**ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ** **към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**ДИПЛОМНА РАБОТА**

Тема: Конфигуратор на продукти

Дипломант: Научен ръководител:

*Костадин Зотев Кирил Митов*

СОФИЯ

2021

**ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ**

**СИСТЕМИ към ТЕХНИЧЕСКИ**  **УНИВЕРСИТЕТ - СОФИЯ**

Дата на заданието: 10.11.2020 г. Утвърждавам:..............................

Дата на предаване: 10.02.2021 г. / /

**ЗАДАНИЕ**

**за дипломна работа**

на ученика Костадин Алексиев Зотев от 12 В клас

1. Тема: Конфигуратор на продукти
2. Изисквания:

Конфигураторът представлява уеб базиран редактор на 3D сцена със следните опции:

2.1  Качване на готов 3D модел

2.2  Местене, завъртане и reset на сцената

2.3  Промяна на светлината и цветовете на средата

2.4  Определяне на потребителски настройки на модела и части от него (цвят, текстура, размер, позиция)

2.5  Потребителска настройка за подмяна на модела

2.6  Подреждане на възможните настройки в процес

2.7  Запаметяване и зареждане след всяка стъпка от редакцията

2.8  Експортиране в HTML формат на конфигуратора

1. Съдържание
   1. Обзор
   2. Същинска част
   3. Приложение

Дипломант:...............................

/Костадин Зотев/

Ръководител:............................

/Кирил Митов/

Директор:.................................

/доц. д-р инж. Ст. Стефанова/

Мнение на научния ръководител

Дипломантът Костадин Зотев се запозна са 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 създаде конфигуратор на 3Д модели за E-commerce магазини. Настоящата дипломна работа представя постигнатите от дипломанта резултати.

# Увод

Настоящата дипломна работа има за цел да разработи уебсайт, който да предоставя възможност на потребителите да създадат персонализиран 3D обект. По този начин потребителят ще може да направи едни специален за него 3D обект и да го съхрани на компютъра си.

В повечето сайтове за персонализиране на продукти хората трябва да преминават през множество продукти, които имат леки промени по тях (като например единият е червен, а другият е зелен), това заема много място и отнема много време на потребителя да премине през всичките тези продукти и с помощта на тази дипломна работа проблемът ще бъде разрешен. При влизането на сайта, потребителят ще може да избира върху кой предмет конкретно иска да има и ще може да го направи по своите изисквания. Така ще остане по-задоволен и няма да има нужда да преминава през множество от излишни продукти, които не са по вкуса му.

Първа глава

# Методи и технологии за реализиране на уеб приложения

## Уебстраница

Уебстраницата[1] е документ или информационен ресурс, който е подходящ за World Wide Web. Достъпът до уебстраниците може да бъде осъществен чрез уеб браузър, а съдържанието им обикновено се показва на монитора на компютър (вкл. на екран на мобилно устройство). Тази информация обикновено се съхранява във формат от тип HTML или XHTML и може да предоставя навигация към други уебстраници чрез хипертекстови връзки. Обикновено уебстраниците включват текст, изображения (картинки, илюстрации, фотографии) и стилове във формат на файла от тип CSS, но може също да включват анимация, мултимедийно съдържание (файл с музика/звук, видеоклип с или без звук) или скриптове, написани на JavaScript, VBScript и др. скриптови програмни езици.

Уебстраниците са основно два вида – статична и динамична. Статична страница е тази страница, чието съдържание се определя предварително, и се съхранява в този вид на уеб сървъра. Докато динамичната уебстраница е тази страница, чието съдържание се създава в зависимост от характеристиките, предоставени от потребителя и в зависимост от данните, съхранявани в базата данни на сайта.

## 3D компютърната графика

3D компютърната графика[2] е графичното изкуство, създадено с помощта на цифрови компютри и специализиран 3D софтуер. Като цяло терминът може да се отнася до процеса на създаване на такива графики или областта на изучаване на 3D компютърни графични техники и свързаните с тях технологии.

3D компютърната графика се различава от 2D компютърната графика по това, че триизмерното представяне на геометрични данни се съхранява в компютъра за целите на извършване на изчисления и рендиране на 2D изображения.

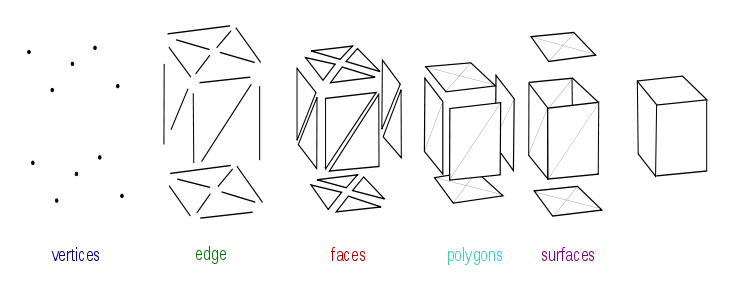
Като цяло изкуството на 3D моделирането, което подготвя геометрични данни за 3D компютърна графика, е подобно на скулптура или фотография, докато изкуството на 2D графиката е аналогично на рисуването. Но в същото време 3D компютърната графика разчита на много от същите алгоритми като 2D компютърната графика.

