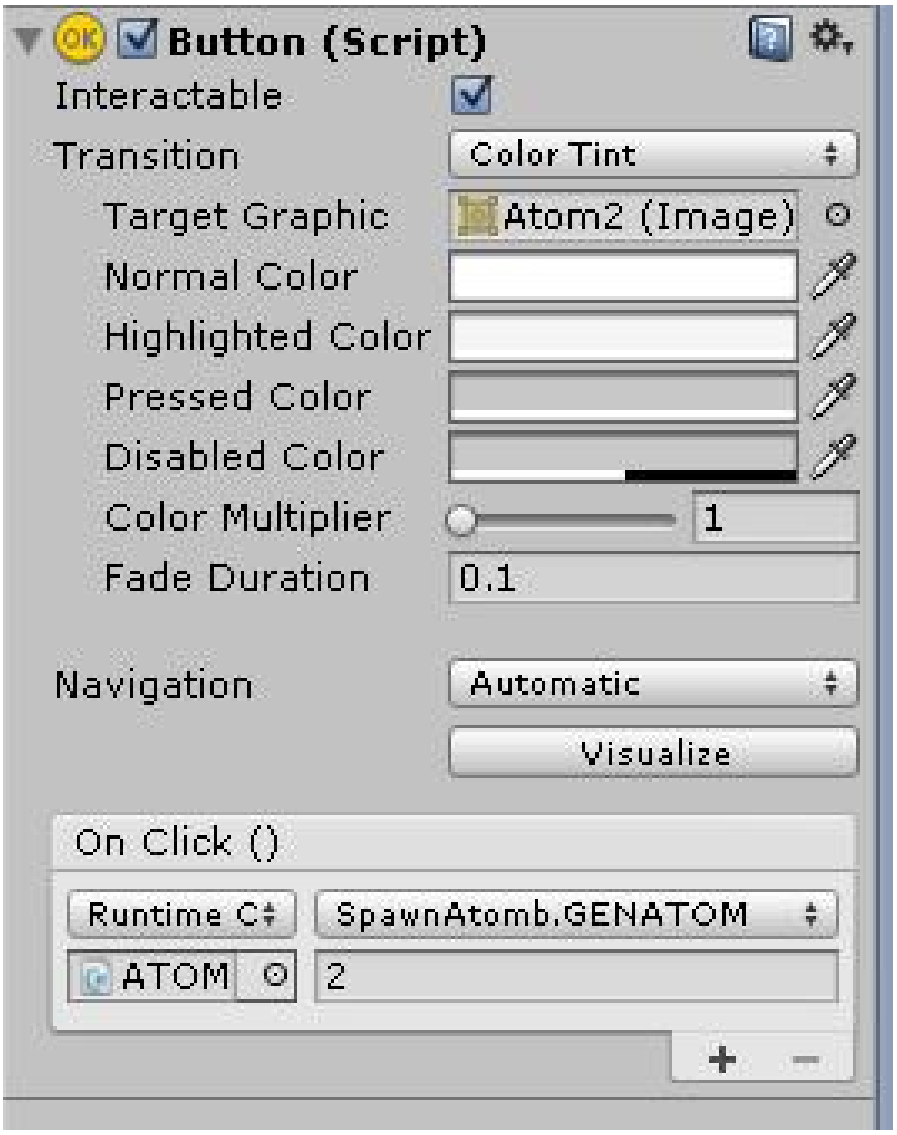

Рисунок 3.3. Налаштування дій кнопок

Для того щоб інтерфейс виглядав однаково на екранах з різним розширенням налаштовую авто розміщення від лівої сторони екрану. Процес налаштування авто розміщення демонструю на рисунку 3.4.

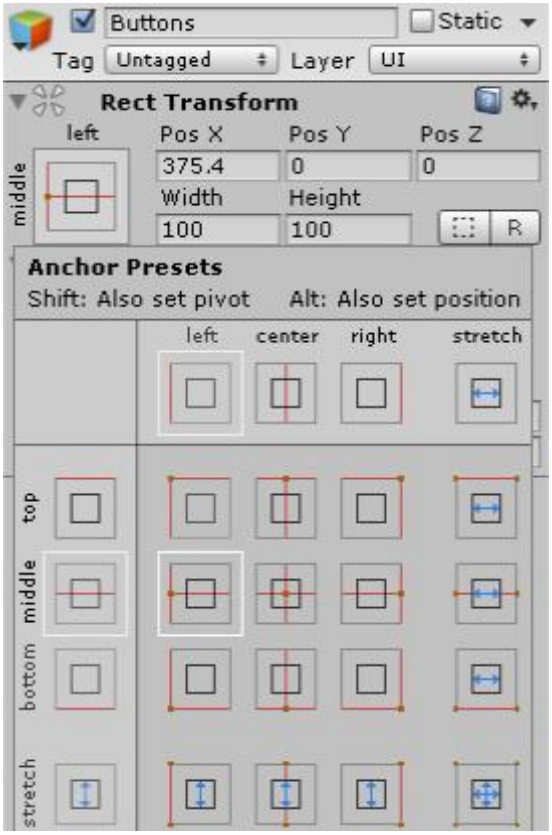

Рисунок 3.4 вікно авто розміщення

Демонструю ієрархію об’єктів інтерфейсу на рисунку 3.5

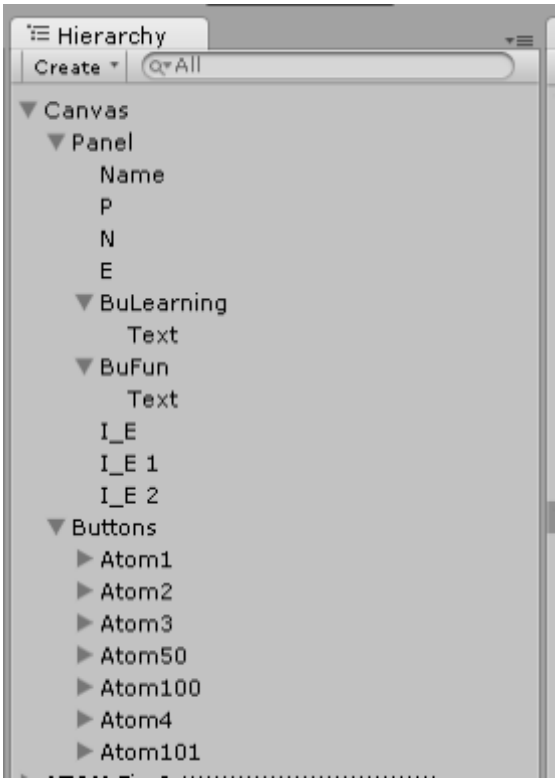

Рисунок 3.5 Ієрархія інтерфейсів

Для візуалізації будови атома необхідно зобразити його елементи а саме електрон, протон, нейтрон. Здійснюється це за допомогою примітивного об’єкта движка Unity а саме сфери. Щоб відрізнити сфери створюю матеріали Electron, Proton, Neutron та накладаю їх на сфери. Розташовую їх на сцені та вибудовую базову структуру атома на сцені використовуючи ігрові об’єкти(Game object).

