МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ОНЛАЙН КУРСУ З ДИСЦИПЛІНИ «ПРОГРАМУВАННЯ КОМП'ЮТЕРНОЇ ГРАФІКИ У WEB»**

Виконав(ла): студент(ка) 2 курсу, групи 8.1219-з

спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

освітньої програми інженерія програмного забезпечення
(назва освітньої програми)

А.А. Ільченко
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.т.н., Мухін В.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної математики, доцент, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення

(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент
Лісняк А.О.

(підпис)

« » 2020 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ(СТУДЕНТЦІ)

Ільченко Антоніні Анатоліївній

(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Розробка онлайн курсу з дисципліни «Програмування комп'ютерної графіки у web»

керівник роботи (проекту) Мухін В.В., доцент кафедри програмної інженерії, к.т.н.

(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » травня 2020 року № 577-с

2. Строк подання студентом роботи 30.11.2020 року

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Використання засобів бібліотеки Three.js для візуалізації тривимірної графіки у веб та
Підбір теоретичного і методичного матеріалу для розробки онлайн курсу

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 20.05.2020 року

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	22.05.2020	Виконано
2.	Збір вихідних даних.	05.06.2020	Виконано
3.	Обробка методичних та теоретичних		
	джерел.	17.09.2020	Виконано
4.	Розробка першого розділу.	30.09.2020	Виконано
5.	Розробка другого розділу.	08.10.2020	Виконано
6.	Оформлення та нормоконтроль		
	кваліфікаційної роботи.	25.11.2019	Виконано
7.	Захист кваліфікаційної роботи.	09.12.2020	Виконано

Студент _____ А.А.Ільченко
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.В. Мухін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.В. Кудін
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка онлайн курсу з дисципліни «Програмування комп'ютерної графіки у web»: 52 с., 18 рис., 12 джерел, 2 додатки.

OPENGL, WEBGL, THREE.JS, HTML, JAVASCRIPT, ТАБЛИЦІ СТИЛІВ CSS, КОМП'ЮТЕРНА ГРАФІКА, АНІМАЦІЯ, AJAX, ОСВІТЛЕННЯ, ТЕКСТУРА.

Об'єкт дослідження – бібліотека Three.js.

Мета роботи: дослідити технології побудови тривимірної графіки у браузері WebGL використовуючи бібліотеку Three.js.

Метод дослідження – аналітичний, порівняльний, моделювання.

Тривимірна графіка вже давно стала невід'ємної частиною сучасного світу. Вона застосовується в усіх інформаційних сферах нашого життя, та навіть у наукових. Ріст популярності даного напрямку створює необхідність у нових фахівцях. Тому розробка онлайн курсу у сфері комп'ютерної графіки є актуальним як ніколи.

У першому розділі стисло викладені необхідні інструменті для розробки веб-сайту та розкрито можливості бібліотеки Three.js. У другому розділі змоделювало структуру онлайн курсу та описано технології, які були застосовані у ході розробки.

SUMMARY

Master’s Qualification Thesis «Development an Online Course in Computer Graphics Programming on the Web»: 52 pages, 18 figures, 12 references, 2 supplements.

OPENGL, WEBGL, THREE.JS, HTML, JAVASCRIPT, CASCADING STYLE SHITS, COMPUTER GRAPHICS, ANIMATION, AJAX, LIGHT, TEXTURE.

The object of the study is library Three.js.

The aim of the study is to explore the technology of building three-dimensional graphics in a WebGL browser using the Three.js library.

The methods of research are analytical, comparative, modeling.

3D graphics have long been an integral part of the modern world. It is used in all information spheres of our lives, and even in science. The growing popularity of this area creates a need for new professionals. Therefore, the development of an online course in computer graphics is more important than ever.

The first section summarizes the necessary tools for website development and reveals the possibilities of the Three.js library. The second section simulates the structure of the online course and describes the technologies that were used during the development.

ЗМІСТ

ВСТУП

Кваліфікаційна робота магістра присвячена розробці онлайн курсу з дисципліни «Програмування комп'ютерної графіки у web» метою якого є розкрити можливості 3D-моделювання за допомогою WebGL, опираючись на бібліотеку Three.js.

Віртуальна реальність та 3D-моделювання стрімко набирає популярність у сучасному світі, з'являються нові напрямки професійної діяльності на базі засобів комп'ютерної графіки. У даній сфері активно відбувається розвиток технологій та накопичення методичного матеріалу.

Актуальність теми кваліфікаційної роботи.

Тема комп'ютерної графіки актуальна як ніколи раніше. Технології віртуальної реальності та 3D-моделювання стали провідним вектором розвитку інформаційних систем усіх галузей застосування.

Цей напрямок уже широко використовується для навчання та стимуляційного тренування воєнно-промисловими комплексами, для інженерного проектування, у галузі охорони здоров'я, рекламі та маркетингу, архітектурі та дизайну інтер'єру, у сфері комп'ютерних ігор та звичайно у кінематографі [1]. Отже, можливості комп'ютерної графіки є дуже різноманітними та перспективними.

Оскільки у інформаційному світі розвиток комп'ютерної графіки породжує нові робочі місця, які користуються великим попитом, навчання та підготовка кадрів в цій сфері є одним із найважливіших аспектів.

З вище сказаного, можна зробити висновок, що тема магістерської роботи є актуальною.

Мета кваліфікаційної роботи.

Метою роботи є розробка онлайн курсу для вступу до комп'ютерної графіки використовуючи технології побудови тривимірної графіки у браузері WebGL.

Завдання кваліфікаційної роботи.

Завдання, які необхідно вирішити для досягнення поставленої мети:

– розширити та поглибити уміння та навичками створення веб-сайтів засобами HTML, CSS, Java Script, JQuery, Bootsrap;

– вивчити основні засоби бібліотеки Three.js для створення тривимірних геометричних об’єктів, анімації та елементів інтерактивності;

– оволодіти методичними навичками у процесі створення структури онлайн курсу;

Об’єкт дослідження – бібліотека Three.js та 3D візуалізація.

Предмет дослідження – створення інтерактивного онлайн курсу з вивчення комп’ютерної графіки у веб.

1 ОГЛЯД ЗАСОБІВ ДЛЯ РОЗРОБКИ САЙТУ ТА 3D ГРАФІКИ

1.1 Опис основних інструментів для створення веб-сайту курсу з дисципліни «Комп'ютерна графіка у веб»

1.1.1 HTML – мова розмітки веб-сторінок

HTML (від англійського HyperText Markup Language) – це мова гіпертекстової розмітки сторінки. Він використовується для того, щоб дати браузеру зрозуміти, як потрібно відображати завантажений сайт.