В софтуера за компютърна графика това има определени ефекти като осветление, докато някои предимно 3D приложения използват 2D визуални техники.

### Моделиране

Процесът на 3D моделирането[3] представлява, произвеждането на цифров обект, който е напълно способен да бъде анимиран, което го прави съществена технология за анимация на персонажи и специални ефекти. Ядрото на модела е мрежата, която най-добре се описва, като колекция от точки в пространството.

Тези точки пък се картографират в триизмерни решетки и се съединяват, като многоъгълни форми, обикновено триъгълници или четириъгълници. Всяка точка или връх има свое собствено положение върху решетката и чрез комбиниране на тези точки във форми се създава повърхността на обект. Това е онагледено на фигура 1.1.



**Фигура 1.1 Моделиране на 3D обект**

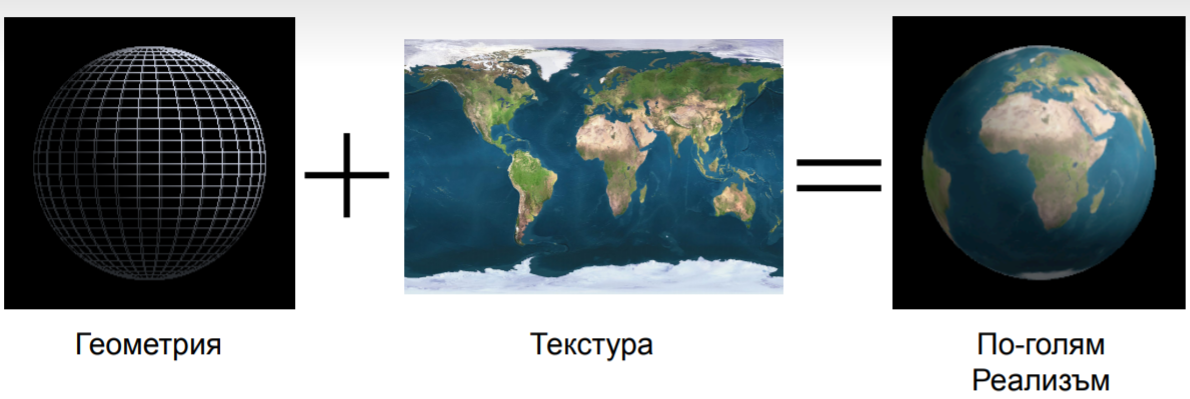
### Рендериране

Рендирането[4] е процес, включен в генерирането на двуизмерно или триизмерно изображение от модел посредством приложни програми. Изобразяването се използва най-вече в архитектурни проекти, видео игри, анимационни филми, симулатори, телевизионни специални ефекти и визуализация на дизайна. Рендирането помага да се увеличи ефективността и да се намалят разходите при проектирането. Има две категории на изобразяване: предварително изобразяване и изобразяване в реално време. Поразителната разлика между двете се крие в скоростта, с която се извършва изчисляването и финализирането на изображенията.

### Материали и текстури

Материалите и текстурите[5] са свойства, които двигателят за изобразяване използва за създаване на модела. В безпристрастен двигател за изобразяване можете да дадете индикация за двигателя въз основа на моделни материали. Например, можете да конфигурирате как да обработвате светлина, когато тя удари повърхността.

Текстурите се използват за оцветяване на материала с помощта на цветна карта или изображение. Може да се използва и за промяна на самия модел във форма, като се използва карта на изместване. Въпреки фактът, че той е създаден чрез изграждането на триизмерни графики, тези процеси оказват силно влияние. Поставянето на текстура е онагледено на фигура 1.2.



**Фигура 1.2 Слагане на текстура на 3D обект**

## Формати

### STL

Стереолитографичния файл[6] се класифицира като CAD файл се използва като стандартен стереолитографичен дизайн формат, в който стереолитографията е известена като технология, разработена за създаването и развитието на дигиталните 3D компоненти на CAD проект. Съдържанието на тези стереолитографични файлове може да се състои на дигитални 3D дизайн обекти, разположени като елементи на основния модел CAD проект за създаването на множество опции за дизайн прототип. Тези стереолитографични файлове са разработени за съхранение на CAD файлове в стандартизиран формат интегрирани с характеристики, които позволяват данните да се четат и тълкуват много CAD програми. Тези стереолитографични файлове са интегрирани с компилация CAD данни спецификации, използвани за формат на STL файл и определени стандарти за кодиране са въведени в съдържанието на тези стереолитографични файлове за да го приберете във формат STL файл. Тези стереолитографични файлове се поставят с [.stl](https://www.reviversoft.com/bg/file-extensions/stl) разширението и могат да се отварят и редактират с помощта на софтуера Dassault Systemes CATIA наред с други CAD програми.