Game object – в Unity це об’єкт в який має набір компонентів і може знаходиться на сцені в нашому випадку це електрони, нейтрони та протони.
Ієрархію об’єктів демонструю на рисунку 3.6 а розташування атомів на сцені демонструю на рисунку 3.7.

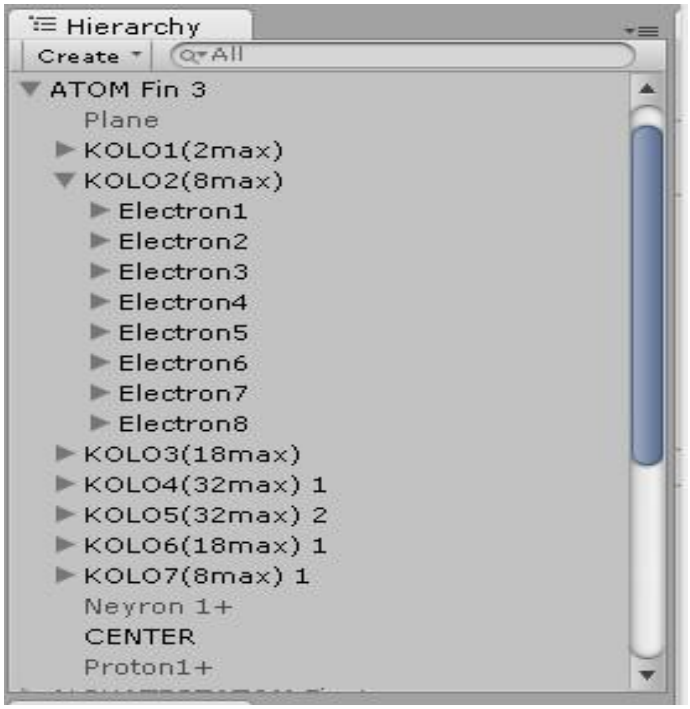

Рис 3.6. Ієрархія об’єктів

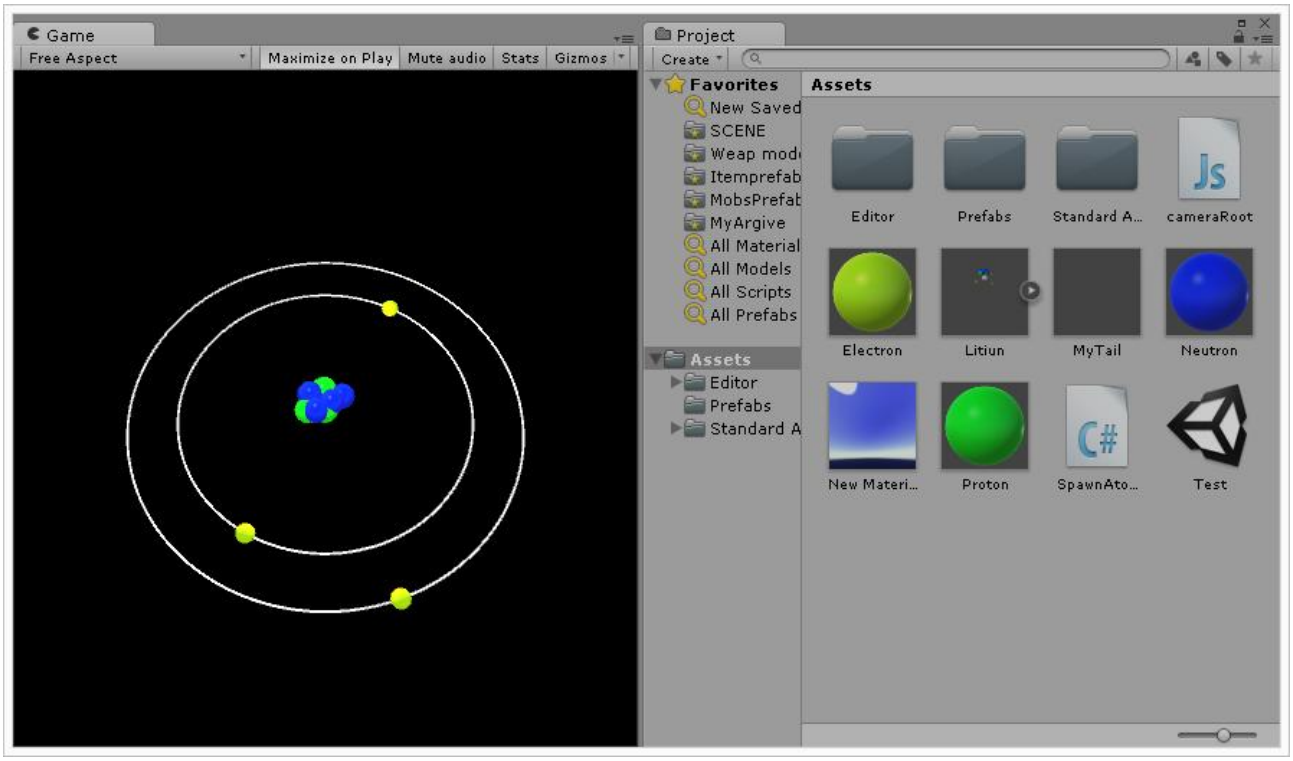

Рис 3.7 Представлення сцени з атомом.

Оскільки необхідно відображати будову різних атомів, кращим методом ніж створювати кожен вручну буде створити метод генерування вмісту атома

Для початку розробляю заготовку або Префаб. Префаб це збережений в проекті об’єкт або набір об’єктів. І створюю протон і нейтрон розмальовую їх в синій та зелений кольори.