Структури HTML-сторінки виглядає так:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Назва сторінки</title>
         …
            (підключення стилів та скриптів)
         …
   </head>
      <body> (тіло сторінки)
            <div>
                  <h1></h2>
                  <p></p>
                  …
                  <img></img>
                  … та інші.
</div>
      </body>
</html>
```

HTML5 є останньою зміненої специфікацією HTML, яка надає деякі додаткові мітки і функції (кросбраузерність, відео, аудіо, анімація і багато іншого), які здатні дати користувачеві більше можливостей для використання різних передових технологій.

Переваги та недоліки HTML5.

Переваги:

1) на відміно від SVG, значно зручніше володіти великою кількістю елементів;
2) містить апаратне прискорення;
3) можливість використання фільтрів обробки зображення;
4) широкий вибір бібліотек.

Недоліки:

1) навантаження на мікропроцесор і оперативну пам'ять;
2) низька продуктивність при великому розширенні;
3) немає можливості очистки пам’яті.

Мова HTML складається з тегів – це своєрідні команди, які перетворюються у візуальні об'єкти в браузері користувача. Наприклад: <teg>. Майже кожен тег парний і вимагає закриваючого тега </teg>. Не правильне використання тегів призводить до проблем з розміткою сайтів.

Нижче пропонується переглянути список тегів з поясненням, які були використані при розробці онлайн курсу з дисципліни «Програмування комп’ютерної графіки у веб» [9].

-
 – перехід на новій рядок;
- <embed> – тег-контейнер для вставки зовнішнього інтерактивного контенту;
- <img> – вставка зображення за допомогою атрибута scr;
- <figure></figure> – тег-контейнер для ілюстрацій;
- <figcaption></figcaption> – заголовок або підпис для елемента <figure>;

- – тег-контейнер для розділів сторінки, використовують для групування;
- <h1-6></ h1-6> – заголовки різних рівнів;
- <p><p> – параграфи в тексті;
- – надає напівжирний шрифт тексту;
- ul> – маркований список;
- – нумерований список;
- – елементи маркованого та нумерованого списків;
- <table></table> – тег для створення таблиці. <tbody> – тіло таблиці, <th> – заголовки колонок таблиці, <td> – створює стовбці таблиці;<tr> – створює строки таблиці;
- <a> – створення гіпертекстового посилання;
- <textarea> </textarea> – створення поля для вводу тексту;
- <video><video> – вставка відео (формати mp4, WebM,Ogg);
- <button></button> – створення інтерактивної кнопки.

1.1.2 Таблиці стилів CSS та BootStrap

CSS це мова стилів, що визначає відображення HTML-документів. Наприклад, CSS працює з шрифтами, кольорами, полями, рядками, висота, ширина зображень, позиціонуванням елементів.

Застосування CSS до HTML-сторінки можна декількома способами:

1) Inline – за допомогою атрибуту <style> безпосередньо до обраного елементу;

2) внутрішній тег <style></style> безпосередньо у розділі HEAD;

3) зовнішній (посилання на таблицю стилів – файл з розширенням .css) <link rel="stylesheet" type="text/css" href="css/StyleText.css"/>.

При роботі з великими проектами краще використовувати 3-й спосіб. Це покращить роботу с основним кодом сторінки і є ознакою хорошого коду.

Bootstrap – це CSS/HTML-фреймворк для створення сайтів. Іншими словами, це набір інструментів для розмітки сайту. У ньому є ряд переваг, завдяки яким Bootstrap вважається найпопулярнішим серед собі подібних, а саме:

1) швидкість роботи – завдяки великій кількості готових елементів, розмітка сайту займає значно менше часу;
2) масштабованість – додавання нових елементів не порушує загальну структуру документів;
3) велика кількість шаблонів, які допомагають редагувати вже змінені елементи під свої потреби;
4) широка сфера застосування – користується популярністю як для створення одностроріночкових так і лендингів (landing page) веб-сторінок [11].

1.1.3 JavaScript та JQuery

JavaScript – мова програмування, що дозволяє реалізувати ряд складних рішень в веб-документах. Вона допомагає зробити сторінки сайту більш інтерактивними, обробляє дії користувачів сайту. Це об'єктно-орієнтована клієнтська мова, яка підтримується додатками, що працюють з дизайном сайту. JavaScript став ще більш популярним в серед програмістів, коли з'явилася AJAX-технологія, що призвело до нового етапу в розробці сайтів.

AJAX (Asynchronous JavaScript And XML) – це підхід до побудови користувацьких інтерфейсів веб-застосунків, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані [10].

Прикладі застосування AJAX-технології:

1) відображення контенту, який періодично оновлюється (інтерактивні карти);
2) створення якісної анімації і графічних об'єктів у форматі 2D/3D;

3) опція прокрутки відеозапису в медіа програвачі.

Поряд з HTML і CSS, JavaScript – третій важливий блок, на основі якого будується більшість стандартних веб-інтерфейсів.

Щоб самостійно розібратися з роботою мови, користувачеві знадобиться знання основ HTML і CSS.

Способи підключення JavaScript на веб-сторінці:

1) за допомогою тегів <script><script> (для версії HTML 4 обов'язково з атрибутом type="text/javascript");

2) підключення з іншого сайту

<script src="http://exzample.com.ua/js/ajax.js"></script>;

3) зовнішнє підключення файлу з розширенням .js

<script src="шлях та ім'я файлу.js"></script>.

JQuery – це бібліотека, що сформована на основі мови програмування JavaScript.

Для підключення бібліотек JQuery потрібно, перш за все, завантажити файли, та підключити на сторінку уже вивантажений код:

<script type="text/javascript" src="jquery.js"></script>

Завдяки тому, що обсяг програмного коду JQuery менше, ніж обсяг стандартного коду Javascript, скорочуються час розробки елементів веб-сторінки. Сам програмний код більш зрозумілий у порівнянні з Javascript.

Однією із основних переваг JQuery – це кросплатформеність і кросбраузерність. Завдяки цьому, немає необхідність піклуватися про синтаксис і особливості взаємодії різних браузерів і операційних систем з кодом.

1.2 Технологія побудови тривимірної графіки у браузері WebGL

1.2.1 Система координат WebGL та моделювання графіки з використання Three.js

З розвитком інформаційних технологій веб-розробка стала потребувати 3D можливостей. WebGL (Web-based Graphics Library) – це програмна бібліотека, яка призначена для створення інтерактивної тривимірної графіки в браузері.

За рахунок використання низькорівневих засобів підтримки OpenGL частина коду на WebGL може виконуватися безпосередньо на відеокартах, що дає виграш по швидкодкості.

Побудова графіки відбувається на об'єкті canvas. Відображення в WebGL здійснюється за допомогою так званих вершинних і піксельних рейдерів [4].

Шейдери – це функції, написані на спеціальній мові програмування GLSL, які виконуються графічною картою і обробляють дані вершин і пікселів, визначаючи остаточні параметри зображення об'єкта. Вони можуть включати в себе опис поглинання і розсіяння світла, накладення текстури, віддзеркалення і переломлення, затінення, і т.д.

Робота з шейдерами – це досить трудомісткий процес. До всього іншого, потрібно ще описати кожну вершину, кожне ребро, кожну грань, нормалі до поверхні, їх колір, положення та ін. Для підвищення швидкості розробки можна використовувати один з декількох фреймворків WebGL - Sylvester, glUtils.js, webgl-utils.js, Three.js, та інші [2].

У даній кваліфікаційній роботі розглянуто докладніше бібліотеку з відкритим вихідним кодом Three.js.

Three.js – це бібліотека JavaScript, що містить набір готових класів для створення і відображення інтерактивної комп'ютерної 3D графіки в WebGL.

Бібліотека Three.js полегшує роботу з WebGL. При використанні Three.js відпадає необхідність в написанні шейдерних процедур і з'являється можливість оперувати з більш звичними і зручними поняттями, такими як: сцени, світла і камери, об'єктами та їх матеріалами.

Бібліотека Three.js також підтримує відображення готових тривимірних моделей формату Collada (це відкритий стандарт файлів для інтерактивних 3D додатків, базується на форматі XML) [11].

Для використання бібліотеки потрібно завантажити з сайту threejs.org файл three.js і підключити до вашого проекту.

Роботу з 3D графікою WebGL за допомогою Three.js можна умовно розбити на наступні етапи:

- додавання сцени;
- додавання камери;
- додавання світла (освітлення);
- додавання графічних об'єктів на сцену;
- створення об'єкта візуалізації;
- рендеринг (візуалізація);
- анімація (рух об'єктів, їх взаємодія) [2].

Для створення сцени оголошуємо глобальну змінну

var scene = newTHREE.Scene ();

Для додавання об'єктів на сцену або видалення їх зі сцени використовуються методи add і remove:

scene.add (object);

scene.remove (object);

Наступним кроком є додавання камера, яка є спостерігачем за нашою сценою. Найпростішим типом камери - є перспективна камера [8].

var camera;

camera = newTHREE.PerspectiveCamera (45,window.innerWidth / window.innerHeight, 1, 10000);

де 45 – це кут огляду;

window.innerWidth / window.innerHeight – пропорція співвідношення сторі;

1 – мінімальна, а 10000 – максимальна відстань від камери, яка потрапляє в рендеринг [6].

Також ми можемо вказати позицію камери camera.position.set(0, -20, 100); та напрямок огляду camera.lookAt(newTHREE.Vector3(10, -100, 100));

Перспективна камера не дає можливості оглянути наш предмет з усіх ракурсів, для можливості керування оглядом на сцену треба скачати та встановити TrackballControls.js (або будь-який інший контрол).

Підлючення: <script type = "text/javascript" src = "js/ TrackballControls.js"> </script>

Тепер можемо підключити наш TrackballControls для керування оглядом.

var controls;

controls = new THREE.TrackballControls(camera, container);

Система координат в WebGL.

Для розташування фігур в просторі використовується декартова система координат (див. рис. 1.1):

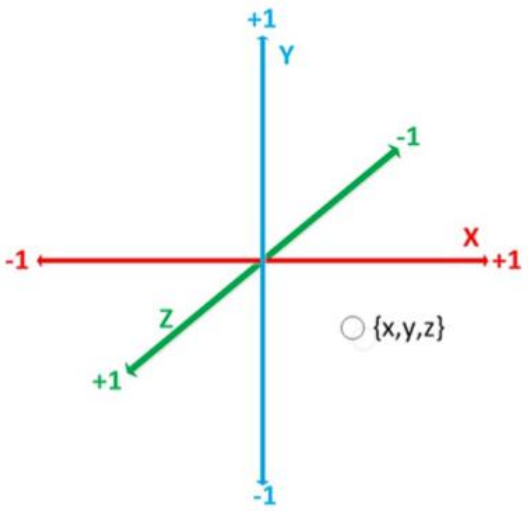


Рисунок 1.1 – Декартова система координат WebGL

Як завжди, для програм, що працюють з 3D, угору направлена вісь Y. На малюнку також вказані правила зміни знаків. Наприклад, при наближенні до нас значення координати Z збільшується.

При додаванні об'єкту на сцену в Three.js можна вказати його координати:

```
object.position.x = -50;
object.position.y = 20;
object.position.z = 60;
```

або одним рядком: object.position.set (-50, 20, 60);

При цьому в зазначеній точці зазвичай розташовується геометричний центр тіла. Якщо ж координати об'єкта не вказані, то вони всі рівні нулю.

Для того, щоб задати кут повороту тіла використовується властивість rotation. Кути вказуються в радіанах.

