МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
підпис

«____»_____ 20__ р.

ДИПЛОМНИЙ ПРОЕКТ (РОБОТА) БАКАЛАВРА

ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

Засоби і алгоритми створення тривимірних астрономічних об'єктів з використанням чисельного моделювання орбітальної динаміки

Освітньо-кваліфікаційний рівень "бакалавр"
Напрям підготовки 6.050102 – "комп'ютерна інженерія"

Керівник проекту:	_____ (підпис)	Скарга-Бандурова І.С. (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Критська Я.О. (ініціали, прізвище)
Студент:	_____ (підпис)	Шаповалов О.О. (ініціали, прізвище)
Група:		КІ-14бд

Сєвєродонецьк 2018

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень бакалавр
Напрям підготовки 6.050102 – "комп'ютерна інженерія"
(шифр і назва)
Спеціальність _____
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри _____
_____ І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) БАКАЛАВРА

Шаповалову Олександру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Засоби і алгоритми створення тривимірних астрономічних об'єктів з використанням чисельного моделювання орбітальної динаміки

керівник проекту (роботи) Скарга-Бандурова І.С., д.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "14" 05 2018 р. № 118/48

2. Термін подання студентом роботи 17.06.2018

3. Вихідні дані до роботи матеріали переддипломної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз предметної області та постановка задачі, Вибір засобів для розробки, Розроблення алгоритмів для чиельного моделювання орбітальної динаміки, Розроблення тивимірних астрономічних об'єктів; Охорона праці та безпека в надзвичайних ситуаціях.

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

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та безпека в надзвичайних ситуаціях	ст. викл. Критська Я.О.		

7. Дата видачі завдання 14.05.2018

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Ознайомлення з предметною галуззю	03.05 – 15.05	
2	Аналіз існуючих аналогів	16.05 – 18.05	
3	Вибір засобів для розробки	19.05 – 25.05	
4	Розробка алгоритмів та тивимірних	26.05 – 03.06	
	астрономічних об'єктів		
5	Розробка розділу «Охорона праці та	04.06 – 08.06	
	безпека в надзвичайних ситуаціях»		
6	Оформлення пояснювальної записки	09.06 – 15.06	

Студент _____ Шаповалов О.О.
(підпис) (прізвище та ініціали)

Керівник _____ Скарга-Бандурова І.С.
(підпис) (прізвище та ініціали)

ЗМІСТ

Р Е Ф Е Р А Т

Пояснювальна записка до дипломного проекту (роботи) бакалавра: 121 с., 70 рис., 7 табл., 113 бібліографічних., 56 джерел посилань, 1 додаток.

Об'єкт розробки: засоби і алгоритми створення тривимірних астрономічних об'єктів з використанням чисельного моделювання орбітальної динаміки.

Мета роботи: Метою роботи є створення навчального додатка, який дозволяє шкільної або студентської аудиторій придбати знання про сонячну систему або поліпшити їх.

В проекті виконано: (перелічити задачі за розділами)

1. Аналіз засобів побудови астрономічних тривмірних об’єктів.
2. Векторна матиматика в Unity3D.
3. Система частинок і робота освітлення в віртуальному просторі.
4. Розробка користувацького інтерфейсу.
5. Охорона праці та безпека в надзвичайних ситуаціях.

Отримано наступні результати: Розроблена віртуальна модель Сонячної Системи.

Практичне значення, галузь застосування роботи: Додаток може використовуватися в середніх та вищих навчальних закладах.

Ключові слова: вектора; модель; моделювання; кватирніон; система; інтерфейс; анімація; об’єкт.

Умови одержання дипломного проекту: СНУ ім. В. Даля, пр. Центральний 59-А,. м. Сєвєродонецьк, 93400.

ВСТУП

На сьогоднішній день в системі освіти все частіше використовують навчальні програми. Комп'ютерні навчальні програми складають великий клас засобів, що відносяться до освітніх інформаційних технологій. На сьогоднішній день вони забезпечують підтримку навчального процесу нарівні з традиційними навчально-методичними засобами. Однак у порівнянні з традиційними навчально-методичними засобами комп'ютерні навчальні програми забезпечують нові можливості, а багато існуючих функції реалізуються з більш високою якістю.

Опис ідеї даної роботи є створення навчального додатка, за допомогою якого можна швидко отримати інформацію про сонячну систему та планети. Котрий представляє собою віртуальним планетарієм і обсерваторією.

Для реалізацій даного додатка були виділені три основні педагогічні задачі які вирішуються за допомогою комп'ютерних програм: 1) початкове ознайомлення з предметною областю, освоєння її базових понять і концепцій; 2) базова підготовка на різних рівнях глибини і детальності; 3) проведення навчально-дослідних експериментів з моделями досліджуваних об'єктів, процесів і середовища діяльності.

Дотримуючись концепту основних педагогічних задач в проекті потрібно реалізувати 3D візуалізовану сцену сонячної системи, при цьому враховуючи реальні фізичні та геометричні параметри об'єктів сонячної системи та розробити точний синхронізований рух всіх об'єктів сонячної системи. Для проведення студентами та школярами навчально - дослідницьких експериментів. Також розробити інтерфейс для отримання студентам і школярам науково теоретичного матеріал про сонячну систему та планети.

Бакалаврська робота являє собою дослідження засобів і алгоритмів створення тривимірних астрономічних об'єктів з використанням чисельного моделювання орбітальної динаміки.

За результатами роботи опубліковано 2 тез доповідей в матеріалах всеукраїнських конференцій: «Електронні апарати та системи. Проблеми створення. Перспективи розвитку» та «Форум IT Ідея-2017».

Метою роботи є створення навчального додатка, який дозволяє шкільної або студентської аудиторій придбати знання про сонячну систему або поліпшити їх. Для досягнення цієї мети в роботі сформульовані й вирішені наступні завдання:

1. Аналіз схожих програмних продуктів;
2. Вивчення принципів 3D моделювання;

3. Визначено набір інструментів, що використовується в розробці;
4. Розробка віртуальної моделі та алгоритму руху об'єктів;
5. Розробка користувальницького інтерфейсу.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася в рамках науково-дослідної роботи «Дослідження в галузі вищої освіти і практики інституційної співпраці» (№ Д.Р. 0113U002236); тематичних планів науково-дослідних робіт Східноукраїнського національного університету ім. В. Даля (м. Сєвєродонецьк) 2016-2018 років.

Під час виконання НДР автором були обрані й проаналізовані універсальні 3D редактори, аналіз принципів роботи векторів у віртуальному просторі, розробка алгоритмів для взаємодії об'єктів у віртуальному просторі.

Технології, що використовуються для реалізації проекту. Для реалізацій проекту була обрана середовище розробки Unity3D яка є гнучкою і потужною платформою для розробки двох-і тривимірних додатків, що працює на різних операційних системах і ігрових приставках.

Практичне значення отриманих результатів. Основні теоретичні результати роботи реалізовано в наступних практичних додатках і положеннях: Проаналізовано програмні продукти такі як: Інтерактивна Сонячна система, Симулятор сонячної системи, Solar System 3D, Сонячна система. Danik, Сонячна система HD, Тривимірний симулятор сонячної системи - Solar System 3D Simulator. Основними переваги запропонованого рішення від автора є реалізація наступних етапів:

1. Орбітальні характеристики обертання об'єктів навколо нульового об'єкта.
2. Орбітальні характеристики обертання об'єктів навколо власної осі.
3. Міжпланетні відстані.
4. Швидкість обертання об'єктів навколо або нульового об'єкта.

Також виходячи проведеного аналізу вимагаєте розробити графічну модель руху і розташування об'єктів в 3 - х і 2 - х вимірному вигляді. Розробити спеціалізований код для реалізацій вище перерахованих пунктів. Також розробити об'ємне освітлення для віртуального простору. Спроектувати користувальницький інтерфейс для навчання і отримання інформацій про об'єктах.

1 АНАЛІЗ ЗАСОБІВ ПОБУДОВИ АСТРОНОМІЧНИХ ТРИВМІРНИХ ОБ'ЄКТІВ

У розділі надано результати аналізу інформації про 3D моделювання; Виконано опис принципів моделювання тривимірних об'єктів в 3D Studio MAX; Проаналізовано принципи роботи з полігональної сіткою; Виконано аналіз роботи шейдерів в 3D моделюванні; Порівняльна характеристика 3D редакторів. Розроблене технічне завдання на розробку.

1.1 Загальна інформація про 3D моделювання

3D-моделювання - це процес створення тривимірної моделі об'єкта. Завдання 3D-моделювання - розробити візуальний об'ємний образ бажаного об'єкта. За допомогою тривимірної графіки можна і створити точну копію конкретного предмета.

Тривимірна графіка активно застосовується для створення зображень на площині екрану або аркуша друкованої продукції в науці і промисловості, наприклад, в системах автоматизації проектних робіт (САПР; для створення твердо тільних елементів: будівель, деталей машин, механізмів), архітектурної візуалізації (сюди відноситься і так звана «віртуальна археологія»).

Категорію «3D» формують програми для створення і редагування 3D-моделей. Сюди входять як професійні пакети для моделювання, анімації і обробки 3D, так і програмне забезпечення для любителів. За допомогою пропонованих програм можна створити 3D-моделі фактично будь-якого ступеня складності. Також в розділ входять спеціалізовані інструменти, які рекомендуються для більш вузьких цілей [1].

3Ds Max у своєму розпорядженні великі засобами для створення різноманітних за формою і складності тривимірних комп'ютерних моделей, реальних чи фантастичних об'єктів навколишнього світу, з використанням різноманітних технік і механізмів, що включають наступні:

1) Полігональне моделювання, в яке входять Editable mesh (редагована поверхня) і Editable poly (редагований полігон) - це найпоширеніший метод моделювання, використовується для створення складних моделей і нізкополігональних моделей для ігор.

2) Як правило, моделювання складних об'єктів з наступним перетворенням в Editable poly починається з побудови параметричного об'єкта «Box», і тому спосіб моделювання загальноприйнята називати «Box modeling»;

3) Моделювання на основі неоднорідних раціональних B-сплайнів.

4) Моделювання на основі тих чи інших «Сіток шматків» або поверхонь (Editable patch) - підходить для моделювання тіл обертання;

5) Моделювання з використанням вбудованих бібліотек стандартних параметричних об'єктів (примітивів) і модифікаторів.

6) Моделювання на основі сплайнів (Spline) з подальшим застосуванням модифікатора Surface - примітивний аналог NURBS, зручний, проте, для створення об'єктів зі складними перетікають формами, які важко створити методами полігонального моделювання.

7) моделювання на основі сплайнів з подальшим застосуванням модифікаторів Extrude, Lathe, Bevel Profile або створення на основі сплайнів об'єктів Loft. Цей метод широко застосовується для архітектурного моделювання. Методи моделювання можуть поєднуватися один з одним.

Моделювання на основі стандартних об'єктів, як правило, є основним методом моделювання і служить відправною точкою для створення об'єктів складної структури, що пов'язано з використанням примітивів в поєднанні один з одним як елементарних частин складових об'єктів.

1.2 Аналіз і принципи роботи в 3D Studio MAX

1.2.1 Реалізація геометричних принципів в 3D Studio MAX

Тривимірний простір: Працюючи з 3D Studio MAX користувач має справу з уявним тривимірним простором. Тривимірний простір - це куб в кібернетичному просторі, що створюється в пам'яті комп'ютера. Кібернетичний простір відрізняється від реального фізичного світу тим, що створюється і існує тільки в пам'яті комп'ютера завдяки дії спеціального програмного забезпечення.

Однак подібно реального простору, тривимірний простір також необмежено велике. Завдання пошуку об'єктів та орієнтації легко вирішується завдяки використанню координат.

Найменшою областю простору, яка може бути зайнята якимось об'єктом, є точка (point). Положення кожної точки визначається трійкою чисел, які називаються координатами (coordinates). Прикладом координат може служити трійка (0; 0; 0), що визначає центральну точку тривимірного простору, яка також називається початком координат (origin point).

Кожна точка тривимірного простору має три координати, з яких одна визначає висоту, інша - ширину, третя - глибину положення точки. Таким чином, через кожну точку можна провести три координатних осі кіберпростору.

Координатна вісь (axis) - це уявна лінія кіберпростору, що визначає напрямок зміни координати. У MAX є три стандартні осі, звані осями X, Y і Z. Можна умовно вважати, що вісь X представляє координату ширини, вісь Y - висоти, а вісь Z - глибину.

Примітиви: Тривимірні примітиви становлять основу багатьох програмних пакетів комп'ютерної графіки і забезпечують можливість створення різноманітних об'єктів простої форми. У багатьох випадках для формування потрібної моделі тривимірні примітиви доводиться об'єднувати або модифікувати. MAX надає вам два набору примітивів: стандартні (Standard Primitives) і поліпшені (Extended Primitives). До числа стандартних примітивів належать паралелепіпед, сфера, геосфера, конус, циліндр, труба, кільце, піраміда, чайник, призма. Поліпшеними називаються примітиви багатогранник, тороїдальний вузол, паралелепіпед з фаскою, цистерна, капсула, веретено, тіло L-екструзії, узагальнений багатокутник. Працюючи з примітивами майже завжди необхідно вдаватися до їх перетворення або модифікації для створення потрібних об'єктів.

Складені об'єкти - це тіла, складені з двох або більше простих об'єктів (як правило об'єктів примітивів). Створення складених об'єктів є продуктивний метод моделювання багатьох реальних об'єктів. 3D Studio MAX надає можливість використовувати шість типів складових об'єктів:

Морфінгові. Об'єкти даного типу дозволяють виконувати анімацію плавного перетворення одного тіла в інше.

Булевські. Об'єкти цього типу дозволяють об'єднувати два або кілька тривимірних тіл для отримання одного нового. Застосовуються для створення отворів або прорізів в об'ємних тілах або для з'єднання декількох об'єктів в один. Цей тип ідеально підходить для архітектурного моделювання або будь-яких інших завдань, в яких необхідно відняти (виключити) об'єм, який займає одним тілом, з іншого.

Розподілені. Об'єкти цього типу являють собою результат розподілу дублікатів одного тривимірного тіла по поверхні іншого.

Відповідні. Даний тип об'єктів дозволяє змусити одне тривимірне тіло прийняти форму іншого.

Такі що з'єднуються. Цей тип об'єктів дозволяє з'єднати між собою отвори в двох вихідних тілах своєрідним тунелем.

Злиті з формою. Об'єкти цього типу дозволяють з'єднувати форму сплайна з поверхнею тривимірного тіла. Фактично, це дозволяє малювати на поверхнях тривимірних тіл [3].

1.2.2 Моделювання тривимірних об'єктів в 3D MAX

3D Max об'єктно-орієнтована програма, тобто все, що створюється в програмі, є об'єктами. Об'єктами в програмі 3D Max є будь-які геометричні фігури, криві, камери, допоміжні об'єкти, об'ємні деформації, системи і джерела світла, які можуть бути включені до складу сцени.

Всі геометричні об'єкти програми 3D Max можна умовно розділити на дві категорії: параметричні та редаговані.

Більшість об'єктів в 3D Max є параметричними. Параметричні об'єкти - це об'єкти, які визначаються сукупністю установок або параметрів, а не є описом його форми. Простіше кажучи, такі об'єкти можна контролювати за допомогою параметрів (сувій Parameters (Параметри) на командній панелі). Зміна значень параметрів модифікує геометрію самого об'єкта. Такий підхід дозволяє управляти розмірами і формою об'єктів.

Параметричними об'єктами в 3D Max є всі об'єкти, які можна побудувати за допомогою меню Create (Створення). Вони мають важливі настройки моделювання та анімації, тому в загальному випадку необхідно якомога довше зберігати параметричні визначення об'єкта. Однак збереження параметричних властивостей об'єктів витрачає велику кількість ресурсів комп'ютера і уповільнює роботу з об'єктами, так як всі параметри, настройки і модифікатори зберігаються в пам'яті комп'ютера. Якщо ви не припускаєте надалі використовувати параметричні властивості об'єкта, перетворіть його в Editable Mesh (Редагована поверхню). Зміна редагованих об'єктів відбувається за рахунок подоб'єктов (вершини, ребра, грані, полігони) або функцій. До складу редагованих об'єктів входять: Editable Spline (Редагований сплайн), Editable Mesh (Редагована поверхню), Editable Poly (Редагована полігональна поверхня), Editable Patch (Редагована патч-поверхня) і NURBS (NURBS-поверхность). Редаговані об'єкти в стеку модифікаторів містять ключове слово Editable (редагується). Виняток становлять NURBS-об'єкти, які називаються NURBS Surfaces (NURBS-поверхні). Редаговані об'єкти виходять шляхом перетворення інших типів об'єктів. Після перетворення параметричного об'єкта в інший тип (наприклад, в Editable Mesh (Редагована поверхню)) він втрачає всі свої параметричні властивості і не може бути змінений шляхом вказівки параметрів. У той же час

редагований об'єкт набуває властивостей, недоступні параметричного, можливість редагування на рівні подоб'ектов.

Подібно величезному будинку, побудованому з маленьких цеглинок, програма 3D Max дозволяє створювати різнопланові сцени, використовуючи в якості будівельних блоків примітиви (параметричні об'єкти). Ви можете використовувати стандартні параметричні об'єкти для початку будь-якої роботи. Після створення до них можна застосовувати модифікатори, будувати складені об'єкти, розрізати, редагувати на рівні подоб'ектов і виконувати багато інших операцій.

Об'єкти категорії Geometry (Геометрія) в 3D Max є базовим матеріалом для створення більш складних моделей. Для редагування поверхні примітивів використовуються різні інструменти моделювання.

Існують різні підходи до тривимірного моделювання:

– моделювання на основі примітивів;

– використання модифікаторів;

– моделювання сплайна;

– правка редагованих поверхонь: Editable Mesh (Редагована поверхню), Editable Poly (Редагована полігональна поверхня), Editable Patch (Редагована патч-поверхня);

– створення об'єктів за допомогою булевих операцій;

– створення тривимірних сцен з використанням частинок;

– NURBS-моделювання (NURBS - Non Uniform Rational B-Splines, неоднорідні нераціональні B-сплайни) [4].

1.2.3 Полігональна сітка

Полігональна сітка [5, 6]- це сукупність вершин, ребр і граней, які визначають форму багатогранного об'єкта в тривимірній комп'ютерній графіці і об'ємному моделюванні. Гранями зазвичай є трикутники, чотирикутники або інші прості опуклі багатокутники (полігони), так як це спрощує рендеринг, але сітки можуть також складатися і з найбільш загальних увігнутих багатокутників, або багатокутників.

Множина операцій, що проводяться над сітками, може включати булеву алгебру, згладжування, спрощення та багато інших. Різні подання полігональних сіток використовуються для різних цілей і програм. Для передачі полігональних сіток по мережі використовуються мережеві подання, такі як «потокові» і «прогресивні» сітки. Об'ємні сітки відрізняються від полігональних тим, що вони явно представляють і поверхню і обсяг структури, тоді як полігональні сітки явно представляють лише поверхню, а не

обсяг. Так як полігональні сітки широко використовуються в комп'ютерній графіці, для них розроблені алгоритми трасування променів, виявлення зіткнень і динаміки твердих тіл. Об'єкти створені за допомогою полігональних сіток повинні зберігати різні типи елементів, такі як вершини, ребра, грані, полігони і поверхні. У багатьох випадках зберігаються лише вершини, ребра і або межі, або полігони. Рендерер може підтримувати лише трьох-сторонні грані, так що полігони повинні бути побудовані з їх множин, як показано на рис. 1.1. Однак багато рендерер підтримують полігони з чотирма і більше сторонами, або вміють тріангулювать полігони в трикутники на льоту, роблячи необов'язковим зберігання сітки в тріангулірованной формі. Також в деяких випадках, таких як моделювання голови, бажано вміти створювати і трьох - і чотирьох-сторонні полігони.

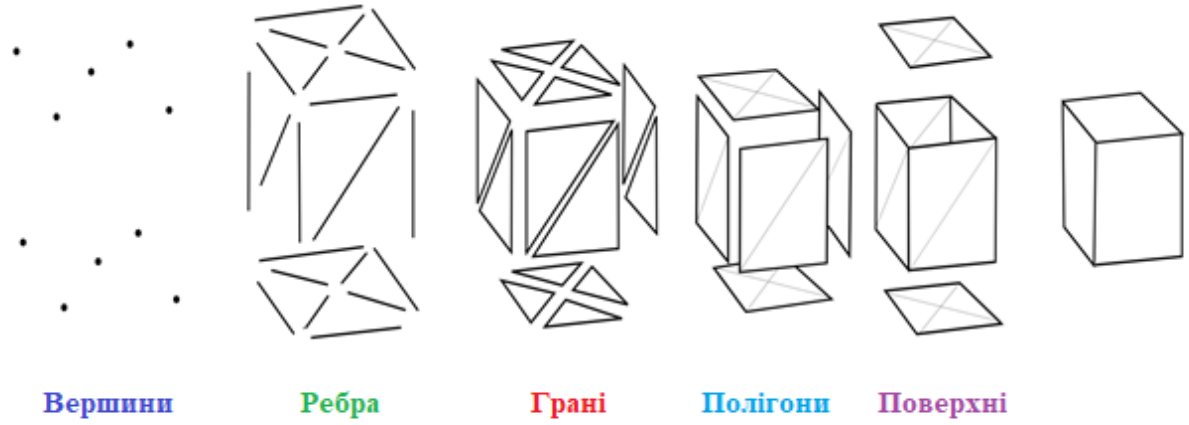


Рисунок 1.1. Елементи моделювання сітки [Автор Colin Smith, 10]

Вершина - це позиція разом з іншою інформацією, такий як колір, нормальний вектор і координати текстури. Ребро - це з'єднання між двома вершинами. Грань - це замкнутий множина ребр, в якому трикутна грань має три ребра, а чотирикутна - чотири. Полігон - це набір компланарних (лежать в одній площині) граней. У системах, які підтримують багатосторонні межі, полігони і межі рівнозначні. Однак, більшість апаратного забезпечення для рендеринга підтримує лише межі з трьома або чотирма сторонам, так що полігони представлені як множина граней. Математично, полігональна сітка може бути представлена у вигляді неструктурованою сітки, або неорієнтованого графа, з додаванням властивостей геометрії, форми і топології.

Вершинне подданя

Вершинне подданя (рис. 1.2) описує об'єкт як множину вершин, з'єднаних з іншими вершинами. Це найпростіше подання, але воно не широко використовується, тому що

інформація про грані і ребрах не виражена явно. Тому потрібно обійти всі дані щоб згенерувати список граней для рендерінга. Крім того, нелегко виконуються операції на ребрах і гранях.

Список вершин

v0	0,0,0	v1 v5 v4 v3 v9
v1	1,0,0	v2 v6 v5 v0 v9
v2	1,1,0	v3 v7 v6 v1 v9
v3	0,1,0	v2 v6 v7 v4 v9
v4	0,0,1	v5 v0 v3 v7 v8
v5	1,0,1	v6 v1 v0 v4 v8
v6	1,1,1	v7 v2 v1 v5 v8
v7	0,1,1	v4 v3 v2 v6 v8
v8	.5,.5,0	v5 v6 v7 v8
v9	.5,.5,1	v0 v1 v2 v3

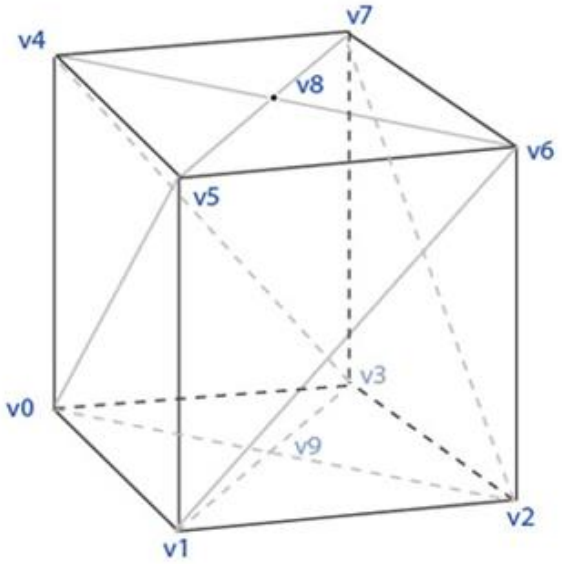


Рисунок 1.2. Вершинноє подання [Автор: Colin Smith, 11]

Однак, сітки ВП отримують вигоду від малого використання пам'яті і ефективної трансформації. Рис. 1.2 показує приклад паралелепіпеда зображений з використанням ВП сітки. Кожна вершина індексує її сусідні вершини. Зауважте, що останні дві вершини, 8 і 9 зверху і знизу паралелепіпеда, мають чотири пов'язаних вершини, а не п'ять. Головна система повинна справлятися з довільним числом вершин пов'язаних з будь-якої даної вершиною.

Список граней

Сітка з використанням списку граней являє об'єкт що має множину граней і множину вершин. Це найбільш широко використовуване подданя.

Список граней краще для моделювання, ніж вершинне подання тим, що він дозволяє явний пошук вершин грані, і граней оточуючих вершину. Рисунок 1.3 показує приклад паралелепіпеда у вигляді сітки з використанням списку граней. Вершина v5 підсвічена, щоб показати межі, які її оточують. Зауважте, що в цьому прикладі у кожній грані обов'язково 3 вершини. Однак це не означає що у кожної вершини одне і те ж кількість оточуючих граней.

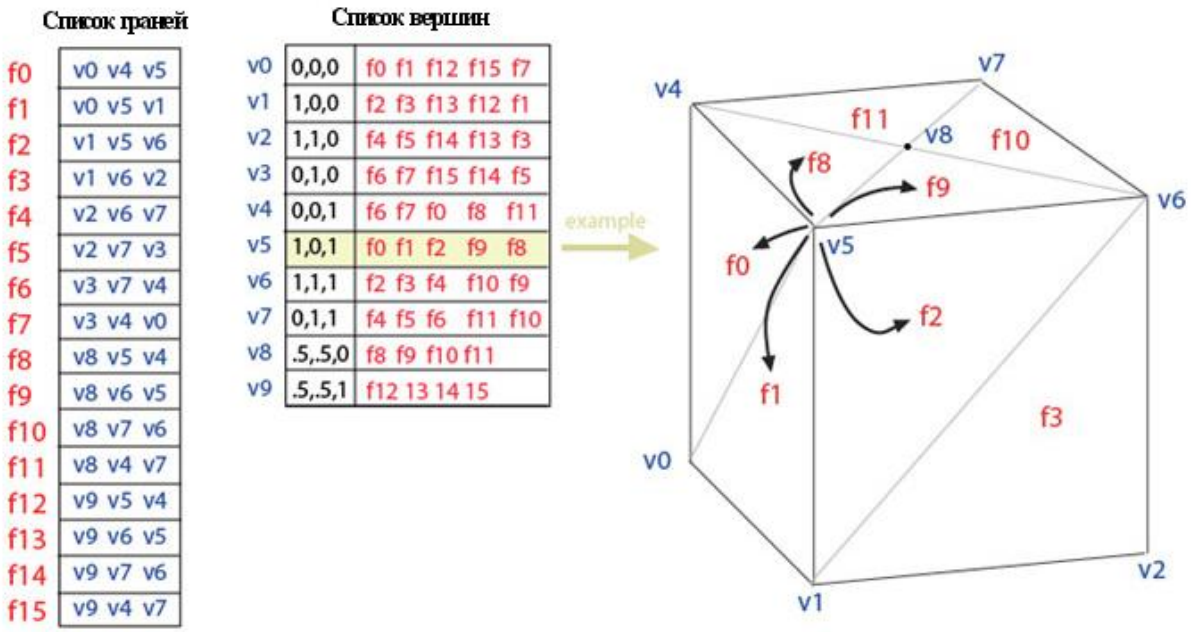


Рисунок 1.3. Список граней [Автор: Colin Smith, 12]

Для рендеринга грань зазвичай надсилається в графічний процесор як множина індексів вершин, і вершини надсилаються як позиція / колір / структури нормалей (на малюнку дана лише позиція). Тому зміни форми, але не геометрії, можуть бути динамічна оновлені просто переславши дані вершини без поновлення зв'язаність граней.

Моделювання вимагає легкого обходу всіх структур. З сіткою використовує список граней дуже легко знайти вершини межі. Також, список вершин містить список всіх граней пов'язаних з кожною вершиною. На відміну від вершинного подання, і межі і вершини явно представлені, так що знаходження сусідніх граней і вершин постійно за часом. Однак, ребра не задані явно, так що пошук все ще потрібен, щоб знайти всі межі, навколишні задану межу. Інші динамічні операції, такі як розрив або об'єднання межі, також складні зі списком граней.

Крилате подання

Крилате подання - представляє вершини, грані і ребра сітки. Це подання широко використовується в програмах для моделювання для надання найвищої гнучкості в динамічній зміні геометрії сітки, тому що можуть бути швидко виконані операції розриву і об'єднання. Їх основний недолік - високі вимоги пам'яті і збільшена складність через вміст множини індексів.

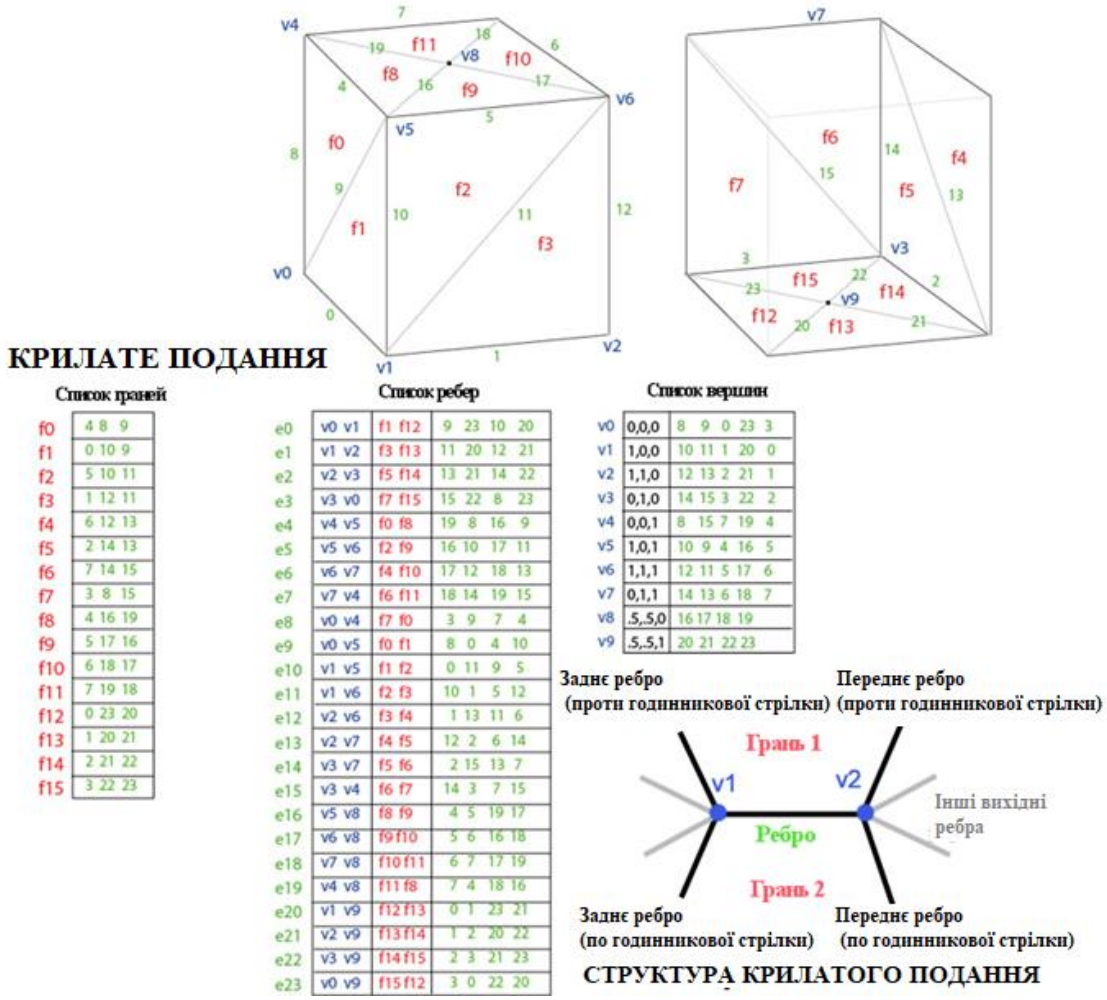


Рисунок 1.4. Крилате подання [Автор: Colin Smith, 13]

"Крилатий" подання вирішує проблему обходу від ребра до ребра і забезпечує впорядковане множина граней навколо ребра. Для будь-якого заданого ребра число вихідних ребр може бути довільним. Щоб спростити це, "крилатий" подання надає лише чотири, найближчі ребра за годинниковою і проти годинникової стрілки на кожному кінці ребра. Інші ребра можна обійти поступово. Тому інформація про кожному ребрі нагадує метелика, тому подання називається "крилатим". Рисунок 1.4 показує приклад паралелепіпеда в "крилатому" поданні. Повні дані по ребру складаються з двох вершин (кінцеві точки), двох граней (по кожну сторону), і чотири ребра ("крила" ребра).