Наступним кроком буде додавання компоненту Spring joint що буде стягувати протони й нейтрони в цент атома. Параметри компонента зображую на рис 3.8.

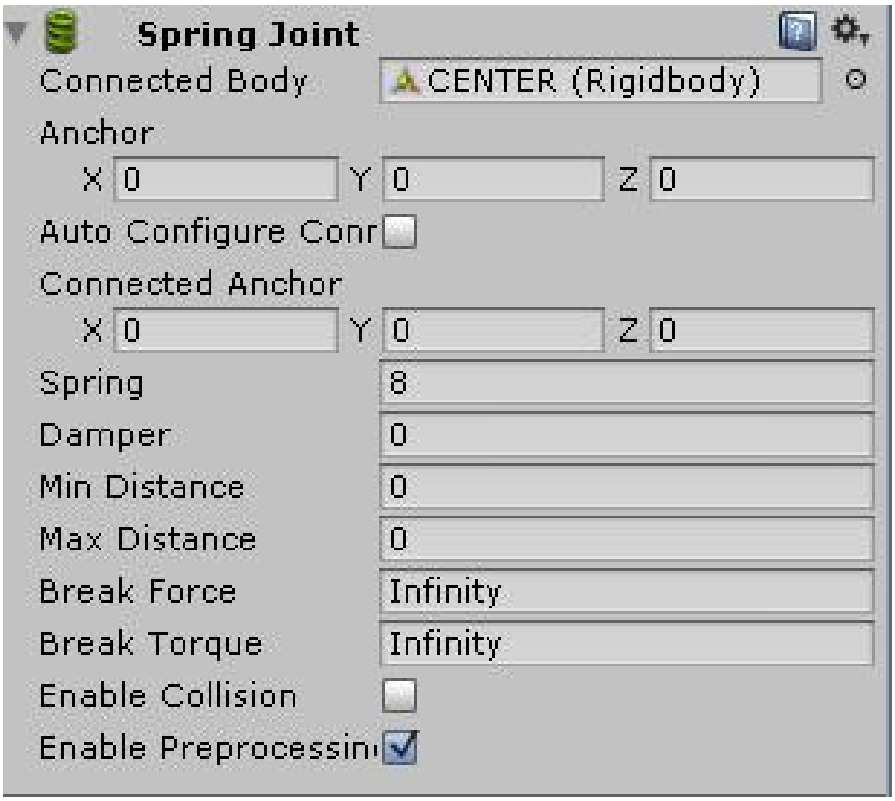

Рис 3.8 Spring joint.

Наступним кроком буде написання класу що реалізує генерацію будови атома з вказаними параметрами. Клас буде написано на мові програмування C# і називатиметься SpawnAtomb.

Для генерації описую змінні Proton і Neutron з типом змінної GameObject в яку присвоюю раніше заготовані префаби.

Приклад присвоєних префабів до класу демонструю на рис 3.9

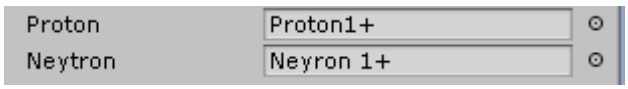

Рис 3.9 Присвоєні префаби.

Тепер коли є все що необхідно реалізується генерація будови атома кодом. Частина коду що реалізує генерацію:

```
for (int i=0; i<300; i++) {
                    if(G_P[i]!=null)Destroy(G_P[i]);
                }
```

```
                for (int i=0; i<P; i++) {
                        Vector3 temp=Center.transform.position;
                        temp.y+= Random.Range(-1f,1f);
                        temp.x+= Random.Range(-1f,1f);
                        temp.z+= Random.Range(-1f,1f);
                        ThatObj=Instantiate(Proton,temp
,Center.transform.rotation) as GameObject;
                        G_P[i]=ThatObj;
                        ThatObj.SetActive(true);}
                for (int i=0; i<300; i++) {
                        if(G_N[i]!=null)Destroy(G_N[i]);
                }
                for (int i=0; i<N; i++) {
                        Vector3 temp=Center.transform.position;
                        temp.y+= Random.Range(-1f,1f);
                        temp.x+= Random.Range(-1f,1f);
                        temp.z+= Random.Range(-1f,1f);
                        ThatObj=Instantiate(Neytron,temp
,Center.transform.rotation) as GameObject;
                        G_N[i]=ThatObj;
                        ThatObj.SetActive(true);
        }
```

Весь код виведено в додатку А

Результат виконання коду демострую на рис 3.10

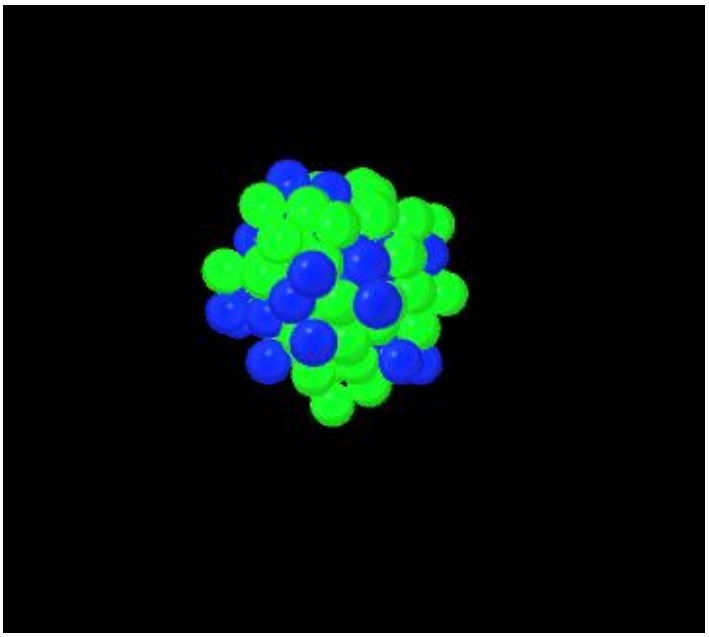

Рис 3.10 побудова атома кодом.