```
object.rotation.y = Math.PI / 2;
```

Синтаксис звернення до rotation такий самий, як і у position [2].

Одним із основних елементів для огляду є світло. Щоб додати на сцену освітленні слід виконати наступний код:

```
var light = new THREE.DirectionalLight( 0xffffff );
light.position.set( 0, 100, 100 );
scene.add( light );
```

Для відображення сцени і її об'єктів за допомогою WebGL, в Three.js використовується спеціальний клас WebGLRenderer. Оголосимо змінні для екземпляра цього класу і для DOM-елемента, з яким цей клас буде працювати:

```
var renderer, container;
```

Разом з класом створюється полотно canvas, який по замовчуванням має ширину 300 пікселів і висоту 150 пікселів. Метод setSize дозволяє змінити його розміри. Наприклад, створимо об'єкт візуалізатор, вказавши розміри полотна «на все вікно»:

```
renderer = new THREE.WebGLRenderer ();
```

renderer.setSize (window.innerWidth, window.innerHeight);

renderer.setClearColor (0xffffff); – колір фону.

Далі слід вказати візуалізатором renderer, де саме буде створюватися полотно. Ми приготували для цього розділ MyWebGLApp на головній сторінці:

container = document.getElementById ('MyWebGLApp');

container.appendChild (renderer.domElement);

Тепер полотно для малювання розміщено в нашому розділі на головній сторінці. Для додання моделям більш реалістичного виду в 3D графіку застосовують спотворення. У Three.js допускається використовувати спотворення за допомогою параметра antialias.

Візуалізація проводиться за допомогою створеного у попередньому пункті об'єкта renderer.

renderer.render (scene, camera);

Анімація є, по суті, послідовний неодноразовий рендеринг сцени (для відображення динаміки). Таке послідовне відображення здійснюється спеціальною функцією requestAnimationFrame:

function animate (){

requestAnimationFrame (animate);

render ();}

1.2.2 Робота з матеріалами та їх властивості. Текстура

При створенні матеріалу майбутньої заготовки можна вказати колір або текстуру, прозорість. Колір має формат 0xHEX, де HEX

var material = new THREE.MeshBasicMaterial ({color: 0x33CCFF, transparent: true, opacity: 0.6});

Матеріали бувають різних видів [7]. Наприклад, це:

1) MeshBasicMaterial – для зафарбовування поверхонь фігур однорідним кольором (див. рис. 1.2, а);

2) MeshLambertMaterial – для градієнтної заливки. Місця, на які падає світло, зображуються більш світлими(див. рис. 1.2, б);

3) MeshPhongMaterial – для блискучих поверхонь. Вимогливий до ресурсів (див. рис. 1.2, в);

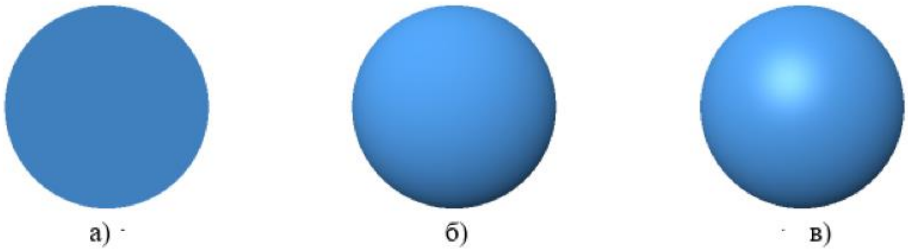


Рисунок 1.2 – Приклад використання матеріалу на сфері,
де а) – MeshBasicMaterial, б) – MeshLambertMaterial, в) – MeshPhongMaterial

4) MeshNormalMaterial – дозволяє підкреслити «тривимірність» об'єкта, розфарбовуючи його межі в різні кольори (див. рис. 1.3);

5) LineBasicMaterial – матеріал для малювання каркасів;

6) LineDashedMaterial – матеріал для малювання пунктирних каркасів. Можна вказати параметри dashSize – довжину пунктиру, і gapSize – довжину розриву (відстань між пунктиром).

Параметр side регулює видимість матеріалу сторін на двосторонніх моделях:

- THREE.FrontSide – видно зовні (у напрямку нормалей) – за замовчуванням;
- THREE.BackSide – видно зсередини;
- THREE.DoubleSide – видно з обох сторін.

Для накладення текстур в Three.js є спеціальний методом loadTextureю.load(), головним параметром якого є адреса (шлях) до зображення текстури.

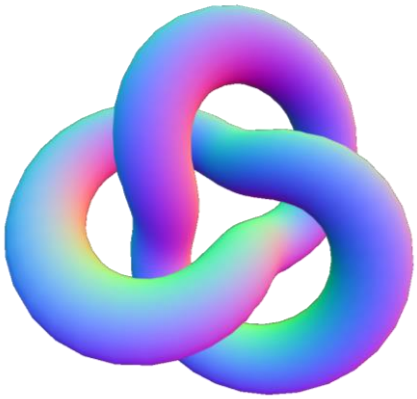


Рисунок 1.3 – Матеріал для спеціального використання MeshNormalMaterial

На основі текстури Texture створюється матеріал з параметром map: Texture. Далі – все як зазвичай, створюється фігура із заданою геометрією і нашим матеріалом.

Для того, щоб текстури нормально відображалися, необхідно, щоб звернення до веб-сторінок відбувалося через сервер (це зв’язано з політикою CORS – Cross-Origin Resource Sharing). Докладніше про налаштування серверу можна ознайомитися у другому розділі даної кваліфікаційної роботи.

Якщо потрібно накласти на різні боки різні текстури (див. рис. 1.4), то потрібно створити масив матеріалів і заповнити його в циклі:

```
var materials = [];
for (i = 1; i <= 6; i ++)
{ var Texture = new THREE. loadTextureю.load() ('Textures /' + String (i) + '.png');
var Material = new THREE.MeshBasicMaterial ({Map: Texture, color: 0x00dcff});
materials.push(Material);}
```

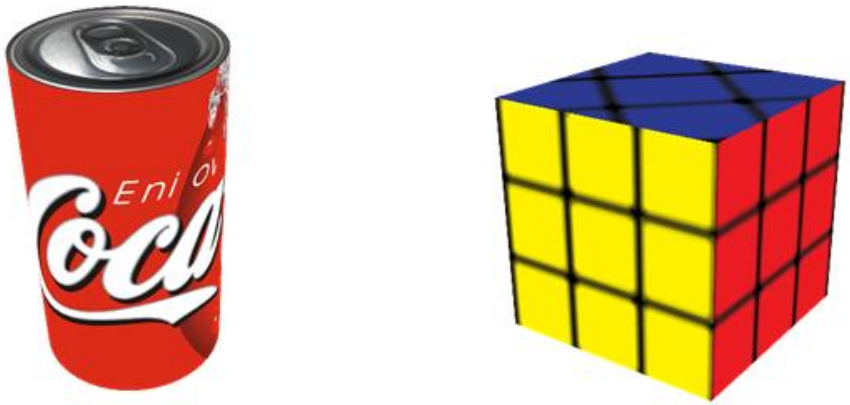


Рисунок 1.4 – Приклад з використання різної текстури
на різні грані об’єкту.

Де а) – циліндр з текстурою обгортки Coca Cola та б) – куб з текстурою кубика Рубика.

1.2.3 Джерела світла та тіні

Без освітлення на сцені, буде складатися враження, що ви знаходитеся в темній кімнаті. Крім цього, за допомогою освітлення сцені можна надати об’єктам більшу реалістичність. Також, кожному світлу можна задати колір. Розглянемо основні класи освітлення бібліотеки Three.js.

Клас AmbientLight представляє собою загальне освітлення, яке застосовується до всіх об'єктів сцени. Воно не має напрямку і зачіпає кожен об'єкт сцени в рівній мірі, незалежно від розташування об'єкта (див. рисунок 1.5, а). Відповідно, у цього світла немає позиції на осі координат. Це джерело світла не може використовуватися для створення тіней [3].

Конструктор класу:

AmbientLight (color, intensity);

де color – числове значення RGB-компонента кольору, за замовчуванням = 0xFFFFFF (білий колір);

intensity – числове значення інтенсивності світла, за замовчуванням зазвичай = 1 (для всіх видів освітлення);

Наприклад:

var light = new THREE.AmbientLight (0x404040);

scene.add (light);

Наступний клас – DirectionalLight (див. рис. 1.5, б). Це джерело прямого спрямованого освітлення, його можна розглядати як джерело світла, що знаходиться дуже далеко. Усі промені, які він випромінює, паралельні один одному. Хорошим прикладом цього є сонце. Вся область, що освітлена джерелом DirectionalLight, отримує однакову інтенсивність світла.

Найбільш загальний вигляд конструктора класу має вигляд: DirectionalLight (color, intensity, distance);

де distance – дорівнює відстані від джерела світла, на якій інтенсивність світла стане рівною нулю.

Наприклад:

var directionalLight = new THREE.DirectionalLight (0xffffff, 0.5);

scene.add (directionalLight);

PointLight – це клас освітлення, який представляє собою точкове джерело світла (див. рис. 1.5, в).. Його можна порівняти з лампочкою. Конструктор класу має аналогічний з DirectionalLight вигляд:

PointLight (color, intensity, distance);

var light = new THREE.PointLight (0xff0000, 1, 100);

light.position.set (50, 50, 50);

scene.add (light);

Клас SpotLight представляє собою прожектор, так зване світло з ефектом конуса (див. рис. 1.5, г). Конструктор класу: SpotLight (color, intensity, distance, angle, exponent);

де angle – кут, який визначає величину променю від джерела світла. Вимірюється в радіанах, за замовчуванням = Math.PI / 3;

exponent – це показник зміни інтенсивністі світла.

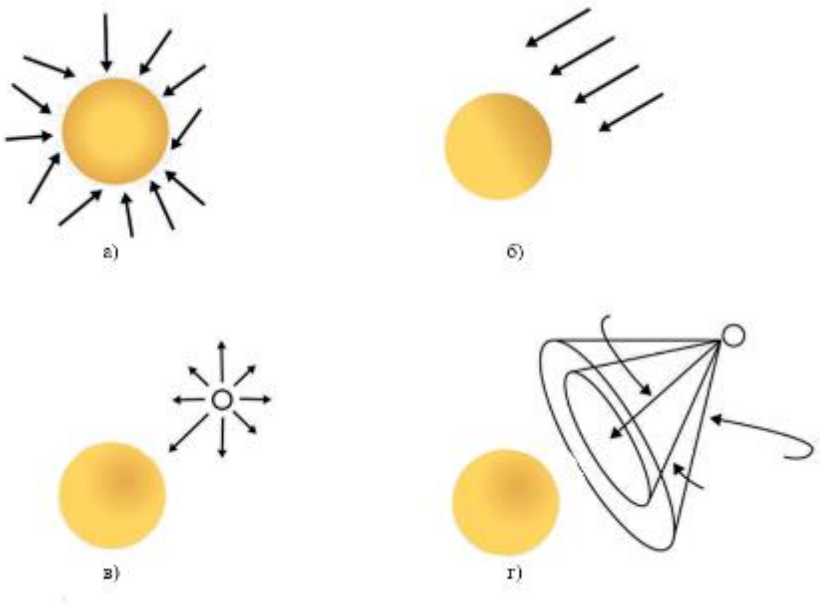


Рисунок 1.5 – Приклади різного освітлення, де а) – Ambient Light, б) – Directional Light, в) – Point Light, г) – Spot Light

Наприклад:

var spotLight = new THREE.SpotLight (0xffffff);

spotLight.position.set (100, 1000, 100);

scene.add (spotLight).

На практиці найчастіше використовують клас освітлювання SpotLight для відображення тіней [12].

Для накладення тіней на сцені потрібно попередньо «ввімкнути» їх в рендері:

renderer.shadowMapEnabled = true;

Далі потрібно ввімкнути освітлення тіней у джерелі світла:

light.castShadow = true;

Цю ж властивість castShadow потрібно ввімкнути для тих тіл та об’єктів, від яких потрібно відкидання тіней.

1.2.4 Робота з геометрією

Перед додаванням об'єктів потрібно створити сцену, додати світло, рендерер, камеру, і так далі.

Додавання об'єктів проводиться таким чином. Спочатку оголошується вид геометрії об'єкту. У найпростіших випадках це може бути паралелепіпед, сфера, циліндр тощо, параметри геометрії включають зазвичай лінійні розміри і кількість сегментів, що відповідає за точність зображення [5].

Наприклад, для прямокутного паралелепіпеда (кубоід) передбачений спеціальний клас BoxGeometry:

var geometry = new THREE.BoxGeometry(200, 300, 50);

Вказується ширина кубоїда (вздовж осі X), висота (вздовж осі Y) і довжина (вздовж осі Z).

Наступним кроком вказується матеріал майбутнього об'єкта:

var material = new THREE.MeshBasicMaterial ({color:0x00ff00});

var Cube = newTHREE.Mesh (geometry, material);

Залишилося додати об'єкт на сцену. Можна вказати позицію, використовуючи властивість position, і поворот щодо осей координат – rotation:

Cube.position.z = -100;

Cube.rotation.z = Math.PI / 6;

scene.add (Cube);

Створимо, наприклад, куля синього кольору. Послідовно створюємо геометрію, матеріал і об'єкт з вибраними геометрією і матеріалом. Робиться це через клас Mesh – сітку майбутнього об'єкта:

var geometry = new THREE.SphereGeometry (100, 50, 50);

var material = new THREE.MeshLambertMaterial ({Color: 0x33CCFF});

var Sphere = new THREE.Mesh (geometry, material);

scene.add (Sphere); (див. рис. 1.6).

Перший параметр SphereGeometry – радіус сфери; другий і третій параметри – кількість сегментів по ширині і висоті сфери. Чим їх більше, тим точніше зображується сфера, на рисунку 1.7 зображена сфера з невеликою кількістю сегментів, а сама 12 сегментів. Як матеріал вказано градієнтну заливку.

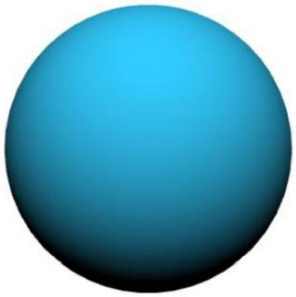


Рисунок 1.6 – Сфера з градієнтною заливкою

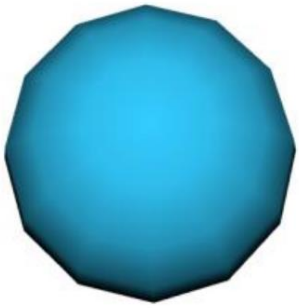


Рисунок 1.7 – Сфера з малою кількістю сегментів
(12 сегментів)

Також клас Mesh має метод scale, якиє дозволяє розтягувати геометрію вздовж зазначеної осі.

Наприклад:

Sphere.scale.x = 1.5;

і отримаємо еліпсоїд (див. рис. 1.8).

Для створення піраміди, призми, циліндра і конуса необхідно скористатися геометрією и CylinderGeometry.

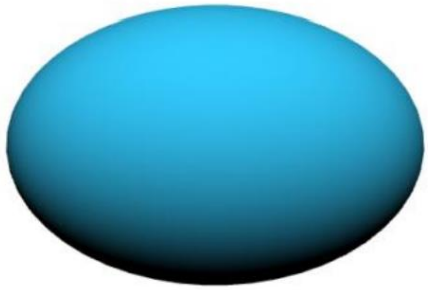


Рисунок 1.8 – Витягнута сфера (еліпсоїд)

Наприклад:

var geometry = new THREE.CylinderGeometry (radius_top, radius_bottom, heigth, segments);

де radius_top та radius_bottom – це радіуси верхньої та нижньої основи відповідно, heigth – висота фігури; segments – кількість сегментів, яка дорівнює кількості сторін.

Для піраміди геометрія має віглядати наступним чином:

var piramida = new THREE.CylinderGeometry (0, 120, 130, 3);

Для призми, радіуси основ не мають долівньвати 0:

var prizma = new THREE.CylinderGeometry (120, 120, 130, 3);

Для циліндру, необхідні однакові радіуси основ та досить велика кількість сегментів, для точного зображення:

var cylinder = new THREE.CylinderGeometry (120, 120, 130, 60);

Для конусу, радіус верхньої (або нижньох) основи має дорівнювати 0, та велика кількість сегментів:

var conus = new THREE.CylinderGeometry (0, 120, 130, 60);

1.2.5 Створення структурних об’єктів

Іноді доводиться створювати складну фігуру з декількох простих об'єктів. Тоді, по ідеї, щоб вони рухалися як єдине ціле, для вирішення цієї проблеми можна використовувати клас тривимірних об'єктів Object3D. При створенні складної фігури все її «Запчастини» додаються не на сцену, а «всередину» об'єкта, і лише потім об'єкт виводиться на сцену.

Наприклад: для початку оголосимо глобальний об'єкт:

var Figure = newTHREE.Object3D ();

Усі елементі фігурі додаємо спочатку до об’єкту

Figure.add(part1);

Figure.add(part2); і так далі.

Після того як всі елементи були додані до нашого об’єкту, можна і його додавати до сцені

Scena.add (Figure);

Рисунок 1.9 – Шахова фігура пішака

На рисунку 1.9 за допомогою класу Object3D створено шахову фігуру пішака.

Для основи з вигином використано клас LatheGeometry, який створює поверхню обертання.

Конструктор класу:

LatheGeometry (points, segments, phiStart, phiLength);

де points – це множина точок для побудови кривої, що обертається, segments – кількість сегментів радіальної окружності (за замовчуванням дорівнює 12), також можна вказати початковий кут – phiStart і величину кута обертання – phiLength [2].

За основу візьмемо функцію (1.1), що зображена на графіку (див. рис. 1.10).

$$f = 10 + 20 * e^{-x^2} \tag{1.1}$$

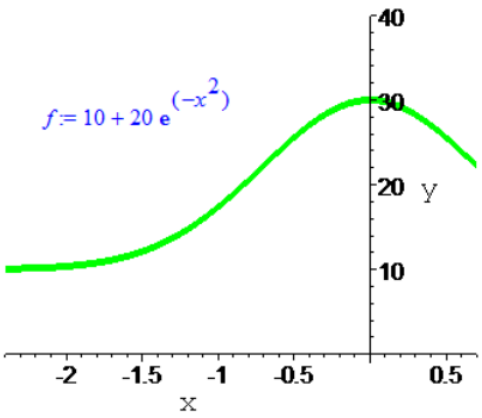


Рисунок 1.10 – Графік функції для основи шахової фігури пішака

У Додатку Б можна докладніше ознайомитися з кодом для шахових фігур (пішак, слон, тура, король та королева).

1.2.6 3D текст

Для створення тривимірного тексту призначений клас TextGeometry (спадкоємець класу ExtrudeGeometry). Конструктор:TextGeometry (text, parameters); text – вміст тексту, параметри parameters включають:

- size: розмір тексту;
- height: товщина тексту;
- curveSegments: кількість точок (сегментів) кривої при малюванні букви;
- font: назва шрифту;
- weight: тип шрифту ("normal", "bold");
- style: стиль шрифту ("normal", "italics");

Далі, можна включити фаску (тобто додати «скоси»):

- bevelEnabled: включення фаски (при true);
- bevelThickness: глибина фаски;
- bevelSize: ширина фаски.

Тривимірний текст виходить з плоского «екструзією» («Видавлюванням») на величину height. Можна використовувати різні матеріали для передньої (задньої) частини тексту і отриманої екструзією бічній частині. Потрібно створити масив матеріалів і вказати:

- material: індекс матеріалу для переднього і заднього планів;
- extrudeMaterial: індекс матеріалу для плану екструзії і плану фаски.

При цьому обов'язково потрібно завантажити і підключити скрипт, генерує шрифти. На сайті threejs.org це, наприклад, файл helvetiker_regular.typeface.js

Підключення шрифту <script src = "../ fonts / arial_regular.typeface.js"> </script>

Тепер можна працювати з текстом зі шрифтом arial, Задамо текст і його геометрію:

```
var text = "TEXT";
vartext_geometry = newTHREE.TextGeometry (text,
{  size: 24,
   height: 5,
   curveSegments: 4,
   font: "arial",
   style: "normal",
   bevelEnabled: true,
   bevelThickness: 2,
   bevelSize: 1,
});
```

Підготуємо матеріал лілового кольору:

```
var text_Material = new THREE.MeshPhongMaterial ({color: 0x62254a});
```

Тепер створимо, власне, тривимірний об'єкт – «меш» з збудованими геометрією і матеріалом:

```
var text3D = new THREE.Mesh (text_geometry, text_Material);
```

Рисунок 1.11 – 3D-текст з текстурою

При рендеринге текст вибудовується зліва направо від зазначеної позиції. Якщо хочеться помістити текст по центру екрана, потрібно знайти його середину. У цьому нам допоможе метод computeBoundingBox(), який здатний обчислити кордони геометрії. Отже,

```
text_geometry.computeBoundingBox ();
var text_Width = text_geometry.boundingBox.max.x -
text_geometry.boundingBox.min.x;
text3D.position.set (-0.5 * text_Width, 0, 0);
scene.add (text3D);
```

Для накладання текстури треба завантажити будь яке зображення та розмножити його (див. рис. 1.11):

```
var Texture = THREE.ImageUtils.loadTexture( 'textures/img.jpg' );
Texture.wrapS = Texture.wrapT = THREE.RepeatWrapping;
Texture.repeat.set( 0.05, 0.05 );
var text_Material = new THREE.MeshBasicMaterial( { map: Texture } );
```

2 РОЗРОБКА ОНЛАЙН КУРСУ

2.1 Установка програмного забезпечення та налаштування робочого середовища

У ході практичної частими кваліфікаційної роботи, було розроблено онлайн курс з дисципліни «Програмування комп'ютерної графіки у веб»

Для запуску курсу на робочій машині необхідно налаштувати робоче місце. Почнемо з інсталяції Node.js (завантажити актуальну версію із офіційного сайту: https://nodejs.org/en/).

Для перевірки інсталяції, введіть node --version у командній строці, якщо Node.js встановлено, то відобразиться актуальна версія, як що ж ні, спробуйте ще раз пройти інсталяцію.

Для роботи сайту встановіть jquery

npm install jquery

З міркувань безпеки WebGL не може безпосередньо використовувати зображення з вашого жорсткого диска. Це означає, що для розробки та запуску курсу необхідно використовувати веб-сервер.

Встановимо сервер наступною командою:

install http-server

Далі перейдіть в папку з файлом під назвою index.html, та запустіть сервер. На рисунку 2.1 показано успішний запуск серверу на Windows.

http-server --cors .

Рисунок 2.1 – Успішний запуск серверу на комп'ютері

2.2 Структура онлайн курсу

Онлайн курс складається з головної веб-сторінки index.html (див. рис. 2.2), яка містить короткий опису курсу, та лекції, які можна назвати робочою програмою курсу.

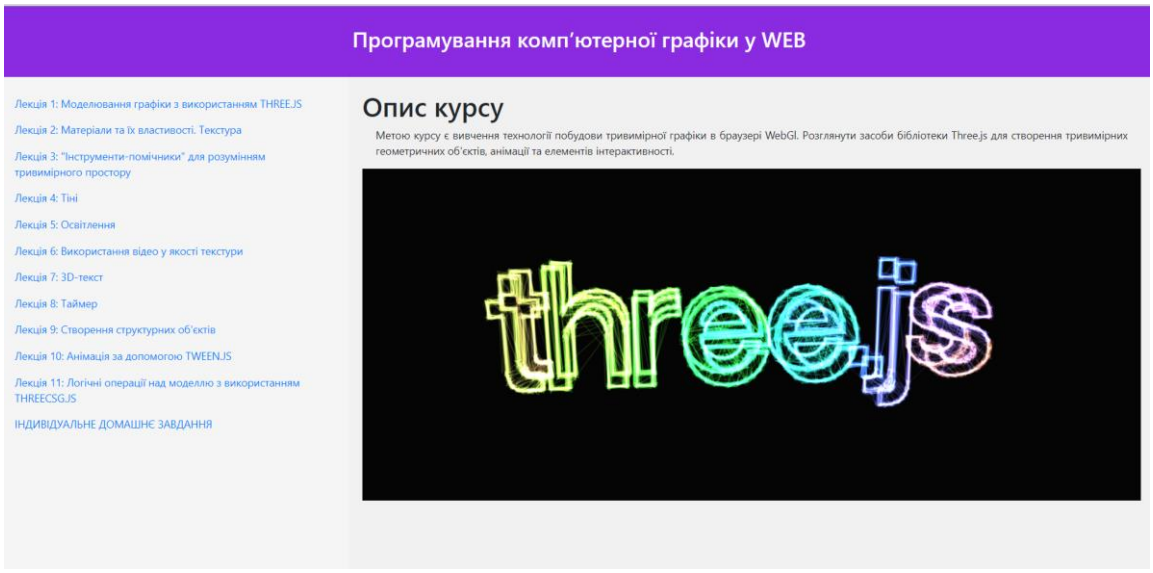


Рисунок 2.2 – Головна сторінка курсу index.html

Веб-сторінки лекцій (файли lecture1.html-lecture11.html) мають однакову структуру. Вони мають HEADER в якому вказано номер лекції, тему та кнопка HOME редіректом на index.html (див. рис. 2.3)

Лекційний матеріал відповідає робочій програмі курсу «Програмування комп'ютерної графіки у WEB».

Усі приклади, до різноманітних сценаріїв використання бібліотеки Three.js, відображаються безпосередню на сторінці у вігляді інтерактивного вікна с кодом програми у лівому частині, та її демонстрацією у правому частині (див. рис. 2.4). Більш докладно про інтерактивні приклади в дивись підрозділі 2.3.

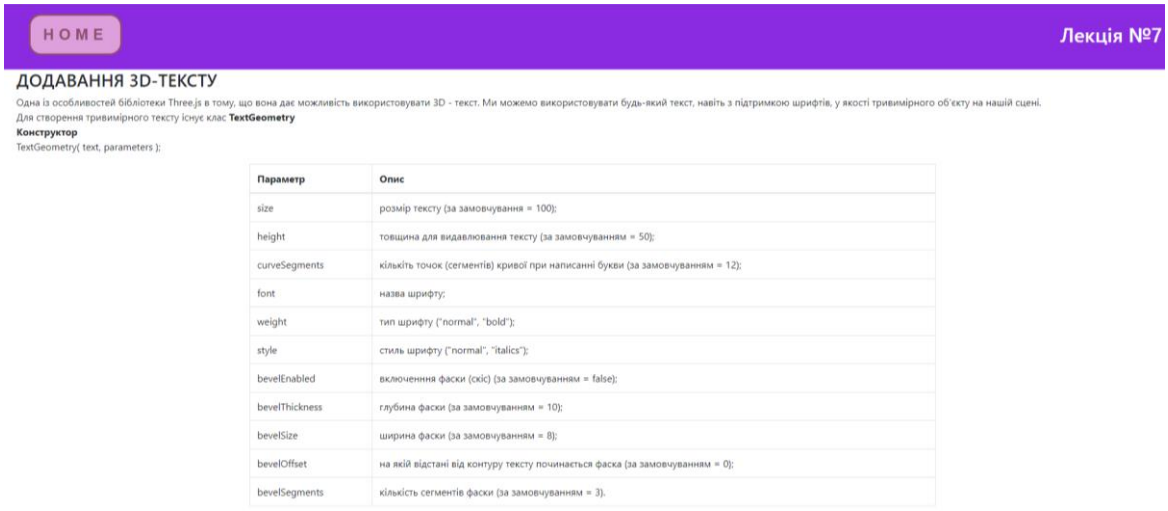


Рисунок 2.3 – Головна сторінка з лекційним матеріалом курсу

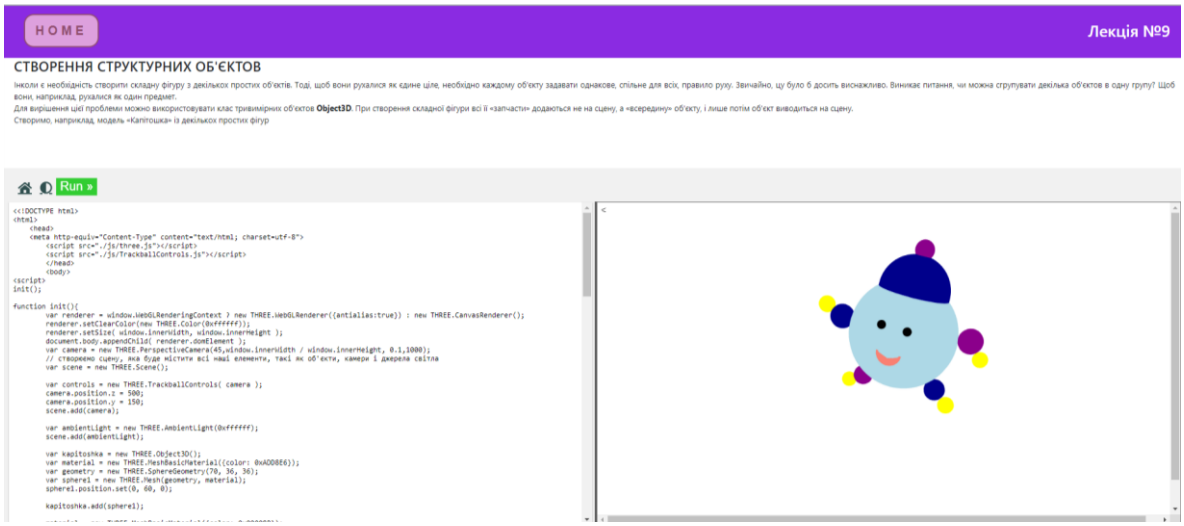


Рисунок 2.4 – Вікно з інтерактивними прикладами на веб-сторінці з лекційним матеріалом

Після кожной лекції є завдання до лабораторної роботи, які реалізовано за допомогою Boostrap JS Collaps (див. рисунок 2.5).