Візуалізація "крилатого" подання графічним обладнанням вимагає генерування списку індексів граней. Зазвичай це робиться тільки коли змінюється геометрія. "Крилатий" подання ідеально підходить для динамічної геометрії, такий як підрозділ поверхонь і інтерактивне моделювання, так як зміни сітки можуть відбуватися локально. Обхід навколо сітки, що може стати в нагоді для виявлення зіткнень, може бути ефективно виконано.

1.3 Робота шейдеров в 3D моделювань

Шейдер – комп'ютерна програма, призначена для виконання процесорами відеокадри (GPU). Шейдери складаються на одному зі спеціалізованих мов програмування і компілюються в інструкції для GPU.

Програми, що працюють з тривимірною графікою і відео (ігри, GIS, CAD, CAM і ін.), Використовують шейдери для визначення параметрів геометричних об'єктів або зображення, для зміни зображення (для створення ефектів зсуву, відображення, заломлення, затемнення з урахуванням заданих параметрів поглинання і розсіювання світла, для накладення текстур на геометричні об'єкти і ін.).

Вершинні шейдери

Верховий шейдер оперує даними, пов'язаними з вершинами багатогранників, наприклад, з координатами вершини (точки) в просторі, з текстурними координатами, з кольором вершини, з вектором дотичній, з вектором бінормалі, з вектором нормалі. Верховий шейдер може використовуватися для видового і перспективного перетворення вершин, для генерації текстурних координат, для розрахунку освітлення.

Геометричні шейдери

Геометричний шейдер, на відміну від вершинного, здатний обробити не тільки одну вершину, а й цілий примітив. Примітивом може бути відрізок (дві вершини) і трикутник (три вершини), а при наявності інформації про суміжних вершинах для трикутного примітиву може бути оброблено до шести вершин. Геометричний шейдер здатний генерувати примітиви (Не задіюючи при цьому центральний процесор).

Піксельні шейдери

Піксельний шейдер працює з фрагментами растрового зображення і з текстурами - обробляє дані, пов'язані з пікселями (наприклад, колір, глибина, текстурні координати). Піксельний шейдер використовується на останній стадії графічного конвеєра для формування фрагмента зображення [7, 8, 9].

1.4 Порівняльна характеристика 3D редакторів

Для створення комп'ютерної графіки використовують множина різних додатків. Універсальні 3D редактори (Cinema 4D, 3Ds Max, Maya, Houidini і т.д.). Універсальні 3D редактори, як правило, містять все необхідне для CG: інструменти моделювання, анімації і візуалізації. При виборі програми вимагаєте звернути увагу на наступні чинники:

– функціонал програми;

– зручність користування (інтуїтивний інтерфейс і т.д.).

У цьому розділі будуть порівнюватися такі редактори як: 3D MAX, Cinema 4D, Blender, Softimage.

3Ds MAX

3Ds Max - серед 3D редакторів, дуже популярний інструмент, №1 у виборі багатьох початківців і просунутих фахівців. Займає провідні позиції в сфері дизайну та архітектурної візуалізації. Часто використовується в ігровій індустрії.

Можливості:

– моделювання на основі полігонів, сплайнів і NURBS,

– потужна система частинок,

– модуль волосся / шерсть,

– розширені шейдери Shader FX,

– підтримка нових і вдосконалених механізмів Iray і mental ray.

– імпорт з Revit і SketchUp,

– інтеграція композітінга.

Плюси: величезний функціонал, множина плагінів і навчальної інформації.

Мінуси: не такий простий в освоєнні, потрібні серйозні оновлення додатка.

Cinema 4D

Cinema 4D - один з найкращих і зручних 3D пакетів. Величезний функціонал: від моделювання, анімації, ефектів до «ліплення» і модуля BodyPaint 3D. У Cinema 4D більш зрозумілий і зручний інтерфейс ніж у 3Ds Max. Широко використовується в моушен-дизайні, кіноіндустрії та рекламі.

Можливості:

– полігональне і NURBS-моделювання;

– BodyPaint 3D (модуль для створення розгорток UV і карт текстур);

– генерація і анімація об'єктів;

– персонаж анімація;

– динаміка м'яких і твердих тіл;

– модуль для створення реалістичних волосся;

– система частинок Thinking Particles;

– непоганий вбудований визуализатор.

Плюси: легкість в освоєнні, інтуїтивний інтерфейс, відмінний функціонал, множина навчальних матеріалів, тісний зв'язок з Adobe After Effects, Houdini і т.д.

Мінуси: несправність система переходу між версіями.

Maya

Maya - промисловий стандарт 3D графіки в кіно і телебаченні. Maya популярна серед великих студій і масштабних проектів в рекламі, кіно, ігрової індустрії. Пакет ідеальний для створення анімації.

Можливості:

- повний набір інструментів для NURBS- і полігонального моделювання;
- розвинена система частинок;
- технологія Maya Fur (створення хутра, волосся, трави);
- технологія Maya Fluid Effects (моделювання рідин, атмосфери);
- динаміка твердих і м'яких тіл;
- широкий набір засобів створення динамічних спецефектів;
- UV-текстури, нормалізує обмін речовин і колірне кодування;
- багатопроцесорний гнучкий рендеринг.

Плюси: величезний функціонал і можливості.

Мінуси: тривала і складна навчання, високі вимоги до системи.

Виходячи з аналізу програмних продуктів, було виявлено що:

- 3Ds Max - комп'ютерні ігри, інтер'єри, візуалізація.
- Maya - анімація, кіноіндустрія, телебачення, кліпи.
- Cinema 4D - спецефекти в кіно і телебаченні, моушен-дизайн, реклама.

Для даного дипломного проекту після проведення аналізу була обрана програма 3Ds Max. Оскільки даний проект розробляється на движку Unity 3D і рентабельним вибором буде 3Ds Max.

1.5 Технічне завдання на розробку

Назва розробки

Засоби и алгоритми створення тривимірних астрономічних об'єктів з використання чисельного моделювання орбітальної динаміки.

Призначення розробки

Призначенням розробки є створення навчального додатка, який дозволяє шкільної або студентської аудиторій придбати знання про сонячну систему або поліпшити їх.

Основною ідеєю розробленого додатка є швидке отримання інформацію про Сонячної систему і її об'єктів (планет). Дане додатки є віртуальний планетарій і обсерваторій.

Вимоги до функціональних характеристик

Для реалізацій цього додатка були виділені три основні педагогічні завдання, які вирішуються за допомогою комп'ютерних програм:

1) Початкове ознайомлення з предметною областю, освоєння її базових понять і концепцій;

2) Базова підготовка на різних рівнях глибини і детальності;

3) Проведення навчально-дослідних експериментів з моделями досліджуваних об'єктів, процесів і середовища діяльності.

Дотримуючись концепту основних педагогічних завдань в проекті було реалізовано 3D візуалізовану сцену сонячної системи, при цьому з огляду на реальні фізичні та геометричні параметри об'єктів сонячної системи і розробити точний синхронізоване рух всіх об'єктів сонячної системи. Потрібно провести такий ряд розробки:

1) Розробити 17 об'єктів сферичної форми та дві багатокутних полігональних моделей.

2) Експортування розроблених 3D моделей в Unity3D для подальшої експлуатації.

3) Розробити спеціалізованого коду на мові C#. Даний код повинен виконувати ряд поставлених завдань для роботи з об'єктами.

4) Розробка ряду анімацій для різних об'єктів і провести роботу з освітленням у віртуальному просторі.

5) Розробити інтуїтивний призначений для користувача інтерфейс. Однією з основною метою даного інтерфейсу повинна мати функцію навчання і надання конкретної інформацій про об'єктах.

Вимоги до інтерфейсу

Інтерфейс системи повинен забезпечувати інтуїтивно зрозуміле уявлення про структуру, розміщену на ньому інформацію, швидкий і логічний перехід до панелей та інших сцен. Навігаційні елементи і функціональні кнопки повинні забезпечувати однозначне розуміння, умовні позначення повинні відповідати загальноприйнятим.

Інтерфейс користувача повинен забезпечувати наочне, інтуїтивно зрозуміле уявлення структури розміщеної на ньому інформації, швидкий і логічний перехід до панелей і сцен.

Розробити головне меню, яка матиме кілька функцій. Першою функцією буде запуск безпосередньо завантаження основної сцени проекту. Друга функція є налаштуванням різних параметрів проекту таких як аудіо і відео.

Також спроектовано керівництво для користувача, яка пояснює принцип роботи інтерфейсу. В цей посібник має входить інформація про кнопки.

Спроектувати панелі які представлятимуть різну інформацію про об'єктах. Панель яка містить різні зміни роботи об'єктів у віртуальному просторі.

Вимоги до програмного забезпечення

Навчальні програми повинен бути розрахований для роботи в сучасних версіях наступних операційних систем: Windows 7; Windows 8; Windows 10; Mac OS. Так само в перспективі даний проект буде експортовано на Android і IOC.

Вимоги до апаратного забезпечення

Мінімальні системні вимоги для запуску програмного забезпечення:

1. Процесор - з тактовою частотою не менше ніж 2 ГГц та кількістю ядр не менш ніж 2;
2. Оперативна пам'ять – не менше ніж 2 ГБ;
3. Відеоадаптер: GeForce 6800 / ATI HD 2400 XT з 256 МБ пам'яті.
4. DirectX: версії DirectX 9.0c.
5. Вільне місце на жорсткому диску: - 30 МБ.

Висновок до першого розділу

У теперішній час 3D моделювання є основним частиною розробки програмних продуктів. Основними інструментами розробки є різні 3D редактори, які дозволяють створювати бажані об'єкти. В даному проекті був обраний редактор 3D Max Studio. Одне з основних призначень 3D Max - моделювання тривимірних об'єктів. Уява дизайнера тривимірної графіки дуже часто малює сцени, які неможливо створити, використовуючи тільки примітиви. Багато об'єктів, які оточують нас у повсякденному житті, мають несиметричну поверхню, відтворити яку в тривимірній графіці досить складно.

Програма 3D Max дозволяє створювати різнопланові сцени, використовуючи в якості будівельних блоків примітиви (параметричні об'єкти). Ви можете використовувати стандартні параметричні об'єкти для початку будь-якої роботи. Після створення до них можна застосовувати модифікатори, будувати складені об'єкти, розрізати, редагувати на рівні подоб'ектов і виконувати багато інших операцій.

Тому була проведена дослідницька робота і аналіз інформація про 3D моделювань, а саме: Основні принципи роботи в 3D Max; Аналіз моделювання тривимірних об'єктів. Після вивчення і аналізу інформації було спроектовано 17 об'єктів сферичної форми та дві багатокутних полігональних моделей. Дані моделі були експортуванні в unity3D для подальшої експлуатації

Список використаної літератури в першому розділі

1. Большаков, Д. І., 3D моделювання / Большаков Д. І.: Техатека, 2011 року.

2. Бочков, М. Д., Основи 3D-моделювання / Бочков, М. Д.,: Питер, 2003.

3. Бондаренко С.В., Бондаренко М. Ю. 3ds Max. Бібліотека, 2011 року.

4. Маров М. Н.3ds max. Матеріали, освітлення та візуалізація, 2013.

5. Colin Smith, On Vertex-Vertex Meshes and Their Use in Geometric and Biological Modeling.

6. Bruce Baumgart, Winged-Edge Polyhedron Representation for Computer Vision. National Computer Conference, May 1975.

7. Боресков А. В. Розширення OpenGL. - БХВ-Петербург, 2005.

8. Олексій Боресков. Розробка та налагодження шейдерів. - БХВ-Петербург, 2006.

9. «Orange Book» - OpenGL Shading Language by Randi J. Rost, Bill M. Licea-Kane, Dan Ginsburg and John M. Kessenich.

10. https://ru.wikipedia.org/wiki/Полигональная_сетка#/media/File:Mesh_overview-rus.svg.

11. https://upload.wikimedia.org/wikipedia/commons/f/f4/Mesh_vv_rus.jpg.

12. https://ru.wikipedia.org/wiki/Полигональная_сетка#/media/File:Mesh_fv_rus.jpg.

13. https://ru.wikipedia.org/wiki/Полигональная_сетка#/media/File:Mesh_we_rus.jpg.

14. http://www.photopyx.com/scachat-texturu/planety/.

2 ВЕКТОРНА МАТЕМАТИКА В UNITY3D

У розділі виконано ряд завдань, такі як: Аналіз загальної інформації про середовище розробки Unity3D; Використання векторної математики для побудови віртуальної моделі; Робота класу Vector3 в Unity3D; Аналіз роботи кутів Ейлера з об'єктами; Аналіз роботи Кватерніонів з об'єктами; Робота класу Quaternion в Unity3D; Порівняльна характеристика 3D платформ для розробки ігрових додатків; Побудова віртуальної моделі в 2 - х і 3 - х вимірному просторі; Розрахунки для об'єктів віртуальної моделі; Реалізація віртуальної моделі на мові C#; Векторна робота з камерою; Реалізація позиційної камери; Прив'язка камери до дочірніх об'єктів по точках; Створення додаткової функції; відстеження камерою найменування об'єктів.

2.1 Загальна інформація про середовище розробки Unity3D

Unity3D є гнучкою і потужною платформою для розробки двох-і тривимірних додатків, що працює на різних операційних системах і ігрових приставках з підтримкою модульною системою компонентів. Unity3D підтримує дві скрипотові мови: C #, JavaScript (модифікація). Розрахунки фізики виробляє фізичний движок PhysX від NVIDIA.

Проект в Unity ділиться на сцени (рівні) - окремі файли, що містять свої ігрові сцени зі своїм набором об'єктів, сценаріїв, і налаштувань. Сцени можуть містити в собі як, власне, об'єкти (моделі), так і порожні ігрові об'єкти , які не мають моделі. Об'єкти, в свою чергу містять набори компонентів, з якими і взаємодіють скрипти. Також у об'єктів є назва (в Unity допускається наявність двох і більше об'єктів з однаковими назвами), може бути тег (мітка) і шар, на якому він повинен відображатися. Так, у будь-якого об'єкта на сцені обов'язково присутній компонент Transform - він зберігає в собі координати місця розташування, повороту і розмірів об'єкта по всіх трьох осях. У об'єктів з видимої геометрією також за замовчуванням присутній компонент Mesh Renderer, що робить модель об'єкта видимою. До об'єктів можна застосовувати колізії (collider).

Також Unity підтримує фізику твердих тіл і тканини, а також фізику типу Ragdoll. У редакторі є система успадкування об'єктів; дочірні об'єкти будуть повторювати всі зміни позиції, повороту і масштабу батьківського об'єкта. Скрипти в редакторі прикріплюються до об'єктів у вигляді окремих компонентів.

У Unity можна згенерувати alpha-канал, mip-рівні, normal-map, light-map, карту відображень, проте безпосередньо на модель текстуру прикріпити не можна, буде створений матеріал, з яким буде призначений шейдер, і потім матеріал прикріпиться до

моделі. Редактор Unity підтримує написання і редагування шейдеров. Редактор Unity має компонент для створення анімації, але також анімацію можна створити попередньо в 3D-редакторі та імпортувати разом з моделлю, а потім розбити на файли.

Unity 3D підтримує систему Level Of Detail, суть якої полягає в тому, що на далекій відстані від гравця високо деталізовані моделі замінюються на менш деталізовані, і навпаки, а також систему Occlusion culling, суть якої в тому, що у об'єктів, які не потрапляють в поле зору камери не візуалізується геометрія і колізія, що знижує навантаження на центральний процесор і дозволяє оптимізувати проект [1].

2.2 Використання векторної математики для побудови віртуальної моделі

Вектор є математичним об'єктом, що характеризується величиною і напрямком. В геометрії і в природничих науках вектор є спрямований відрізок прямої в евклідовому просторі (або на площині).

Основних напрямів вектора є: радіус-вектор, швидкість, напрямок. Якщо в просторі задана система координат, то вектор однозначно задається набором своїх координат. Вектор в математиці розглядається як елемент деякого векторного (лінійного) простору.

Вектор розташування (також званий «радіус-вектором») показує, що об'єкт коштує в двох метрах на схід і на в одному метрі на північ від вихідної точки (рис. 2.1).

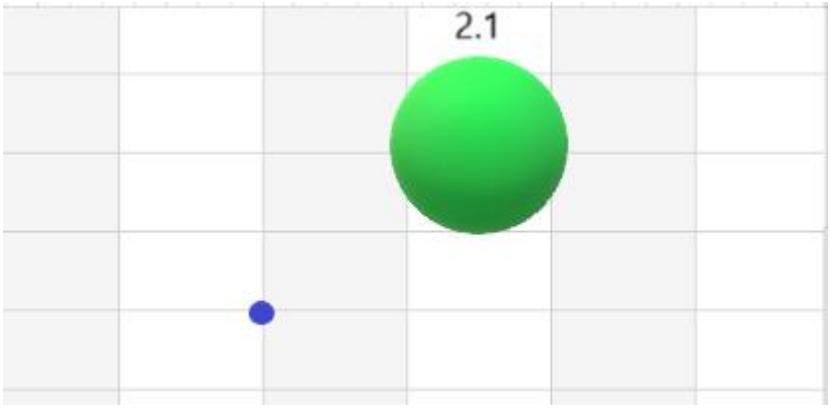


Рисунок 2.1. Вектор розташування «радіус-вектор» [Автор: Шаповалов Олександр]

Вектор швидкості показує, що за одиницю часу об'єкта переміщається на три кілометри вгору і на два - вліво (рис. 2.2).

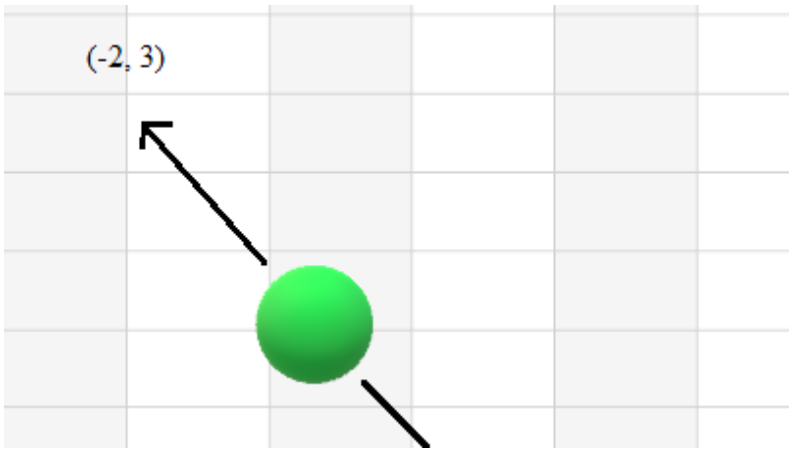


Рисунок 2.2. Вектор швидкості [Автор: Шаповалов Олександр]

Вектор спрямування говорить нам про те, що об'єкта спрямований вправо (Рисунок 2.3).

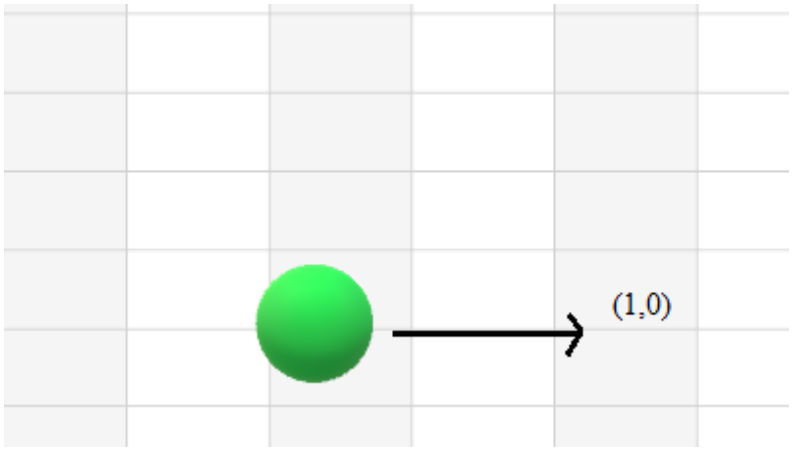


Рисунок 2.3. Вектор напрямку [Автор: Шаповалов Олександр]

Система координат це - комплекс визначень, який реалізує метод координат, тобто спосіб визначати положення і переміщення точки або тіла за допомогою чисел або інших символів. Сукупність чисел, що визначають положення конкретної точки, називається координатами цієї точки.

Вектор є одним з основоположних понять лінійної алгебри. При використанні найбільш загального визначення векторами виявляються практично всі досліджувані в лінійної алгебри об'єкти, в тому числі матриці й тензори.

Точки (Points) і Відрізки (Segments)

Будь-яка точка в 3D програмуванні задається набором з 3-х координат по ортогональним (взаємно перпендикулярним) осях (x, y, z) відповідно. Точками задаються також координати вершин об'єкта.

Відрізок - частина прямої, що з'єднує дві точки. Здається координатами кінців.

Довжина відрізка обчислюється за теоремою Піфагора:

$$len = \sqrt{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2 + \left(z_2 - z_1\right)^2}$$ де $\left(x_2, y_2, z_2\right)$ координати кінців відрізка.

Відрізок і точка мають розташування в просторі.

Спрямований відрізок вектора

У 3D програмуванні рівними вважаються всі вектора, що мають однакову довжину (або модуль вектора) і напрям. У вектора в даному розумінні немає розташування, на відміну від відрізка або точки. Тому будь-який вектор можна задати лише напрямом (тобто одиничним, нормалізованим вектором або ортом) і його довжиною.

Вектор задається лише координатами (по 3-м осях) кінця вектора, вважаючи, що початок вектора знаходиться на початку координат. Нехай

$$\vec{A}\left(x_1, y_1, z_1\right)$$

Тоді модуль цього вектора обчислюється таким чином:

$$\left|\vec{A}\right| = \sqrt{x_1^2 + y_1^2 + z_1^2}.$$

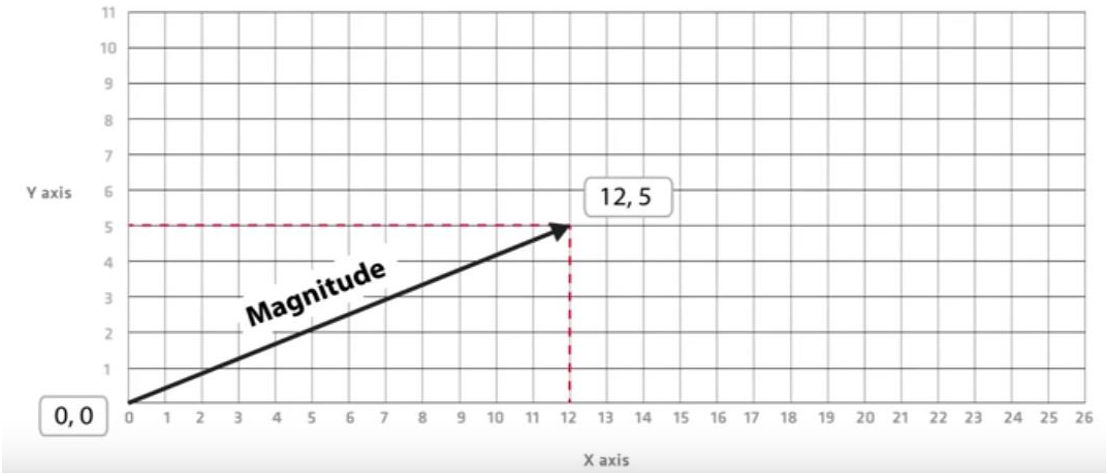


Рисунок 2.4. Знаходження направленого відрізка в 2-х вимірному просторі [Автор: Шаповалов Олександр]

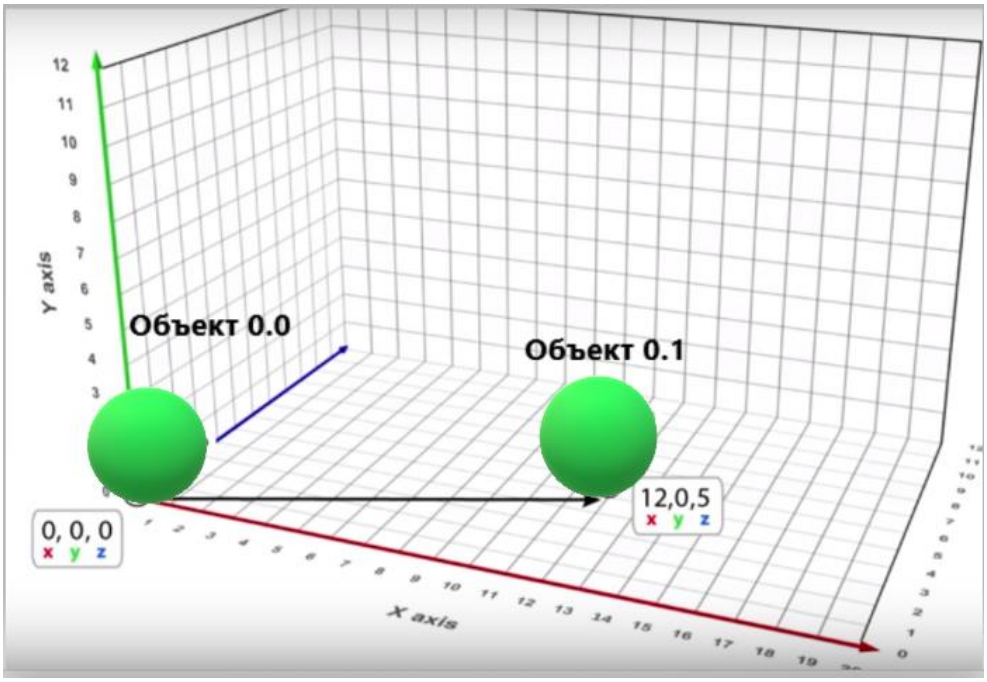


Рисунок 2.5. Знаходження направленого відрізка в 3-х мірному просторі [Автор Шаповалов Олександр]

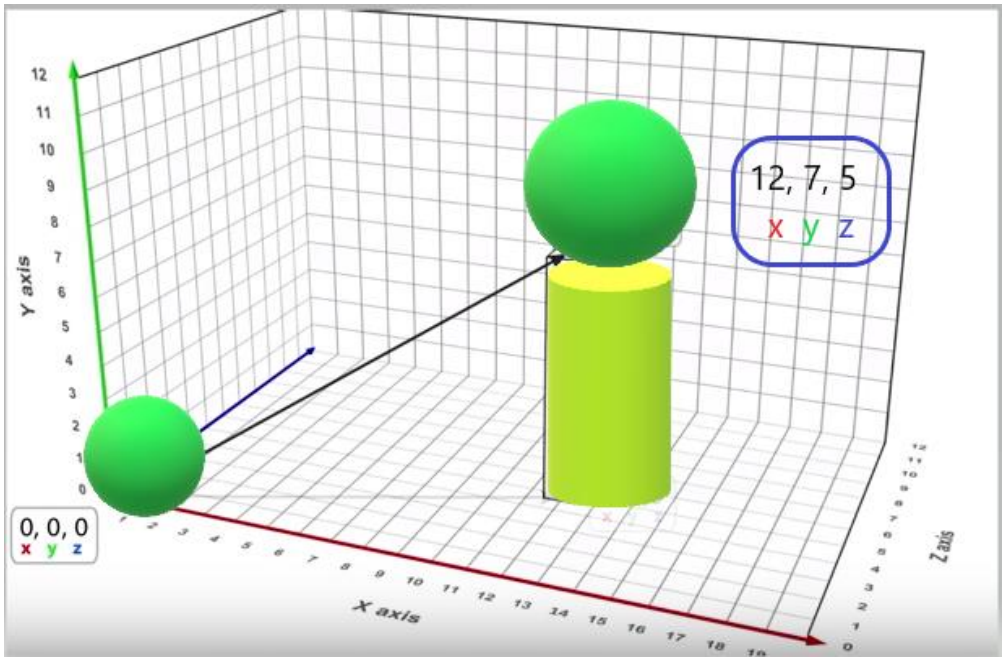


Рисунок 2.6. Знаходження направленого відрізка в 3-х мірному просторі [Автор Шаповалов Олександр]

Сума векторів

Щоб скласти вектора, нам треба просто скласти кожну їх складову один з одним.

Сумою двох векторів $\overrightarrow{A}\left(x_1,y_1,z_1\right)$ и $\overrightarrow{B}\left(x_2,y_2,z_2\right)$ називається вектор $\overrightarrow{C}\left(x_1+x_2,y_1+y_2,z_1+z_2\right)$.

Сума векторів у віртуальному просторі застосовується для фізичного інтегрування. Будь-який фізичний об'єкт буде мати вектора для розташування, швидкості і прискорення. Для кожного кадру (зазвичай це одна шістдесята частина секунди), потрібно інтегрувати два вектора: додати швидкість до місця розташування і прискорення до швидкості.

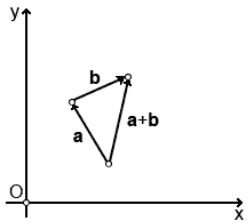


Рисунок 2.7. Додавання двох векторів [18]

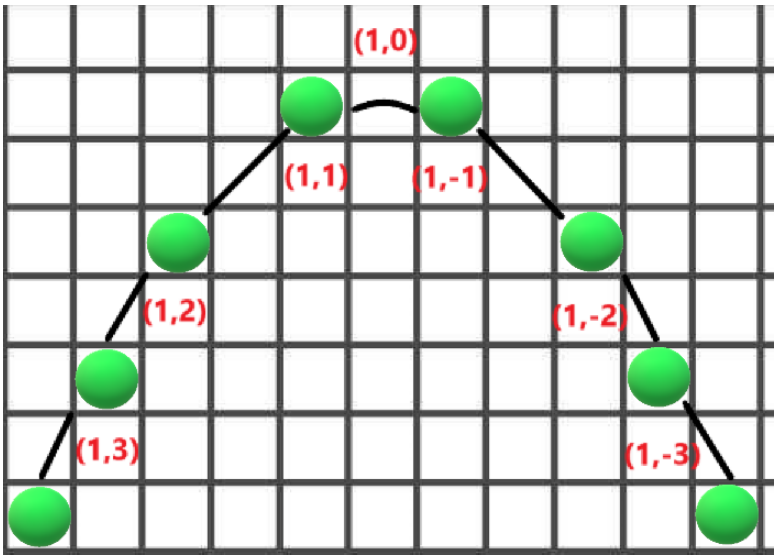


Рисунок 2.8. Сума векторів при русі об'єкта [Автор Шаповалов Олександр]

Для першого кадру, ми додаємо швидкість об’єкта (1, 3) до його розташування (0, 0) і отримуємо його нові координати (1, 3). Потім складаємо прискорення (0, -1) з його швидкістю (1, 3) і отримуємо нове значення швидкості об'єкта (1, 2).

Робимо те-ж саме для другого кадру. Додаємо швидкість (1, 2) до місця розташування (1, 3) і отримуємо координати (2, 5). Потім додаємо прискорення (0, -1) до його швидкості (1, 2) і отримуємо нову швидкість (1, 1).

Віднімання векторів

Віднімання векторів використовується, щоб дізнатися відстань і напрямок одного об'єкта щодо іншого. Для отримання різниці в координатної формі треба відняти відповідні координати векторів:

$$\vec{a} - \vec{b} = (a_x - b_x, a_y - b_y, a_z - b_z)$$

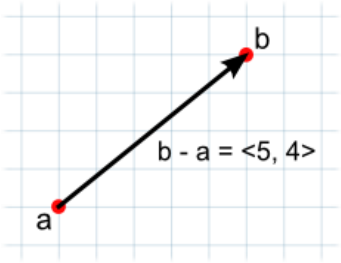


Рисунок 2.9. Віднімання векторів [16]

Нормалізація вектора

Це перетворення заданого вектора в вектор в тому ж напрямку (тобто в колінеарний, паралельний вектор), але довжиною в один символ.