Наступним кроком буде створення електронів. Створюю новий об’єкт який матиме компонент Constat forse завдяки якому реалізую поворот електронів відносно центру атома. Параметри компоненту Constat forse зображую на рис Рис 3.11

Рис 3.11 Constat forse

Для відображення кіл електронів використовую компонент Trail Renderer. Додаю компонент на електрон та налаштовую такі параметри як Start Width та End Winth які відповілають за вигляд лінії на початку і при закінченні а також встановлюю Time час зникнення лінії.Парметри компонента демострую на рис Рис 3.12, результат демонструю на рис Рис 3.13.

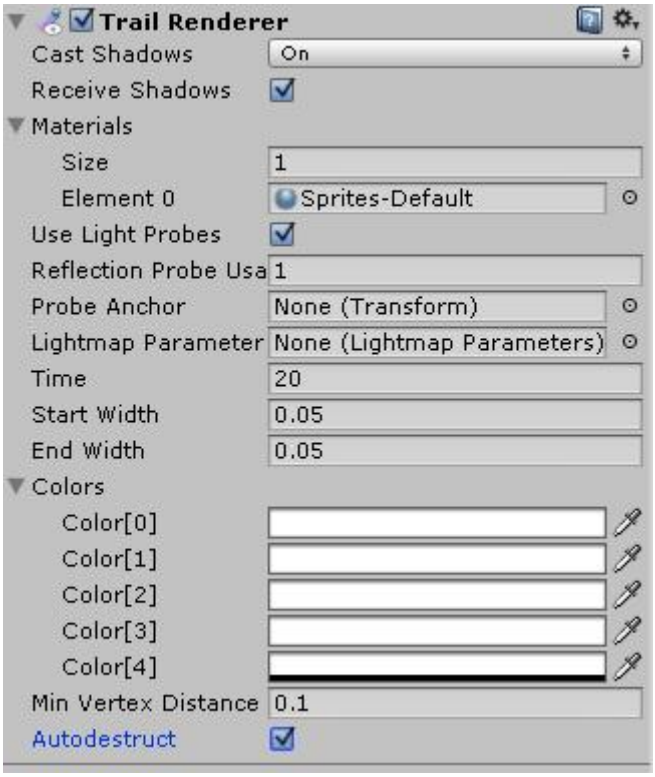

Рис 3.12. Trail Renderer параметри.

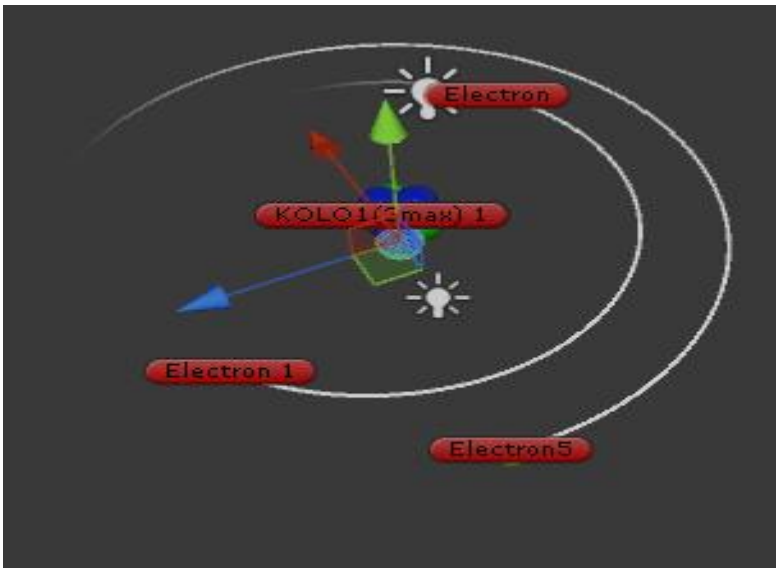

Рис 3.13. Trail Renderer

Схематичний метод відображення атома не виглядає цікавим як хаотичний тому розробляю метод що дозволяє переглядати атом в 2х режимах: режим навчання та розважальний режим. В режимі навчання кола електронів складені в ряд а в розважальному режимі вони розкладені хаотично що виглядає красивіше.

Зміна режимів досягається зміною параметрів розвороту об’єкту в сцені. При розважальному режимі ці параметри випадкові при навчальному сталі. Користувач може змінити режим кнопками режим навчання та розважальний режим.

Частина коду програми що реалізує зміну параметрів повороту:

```
using UnityEngine;
using System.Collections;
public class Mode : MonoBehaviour {
	public GameObject[] Ring;
	public void Learning(){
		for (int i=0; i<7; i++) {
			Ring[i].transform.rotation=new Quaternion(0,0,90,0);
		}
	}
public void Fun(){
		for (int i=0; i<7; i++) {
			Ring[i].transform.rotation=new
Quaternion(Random.Range(0,360),Random.Range(0,360),Random.Range(0,360),Ra
ndom.Range(0,360));
		}	}
}
```