Клас collapse вказує на розбірний елемент – це вміст, який буде показано або приховано натисканням кнопки [9].

На index.html вкінці всіх лекцій доступні індивідуальні домашні завдання. Список завдань можна розширювати по мірі необхідності.

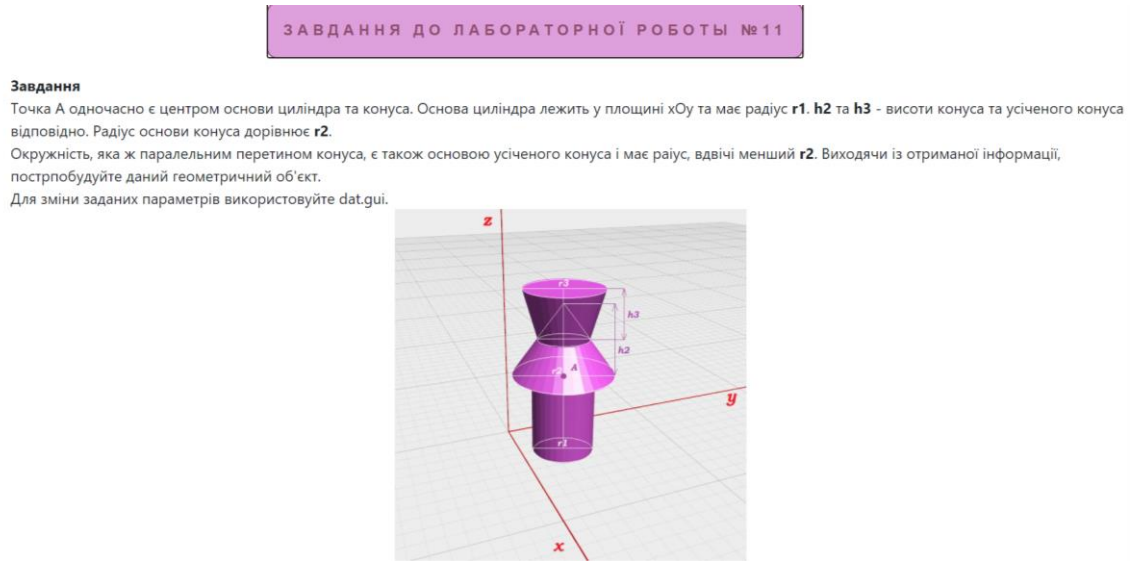


Рисунок 2.5 – Приклад завдання до лабораторної роботи

2.3 Інтерактивні приклади та 3D візуалізація

Розглянемо докладніше роботу інтерактивних прикладів. Із прикладом коду можна ознайомитися у Додатку А.

Інтерактивне вікно додається до лекції за допомогою тегу <embed>.

Панель курування складається з трьох кнопок:

- HOME – відкриває головну сторінку у новому вікні;
- RETHEME – змінює тему інтерактивного вікна із світлої в темну, і навпаки (див. рисунок 2.6 та рисунок 2.7);
- RUN – запускає код (повторно, або після змін).

Вікно поділено на 2 частини, з кодом роботи – зліва, та його візуалізація – справа.

Оскільки вікно інтерактивне, то користувач може виконувати зміни в коді безпосередньо в самій лекції, ці зміни не впливають на вихідній код з прикладом, тому після перезавантаження сторінки з лекцією всі зміни, які ввів користувач, будуть втрачені.

Мета – візуалізація лекційного матеріалу та покращення його запам’ятування шляхом миттєвого тестування отриманих знань.

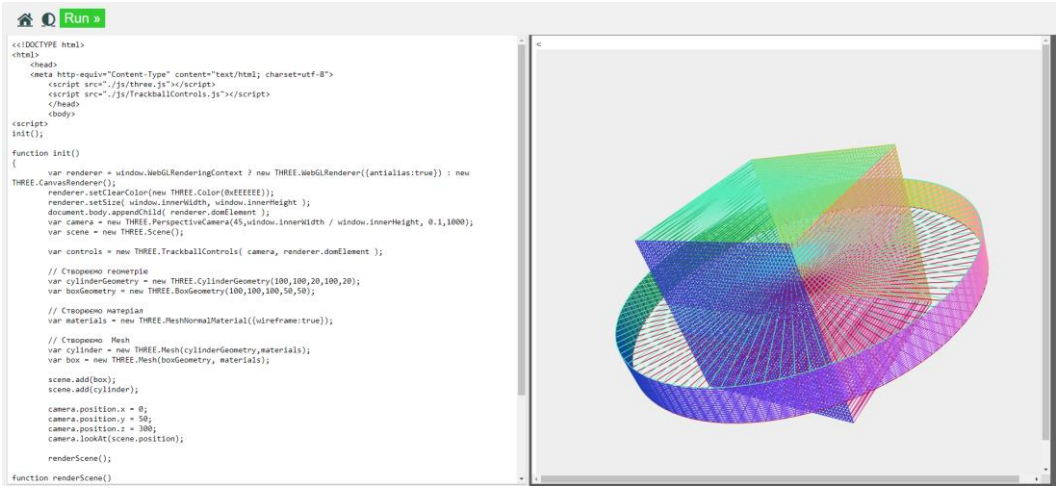


Рисунок 2.6 – Приклад інтерактивного вікна зі світлою темою

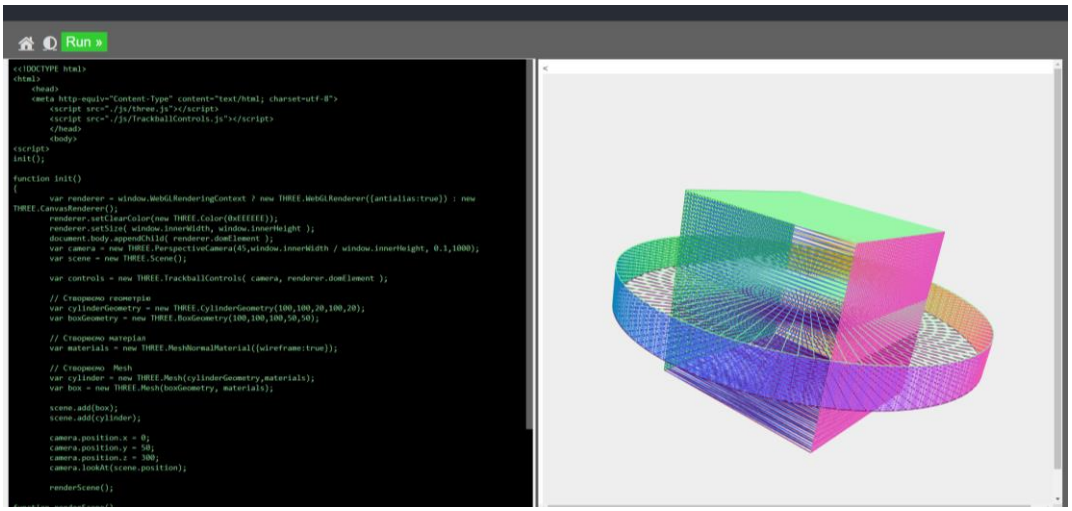


Рисунок 2.7 – Приклад інтерактивного вікна із темною темою

ВИСНОВКИ

У кваліфікаційній роботі пропонується підхід до оволодіння навичками комп’ютерної графіки, який не вимагає поглиблених знань OpenGL або WebGl.

У зв’язку з широким застосуванням у сучасному світі тривимірної графіки та візуалізації тема кваліфікаційної роботи досить актуальною. Використання Three.js бібліотеки у розробці графічних інтерфейсів має перевагу у швидкості та якості розробки на відмінно від інших технологій.

При розробці онлайн курсу з дисципліни «Програмування комп’ютерної графіки у web» було розширено знання та уміння із використання засобів для конструювання веб-сайтів: а саме: HTML, CSS, Java Script, JQuery, Bootsrap.

Весь матеріал викладено доступно з урахуванням змісту робочої програми та методичних рекомендацій викладання для студентів вищих навчальних закладів освіти.

Глибоко розкрито питання з основних засобів для побудови 3D візуалізації, використовуючи можливості бібліотеки Three.js

Отже, мета кваліфікаційної роботи повністю досягнута.

ПЕРЕЛІК ПОСИЛАНЬ

1. 3D-моделювання: програми та реалізація. URL: https://sites.google.com/site/3dmodeluvana/realizacia-3d-modeluvanna-sferi-ta

2. Вильданов А. Н. 3d – моделирование на WebGL с помощью библиотеки Three.js. Уфа: РИЦ БашГУ, 2014. 114 с.

3. Dirksen J. Learning Three.js: The javascript 3D library for WebGL. Birmingham - Mumbai: Packt Publishing, 2013. 402 с.

4. Мацуда Коичи. WebGL. Программирование трехмерной графики. М. : ДМК Пресс, 2015. 494 с.

5. Dirksen J. Three.js Cookbook. Birmingham – Mumbai: Packt Publishing, 2015. 300 с.

6. Three.js – JavaScript 3D library. URL: http://threejs.org,

7. Основи Three.js .URL: https://threejsfundamentals.org/threejs/

8. Керівництво для новачка по Three.js. URL: https://webdesign.tutsplus.com/ru/tutorials/a-noobs-guide-to-threejs--cms-28639

9. Повна документація та довідкові матеріали мови HTML та Java Script. URL: https://www.w3schools.com/

10. Вікіпедія – вільна енциклопедія. URL: https://uk.wikipedia.org

11. Уроки з Bootstrap v3. URL: https://tokar.ua/read/6901#:~:text=Bootstrap.

12. Введення в 3D. Основи Three.js. URL: https://habr.com/ru/post/494810/

ДОДАТОК А

Код програми для інтерактивного вікна

