ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ
към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ

ДИПЛОМНА РАБОТА

Тема: Инструмент за брандиране на 3D модели

Дипломант:
Виолета Георгиева

Научен ръководител:
Кирил Митов

СОФИЯ

2 0 2 1

ТЕХНОЛОГИЧНО УЧИЛИЩЕ
ЕЛЕКТРОННИ СИСТЕМИ
към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ

Дата на заданието: 15.11.2020 г. Утвърждавам:.....
Дата на предаване: 15.02.2021 г. /проф. д-р инж. Т. Василева/

ЗАДАНИЕ
за дипломна работа

на ученичката Виолета Георгиева Георгиева 12 в клас

1. Тема: "Инструмент за брандиране на 3Д модели"
2. Изисквания:
   - Зареждане на празна сцена с възможност за качване на 3Д модел и 3Д лого
   - Да може да се запаметява редактираната сцена и да се зарежда
   - Да може моделите да се местят и завъртат
   - Инструмент за изкривяване на плоско лого
   - Инструмент за "долепяне" (snap) на двата модела
   - Инструмент за съединяване на двата модела в един
   - Генериране на 3Д модел за парче текст
   - Инструмент за автоматично залепване на логото върху една или повече повърхнини на модел
3. Съдържание 3.1 Обзор

   3.2 Същинска част

   3.3 Приложение

Дипломант :.....
/Виолета Георгиева/
Ръководител:.....
/Кирил Митов/
Директор:.....
/ доц. д-р инж. Ст. Стефанова /

Мнение на научния ръководител

Дипломантът Виолета Георгиева се запозна с някои от начините, средствата за разработка на интернет приложения за обработка на 3Д модели. Запозна се с популярни инструменти и библиотеки. Успя да разработи първите няколко етапа от заданието, което имаше за цел да автоматизира поставянето на информация за търговски марки върху 3Д модели. Настоящата дипломна работа представя постигнатите от дипломанта резултати.

За рецензент предлагам Михаил Кирилов.

Научен ръководител:

/Кирил Митов/

Увод

В настоящата дипломна работа се има за цел да се разработи инструмент за брандиране на триизмерни модели, който да дава възможност на потребителите да брандират, със своето лого, продуктите си.

Всичко, което човек знае за един продукт, който използва, се дължи на марката му. Това е връзката, която свързва компанията с клиента и обратно. Брандирането е това, което подсигурява спокойствието на създателя, че неговото творение няма да бъде откраднато. То е гаранция, в случай, че се стигне до разправа, с човека, откраднал нашето произведение. Марката е и тази, която придава стойност на нашия продукт.

В триизмерния свят брандирането е скъпо нещо, но по този начин ние запазваме авторското си право над модела ни. Когато брандираме един триизмерен модел логото става част от него и не може да бъде премахнато. Ако някой обаче се опита ще е доста трудна задача, която не си струва усилията. А когато ние притежаваме един много ценен триизмерен обект, брандирането му е наистина желателно, тъй като за създаването му са вложени много ресурси.

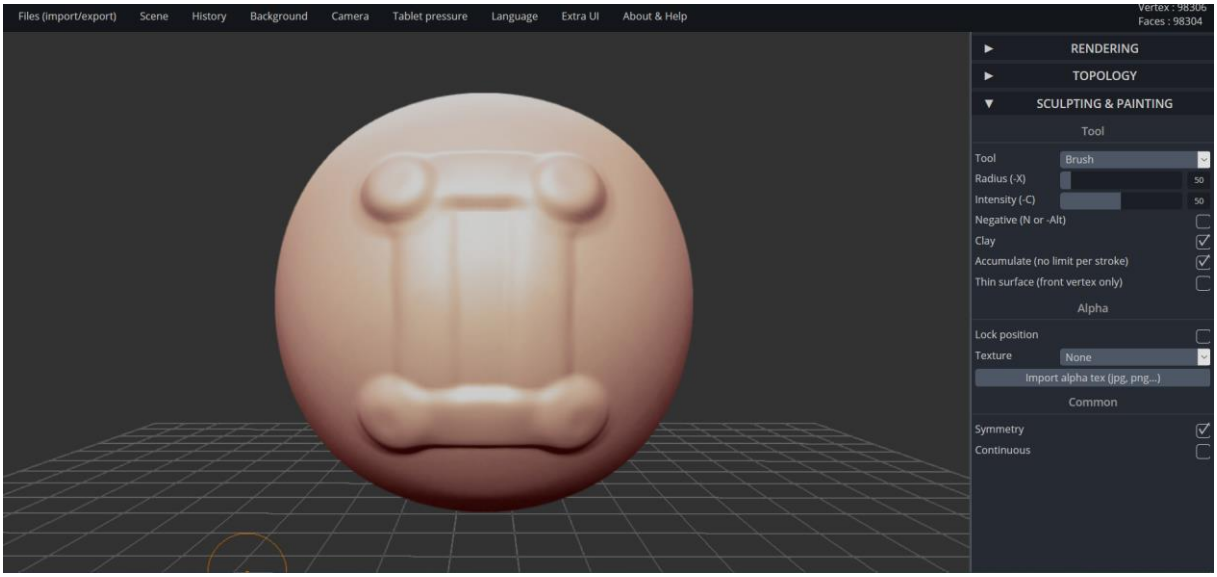
Запазването на правата на създателя е много важно за мен, това е и причината да избера тази тема за дипломната си работа.

Глава Първа

Преглед на съществуващи системи и продукти за работа с 3D обекти и развойни средства и среди, спомагащи визуализацията им

1.1. Преглед на съществуващи подобни инструменти за брандиране на 3D модели

1.1.1. SculptGL – уеб приложение за скулптуриране



(Фиг. 1.1.1.1)

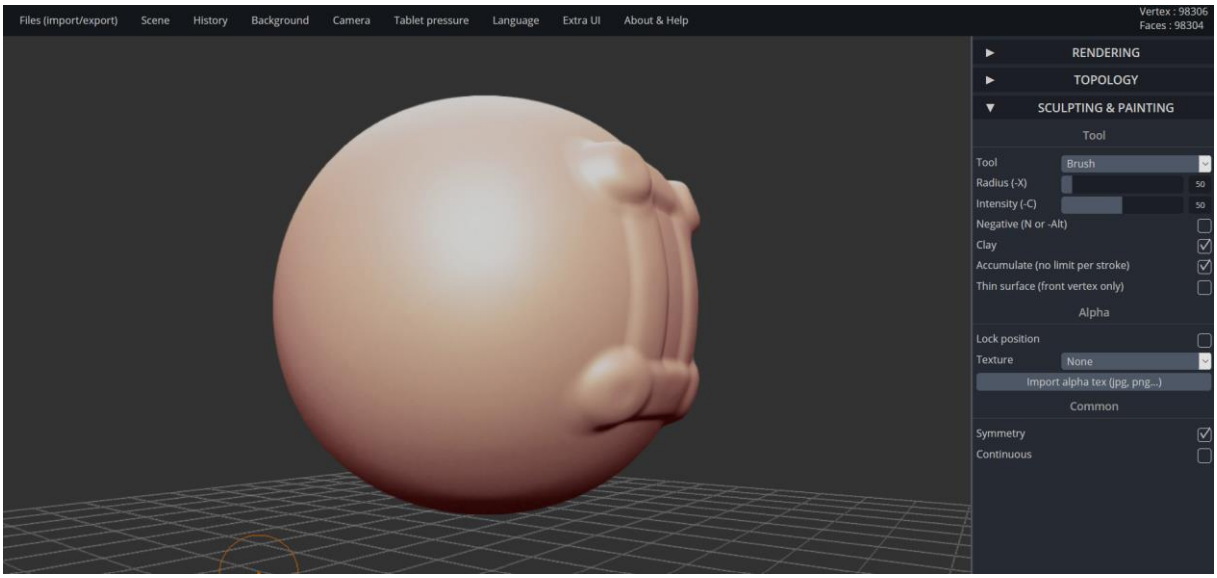
SculptGL е безплатна, уеб базирана, CAD (Computer Assisted Design) програма за 3D скулптуриране. Тя позволява на потребителите да извайват 3D дизайни, да прилагат симетрия, да импортират / експортират тези файлове в триизмерни формати за печат и автоматично да ги споделят в портфолио сайтове като Sketchfab. Не кара потребителите да създават акаунти и може да се използва на устройства, които имат ограничено вътрешно пространство за съхранение или памет. За да направите файловете за печат, може да ви е необходим втори софтуер (като Blender).**[1]** Всеки може лесно да създаде модел, като просто отвори URL адреса на SculptGL (https://stephaneginier.com/sculptgl/). Този онлайн софтуер позволява да се създават обект с помощта на различни инструменти.

SculptGL е добър за:

- Скулптуриране: Скулптурният интерфейс е интуитивен и отзивчив.

- Симетрия / Асиметрия: Симетрията може да се включва / изключва по време на работата, без автоматично да прави целия си дизайн симетричен. Симетрията засяга само действията, които извършвате след включването му. При изключване могат да се правят асиметрични детайли.

- Експортиране и импортиране на 3D файлове (поддържаните формати включват .OBJ, .STL и .PLY) **[1]**

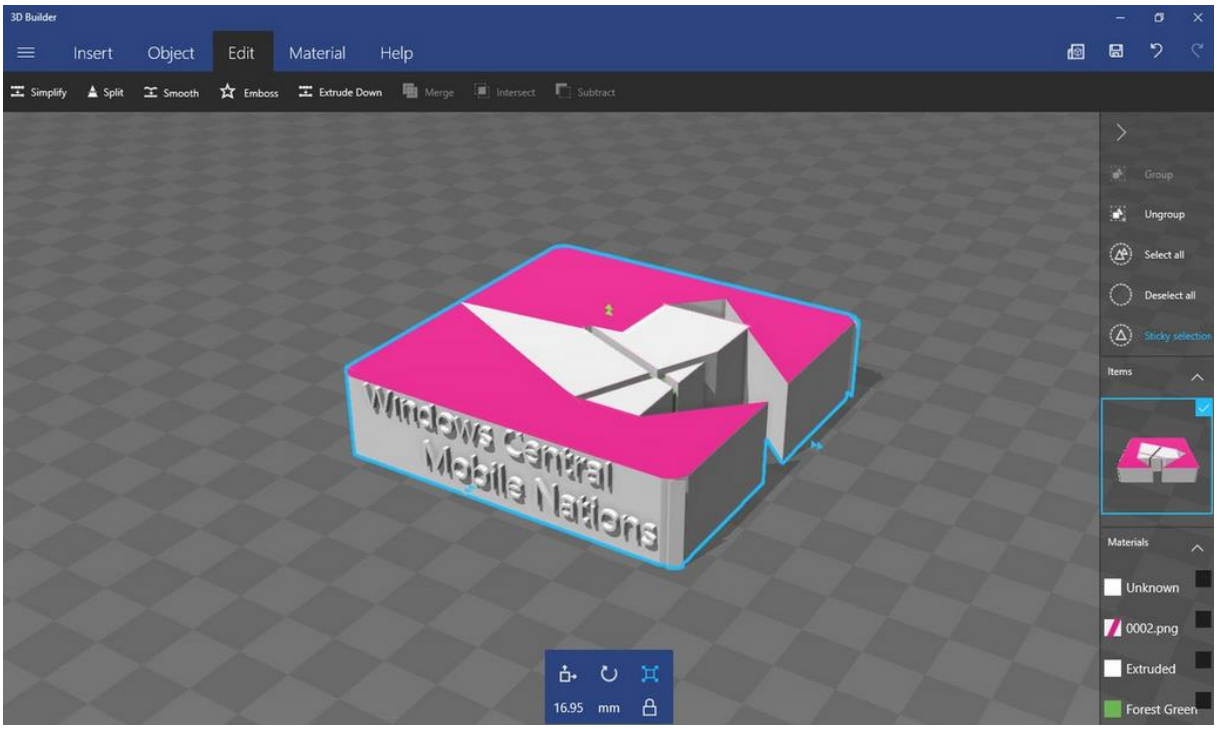


(Фиг. 1.1.1.2)

1.1.2. Microsoft 3D Builder

3D Builder е софтуер за 3D моделиране на Microsoft, който позволява лесно да се проектират, визуализират, редактират и модифицират 3D модели. Това е безплатно решение, инсталирано по подразбиране на всички компютри с Windows 10. 3D Builder е съвместим с 3D печат, тъй като позволява експортиране на модели във формати STL, OBJ или 3MF. Microsoft също предлага богат избор от 3D модели, които са готови за безплатно изтегляне директно от интерфейса. Днес този софтуер получава много положителни отзиви за многобройните функционалности, които предлага.

3D Builder се основава на моделиране, базирано на конструктивната геометрия на твърдите тела, точно както Tinkercad. Например потребителят може да създаде обект чрез комбиниране на различни прости форми като кубчета или сфери, използвайки булеви геометрични оператори (събиране, изваждане и т.н.).

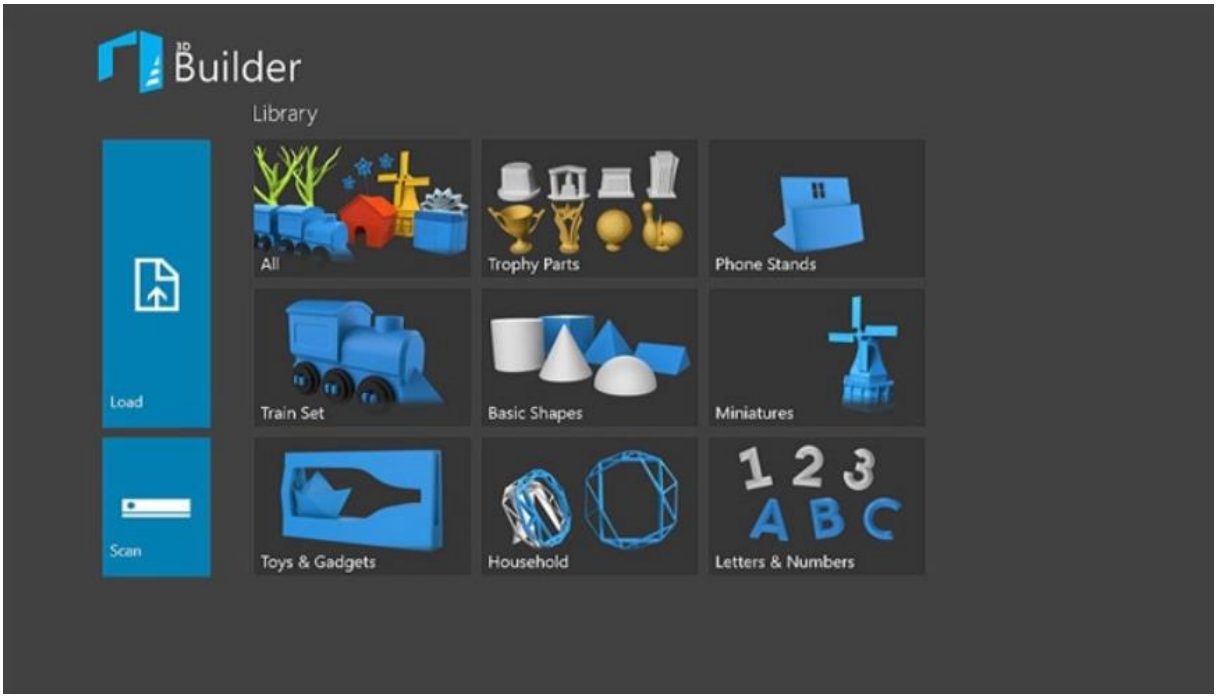


(Фиг. 1.1.2.1)

3D Builder може да бъде изтеглен безплатно от магазина на Windows и е съвместим само с Windows. Предлага се на компютър и мобилни телефони, но също така и на Xbox One, HoloLens и Surface Hub. 3D Builder предлага три основни функции: създаване на 3D модел, модифициране на 3D файл и преглед. Когато потребителят отвори софтуера, той може да види наскоро прегледаните модели, да зареди модел от библиотека, предлагана от Microsoft, или да започне създаването на собствен файл - или от нулата, чрез 3D сканиране, или от модел, качен преди това на друга платформа.