Варіанти використання режимів представляю на рис 3.14 і 3.15

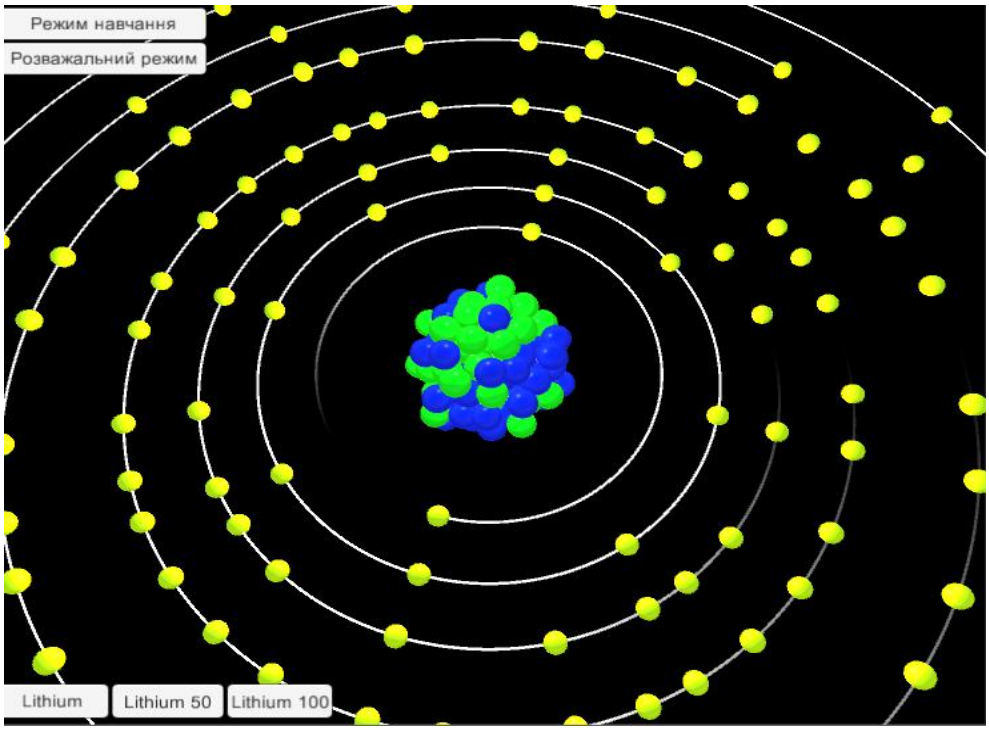

Рис 3.14. Режим навчання

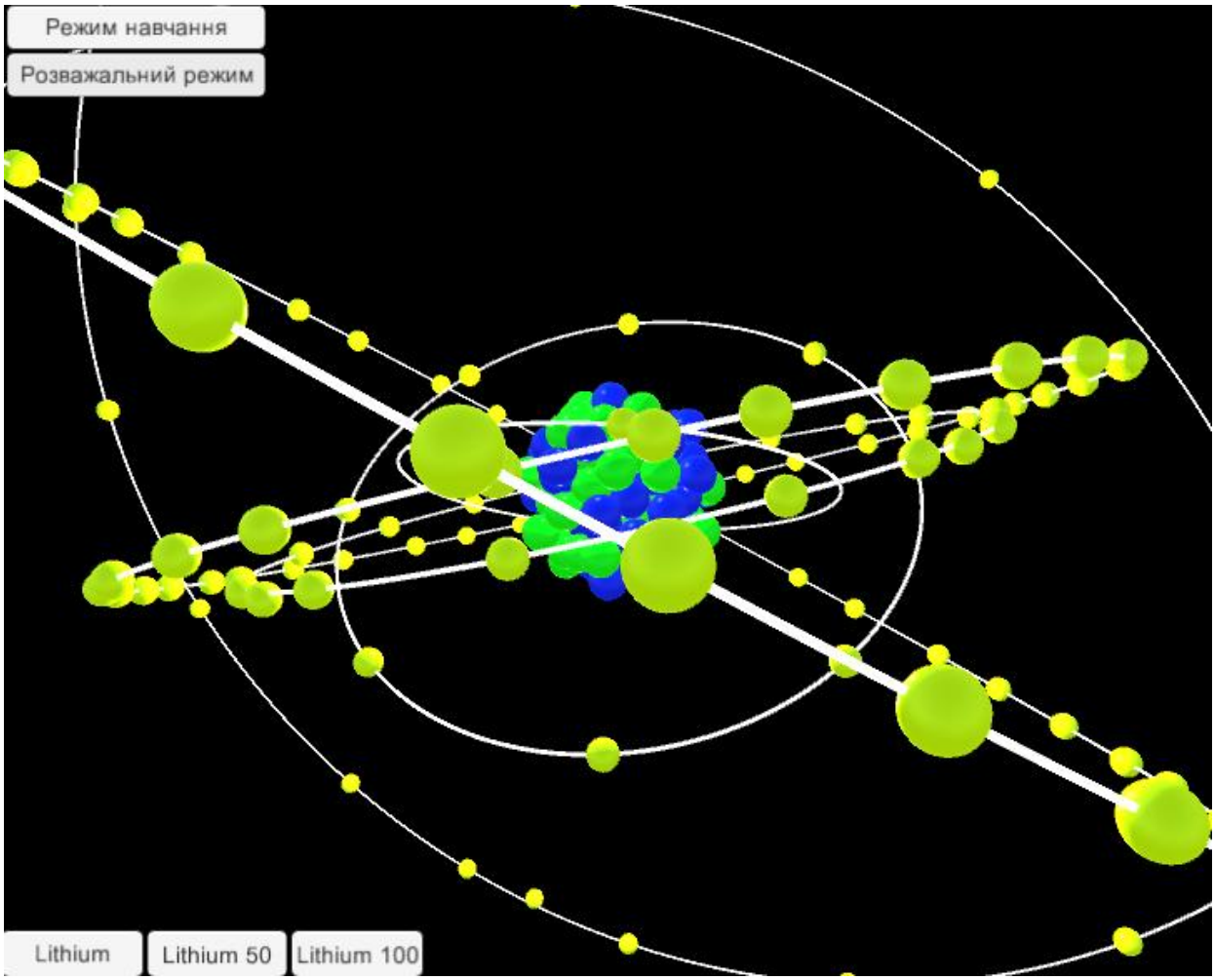

Рис 3.15. Розважальний режим

Важливим елементом зацікавлення користувача є взаємодія з програмою і не тільки через користувацький інтерфейс ай через контролери. Тому наступним кроком буде розробка контролеру що дозволить користувачу крутити камеру навколо атома.

Оскільки за основу використовую стандартний контролер який написаний на мові Java script то і мій контролер буде написаний на ній.

Частина коду що реалізує повороти камери:

```
var rotation = Quaternion.Euler(y, x, 0);
    var targetPos = target.position + targetOffset;
    var direction = rotation * -Vector3.forward;

    var targetDistance = AdjustLineOfSight(targetPos, direction);
        currentDistance = Mathf.SmoothDamp(currentDistance, targetDistance, distanceVelocity, closerSnapLag * .3);

    transform.rotation = rotation;
    transform.position = targetPos + direction * currentDistance;
```

Створюю поле в якому відображатиметься інформація про атом рис Рис 3.16.