Для нормалізації вектора потрібно кожну компоненту поділити на довжину вектора. Довжина вектора (або модуль) в 3-х мірному просторі з Евклідовій метрикою визначається наступною формулою:

$$length = \sqrt{(x * x + y * y + z * z)}$$

Скалярні множення і ділення

Скаляр, математичне число, що має тільки величину, на відміну від вектора, який має ще й напрямок. Маса і енергія є скалярними величинами, тоді як вага і сила представлені векторними величинами.

Множення вектора на скаляр дає в результаті вектор, з тим же напрямком, що і вихідний вектор. Проте, величина нового вектора дорівнює початкової величини помноженої на скалярний значення.

Аналогічно, скалярний розподіл ділить вихідну величину вектора на скаляр.

Ці операції корисні, коли вектор вдає із себе зміщення руху або силу. Вони дозволяють вам змінити величину вектора без впливу на його напрямок.

Коли будь-який вектор ділиться на власну величину, то в результаті виходить вектор величиною 1, відомий як нормований (одиничний) вектор. Якщо нормований вектор помножити на скаляр, то величина результату буде дорівнює значенню скаляра. Це корисно, коли напрям сили постійно, а величина - немає.

Формула множення вектора на скаляр:

$$S * (Ax, By, Cz) = (S * Ax, S * By, S * Cz) = (S, Ax, By, Cz)$$

Формула розподілу вектора на скаляр:

$$S * (Ax, By, Cz) = (S : Ax, S : By, S : Cz) = (S, Ax, By, Cz)$$

Скалярний добуток (Dot Product)

Скалярний добуток отримує 2 вектора і повертає скаляр. Цей скаляр дорівнює добутку величин цих векторів, помноженому на косинус кута між ними. Коли обидва вектори - нормовані, косинус стверджує, як далеко перший вектор простягається в напрямку другого або навпаки. Формула скалярного добутока:

$$\overline{a} \cdot \overline{b} = a_x \cdot b_x + a_y \cdot b_y + a_z \cdot b_z$$

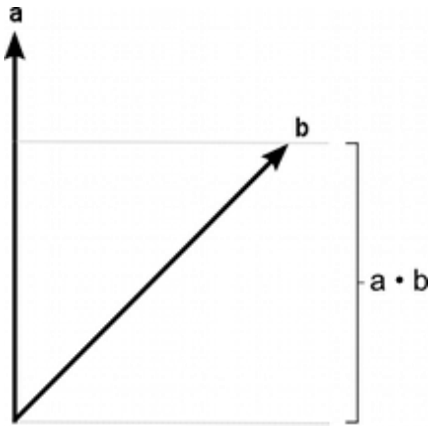


Рисунок 2.10. Скалярний добуток двох векторів [17]

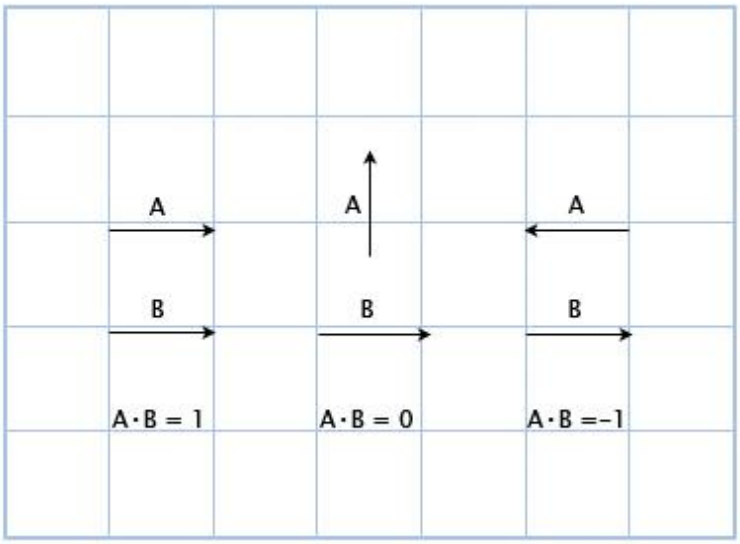


Рисунок 2.11. Скалярний добуток двох векторів [18]

Якщо вектора вказують в одному напрямку, то їх скалярний добуток більше нуля. Коли вони перпендикулярні один одному, то скалярний добуток дорівнює нулю. Коли вектора вказують в протилежних напрямках, їх скалярний добуток менше нуля.

В основному, за допомогою скалярного множення векторів можна розрахувати, скільки їх вказує в одному напрямку.

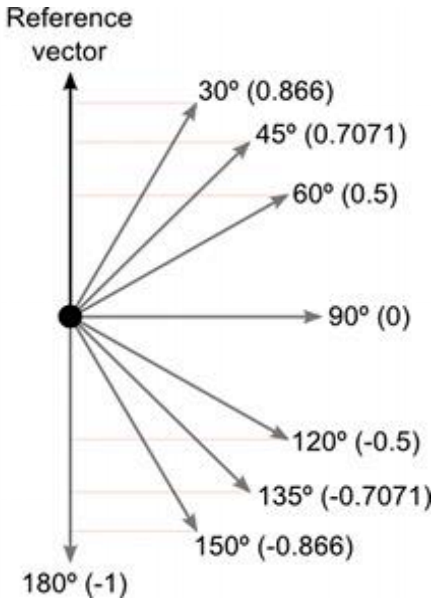


Рисунок 2.12. Знаходження кута огляду об'єкта [19]

Математичний вираз скалярного множення знаходження кутів огляду векторів виглядає так:

$$A * B = |A||B|cos\Delta$$

.

Де Δ кут між векторами A і B.

Це дозволяє нам знайти Δ (Кут) за допомогою формули:

$$\Delta = acos([AB]/[|A||B|])$$

.

Якщо A і B нормалізовані, то вираз спрощується таким чином:

$$\Delta = acos(AB)$$

.

Векторне множення (Cross Product)

Векторний добуток застосовувати тільки для 3D векторів. Воно використовує 2 вектора як вхідну інформацію і повертає ще один вектор в якості результату.

Підсумковий вектор перпендикулярний двом вихідним векторах. Якщо векторний множення A і B буде дорівнює вектору C, то він буде перпендикулярний A і B.

Величина результату дорівнює множення величин вихідних векторів, помноженому на синус кута між ними.

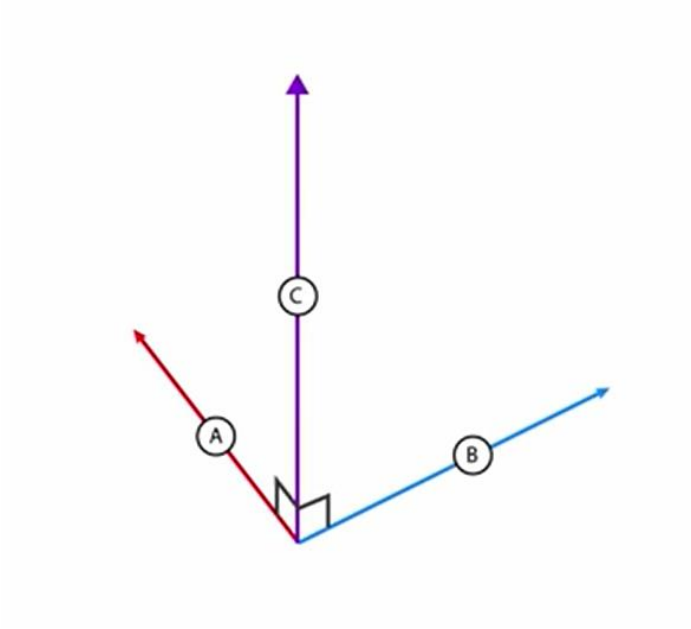


Рисунок 2.13. Векторне множення A * B = C [20]

Формула знаходження векторного множення:

$$\begin{pmatrix} Ax \\ Ay \\ Az \end{pmatrix} \begin{matrix} \times \\ \\ \\ \\ \\ \\ \end{matrix} \begin{pmatrix} Bx \\ By \\ Bz \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \end{matrix} \begin{pmatrix} Ax \\ Ay \\ Az \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \end{matrix} \begin{pmatrix} Bx \\ By \\ Bz \end{pmatrix} = \begin{pmatrix} Ay*Bz - Az*By \\ Az*Bx - Ax*Bz \\ Ax*By - Ay*Bx \end{pmatrix} = \begin{pmatrix} Cx \\ Cy \\ Cz \end{pmatrix}$$

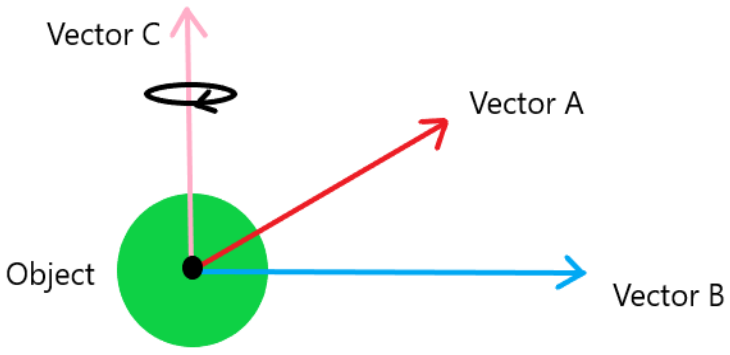


Рисунок 2.14. Обертання об'єкта по власної осі [Автор Шаповалов Олександр]

Vector A - є початковим напрямком об'єкта, а Vector B є новим напрямком. Що б направити об'єкт за новим вектору, потрібно розгорнути об'єкт по Vector C [2][3][4][5].

2.2.1 Клас Vector3 в Unity3D

Клас Vector3 являє собою набір тривимірних векторів і точок. Ця структура використовується в Unity3D для передачі 3D - позицій. Так само містить різні функції для виконання спільних векторних операцій [6].

Статичні властивості

Back	змінює значення вектора Vector3 (0, 0, -1)
Down	змінює значення вектора Vector3 (0, -1, 0)
Forward	змінює значення вектора Vector3 (0, 0, 1)
Left	змінює значення вектора Vector3 (-1, 0, 0)
One	змінює значення вектора Vector3 (1, 1, 1)
Zero	змінює значення вектора Vector3 (0, 0, 0)
Up	змінює значення вектора Vector3 (0, 1, 0)
Right	змінює значення вектора Vector3 (1, 0, 0)

Властивості

Magnitude	Повертає довжину вектора.
Normalized	Повертає вектор з величиною 1.
sqrMagnitude	Повертає квадрат довжини вектора.
This[int]	Отримання доступу до компонентів x, y, z.
X	компонент вектора X.
Y	компонент вектора Y.
Z	компонент вектора Z.

Конструктор

Vector3	Створює новий вектор із заданими компонентами x, y, z.

Методи

Equals	Повертає значення вектора.
Set	Створює x, y і z компоненти для існуючого вектору.
ToString	Повертає відформатований рядок для вектора.

Статичні методи

Angle	Повертає кут між двома векторами

ClampMagnitude	Повертає копію вектора з його величиною.
Cross	Повертає нормаль площині утворений векторами A і B.
Distance	Повертає відстань між вектором A і B.
Dot	Визначає скалярний твір між двома векторами A і B.
Lerp	Визначає лінійну інтерполяцію між двома векторами.
Max	Повертає вектор, який зроблений з найбільших компонентів двох векторів.
Min	Повертає вектор, який зроблений з найбільших компонентів двох векторів.
Normalize	Робить вектор величиною рівною 1.
MoveToward	Переміщення по вектору.
OrthoNormalize	Робить вектори нормалізованими і ортогональними один одному.
Project	Проектує вектор на інший вектор.
ProjectOnPlane	Проектує вектор на площину.
RotateTowards	Повертає поточний вектор в напрямку мети.
Scale	Примножує два вектора по компонентах.

2.3 Кути Ейлера

Кути Ейлера визначають три повороти системи, які дозволяють привести будь-яке положення системи до поточного. Позначимо початкову систему координат як (x, y, z) кінцеву як (X, Y, Z) Перетин координатних площин xy і XY називається лінією вузлів N.

Кут alpha між віссю x і лінією вузлів - кут прецесії.

Кут beta між осями z і Z - кут нутації.

Кут gamma між віссю X і лінією вузлів - кут власного обертання.

Повороти системи на ці кути називаються прецесія, нутація і поворот на власний кут (обертання). Такі повороти некомутативними і кінцеве положення системи залежить від порядку, в якому відбуваються повороти. У разі кутів Ейлера проводиться спочатку поворот на кут alpha навколо осі z, потім поворот на кут beta навколо осі N, і останнім поворот на кут gamma навколо осі Z [7] [8] [9].

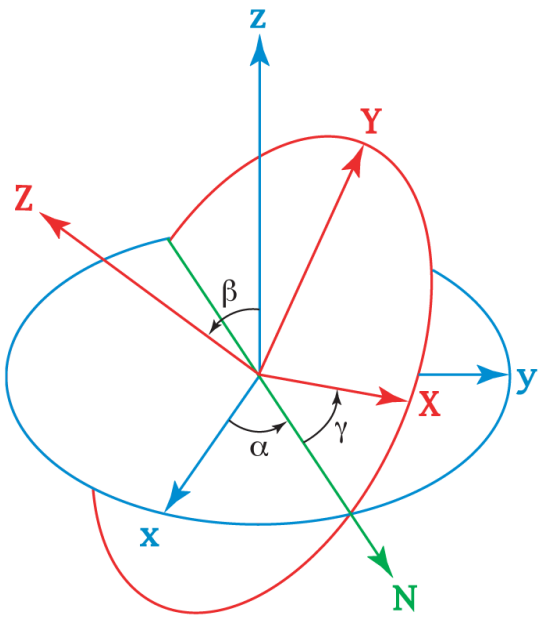


Рисунок 2.15. Кути Ейлера [21]

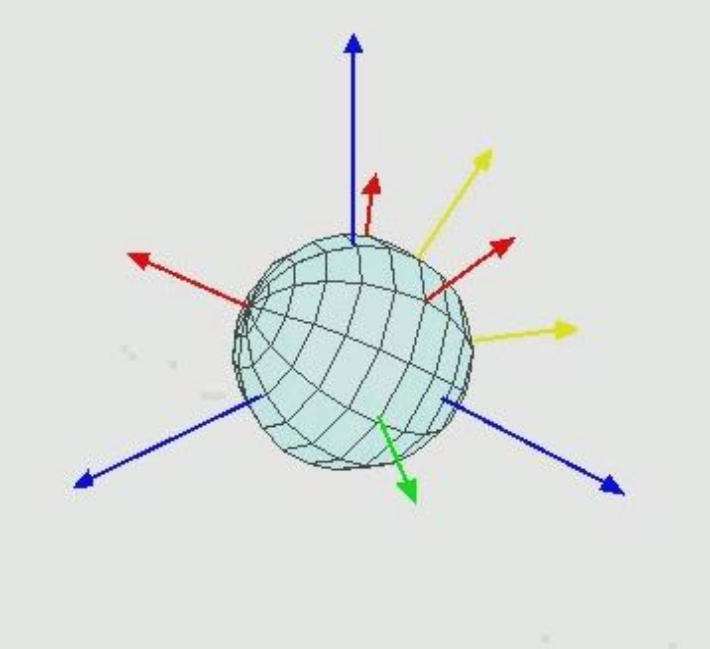


Рисунок 2.16. Сферичні повороти по кутах Ейлера [22]

2.4 Кватерніон

Кватерніони - це система Гіперкомплексні чисел, що утворює векторний простір розмірністю чотири над полем дійсних чисел.

Кватерніони зручні для опису ізометрії тривимірного і чотиривимірного евклідового простора.

q = [x, y, z, w] = [v, w]

Зберігання обертання за допомогою кватерніона. Практично так само як і в "Axis Angle" поданні, перші три компонента представляють вектор, що лежить на осі обертання, причому довжина вектора залежить від кута повороту. Четвертий компонент залежить тільки від величини кута повороту. Залежність досить проста - якщо взяти одиничний вектор V за вісь обертання і кут alpha за обертання навколо цієї осі, тоді кватерніон представляє це обертання, можна записати як:

q = [V*sin(alpha/2), cos(alpha/2)]

Обертання в площині можна задати матрицею 2x2, в якій будуть записані косинуси і синуси кута повороту. Можна уявити, що кватерниіон зберігає комбінацію осі обертання і матриці половини повороту навколо цієї осі.

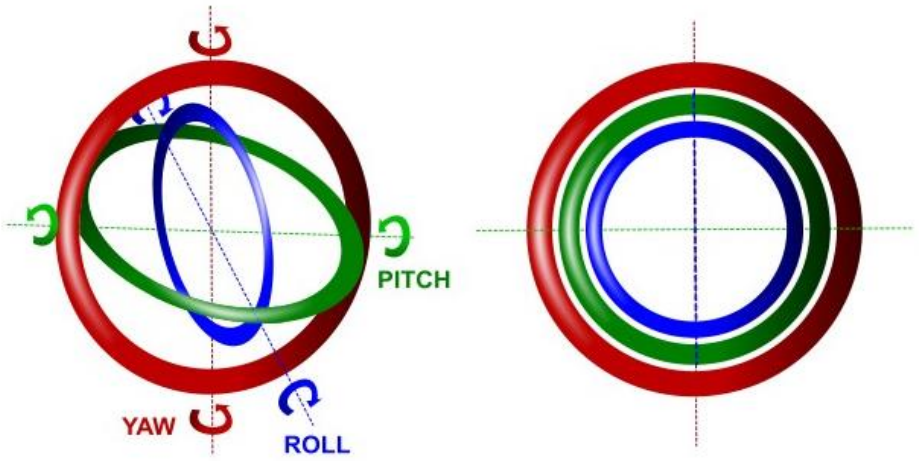


Рисунок 2.17. Обертання по кватерніонами [23]

2.4.1 Основні операції над кватерніонами

Додавання і віднімання кватерніонов

q + q' = [x + x', y + y', z + z', w + w']

q – q' = [x - x', y - y', z - z', w - w']

qs = [xs, ys, zs, ws]

Операції додавання можна описати як "суміш" обертань, тобто ми отримаємо обертання, яке знаходиться між q і q '. У розділі присвяченій інтерполяції орієнтації ми розглянемо операцію складання більш докладно. Множення на скаляр на обертанні не відбивається. Кватерніон, помножений на скаляр, представляє те ж саме обертання, крім випадку множення на 0. При множенні на 0 ми отримаємо "невизначений" обертання.

Множення двох кватернионів

qq' = [vv' + wv' + w'v, ww' – v•v']

де vv '- векторний добуток, v • v' - скалярний добуток векторів.

Дана операція аналогічна множенню двох матриць повороту. Підсумковий кватерніон являє собою комбінацію обертань - об'єкт повернули на q 'а потім на q (якщо дивитися з глобальної системи координат).

Кватерніони не комутативні по множенню, як і матриця обертання: qq 'не дорівнює q'q.

Норма (norm)

N(q) = xx + yy + zz + ww

Модуль (magnitude), довжина кватерніона:

|q| = sqrt(N(q)) = sqrt(xx + yy + zz + ww)

Сполучення (conjugate)

conjugate(q) = [-v, w]

Задає обертання зворотне даному. Зворотне обертання можна також отримати, змінивши знак скалярного "w" компонента на зворотний.

Інверсний (inverse) кватерніон.

Існує такий кватерніон, при множенні на який твір дає нульове обертання і відповідне тотожному кватерніонами (identity quaternion), і визначається як:

q–1 = conjugate(q)/N(q)

Тотожний кватерніон записується як q [0, 0, 0, 1]. Він описує нульовий поворот (за аналогією з одиничною матрицею), і не змінює інший кватерніон при множенні.

Скалярний добуток (inner product).

q•q' = x*x' + y*y' + z*z' + w*w'

Скалярне добуток дає косинус половини кута між двома кватерніонами помножену на їх довжину. Відповідно скалярне добуток двох одиничних кватерніонів дасть косинус половини кута між двома орієнтаціями. Кут між кватерніонами це кут повороту з q в q '(по найкоротшій дузі) [10][11][12].

Обертання 3d вектора.

Обертання 3d вектора v кватерніонами q визначається як:

V' = qvq–1

причому вектор конвертується в кватерніон як:

q = [x, y, z, 0]

і кватерніон назад в вектор як:

v = [x, y, z]

2.4.2 Quaternion клас в Unity3D

Кватерніони використовуються для подання обертань об'єктів.

Даний клас багатофункціональний, і не має проблеми такої як карданний замок. Unity автоматично використовує Quaternions для подання всіх поворотів об'єктів.

Даний клас містить комплексні числа. Даний клас є статичним і не має властивості змінювати компоненти всередині класу Quaternion (x, y, z, w); основною властивістю класу є обертання об'єктів (наприклад, з Transform) і створення нових поворотів об'єкта (наприклад, для плавної інтерполяції між двома обертаннями). Основні функції Quaternion бути: Quaternion.LookRotation, Quaternion.Angle, Quaternion.Euler, Quaternion.Slerp, Quaternion.FromToRotation і Quaternion.identity [13].

Статичні властивості

Identity	Обертання ідентичності
eulerAngles	Повертає кутовий подання кута повороту.
this[int]	Отримати доступ до компонентів x, y, z, w, використовуючи [0], [1], [2], [3].
W	W отримання компонента кватерніона
Y	Y отримання компонента кватерніона
Z	Z отримання компонента кватерніона
X	X отримання компонента кватерніона

Конструктор

Quaternion	Створює новий кватерніон з заданими компонентами x, y, z, w.

Загальні методи

Set	Встановіть x, y, z і w компоненти в існуючий кватерніон.
SetFromToRotation	Створює обертання за певним напрямом.
SetLookRotation	Створює обертання із зазначеними напрямками вперед і вгору.
ToAngleAxis	Перетворює поворот в кутовий подання (кути в градусах).
ToString	Повертає відформатований рядок з

	Quaternion.

Статичні методи

Angle	Повертає кут в градусах між двома поворотами a і b.
AngleAxis	Створює поворот, який обертає кути навколо осі.
Dot	Точковий продукт між двома обертаннями.
Euler	Повертає поворот, який обертає z градусів навколо осі z, x градусів навколо осі x і y градусів навколо осі y.
FromToRotation	Створює поворот за певним напрямом.
Inverse	Повертає інверсію обертання.
Lerp	Інтерполює між a і b на t і потім нормалізує результат. Параметр t затискається до діапазону [0, 1].
LerpUnclamped	Інтерполює між a і b на t і потім нормалізує результат. Параметр t не стискається.
LookRotation	Створює обертання із зазначеними напрямками вперед і вгору.
RotateTowards	Повертає об'єкт у напрямку.
Slerp	Сферично інтерполяцію між a і b на t. Параметр t затискається до діапазону [0, 1].
SlerpUnclamped	Сферично інтерполяцію між a і b на t. Параметр t не стискається.

2.5 Порівняльна характеристика 3D платформ для розробки ігрових додатків

Для створення комп'ютерної графіки використовують множину різних додатків. Є множина 3D платформ для розробки ігрових додатків, але основними платформами являються (Unity 5, UnrealEngine 3).

Порівняння платформи Unity 5 з UnrealEngine 3:

Unity3D

Переваги Unity:

– IDE: поєднання редактора сцен (в комплексі загального редактора) з редактором ігрових об'єктів і редакторів скриптів. Додатково додаються генератори дерев і террейнов.

– Покращені можливості скриптинга, в Unity доступні три мови: JavaScript, C # і різновид Python's Boo.

– Кросплатформеність - підтримують Windows, MacOS, Wii, iPhone, iPod, iPad, Android, PS3 і XBox 360.

– Ряд інструментів для створення якісної графіки. Різні режими освітлення, шейдери, ефекти і інші технології забезпечать гідне візуальне оформлення.

– Unity володіє такими можливостями, як deferred освітлення, стандартний набір пост процесингових ефектів, SSAO, прискорена опрацювання лайтмапов.

– Масштабованість і продуктивність. Більшу частину простих процесів движок обробляє на чудовому рівні.

Недоліки Unity:

– Закритість коду. Неможливість отримання вихідних кодів движка навіть за ліцензією.

– Неможливість доповнення фізики движка сторонніми можливостями. Ви не зможете додати в движок сторонню фізику, або SpeedTree.

– Користувачі, які не володіють англійською мовою, можуть зіткнутися з проблемами при використанні Unity. Ігровий движок російською в даний час поки ще недоступний. Русифікатори до даної платформі також відсутні.

Системні вимоги:

– OS: Windows 7 SP1 +, 8, 10, тільки 64-розрядні версії; Mac OS X 10.9+.

– Серверні версії Windows & OS X б не протестували.

– Центральний процесор: підтримка набору інструкцій SSE2.

– Графічний процесор: відеокарта з підтримкою DX10 (версія шейдеров 4.0).

– Решта залежить, головним чином, від складності ваших проектів.

– Додаткові вимоги до платформи розробки:

– iOS: Mac з операційною системою не нижче OS X 10.9.4 і Xcode 7.0 або вище.

– Android: Android SDK і Java Development Kit (JDK); IL2CPP-скриптинг для бекенда вимагає Android NDK.

– Universal Windows Platform: Windows 10 (64-bit), Visual Studio 2015 with C ++ Tools component or later and Windows 10 SDK

Unreal Engine 3

Переваги Unreal Engine 3:

– Unreal Engine 3 для роботи має інтегровану середу редагування і управління Unreal Editor. Всі ключові інструменти доступні через UnrealEd. Unreal Content Browser робить зручною і легкою роботу з мешамі, матеріалами, звуками і анімацією.

– Рендер-система UDK Gemini дає 64-bit HDR рендеринг.

– AnimSet Viewer - інструмент для роботи з анімацією і створення мешів.

– UnrealScript - це проста мова програмування високого рівня, який дає повний контроль скриптів. Скрипти можуть зв'язуватися за допомогою візуальної системи Unreal Kismet, що не вимагає ніяких програмних знань. UnrealScript дозволяє в гнучкому режимі будувати геймплейні елементи, підключаючи або відключаючи якісь елементи і особливості.

– Використовується фізичний движок від NVIDIA's PhysX, який дає неймовірний над персонажем, рідкими і м'якими тілами. Для настройки і редагування фізики використовується Unreal PhAT.

– Освітлення з системою затінення, технологія Unreal Lightmass, глобальне освітлення для високоякісного статичного світла і ефектів. UDK підтримує сучасні технології освітлення і рендеринга, контролює матеріалами. Unreal Swarm працює в тандемі з Unreal Lightmass для складних обчислень.

– У UDK є всі інструменти для управління системою кінематики, які дозволяють працювати з усіма об'єктами ігрового світу. Unreal Matinee забезпечує контроль над усіма об'єктами і камерою, попередній перегляд в реальному часі.

– Пропонується можливість роботи з динамічної, що деформується картою висот ландшафту. Вбудована організація мережі UDK пропонує LAN і пряме IP з'єднання. Що дозволяє створювати мережеві онлайн ігри.

– Є підтримка формату FBX від Autodesk для імпорту мешів і анімації; інструмент Attachment Editor для побудови і відстеження взаємозв'язків між об'єктами; опція підвищення чіткості генеруються MIP-рівнів для текстури без додаткових витрат пам'яті або швидкодії.

– Якісна звукова система, що підтримує 3D звук. UDK підтримує всі останні системи компресії звуку, включаючи Ogg Vorbis. Вам дається повний контроль над подачею, рівнем, відтворенням, зацикленням, фільтрацією, модуляцією і рандомізація. Для відео використовується Bink Video Codec, який дозволяє управляти відеороликами на ваш розсуд.

– З UDK можлива широка робота зі штучним інтелектом, поміщеним в простір і геймплей гри. AI-система може дати життя натовпам персонажів, які будуть діяти по змодельованих умов.

Недоліки Unreal Engine 3:

– Складність у вивченні, оскільки має множину функцій.

– Велике навантаження на систему, вимагаєте високоякісне обладнання.

– Застаріла документація або відсутність її для багатьох модулів.

– Проблеми в написанні коду оскільки використовує власну мову програмування

Системні вимоги:

– Процесор чотирьохядерний процесор Intel або AMD, 2.5 ГГц.

– Пам'ять 8 ГБ оперативної пам'яті.

– Відеокарта NVIDIA GeForce 470 GTX або AMD Radeon 6870 HD серії або вище.

– ОС Windows 7/8 64bit.

– Версія DirectX - DirectX End-User Runtimes (June 2010).

– Версія .NET - .NET 4.0 (встановлюється за допомогою Windows Update).

– Visual Studio version - Visual Studio 2013 Professional, або Express для Windows Desktop.

Після проведення порівняльного аналізу, була обрана платформа Unity3D. По скільки дана платформа легка у вивченні, і має мінімальні вимоглива до комп'ютера.

2.6 Побудова віртуальної моделі в 2 - х і 3 - х вимірному просторі

Потрібно розробити схему віртуальної моделі з використанням 3D векторної математики.

На векторному графіку зображені 9 об'єктів ob0, ob1, ob2 ... ob9. Все об'єкт спочатку встановлюватися по вектору (Y). У свою чергу нульовий об'єкт (ob0) виступає батьком наступних об'єктів, задає їм позицію розташування по вектору, обсяг і радіус об'єктів, швидкість рух об'єктів по орбіті і навколо самого себе, швидкість і період обертання об'єктів навколо власної осі.

Так само батьками являються (ob3, ob4, ob5, ob6, ob8, ob9).

Таблиця 2.1. Побудова об'єктів віртуальної моделі.

Батьки	Дочірній об'єкт
Ob3	Ob3.1
Ob4	Ob4.1 and Ob4.2

Ob5	Ob5.1 and Ob5.2
Ob6	Ob6.1 and Ob6.2
Ob8	Ob8.1 and Ob8.2
Ob9	Ob9.1

Дані об'єкти виступають супутниками, і мають тільки параметри швидкості обертання навколо своїх батьків, і позицій розташування по вектору.