Създаването на 3D модел е доста интуитивно, тъй като всичко се прави с прости геометрични фигури: куб, цилиндър, пирамида, шестоъгълник, тетраедър и др. Просто добавете, наслагнете и пресечете тези форми, за да получите желания резултат. Потребителят ще може да добавя текст към тези фигури, както и да променя текстурата, цвета и т.н.

Разделът „Display“ позволява да получите рендериране, различни нива на сенки, прозрачност и т.н., като по този начин предлага много реална визуализация. След като моделът приключи, можете да го експортирате във формати 3MF, OBJ, STL или PLY, съвместими с 3D печат.



(Фиг. 1.1.2.2)

Ако не искате да създавате своя 3D модел, можете да започнете с вече съществуващ файл и да го промените директно от интерфейса. Интерфейсът ви позволява да вмъквате фигури, да променяте размера на обекта, да го деформирате и т.н. По принцип потребителят може да променя модела, както желае. Също така е важно да се отбележи, че 3D Builder ви позволява да поправяте обекти преди печат: софтуерът открива потенциални дефекти и по този начин гарантира по-чист производствен процес.

Създадена е и специална поддръжка, която ви позволява да следвате уроци, да получавате съвети и да решавате потенциалните си проблеми.**[2]**

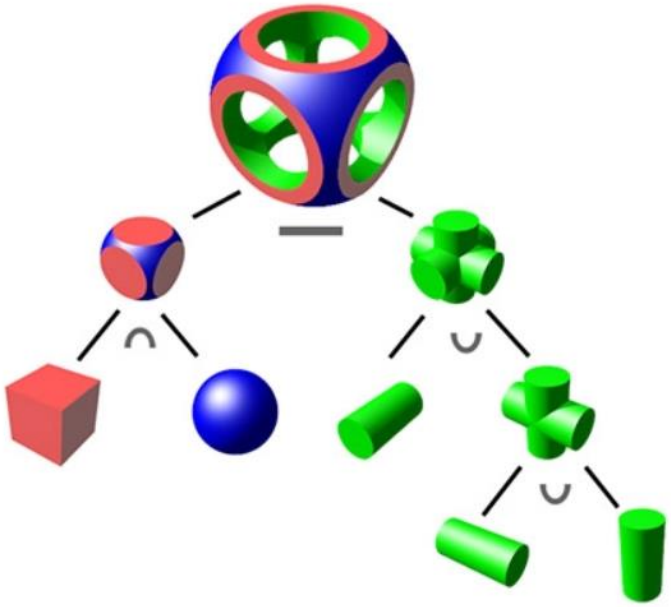
1.1.3. Tinkercad

Tinkercad е онлайн колекция от софтуерни инструменти на Autodesk, които дават възможност на напълно начинаещите да създават 3D модели. Този CAD софтуер се основава на конструктивна твърда геометрия (CSG), която позволява на потребителите да създават сложни модели чрез комбиниране на по-

прости обекти заедно. В резултат на това този софтуер за 3D моделиране е лесен за ползване и в момента се използва от особено много учители, деца, любители и дизайнери. Освен това е безплатен. Tinkercad е добра алтернатива на друг софтуер за 3D моделиране.

Основни характеристики на Tinkercad:

Въпреки че Tinkercad е идеален за начинаещи, това не означава, че по-опитните в 3D моделирането няма да оценят този софтуер. Като се има предвид, че се основава на CSG за създаване на солидни модели, винаги можете да усложните модела си, като добавите още фигури. Освен това софтуерът позволява да се добавят електронни схеми към 3D дизайни, за да се създават обекти със светлина и движение. Крайният резултат може дори да бъде симулиран на софтуера, за да се провери как компонентите ще реагират в реалния живот. Друга възможност на Tinkercad е да трансформира 3D дизайн в изграждащи се тухлени модели, подобно на създаването на лего. И накрая, за тези, които обичат Minecraft, ще бъдете добре обслужени, тъй като ще можете да правите творения, съвместими с приложението.



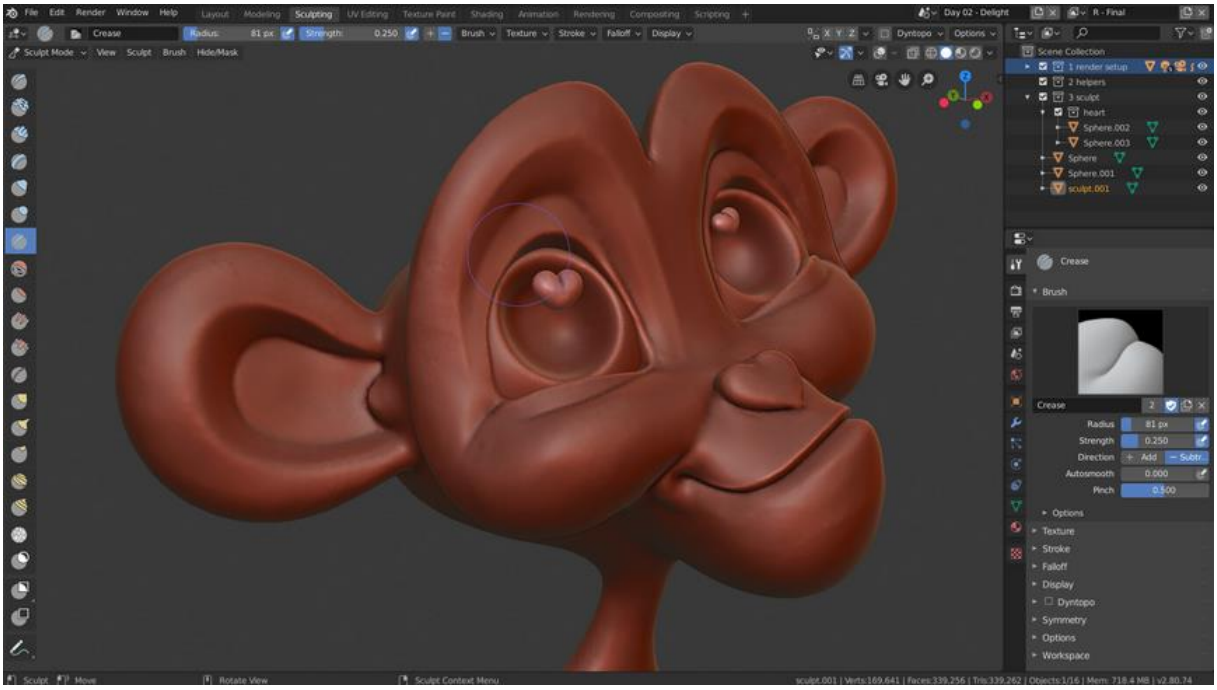
(Фиг. 1.1.3.1)

Следователно Tinkercad може да се използва за редица приложения, включително 3D печат. 3D моделите могат да бъдат запазени в три различни формата, STL, OBJ и SVG. **[3]**

1.1.4. Blender

Blender е безплатен софтуер с отворен код за разработване на 3D, който поддържа почти всички аспекти на 3D разработването. Със здрава основа на моделиращите възможности, има и стабилна текстура, анимация, осветление и множество други инструменти. Този софтуер е чудесен, независимо дали искате да се занимавате само със статични модели или да влезете в света на анимацията. Въпреки че е безплатен, Blender е достъпен, ценен и значим за широк кръг потребители, от начинаещия любител до професионалния аниматор. Дори НАСА го използва за много от своите публични модели. Той непрекъснато се усъвършенства от напреднали потребители.

Blender дава на хората творческа сила, за да могат да изградят всичко, което им е във въображението. За тези, които искат да направят свои собствени модели за 3D печат, това е безценен инструмент.



(Фиг. 1.1.4.1)

Blender е създаден под общия публичен лиценз на GNU, който позволява на хората да: използват Blender за всякакви цели, разпространяват Blender, проучват и променят начина на работа на Blender, и разпространяват променените версии на Blender. Това означава, че всичко, което се създава в Blender, е не само

собственост на потребителя, но той може също така да го продава и разпространява по негово желание.

Друга чудесна характеристика е, че Blender е напълно съвместим с различни платформи, така че ще работи на Windows, Mac OS и Linux. Компютрите, които са на повече от 10 години, може да имат малко затруднения с Blender.

Blender е пълен с полезни инструменти, но някои ще бъдат по-подходящи за начинаещи от други. За мнозина най-популярните инструменти са тези за: моделиране, скулптуриране и текстуриране, както и анимация. Тези, които създават обекти за 3D печат, може дори да не надхвърлят моделирането и скулптурирането.**[4]**

1.2. Програмни езици, развойни среди и библиотеки, рамки спомагащи визуализацията на 3D обекти

1.2.1. Програмни езици

3D програмирането днес се прави с помощта на OpenGL. Това е отворен API, който има разновидности за всеки език за програмиране. Така че изборът на език до голяма степен зависи от това на каква платформа ще се разработва 3D.

- Ако е уеб, това ще бъде JavaScript с WebGL.
- Ако е Android, ще е Java.
- Ако е iOS, това ще бъде Objective-C или Swift.
- Ако е Windows, това ще бъде един от .NET езиците, най-вероятно C #.

Няма „най-добър“ език за програмиране за създаване наЗD среди. Всъщност повечето 3D приложения не вървят директно с OpenGL, тъй като това е доста досадно. Образователно, но досадно. Повечето използват 3D рамки за OpenGL, за да осигурят удобство и повече възможности. Единството е една такава рамка. И могат да се използват инструменти за 3D моделиране като Blender, за да създаване обекти за 3D среда.Платформата и нуждите на приложението определят кой език е най-добър.**[5]**

1.2.2. Развойни среди

Програмистите на JavaScript, HTML5 и CSS имат голям избор от много добри инструменти. Основната разлика между редакторите и IDE е, че IDE могат да отстраняват грешки от кода, също така те имат поддръжка за системи за управление на жизнения цикъл на приложенията (ALM).

Sublime Text и Visual Studio Code са на върха на редакторите на JavaScript - Sublime Text със своята скорост, както и удобните му функции за редактиране, и Visual Studio Code с още по-добри функции и скорост.

1.2.2.1. Sublime Text - гъвкав, мощен текстов редактор за програмиране, който е светкавичен, но за проверката на кода е необходимо да се превключва към други прозорци.Освен бързината, много забележителни силни страни на Sublime Text са: поддръжката на повече от 70 типове файлове, сред които JavaScript, HTML и CSS; почти незабавна навигация и незабавно превключване на проекти; множество селекции (правене на множество промени наведнъж), включително селекции на колони, множество прозорци (използване на всичките монитори) и разделени прозорци; пълна персонализация с прости JSON файлове; Python-базиран API плъгин.**[6]**

Sublime Text е наличен за OS X, Windows и Linux. Един лиценз е всичко, от което се нуждае потребителя, за да използва Sublime Text на всеки компютър, който притежава, без значение каква операционна система използва.

1.2.2.2 Visual Studio Code

Visual Studio Code предоставя на разработчиците нов набор от инструменти, който съчетава простотата и рационализирания опит на редактор на код с най-доброто от това, което разработчиците се нуждаят за техния основен цикъл за редактиране и отстраняване на грешки в кода. Visual Studio Code е първият редактор на код и първият инструмент за разработване на различни платформи - поддържащ OS X, Linux и Windows - в семейството на Visual Studio. Едно от предимствата на използването на VSCode е способността му да се интегрира с Git, това е лесно и бързо, друго нещо е способността му да се интегрира дистанционно в различни екипи, няколко

разработчици могат да работят по един и същ проект или дори един и същ код с помощта на този инструмент: Visual Studio Live Share помага на няколко разработчици да работят заедно по-ефективно в сравнение с Sublime Text.

Предимства – Лек и бърз, безплатен, има прост и изчистен потребителски интерфейс, вграден терминал. IntelliSense – това е функция за автоматично довършване, помощник за подчертаване на синтаксиса, за да се елиминират грешките при писане и възможните синтактични грешки. Има огромна общност за подкрепа и изграждане на разширения и допълнителни ресурси. Има най-добрите функции като Git, Terminal, WebView, Live Share. Вътрешна поддръжка за съвместна работа с Github и няма сложни настройки. Има разширение Emmet, супер бързо HTML, CSS и JavaScript кодене.

Недостатъци - Няма поддръжка за по-стара версия за програмиране като ASP.NET. Функциите за отстраняване на грешки са ограничени.

VSCode е най-популярният редактор.**[7]**

1.2.3. Библиотеки и рамки спомагащи визуализацията на 3D обекти

1.2.3.1. WebGL

WebGL е графичен интерфейс за програмиране на приложения (API), създаден за използване в уеб приложения. Той се основава на отворения графичен език (OpenGL)(ES).

WebGL се използва от разработчиците, за да предостави независими от платформата средства за създаване на интерактивни графични приложения в мрежата. WebGL не се използва само за рисуване на графики на 2D и 3D игри, но също така и за ускоряване на функциите на уеб базирани редактори на изображения и техните ефекти, както и физически симулации.

Въпреки че WebGL е функционално базиран на OpenGL ES, той е частично написан на JavaScript. WebGL

се използва за визуализиране на интерактивни 2D и 3D графики в съвместими уеб браузъри. API позволява на потребителите да достъпват интерактивно съдържание на уеб страници, с ускорение на GPU, без да се налага първо да изтеглят или инсталират приставки. За разработчиците WebGL осигурява достъп до хардуер на ниско ниво с познатата кодова структура на OpenGL ES.**[8]**

Налице са редица рамки, които капсулират възможностите на WebGL, което улеснява изграждането на 3D приложения и игри, като THREE.js и BABYLON.js.