```
<!DOCTYPE html>
<html>
        <head>
                <title>Try Code</title>   // Назва сторінки у браузері
                //підключення фійлу із стилями, бібліотек, тощо.
                <link rel="stylesheet" href="main.css">
                <link rel="stylesheet"
                href="https://use.fontawesome.com/releases/v5.6.3/css/all.css"
                integrity="sha384-
                UHRtZLI+pbxtHCWp1t77Bi1L4ZtiqrqD80Kn4Z8NTSRyMA2Fd33n
                5dQ8lWUE00s/" crossorigin="anonymous">
        </head>
        <script src=" https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/
        jquery.min.js" > </script>
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/js
        /bootstrap.min.js"></script>
<script>
var fileID = "";
var loadSave = false;
// Функція для підключення та запуск коду в правій частині інтерактивного
вікна
function getSavedFile() {
   loadSave = true;
   var htmlCode;
   var paramObj = {};
   paramObj.fileid = "";
   fileID = paramObj.fileid;
   var paramA = JSON.stringify(paramObj);
   var httpA = new XMLHttpRequest();
   httpA.open("POST", globalURL, true);
   httpA.setRequestHeader("Content-Type", "application/x-www-form-
urlencoded");
   httpA.onreadystatechange = function() {
      if(httpA.readyState == 4 && httpA.status == 200) {
         document.getElementById("textareaCode").value = httpA.responseText;
         window.editor.getDoc().setValue(httpA.responseText);
         loadSave = false; }
   }
   httpA.send(paramA);  }
```

```
</script>
<body>
        <div class="trytopnav"> //кнопки в хедері інтерактивного вікна
                 <a id="tryhome" href="../../index.html" target="_blank"
                class="button-icons fa fa-home" style="font-size:25px"></a> //хоум
                <a href="javascript:void(0);" onclick="retheme()" title="Change
                Theme" class="button-icons fa fa-adjust" style="font-size:25px"></a>
                //зміна теми
                 <button class="run-button-text-green" style="font-size:25px"
                onclick="submitTryit(1);">Run &raquo;</button>//запуск коду
        </div>
<div id="container">
  <div id="textareacontainer">
    <div id="textarea">
      <div id="textareawrapper">
        <textarea autocomplete="off" id="textareaCode" wrap="logical"
spellcheck="false">
<<!DOCTYPE html> // тут може бути будь-який код
<html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <script src="./js/three.js"></script>
        <script src="./js/TrackballControls.js"></script>
        <script src="./js/ThreeCSG.js"></script>
        </head>
        <body>
<script>
init();
function init(){
        var renderer = window.WebGLRenderingContext ? new
THREE.WebGLRenderer({antialias:true}) : new THREE.CanvasRenderer();
        renderer.setClearColor(new THREE.Color(0xEEEEEE));
        renderer.setSize( window.innerWidth, window.innerHeight );
        // додаємо вивід візуалізатора до html-елементу
        document.body.appendChild( renderer.domElement );
        // створюємо камеру
        var camera = new THREE.PerspectiveCamera(45,window.innerWidth /
window.innerHeight, 0.1,1000);
        var scene = new THREE.Scene();
        var controls = new THREE.TrackballControls( camera,
renderer.domElement );
        var light = new THREE.PointLight(0xffffff);
        light.position.set(0,250,0);
        scene.add(light);
```

```
        var cylinderGeometry = new
THREE.CylinderGeometry(100,100,20,100,20);
        var boxGeometry = new THREE.BoxGeometry(100,100,100,50,50);

        var materials = new THREE.MeshNormalMaterial({wireframe:true});
        var materials1 = new THREE.MeshNormalMaterial({wireframe:false});

        var cylinder = new THREE.Mesh(cylinderGeometry,materials);
        var box = new THREE.Mesh(boxGeometry, materials);

        var cylinderBSP = new ThreeBSP(cylinder);
        var boxBSP = new ThreeBSP(box);

        var newBSP = cylinderBSP.intersect(boxBSP);
        var newMesh = newBSP.toMesh(materials1);
        scene.add( newMesh );

        camera.position.x = 0;
        camera.position.y = 200;
        camera.position.z = 300;
        camera.lookAt(scene.position);
        renderScene();

function renderScene() {
        requestAnimationFrame( renderScene );
        controls.update();
        renderer.render( scene, camera );
        }       };
</script>
        </body>
</html>
</textarea>
      </div>
    </div>
  </div>
  <div id="iframecontainer">
    <div id="iframe">
      <div id="iframewrapper"></div>
    </div>
  </div>
</div>

<script>
submitTryit() //функція запуску коду
function submitTryit(n) {
```

```
  if (window.editor) {
    window.editor.save();
  }
  var text = document.getElementById("textareaCode").value;
  var ifr = document.createElement("iframe");
  ifr.setAttribute("frameborder", "0");
  ifr.setAttribute("id", "iframeResult");
  ifr.setAttribute("name", "iframeResult");
  ifr.setAttribute("allowfullscreen", "true");
  document.getElementById("iframewrapper").innerHTML = "";
  document.getElementById("iframewrapper").appendChild(ifr);
  if (loadSave == true ) {
    ifr.setAttribute("src", "/code/opentext.htm");
  } else if (fileID != "" && loadSave == false) {
    var t=text;
    t=t.replace(/=/gi,"w3equalsign");
    t=t.replace(/\+/gi,"w3plussign");
    var pos=t.search(/script/i)
    while (pos>0) {
      t=t.substring(0,pos) + "w3" + t.substr(pos,3) + "w3" + t.substr(pos+3,3) + "tag"
+ t.substr(pos+6);
            pos=t.search(/script/i);
    }
    document.getElementById("code").value=t;
    document.getElementById("codeForm").action =
"https://tryit.w3schools.com/tryit_view.php?x=" + Math.random();
    document.getElementById('codeForm').method = "post";
    document.getElementById('codeForm').acceptCharset = "utf-8";
    document.getElementById('codeForm').target = "iframeResult";
    document.getElementById("codeForm").submit();
  } else {
    var ifrw = (ifr.contentWindow) ? ifr.contentWindow :
(ifr.contentDocument.document) ? ifr.contentDocument.document :
ifr.contentDocument;
    ifrw.document.open();
    ifrw.document.write(text);
    ifrw.document.close();
    if (ifrw.document.body && !ifrw.document.body.isContentEditable) {
      ifrw.document.body.contentEditable = true;
      ifrw.document.body.contentEditable = false;
    }}}
function retheme() { //функція зміни теми інтерактивного вікна із світлої в
темну і навпаки
  var cc = document.body.className;
  if (cc.indexOf("darktheme") > -1) {
```