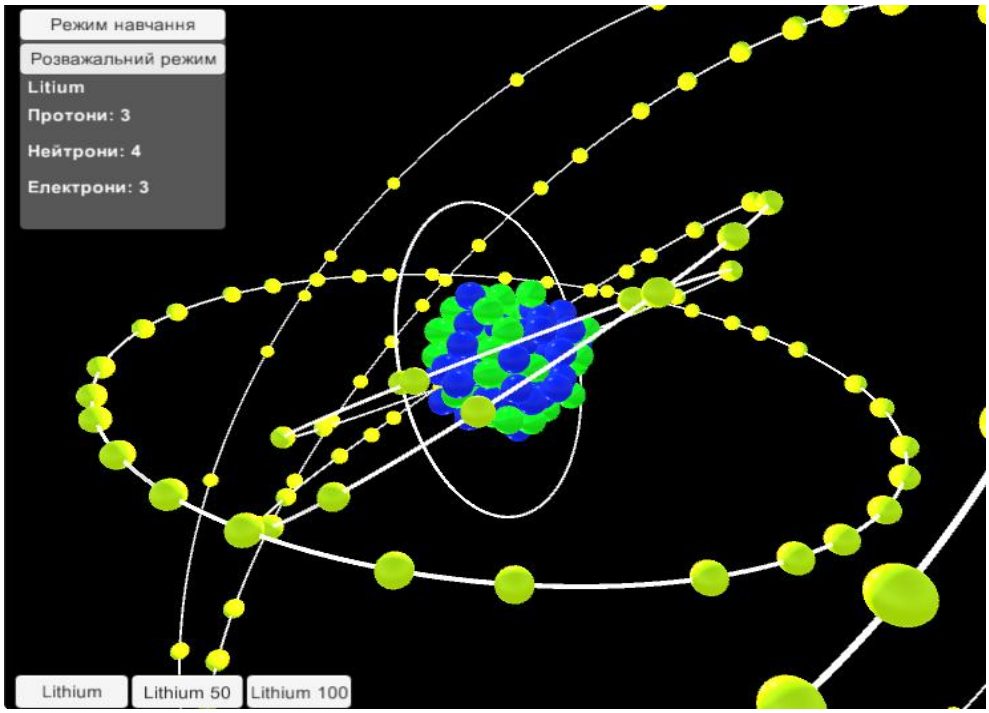

Рис 3.16. поле нформація про атом

3.3. Програмна реалізація бази даних

В якості бази даних обираю SqlLite оскільки для реалізації даного продукту обрано централізовану систему бази даних і SQLite не використовує систему клієнт сервер, не є окремим процесом а просто надає бібліотеку.

SQLite- спрощена реляційна система керування базами даних. Реалізована у вигляді бібліотеки, де реалізовано багато зі стандарту Sql 92. Ще одною перевагою SQLite є те що він розповсюджується під безплатною ліцензією.

Особливості

- Не потребує встановлення та адміністрування.
- Підтримує більшу частину стандарту SQL 92
- Бд зберігається в файлі
- Простий у використанні
- Підтримує багато операційних систем

Для створення і редагування БД наш вибір припав на SQLite Manager. SQLite Manager - це плагін для Firefox, він безкоштовний, зручний і багатоплатформовий.

Розробка бази даних починається з створення файлу бази даних MyAtom.sqlite.

Далі написано SQL код який створить таблицю з такими даними як: номер атома, ім’я атома, кількість електронів, кількість протонів, кількість нейтронів.

SQL код:

CREATE TABLE "Tatom" ("id" INTEGER PRIMARY KEY NOT NULL , "name" TEXT, "n" INTEGER, "e" INTEGER, "p" INTEGER)

Структуру таблиці зображую на рис 3.17.

Column ID	Name	Type	Not Null	Default Value	Primary Key
0	id	INTEGER	1	null	1
1	name	TEXT	0	null	0
2	n	INTEGER	0	null	0
3	e	INTEGER	0	null	0
4	p	INTEGER	0	null	0

Рис 3.17. Структура таблиці

Таблицю з внесеними даними зображено на рис 3.18

id	name	n	e	p
1	Hydrogen	0	1	1
2	Helium	2	2	2
3	Lithium	4	3	3
22	Titanium	26	22	22
50	Tin	69	50	50
81	Thalium	123	81	81
100	Fermium	157	100	100

Рис 3.18. Таблицю з внесеними даними

Наступним кроком буде підключення бази даних до нашого проекту. Для цього потрібно перемістити наступні файли в папку Unity/Editor

- System.Data.dll
- Mono.Data.SQlite.dll
- System.Security.dll
- System.Configuration.dll
- System.EnterpriseServices.dll

Наступним кроком буде написання коду що забезпечить зчитування з бази даних:

```
public void loadDATA(int DATA){
```

```
string connectionString = "URI=file:" + Application.dataPath + "/db/MyAtom.sqlite";
        using (IDbConnection dbcon = (IDbConnection)new SqliteConnection(connectionString)){ dbcon.Open();
            var sql = "SELECT id,name, p, e, n FROM Tatom where id like "+DATA;//where id like ZMINNA  var sql = "SELECT id,name, p, e, n FROM Tatom";
            using (IDbCommand dbcmd = dbcon.CreateCommand()){
                dbcmd.CommandText = sql;
using (IDataReader reader = dbcmd.ExecuteReader()){
while (reader.Read()){
const string frmt = "id: {0};name: {1};  n: {2}; p: {3};";
GetComponent<SpawnAtomb>().P=reader. GetInt32(4);
GetComponent<SpawnAtomb>().E=reader. GetInt32(3);
GetComponent<SpawnAtomb>().N=reader.GetInt32(2);
            GetComponent<SpawnAtomb>().Aname=reader.GetString(1);
                }}}dbcon.Close();}}
```

Завершальним кроком в розробці програмного забезпечення буде компілювання його разом з базою даних під платформу Windows (Рис 3.19)