1.2.3.2. THREE.js

Three.js е JavaScript библиотека с отворен код, която се използва за показване на графики, 3D и 2D обекти в уеб браузъра. Той използва WebGL API зад сцената. Three.js позволява да се използва графичен процесор (GPU), за да СЕ изобразяват графични и 3D обекти на платно в уеб браузъра, тъй като използваме JavaScript, за да можем да взаимодействаме и с други HTML елементи.**[9]**

1.2.3.3. BABYLON.js

Babylon.js е JavaScript рамка с отворен код, която се използва за разработване на 3D приложения / видео игри. Официалният уебсайт на BabylonJS е www.babylonjs.com. Използването на Babylon.js рамка е лесно за потребителите. Той съдържа всички необходими инструменти за създаване и управление на 3D обекти, специални ефекти, звуци и т.н.

Babylon.js е един от най-популярните двигатели за 3D игрите и се използва широко от разработчиците. Като 3D библиотека, тя осигурява вградени функции. Тези функции помагат за внедряването на обща 3D функционалност с ефективни и точни методи.

Той е разработен с помощта на езика TypeScript, базиран е на WebGL и javascript.**[10]**

1.3. Основни 3D файлови формати

Форматът на 3D файл се използва за съхраняване на информация за 3D модели. Най-популярните формати STL, OBJ, FBX, COLLADA и др. Те се използват широко в 3D печат, видео игри, филми, архитектура, академични среди, медицина, инженерство и науки за земята. Всяка индустрия има свои собствени популярни 3D файлови формати по исторически и практически причини.

Основната цел на 3D файловия формат е да съхранява информация за 3D модели като обикновен текст или двоични данни. По-специално, те кодират геометрията, външния вид, сцената и анимациите на 3D модела.

Геометрията на модела описва неговата форма. Под външен вид имаме предвид цветове, текстури, тип материал и т.н. Сцената на 3D модел включва позицията на светлинни източници, камери и периферни обекти. И накрая, анимацията определя как се движи 3D модел.

Не всички 3D файлови формати обаче съхраняват всички тези данни. 3D файловите формати като STL съхраняват само геометрията на 3D модела и игнорират всички други атрибути. От друга страна, форматът COLLADA съхранява всичко.

Проблемът с 3D файловите формати е, че има буквално стотици от тях. Всеки производител на CAD софтуер като AutoDesk и Blender има свой собствен патентован формат, който е оптимизиран за техния софтуер. Така че, ако използвате AutoCAD, получавате DWG файл. Ако използвате Blender, получавате BLEND файл. Тук възниква проблемът, че няма как двама човека да работят върху един файл, ако единия ползва BLEND файл, а другия DWG.

За да се реши проблемът с оперативната съвместимост, неутрални или отворени формати са изобретени като междинни формати за конвертиране между два патентовани формата. Естествено, тези формати са изключително популярни сега.**[11]**

1.4. 3D моделиране и основни понятия в него

3D моделирането е цифрово представяне и създаване на триизмерен обект, в специален софтуер за 3D моделиране. Обектът

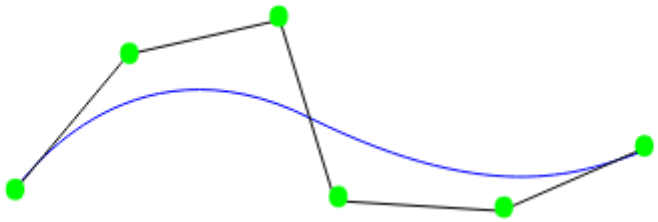
може да бъде създаден от прости форми до сложни модели с много полигони. Многоъгълникът е един триъгълник и са необходими много триъгълници, за да се направи кръг или сложен обект.

Често и в зависимост от формата на моделиране, която се опитвате да постигнете - обектите от реалния свят се сканират в софтуера чрез устройство за 3D сканиране; след това тези обекти се използват като дигитална проследяваща хартия за създаване на крайния модел, използвайки същия процес, споменат по-горе. Веднъж създадени, тези обекти могат да се мащабират и манипулират, както потребителят намира за добре.

1.4.1. Моделиране на NURBS

NURBS означава неравномерно рационално основание.

Моделът NURBS е тип математическо моделиране, често използван за генериране на криви и повърхности. Основните предимства на тази техника за моделиране са голямата гъвкавост и прецизност, които имате при генерирането на вашите фоми.



(Фиг. 1.4.1.1)

1.4.2. Моделиране на многоъгълници

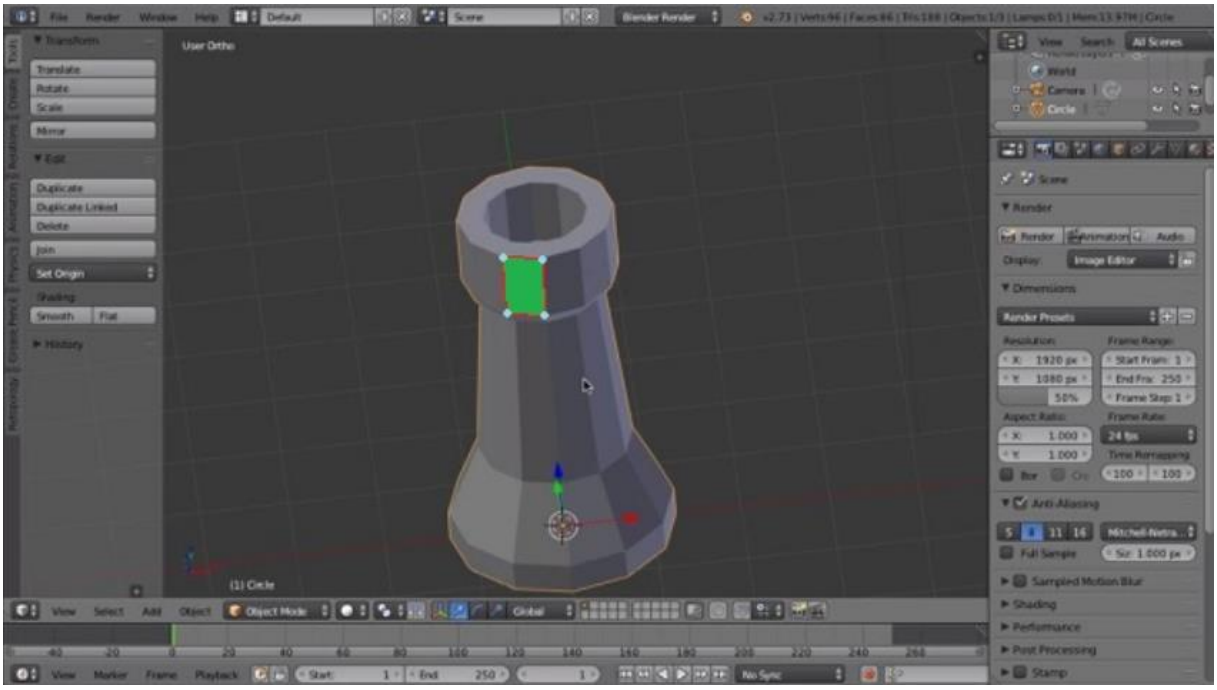
Един добър начин за въвеждане на модели на многоъгълник е да ги сравните с модели на NURBS: NURBS е математически модел, докато моделите на многоъгълник (известни също като мрежи) са колекция от върхове, ръбове и лица, които определят модела, което позволява лесно и прецизно редактиране на части от вашия обект. Чрез промяна на координатите на един или няколко върха можете да промените формата на модела.

Лица: Лицето е най-основната част от 3D полигона. Когато три или повече ръба са свързани заедно, лицето е това, което запълва празното пространство между ръбовете и съставя това,

което се вижда на полигонната мрежа. В зелено е оцветено едно лице по-долу на фигурата (Фиг. 1.4.2.1.).

Ръбове: Ръбът е друг компонент на многоъгълник. Ръбовете помагат да се определи формата на моделите, но те могат да се използват и за тяхното преобразуване. Ръбът се определя от два върха в крайните им точки. Ръбовете на един многоъгълник са изобразени в червено на изображението по-долу (Фиг. 1.4.2.1).

Върхове: Върхът е най-малкият компонент на модел на многоъгълник. Това е просто точка в триизмерно пространство. Чрез свързване на множество върхове с ръбове можете да създадете многоъгълник. Тези точки могат да бъдат манипулирани, за да се създаде желаната форма. Малките светлосини точки представляват четири върха. (Фиг. 1.4.2.1)



(Фиг. 1.4.2.1)

1.4.3. Подразделение на повърхности / NURMS моделиране

Подразделните повърхности, които понякога се наричат NURMS (неравномерна рационална гладка мрежа) - това е метод, използван за изглаждане на тези пикселизирани мрежи.

Поделените повърхности използват алгоритъм, за да вземат геометрията на многоъгълника и да я изгладят автоматично. Те

всъщност подразделят всяко многоъгълно лице на по-малки лица, които по-добре приближават гладката повърхност. Например, на изображението по-долу можете да видите една и съща фигура с по-гладки повърхности. (Фиг. 1.4.3.1) Това е така, защото мрежата е просто много по-подробна сега и изглежда гладка, благодарение на техниката за моделиране NURMS.**[12]**

(Фиг. 1.4.3.1)

1.4.4. Многоъгълна мрежа(polygon mesh)

В компютърната графика многоъгълна мрежа е колекцията от върхове, ръбове и лица, които съставляват 3D обект. Полигонната мрежа определя формата и контура на всеки 3D герой и обект, независимо дали ще се използва за 3D анимационен филм, реклама или видео игри.

Големите форми се изграждат от по-малки, взаимосвързани равнини - обикновено триъгълници или правоъгълници - които си пасват като 3D пъзел. Всеки връх в многоъгълната мрежа съхранява информация за координатите x, y и z. Тогава всяко лице на този многоъгълник съдържа информация за повърхността, която се използва от механизма за визуализиране за изчисляване на светлини и сенки. Полигонните мрежи могат да се използват за моделиране на почти всеки обект. Също така е възможно да се

генерират полигонни мрежи в реално време, което ги прави едновременно мощни и динамични. Като такива те се използват широко в целия свят на компютърната графика.

(Фиг. 1.4.4.1)

1.4.5. Текстура

Без текстура многоъгълната мрежа няма да изглежда по този начин.

С текстури художниците могат да променят външния вид на своите модели, за да изглеждат като всякакви различни повърхности. И чрез използване на UV координати текстурите могат да се прилагат върху повърхността на моделите. Чрез внимателно картографиране на местата на окото върху изображение, художниците могат да дадат на всяка равнина уникална текстура. Тази UV карта може да се въведе във всяка програма за обработка на изображения и да се редактира с подробности, необходими за проекта.

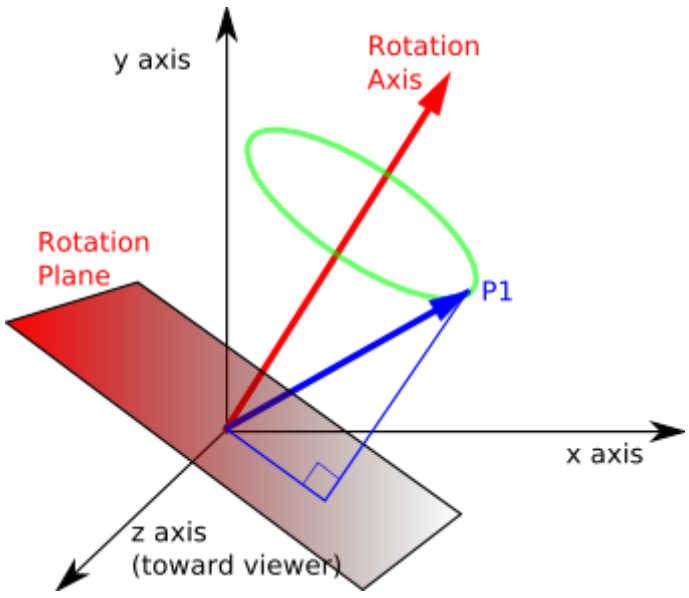
Възможно е също така да комбинирате различни текстури в някои програми. Например, чрез наслагване на цветна карта с огледална карта, можете да придадете вид на повърхностни детайли. Повечето търговски продукти са изградени с помощта на CAD и друг софтуер за създаване на мрежи, което прави всичко много по-лесно.**[13]**

(Фиг. 1.4.5.1)

1.4.6. Кватерниони

Кватернионът(quaternion) представлява ъглово въртене около оста; или по-технически, ротационната трансформация на пространство от една референтна рамка в друга. На практика кватерионът ни дава начин да завъртим вектор около ос под някакъв ъгъл. Кватернионите не страдат от бързина.

Кватернион може да бъде конструиран като ъгъл и ос на въртене или чрез отливане на други обекти на въртене. За нас не е важно да разберем вътрешната математика, свързана с използването на кватерниони. Вместо това тук се грижим само за приложението му към ротациите.**[14]**



(Фиг. 1.4.6.1)

1.5. Приложно-програмен интерфейс

Приложно-програмен интерфейс (API) е набор от програмен код, който позволява предаване на данни между един софтуерен продукт и друг. Той също така съдържа условията на този обмен на данни. API се състои от два компонента:

- Техническа спецификация, описваща опциите за обмен на данни между решенията със спецификацията, направена под формата на заявка за обработка и протоколи за доставка на данни
- Софтуерен интерфейс, написан по спецификацията, която го представлява

API служат за множество цели. Като цяло те могат да опростят и ускорят разработването на софтуер. Разработчиците могат да добавят функционалности от други доставчици към съществуващи решения или да създават нови приложения, използвайки услуги от трети страни доставчици. Във всички тези случаи специалистите не трябва да се занимават с изходния код, опитвайки се да разберат как работи другото решение. Те просто свързват своя софтуер с друг. С други думи, API служат като абстрактен слой между две системи, скривайки сложността и работните детайли от последната. **[15]**

1.6. Системи за автоматизирано и компютърното проектиране

Софтуерът, който трябва да използвате, когато проектирате 3D обект, изцяло зависи от това, което се опитвате да направите. CAD софтуерът обикновено се използва при създаване на индустриални обекти като механични предмети. От друга страна, някои CAD софтуери позволяват по-голяма артистична свобода, тъй като дизайните не трябва да работят механично, да са функционални или да отговарят на устройство от реалния свят. В исторически план софтуерът за 3D моделиране се използва във филмови анимации и видео игри за изработване на органични дизайни. Той обаче може да се използва и за създаване на 3D модели за печат.

Тук се фокусираме върху софтуера CAD (Computer Aided Design) за механични обекти. Софтуерът може да бъде изключително специфичен, той е проектиран да бъде технически инструмент с

функции в индустриалния дизайн, механичния дизайн, архитектурата и области като аерокосмическото инженерство и астронавтиката. CAD моделът ще съдържа данни като свойства на материала, размери, измерения и информация за конкретния производствен процес. Освен това много CAD приложения предлагат усъвършенствани възможности за рендиране и анимация, за да визуализират по-добре дизайна на продукта.**[16]**

1.7. Уеб приложение

Уеб приложенията са изключително популярни в днешно време и са един от най-използваните софтуерни продукти. Причината за това е, че не е необходимо те да бъдат изтеглени на нашето устройство, за да можем да ги използваме. Необходимо ни е само устройството, което използваме да има връзка с интернет и уеб браузър, без значение дали това е настолен компютър, лаптоп, таблет или смартфон. Лесно е тези условия да бъдат изпълнени предвид, че живеем в XII век.