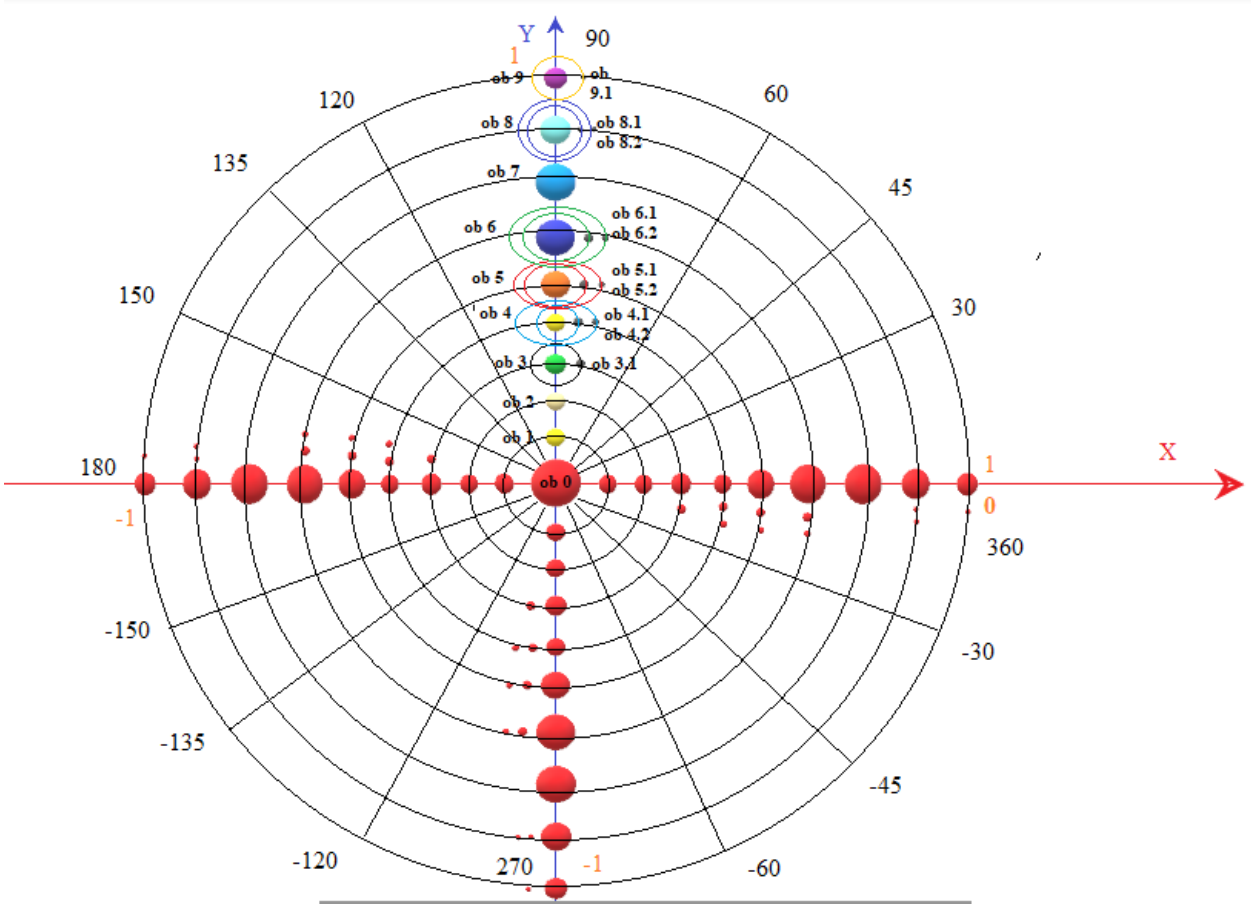


Рисунок 2.18. Схема віртуальної модель в 2 - х вимірному просторі [Автор Шаповалов Олександр]

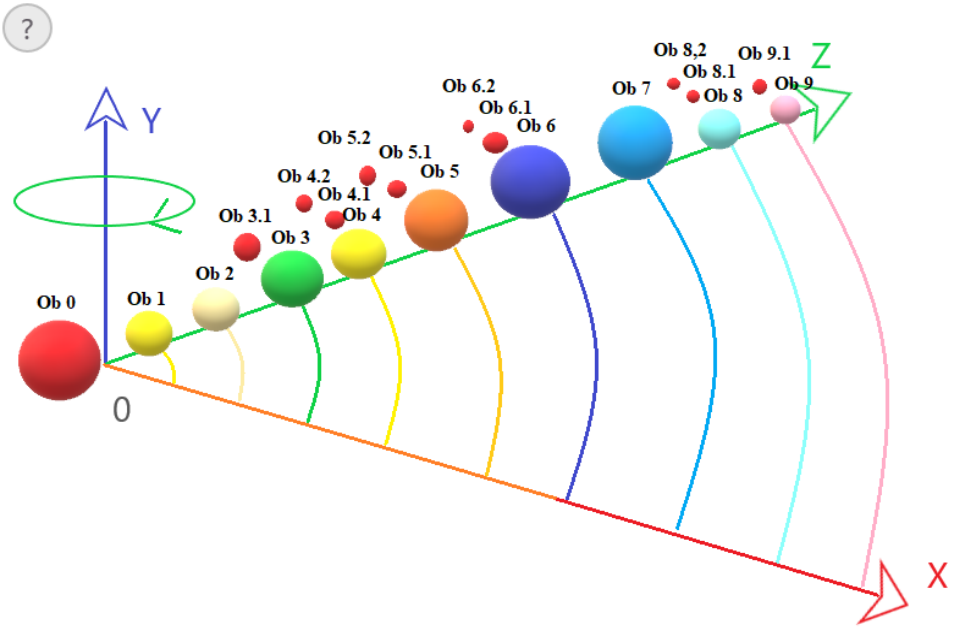


Рисунок 2.19. Схема віртуальної модель в 3 - х вимірному просторі [Автор Шаповалов Олександр]

2.7 Розрахунки для об'єктів віртуальної моделі

Визначення відстані від нульового об'єкта

Для визначення відстані між об'єктів потрібні провести розрахунки. Знаходження відстані між об'єктів буде здійснюватися за формулою:

$$R = \sqrt[3]{T^2 * (1 + M)} = \sqrt[2]{T^2}$$

Середня відстань R об'єкта від нульового об'єкта (в частках а. Е) знаходять по періоду обертання Т.

а. е = 149597870 ± 2 км.

Т - є повним періодом обертання об'єкта навколо нульового об'єкта.

М - є масою об'єкта в порівнянні з масою нульового об'єкта.

Визначення швидкості об'єктів

Швидкість, яку треба повідомити об'єкту, щоб воно рухалося по круговій орбіті під дією гравітаційної сили, називається першою космічною швидкістю.

Формули для розрахунку першої космічної швидкості:

$$a_{ц} = \frac{v_I^2}{r} = g \qquad v_I^2 = gr \qquad v_I = \sqrt{gr} \qquad v_I = \sqrt{gR}$$

$$g = G\frac{M}{R^2} \qquad v_I = \sqrt{G\frac{M}{R^2} * R} = \sqrt{G\frac{M}{R}} \qquad v_I = \sqrt{G\frac{M}{R}}$$

$$g = G\frac{M}{(R+h)^2} \qquad v_I = \sqrt{G\frac{M}{(R+h)^2} * (R+h)} = \sqrt{G\frac{M}{R+h}} \qquad v_I = \sqrt{G\frac{M}{R+h}}$$

Де G - гравітаційна стала,

M - маса об'єкта,

R - радіус об'єкта,

H - Висота над поверхнею,

V - модуль швидкості,

r- радіус кола,

g- прискорення вільного падіння.

Орбітальний період об'єктів

Орбітальний період є проміжком часу, протягом якого будь-який об'єкт-супутник здійснює навколо головного тіла повний оборот щодо нульового об'єкта.

Звернення малих об'єктів по еліптичній або круговій орбіті навколо більшого центрального нульового об'єкта розраховується за формулою[15]:

$$\left(\frac{P}{2\pi}\right)^2 = \frac{a^3}{G \cdot (M+m)}$$

Де P - орбітальний період,

a - велика піввісь,

M, м - маси нульового об'єкта та інших об'єктів.

$$P = 1\,год \cdot \sqrt{\frac{(a/1\,a.e.)^3}{M/M_{сон}}}$$

Де P - орбітальний період,

a - велика піввісь,

M - маси нульового об'єкта.

Орбітальний період двох об'єктів (час, через яке повторюється їх взаємне розташування).

$$1/S = |1/P_1 - 1/P_2|$$

Де P1, P2 - сидеричні періоди.

2.8 Реалізація віртуальної моделі на мові C#

Для реалізацій скрипта руху об'єктів, потрібно скласти таблицю з даними які були отримані вище.

Таблиця 2.2. Дані об'єктів віртуальної моделі

№ Об'єкта	Орбітальний період	Середня відстань від нульового об'єкта (мл.км).	Середній екваторіальний радіус об'єкта. (Км)	Відстань від батьків до супутників (км)	Нахил осей обертання об'єктів
Ob 0	25,38	0	695 500	0	0
Ob 1	87.969	57	2439	0	- 0.1
Ob 2	224,7	108	6051	0	177.36
Ob 3	365,26	149	6378	0	23.19
Ob 3.1	0	0	1723	384.467	1.5424
Ob 4	686.98	227	3397	0	25.19
Ob 4.1	0	0	1100	9380	1.093
Ob 4.2	1,26244	0	600	23.458	0.93
Ob 5	4332.589	778	71492	0	3.13
Ob 5.1	0	0	1821	422.000	0
Ob 5.2	0	0	2631	800.000	0
Ob 6	10759.22	1433	60268	0	26.73
Ob 6.1	0	0	531	295 000	1.12
Ob 6.2	0	0	1123	377.400	0
Ob 7	30685.4	2870	25559	0	97.77
Ob 8	60190.03	4491	24764	0	28.32
Ob 8.1	0	0	168	73.000	0

Ob 8.2	0	0	100	118.00	0
Ob 9	90553.02	5868	1151	0	119.61
Ob 9.1	0	0	100	19 640	112.78

Реалізація віртуальної моделі руху об'єктів на мові C#

Програмний код закріплюється в інспектора нульового об'єкта.

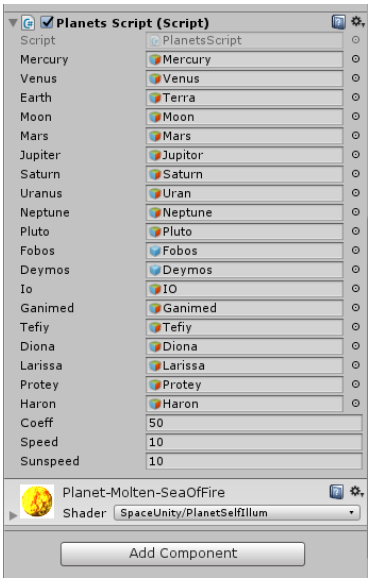


Рисунок 1.20. Інспектор нульового об'єкта [Автор Шаповалов Олександр]

Підключити бібліотеки для роботи з середовищем Unity3d.

```
using UnityEngine;
using System.Collections;
public class PlanetsScript : MonoBehaviour {
```

Створюємо змінні типу (public) які можуть працювати з об'єктами і з іншим скрипт даного проекту.

GameObject (name - є зміною) - створює осередки в інспектора об'єкта, на якому закріплений даний скрипт. В осередку поміщаються об'єкти для подальший експлуатація та роботи з цими об'єктами через даний скрипт.

```
public GameObject mercury;
public GameObject venus;
public GameObject earth;
public GameObject moon;
public GameObject mars;
public GameObject jupiter;
public GameObject saturn;
```

```
public GameObject uranus;
public GameObject neptune;
public GameObject pluto;
public GameObject Fobos;
public GameObject Deymos;
public GameObject Io;
public GameObject Ganimed;
public GameObject Tefiy;
public GameObject Diona;
public GameObject Larissa;
public GameObject Protey;
public GameObject Haron;
```

Створюємо змінні типу (private) цей тип змінних використовується тільки для обчислень в даному скрипті.

Період обертання об'єкта 1 навколо нульового об'єкта.

```
private float baseMercuryPeriod = 87.969f;
```

Змінні звернення кожної з планет навколо нульового об'єкта в відносних одиницях (по відношенню до періоду обертання об'єкта 1 навколо нульового об'єкта)

```
private float periodmars;
private float periodjupiter;
private float periodsaturn;
private float perioduranus;
private float periodneptune;
private float periodpluto;
private float periodMercury;
private float periodVenus;
private float periodEarth;
```

Період обертання об'єктів навколо власних осей.

```
public float coeff = 360.0f;
```

Швидкість рух об'єктів.

```
public float speed = 5;
```

Швидкість обертання нульового об'єкта навколо власної осі.

```
public float Sunspeed = 10;
```

void Start ()

Встановлюємо орбітальні періоди для об'єктів зі зверненням до періоду об'єкта 1.

```
{       period = 25.38f/baseMercuryPeriod;
        periodMercury = 87.969f/baseMercuryPeriod;
        periodVenus = 224.7f/baseMercuryPeriod;
        periodEarth = 365.26f/baseMercuryPeriod;
        periodmars = 686.98f/baseMercuryPeriod;
        periodjupiter = 4332.589f/baseMercuryPeriod;
        periodsaturn = 10759.22f/baseMercuryPeriod;
        perioduranus = 30685.4f/baseMercuryPeriod;
        periodneptune = 60190.03f/baseMercuryPeriod;
        periodpluto = 90553.02f/baseMercuryPeriod;
```

Встановлюємо позицій для об'єктів по вектору (x) використовуючи клас Vector3. Що б легше було будувати модель, значення об'єктів потрібно розділити на 1 мл. Але для точність можна поставити початкові значення.

```
transform.position = new Vector3 (0.0f,1.0f,0.0f);
mercury.transform.position = new Vector3 (57.9f,1.0f,0.0f);
venus.transform.position = new Vector3 (108.0f,1.0f,0.0f);
earth.transform.position = new Vector3 (149.0f,1.0f,0.0f);
moon.transform.position = new Vector3 (145.0f,1.0f,0.0f);
mars.transform.position = new Vector3 (227.0f,1.0f,0.0f);
Fobos.transform.position = new Vector3(226.0f, 1.0f, 0.0f);
Deymos.transform.position = new Vector3(225.0f, 1.0f, 0.0f);
jupiter.transform.position = new Vector3 (778.0f,1.0f,0.0f);
Io.transform.position = new Vector3(774.0f, 1.0f, 0.0f);
Ganimed.transform.position = new Vector3(770.0f, 1.0f, 0.0f);
saturn.transform.position = new Vector3 (1433.0f,1.0f,00.0f);
Diona.transform.position = new Vector3(1431.0f, 1.0f, 0.0f);
Tefiy.transform.position = new Vector3(1430.0f, 1.0f, 0.0f);
uranus.transform.position = new Vector3 (2870.0f,1.0f,0.0f);
neptune.transform.position = new Vector3(4491.0f, 1.0f, 0.0f);
Larissa.transform.position = new Vector3(4490.0f, 1.0f, 0.0f);
Protey.transform.position = new Vector3(4489.0f, 1.0f, 0.0f);
pluto.transform.position = new Vector3 (5868.0f,1.0f,0.0f);
Haron.transform.position = new Vector3(5867.0f, 1.0f, 0.0f);
```

Використовуючи клас Quaternion встановлюємо нахил осей обертання об'єктів.

```
mercury.transform.rotation = Quaternion.Euler(-1, 0, 0);
venus.transform.rotation = Quaternion.Euler(177, 0, 0);
earth.transform.rotation = Quaternion.Euler(23, 0, 0);
moon.transform.rotation = Quaternion.Euler(1, 0, 0);
mars.transform.rotation = Quaternion.Euler(25, 0, 0);
Fobos.transform.rotation = Quaternion.Euler(1, 0, 0);
Deymos.transform.rotation = Quaternion.Euler(1, 0, 0);
```

```
jupiter.transform.rotation = Quaternion.Euler(3, 0, 0);
saturn.transform.rotation = Quaternion.Euler(27, 0, 0);
Diona.transform.rotation = Quaternion.Euler(1, 0, 0);
uranus.transform.rotation = Quaternion.Euler(97, 0, 0);
neptune.transform.rotation = Quaternion.Euler(28, 0, 0);
pluto.transform.rotation = Quaternion.Euler(119, 0, 0);
Haron.transform.rotation = Quaternion.Euler(112, 0, 0);
```

Задаємо скаляр для об'єктів віртуальної моделі, тобто встановлюється розмір і обсяг об'єкта по 3 векторах.

```
transform.localScale = new Vector3(695000.500f, 695000.500f, 695000.500f);
    Haron.transform.localScale = new Vector3(0.100f, 0.100f, 0.100f);
    pluto.transform.localScale = new Vector3(1.151f, 1.151f, 1.151f);
    Protey.transform.localScale = new Vector3(0.100f, 0.100f, 0.100f);
    Larissa.transform.localScale = new Vector3(0.168f, 0.168f, 0.168f);
    neptune.transform.localScale = new Vector3(247.640f, 247.640f, 247.640f);
    uranus.transform.localScale = new Vector3(255.590f, 255.590f, 255.590f);
    Diona.transform.localScale = new Vector3(1.123f, 1.123f, 1.123f);
    Tefiy.transform.localScale = new Vector3(0.531f, 0.531f, 0.531f);
    saturn.transform.localScale = new Vector3(602.680f, 602.680f, 602.680f);
    Io.transform.localScale = new Vector3(1.821f, 1.821f, 1.821f);
    Ganimed.transform.localScale = new Vector3(2.631f, 2.631f, 2.631f);
    jupiter.transform.localScale = new Vector3(714.920f, 714.920f, 714.920f);
    Deymos.transform.localScale = new Vector3(0.600f, 0.600f, 0.600f);
    Fobos.transform.localScale = new Vector3(1.100f, 1.100f, 1.100f);
    mars.transform.localScale = new Vector3(3.397f, 3.397f, 3.397f);
    moon.transform.localScale = new Vector3(1.723f, 1.723f, 1.723f);
    eartch.transform.localScale = new Vector3(6.378f, 6.378f, 6.378f);
    venus.transform.localScale = new Vector3(6.051f, 6.051f, 6.051f);
    mercury.transform.localScale = new Vector3(2.439f, 2.439f, 2.439f);
      }

void Update () {


Об'єкт (Ob 0)

transform.RotateAround(Vector3.zero, Vector3.up, Sunspeed * Time.deltaTime);
Об'єкт (Ob1)
mercury.transform.RotateAround(Vector3.zero, Vector3.up, speed / periodMercury
*Time.deltaTime);
mercury.transform.Rotate(Vector3.up * coeff / periodMercury * Time.deltaTime);
Об'єкт (Ob2)
venus.transform.RotateAround(Vector3.zero, Vector3.up, speed / periodVenus *
Time.deltaTime);
venus.transform.Rotate(Vector3.up * coeff * Time.deltaTime);
Об'єкт (Ob3)
earth.transform.RotateAround(Vector3.zero, Vector3.up, speed / periodEarth *
Time.deltaTime);
earth.transform.Rotate(Vector3.up * coeff* Time.deltaTime);
Об'єкт (Ob3.1)
```

```
        moon.transform.RotateAround(earth.transform.position, Vector3.up, speed / periodEarth
* Time.deltaTime);
        Об'єкт (Ob4)
        mars.transform.RotateAround(Vector3.zero, Vector3.up, speed / periodmars *
Time.deltaTime);
        mars.transform.Rotate (Vector3.up * coeff * Time.deltaTime);
        Об'єкт (Ob4.1)
        Fobos.transform.RotateAround(mars.transform.position, Vector3.up, speed / periodmars
* Time.deltaTime);
        Об'єкт (Ob4.2)
        Deymos.transform.RotateAround(mars.transform.position, Vector3.up, speed /
periodmars * Time.deltaTime);
        Об'єкт (Ob5)
        jupiter.transform.RotateAround(Vector3.zero, Vector3.up, speed / periodjupiter *
Time.deltaTime);
        jupiter.transform.Rotate (Vector3.up * coeff * Time.deltaTime);
        Об'єкт (Ob5.1)
          Io.transform.RotateAround(jupiter.transform.position, Vector3.up, speed / periodjupiter
* Time.deltaTime);
        Об'єкт (Ob5.2)
        Ganimed.transform.RotateAround(jupiter.transform.position, Vector3.up, speed /
periodjupiter * Time.deltaTime);
        Об'єкт (Ob6)
        saturn.transform.RotateAround(Vector3.zero, Vector3.up, speed / periodsaturn *
Time.deltaTime);
            saturn.transform.Rotate (Vector3.up * coeff * Time.deltaTime);
        Об'єкт (Ob6.1)
        Diona.transform.RotateAround(saturn.transform.position, Vector3.up, speed /
periodsaturn * Time.deltaTime);
        Об'єкт (Ob6,2)
        Tefiy.transform.RotateAround(saturn.transform.position, Vector3.up, speed /
periodsaturn * Time.deltaTime);
        Об'єкт (Ob7)
        uranus.transform.RotateAround(Vector3.zero, Vector3.up, speed / perioduranus *
Time.deltaTime);
                uranus.transform.Rotate (Vector3.up * coeff * Time.deltaTime);
        Об'єкт (Ob8)
        neptune.transform.RotateAround(Vector3.zero, Vector3.up, speed / periodneptune *
Time.deltaTime);
        neptune.transform.Rotate (Vector3.up * coeff * Time.deltaTime);
        Об'єкт (Ob8.1)
        Protey.transform.RotateAround(neptune.transform.position, Vector3.up, speed /
periodneptune * Time.deltaTime);
        Об'єкт (Ob8.2)
        Larissa.transform.RotateAround(neptune.transform.position, Vector3.up, speed /
periodneptune * Time.deltaTime);
        Об'єкт (Ob9)
        pluto.transform.RotateAround(Vector3.zero, Vector3.up, speed / periodpluto *
Time.deltaTime);
        pluto.transform.Rotate (Vector3.up * coeff * Time.deltaTime);
        Об'єкт (Ob9.1)
```

```
        Haron.transform.RotateAround(pluto.transform.position, Vector3.up, speed / periodpluto
* Time.deltaTime);
                }}
```

2.9 Векторна робота з камерою

Камера - це об'єкт, який визначає вид в просторі сцени. Позиція об'єкта визначає точку огляду, тоді як осі вперед (Z) і вгору (Y) об'єкта визначають напрямок перегляду і верхню частину екрана відповідно. Компонент камери також визначає розмір і форму області, яка потрапляє в подання. Під час налаштування цих параметрів камера може відображати те, що вона в даний час «бачить» на екрані. Коли об'єкт камери переміщується і обертається, відображається картинка також переміщається і повертається відповідно.

Unity3D має перспективні і орфографічні режими перегляду сцени відомі як проекції камери.

Форма розглянутої області

Як перспективні, так і орфографічні камери мають обмеження на те, як далеко вони можуть «бачити» від свого поточного положення. Межа визначається площиною, перпендикулярної напрямку вперед (Z) камери. Це відомо як площину відсікання, так як об'єкти на більшій відстані від камери «обрізані». Існує також відповідна площину відсікання поблизу камери - видимий діапазон відстані - це відстань між двома площинами.

Без перспективи об'єкти виглядають однаково незалежно від їх відстані. Це означає, що обсяг перегляду орфографічною камери визначається прямокутної коробкою, що проходить між двома площинками відсікання.

Коли використовується перспектива, об'єкти зменшуються в міру збільшення відстані від камери. Це означає, що ширина і висота видимої частини сцени ростуть зі збільшенням відстані. Таким чином, обсяг перегляду перспективною камери не є коробкою, а пірамідальної формою з вершиною в положенні камери і базою на дальній площині відсікання. Однак форма не є пірамідою, оскільки верхня частина обрізана площиною ближнього відсікання; ця форма усіченої піраміди відома як усічений конус. Так як його висота не постійна, усечка визначається відношенням її ширини до її висоті (відомої як ставлення розмірів) і кутом між верхньою і нижньою частиною вершини (відомої як поле зору FOV) [14].

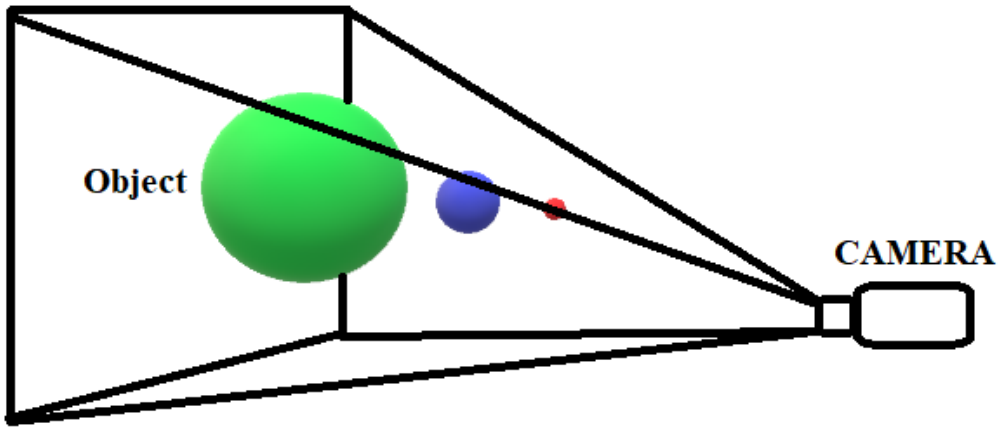


Рисунок 2.21. Робота камера в перспективі [Автор Шаповалов Олександр]

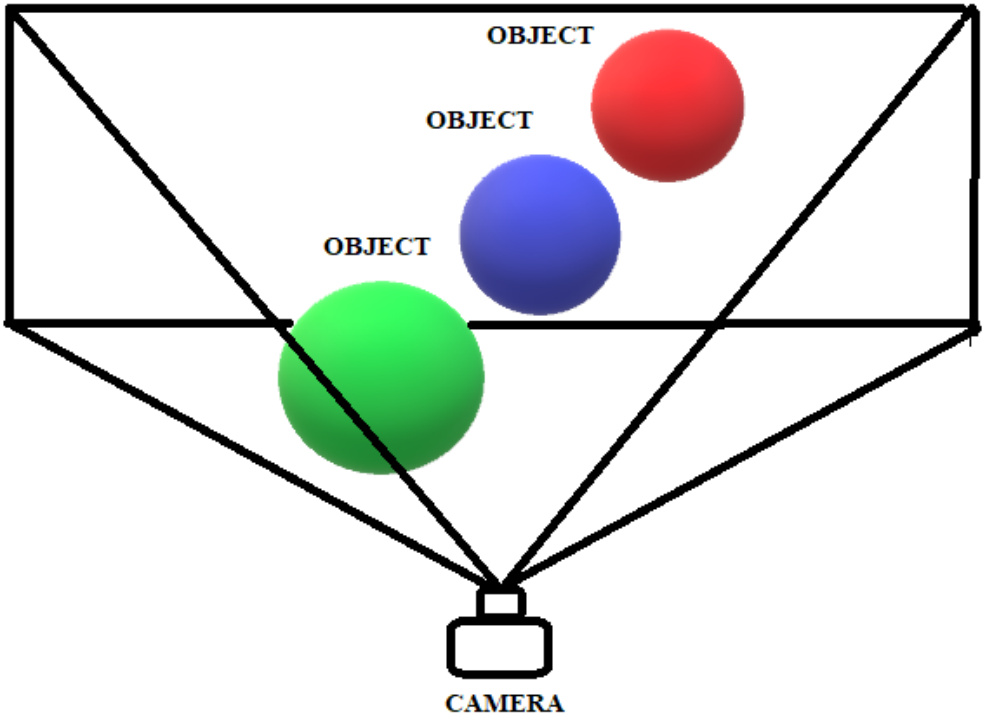


Рисунок 2.22. Робота камера в орфографічні режими [Автор Шаповалов Олександр]

2.9.1 Реалізація позиційної камери

Потрібно розробити камеру для відстеження об'єктів у віртуальному просторі. Початковою позицією і прив'язкою камери є нульовий об'єкт. Камера встановлюється на певному відстань від нульового об'єкта з розрахунком захоплення всіх дочірніх об'єктів в полі зору. Встановлюється функція віддалення від нульового об'єкта і наближення.

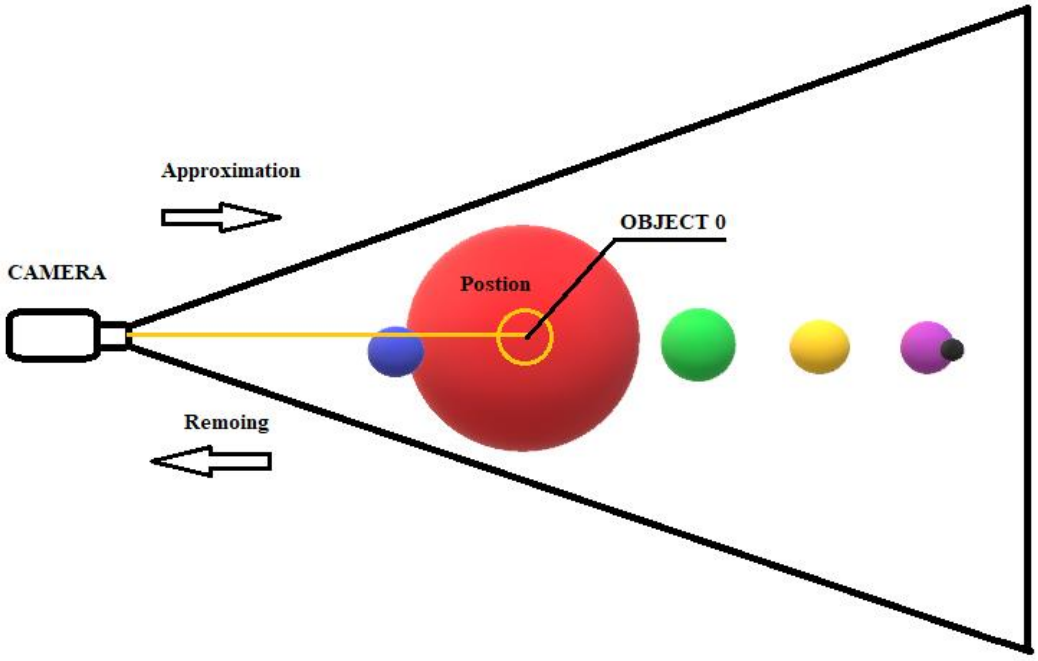


Рисунок 2.23. Прив'язка камери до нульового об'єкту [Автор Шаповалов Олександр]

Камера в початкової позиції прив'язана до нульового об'єкту і робить рухи на 360 градусів навколо об'єкта по вектору (X, Y). Для виконання даних функція потрібно розробити скрипт RCObject на мові C #. Даний скрипт виглядає наступним чином:

Підключити бібліотеки.

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

Створюємо змінні, де змінна: sensitivity є чутливістю мишки і чистоти кадрів при русі камери; limit є обмеженням градусів обертання камери по осі Y; zoom є чутливості наближення камери до об'єкта; zoomMax встановлює межі максимального віддалення від об'єкта; zoomMin встановлює межі мінімального наближення до об'єкта.

```
public class CameraRotateAround : MonoBehaviour {
public Transform target;
public Vector3 offset;
public float sensitivity = 3;
public float limit = 360;
public float zoom = 0.25f;
public float zoomMax = 10;
public float zoomMin = 3;
private float X, Y;
```

Функція яка встановлює камеру в початкову позицію.

```
void Start ()
{limit = Mathf.Abs(limit);
if(limit > 90) limit = 90;
offset = new Vector3(offset.x, offset.y, -Mathf.Abs(zoomMax)/2);
transform.position = target.position + offset;}
```

Функція яка відповідає за віддалення і наближення камери до об'єкта.

```
void Update (){
if(Input.GetAxis("Mouse ScrollWheel") > 0) offset.z += zoom;
else if(Input.GetAxis("Mouse ScrollWheel") < 0) offset.z -= zoom;
offset.z = Mathf.Clamp(offset.z, -Mathf.Abs(zoomMax), -Mathf.Abs(zoomMin));
```

Функція яка відповідає за обертання камери навколо об'єкта.

```
X = transform.localEulerAngles.y + Input.GetAxis("Mouse X") * sensitivity;
Y += Input.GetAxis("Mouse Y") * sensitivity;
Y = Mathf.Clamp (Y, -limit, limit);
transform.localEulerAngles = new Vector3(-Y, X, 0);
transform.position = transform.localRotation * offset + target.position; }}
```

2.9.2 Прив'язка камери до дочірніх об'єктів по точках

Для кожного дочірнього об'єкта в віртуальному просторі встановлюється порожній об'єкт, тобто позиційні точки де фіксуватиметься камера після виклику функції при натисканні на об'єкт. Після виклику функцій камера буде відслідковувати не нульовий об'єкт, а будь-який інший дочірній об'єкт.

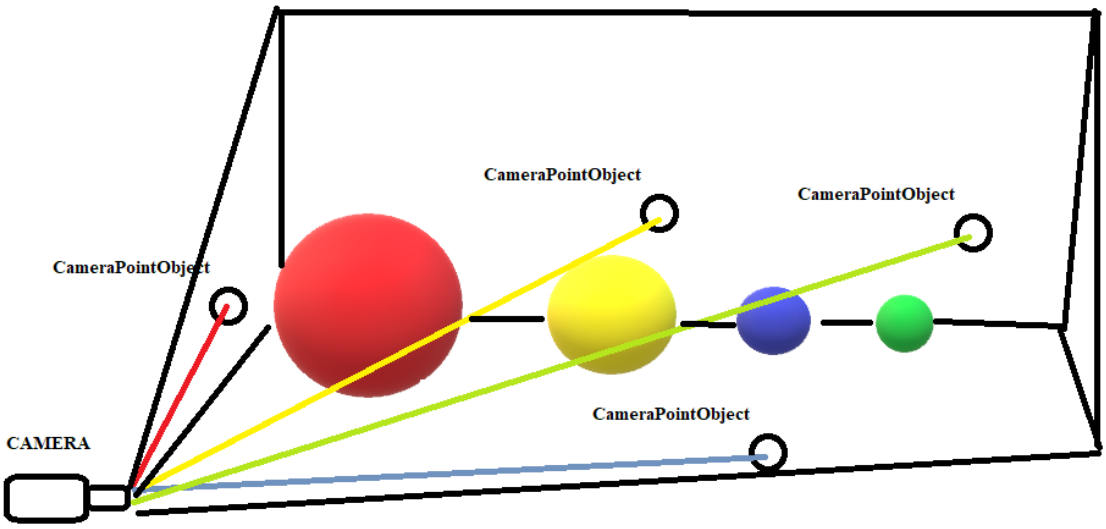


Рисунок 2.24. Переміщення камери до дочірніх об'єктів по точках [Автор Шаповалов Олександр]

Для кожного об'єкта встановлюється аналогічний скрипт RCObject обертання камери навколо об'єкта. Скрипти для дочірніх об'єктів знаходяться в неактивному стані у запобігання конфронтація з основним скриптом, який прив'язаний до нульового об'єкту.