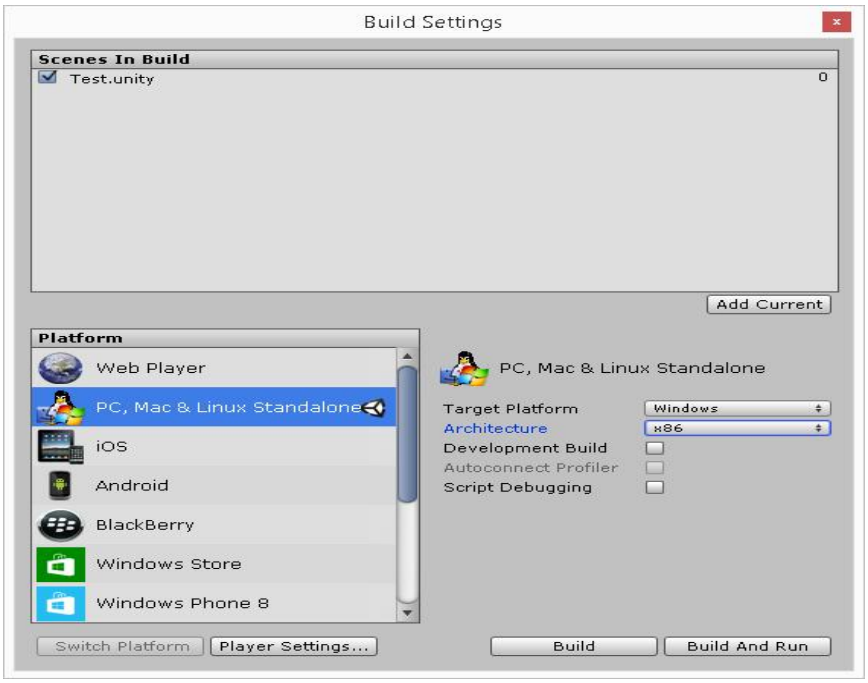

Рис 3.19. Компілювання додатку

Розділ 4. Тестування та дослідна експлуатація

4.1 Тестування

Для проведення модульного тестування створю скріпт ModulTest.cs який перевірить модуль вивантаження даних з бази даних та запускаю його, результат помилок не виявлено.

Код скрипта ModulTest.cs:

```
using UnityEngine;
using System.Collections;
public class ModulTest : MonoBehaviour {
   [TestFixture]
   public class GenTest
   {
      [Test]
      public void ApplyAtom()
      {  int x=1;

     GetComponent<Test>().loadDATA(x)
GetComponent<SpawnAtomb>().N;
Assert.AreEqual(GetComponent<SpawnAtomb>().N;,x);
}

}
```

Одним із популярних видів тестування є системне тестування**.** Основним завданням системного тестування є перевірка як функціональних, так і не функціональних вимог в системі в цілому. При цьому виявляються дефекти, такі як неправильне використання ресурсів системи, непередбачені комбінації даних користувача рівня, несумісність з оточенням, непередбачені сценарії використання, відсутня або неправильна функціональність, незручність використання і т.д.

Для мінімізації ризиків, пов'язаних з особливостями поведінки в системи в тому чи іншому середовищі, під час тестування рекомендується

використовувати оточення максимально наближене до того, на яке буде встановлено продукт після видачі.

На основі уявлення про способи використання продукту створено випадки використання системи . За конкретного випадку використання можна визначити один або більше сценаріїв. На перевірку кожного сценарію пишу тест кейси і відображаю їх в таблиці 4.1**.**

Таблиця 4.1

Дія	Очікуваний результат	Тестовий результат
Натискання кнопки “Режим навчання”	Програма перейде в режим навчання	Пройдено
Натискання кнопки “Розважальний режим”	Програма перейде в розважальний режим	Пройдено
Натискання кнопки “1”	Змінется атом	Пройдено
Натискання кнопки “2”	Змінется атом	Пройдено
Натискання кнопки “3”	Змінется атом	Пройдено
Натискання кнопки “22”	Змінется атом	Пройдено
Натискання кнопки “50”	Змінется атом	Пройдено
Натискання кнопки “100”	змінется атом	Пройдено
Натискання кнопки“Вихід з програми”	Програма закриється	Пройдено

4.2. Розгортання програмного продукту

Для запуску програмного забезпечення, апаратна частина повинна підтримувати наступні вимоги:

- Графічна карта з підтримкою directx X9.
- Центральний процесор з підтримкою набору інструкцій sse2

Також для запуску додатка потрібна операційна система Windows Xp або новіші версії Windows. Оскільки движок Unity підтримує багато платформ, він може бути скомпільованим на такі операційні системи як Linux та Mac OS.

4.3. Інструкція користувача

Програмне забезпечення візуалізації будови атома розроблено засобами Unity 3D версії 5.0 на мові програмування C# і може працювати на операційних системах сімейства Windows.

Перелік файлів необхідних для роботи додатку наведено в таблиці 4.2.

Таблиця 4.2

№	Файл	Призначення
1	Diplom Atom.exe	Виконавчий файл
2	sqlite3.dll	Бібліотека бази даних SQLite
3	UnityEngine.dll	Бібліотека Unity
4	MyAtom.sqlite	База даних
5	output_log.txt	Файл виведення помилок

Додаток підтримує такі налаштування:

- розмір вікна додатку можна підібрати під монітор
- запуск додатку на весь екран або в вікні.
- зміна якості графіки
- зміна клавіш вводу
- вибір дисплею

Вікно налаштувань представлено на рисунку 4.1.

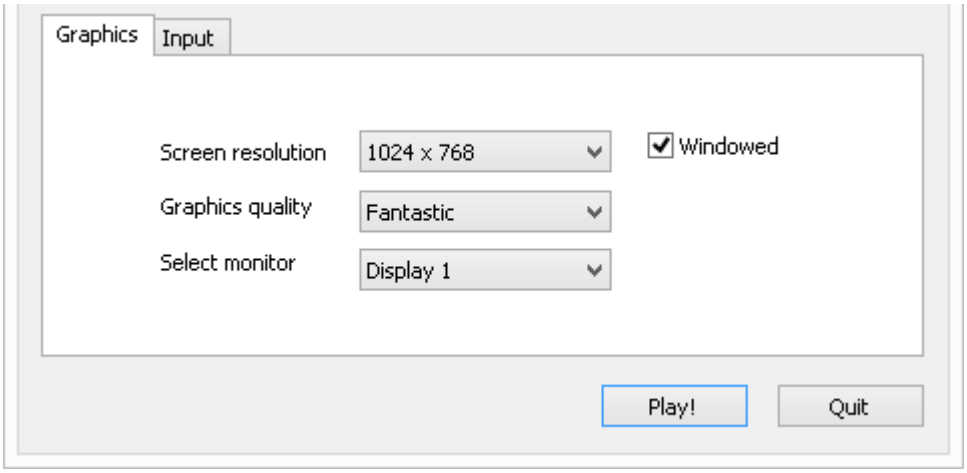

Рис 4.1. Вікно налаштувань

Для аналізу помилок, що появляються в процесі використання будуть записуватися в файл output_log.txt.