Уеб приложението е компютърна програма, която се съхранява на отдалечен сървър и се предоставя на потребителя през интернет, чрез интерфейса на браузъра. Според редактора на “Web.AppStorm” Джарел Ремик, всеки компонент на уебсайта, който изпълнява някаква функция за потребителя, се квалифицира като уеб приложение.

Повечето уеб приложения са написани на програмни езици като JavaScript, HTML5 или Cascading Style Sheets (CSS). **[17]**

За разлика от десктоп приложенията, уеб приложенията нямат нужда от инсталация на компютър и не са зависими от операционната система на потребителя. Могат да се ползват по всяко време, навсякъде, на всякакви устройства, които имат връзка с интернет. След като бъде разработена и качена нова версия на уеб сървъра, всички потребители могат да получат достъп до нея веднага, без нужда от обновяване. Ако приложението изисква повече ресурси, това засяга единствено хардуера на сървъра. Уеб апликациите предлагат по-голяма сигурност, тъй като нямат нужда от поддръжка на многобройни клиентски компютри, а единствено от мониторинг на уеб сървъра. Уеб базираният софтуер е по-лесен за ползване, разработка, поддръжка и надграждане. Почти всяка програма може да бъде реализирана като уеб базирано приложение.**[18]**

Глава Втора

Изисквания към уеб приложението. Проектиране на структурата на уеб базирания инструмент за брандиране на 3D модели.

2.1. Изисквания към уеб приложението

2.1.1. Да се зарежда сцена, с възможност за добавяне на 3D модел и 3D лого

Всеки потребител трябва да може да отвори в браузъра си уеб приложението. На страницата в браузъра, се зарежда сцена, с добавени на нея светлина и камера, която ни позволява да разглеждаме модела от различни гледни точки, на която има възможност да се качват 3D модел и 3D лого. За да качваме 3D модели трябва да имаме бутон за избор на запаметен файл от устройството ни, след като намерим избрания файл в паметта на устройството ни, го селектираме и отваряме. За да бъде възможно това качване, чрез бутоните, трябва да подсигурим качването на 3D файлов формат, в случая това ще бъде OBJ и BABYLON.

2.1.2. Да се запаметява редактирана сцена и да се зарежда

След направени промени по сцената, трябва да се добави възможност за запазване на редактирана сцена. Това ще се случва чрез бутон за изтегляне, а вече изтеглената композиция може да се зарежда отново, чрез бутона за качване. При качването и намираме запаметения файл в устройството, след това го селектираме и отваряме отново. При зареждането на композицията отново на сцената, тя може да бъде редактирана и свалена наново.

2.1.3. Да се местят и завъртат моделите

На сцената 3D обектите могат да се местят и завъртат, това е редакцията на позицията. Трябва да се добави възможност моделите да се местят и по трите си оси (x, y, z). Местенето на избрания модел ще става чрез хващане, задържане и теглене до желаната нова позиция. Завъртането на модела ще става по x и y оста, чрез плъзгачи за двете оси, а движението на индикатора ще променя стойностите за x и y . За да е възможна обаче тази редакция на позицията и ротацията, трябва да имаме възможност за селектиране на моделите, чрез хващане на събития от мишката, местим или завъртаме последно избраният от нас модел на сцената.

2.1.4. Да се изкривява плоско лого

За изкривяването на плоско лого ще ни трябва допълнителен бутон за качване на логото, чрез който избираме запаметен файл от устройството ни, след като намерим избрания файл в паметта на устройството ни, го селектираме и отваряме. За изкривяването на каченото лого ще ни трябва плъзгач, с който колкото повече местим индикатора, толкова повече се изкривява логото.

2.1.5. Да се долепят два модела

Долепянето на двата меша може да става ръчно чрез местенето на логото върху модела и поставянето му отгоре и в самия модел. След като това е направено и двата меша изглеждат като един общ, трябва да имаме инструмент, който да съединява двата меша в един.

2.1.6. Да се съединяват два модела в един

След като се долепят два модела, трябва да имаме възможността да обединим двата им меша в един. Така при свалянето на сцената, двата меша(модела и логото) ще бъдат обединени в един общ меш.

2.1.7. Да се генерира 3D модел за парче текст

Генерирането на триизмерен модел за парче да става,чрез изписване на името, което ни е логото, в определено за това поле и от този текст да се генерира триизмерен модел.

2.1.8. Да се залепва автоматично логото върху една или повече повърхнини

Чрез допълнителен бутон, трябва да можем да слагаме логото на n на брой повърхнини. С този инструмент се опитваме да позиционираме автоматично вече каченото лого върху качения модел.

2.2. Аргументация на избора на средите за разработка

2.2.1. HTML5

Избрах HTML, защото той осигурява структурата на една уеб страница.

Hypertext Markup Language или HTML се използва за описване на структурата на информацията на една уеб страница. Затова HTML, CSS и JavaScript съставят съществените градивни елементи на уебсайтовете, като CSS контролира външния вид на страницата, а JavaScript нейната функционалност.

2.2.2. CSS

Избрах CSS, защото той контролира външния вид на едно уеб приложение. Той ни осигурява бърза стилизация на желаните от нас елементи.

2.2.3. JavaScript

За основен програмен език за разработване на уеб приложението избрах JavaScript.

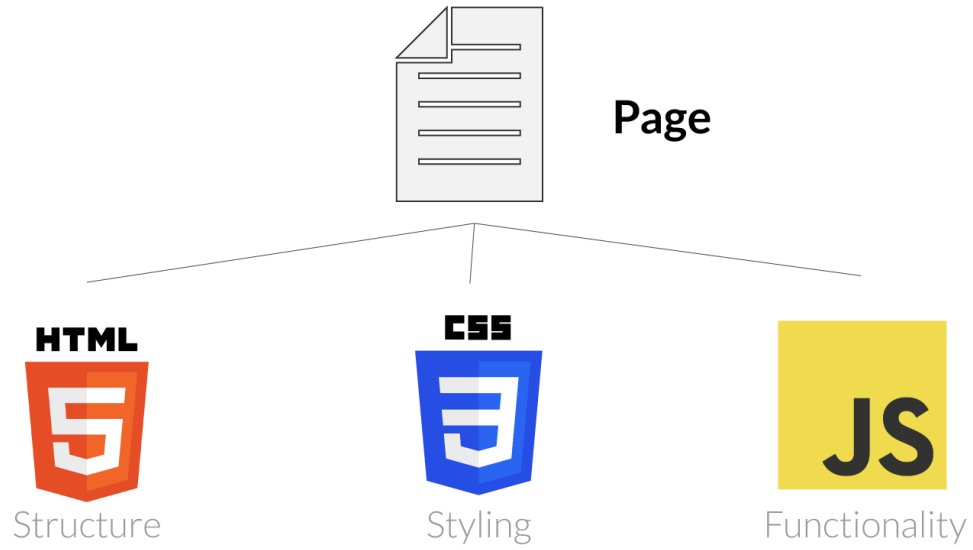
Откакто е пуснат, JavaScript е най-популярният език за програмиране за уеб разработка. В момента той е използван от 94% от всички уебсайтове.

JavaScript е клиентски език за програмиране, който помага на уеб разработчика да разработва уеб приложения и да създава динамични и интерактивни уеб страници, чрез внедряване на персонализирани скриптове от страна на клиента.

Разработчиците могат да създават уеб страници, които работят добре в различни браузъри, платформи и устройства, като комбинират JavaScript, HTML и CSS.**[19]**

Причините да избера този език са няколко:

- Голямо предимство на JavaScript е, че за него съществуват много софтуерни рамки, които намаляват значително времето и усилията, за разработването на JS базирани сайтове и приложения.
- Друго голямо предимство, едно от най-хубавите неща, поради които избрах JavaScript е, че той поддържа множество уеб браузъри като Google Chrome, Internet Explorer, Safari, Firefox и Opera. Това улеснява достъпа на потребителите до уеб приложенията във всеки избран от тях браузър.
- Множеството библиотеки и рамки, които са разработени за JS, също са от голяма помощ. В разработката на моето приложение без използването на една такава рамка – Babylon.js, реализацията му щеше да е безуспешна.



(Фиг. 2.2.3.1)**[20]**

2.2.4. Babylon.js

Избрах Babylon.js за JS рамка, която да ми помага с визуалицацията на 3D елементите в уеб приложението.

Babylon.js е рамка, която капсулира възможностите на WebGL, и улеснява изграждането на 3D приложения. Базирана е на WebGL и JavaScript.

Избрах тази рамка, защото е лесна за използване и предоставя всякакви инструменти за създаване и управляване на 3D обекти. Тя е една от най-популярните сред разработчиците в 3D сферата и осигурява много вградени функции.

2.2.5. OBJ

Избрах да използвам OBJ файлов формат.

Причините поради, кото избрах този формат са няколко:

- Той е част от неутралните файлови формати. Това означава, че е междинен формат за конвертиране между два патентовани формата.
- Изключително популярен е.
- Поддържа както приблизително, така и прецизно кодиране на геометрията на повърхността.

2.2.6. Visual Studio Code

Избрах Visual Studio Code за развойна среда.

Причините са доста:

- Той съчетава простотата на редактора на изходния код с много мощни инструменти, което го прави изключително лесен за използване. Редактирането на грешките става лесно.
- Предлага се за macOS, Linux и Windows, ми създава удобството да работя по проекта си и на двете операционни системи, които използвам.
- Има много полезни клавишни комбинации, които мога да персонализирам.
- Има поддръжка за Git.
- Има много добри инструменти за уеб разработчици и най-удобното му нещо за мен е неговият live server, с който визуализацията на кода и промените по него стават в реално време, запазването е автоматично и не се налага всеки път преди да се пуска наново цялото приложение, след ръчно запазване.

Глава Трета
Разработка на уеб базиран инструмент за брандиране на 3D модели

3.1. Създаване на файлове

Отварям VS code и създавам нов файл, който се казва по подразбиране index.html, съдържащ структурата на един HTML

файл. В него импортирам всички Babylon.js рамки, които ще са ми необходими, в head частта (Фиг. 3.1.1). В следващата стъпка добавям в body частта елемента <canvas>, той е част от HTML5 и позволява динамичното изобразяване на скриптове на 2D форми и растерни изображения (Фиг. 3.1.2). Тогава в същата папка, където се намира проекта ми и index.html файла, създавам main.js файл. Точно след <canvas>, добавям <script> таг, който се използва за дефиниране на скрипта от страна на клиента (JavaScript) (Фиг. 3.1.2). Този елемент <script> сочи към външния файл main.js, чрез атрибута src. И последната стъпка е да стилизирам проекта, чрез style.css файл, който също се намира в същата папка. В head частта добавям <link> таг, в който подавам style.css файла (Фиг. 3.1.1).

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

    <title>Diplomna-Violeta</title>

    <!-- Babylon.js -->
    <script src="https://code.jquery.com/pep/0.4.2/pep.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/dat-gui/0.6.1/dat.gui.min.js"></script>
    <script src="https://preview.babylonjs.com/ammo.js"></script>
    <script src="https://preview.babylonjs.com/cannon.js"></script>
    <script src="https://preview.babylonjs.com/Oimo.js"></script>
    <script src="https://preview.babylonjs.com/earcut.min.js"></script>
    <script src="https://preview.babylonjs.com/babylon.js"></script>
    <script src="https://preview.babylonjs.com/materialsLibrary/babylonjs.materials.min.js"></script>
    <script src="https://preview.babylonjs.com/proceduralTexturesLibrary/babylonjs.proceduralTextures.
min.js"></script>
    <script src="https://preview.babylonjs.com/postProcessesLibrary/babylonjs.postProcess.min.js"></scr
ipt>
    <script src="https://preview.babylonjs.com/loaders/babylonjs.loaders.js"></script>
    <script src="https://preview.babylonjs.com/serializers/babylonjs.serializers.min.js"></script>
    <script src="https://preview.babylonjs.com/gui/babylon.gui.min.js"></script>
    <script src="https://preview.babylonjs.com/inspector/babylon.inspector.bundle.js"></script>
    <script src="https://preview.babylonjs.com/loaders/babylonjs.loaders.min.js"></script>

    <link rel="stylesheet" href="style.css">

  </head>
```

(Фиг. 3.1.1. Импортиране на Babylon.js рамки и свързване на стилизиращия файл)

```
    <canvas id="renderCanvas"></canvas>

    <script src="main.js"></script>

</body>
</html>
```

(Фиг. 3.1.2. Добавяне на <canvas> елемента и дефиниране на скрипта от страна на клиента)

3.2. Основни, необходими елементи за създаване на триизмерен свят

Без значение дали използваме Babylon.js, за да създаваме цял триизмерен свят или само, за да представим един модел в уеб страница, задължително ще са ни необходими няколко неща. Трябва да имаме сцена, която да съдържа триизмерния свят или модела, камера, за да имаме възможността да виждаме, светлина, която да го осветява и поне един обект, като модел. Всеки един модел, независимо дали е само кутия или сложна фигура, е направен от мрежа от триъгълници или фасети и се нарича меш. От това следва, че първото нещо, което трябва да направим в main.js файла, щом използваме Babylon.js е да създадем сцена с добавена камера и светлина и чак тогава можем да започнем да добавяме обекти.

3.2.1. Canvas

Първото, което правя е да взема <canvas>елемента от index.html файла с помащта на JS метода *document.getElementById()*, това ми дава възможност вече да работя с <canvas> елемента. Той се използва, за да се изобрази съдържанието.

3.2.2. Engine

Създавам engine, като му предавам елемента <canvas>, който да се изобрази. Класът engine е отговорен за взаимодействието с всички API-та на по-ниско ниво, като WebGL.

3.2.2.1. Глобален обект BABYLON

Глобалният обект BABYLON съдържа всички функции на Babylon.js, които са налични в engine.

3.2.3. Scene

Сцената е мястото, където се показва цялото съдържание на приложението. Докато създавам нови обекти в демото ще ги добавям на сцената, така че да са видими на екрана. Създаването на сцена става на: (Фиг.3.2.3.1.).