На кожному об'єкті прив'язана кнопка «Panel button» дана кнопка викликає скрипт, який знаходитися в «Inspector» камери. Скрипт встановлюють позицію камери, на точку яка прив'язана до об'єкта. Так само паралельно перемикаються між скриптами обертання камери. Даний скрипт виглядає наступним чином:

Підключити бібліотеки.

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

public class GamimedPoint : MonoBehaviour {

Transform mount;
```

Transform (CP - є зміною) - створює осередки в інспектора об'єкта, на якому закріплений даний скрипт. В осередку поміщаються об'єкти для подальший експлуатація та роботи з цими об'єктами через даний скрипт.

```
public Transform[] CP;
```

Створюємо дві змінні: speedFactor відповідає за швидкість зміни позицій камери; angleFactor кут позиції камери.

```
private float speedFactor = 0.5f;
private float angleFactor = 10f;
bool Conect = true;
```

Функція для зміни позиції.

```
   void Start()

   {
transform.position = Vector3.Lerp(transform.position, mount.position, speedFactor);
transform.rotation = Quaternion.Slerp(transform.rotation, mount.rotation, angleFactor);
   }

   void Update(){    }
```

Функція для паралельного перемикання скриптів. Якщо позиція камери була змінена, то скрипти обертання інших об'єктів відключається. Крім тих об'єктів на який був здійснений перехід.

```
public void SetNewMount(Transform m)
{

   if (Conect)
   {

      Conect = true;

      GameObject object1 = GameObject.Find("Main Camera");
      ScriptStop script1 = object1.GetComponent<ScriptStop>();
      script1.enabled = false;

      GameObject objects = GameObject.Find("Main Camera");
      CameraRotateAround scripts = object1.GetComponent<CameraRotateAround>();
      scripts.enabled = false;

      GameObject objectRCDeymos = GameObject.Find("Main Camera");
      RCDeymos scriptRCDeymos =
objectRCDeymos.GetComponent<RCDeymos>();
      scriptRCDeymos.enabled = false;

      GameObject objectRCGanimed = GameObject.Find("Main Camera");
      RCGanimed scriptRCGanimed =
objectRCGanimed.GetComponent<RCGanimed>();
      scriptRCGanimed.enabled = true;


      GameObject objectRCVenus = GameObject.Find("Main Camera");
      RCVenus scriptRCVenus = objectRCVenus.GetComponent<RCVenus>();
      scriptRCVenus.enabled = false;


      GameObject objectRCFobos = GameObject.Find("Main Camera");
      RCFobos scriptRCFobos = objectRCFobos.GetComponent<RCFobos>();
      scriptRCFobos.enabled = false;

      GameObject objectRCNeptune = GameObject.Find("Main Camera");
      RCNeptune scriptRCNeptune =
objectRCNeptune.GetComponent<RCNeptune>();
      scriptRCNeptune.enabled = false;

      GameObject objectRCPluto = GameObject.Find("Main Camera");
      RCPluto scriptRCPluto = objectRCPluto.GetComponent<RCPluto>();
      scriptRCPluto.enabled = false;

      GameObject objectRCUran = GameObject.Find("Main Camera");
      RCUran scriptRCUran = objectRCUran.GetComponent<RCUran>();
```

```
            scriptRCUran.enabled = false;

            GameObject objectRCSaturn = GameObject.Find("Main Camera");
            RCSaturn scriptRCSaturn = objectRCSaturn.GetComponent<RCSaturn>();
            scriptRCSaturn.enabled = false;

            GameObject objectRCMars = GameObject.Find("Main Camera");
            RCMars scriptRCMars = objectRCMars.GetComponent<RCMars>();
            scriptRCMars.enabled = false;


            GameObject objectRCJupitor = GameObject.Find("Main Camera");
            RCJupitor scriptRCJupitor = objectRCJupitor.GetComponent<RCJupitor>();
            scriptRCJupitor.enabled = false;

            GameObject objectRCMoon = GameObject.Find("Main Camera");
            RCMoon scriptRCMoon = objectRCMoon.GetComponent<RCMoon>();
            scriptRCMoon.enabled = false;

            GameObject objectMer = GameObject.Find("Main Camera");
        CameraRotateAroundMercury scriptMer =
objectMer.GetComponent<CameraRotateAroundMercury>();
            scriptMer.enabled = false;

            GameObject objectRCEarth = GameObject.Find("Main Camera");
            RCEarth scriptRCEarth = objectRCEarth.GetComponent<RCEarth>();
            scriptRCEarth.enabled = false; }}}
```

2.9.3 Додаткова функція відстеження камерою найменування об'єктів

У кожного об'єкта є найменування формату «Text», що б найменування об'єктів завжди було видно для камери в будь-якій позиції в віртуальному просторі, потрібно реалізувати скрипт:

Підключити бібліотеки.

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

public class LoopCamera : MonoBehaviour {

void Start () {}
```

Функція відстеження найменування об'єкта.

```
void Update ()
{
 Camera cam = Camera.main;
 transform.LookAt(transform.position + cam.transform.rotation * Vector3.forward,
```

```
cam.transform.rotation * Vector3.up);}}
```

Даний скрипт закріплюється в інспектора об'єкта формату «Text».

Висновок до другого розділу

У даній роботі був проведений аналіз принципу роботи векторів в 3D графіці, а саме в Unity3D. Були проаналізовані основні операцій над векторами, а також принцип роботи класу Vector3 і вивчені його основні функції.

Була складена таблиця числових даних для об'єктів з використанням астрономічних формул. Для більш точного побудови моделі в віртуальному просторі. Після проведення обчислень була побудована графічна модель розміщення об’єктів у віртуальному просторі, для полегшення установки об'єктів по векторах. Також в цьому розділі був проведений аналіз принципів роботи кватерніонов. Кватерніон дозволяє змінювати нахил або обертання об'єкта в 4 - х вимірному просторі. Були проаналізовані основні операцій над кватерніонами, а також принцип роботи класу Quaternion і вивчені його основні функції.

Після проведення математичних обчислень і побудов графічної моделі, був написаний спеціалізований код для руху об'єктів по векторах. При написані коду все числові дані були реалізовані для кожного об'єкта.

Список використаної літератури в другому розділі

1. unity3d.com - офіційний сайт Unity.

2. Article by David Roshen. Linear algebra for game developers.

3. Lipschutz, M. Lipson. Linear Algebra (Schaum's Outlines). - 4th. - McGraw Hill 2009.

4. Crowe M. J. A History of Vector Analysis - The Evolution of the Idea of a Vectorial System. - Courier Dover Publications, 1994.

5. Documentation of vector arithmetic in Unity3D - https://docs.unity3d.com/ru/530/Manual/UnderstandingVectorArithmetic.html.

6. Documentation of class vector3 in Unity3D - https://docs.unity3d.com/ScriptReference/Vector3.html.

7. Березкін Е.Н. Курс теоретичної механіки - 2-е изд., Пер. - М .: Изд-во МГУ. 1974.

8. Журавльов В. Ф. Основи теоретичної механіки - 2-е вид. - М .: Физматлит, 2001..

9. Уїттекер Е. Аналітична динаміка.

10. "Quaternions", Ken Shoemake.

11. Euler Angle Conversion ", Ken Shoemake, in" Graphics Gems IV ".

12. "Hacking Quaternions", The Inner Product (March 2002) Jonathan Blow, Game Developer Magazine.

13. Documentation of class Quaternion in Unity3D -https: //docs.unity3d.com/ScriptReference/Quaternion.html.

14. Documentation of Camera in Unity3D - https://docs.unity3d.com/ru/530/Manual/CamerasOverview.html.

15. Ибатуллин Р. У. Астрономічні формули. 2014.

16.https://ru.wikipedia.org/wiki/Вектор_(геометрия)#/media/File:Vector_subtraction.png.

17. https://docs.unity3d.com/ru/530/Manual/UnderstandingVectorArithmetic.html.

18. https://ru.wikipedia.org/wiki/Вектор_(геометрия)#/media/File:Vector_addition.svg.

19. https://docs.unity3d.com/ru/530/Manual/UnderstandingVectorArithmetic.html.

20. https://docs.unity3d.com/ru/530/Manual/UnderstandingVectorArithmetic.html.

21. https://ru.wikipedia.org/wiki/Углы_Эйлера#/media/File:Euler.png.

22. https://ru.wikipedia.org/wiki/Углы_Эйлера#/media/File:Euler2a.gif.

23. https://gamedev.ru/files/images/82238_1350915041_gimbal.jpg.

3 СИСТЕМА ЧАСТИНОК І РОБОТА ОСВІТЛЕННЯ В ВІРТУАЛЬНОМУ ПРОСТОРІ

У розділі "Система частинок і робота освітлення в віртуальному просторі" було виконано ряд поставлених завдань для дипломного проекту, такі як: Аналіз роботи системи частинок в Unity3D; Розробка системи частинок для нульового об'єкта; Аналіз роботи освітлення у віртуальному просторі.

3.1 Розробка моделей для побудови сцени

Для даної роботи потрібно спроектувати 17 об'єктів. Сфера-подібної форми потрібно розробити 15 об'єктів, 2 об'єкти матиме ламану ліній (багатокутна полігональну модель).

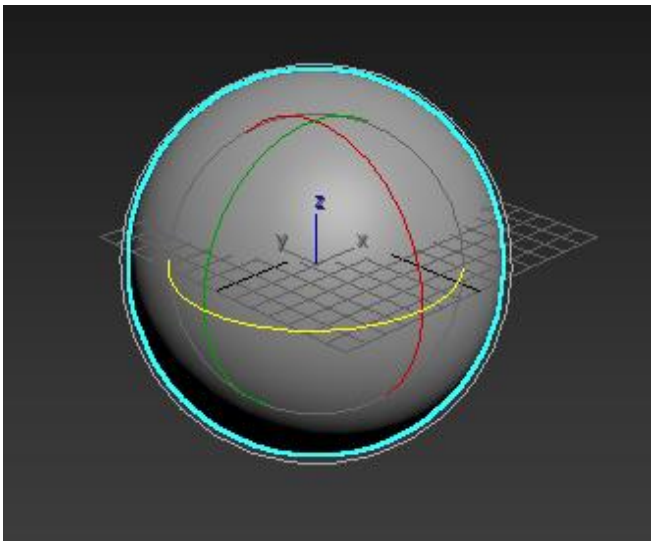


Рисунок 1.5. Побудови 3D модель сферо-подібної форми [Автор: Шаповалов Олександр]

Створюємо об'єкт під номером 3.0. Відкриємо Standard Primitives (стандартні примітиви) і створимо об'єкт sphere (сфера) і поставимо її параметри Radius (радіус) - 2500mm, Segments (сегменти) - 100 (від цього параметра залежить, наскільки об'єкт буде круглим).

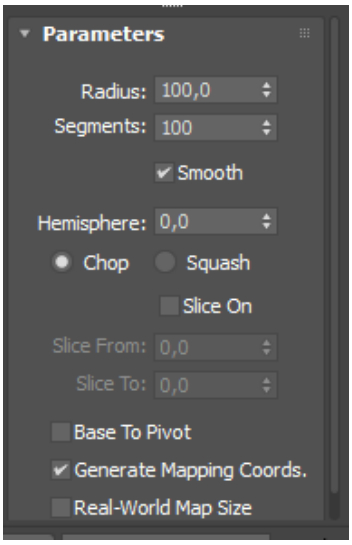


Рисунок 1.6. Встановлення параметрів об'єкта [Автор: Шаповалов Олександр]

Відкриємо Material Editor натиснувши M на клавіатурі або через головне меню: Rendering> Material Editor; Кількома по кнопці Material Editori обираємо зі списку Materail / Map Browser матеріал VRayMlt як показано на малюнку нижче.

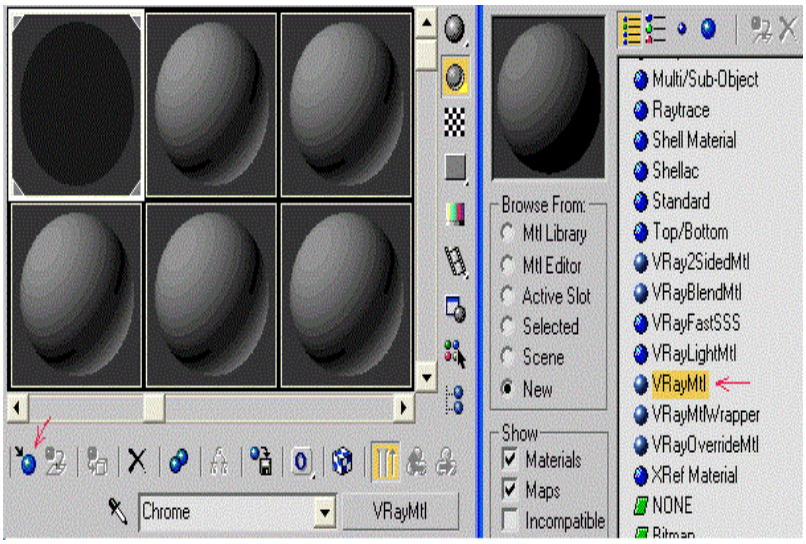


Рисунок 1.7. Додавання матеріал [Автор: Шаповалов Олександр]

Накладаємо текстуру на дану модель.

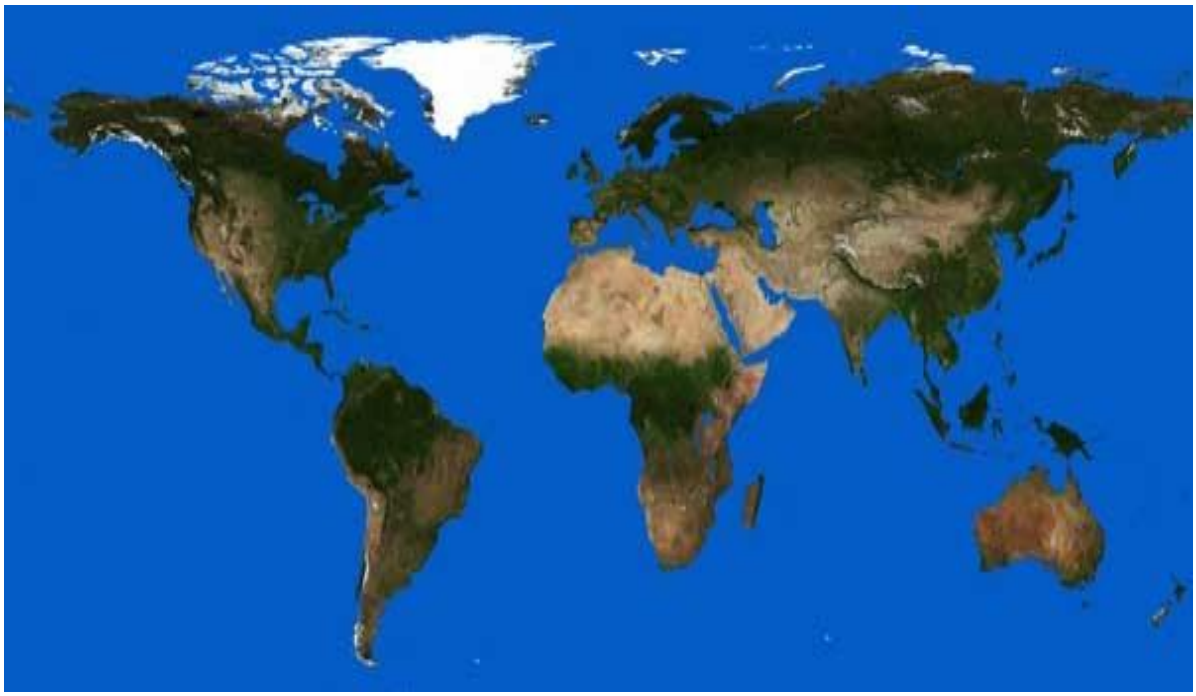


Рисунок 1.8. Перша текстура [14]

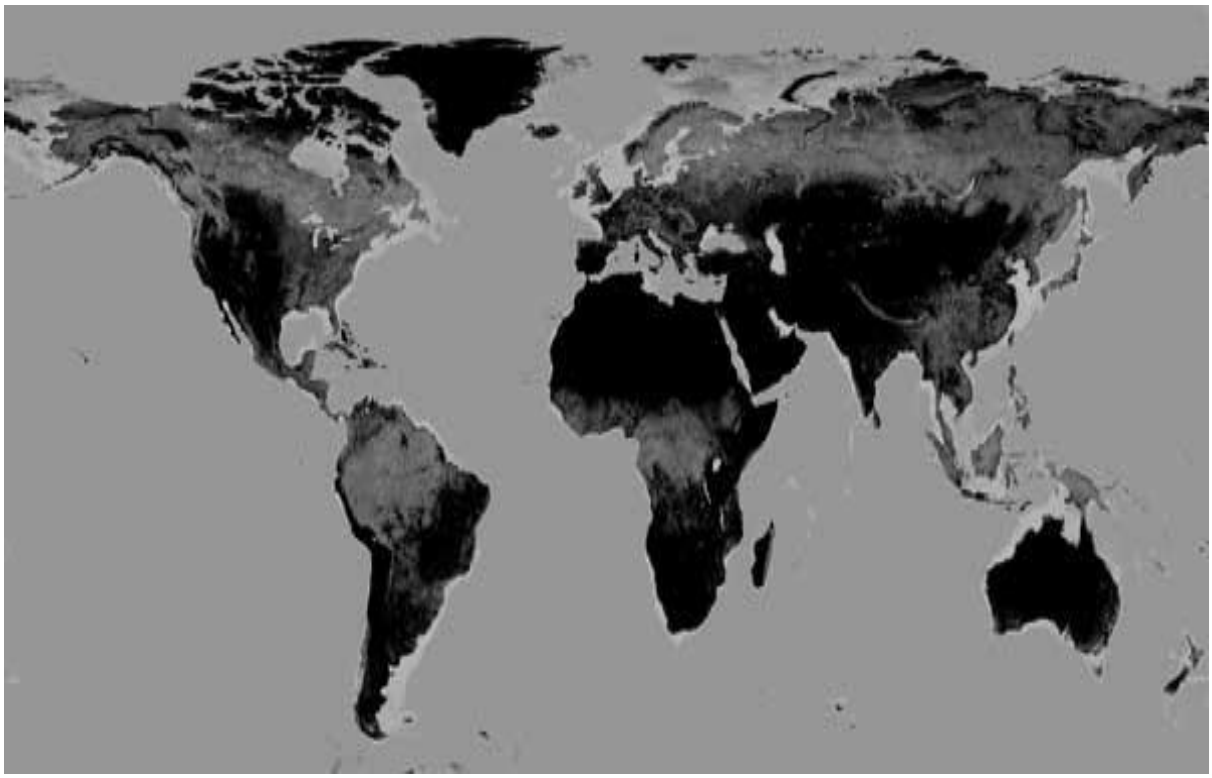


Рисунок 1.9. Друга текстура [14]

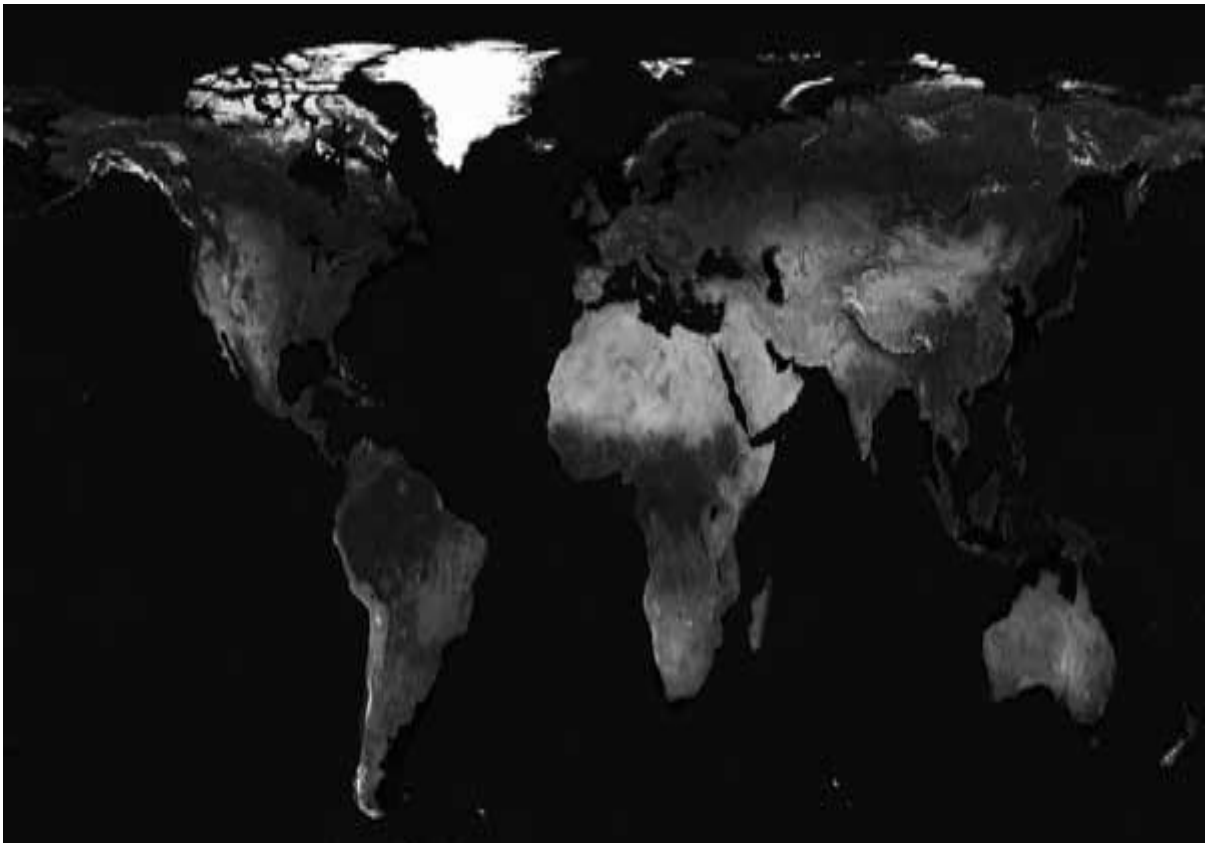


Рисунок 1.10. Третья текстура [14]

Тепер перейдемо до самих налаштувань матеріалу VRayMlt. Встановимо настройки як показано на малюнку внизу. Як карти Diffuse встановимо Texture 1, Reflect - Texture 2, Refl.glossiness - Texture 3, Refract Texture 2.

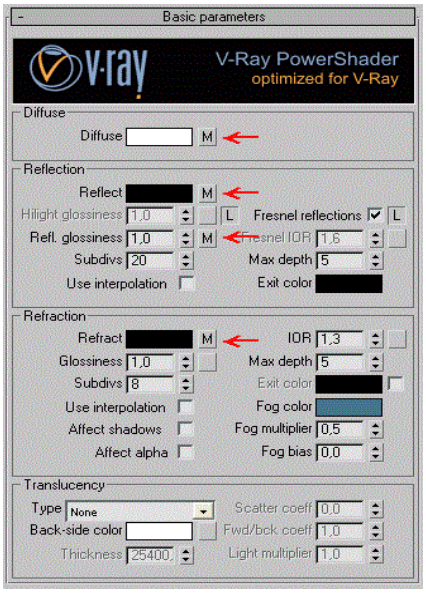


Рисунок 1.11. Налаштування V-ray [Автор: Шаповалов Олександр]

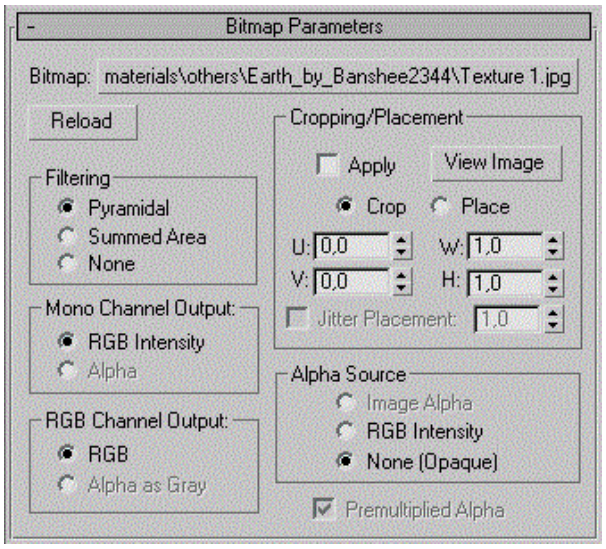


Рисунок 1.12. Налаштування Bitmap Parameters [Автор: Шаповалов Олександр]

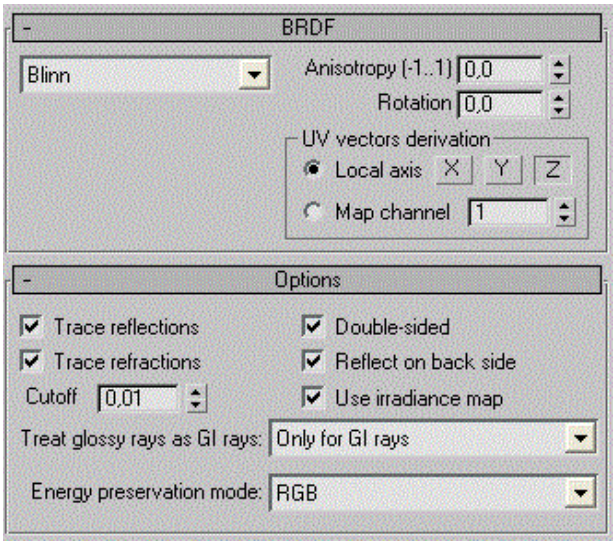


Рисунок 1.13. Налаштування BRDF Options [Автор Шаповалов Олександр]

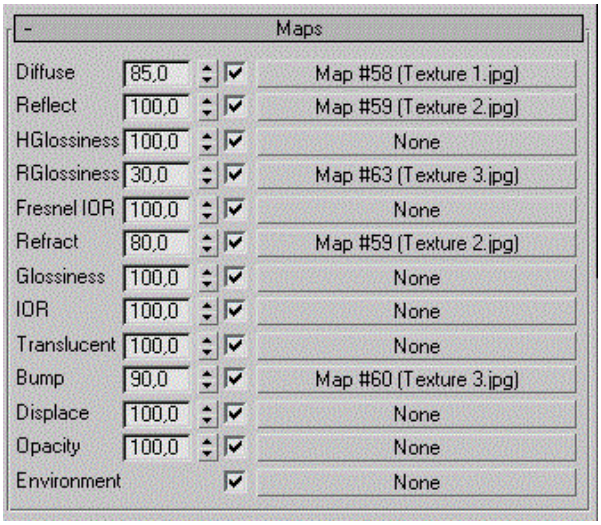


Рисунок 1.14. Налаштування Maps (Автор Шаповалов Олександр)

Створюємо многокутну полігональну модель типу (box).

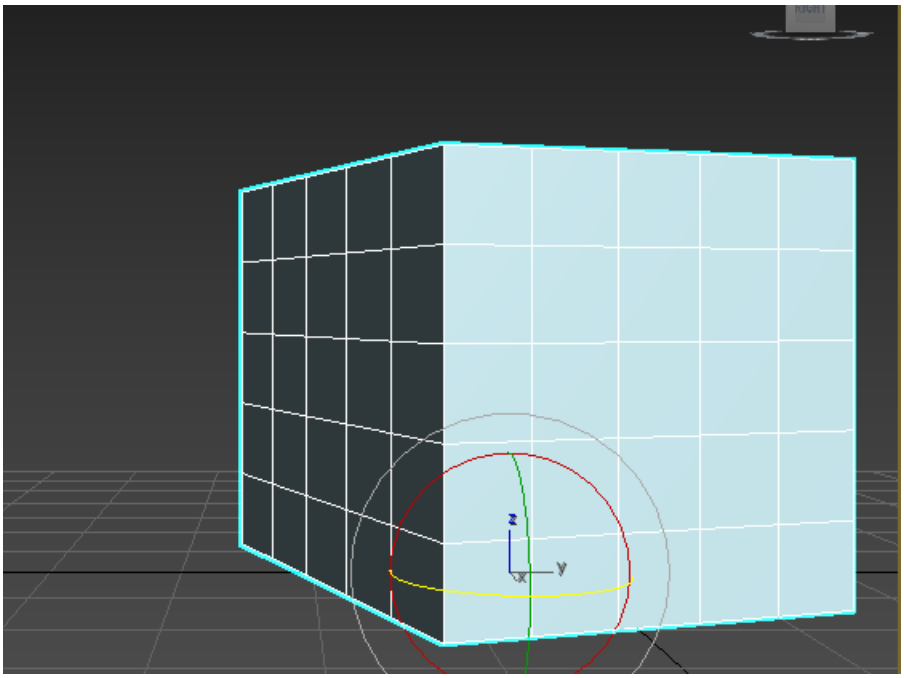


Рисунок 1.15. Модель типу Box [Автор Шаповалов Олександр]

Для даного об'єкта додаємо модифікатор (Spherfiy)

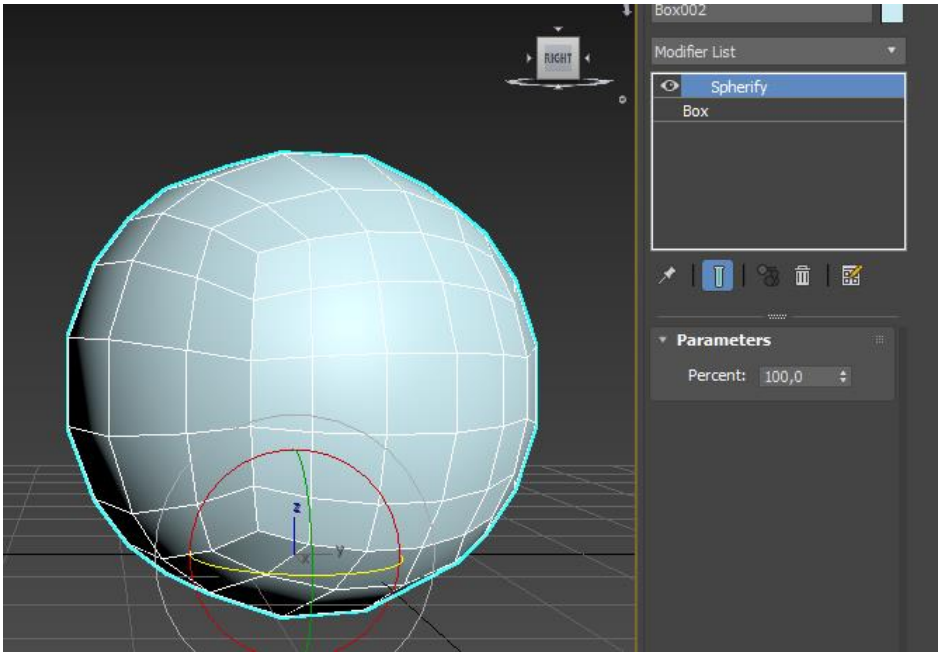


Рисунок 1.16. Додавання модифікатора (Spherfiy) (Автор Шаповалов Александр)

Так само додаємо модифікатор (TurboSmooth). І виставляємо такі параметри: Iteration - 6 збільшуємо кількість полігонів на об'єкті; Render items - 6 тим самим витрата обчислювача комп'ютера саме задіяний в процесі рендеринга.