ВИСНОВКИ

В процесі виконання бакалаврської роботи на тему “Програмне забезпечення візуалізації будови атома засобами Unity 3D” було:

1) досліджено предметну область що стосується будови атома та освітнього програмного забезпечення.
2) спроектовано для розробки додатку проект програмної системи, його бізнес логіку та архітектуру.
3) розроблено додаток що забезпечує візуалізацію будови атома в тривимірному просторі, взаємодію та інтерфейс засобами Unity 3D
4) проведено тестування додатку і написана інструкція до нього.

Такий додаток може зіграти значну роль в школах оскільки сприяє формуванню інтересу до навчального процес, що призводить до накопичення знань в учнів.

Одним із найцінніших навичок здобутих в цій роботі стало вміння розробляти цікавий контент. Оскільки в поставленій задачі була не стільки розробка функціональної частини як візуальної. Такі навички дуже цінні в розробці додатків.

В наш час коли в користувачів є великий вибір між похожими програмними продуктами вже недостатньо випустити продукт з деяким функціоналом, а потрібно ще, щоб він був зручним і красивим.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вежневец А. Разработка Web-приложений на Microsoft Visual Basic .NET и Microsoft Visual C# / Вежневец А. – М.: Издательско-торговый дом «Русская Редакция», 2003. – 704 с.

2. Когнитивное управление в интеллектуальных обучающих системах / [Верлань А. Ф., Пискун А. В., Федорчук В. А.] – Черкаси: Редакційно-видавничий відділ Черкаського інституту управління, 2000. – 88с.

3. Петцолъд Ч. Программирование для Microsoft Windows на C# / Петцолъд Ч. – М.: Издательско-торговый дом «Русская Редакция», 2002. –236 с.

4. Троелсен Э. C# и платформа .NET. Библиотека программиста / Троелсен Э. – СПб.: Питер, 2004. – 796 с.

5. Купцевич Ю.Е. Альманах программиста, том II: ASP.NET, Web-сервисы, WEB-приложения / Купцевич Ю.Е. – М.: Издательско-торговый дом «Русская Редакция», 2003. – 587 с.

6. Фурман Я. А. Microsoft Corporation. Тестирование производительности Web-приложений Microsoft .NET / Фурман Я. А – М.: Издательско-торговый дом «Русская Редакция», 2003. – 352 с.

7. Рейли Д. Создание приложений Microsoft ASP.NET. / Рейли Д. – М.: Издательско-торговый дом «Русская Редакция», 2002. – 480 с.

8. Рихтер Дж. Программирование на платформе Microsoft .NET Framework / Рихтер Дж. – М.: Издательско-торговый дом «Русская Редакция», 2003.– 512 с.

9. Вилдермьюс Ш. Практическое использование ADO.NET. Доступ к данным в Internet / Вилдермьюс Ш. – М.: Издательский дом «Вильяме», 2003. – 288 с.

10. Сеппа Д. Microsoft ADO.NET / Сеппа Д. – М.: Издательско-торговый дом «Русская Редакция», 2003. – 640 с.

11. Андерсон Р. ASP.NET для профессионалов Т.1, Т.2 / Андерсон Р. – Лори, 2005. – 1164 с.

12. Митчелл Скотт, Уолтер С. ASP.NET: советы, рекомендации, примеры / Митчелл Скотт, Уолтер С. – Диалектика, 2002. – 864 с.

13. Ватсон К. C# / Ватсон К. – Лори, 2005. – 862 с.

14. Шилдт Г. C#. Учебный курс / Шилдт Г. – Питер, 2003. – 512 с.

15. Робисон У. C# без лишних слов / Робисон У. – ДМК, 2002. – 352 с.

16. Чен К. Специалисты NIIT Использование C# / Чен К. – Диалектика-Вильямс, 2002. – 528 с.

17. Герберт Ш. Полный справочник по C# / Герберт Ш. – Диалектика-Вильямс, 2004. – 752 с.

18. Петцольд Ч. Программирование в тональности C# / Петцольд Ч. – М.: Издательский дом «Вильяме», 2004. – 512 с.

19. Либерти Д. Программирование на C# / Либерти Д. – Символ, 2003. – 688с.

20. Секунов Н.Ю. Разработка приложений на C++ и C# / Секунов Н.Ю. – Питер, 2003. – 608 с.

21. Микелсен К. Язык программирования C#. Лекции и упражнения. Учебник / Микелсен К. – Dia Soft, 2002. – 656 с.

22. Зайченко Ю.П. Комп’ютерні мережі: Навчальний посібник / Зайченко Ю.П./. – К.: Слово, 2003. – 286 с.

23. Лозікова Г.М. Комп’ютерні мережі / Лозікова Г.М. – К.: Центр навчальної літератури, 2004. – 128 с.

24. Валецька Т. М. Комп’ютерні мережі. Апаратні засоби / Валецька Т. М. – К.: Центр навчальної літератури, 2004. – 208 с.

25. Руденко О.Г. Штучні нейронні мережі / Руденко О.Г., Бодянський Є. В. – Харків: Компанія СМІТ, 2006. – 258 с.

26. Бакурова А. В. Нейронні мережі / Бакурова А. В., Очеретін Д.В., Мартиненко Т. Б.– Запоріжжя: ЗНУ, 2007. – 29 с.

27. Глушков В.М. Нові комп’ютерні засоби, обчислювальні машини та мережі / Глушков В.М.– К.: НАНУ, 2001. – 170 с.

28. Юринець В. Є. Комп’ютерні мережі. Інтернет / Юринець В. Є., Юринець Р. В. – Львів: Видавничий центр ЛНУ ім. І. Франка, 2006. – 270 с.

29. Глазунова О. С. Методи управління стратегією дилерської мережі транснаціональної корпорації / Глазунова О. С. – Донецьк, 2004. – 258 с.

30. Павловская Т.А. С/С++. Программирование на языке высокого уровня / Павловская Т.А. –СПб.:Питер, 2003. – 461 с.

31. Павловская Т.А. С/С++. Практикум / Павловская Т.А., Щупак Ю.А. – СПб.: Питер, 2002. –204 с.

32. Дейтел Х. Как программировать на С++ / Дейтел Х., Дейтел П. – М.: Бином, 2000. –1024 с.

33. Либерти Дж. Освой самостоятельно С++ / Либерти Дж. –М.: Вильямс, 2001. –456 с.

34. Культин Н. С/С++ в задачах и примерах / Культин Н. –СПб.:БХВ-Петербург, 2001. –288 с.