```
var canvas = document.getElementById("renderCanvas");
var engine = null;
var scene = null;
var sceneToRender = null;
var assetsManager = null;
var createDefaultEngine = function () { return new BABYLON.Engine(canvas, true, {
  preserveDrawingBuffer: true,stencil: true, disableWebGL2Support: false }); };
var createScene = function () {

// This creates a basic Babylon Scene object (non-mesh)
  var scene = new BABYLON.Scene(engine);
```

(Фиг.3.2.3.1. Създаване на сцена)

А всичко, което създавам и искам да визуализирам на сцената влиза във функцията *createScene* , в чиито край се връща всичко вкарано вътре и се визуализира.

3.2.4. BABYLON.Vector(3)

BABYLON.Vector3() е метод, който определя 3D позицията на сцената. Babylon.js е в комплект с пълна математическа библиотека за работа с вектори, цветове, матрици и др.

3.2.5. Внедряване на стандартните компоненти на сцената

3.2.5.1. Камера

Започвам с камерата и добавям Arc Rotate Camera – която действа като сателит в орбита около някаква цел и винаги сочи към позицията на целта. За да инициализирам камерата и подавам името, с което искам да я използвам, координатите, където ще бъде тя в 3D пространството, и сцената, към която я добавям. Камерата има допълнителни функционалности за задаване на позиция, цел и контролиране на мястото, към което сочи камерата.

3.2.5.2. Светлина

В Babylon.js има различни източници на светлина.Основният източник на светлина в моя случай избрах да е HemisphericLight, тя симулира светлината на околната среда, така че преминалата посока е посоката на отражение на светлината, а не на директно входящата. Параметрите на светлината са нейното име, позицията и в 3D пространството и сцената, към която е добавена светлината. Като допълнителна характеристика на светлината съм добавила и интензитет на излъчване на светлината.

```
var canvas = document.getElementById("renderCanvas");
var engine = null;
var scene = null;
var sceneToRender = null;
var assetsManager = null;
var createDefaultEngine = function () { return new BABYLON.Engine(canvas, true, {
  preserveDrawingBuffer: true,stencil: true, disableWebGL2Support: false }); };
var createScene = function () {

// This creates a basic Babylon Scene object (non-mesh)
  var scene = new BABYLON.Scene(engine);

  // This creates and initially positions a arc rotate camera
  var camera = new BABYLON.ArcRotateCamera("Camera", 0, 0, 10, new BABYLON.Vector3(0, 0, 0), sc
ene);




  // This positions the camera
```

```
camera.setPosition(new BABYLON.Vector3(0, 3, -11));

//camera.target is set after the target's creation// This targets the camera to scene origin
camera.setTarget(BABYLON.Vector3.Zero());

// This attaches the camera to the canvas
camera.attachControl(canvas, true);

// This creates a light, aiming 0,1,0 - to the sky (non-mesh)
var light = new BABYLON.HemisphericLight("light", new BABYLON.Vector3(0, 10, 0), scene);

// Default intensity is 1. Let's dim the light a small amount
light.intensity = 0.7;
```

(Фиг. 3.2.5.1. Създаване на сцена, добавяне на камера и светлина)

Когато сцената е вече готова и прави правилно изобразяване, мога да започна да добавям 3D фигури към нея.

3.3. Добавяне на допълнителни елементи към сцената

3.3.1. Добавяне на меш

В Babylon.js създаването на меш става, чрез импортирането на модели от мрежи, създадени с помощта на друг софтуер. За да ускори разработката, Babylon.js предоставя много базови фигури, които могат да се използват, за да се създават цели фигури в един ред код. Налични са кубчета, сфери, цилиндри и по-сложни форми. Мешът е начин, по който engine-а създава геометрични фигури, така че материалът да може лесно да бъде нанесен върху тях по-късно. В моя случай създавам няколко сфери и следвам синтаксиса за създаване на сфера с помощта на meshbuilder.

Параметрите на сферата са:

- Диаметър – това е диаметърът на сферата, чиято стойност по подразбиране е 1.

- ДиаметърX – диаметърът по оста X замества свойството диаметър. Ако не е посочено, използва свойството диаметър
- ДиаметърУ – диаметърът по оста У замества свойството диаметър. Ако не е посочено, използва свойството диаметър.
- ДиаметърZ –диаметърът на оста Z, замества свойството диаметър. Ако не е посочено, използва свойството диаметър.
- Arc – е съотношението на обиколката, между 0 и 1.
- Slice – това е съотношението на височината между 0 и 1.
- Boolean – булевата стойност е вярна, ако меша може да се актуализира. По подразбиране е false – невярно.
- SideOrientation – това се отнася до страничната ориентация.

Включването на всичките параметри не е задължително. Допълнително може да се зададе позицията на меша.

3.3.1.1. Материи

Към 3D обектите можем да добавяме материи. Те ни позволяват да покрием мешовете в цвят и текстура. Начинът, по който се появява материалът, зависи от светлините, използвани в сцената, и от това как е настроен да реагира. Съществуват четири възможни начина материалите да реагират на светлината и аз съм избрала да използвам Ambient(атмосфера) – цветът или текстурата на материала е осветена от фоновото осветление на околната среда. Той се прилага само ако е зададен цветът на околната среда на сцената.

3.3.1.2. Текстура

Текстурите се формират с помощна на запазено изображение. Първо се създава материал, а след това се задава текстурата на материала, в моя случай с използването на ambientTexture, той се прилага без да се използва цветът на околната среда на сцената.

```
var color = new BABYLON.StandardMaterial("color", scene);
color.emissiveTexture = new BABYLON.Texture("textures/color.jpg", scene);
```

```
var sphere = BABYLON.MeshBuilder.CreateSphere("sphere", { diameter: 2, segments: 32 }, scene);
sphere.position.y = 1;


sphere.position.x = 0;
sphere.position.z = 0;
sphere.material = color;
```

(Фиг. 3.3.1.1.1. Създаване на сфери и задаване на текстурата и материала им)

3.3.2. Добавяне на земя

Земята е плоска хоризонтална равнина, успоредна на равнината xz, която може да бъде подразделена на правоъгълни области. Началото на земята е центъра на равнината. Незадължителните свойства за определяне на размера на земята са ширина (x) и височина (z).

Подобно на създаването на сфери е и създаването на земя и на нея също може да и бъде зададен цвят.

```
var ground = BABYLON.MeshBuilder.CreateGround("ground", { width: 13, height: 13 }, scene);
const material_ground = new BABYLON.StandardMaterial('material_ground', scene);
material_ground.diffuseColor = new BABYLON.Color3(153/255, 255/255, 230/255);
ground.material = material_ground
```

(Фиг. 3.3.2.1. Създаване на земя)

3.4. Добавяне на допълнителни функционалности към сцената

3.4.1. Селектиране

```
var selected = null;
  scene.onPointerObservable.add(function(evt){
    if(selected) {
      //selected.material.diffuseColor = BABYLON.Color3.Gray();
      selected = null;
    }
    if(evt.pickInfo.hit && evt.pickInfo.pickedMesh && evt.event.button === 0){
      selected = evt.pickInfo.pickedMesh;
```

```
        //evt.pickInfo.pickedMesh.material.diffuseColor = BABYLON.Color3.Green();
    }
}, BABYLON.PointerEventTypes.POINTERUP);


var startingPoint;
var currentMesh;

var getGroundPosition = function () {
    var pickinfo = scene.pick(scene.pointerX, scene.pointerY,
      scene.pointerZ, function (mesh) { return mesh == ground; });
    if (pickinfo.hit) {
        return pickinfo.pickedPoint;
    }

    return null;
}
```

(Фиг. 3.4.1.1. Селектиране)

Селектирането на модели ни е необходимо, за да можем да контролираме всеки един модел от сцената, който пожелаем. То е, за да се осъществи редакцията на позицията и ротацията на мешовете. За да е възможно това трябва да хванем събитията от мишката.

Обектът на сцената на Babylon.js има над 20 observables, които действат при различни условия. Observable е свойството на обект, който представлява дадено събитие. Имплементатора използва Observable, за да създаде свойство, което ще задейства всички регистрирани наблюдатели. Повечето от тях се проверяват на всеки кадър или в предсказуем ред, или последователно. Observable е този, който проверява какво се случва с показалеца на екрана, независимо дали с мишка, с пръст или контролер. Това което е най-полезно за селектирането е scene.onPointerObservable.

Намирането на точката, която потребителят е докоснал в пространството, се извършва с помощта на обекта PickingInfo, изпращан от всяко събитие на пойнтера нагоре и надолу. Обектът PickingInfo съдържа информация за точката, в която е задействано събитието, включително меша, който е бил докоснат, точката на меша, която е била докосната.

```
        var pointerDown = function (mesh) {
    currentMesh = mesh;
```

```
startingPoint = getGroundPosition();
    if (startingPoint) { // disconnecting camera from canvas
        setTimeout(function () {
            camera.detachControl(canvas);
        }, 0);
    }
}

  var pointerUp = function () {
    if (startingPoint) {
        camera.attachControl(canvas, true);
        startingPoint = null;
        return;
    }
  }

  var pointerMove = function () {
    if (!startingPoint) {
        return;
    }
    var current = getGroundPosition();
    if (!current) {
        return;
    }

    var diff = current.subtract(startingPoint);
    currentMesh.position.addInPlace(diff);

    startingPoint = current;

 }
```

(Фиг. 3.4.1.1. Селектиране)

Babylon.jsима две полезни функции за обратно извикване, които ще ми помогнат да получа информацията за избор: onPointerUp и OnPointerDown. Тези две обратни извиквания се задействат, когато се задействат събития с указатели.

Променливата evt е първоначалното събитие на JS, което е задействано. PickInfo е информацията за избор, генерирана от рамката за всяко задействано събитие.

```
  scene.onPointerObservable.add((pointerInfo) => {
```

```
  switch (pointerInfo.type) {
    case BABYLON.PointerEventTypes.POINTERDOWN:
      if(pointerInfo.pickInfo.hit && pointerInfo.pickInfo.pickedMesh != ground) {
                pointerDown(pointerInfo.pickInfo.pickedMesh)
              }
      break;

    case BABYLON.PointerEventTypes.POINTERUP:
                pointerUp();
      break;
    case BABYLON.PointerEventTypes.POINTERMOVE:
                pointerMove();
      break;
      }
  });
```

(Фиг. 3.4.1.2. Селектиране)

3.4.2. Местене – PointerDragBehavior

Поведението на мешовете е поведение, което може да бъде отнесено към мешовете. PointerDragBehavior се използва за плъзгане и местене на меша около равнина или ос с помощта на мишката или vr контролер. В моя случай използвам плъзгане по осите – dragAxis. По подразбиране оста на плъзгане ще бъде променена от ориентацията на обектите, за да запазим посочената ос го задаваме на false.

```
var pointerDragBehavior = new BABYLON.PointerDragBehavior({ dragAxis: new BABYLON.Vector3(1
, 0, 0) });
 var pointerDragBehavior = new BABYLON.PointerDragBehavior({ dragYxis: new BABYLON.Vector3(
0, 1, 0) });
 var pointerDragBehavior = new BABYLON.PointerDragBehavior({ dragZxis: new BABYLON.Vector3(
0, 0, 1) });

 // Use drag plane in world space
 pointerDragBehavior.useObjectOrientationForDragging = false;

 // Listen to drag events
 pointerDragBehavior.onDragStartObservable.add((event) => {
  console.log("dragStart");
  console.log(event);
 })
 pointerDragBehavior.onDragObservable.add((event) => {
```

```
console.log("drag");
  console.log(event);
 })
 pointerDragBehavior.onDragEndObservable.add((event) => {
  console.log("dragEnd");
  console.log(event);
 })

 sphere0.addBehavior(pointerDragBehavior);
```

(Фиг. 3.4.2.1. Местене по трите оси)

3.4.3. Ротация

За ротацията ми помага GUI библиотеката на Babylon.js, тя е разширение, което се използва за генериране на интерактивен потребителски интерфейс и е изграден върху DynamicTexture.

Babylon.js използва DinamicTexture, за да генерира напълно функционален потребителски интерфейс, който е гъвкав.

Първото нещо, което ми е необходимо при използването на Babylon.GUI е обекта AdvancedDynamicTexture. В моя случай аз използвам режима на Babylon.GUI за целия екран, той ще покрие целия екран и мащабира винаги, за да се адаптира към моята резолюция. Той също така прихваща кликвания и докосвания. За да създам AdvancedDynamicTexture изпълнявам (Фиг. 3.4.3.1.).

```
var advancedTexture = BABYLON.GUI.AdvancedDynamicTexture.CreateFullscreenUI("UI");
```

(Фиг. 3.4.3.1. Създаване на GUI на цял екран)

Следващото нещо, което ще ни е необходимо е StackPanel. StackPanel е контрол, който подрежда своите деца въз основа на ориентацията му, в моя случай е хоризонтална. На всички деца трябва да дефинирам ширина и височина, които са в пиксели. Аз избрах по хоризонтала да бъде центриран от дясната страна на сцената, а по вертикала в средата

Крайния резултат е показан на (Фиг. 3.4.3.2. Създаване на StackPanel)

```
var panel = new BABYLON.GUI.StackPanel();
panel.height = "200px";
panel.width = "220px";
panel.horizontalAlignment = BABYLON.GUI.Control.HORIZONTAL_ALIGNMENT_RIGHT;
panel.verticalAlignment = BABYLON.GUI.Control.VERTICAL_ALIGNMENT_CENTER;
advancedTexture.addControl(panel);
```

(Фиг. 3.4.3.2. Създаване на StackPanel)

След като създадох StackPanel използвам TextBlock, той е елемент, който служи за управление на текста, който изобразявам. Създавам го и му задавам му име, височина и цвят, както е показано на (Фиг. 3.4.3.3. Създаване на TextBlock)

```
var header = new BABYLON.GUI.TextBlock();
header.text = "Y-rotation: 0 deg";
header.height = "30px";
header.color = "white";
panel.addControl(header);
```

(Фиг. 3.4.3.3. Създаване на TextBlock)

И последното нещо, което правя за ротацията на обектите е създаването на Slider (плъзгач). Плъзгачът се използва за управление на стойност в определен диапазон. Това, което е необходимо е да се зададе максимум и минимум на този диапазон (slider.maximum, slider.minimum). Самата стойност посочвам в slider.value и повиши observable при всяка промяна на (slider.onValueChangedObservable). Този елемент е показан на (Фиг. 3.4.3.4. Създаване на Slider)

```
var slider = new BABYLON.GUI.Slider();
slider.minimum = 0;
slider.maximum = 2 * Math.PI;
slider.value = 0;
slider.height = "20px";
slider.width = "200px";
slider.onValueChangedObservable.add(function(value) {
    header.text = "Y-rotation: " + (BABYLON.Tools.ToDegrees(value) | 0) + " deg";
    if (currentMesh) {
        currentMesh.rotation.y = value;
```

```
    }


});
panel.addControl(slider);
```

(Фиг. 3.4.3.4. Създаване на Slider)

3.4.4. Цветове

Избирането на цветове става по същия начин като ротацията. Единствената разлика е, че тук имаме ColorPicker на мястото на плъзгача. Контролът по избор на цвят ми позволява да задавам цветове на избраните от мен модели. Винаги, когато взаимодействам с този инструмент за избор на цвета, се задейства (colorPicker.onValueChangeObservable), което ми връща текущата стойност на (Color 3) на инструмента за избор на цвят. Контролът се изобразява с размерът, ширината и височината, но те са винаги еднакви, тъй като инструментът за избор на цвят може да бъде само квадрат. Този инструмент е на (Фиг. 3.4.4.1. Инструмент за избор на цвят)