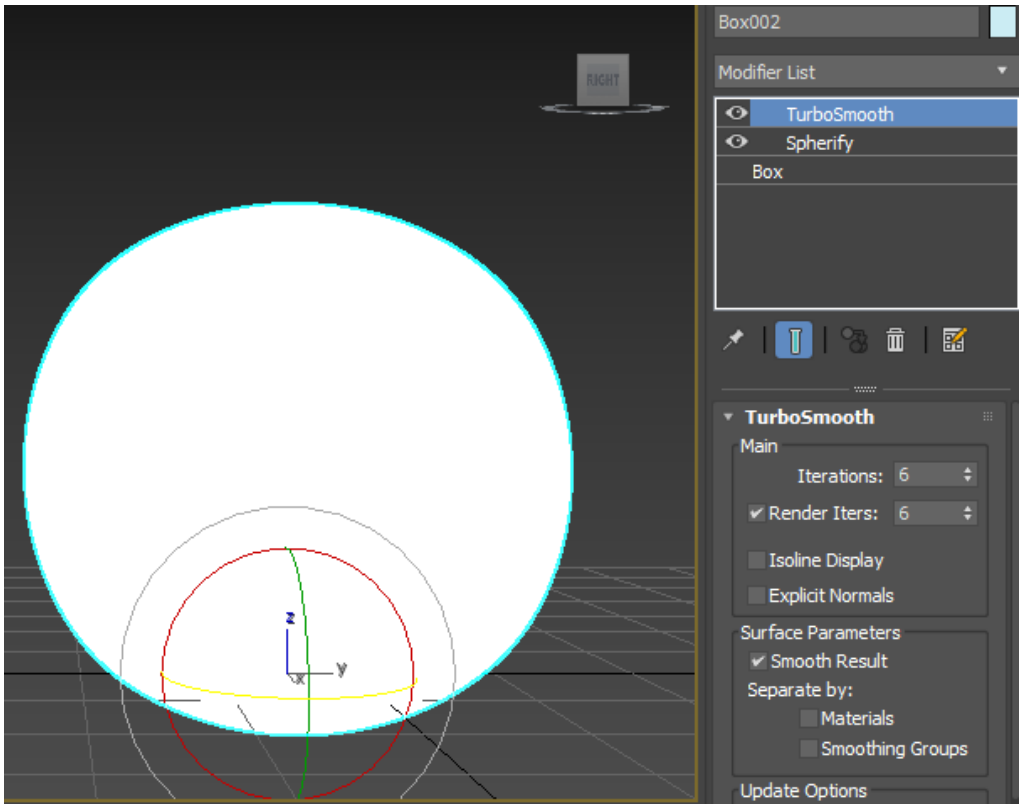


Рисунок 1.17. Додавання модифікатора (Spherfiy) [Автор Шаповалов Олександр]

Відкриємо Material Editor натиснувши M на клавіатурі або через головне меню: Rendering> Material Editor; Кількома по кнопці Material Editori обираємо Get Material. Вибираємо матеріал у вільний слот. Даний матеріал використовують для надання об'єкту нерівностей.

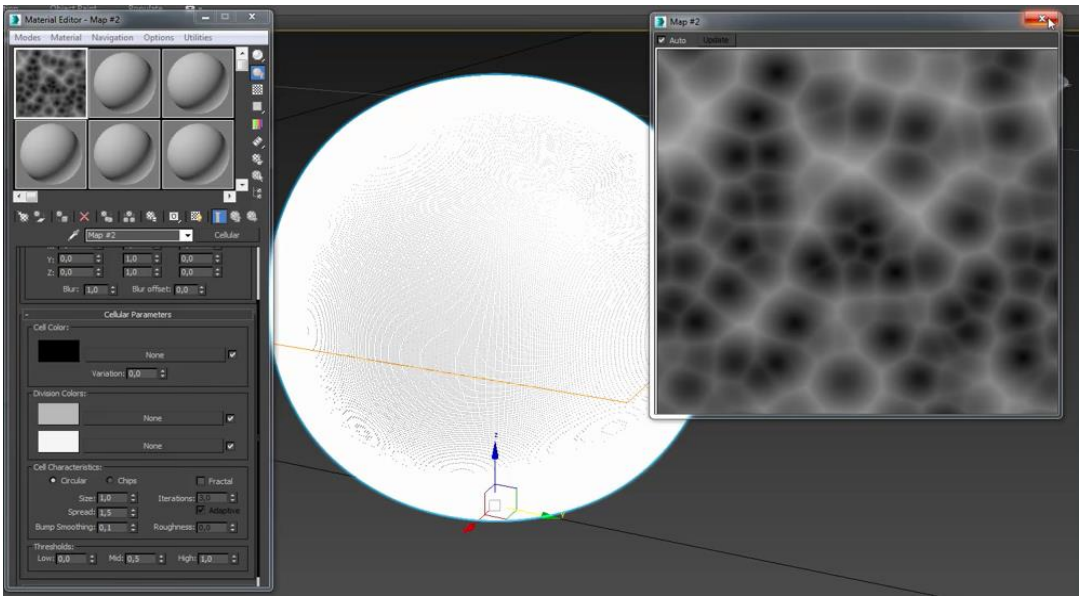


Рисунок 1.18. Додавання матеріалу [Автор Шаповалов Олександр]

Додаємо два модифікатор (Desplace) який виступає інструментом для зміни форми тіла об'єкта.

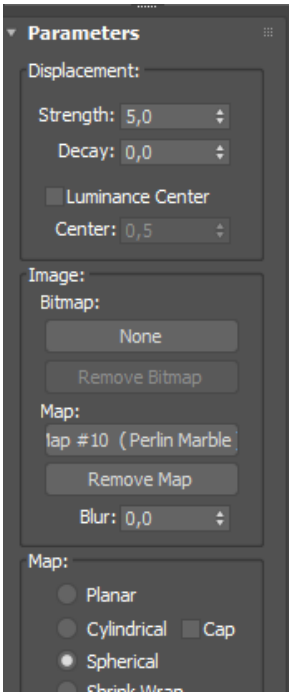


Рисунок 1.19. Параметри модифікатора (Desplace 1) [Автор Шаповалов Олександр]

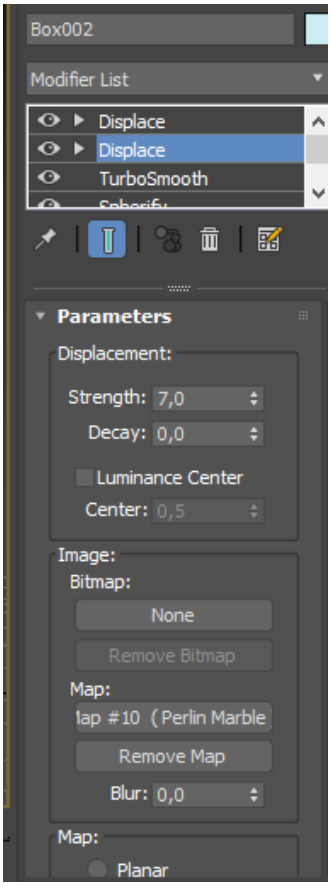


Рисунок 1.20. Параметри модифікатора (Desplace 2) [Автор Шаповалов Олександр]

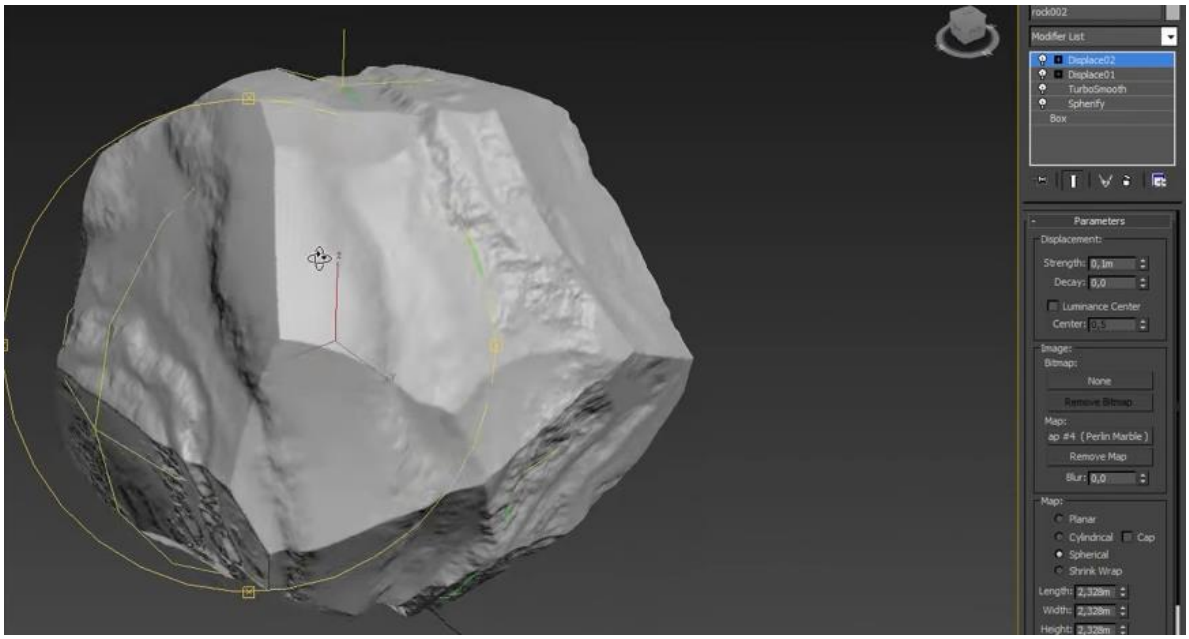


Рисунок 1.21. Многокутну полігональну модель [Автор Шаповалов Олександр]

3.2 Система частинок

Unity3D має компонент Particle System. Компонент Particle System створює різні ефекти шляхом генерації і анімації в сцені великої кількості невеликих 2D зображень.

Частинки представляють собою невеликі, прості зображення або сітки, які відображаються і переміщаються в великих кількостях системою частинок. Кожна частинка являє собою невелику частину рідкої або аморфної сутності, і ефект всіх частинок разом створює враження повної сутності.

Динаміка системи. Кожна частка має певний час життя (зазвичай кілька секунд), протягом якого частка може зазнати різні зміни. Частка починає своє існування, коли вона створена, або видана своєю системою частинок. Система випускає частки в випадкових точках в просторі, обмеженому регіоном в формі сфери, півсфери, конусом, прямокутним паралелепіпедом або мешем довільної форми. Частка відображається до тих пір, поки не закінчиться час її життя, потім вона видаляється з системи. Частота випускання частинок системи жорстко визначає кількість частинок, що випускаються в секунду, хоча точний час випускання містить невеликий фактор випадковості. Частота випускання в сукупності із середнім часом життя частинки визначають кількість "стабільних" частинок (тобто, тих, чиє випускання і знищення відбуваються з однаковою частотою) і час, необхідний системі для досягнення такого стану.

Динаміка частинок. Налаштування випускання і часу життя впливають на загальну поведінку системи, однак окремі частинки теж можуть змінюватися з часом. Кожна частка

має вектор швидкості, що визначає напрямок і відстань, яку проходить частинка за один кадр. Швидкість може бути змінена за допомогою сил і гравітації, що застосовуються самою системою або коли частки здуваються в зонах вітру. Колір, розмір і обертання кожної частки також можуть змінюватися протягом часу життя або пропорційної її поточної швидкості руху. Колір включає альфа-канал (для прозорості), так що частка може плавно зникати і з'являтися замість різкої зміни видимості в таких випадках[1].

Властивості. Компонент Particle System має множину різних налаштувань, перелік модулів для редагування анімації і ефектів. Дані налаштування знаходяться на панелі інспектора.

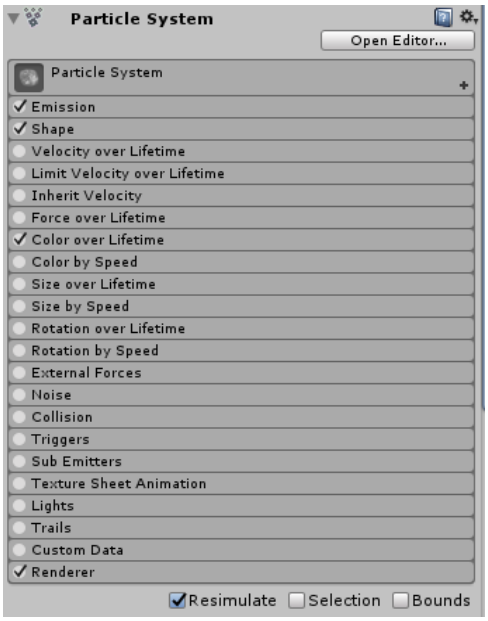


Рисунок 1.22. Компонент Particle System [Автор Шаповалов Олександр]

3.3 Розробка системи частинок для нульового об'єкта

Для нульового об'єкта потрібно розробити систему частинок. Підключаємо модулі. такі як:

Модуль системи частинок (Particle System Module) [2]

Даний модуль містить глобальні настройки, які впливають на всю систему в цілому.

Duration	Тривалість часу, протягом якого буде працювати система.
Looped	Якщо включена, система почне свою роботу в місці завершення свого duration циклу і почне все заново.
Prewarm	Якщо включена, система буде форматувати так, ніби вона вже пройшла

	через весь цикл своєї роботи (працює виключно тоді, коли Looping також включений).
Start Delay	Затримка в секундах до запуску системи випромінювання частинок після її включення.
Start Lifetime	Початкова тривалість життя частинок.
Start Speed	Початкова швидкість кожної частки у відповідному їй напрямку.
Start Size	Початковий розмір кожної частки.
Start Rotation	Початковий кут повороту кожної частки.
Start Color	Початковий колір кожної частки.
Gravity Multiplier	Масштабування значення гравітації в менеджері фізики (physics manager). Значення рівне 0 призведе до відключення гравітації.
Inherit Velocity	Всі частинки починають свій рух з однієї і тієї ж швидкістю.
Simulation Space	Щодо який з систем координат повинні почати свій рух частинки? Локальної (і як наслідок рухатися разом з самими об'єктом) або світової?
Play on Awake	Чи починає система свою роботу автоматично в момент створення об'єкту?
Max Particles	Максимальна кількість частинок одночасно знаходяться в системі. Старі частки будуть видалені, коли буде досягнута межа їх життєвого циклу.

В даному модулі виставляємо наступні настройки.

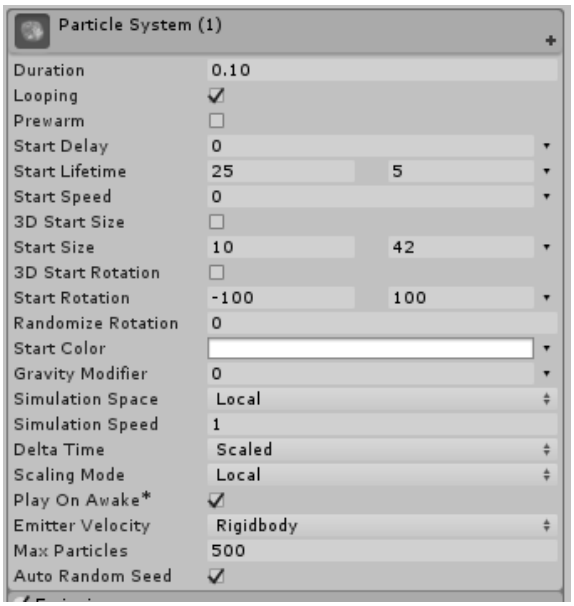


Рисунок 1.23.Налаштування компонент Particle System [Автор Шаповалов Олександр]

Модуль випромінювання (Emission Module)

Властивості цього модуля впливають на частоту і синхронізацію випромінювання частинок.

Rate	Кількість частинок випромінювання за одиницю часу або пройденого шляху (вибирається з відповідного меню, що випадає).
Bursts	Можливість додатковим часткам випромінюватися в потрібний момент часу (доступно тільки коли Rate знаходиться в режимі Time).

В даному модулі виставляємо наступні настройки.

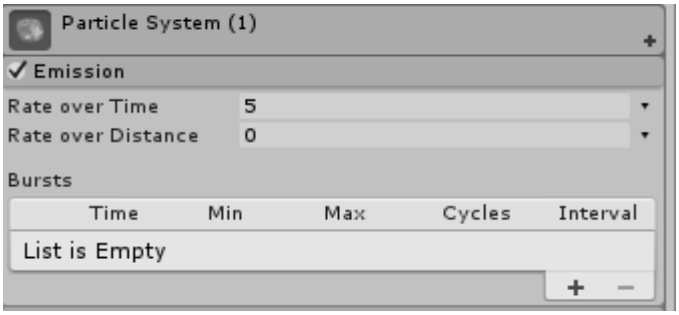


Рисунок 1.24. Модуль випромінювання (Emission Module**)** [Автор Шаповалов Олександр]

Модуль-Форма (Shape Module)

Цей модуль використовується для виділення форми випромінюваного модуля і місця звідки випромінюються самі частинки.

Shape	Форма обсягу випромінювача: сфера (Sphere), півсфера (Hemisphere), конус (Cone), куб (Box) або меш (Mesh). Для меш форми є додаткове меню для вибору того чи будуть частки випромінюватися з його вершин, трикутників або ж граней.
Random Direction	При включенні, початковий напрямок частинок буде вибрано довільним чином (випадково).
Radius	Радіус радіального співвідношення форми (тільки для сфери (Sphere), півсфери (Hemisphere) і конуса (Cone)).
Angle	Кут конуса, розташований в своїй точці (тільки для конуса). Значення кута, що дорівнює 0 на виході дасть циліндр, в той час як значення рівне 90 градусам дасть на виході форму плоского диска.
Box X, Y, Z	Ширина, висота і глибина кубічної форми (тільки для кубів (Box)).
Mesh	Меш, що задає форму випромінювача (тільки для мешів (Mesh)).
Emit from Shell	Чи повинні частинки випромінюватися з зовнішньої поверхні форми або все ж з внутрішньої? (Тільки для сфер і півсфер).
Emit from	Чи повинні частинки випромінюватися з зовнішньої поверхні форми або все ж з внутрішньої? (Тільки для сфер и півсфер).

В даному модулі виставляємо наступні настройки.

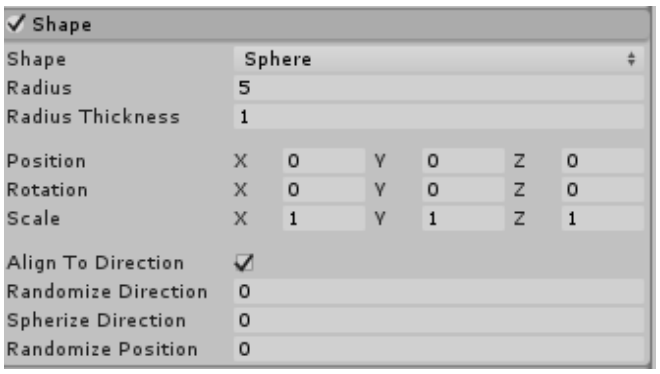


Рисунок 1.25. Модуль-Форма (Shape Module)

[Автор Шаповалов Олександр]

Модуль кольору щодо періоду життя частинок (Color Over Lifetime)

Даний модуль визначає як частки з плином часу змінюють свій колір і прозорість.

Color	Градієнт кольору частки за час свого існування.

В даному модулі виставляємо наступні настройки.

Рисунок 1.26 Модуль кольору щодо періоду життя частинок (Color Over Lifetime) [Автор Шаповалов Олександр]

Модуль обертання по відношенню до періоду життя частинок (Rotation Over Lifetime Module)

Організованість обертання частинок в потрібному порядку під час їх руху.

Angular Velocity	Швидкість обертання в градусах в секунду.

В даному модулі виставляємо наступні настройки.

Рисунок 1.26 Модуль обертання по відношенню до періоду життя частинок (Rotation Over Lifetime Module) [Автор Шаповалов Олександр]

Модуль рендеринга

Ці настройки модуля визначають як трансформується текстура частинок або меш, промальовані або перекриті іншими частинками.

Render Mode	Як виходить відрендерене зображення з графічного зображення (або меша). Варіанти такі Billboard (частка завжди дивиться на камеру, Stretched Billboard (дивиться на камеру, але з застосованим масштабуванням), Horizontal Billboard (площину частинок паралельна площині XZ площині "статі"), Vertical Billboard (частка перпендикулярна щодо світової осі Y, але повернута обличчям до камери) і Mesh (частки візуалізуються з заважав замість текстури).
Camera Scale	Кількість розтягування застосоване пропорційно руху камери (доступно тільки в режимі Stretched Billboard).

Speed Scale	Кількість розтягування застосоване пропорційно швидкості частинок (доступно тільки в режимі Stretched Billboard).
Length Scale	Кількість розтягування застосоване пропорційно довжині частинок (доступно тільки в режимі Stretched Billboard).
Mesh	Для рендеру частинок використовується один або кілька мешів (тільки для режиму Mesh).
Normal Direction	Зсув нормалей світла використовується для відображення частинок. Значення рівне 1.0 направляє нормалі в сторону камери, в той час як значення 0.0 направляє їх в центр екрану (тільки в режимі Billboard).
Material	Матеріал використаний при рендер частинок.
Sort Mode	Порядок, в якому відмальовані частки (і отже накладені). Допустимими значеннями є By Distance (тобто з камери), Youngest First і Oldest First.
Sorting Fudge	Зсув порядку сортування частинок. Невеликі значення збільшують ймовірність того, що частинки будуть відображатися поверх інших прозорих об'єктів, в тому числі частинок з інших систем.
Cast Shadows	Чи повинні частинки відкидати тіні на інші об'єкти? Тільки непрозорі матеріали відкидають тіні.
Receive Shadows	Чи повинні тіні відкидатися на частки? Тільки непрозорі матеріали здатні приймати тіні.
Max Particle Size	Найбільший розмір часток (незалежно від інших параметрів), по відношенню до розміру вікна проекції.

В даному модулі виставляємо наступні настройки.

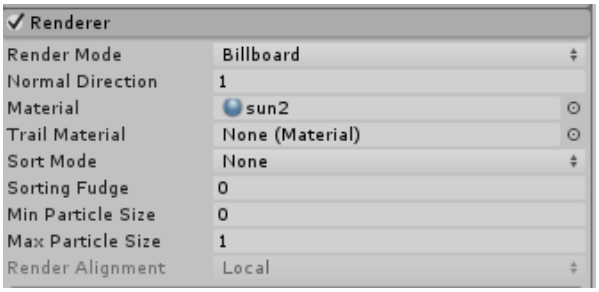


Рисунок 1.28 Модуль рендеринга[Автор Шаповалов Олександр]

Після настройки Practical System, система частинок закріплюється за нульовим об'єктом, нульовий об'єкт буде виглядає наступним чином.

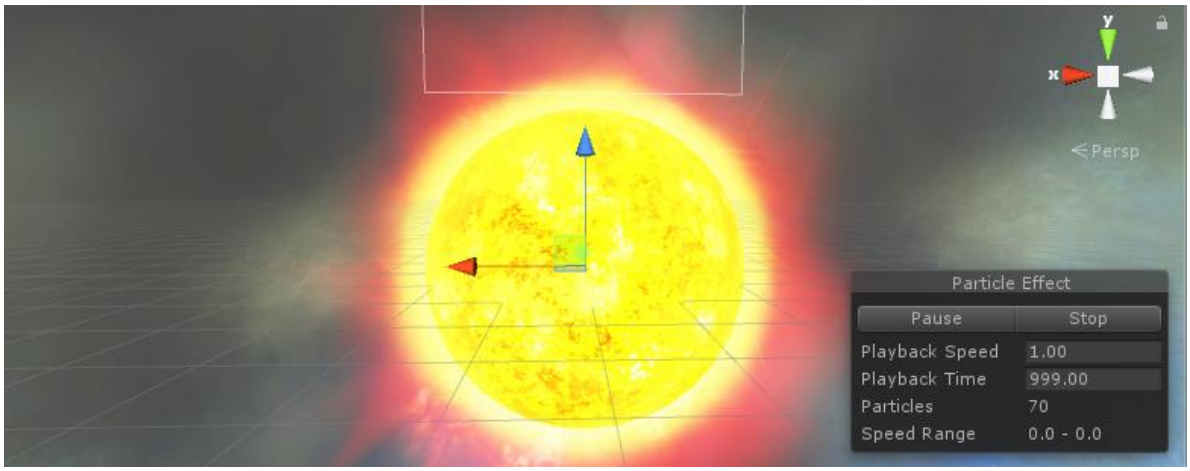


Рисунок 3.1. Нульовий об'єкт з системою частинок [Автор Шаповалов Олександр]

3.4 Робота освітлення у віртуальному просторі

Освітлення є невід'ємною частиною графічної обробки і стилістики побудови сцен у віртуальному просторі. Освітлення змінює зовнішній вигляд об'єктів і текстур, тим самим вносить глибину побудови об'єктів і реалістичність.

Освітлення в комп'ютерній графіці і в побудові сцен ділитися на дві категорії:

1. **Direct Illumination** - пряме попадання променів світла на поверхню.
2. **Indirect Illumination** - промені відбиваються від поверхні, розсіюються і утворюють м'який світло, що заповнює [3].

Метод обчислення відбитого освітлення

Global Illumination (GI) - вдає із себе найбільш "чесний" спосіб симуляції відбитого світла. З джерела світла вилітають фотони, що несуть інформацію про колір і яскравість світла. Вдаряючись об яку-небудь поверхню, вони висвітлюють її, але втрачають частину енергії, внаслідок чого їх колір і яскравість змінюються. Потім фотони відскакують і вдаряються об наступну поверхню, повторно втрачаючи частину енергії. Так відбувається кілька разів залежно від налаштувань рендера.

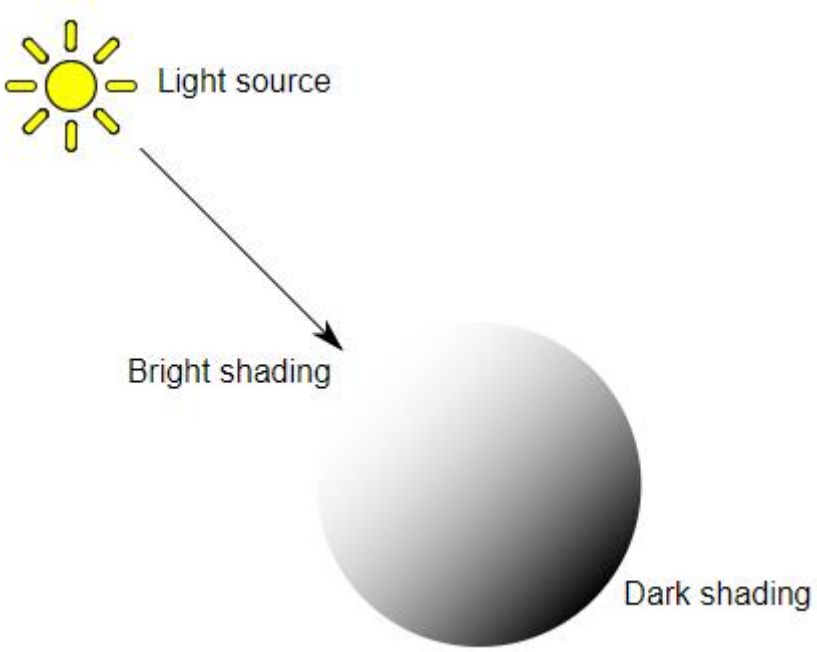


Рисунок 3.2. Принцип роботи освітлення з об'єктами [4]

Принципи роботи освітлення в Unity3D

У Unity3D доступно 5 джерел світла:

1. Point Lights

Point Lights розташований в певній точці в просторі і рівномірно посилає світло в усіх напрямках. Напрямок світла, що потрапляє на поверхню - це лінія від точки контакту до центру світового об'єкта. Інтенсивність зменшується з відстанню від світла, досягаючи нуля в заданому діапазоні.

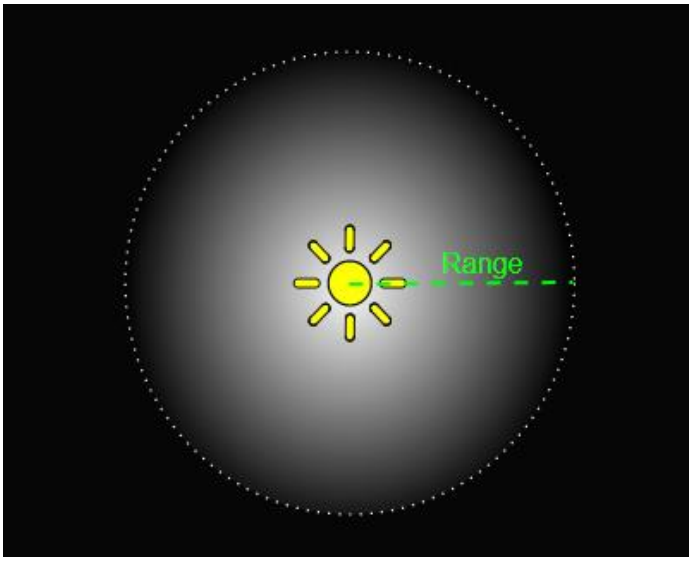


Рисунок 3.3. Принцип роботи Point Lights [4]

2. Spot Lights

Spot Light має певне місце розташування і діапазон, за яким світло падає. Проте, пляма світла обмежена кутом, що призводить до конусоподібної області освітлення.

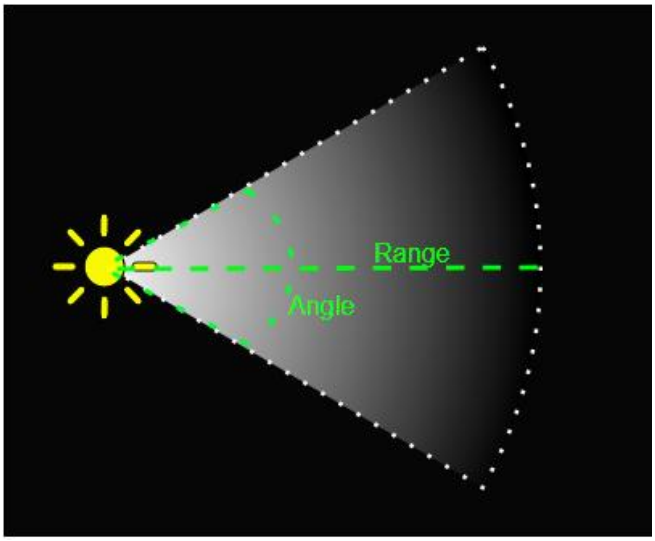


Рисунок 3.4. Принцип роботи Spot Light [4]

3. Directional Lights

Directional Lights не має будь-якої ідентифікованої позиції джерела освітлення. Являє собою множину паралельних один одному променів. Відстань світла від цільового об'єкта не визначено, тому світло не зменшується і потрапляє на всі об'єкти з однаковою інтенсивністю.

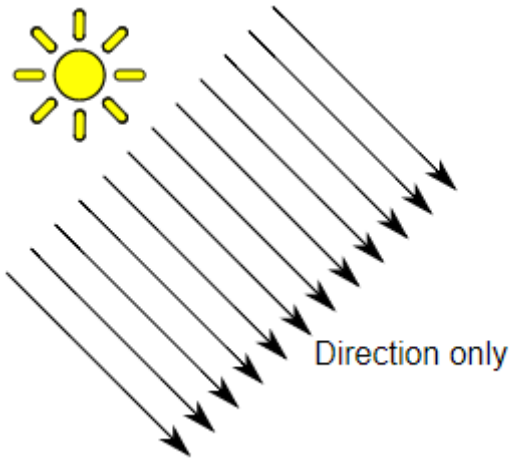


Рисунок 3.5. Принцип роботи Directional Lights [4]

4. Area Lights

Світло визначається прямокутником в просторі. Світло випромінюється у всіх напрямках, але тільки з одного боку прямокутника. Світло падає в зазначеному діапазоні. Оскільки область світла висвітлює об'єкт з декількох різних напрямків одночасно, затінення має тенденцію бути більш м'яким і тонким, ніж інші типи світла.

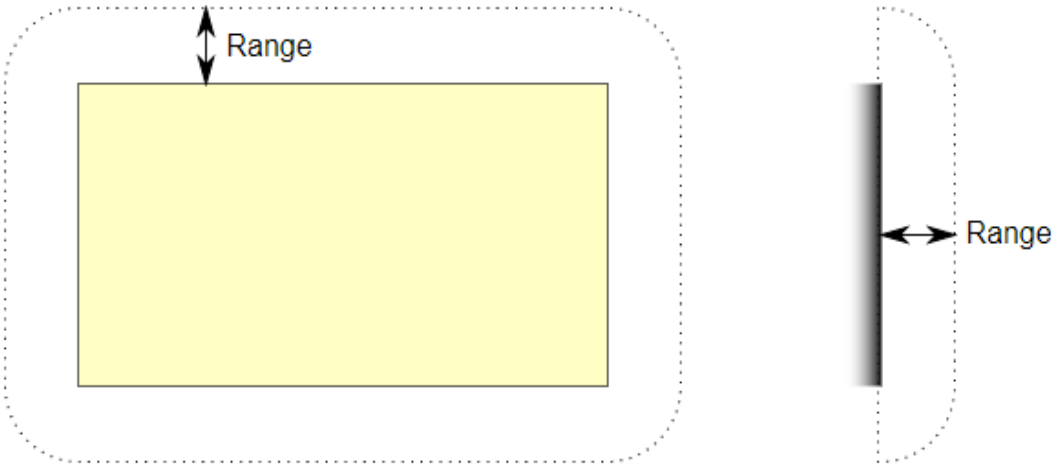


Рисунок 3.6. Принцип роботи Area Lights [4]

У Unity всі об'єкти діляться на динамічні (dynamic) і статичні (static). Статичними об'єктами називаються ті, які завжди стоять на місці і нікуди не зміщуються. Для них відбувається "запечені" освітлення. Динамічні об'єкти - це ті, які знаходяться в русі.