```
    document.body.className = cc.replace("darktheme", "");
    if (opener) {opener.document.body.className = cc.replace("darktheme", "");}
    localStorage.setItem("preferredmode", "light");
  } else {
    document.body.className += " darktheme";
    if (opener) {opener.document.body.className += " darktheme";}
    localStorage.setItem("preferredmode", "dark");
  }}
(
function setThemeMode() {
  var x = localStorage.getItem("preferredmode");
  if (x == "dark") {
    document.body.className += " darktheme";
  }
})();
</script>
</body>
```

ДОДАТОК Б

Код програми для візуалізації шахових фігур

Пішак

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/three.js/89/three.min.js"></script>
<script src="js/TrackballControls.js"></script>
<html>
<head>
   <title>SiteName</title>
   <style>
      body { margin: 0; overflow:hidden;}
      canvas { width: 100%; height: 100% }
   </style>
</head>
<body>
<script>
  // створюємо сцену, яка буде містити всі наші елементи, такі як об'єкти, камери і джерела світла
   var scene = new THREE.Scene();
// створюємо камеру
   var camera = new THREE.PerspectiveCamera( 60, window.innerWidth / window.innerHeight, 0.1, 1000);
         camera.position.z = 400;
         camera.position.y = 200;
   // створюємо рендер і задаємо розмір
   var renderer = new THREE.WebGLRenderer();
   renderer.setSize( window.innerWidth, window.innerHeight );
   renderer.setClearColor (new THREE.Color(0xffffff));
  // додаємо вивід візуалізатора до html-елементу
   document.body.appendChild( renderer.domElement );

   var light = new THREE.HemisphereLight( 0xffffff, 0xc0c0c0, 1 );
        scene.add( light );
// створюємо тривимірний об’єкт для пішака
  var pawn = new THREE.Object3D();
// Створюємо матеріал
  var material = new THREE.MeshPhongMaterial ({color: 0xdaa520, specular:0x00b2fc, shininess:50, blending:THREE.NormalBlending, depthTest:true});
```

```
// Заповнюємо масив вершин
   var points = [];
   for (var i=-2.4; i<0.7; i=i+0.1)
   {points.push(new THREE.Vector2(10+20*Math.exp(-i*i),24*i));}
// Додаємо вершини до об’єкту та повертаємо
   var geometry = new THREE.LatheGeometry(points,32);
   object=new THREE.Mesh(geometry,material);
   object.position.set(0,26,0);
   object.rotation.x=Math.PI;
   pawn.add(object);
// Додаємо інші елементі  об’єкту
   var geometry = new THREE.SphereGeometry(16,50,50);
   var sphere =  new THREE.Mesh(geometry,material);
   sphere.position.set(0,102,0);
   sphere.rotation.x=Math.PI;
   pawn.add(sphere);

   var geometry = new THREE.CylinderGeometry(22,22,6,32);
   var material = new THREE.MeshPhongMaterial ({color: 0x4c3c18, specular:0x00b2fc, shininess:50, blending:THREE.NormalBlending, depthTest:true});
   var disc1 =  new THREE.Mesh(geometry,material);
   disc1.position.set(0,86,0);
   pawn.add(disc1);

   var geometry = new THREE.CylinderGeometry(32,32,6,32);
   var material = new THREE.MeshPhongMaterial ({color: 0xdaa520, specular:0x00b2fc, shininess:50, blending:THREE.NormalBlending, depthTest:true});
   var disc2 =  new THREE.Mesh(geometry,material);
   disc2.position.set(0,10,0);
   pawn.add(disc2);

var geometry = new THREE.CylinderGeometry(32,32,8,32);
var material = new THREE.MeshPhongMaterial({color:0x4c3c18, specular:0x00b2fc,shininess:50,blending:THREE.NormalBlending,depthTest:true} );
   var disc3 =  new THREE.Mesh(geometry,material);
   disc3.position.set(0,3,0);
   pawn.add(disc3);
// Додаємо тривимірний об’єкту на сцену
   scene.add(pawn);
   // Додаємо додаємо контроли
                controls = new THREE.TrackballControls( camera );
                controls.rotateSpeed = 1.0;
```

```
            controls.zoomSpeed = 1.2;
            controls.panSpeed = 0.8;
            controls.noZoom = false;
            controls.noPan = false;
            controls.staticMoving = true;
            controls.dynamicDampingFactor = 0.3;

        function render() {
            requestAnimationFrame(render);
            controls.update();
            renderer.render(scene, camera);
        }
        render();

</script>
</body>
```

Слон

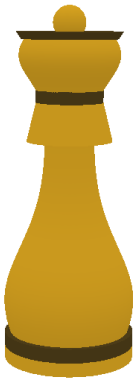
```
var bishop = new THREE.Object3D();

//основа фігури за такою самою функцією, як і для пішака

var geometry = new THREE.SphereGeometry(5,50,50);
var sphere =  new THREE.Mesh(geometry,material);
sphere.position.set(0,137,0);
sphere.rotation.x=Math.PI;
bishop.add(sphere);

var geometry = new THREE.CylinderGeometry(22,22,6,32);
scene.add(bishop);
```

Typa

```
var castle = new THREE.Object3D();
var points = [];
for (var i=-2.4; i<0.7; i=i+0.1)
{ points.push(new THREE.Vector2(15+1*Math.exp(-i),24*i)); }
var geometry = new THREE.LatheGeometry(points,32);
object=new THREE.Mesh(geometry,material);
object.position.set(0,26,0);
castle.add(object);

var geometry = new THREE.CylinderGeometry(28,28,6,32);
var disc1 =  new THREE.Mesh(geometry,material);
disc1.position.set(0,-40,0);
castle.add(disc1);

 var geometry = new THREE.CylinderGeometry(28,28,6,32);
var disc2 =  new THREE.Mesh(geometry,material);
disc2.position.set(0,-34,0);
castle.add(disc2);

var geometry = new THREE.CylinderGeometry(24,24,6,32);
var disc3 =  new THREE.Mesh(geometry,material);
disc3.position.set(0,34,0);
castle.add(disc3);

 var geometry = new THREE.CylinderGeometry(22,24,6,32);
var disc4 =  new THREE.Mesh(geometry,material);
disc4.position.set(0,40,0);
castle.add(disc4);

var geometry = new THREE.CubeGeometry(5,13,10);
var cube1 =  new THREE.Mesh(geometry,material);
cube1.position.set(19,45,0);
```

```
cube1.rotation.x=Math.PI;
castle.add(cube1);

var geometry = new THREE.CubeGeometry(5,13,10);
var cube2 =  new THREE.Mesh(geometry,material);
cube2.position.set(-19,45,0);
cube2.rotation.x=Math.PI;
castle.add(cube2);

var geometry = new THREE.CubeGeometry(5,13,10);
var cube3 =  new THREE.Mesh(geometry,material);
cube3.position.set(0,45,19);
cube3.rotation.y=Math.PI/2;
castle.add(cube3);

var geometry = new THREE.CubeGeometry(5,13,10);
var cube4 =  new THREE.Mesh(geometry,material);
cube4.position.set(0,45,-19);
cube4.rotation.y=Math.PI/2;
castle.add(cube4);

scene.add(castle);
```

Королева



```
var queen = new THREE.Object3D();
var material = new THREE.MeshPhongMaterial ({color: 0xdaa520, specular:0x00b2fc, shininess:50, blending:THREE.NormalBlending, depthTest:true});
var points = [];
  for (var i=-2.4; i<0.7; i=i+0.1)
  {points.push(new THREE.Vector2(10+18*Math.exp(-i*i),34*i));}
  var geometry = new THREE.LatheGeometry(points,32);
```

```
object1=new THREE.Mesh(geometry,material);
object1.position.set(0,26,0);
object1.rotation.x=Math.PI;
queen.add(object1);

var points = [];
for (var i=-2.4; i<0.7; i=i+0.1)
{ points.push(new THREE.Vector2(10+11*Math.exp(-i*i),14*i)); }
var geometry = new THREE.LatheGeometry(points,32);
object2=new THREE.Mesh(geometry,material);
object2.position.set(0,130,0);
queen.add(object2);

var geometry = new THREE.CylinderGeometry(28,28,6,32);
var disc1 =  new THREE.Mesh(geometry,material);
disc1.position.set(0,10,0);
queen.add(disc1);

 var geometry = new THREE.CylinderGeometry(28,28,6,32);
var disc2 =  new THREE.Mesh(geometry,material);
disc2.position.set(0,4,0);
queen.add(disc2);

var geometry = new THREE.CylinderGeometry(14,14,6,32);
var disc3 =  new THREE.Mesh(geometry,material);
disc3.position.set(0,114,0);
queen.add(disc3);

 var geometry = new THREE.CylinderGeometry(14,18,16,32);
var disc4 =  new THREE.Mesh(geometry,material);
disc4.position.set(0,103,0);
queen.add(disc4);

 var geometry = new THREE.CylinderGeometry(22,20,3,32);
var disc5 =  new THREE.Mesh(geometry,material);
disc5.position.set(0,138,0);
queen.add(disc5);

var geometry = new THREE.SphereGeometry(7,9,10);
var sphere =  new THREE.Mesh(geometry,material);
sphere.position.set(0,143,0);
queen.add(sphere);
scene.add(queen);
queen.position.set(0,-40,0);
```

Король

```
var king = new THREE.Object3D();  //Основа для короля така як і в Королеви.
 var korona1 =  new THREE.Mesh(geometry,material);
 korona1.position.set(0,150,0);
 korona1.rotation.x=Math.PI;
 king.add(korona1);

 var geometry = new THREE.CylinderGeometry(6,2,13,4);
 var korona2 =  new THREE.Mesh(geometry,material);
 korona2.position.set(0,160,0);
 king.add(korona2);

 var geometry = new THREE.CylinderGeometry(6,2,13,4);
 var korona3 =  new THREE.Mesh(geometry,material);
 korona3.position.set(-8,155,0);
 korona3.rotation.x=Math.PI/2; korona3.rotation.z=Math.PI/2;
 king.add(korona3);

 var geometry = new THREE.CylinderGeometry(6,2,13,4);
 var korona4 =  new THREE.Mesh(geometry,material);
 korona4.position.set(8,155,0);
 korona4.rotation.x=Math.PI/2;korona4.rotation.z=-Math.PI/2;
 king.add(korona4);

 var geometry = new THREE.SphereGeometry(5,5,10);
 var sphere =  new THREE.Mesh(geometry,material);
 sphere.position.set(0,155,0);
 king.add(sphere);
 scene.add(king);
```