```
var advancedTexture = BABYLON.GUI.AdvancedDynamicTexture.CreateFullscreenUI("UI");

var panel = new BABYLON.GUI.StackPanel();
panel.height = "600px";
panel.width = "200px";
panel.isVertical = true;
panel.horizontalAlignment = BABYLON.GUI.Control.HORIZONTAL_ALIGNMENT_RIGHT;
panel.verticalAlignment = BABYLON.GUI.Control.VERTICAL_ALIGNMENT_CENTER;
advancedTexture.addControl(panel);

var textBlock = new BABYLON.GUI.TextBlock();
textBlock.text = "Diffuse color:";
textBlock.height = "30px";
panel.addControl(textBlock);

var picker = new BABYLON.GUI.ColorPicker();
picker.value = new BABYLON.Color3(0.9,0.1,0.0);
picker.height = "150px";
picker.width = "150px";
picker.horizontalAlignment = BABYLON.GUI.Control.HORIZONTAL_ALIGNMENT_CENTER;
picker.onValueChangedObservable.add(function(value) { // value is a color3
    sphere2.material.diffuseColor.copyFrom(value);
```

```
});

panel.addControl(picker);
```

(Фиг. 3.4.4.1. Инструмент за избор на цвят)

3.4.5. Редтартиране на сцена

Методът location.reload() презарежда текущия URL адрес, като бутона Refresh. Рестартирането на сцената е показано на (Фиг. 3.4.5.1. Рестартиране на сцена)

```
document.getElementById("restart-btn").addEventListener("click",function () {
    location.reload();
  });
```

(Фиг. 3.4.5.1. Рестартиране на сцена)

3.4.6. Запаметяване на сцена

Запаметяване на сцената в компютъра се постига с помощта на scene serializer, който се използва за сериализиране на сцена в низ. Това е показано на (Фиг. 3.4.6.1. Сериализиране на сцена)

```
var serializedScene = BABYLON.SceneSerializer.Serialize(scene);
```

(Фиг. 3.4.6.1. Сериализиране на сцена)

Получения низ се конвертира в JSON низ на (Фиг. 3.4.6.2. Конвертиране на низа)

```
var strScene = JSON.stringify(serializedScene);
```

(Фиг. 3.4.6.2. Конвертиране на низа)

И вече имам JSON низ, който представлява сцената. За да мога да работя с геометричните данни, които съм получила използвам blob. Обектът Blob представлява blob, който е подобен на файл, обект от неизменяеми данни, те могат да се

четат като текст или двоични данни или да се преобразуват, за да могат методите му да се използват за обработка на данните. За да конструирам blob от srtScene-а, който получавам преди това използвам конструктора Blob() – (Фиг. 3.4.6.3. Конструиране на blob)

```
var blob = new Blob ( [ strScene ], { type : "octet/stream" } );
```

(Фиг. 3.4.6.3. Конструиране на blob)

Конструктура връща новосъздаден Blob обект, който съдържа обединените данни от масива, предаден в конструктора. След това превръщам blob в URL обект, който по-късно ще освободя, с извикването на функцията revokeObjectURL(). Статичния метод URL.createObjectURL() създава DOMString (последователност от 16-битови числа), съдържащ URL, който представлява обекта, зададен в параметъра. Новосъздадения обект URL представлява посочения Blob – (Фиг. 3.4.6.4. Blob в URL обект)

```
objectUrl = (window.webkitURL || window.URL).createObjectURL(blob);
```

(Фиг. 3.4.6.4. Blob в URL обект)

Цялото запаметяване на сцена е показано на – (Фиг. 3.4.6.5. Запаметяване на сцена)

```
var objectUrl;
 document.getElementById("save-btn").addEventListener("click",function () {
  if (confirm('Do you want to download that scene?')) {
    doDownload('scene', scene);
  } else {
    // Do nothing!
  }
 });
 var objectUrl;
 function doDownload(filename, scene) {
    if(objectUrl) {
       window.URL.revokeObjectURL(objectUrl);
    }
```

```
    var serializedScene = BABYLON.SceneSerializer.Serialize(scene);



 var strScene = JSON.stringify(serializedScene);

    if (filename.toLowerCase().lastIndexOf(".babylon") !== filename.length - 8 || filename.length < 9){
       filename += ".babylon";
    }

    var blob = new Blob ( [ strScene ], { type : "octet/stream" } );

    objectUrl = (window.webkitURL || window.URL).createObjectURL(blob);

    var link = window.document.createElement('a');
    link.href = objectUrl;
    link.download = filename;


    var click = document.createEvent("MouseEvents");
    click.initEvent("click", true, false);
    link.dispatchEvent(click);
 }
```

(Фиг. 3.4.6.5. Запаметяване на сцена)

3.4.7. Качване на триизмерен обект

За качването на файл, трябва да подадем бутона, елемента, с който ще качваме файловете, това става с id-то на бутона. При настъпила промяна се изпълнява функцията за зареждане на файлове, отново в нея използвам blob, за да мога да работя с геометричните данни.

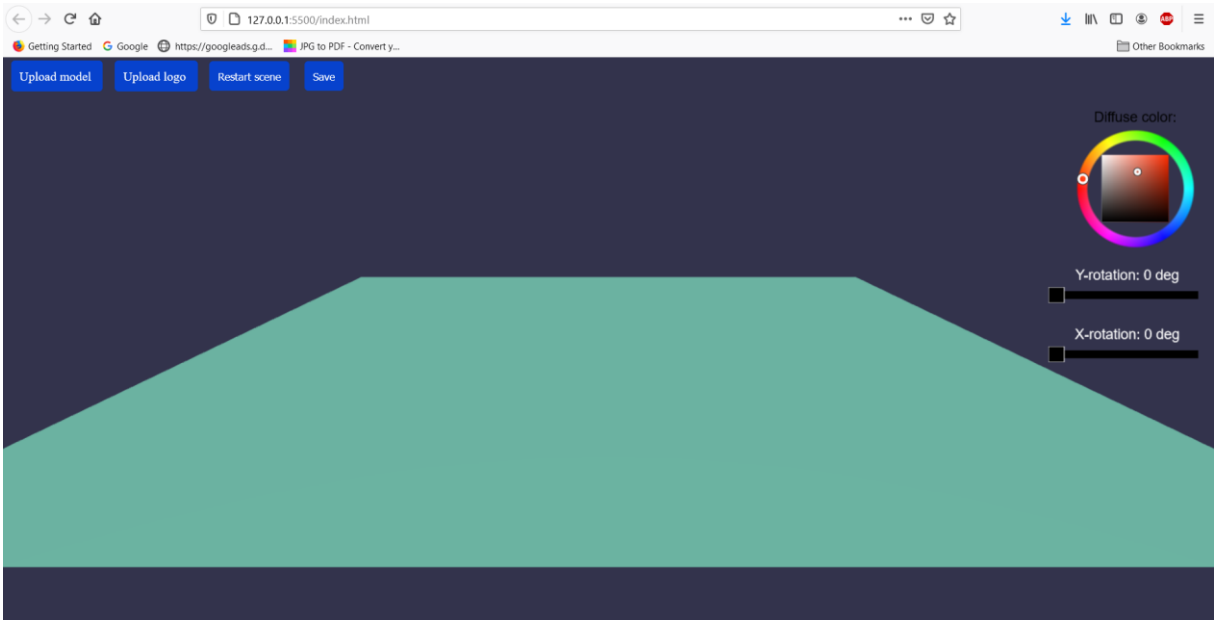
```
//Upload input for file type
const htmlInput = document.getElementById("is-local-files");

htmlInput.onchange = function (evt) {
  var files = evt.target.files;
  var filename = files[0].name;
  var blob = new Blob([files[0]]);

  BABYLON.FilesInput.FilesToLoad[filename] = blob;