В інспектор компоненту Practical System підключається джерело освітлення типу Point Light. Встановлюємо радіус напрямки променів по сцені [4].

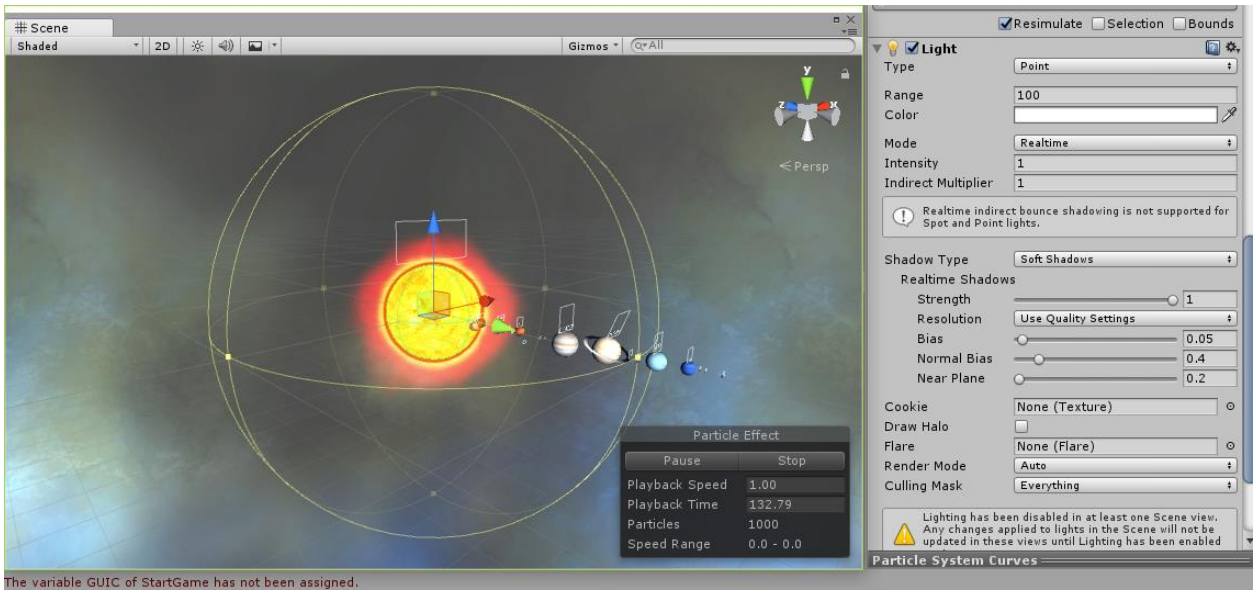


Рисунок 3.7. Розміщення точки освітлення в сцені [Автор Шаповалов Олександр]

Висновок до третього розділу

Освітлення, анімація є одним з ключових чинників при створінь 3D додатків. Дані два чинника дозволяє надати реалістичність сцені. Правильна настройка освітлення і анімацій може змінювати кардинально візуально сприйняття.

У даній роботі була розробка анімація для нульового об'єкта з допомогу функції PracticalSystem яка вбудована в Unity3D. Шляхом різних налаштувань був створений сонячний ефект. Дана анімація була закріплена за нульовим об'єктом. Також за нульовим об'єктом закріплюватися точка освітлення Point Light. Основними настройки освітлення були: установка певного радіуса поширення світла по сцені; установка колірної гами освітлення.

Список використаної літератури в третьому розділі

1. Particle system documentation in Unity3D - https://docs.unity3d.com/ru/530/Manual/class-ParticleSystem.html.

2. Documentation of the particle system modules in Unity3D - https://docs.unity3d.com/ru/530/Manual/PartSysMainModule.html.

3. Артем Філіпов. Робота з освітленням в Unity - https://habr.com/post/266839/.

4. Documentation of lighting overview in Unity3D - https://docs.unity3d.com/ru/530/Manual/Lighting.html.

4 РОЗРОБКА КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ

У розділі "Розробка користувацького інтерфейсу" було виконано ряд поставлених завдань для дипломного проекту, такі як: Аналіз основних принципів розробки інтерфейсу; Аналіз основних аспектів реалізації інтерфейсу; Розробка головного меню; Розробка завантажувального екрану; Розробка інтерфейсу основної сцени.

4.1 Основні принципи розробки інтерфейсу призначеного для користувача

1. Принцип структуризації. Інтерфейс повинен бути структурований так, щоб близькі за змістом, родинні його частини були пов'язані видимим чином, а незалежні - розділені; схожі елементи повинні виглядати схоже, а несхожі - відрізнятися.
2. Принцип простоти. Найбільш поширені дії повинні виконуватися максимально просто, щоб істотно знизити зусилля, що витрачаються.
3. Принцип спостережливості. Всі операції і дані, необхідні для вирішення певного завдання, повинні бути добре видно, коли користувач намагається її вирішити. Зайва і непотрібна інформація не повинна відволікати увагу користувача.
4. Принцип зворотного зв'язку. Користувач повинен отримувати повідомлення про дії програми та про важливі події всередині неї. Повідомлення повинні бути ясними, короткими, недвозначними і написаними на мові, зрозумілій користувачеві.
5. Принцип толерантності. Інтерфейс повинен бути гнучким і терпимим до помилок користувачів. Збиток від помилок повинен знижуватися за рахунок можливості скасування або повтору дій, а також за рахунок розумної інтерпретації будь-яких розумних дій користувача і введених їм даних (наприклад, допускати різну послідовність виконання етапів завдання і різні форми введення вихідних даних).
6. Принцип повторного використання. Щоб поведінку програми було передбачувано, корисно повторно використовувати відомі прийоми і рішення, застосовувані при розробці інтерфейсів для інших програм.

4.2 Основні аспекти реалізації призначеного для користувача інтерфейсу

Невдалі інтерфейси сильно уповільнюють роботу користувачів, сприяють швидкої стомлюваності, великому числу помилок і, як наслідок, відбивають у користувачів подальше бажання працювати з програмою. У зв'язку з цим розробнику програми слід

враховувати відомі рекомендації провідних фахівців в області створення призначеного для користувача інтерфейсу.

Розподіл інформації на екрані дисплея

Логічніше і найприродніше починати взаємодія з користувачем з лівого верхнього кута екрану, поступово переводячи увагу користувача на правий нижній кут. Найважче працювати з інтерфейсами, окремі частини яких розташовані на екрані в хаотичному, непередбачуваному порядку. В цьому випадку користувач змушений постійно перемикати свій погляд на різні, що не пов'язані між собою ділянки екрану для пошуку потрібної інформації. Особливо погано, якщо елементи (візуалізуються на екрані об'єкти) при цьому відрізняються мініатюрними розмірами.

Не слід також перевантажувати екранний простір різними другорядними елементами (об'єктами). Користувачі в цьому випадку будуть постійно плутатися в інтерфейсі, губитися серед нагромадження різних (як основних, так і другорядних) елементів (об'єктів). Не треба впадати і в іншу крайність, коли на екрані залишається величезна кількість вільного місця. Все це призводить до неефективного використання екранного простору.

Так само вимагаєте прагнути до збалансованості при виборі місця розташування для керуючих елементів і видимих об'єктів: зліва і справа, зверху і знизу їх кількість повинна бути приблизно однаковим. Не варто будь-що-будь домагатися при цьому ідеальної симетрії. Однак занадто перекошений інтерфейс навряд чи може сприяти створенню у користувача відчуття комфорту. Не останню роль в естетичної привабливості інтерфейсу грає вирівнювання елементів (об'єктів) з верхнім і лівим кордонів екрану, дотримання пропорцій при виборі розмірів для зображення цих елементів (об'єктів)[1].

4.3 Розробка головного меню

Спочатку для проекту потрібно розробити головне меню. Дане меню буде з'являтися після запуску проекту. І буде мати кілька основних функцій. Головне меню буде мати наступний вигляд:

Рисунок 4.1 Стартове меню [Автор Шаповалов Олександр]

Функція виходу викликає закриття програми. Код має наступний вигляд:

Підключити бібліотеки

using UnityEngine;

using System.Collections;

Функція виходу

public class ApplicationManager : MonoBehaviour {

public void Quit () {

#if UNITY_EDITOR

UnityEditor.EditorApplication.isPlaying = false;

#else

Application.Quit();

#endif}}

Функція налаштування викликає підменю різних налаштувань аудіо і відео. Підміню буде виглядає наступним чином:

Рисунок 4.2 Підміню налаштувань [Автор Шаповалов Олександр]

Меню настройки викликає приховане підміню налаштувань. Так само приховує головне меню. Код має наступний вигляд:

Підключити бібліотеки.

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

Підключаємо змінні.

```
public class PanelManager : MonoBehaviour {
public Animator initiallyOpen;
private int m_OpenParameterId;
private Animator m_Open;
private GameObject m_PreviouslySelected;
const string k_OpenTransitionName = "Open";
const string k_ClosedStateName = "Closed";
```

Функція відображення панелі.

```
public void OnEnable(){
m_OpenParameterId = Animator.StringToHash (k_OpenTransitionName);
if (initiallyOpen == null)
return;
```

Функція відкриття панелі.

```
OpenPanel(initiallyOpen);
```

```
public void OpenPanel (Animator anim)
{if (m_Open == anim)
return;
anim.gameObject.SetActive(true);
var newPreviouslySelected = EventSystem.current.currentSelectedGameObject;
anim.transform.SetAsLastSibling();
Функція закриття панелі.
CloseCurrent();
m_PreviouslySelected = newPreviouslySelected;
m_Open = anim;
m_Open.SetBool(m_OpenParameterId, true)
GameObject go = FindFirstEnabledSelectable(anim.gameObject);
SetSelected(go);}
public void CloseCurrent(){
if (m_Open == null)
return;
m_Open.SetBool(m_OpenParameterId, false);
SetSelected(m_PreviouslySelected);
StartCoroutine(DisablePanelDeleyed(m_Open));
m_Open = null;}
IEnumerator DisablePanelDeleyed(Animator anim){
bool closedStateReached = false;
bool wantToClose = true;
while (!closedStateReached && wantToClose){
if (!anim.IsInTransition(0))
closedStateReached =
anim.GetCurrentAnimatorStateInfo(0).IsName(k_ClosedStateName);
wantToClose = !anim.GetBool(m_OpenParameterId);
yield return new WaitForEndOfFrame();}
if (wantToClose)
anim.gameObject.SetActive(false);}
```

Налаштування графіки складається з трьох позицій: Низька якість; Середня якість; Висока якість. Підміню налаштування графіки буде виглядає наступним чином:

Рисунок 4.3 Підміню налаштувань графіки [Автор Шаповалов Олександр]

Код для налаштування графіки матиме такий вигляд:

Підключити бібліотеки.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class MenuScriptSetting : MonoBehaviour {
public int level1;
// Update is called once per frame
void Update () {}
```

Функція низької якості графіки.

```
public void lowQulity(){
QualitySettings.SetQualityLevel(0, true);}
```

Функція середньої якості графіки.

```
public void MedQulity()  {
QualitySettings.SetQualityLevel(2, true);}
```

Функція високої якості графіки.

```
public void HQulity(){
QualitySettings.SetQualityLevel(4, true);}
```

Налаштування аудіо має: ScrollBar який налаштовує гучності музики в проекті. Підміню настройки аудіо буде виглядає наступним чином:

Рисунок 4.4 Підміню налаштувань аудіо [Автор Шаповалов Олександр]

Код для налаштування аудіо матиме такий вигляд:

Підключити бібліотеки.

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
public class Settings : MonoBehaviour {
```

Підключаємо змінні.

```
public static float moveSpeed = 0.4f;
public static float lookSpeed = 5;
public static float music = 0.8f;
public static float volume = 1;
// Use this for initialization
void Start () {}
// Update is called once per frame
```

Функція налаштування звуку.

```
void Update () {}
public void moveSpeed_slider (float speed) {
moveSpeed = speed;}
public void lookSpeed_slider (float speed) {
lookSpeed = speed;}
public void music_slider (float v) {
music = v;}
public void volume_slider (float v) {
```

```
volume = v;}}
```

Функція старт запускає під сцену завантаження, так званий екран завантаження. Код завантаження матиме такий вигляд:

```
Підключити бібліотеки.
using UnityEngine;
using System.Collections;
public class StartGame : MonoBehaviour {
public static bool first_load = true;
public GameObject GUIC;
// Use this for initialization
void Start () {}
// Update is called once per frame
Функція збереження налаштування звуку.
void Update () {
GUIC.GetComponent<AudioSource> ().volume = Settings.music;
AudioListener.volume = Settings.volume;}
Функція завантаження під сцени.
public void NewGame () {
first_load = false;
Time.timeScale = 0;
Application.LoadLevel("Loadscena3");
Time.timeScale = 1;}}
```

4.4 Розробка завантажувального екрану

Завантажувальний екран - заставка, що з'являється на екрані в момент завантаження збереженої гри, або при переході між локаціями. Завантажувальний екран служить для завантаження анімації, системи частинок, промальовування і завантаження 3D об'єктів на сцені. Завантажувальний екран виглядає наступним чином:

Рисунок 4.5. Завантажувальний екран [Автор Шаповалов Олександр]

Коду завантажувального екрану матиме такий вигляд:

Підключаємо змінні.

```
public  var level_Name:      String;
public  var texture_loading : Texture;
private var AsOp:        AsyncOperation;
private var loading_progress: float = 0;
private var Round_load;
function Start () {
```

Завантаження рівня.

```
AsOp=Application.LoadLevelAsync(level_Name);
```

Забороняємо перемикатися на завантажений рівень відразу після завантаження.

```
AsOp.allowSceneActivation=false; }
```

Функція завантаження сцени.

```
function OnGUI(){
if(loading_progress <= 100){
loading_progress += Time.deltaTime*25;
Round_load = Mathf.RoundToInt(loading_progress);
```

Якщо завантаження завершена тоді.

```
if(AsOp.progress == 0.9 && Round_load == 100){
```

З'являється повідомлення (натисніть на будь-яку клавішу).

```
GUI.Label(new Rect(Screen.width/2-100,Screen.height/2+200,300,300),"Нажмите на любую клавишу");
```

Якщо натиснута будь-яка клавіша тоді.

if(Input.anyKey){

Завантажуємо рівень.

AsOp.allowSceneActivation=true;

Якщо завантаження виконується тоді.

if(AsOp != null){

Завантаження відбувається в процентах.

GUI.Label (Rect (Screen.width/2-65,Screen.height/2+220,Screen.width,Screen.height),"Загрузка: " + Round_load);

Промальовування текстури: Ширина; Висота.

GUI.DrawTexture(Rect(0,Screen.height-60,loading_progress/100*Screen.width,40), texture_loading); }

4.5 Розробка інтерфейсу основної сцени

Інтерфейс основної сцени має кілька функції, а саме: подання інформації про об'єкти містять в цій сцені; внесення зміни роботи об'єктів в реальному часі. Інтерфейс основної сцени виглядає наступним чином:

Рисунок 4.6. Інтерфейс основної сцени [Автор Шаповалов Олександр]

У верхній частині екрану розташовані кнопки для управління швидкості руху об'єктів. А саме: зменшення швидкості руху об'єктів; зупинка руху об'єктів; запуск руху об'єктів; збільшення швидкості руху об'єкта. Також панель має кнопку (меню) яка повертає в головне меню.

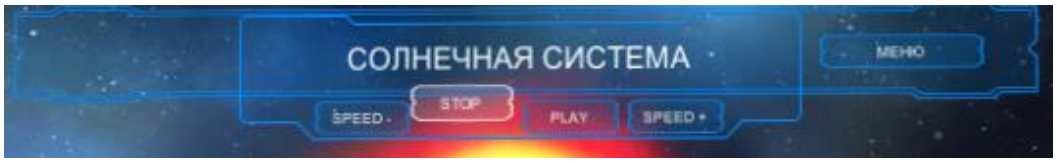


Рисунок 4.7. Верхня частина інтерфейсу [Автор Шаповалов Олександр]

Для зміни швидкості руху об'єктів потрібно реалізувати скрипт, який буде змінювати пармаметри руху об'єктів в скрипті який зафіксований на нульовому об'єкті. Коду матиме такий вигляд:

Підключити бібліотеки.

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

Підключаємо змінні.

```
public class SpeedButton : MonoBehaviour {
private GameObject Suntr;
private GameObject Suncf;
private GameObject SunSpeed;
void Update () {}
```

Функція запуску руху об'єктів.

```
public void OnClickStart() {
Suntr = GameObject.Find("Sun");
Suntr.GetComponent<PlanetsScript>().speed = 2;
Suncf = GameObject.Find("Sun");
Suncf.GetComponent<PlanetsScript>().coeff = 100.0f;
SunSpeed = GameObject.Find("Sun");
SunSpeed.GetComponent<PlanetsScript>().Sunspeed = 10;}
```

Функція зупинки руху об'єктів.

```
public void OnClickStop(){
Suntr = GameObject.Find("Sun");
Suntr.GetComponent<PlanetsScript>().speed = 0;
Suncf = GameObject.Find("Sun");
Suncf.GetComponent<PlanetsScript>().coeff = 0.0f;
SunSpeed = GameObject.Find("Sun");
SunSpeed.GetComponent<PlanetsScript>().Sunspeed = 0;}
```

Функція збільшення швидкості руху об'єктів.

```
public void OnClickSpeedUp(){
```

```
Suntr = GameObject.Find("Sun");
Suntr.GetComponent<PlanetsScript>().speed = 20;
Suncf = GameObject.Find("Sun");
Suncf.GetComponent<PlanetsScript>().coeff = 1000.0f;
SunSpeed = GameObject.Find("Sun");
SunSpeed.GetComponent<PlanetsScript>().Sunspeed = 50;}
```

Функція зменшення швидкості руху об'єктів.

```
public void OnClickSpeedDown(){
Suntr = GameObject.Find("Sun");
Suntr.GetComponent<PlanetsScript>().speed = 1;
Suncf = GameObject.Find("Sun");
Suncf.GetComponent<PlanetsScript>().coeff = 50.0f;
SunSpeed = GameObject.Find("Sun");
SunSpeed.GetComponent<PlanetsScript>().Sunspeed = 5;}}
```

У нижній частині екрана розміщено кнопки подання інформації про об'єкт. Спочатку при завантаженя рівня панель з кнопками буде предаставляет інформацію про нульовий об'єкт.

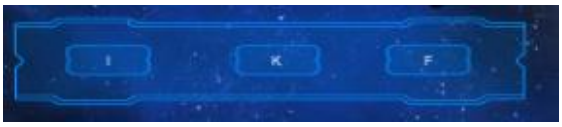


Рисунок 4.8. Нижня частина інтерфейсу [Автор Шаповалов Олександр]

Основні функції кнопок: Кнопка (I) відкриває панель в якій міститься теоретична інформація про об'єкт;

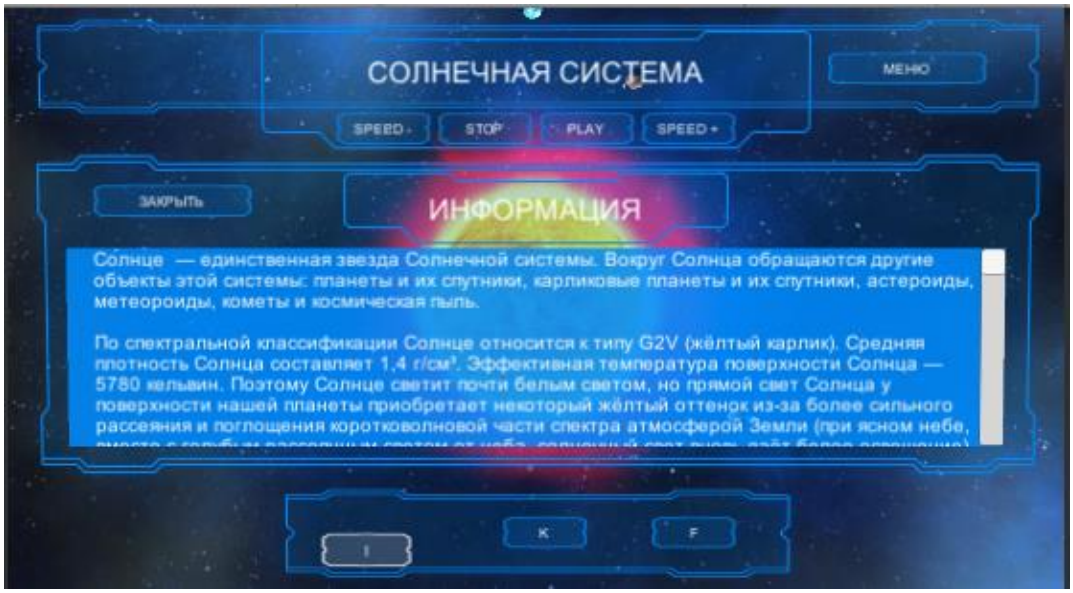


Рисунок 4.9. Панель теоретичної інформації [Автор Шаповалов Олександр]

Кнопка (K) відкриває панель в якій міститься характеристика об'єкта;

Рисунок 4.10. Панель характеристики об'єкта [Автор Шаповалов Олександр]

Кнопка (F) відкриває панель в якій міститься реальні фотографії об'єкта.

Рисунок 4.11. Панель реальних фотографії об'єкта [Автор Шаповалов Олександр]

Код відкриття панелей інформації про об'єкти:

Підключити бібліотеки.

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

Підключаємо змінні.

```
public class Interface2 : MonoBehaviour {
```

Дані змінні створюють в інспектора осередку куди поміщають панелі.

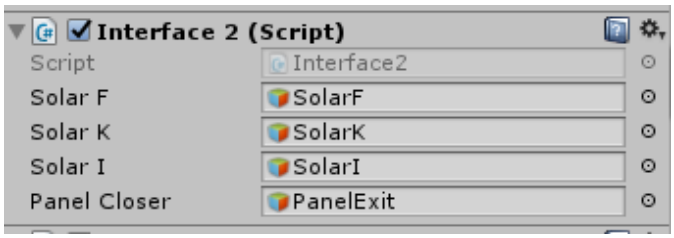


Рисунок 4.12. Інспектор камери [Автор Шаповалов Олександр]

```
public GameObject SolarF;
public GameObject SolarK;
public GameObject SolarI;
public GameObject PanelCloser;
void Update () {}
```

Функція відкриття панелі інформації.

```
public void OnClickSolarI() {
SolarI.SetActive(true);
SolarK.SetActive(false);
SolarF.SetActive(false);
PanelCloser.SetActive(false);}
```

Функція відкриття панелі характеристики об'єкта.

```
public void OnClickSolarK(){
SolarI.SetActive(false);
SolarK.SetActive(true);
PanelCloser.SetActive(false);
SolarF.SetActive(false); }
```

Функція відкриття панелі фотографій об'єкта.

```
public void OnClickSolarF(){
SolarI.SetActive(false);
SolarK.SetActive(false);
PanelCloser.SetActive(false);
SolarF.SetActive(true); }}
```

Для кожного об'єкта так само створені дані панелі. Спочатку вони знаходяться в режимі скритності. Після натискання на об'єкт дана панель буде відображена користувачеві. А панель яка була відкрита перейде в режим скритності. Код режима скритності буде мати такий вигляд:

Підключимо бібліотеки.

```
using System.Collections;
```

using System.Collections.Generic;

using UnityEngine;

Підключаємо змінні.

public class StepPlanetInterface : MonoBehaviour {

Дані змінні створюють в інспектора осередку куди поміщають панелі.

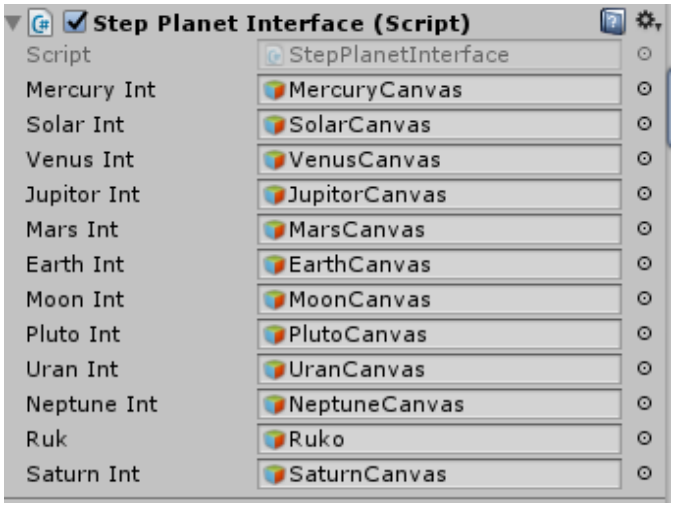


Рисунок 4.13. Інспектор камери [Автор Шаповалов Олександр]

public GameObject MercuryInt;

public GameObject SolarInt;

public GameObject VenusInt;

public GameObject JupitorInt;

public GameObject MarsInt;

public GameObject EarthInt;

public GameObject MoonInt;

public GameObject PlutoInt;

public GameObject UranInt;

public GameObject NeptuneInt;

public GameObject Ruk;

public GameObject SaturnInt;

void Update () { }

Функція режиму скритності панелі і функція відображення панелі залежна від перехід на опредленія об'єкт. Під час виконання функцій всі інші панелі переходять в режим скритності.

public void OnClickMercury(){

MercuryInt.SetActive(true);

SolarInt.SetActive(false);

JupitorInt.SetActive(false);

```
        VenusInt.SetActive(false);
        MarsInt.SetActive(false);
        EarthInt.SetActive(false);
        MoonInt.SetActive(false);
        PlutoInt.SetActive(false);
        UranInt.SetActive(false);
        NeptuneInt.SetActive(false);
        SaturnInt.SetActive(false);
    }
```

Аналогічна функція пишеться для всіх об'єктів які знаходяться на сцені.

Висновок до четвертого розділу

Цінність будь-якої програми визначається не тільки її здатністю безпомилково і якісно вирішувати поставлену задачу, а й тим, наскільки легко і зручно цією програмою можна користуватися. Навіть найпростіша, і в той же час, функціональна програма виявиться незатребуваною, якщо користувач при роботі з нею швидко втомлюється, дратується, відчуває себе некомфортно. У цьому випадку користувач зазвичай робить більше помилок і працює менш продуктивно (тобто неефективно). Для отримання дійсно зручної програми, з якою б було приємно працювати, необхідний добре продуманий інтерфейс. Інтерфейс є найважливішою частиною будь-якої програми, так як саме з інтерфейсом користувач зустрічається в процесі своєї роботи. З точки зору користувача саме інтерфейс є кінцевим продуктом. Інтерфейс являє собою сукупність використовуваних в програмі засобів для введення даних, способів відображення інформації на екрані дисплея, елементів для управління різними етапами рішення задачі. Основне призначення (головна мета) призначеного для користувача інтерфейсу - забезпечити короткий і зручний шлях до отримання результату.

Інтерфейс був розділений на три етапи: Перший етап є головне меню програми; Другий етап є завантажувальним екраном між сценами програми; Третій етап є інтерфейс який надає інформацію про об'єкти, і зміни параметрів руху об'єктів.

Список використаної літератури в четвертому розділі

1. І.В.Горячей. Інтерфейс: призначення, принципи розробки та практичні рекомендації

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Завданням даної роботи бакалавра являє собою дослідження засобів і алгоритмів створення тривимірних астрономічних об'єктів з використанням чисельного моделювання орбітальної динаміки. Метою роботи є створення навчального додатка, який дозволяє шкільної або студентської аудиторій придбати знання про сонячну систему або поліпшити їх. Так як в процесі проектування виконувалось у побутових умовах, то аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для персонального комп'ютера, на якому розробляється система.

5.1 Загальні питання з охорони праці

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. В законі України «Про охорону праці» визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

При роботі з обчислювальною технікою змінюються фізичні і хімічні фактори навколишнього середовища: виникає статична електрика, електромагнітне випромінювання, змінюється температура і вологість, рівень вміст кисню і озону в повітрі. Повітря забруднюється шкідливими хімічними речовинами антропогенного походження за рахунок деструкції полімерних матеріалів, які використовуються для обробки приміщень та обладнання. Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці.

5.1.1 Правові та організаційні основи охорони праці

Основним організаційним напрямом у здійсненні управління в сфері охорони праці є усвідомлення пріоритету безпеки праці і підвищення соціальної відповідальності держави, і особистої відповідальності працівників.

Державна політика в галузі охорони праці визначається відповідно до Конституції України Верховною Радою України і спрямована на створення належних, безпечних і здорових умов праці, запобігання нещасним випадкам та професійним захворюванням. Відповідно до статті 3 Закону України «Про охорону праці» (далі – Закону) законодавство про охорону праці складається з Закону, Кодексу законів про працю України, Закону України "Про загальнообов'язкове державне соціальне страхування від нещасного випадку на виробництві та професійного захворювання, які спричинили втрату працездатності" та прийнятих відповідно до них нормативно-правових актів, норм міжнародного договору (ратифіковані Конвенції і Рекомендації МОТ, директиви Європейської Ради).

На законодавчому рівні визначено такі пріоритетні напрямки з безпеки праці:

- кожен працівник несе безпосередню відповідальність за порушення зазначених Законом, нормами і правилами вимог;
- напрямки реалізації конституційного права громадян на їх життя і здоров’я в процесі трудової діяльності:
- пріоритет життя і здоров’я працівників по відношенню до результатів виробничої діяльності підприємства;
- повна відповідальність роботодавця за створення належних – безпечних і здорових умов праці;
- соціальний захист працівників, повне відшкодування збитків особам, які потерпіли від нещасних випадків на виробництві та професійних захворювань;
- комплексне розв’язання завдань охорони праці;
- підвищення рівня промислової безпеки шляхом забезпечення суцільного технічного контролю за станом виробництв, технологій та продукції, а також сприяння підприємствам у створенні безпечних та нешкідливих умов праці;
- соціальний захист працівників, повне відшкодування збитків особам, які потерпіли від нещасних випадків на виробництві та професійних захворювань;
- використання економічних методів управління охороною праці, участь держави у фінансуванні заходів щодо охорони праці;

- використання світового досвіду організації роботи щодо поліпшення умов і підвищення безпеки праці на основі міжнародної співпраці.

Обов'язки працівників щодо дотримування вимог нормативно-правових актів з охорони праці (ст. 14), відповідальність робітників всіх категорій за порушення вимог щодо охорони праці (ст. 44), та іншими затвердженими власними нормативними актами з питань охорони праці (правилами, нормами, регламентами, положеннями, стандартами, інструкціями та іншими документами, обов'язковими до виконання), тобто тих, що діють на підприємстві/організації.

Наявні трудові відносини між працівниками і роботодавцями в Україні за темою дипломного проекту регулюються Кодексом законів про працю (КЗпП) України, відповідно до якого права працюючої людини на охорону праці охороняються всебічно та норми охорони праці неухильно інтегровані до правил внутрішнього розпорядку організації/підприємства.

5.1.2 Організаційно-технічні заходи з безпеки праці

В організації/підприємстві проводиться навчання і перевірка знань з питань охорони праці відповідно до вимог Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженого наказом Держнаглядохоронпраці України від 26.01.2005 N 15, зареєстрованого в Міністерстві юстиції України 15.02.2005 за N 231/10511 [10].

Також впроваджені організаційні заходи з пожежної безпеки - навчання і перевірку знань відповідно до вимог Типового положення про інструктажі, спеціальне навчання та перевірку знань з питань пожежної безпеки на підприємствах, в установах та організаціях України, затвердженого наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи від 29.09.2003 N 368, зареєстрованого в Міністерстві юстиції України 11.12.2003 за N 1148/8469 [12].

Обов'язковими вимогами враховане наступне:

- не слід допускати до роботи осіб, що в установленому порядку не пройшли навчання, інструктаж та перевірку знань з охорони праці, пожежної безпеки та цих Правил.

- на підприємстві/організації, де експлуатуються ЕОМ з відео дисплейними терміналами (ВДТ) і периферійними пристроями (ПП), розробляється інструкція з охорони праці відповідно до Положення про розробку інструкцій з охорони праці,

затвердженого наказом Держнаглядохоронпраці від 29.01.98 N 9, зареєстрованого в Міністерстві юстиції України 07.04.98 за N 226/2666 [11].