  assetsManager.addMeshTask('task1', "", "file:", filename);
  assetsManager.load();
  return scene;
};
```

(Фиг. 3.4.7.1 Качване на файл)

Глава Четвърта
Ръководство на потребителя

4.1. Използване на уеб приложението за брандиране на 3D модели

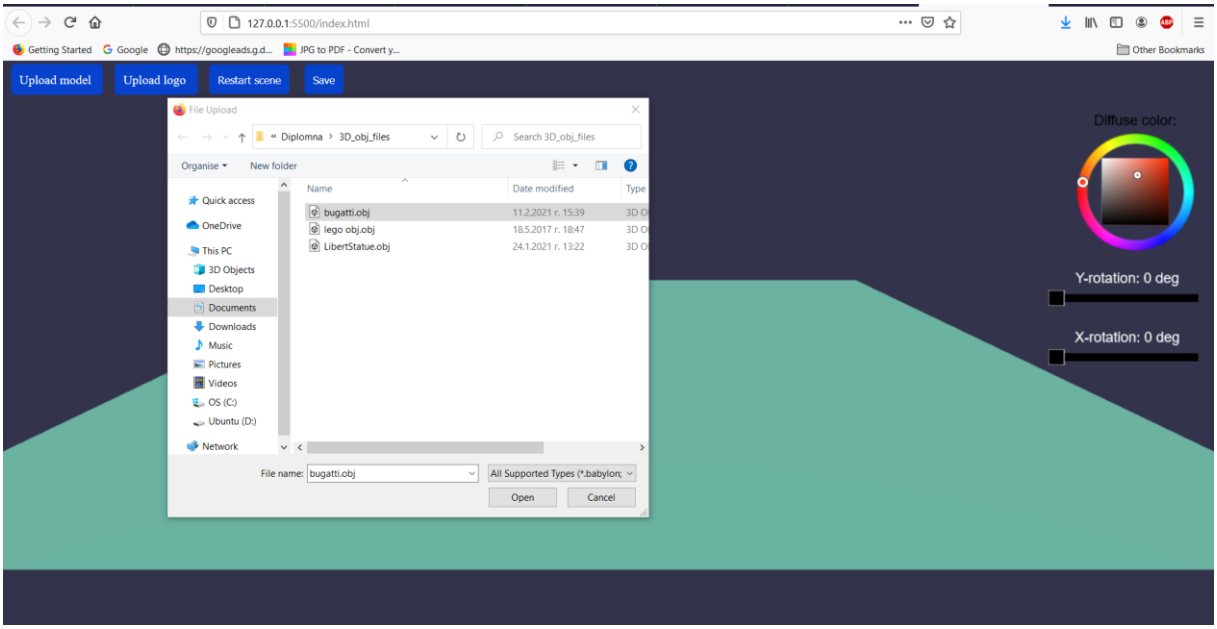
На страницата (показана на Фиг. 4.1.1.) потребителят има няколко бутона, два плъзгача, един инструмент за избор на цвят и земя.



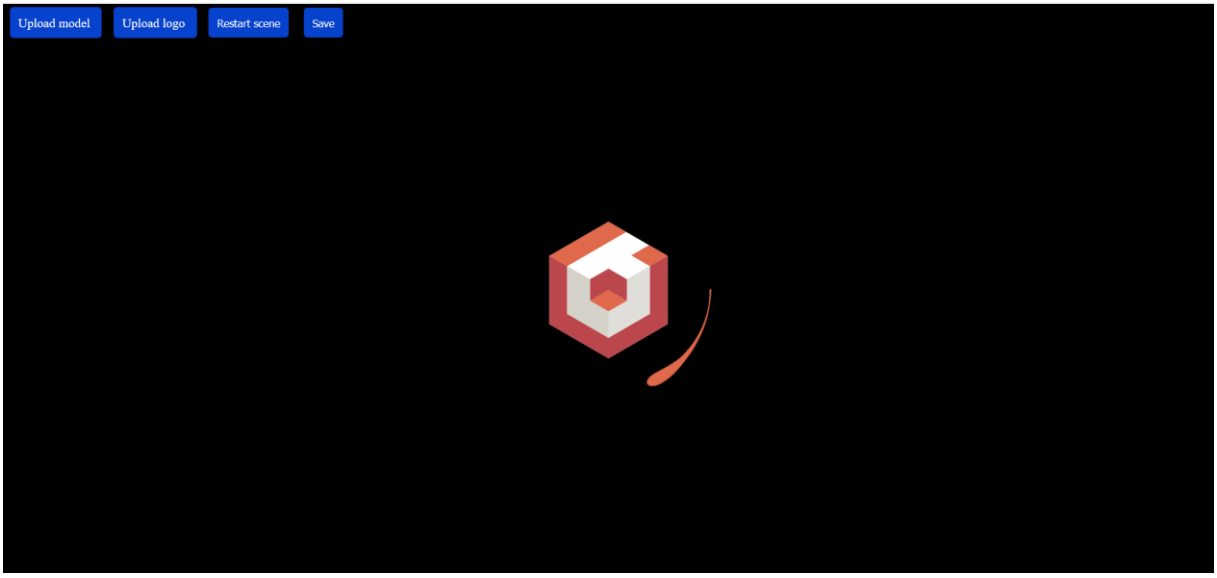
(Фиг. 4.1.1. Уеб страницата на инструмента за брандиране)

4.1.2. Бутони за качване на 3D модел и 3D лого

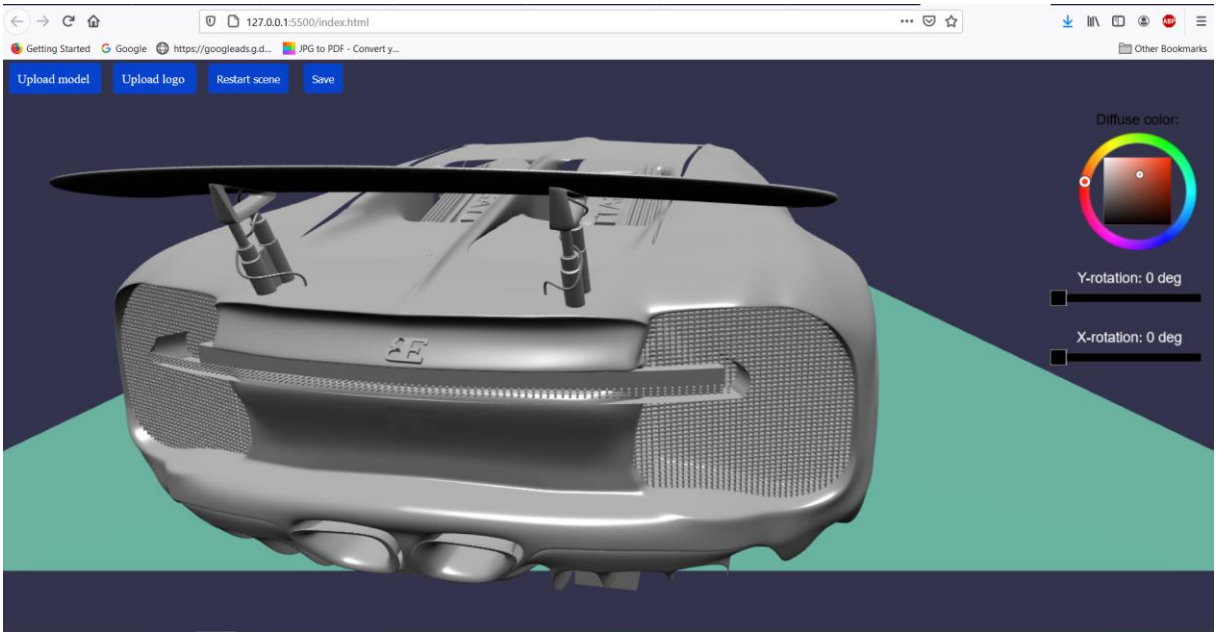
Качването на триизмерните обекти става по един и същи начин. Натиска се бутона за качване, след което ни се отваря изскачащ прозорец за избор на файла от устройството ни и след селектиране на избрания файл и натискане на бутона за отваряне на файла, триизмерния обект се зарежда на сцената (Това е показано на Фиг. 4.1.2.1., Фиг.4.1.2.2. и Фиг.4.1.2.3.).



(Фиг. 4.1.2.1. Избор на триизмерен файл от устройството)



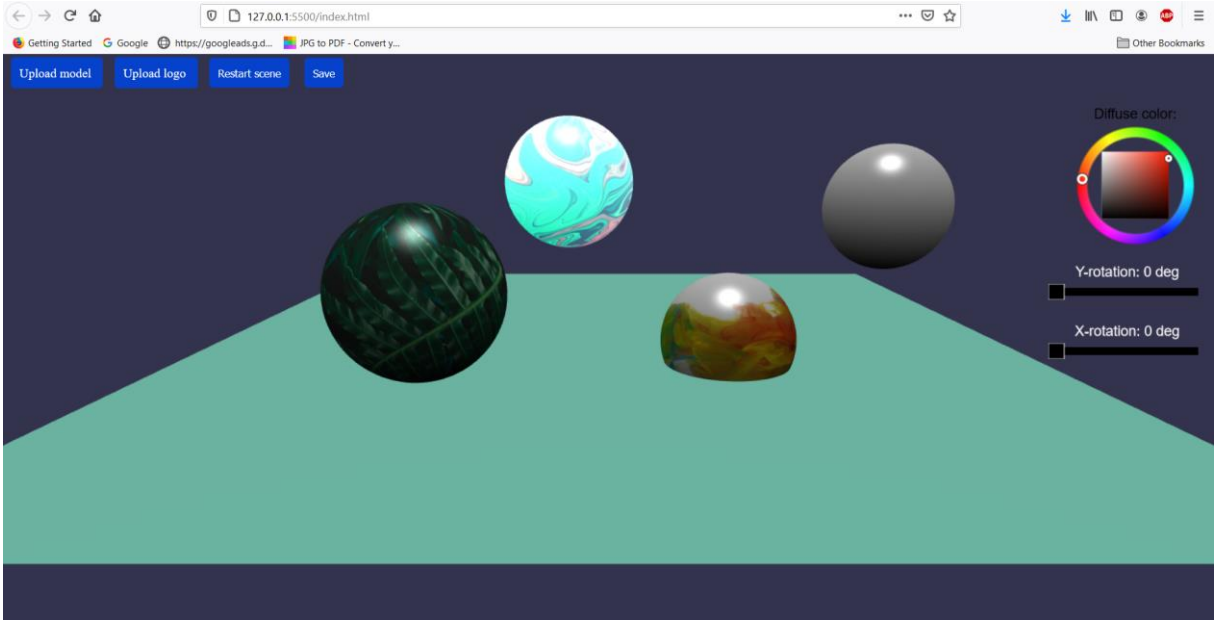
(Фиг. 4.1.2.2. Зареждане на избрания файл от устройството)



(Фиг. 4.1.2.3. Визуализация на обекта върху земята)

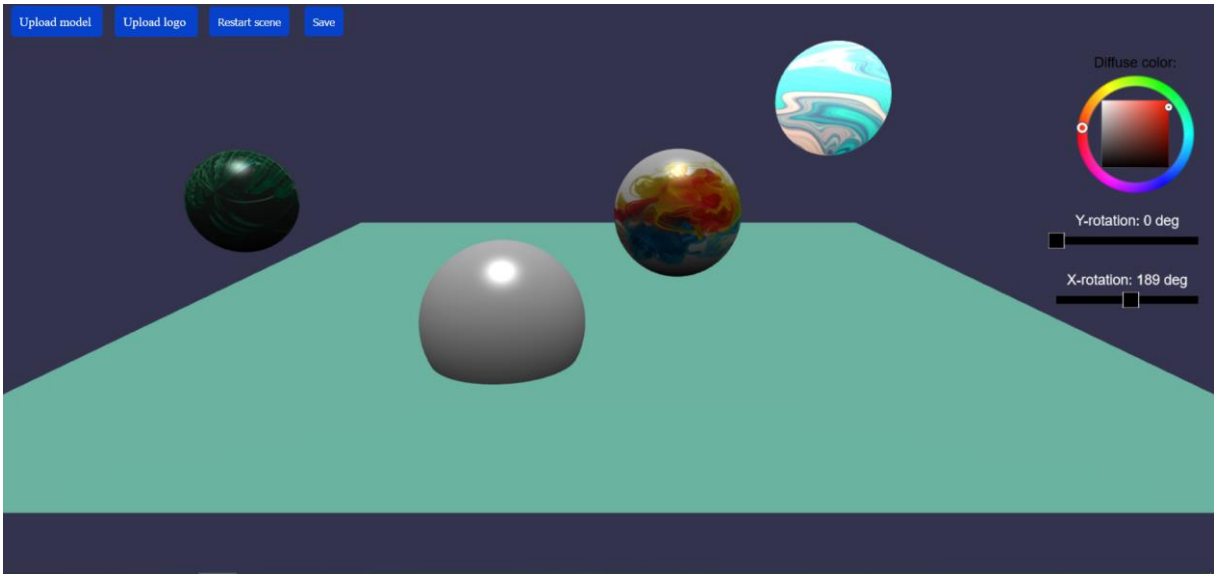
4.1.3. Позициониране и ротация на обектите

На сцената имам поставени няколко сфери (Фиг. 4.1.3.1.), които могат да бъдат разместени и завъртани, след селектирането им с натискане върху тях. Те могат да бъдат местени и по трите си оси и завъртани по x и y оста, също така можем да променяме позицията си, от която гледаме сцената, да се отдалечаваме от нея и приближаваме. (Тези функции са показани на Фиг. 4.1.3.2. и Фиг. 4.1.3.3.)



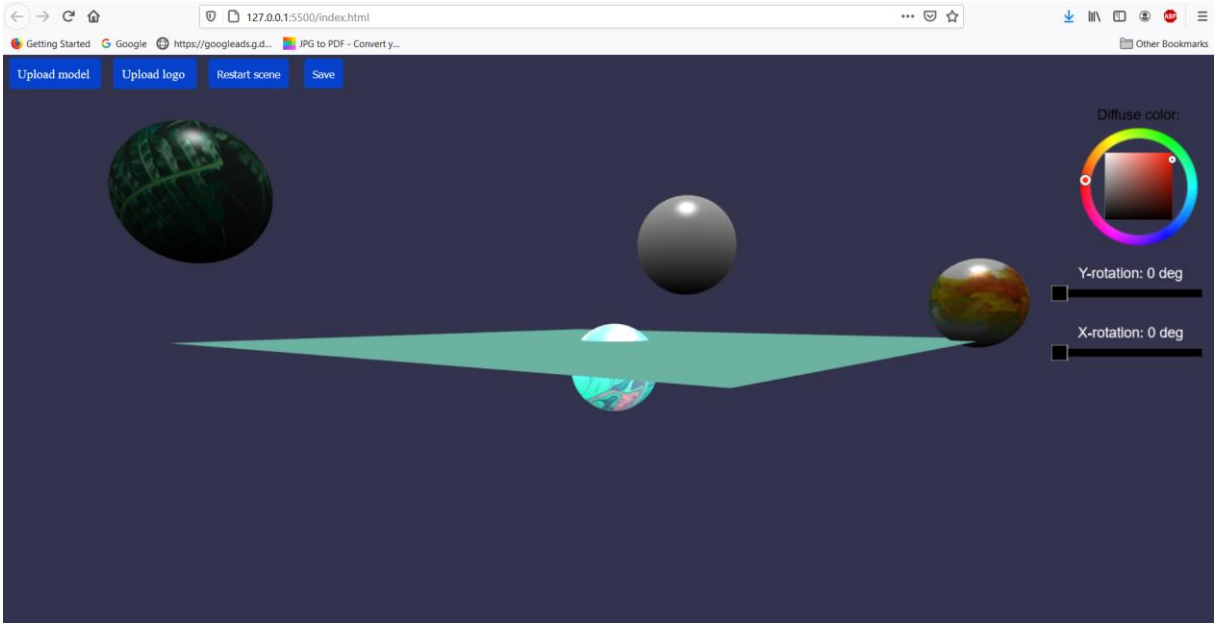
(Фиг. 4.1.3.1. Добавяне на сфери на сцената)

Позиционирането на сферите се случва със задържането им и тегленето с мишката по екрана, а завъртането става с двата плъзгача, за двете различни оси(Това е показано на Фиг. 4.1.3.2. Позициониране и ротация на обектите)



(Фиг. 4.1.3.2. Позициониране и ротация на обектите)

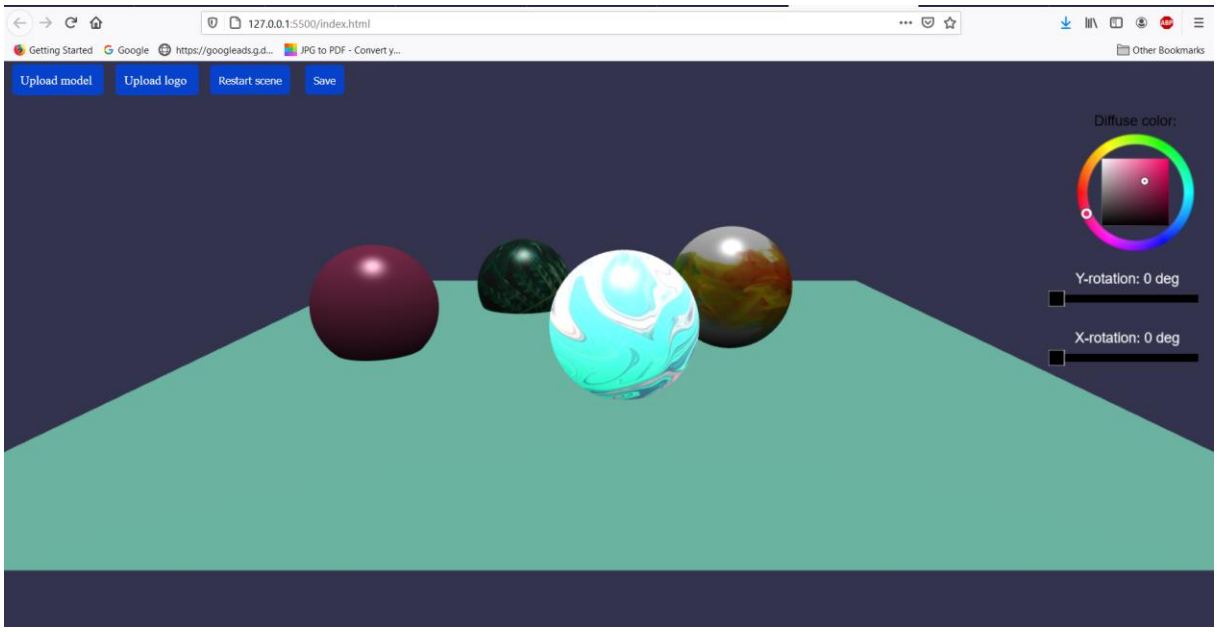
Управлението на изгледа се случва, чрез задържане с мишката някъде на екрана и позиционирането на гледната точка, така както желае потребителя. Потребителя също така може да се приближава и отдалечава от сцената. (Това е показано на Фиг. 4.1.3.3.)



(Фиг. 4.1.3.3. Редактиране на гледната точка на потребителя)

4.1.4. Добавяне на цвят на обектите

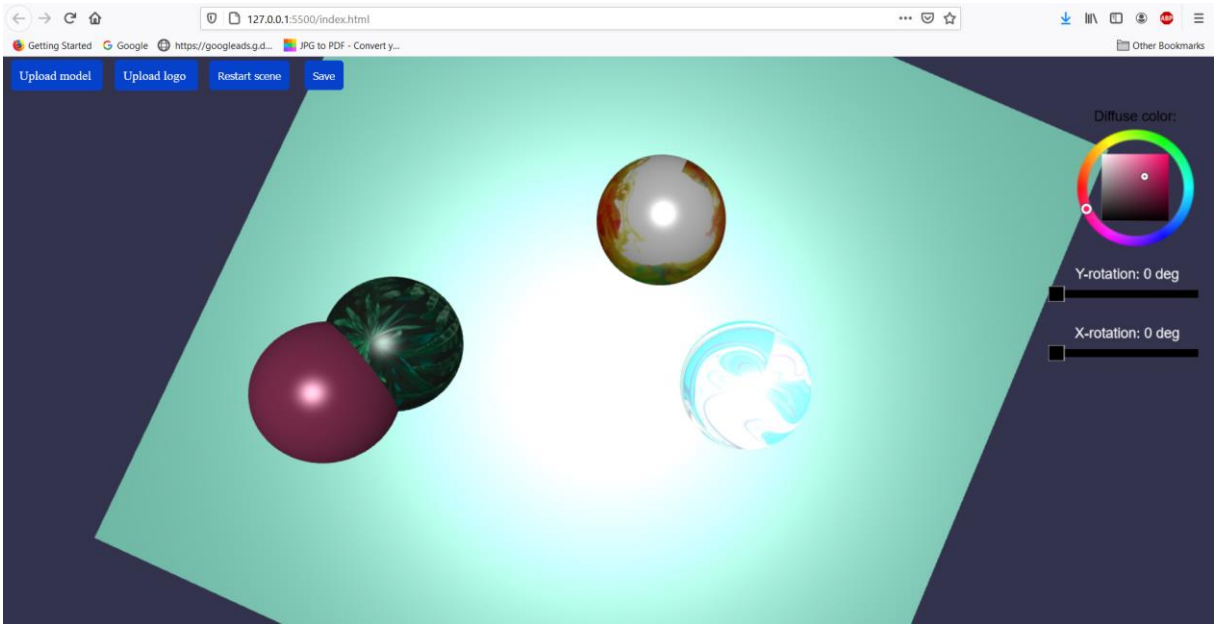
Добавянето на цвят става, чрез елемента за избор на цвят, като той регулира цветовия тон, наситеността на цвета и неговата яркост. (На Фиг. 4.1.4.1 е показано как цвета на сивата сфера се променя на червен с помощта на инструмента)



(Фиг. 4.1.4.1. Редактиране на цвета на даден обект)

4.1.5. Събиране на два меша

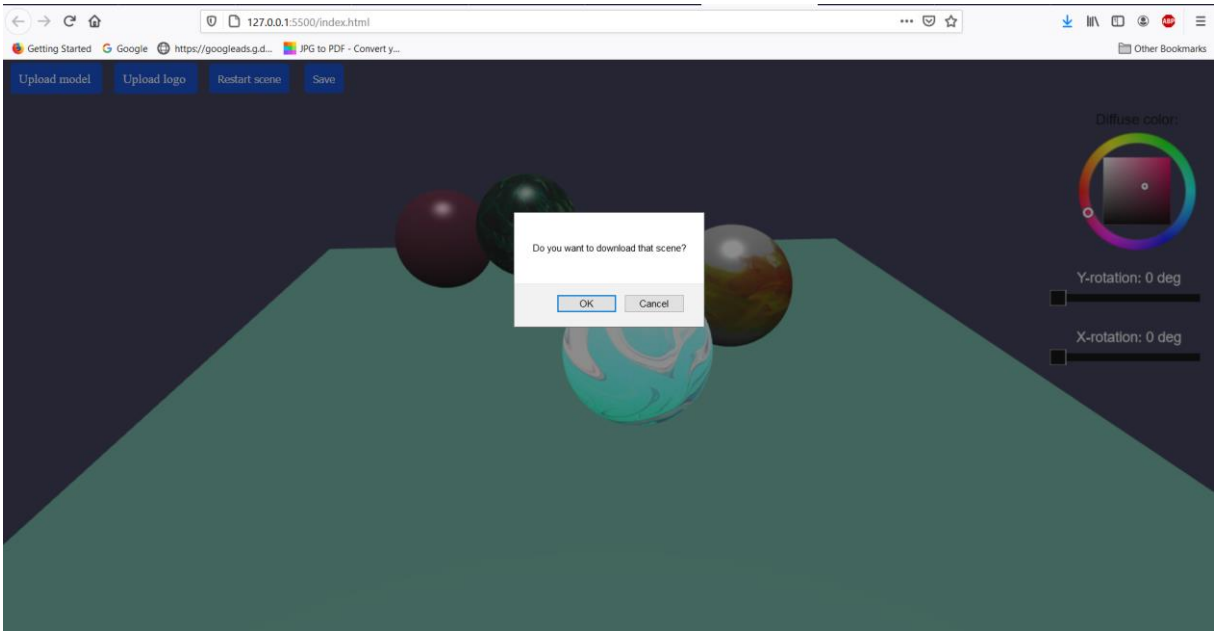
Обединяването на два меша може да стане, чрез ръчното им долепяне един до друг, което представлява ръчното брандиране на един модел. В този случай аз взимам зелената сфера като модел, а цикламената като лого и чрез ръчното им долепяне един до друг се получава един обект. (Това е показано на Фиг. 4.1.5.1.).



(Фиг. 4.1.5.1. Ръчно брандиране на модел)

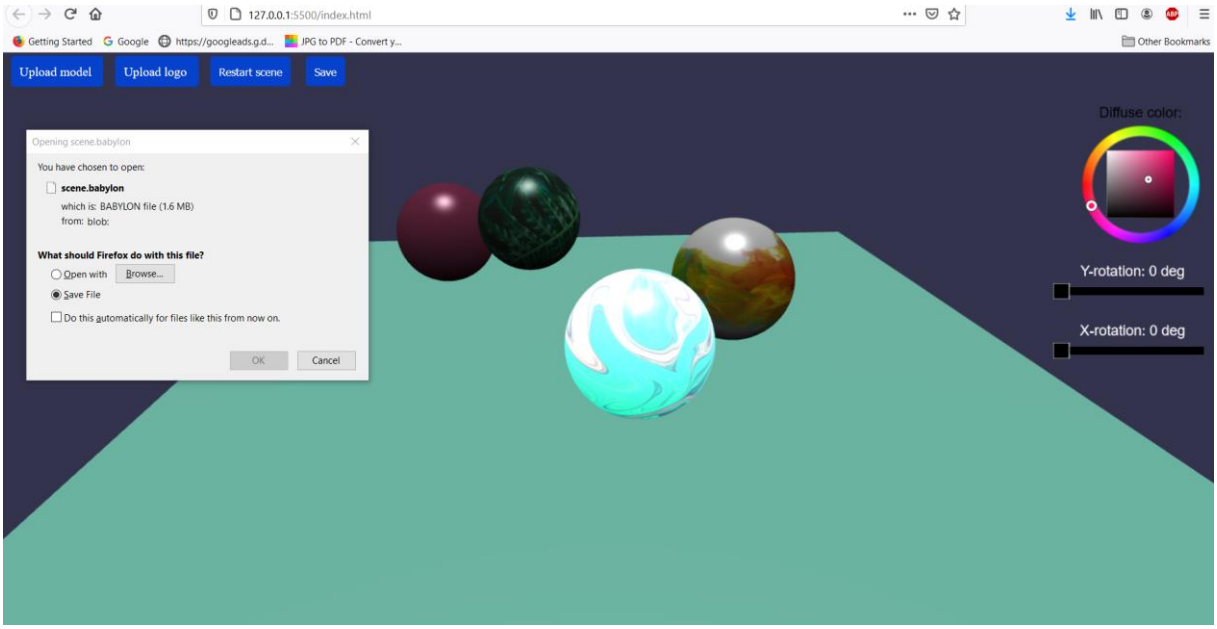
4.1.6. Запазване на редактираната сцена и качването и отново

Запазването на сцена става, чрез натискането на бутона за запазване, тогава ни изскача прозорец, който ни пита дали искаме да запазим файла (Фиг. 4.1.6.1).



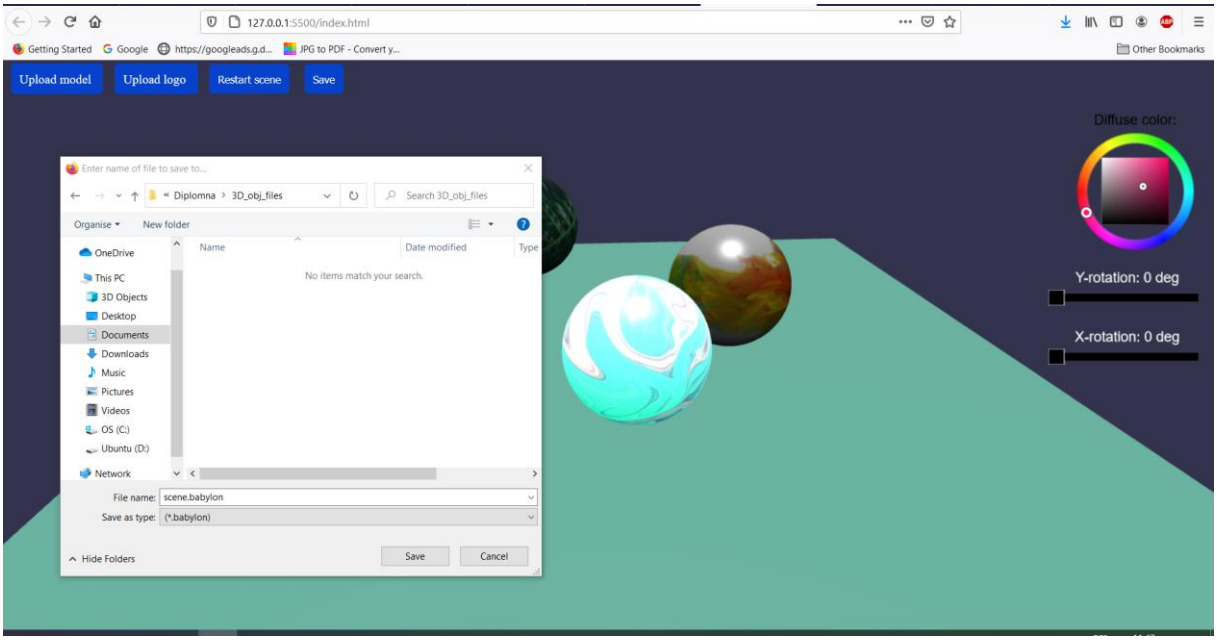
(Фиг. 4.1.6.1 Запазване на сцена)

След като се съгласим да изтеглим сцената ни се появява нов изскачащ прозорец, на който трябва да изберем опцията да се запази файла. (Това е показано на Фиг. 4.1.6.2.).



(Фиг. 4.1.6.2. Избиране на запазване на файла)

Избираме място, където желаем да запазим файла и го запазваме (Фиг. 4.1.6.3. Запазване на файла)



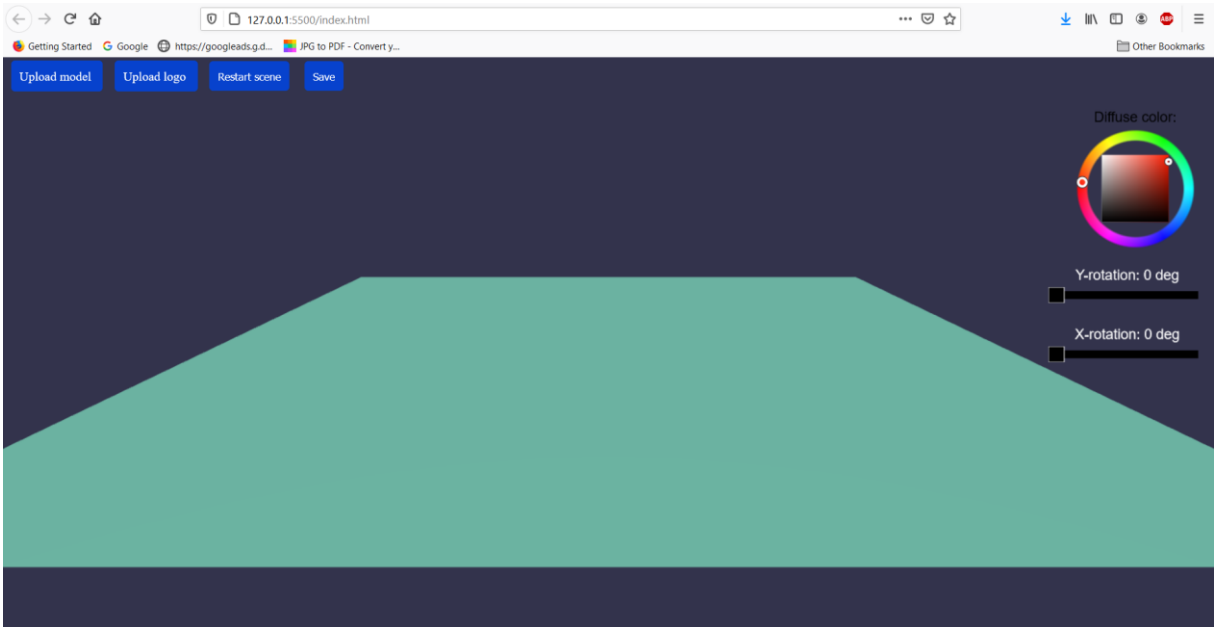
(Фиг. 4.1.6.3. Запазване на файла)

Качването на тази сцена, която сме запазили и повторното и редактиране и запазване става отново, чрез бутона за качване и отново за запазване.

4.1.7. Рестартиране на сцената

Рестартирането на сцената става, чрез бутона за рестартиране на сцената и така вече, ако не ни е необходимо

това, което сме създали. По този начин отново се връщаме на началната сцена. (Фиг. 4.1.7.1.)



(Фиг. 4.1.7.1. Рестартиране на сцена)

Заключение

В настоящата дипломна бе разработено уеб приложение, което предоставя възможност на потребителите да брандират своите триизмерни модели. То е създадено с цел потребителите да гарантират авторските си права над собствената си работа.

В дипломната работа са постигнати следните цели:

- Зареждане на сцена
- Добавяне на камера на сцената
- Добавяне на светлина на сцената
- Добавяне на бутон за качване на 3D модел и 3D лого от два различни файлови формата
- Добавяне на бутон за запазване на редактирана сцена
- Хващане на събития от мишката, които ни помагат в селектирането на обектите

- Местене на модела по трите оси (x, y, z)
- Добавяне на плъзгач за завъртане на модела по x и y оста
- Ръчно долепяне на два модела

За жалост не успях да довърша инструмента за изкривяване на лого и автоматичното му генериране и залепване към модел.

Идеи за бъдещо развитие на проекта:
- Добавяне на недовършените инструменти
- Добавяне на инструмент, които да коригира размерите на моделите
- Добавяне на още файлови формати, които да поддържа приложението
- Добавяне на невидимо (за човешкото око) брандиране на 3D триъгълни мрежи

Използвана литература

[1]What is SculptGL, https://i.materialise.com/blog/en/sculptgl-for-beginners-powerful-3d-sculpting-without-software-downloads-logins-or-headaches/ , August 28, 2014

[2] 3D Builder, free 3D modeling software by Microsoft, Published on August 31, 2020 by Aysha M., https://www.3dnatives.com/en/3d-builder-microsoft-310820206/

[3] Tinkercad: All you need to know before getting started Published on April 20, 2020 by Carlota V., https://www.3dnatives.com/en/tinkercad-all-you-need-to-know-120320204/#!

[4] What Is Blender (Software)?, 2020, https://all3dp.com/2/blender-simply-explained/

[5] Which is best programming language software to create 3D?, 2017, https://www.quora.com/Which-is-best-programming-language-software-to-create-3D

[6] Review: The 10 best JavaScript editors, Jan 7, 2019, https://www.infoworld.com/article/3195951/review-the-10-best-javascript-editors.html

[7] The 5 Best Code Editors For Web Development, mar 24, 2020, https://medium.com/for-self-taught-developers/the-5-best-code-editors-for-web-development-deaaa0b68fd5

[8] WebGL, April 2019, https://whatis.techtarget.com/definition/WebGL

[9] Introduction to Three.js, 09 Aug, 2020, https://www.geeksforgeeks.org/introduction-to-three-js/

[10] BabylonJS – Introduction, https://www.tutorialspoint.com/babylonjs/babylonjs_introduction.htm

[11] The Most Common 3D File Formats, August 31, 2019, https://all3dp.com/3d-file-format-3d-files-3d-printer-3d-cad-vrml-stl-obj/

[12] Key 3D Modeling Terms Beginners Need to Know, May 16, 2017, https://i.materialise.com/blog/en/3d-modeling-terms/

[13] What is a Polygon Mesh?, https://conceptartempire.com/polygon-mesh/

[14] Quaternions, https://cathyatseneca.gitbooks.io/3d-modelling-for-programmers/content/mathematical_background/quaternions.html

[15] What is API - https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation

[16] What is CAD Software?, March 18, 2019, https://www.3dnatives.com/en/top10-cad-software-180320194/#!

[17] What is web app?, https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app

[18] Advantages of web-based software, https://www.simdim.com/services/web-app

[19] Why Millions of Developers use JavaScript for Web Application Development, June 7, 2018, https://torquemag.io/2018/06/why-millions-of-developers-use-javascript-for-web-application-development/

[20] The relationship between HTML, CSS and JavaScript, https://www.freecodecamp.org/news/the-relationship-between-html-css-and-javascript-explained-by-building-a-city-a73a69c6343/

Съдържание