- ознайомлення з правилами безпеки праці, одержання відповідних інструктажів засвідчується у журналі інструктажів.
- перед допуском до самостійної роботи кожен працівник має право на навчання з питань охорони праці і роботодавець зобов'язаний, і проводить таке навчання у вигляді двох інструктажів з питань охорони праці:
- обов'язкові організаційні заходи перед початком, під час і після завершення роботи повинні включати перевірку (візуально) наявності і справності електрообладнання та його заземлення, а під час виконання роботи вимогу «не залишати без нагляду обладнання, яке працює». Після закінчення роботи - вимагається прибирання робочого місця, відключення всіх електроприладів від електромережі.

5.2 Аналіз стану умов праці

Робота над проектуванням та розробка навчального додатку проходить в побутовому приміщенні. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп'ютером.

5.2.1 Вимоги до приміщення

Геометричні розміри приміщення зазначені у таблиці 5.1

Таблиця 5.1 – розміри робочого місця

Параметр	Значення
Довжина, м	5
Ширина, м	3
Висота, м	2,5
Площа, $м^2$	15
Об'єм, $м^3$	37,5

Згідно до санітарних норм мікроклімату виробничих приміщень [8] розмір площі для одного робочого місця оператора персонального комп'ютера має бути не менше 6 кв.

м, а об’єм — не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

5.2.2 Вимоги до організації робочого місця

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця [13] (табл. 5.2) і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Таблиця 5.2 – Характеристика робочого місця

Найменування параметра	Фактичне значення	Нормативне значення
Висота робочої поверхні, мм	750	680 ÷800
Висота простору для ніг, мм	730	не менше 600
Ширина простору для ніг, мм	660	не менше 500
Глибина простору для ніг, мм	700	не менше 650
Висота поверхні сидіння, мм	470	400 ÷500
Ширина сидіння, мм	400	не менше 400
Глибина сидіння, мм	400	не менше 400
Висота поверхні спинки, мм	600	не менше 300
Ширина опорної поверхні спинки, мм	500	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	800	700 ÷800

Приміщення кабінету знаходиться на першому поверсі п'ятиповерхової будівлі і має об’єм 37,5 м3, площу – 15 м2. У цьому кабінеті обладнано одне робоче місце, яке укомплектовано 1 ПК.

Температура в приміщенні протягом року коливається у межах 18–24°С, відносна вологість — близько 50%. Система вентилювання приміщення — природна, а опалення — централізоване.

5.2.3 Навантаження та напруженість процесу праці

За фізичним навантаженням робота відноситься до категорії легкі роботи (Ia), її виконують сидячи з періодичним ходінням. Щодо характеру організування виконання дипломної роботи, то він підпадає під нав’язаний режим, оскільки певні розділи роботи необхідно виконати у встановлені конкретні терміни. За ступенем нервово-психічної напруги виконання роботи можна віднести до II – III ступеня і кваліфікувати як помірно напружений – напружений за умови успішного виконання поставлених завдань.

Найбільшому ризику виникнення різноманітних порушень піддаються: органи зору, м’язово скелетна система, нервово-психічна діяльність, репродуктивна функція у жінок.

Роботу за дипломним проектом визнано, таку, що займає 50% часу робочого дня та за восьмигодинної робочої зміни рекомендовано встановити додаткові регламентовані перерви тривалістю 15 хв через кожну годину роботи [7].

5.3 Виробнича санітарія

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

5.3.1 Аналіз небезпечних та шкідливих факторів при розробці виробу

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл. 5.3). Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання НПАОП 0.00.-1.28-10 [9], які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП. Переважно роботи за проектами виконують у кабінетах чи інших приміщеннях, де використовують різноманітне електрообладнання, зокрема персональні комп’ютери (ПК) та периферійні пристрої. Основними робочими характеристиками персонального комп'ютера є:

– робоча напруга U=+220В +-5%;
– робочий струм I=2A;
– споживана потужність P=600 Вт.

Робочі місця мають відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [7].

Таблиця 5.3 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Нормативні документи
Фізичні		
підвищена температура поверхонь обладнання	експлуатація ЕОМ, серверного обладнання для роботи	ДСН 3.3.6.042-99 [5]
підвищена або знижена вологість повітря	-//-	ДСН 3.3.6.042-99 [8]
підвищена або знижена рухливість повітря	-//-	ДСН 3.3.6.042-99 [8]
підвищений рівень напруги електричної мережі	-//-	ГОСТ 12.1.030-81 [2] ГОСТ 13109-97 [4]
підвищений рівень статичної електрики	-//-	ГОСТ 12.1.030-81 [2]
підвищена напруженість електромагнітного поля	-//-	ГОСТ 12.1.006-84 [1]
недостатність природного світла	порушення умов праці (вимог до приміщень)	ДБН В.2.5-28:2015 [6]
недостатнє освітлення робочої зони	порушення гігієнічних параметрів виробничого середовища	ДБН В.2.5-28:2015 [6]
Психофізиологічні		
нервово-психічна перевантаження	Розумова робота над проектом	НПАОП 0.00-1.28-10 [1] ДСанПіН 3.3.2.007-98 [7]
фізичні (статичне)	порушення умов праці та організації робочого часу	НПАОП 0.00-1.28-10 [1]

5.3.2 Пожежна безпека

Небезпека розвитку пожежі на обчислювальному центрі обумовлюється застосуванням розгалужених систем електроживлення ЕОМ, вентиляції і кондиціювання,. Небезпека загоряння пов'язана з особливістю комп'ютерів - із значною кількістю щільно розташованих на монтажній платі і блоках електронних вузлів і схем, електричних і комутаційних кабелів, резисторів, конденсаторів, напівпровідникових діодів і транзисторів. Надійна робота окремих елементів і мікросхем в цілому забезпечується тільки в певних інтервалах температури, вологості і при заданих електричних параметрах.

При відхиленні реальних умов експлуатації від розрахункових можуть виникнути пожежонебезпечні ситуації.

Простори усередині приміщень в межах, яких можуть утворюватися або знаходиться пожежонебезпечні речовини і матеріали відповідно до НАПБ Б.03.002-2007 [14] відносяться до пожежонебезпечної зони класу П-IIа. Це обумовлено тим, що в приміщенні знаходяться тверді горючі та важкозаймисті речовини та матеріали. Приміщенню, у якому розташоване робоче місце, присвоюється II ступень вогнестійкості.

Потенційними джерелами запалювання можуть бути:

- іскри і дуги короткого замикання;
- електрична іскра при замиканні і розмиканні ланцюгів;
- перегріви від тривалого перевантаження,
- відкритий вогонь і продукти горіння,
- наявність речовин, нагрітих вище за температуру самозаймання,
- розрядна статична електрика.

Причинами можливого загоряння і пожежі можуть бути:

- несправність електроустановки;
- конструктивні недоліки устаткування;
- коротке замикання в електричних мережах;
- запалювання горючих матеріалів, що знаходяться в безпосередній близькості від електроустановки[3].

5.3.3 Електробезпека

На робочому місці виконуються наступні вимоги електробезпеки: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія електромережі для живлення ПК, периферійних пристроїв і устаткування для обслуговування, виконана як окрема групова три- провідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників мають спеціальні контакти для підключення нульового захисного провідника. Електромережа штепсельних розеток для живлення персональних ПК, укладено по підлозі поруч зі стінами відповідно до затвердженого плану розміщення

обладнання та технічних характеристик обладнання. Металеві труби та гнучкі металеві рукави заземлені. Захисне заземлення включає в себе заземлюючих пристроїв і провідник, який з'єднує заземлюючий пристрій з обладнанням, яке заземлюється - заземлюючий провідник.

5.4 Гігієнічні вимоги до параметрів виробничого середовища

5.4.1 Мікроклімат

Мікроклімат робочих приміщень – це клімат внутрішнього середовища цих приміщень, що визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря. В даному приміщенні проводяться роботи, що виконуються сидячи і не потребують динамічного фізичного напруження, то для нього відповідає категорія робіт Іа. Отже оптимальні значення для температури, відносної вологості й рухливості повітря для зазначеного робочого місця відповідають нормам [8] і наведені в табл. 5.4.

Таблиця 5.4 – Норми мікроклімату робочої зони об’єкту

Період року	Категорія робіт	Температура C^0	Відносна вологість %	Швидкість руху повітря, м/с
Холодна	легка-1 а	22 - 24	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

5.4.2 Освітлення

Світло є природною умовою існування людини. Воно впливає на стан вищих психічних функцій і фізіологічні процеси в організмі. Хороше освітлення діє тонізуюче, створює гарний настрій, покращує протікання основних процесів вищої нервової діяльності.

У проекті, що розробляється, передбачається використовувати суміщене освітлення. У світлий час доби використовуватиметься природне освітлення приміщення через віконні отвори, в решту часу використовуватиметься штучне освітлення. Штучне освітлення створюється газорозрядними лампами.

У приміщенні, де розташовані ЕОМ передбачається природне бічне освітлення, рівень якого відповідає ДБН В.2.5-28:2015 Природнє і штучне освітлення [6]. Джерелом

природного освітлення є сонячне світло. Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДБН і для даного приміщення в світлий час доби достатньо природного освітлення.

Розрахунок освітлення.

Для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше -1/8, в побутових – 1/10:

$$S_b = \left(\frac{1}{5} \div \frac{1}{10}\right) \cdot S_n, \quad (5.1)$$

де Sb – площа віконних прорізів, м2;

Sn – площа підлоги, м2.

Sn = a·b = 5·3 = 15 м2,

S = 1/10·15 = 1,5 м2.

Приймаємо 1 вікно площею S=1,5 м2.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників n виробляється по формулі (5.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M}, \quad (5.2)$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м2; S = 15 м2;

Z – поправочний коефіцієнт світильника (1,1 для люмінесцентних ламп);

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. [14] – 0,575

M – число люмінесцентних ламп в світильнику – 3;

F – світловий потік лампи – 5200лм (для ЛБ-80-7).

Підставивши числові значення у формулу (5.2), отримуємо:

$$n = \frac{300 \times 15 \times 1.1 \times 1.5}{5200 \times 0.54 \times 3} = 0{,}8 \approx 1$$

Приймаємо освітлювальну установку, яка складається з 1-го світильника, які складаються з трьох люмінесцентних ламп загальною потужністю 80 Вт, напругою – 220 В [15].

5.4.3 Вентилювання

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в СНіП.

5.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Відповідно до санітарно-гігієнічних нормативів та правил експлуатації обладнання наводимо приклади деяких заходів безпеки.

1) Заходи безпеки під час експлуатації персонального комп'ютера та периферійних пристроїв передбачають:

- правильне організування місця праці та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;
- експлуатацію сертифікованого обладнання;
- дотримання заходів електробезпеки;
- забезпечення оптимальних параметрів мікроклімату;
- забезпечення раціонального освітлення місця праці (освітленість робочого місця не перевищувала 2/3 нормальної освітленості приміщення);
- облаштовуючи приміщення для роботи з ПК, потрібно передбачити припливно-витяжну вентиляцію або кондиціювання повітря:

2) Заходи безпеки під час експлуатації інших електричних приладів передбачають дотримання таких правил:

- постійно стежити за справним станом електромережі, розподільних щитків, вимикачів, штепсельних розеток, лампових патронів, а також мережевих кабелів живлення, за допомогою яких електроприлади під'єднують до електромережі;
- постійно стежити за справністю ізоляції електромережі та мережевих кабелів, не допускаючи їхньої експлуатації з пошкодженою ізоляцією;
- не тягнути за мережевий кабель, щоб витягти вилку з розетки;
- не закривати меблями, різноманітним інвентарем вимикачі, розетки;
- не підключати одночасно декілька потужних електропристроїв до однієї розетки, що може викликати надмірне нагрівання провідників, руйнування їхньої ізоляції, розплавлення і загоряння полімерних матеріалів;
- не залишати включені електроприлади без нагляду;

- не допускати потрапляння всередину електроприладів крізь вентиляційні отвори рідин або металевих предметів, а також не закривати їх та підтримувати в належній чистоті, щоб уникнути перегрівання та займання приладу;

- не ставити на електроприлади матеріали, які можуть під дією теплоти, що виділяється, загорітися (канцелярські товари, сувенірну продукцію тощо).

Висновки до п'ятого розділу

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом написаному в кваліфікаційній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника.

Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Була наведена схема, розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

Список використаної літератури в п'ятому розділі

1. ГОСТ 12.1.006-84 ССБТ. Электромагнитные поля радиочастот. Общие требования безопасности. Допустимые уровни на рабочих местах и требования к проведению контроля.

2. ГОСТ 12.1.030-81 ССБТ. Электробезопасность.Защитное заземление. Зануление.

3. ГОСТ 12.1.044-89 ССБТ. Пожаровзрывоопасность веществ и материалов. Номенклатура показателей и методы их определения.

4. ГОСТ 13109-97 Электрическая энергия. Совместимость технических средств электромагнитных.

5. ДБН В.1.2.7-2007 Основні вимоги до будівель та споруд.

6. ДБН В.2.5-28:2015 Природнє і штучне освітлення.

7. ДСанПіН 3.3.2.007-98 Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин.

8. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих.

9. НПАОП 0.00-1.28-10 Правила охорони праці під час експлуатації електронно- обчислювальних машин.

10. НПАОП 0.00-4.12-05 Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці.

11. НПАОП 0.00-4.15-98 Про розробку інструкцій з охорони праці.

12. НПАОП Б.02.005-2003 Про інструктаж, спецнавчання з питань пожежної безпеки.

13. НАПБ Б.03.001-2004 Типових норм належності вогнегасників

14. НАПБ Б.03.002-2007 Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою.

ВИСНОВОК

В даному проекті були проведені такі роботи як: розробка 3D моделей планет; моделювання симуляцій сонячної системи безпосередньо з використанням 3D об'єктів і спеціалізованих скриптів; розробка інтуїтивного 3D інтерфейсу. Так само вивчення матеріалів та інформацій про сонячну систему.

В процесі розробки проекту було проведено аналіз і принципи моделювання тривимірних об'єктів. Була проведена дослідницька робота і аналіз інформація про 3D моделювань, а саме: Основні принципи роботи в 3D Max; Аналіз моделювання тривимірних об'єктів. В даній роботі було спроектовано 17 об'єктів сферичної форми і дві багатокутна полігональні моделі.

1. Проведений аналіз принципу роботи векторів в 3D графіці, а саме в Unity3D
2. Основний напрямком роботи є побудова віртуальної моделі Сонячної Системи. Виходячи з даної постановки завдання була проведена така робота:
3. Проведено аналіз принципу роботи векторів в 3D графіці, а саме в Unity3D.
4. Аналіз основних операції над векторами.
5. Аналіз основних операції над кватериіонами.
6. Була складена таблиця числових даних для об'єктів з використанням астрономічних формул.
7. Побудована графічна модель розміщення об'ектів у віртуальному просторі, для полегшення устоновки об'єктів по векторах.
8. Написаний код для руху об'єктів по векторах.

Проведена робота над візуальному оформленні. Така як:

1. Розробка анімацій для об'єкта за допомогою компонента Practical System.
2. Налаштування освітлення за допомогою компонента Point Light.

Так же для программного проекта требовалась разработать користувальницький інтерфейс. Інтерфейс складається з таких етапів:

1. Перший етап є головне меню програми.
2. Другий етап є завантажувальним екраном між сценами програми.
3. Третій етап є інтерфейс який надає інформацію про об'єкти, і зміни параметрів руху об'єктів.

У перспективі даний проект буде у вільному доступі, кожен навчальний заклад зможе експлуатувати його в своїх цілях. Даний проект буде оновлюватися і привноситься нові функцій для поліпшення роботи цього додатка. Так само в перспективі даний проект буде експортований на Android і IOC.

Додаток А. Презентація дипломного проекту

Міністерство освіти і наукиУкраїни
Східноукраїнський Національний Університету
ім. В.Даля

ЗАСОБИ І АЛГОРИТМИ СТВОРЕННЯ ТРИВИМІРНИХ АСТРОНОМІЧНИХ ОБ'ЄКТІВ З ВИКОРИСТАННЯМ ЧИСЕЛЬНОГО МОДЕЛЮВАННЯ ОРБІТАЛЬНОЇ ДИНАМІКИ

Студент гр. КІ-146Д — Шаповалов О.О.

Керівник проекту — **Скарга-Бандурова І.С.**

1

Технічне завдання

Назва розробки

Навчальна програма для вивчення принципів роботи астрономічних об'єктів.

Мета розробки

Метою розробки є розробка віртуальної моделі та алгоритму руху об'єктів Сонячної системи, что может використовуват в якості навчального додатка на основі чисельних моделей орбітальної динаміки

Завдання на розробку

- Моделювання тривимірних об'єктів
- Побудова моделі сонячної системи
- Розробка користувальницького інтерфейсу

2

Огляд 3D редакторів для створення тривимірних об'єктів

Для створення комп'ютерної графіки використовують безліч різних додатків. Універсальні 3D редактори, як правило, містять все необхідне для CG: інструменти моделювання, анімації і візуалізації.

При виборі програми вимагаєте звернути увагу на наступні чинники: функціонал програми; зручність користування (інтуїтивний інтерфейс).

Було розглянуто чотири 3D редактора для створення тривимірних об'єктів.

- **3D MAX.**
- **Cinema 4D.**
- **Blender.**
- **Softimage.**

По кожному з цих редакторів було проведено аналіз їх характеристик і приципи розробки 3D моделей. З огляду на порівняльний аналіз для роботи був обраний редактор 3Ds Max. Оскільки даний проект розробляється на движку Unity 3D і даний редактор найбільш підходить для виконання роботи.

3

Редактор 3DS Max Studio

3DS Max займає провідні позиції в сфері дизайну та архітектурної візуалізації. Редактор розроблений для тривимірної графіки, моделювання та створення анімації. При його допомоги також створюють плоскі зображення - цифрові растрові картинки з однієї 3D сцени.

Можливості редактора

- моделювання на основі полігонів, сплайнів і NURBS,
- потужна система частинок,
- модуль волосся / шерсть,
- розширені шейдери Shader FX,
- підтримка нових і вдосконалених механізмів Iray і mental ray.
- інтеграція композітінга.

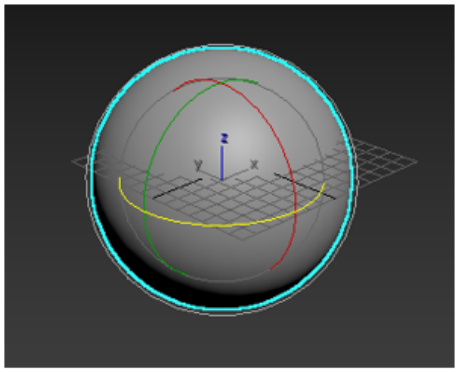
4

Проектування астрономічних 3D моделей

Для побудови Сонячної Системи було спроектуванно 17 об'єктів. Сфера-подібної форми потрібно було розробити 15 об'єктів, та 2 об'єкти з ламаними лініями (багатокутна полігональну модель).

Проектування об'єкта сфера-подібної форми здійснювалося за допомогою стандартних примітивів 3Ds Max Studio. І має такі налаштування для об'єкта:

- Radius (радіус) – 2500mm;
- Segments (сегменти) – 100 від цьогопараметра цього параметра залежить, наскільки об'єкт буде круглим).



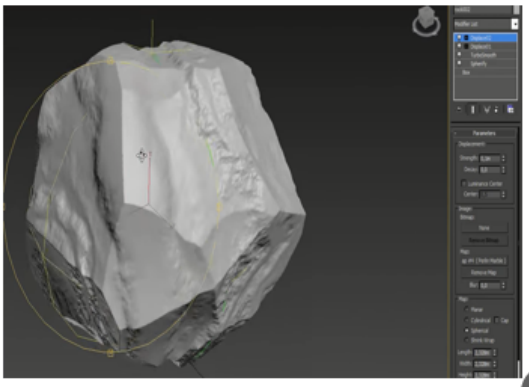
5

Проектування астрономічних 3D моделей

Побудова багатокутної полігональної моделі здійснювалося за допомогою стандартних примітивів типа (box). Для зміни тіла об'єкта потрібно підключити модифікатори, які закріплюються за об'єктом. Модифікатори мають ряд параметрів які впливають на зміни тіла об'єкта.

Модифікатори

- TurboSmooth – І виставляємо такі параметри: Iteration - 6 збільшуємо кількість полігонів на об'єкті; Render items - 6 тим самим витрата обчислювача комп'ютера саме задіяний в процесі рендеринга зменшується.
- Spherfiy – надає сферичну форму об'єкту.
- Desplace 1. Desplace 2. – який виступає інструментом для зміни форми тіла об'єкта. Segments 1 – 5. Segments 1– 7.

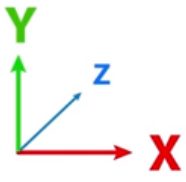


6

Застосування векторної математики для побудови моделі Сонячної Системи в Unity3D

Векторна математика в Unity3d ділитися на два типи: 2D вектор має дві осі (X, Y). Різні дії з об'єктами будуть відбувається тільки по двох осях (горизонталі і вертикалі); 3D вектор має три осі (X, Y, Z). Осі Z являє собою глибину. При додаванні осі Z, осі X і Y утворюють горизонтальну площину. ОсьY є висотою. Unity при роботі з об'єктами руководствуються системою координат, тобто розміщення і рух об'єктів буде відбуватися за координатами.

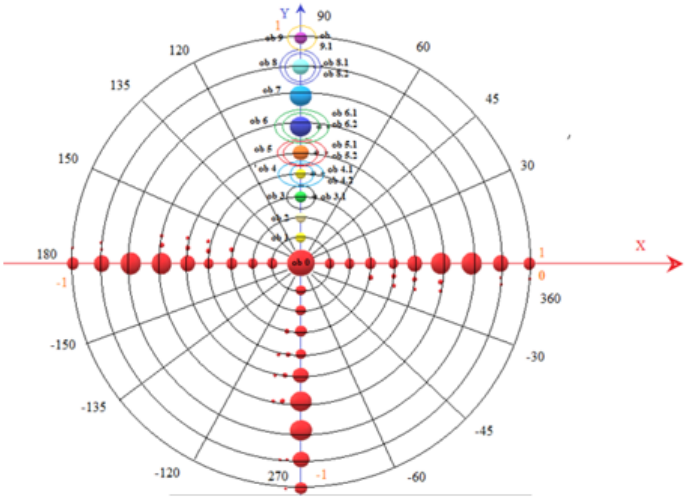
2D Vector



3D Vector

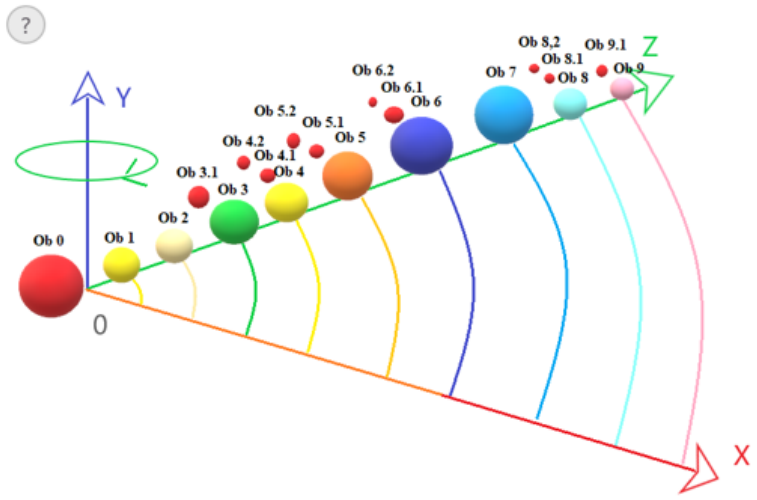
7

Схема віртуальної модель в 2 - х вимірному просторі



8

Схема віртуальної модель в 3 - х вимірному просторі



9

Розрахунки для об'єктів віртуальної моделі

Основні формули для чисельного моделювання орбітальної динаміки

Визначення відстані від нульового об'єкта

$$R = \sqrt[3]{T^2 * (1 + M)} = \sqrt[2]{T^2}$$

Орбітальний період об'єктів

$$\left(\frac{P}{2\pi}\right)^2 = \frac{a^3}{G \cdot (M + m)} \qquad P = 1\,год \cdot \sqrt{\frac{(a/1\,a.e.)^3}{M/M_{сон}}}$$

$$1/S = |1/P_1 - 1/P_2|$$

Визначення швидкості об'єктів

$$a_0 = \frac{v_I^2}{r} = g \qquad v_I^2 = gr \qquad v_I = \sqrt{gr} \qquad v_I = \sqrt{gR}$$

$$g = G\frac{M}{R^2} \qquad v_I = \sqrt{G\frac{M}{R^2} * R} = \sqrt{G\frac{M}{R}} \qquad v_I = \sqrt{G\frac{M}{R}}$$

$$g = G\frac{M}{(R+h)^2} \qquad v_I = \sqrt{G\frac{M}{(R+h)^2} * (R+h)} = \sqrt{G\frac{M}{R+h}} \qquad v_I = \sqrt{G\frac{M}{R+h}}$$

10

Розрахунки для об'єктів віртуальної моделі

За допомогою формул та аналізу інформацій науково-поплярних сайтів, наукових дослідний статей, літератури про астрономічні об'єкти, була сформована таблиця з такими даними як:

- Орбітальний період.
- Середня відстань від нульового об'єкта
- Середній еваторіальний радіус об'єкта
- Відстань від батьків до супутників
- Нахил осей обертання об'єктів

№ Об'єкта	Орбітальний період	Середня відстань від нульового об'єкта (мл.км).	Середній екваторіальний радіус об'єкта. (Км)	Відстань від батьків до супутників (км)	Нахил осей обертання об'єктів
Ob 0	25,38	0	695 500	0	0
Ob 1	87.969	57	2439	0	- 0.1
Ob 2	224,7	108	6051	0	177.36
Ob 3	365,26	149	6378	0	23.19
Ob 3.1	0	0	1723	384.467	1.5424
Ob 4	686.98	227	3397	0	25.19
Ob 4.1	0	0	1100	9380	1.093
Ob 4.2	1,26244	0	600	23.458	0.93
Ob 5	4332.589	778	71492	0	3.13
Ob 5.1	0	0	1821	422.000	0
Ob 5.2	0	0	2631	800.000	0
Ob 6	10759.22	1433	60268	0	26.73
Ob 6.1	0	0	531	295 000	1.12
Ob 6.2	0	0	1123	377.400	0
Ob 7	30685.4	2870	25559	0	97.77
Ob 8	60190.03	4491	24764	0	28.32
Ob 8.1	0	0	168	73.000	0
Ob 8.2	0	0	100	118.00	0
Ob 9	90553.02	5868	1151	0	119.61
Ob 9.1	0	0	100	19 640	112.78

11

Віртуальна модель руху об'єктів на мові C

Виходячи з розрахунків для астрономічних об'єктів, потрібно реалізувати роботу об'єктів за допомогою коду. Розробити алгоритм руху об'єктів навколо нульового об'єкта.

GameObject (name - є зміною) - створює осередки в інспектора об'єкта, на якому закріплений даний скрипт. В осередку поміщаються об'єкти для подальший експлуатація та роботи з цими об'єктами через даний скрипт.

public GameObject mercury;

public GameObject venus;

Період обертання об'єктів навколо власних осей.

public float coeff = 360.0f;

Швидкість рух об'єктів.

public float speed = 5;

Швидкість обертання

public float Sunspeed = 10;

12

Віртуальна модель руху об'єктів на мові C

Встановлюємо орбітальні періоди для об'єктів зі зверненням до періоду об'єкта 1.

```
period = 25.38f/baseMercuryPeriod;
periodVenus = 224.7f/baseMercuryPeriod;
```

Встановлюємо позицій для об'єктів по вектору (x) використовуючи клас Vector3.

```
transform.position = new Vector3 (0.0f,1.0f,0.0f);
mercury.transform.position = new Vector3 (57.9f,1.0f,0.0f);
```

Використовуючи клас Quaternion встановлюємо нахил осей обертання об'єктів.

```
mercury.transform.rotation = Quaternion.Euler(-1, 0, 0);
venus.transform.rotation = Quaternion.Euler(177, 0, 0);
```

Задаємо скаляр для об'єктів віртуальної моделі, тобто встановлюється розмір і обсяг об'єкта по 3 векторах.

```
transform.localScale = new Vector3(695000.500f, 695000.500f, 695000.500f);
mercury.transform.localScale = new Vector3(2.439f, 2.439f, 2.439f);
```

13

Віртуальна модель руху об'єктів на мові C

Алгоритм руху об'єктів.

Даний код вказує на обертання об'єкта по орбіті нульового об'єкта. Початкова значення вектора задається 0 за трьома координатами. Наступні значення вектора будуть змінюватися за координатами внаслідок руху об'єкта. Швидкість руху об'єкта ділитися на період обертання об'єкта. Тобто об'єкт повинен здійснити N кількість обертань і з заданою швидкістю по орбіті нульового об'єкта, і стати в початкову позицію. Також помножити на time.deltatime для плавного переміщення об'єкта.

earth.transform.RotateAround(Vector3.zero, Vector3.up, speed / periodEarth * Time.deltaTime);

Для того щоб об'єкт обертався навколо власної осі, вектор руху потрібно помножити на повний обротов об'єкта на 360 градусів.

earth.transform.Rotate(Vector3.up * coeff* Time.deltaTime);

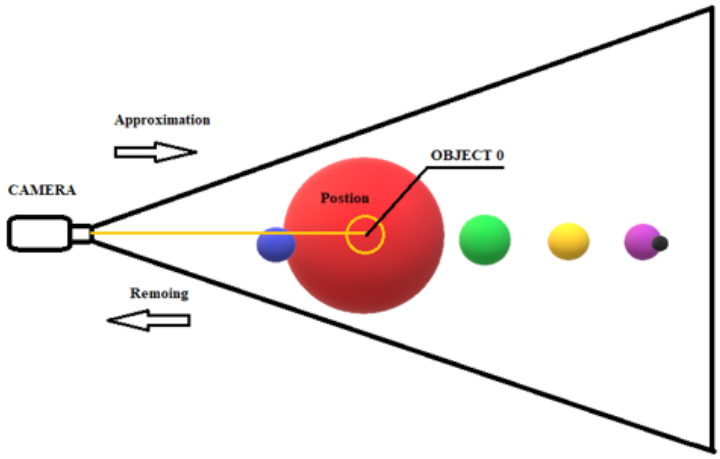
Обертання супутників залежать від періода обертання батьківських об'єктів.

moon.transform.RotateAround(earth.transform.position, Vector3.up, speed / periodEarth * Time.deltaTime);

14

Векторна робота з камерою

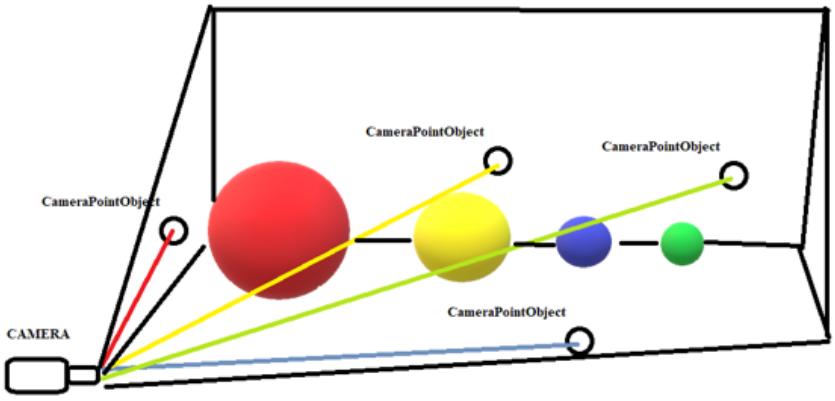
Прив'язка камери до нульового об'єкту



15

Векторна робота з камерою

Переміщення камери до дочірніх об'єктів по точках



16

Користувальницький інтерфейс

Стартове меню

17

Користувальницький інтерфейс

Інтерфейс основної сцени

18

ВИСНОВОК

Виходячи з поставлених завдань в даній роботі були досягнуті такі результати як: розробка 3D моделей планет; моделювання симуляцій сонячної системи безпосередньо з використанням 3D об'єктів і спеціалізованих скриптів; розробка інтуїтивного 3D інтерфейсу. Так само аналіз матеріалів та інформацій про сонячну систему.

19

ДЯКУЮ ЗА УВАГУ

20