### OBJ

Разширението на OBJ файл[7] е известен като Wavefront 3D Object File, който е разработен от вълновия фронт Technologies. Той е файлов формат, използван за триизмерен обект, съдържащ 3D координати (полигон линии и точки), текстура, карти и друга информация. Той съдържа стандартен формат 3D изображение, което може да бъде изнесено и открито от няколко програми за редактиране на 3D изображения. Обект файлове могат да бъдат в ASCII формат или двоичен формат, който обаче не съдържа цветни определения за лицата. Той поддържа и полигонални обекти, и свободна форма обекти. В Polygonal, геометрия използва точки, линии, докато в свободна форма, геометрия използва криви и повърхнини. Файлове в OBJ формат могат да бъдат отворени с Autodesk Maya 2013, Blender, и MeshLab в Microsoft Windows, Mac OS и Linux платформи.

## Езици за програмиране на уебстраница

### HTML

HTML[8] е съкращение от Hypertext Markup Language. HTML се използва за създаване на електронни документи (наречени страници), които се показват в глобалната мрежа. Всяка страница съдържа поредица от връзки към други страници, наречени хипервръзки. Всяка уеб страница в Интернет е написана с помощта на една или друга версия на HTML кода.

HTML кодът осигурява правилното форматиране на текст и изображения за вашия интернет браузър. Без HTML браузърът не би знаел как да показва текста като елементи или да зарежда изображения или други елементи. HTML също осигурява основна структура на страницата, върху която се наслояват каскадни таблици със стилове, за да се промени външният й вид. Може да се каже, че HTML е като кости (структура) на уеб страница, а CSS като негова кожа (външен вид).

HTML файловете използват разширението .htm или .html. По-старите версии на Windows (Windows 3.x) позволяват само разширения на файлове с три букви, така че те използват .htm вместо .html. И двете разширения на файлове имат едно и също значение и двете могат да се използват днес.

### CSS

CSS[9] означава Cascading Style Sheets. Докато HTML се използва за структуриране на уеб документ (инплементирането на неща като заглавия и абзаци и ви позволява да вграждате изображения, видео и други носители), CSS преминава и определя стила на вашия документ - оформленията на страниците, цветовете и шрифтовете се определят с CSS.

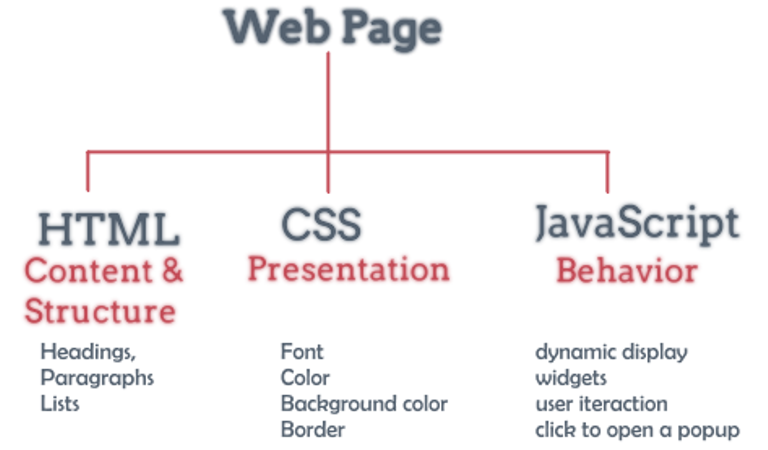
Подобно на HTML, CSS се пише в обикновен текстов редактор или текстов процесор на вашия компютър и има три основни начина да добавите този CSS код към вашите HTML страници. CSS кодът може да бъде външен, вътрешен или вграден. Външният CSS се запазва като .css файл и може да се използва за определяне на външния вид на цял уебсайт чрез един файл (вместо да се добавят отделни копия на CSS код към всеки HTML елемент, който искате да коригирате). За да използвате външна таблица със стилове, вашите .html файлове трябва да включват заглавна секция, която води към външната таблица със стилове. Вътрешният CSS е записани директно в header на конкретна .html страница. (Това е особено полезно, ако имате една страница на сайт, която има уникален външен вид.) И последните, вградените стилове, са записани директно в HTML код и приложими само за един екземпляр на кодиране.

### JavaScript

JavaScript[10] (често срещан като JS) е интерпретиран, обектно-ориентиран език с първокласни функции и е най-известен като скриптовия език за уеб страници, но се използва и в много не-браузърни среди. Той е базиран на прототип скриптов език, който е динамичен и поддържа обектно-ориентирани, императивни и функционални стилове на програмиране.

JavaScript работи на клиентската страна на мрежата, която може да се използва за проектиране или програмиране на поведението на уеб страниците при възникване на събитие. JavaScript е лесен за научаване и мощен скриптов език, широко използван за контрол на поведението на уеб страниците.

На фигура 1.3 е показана връзката между JavaScript, HTML и CSS за изработката на една уебстраница.



**Фигура 1.3 Схема на връзките на уебстраница**

### BabylonJS

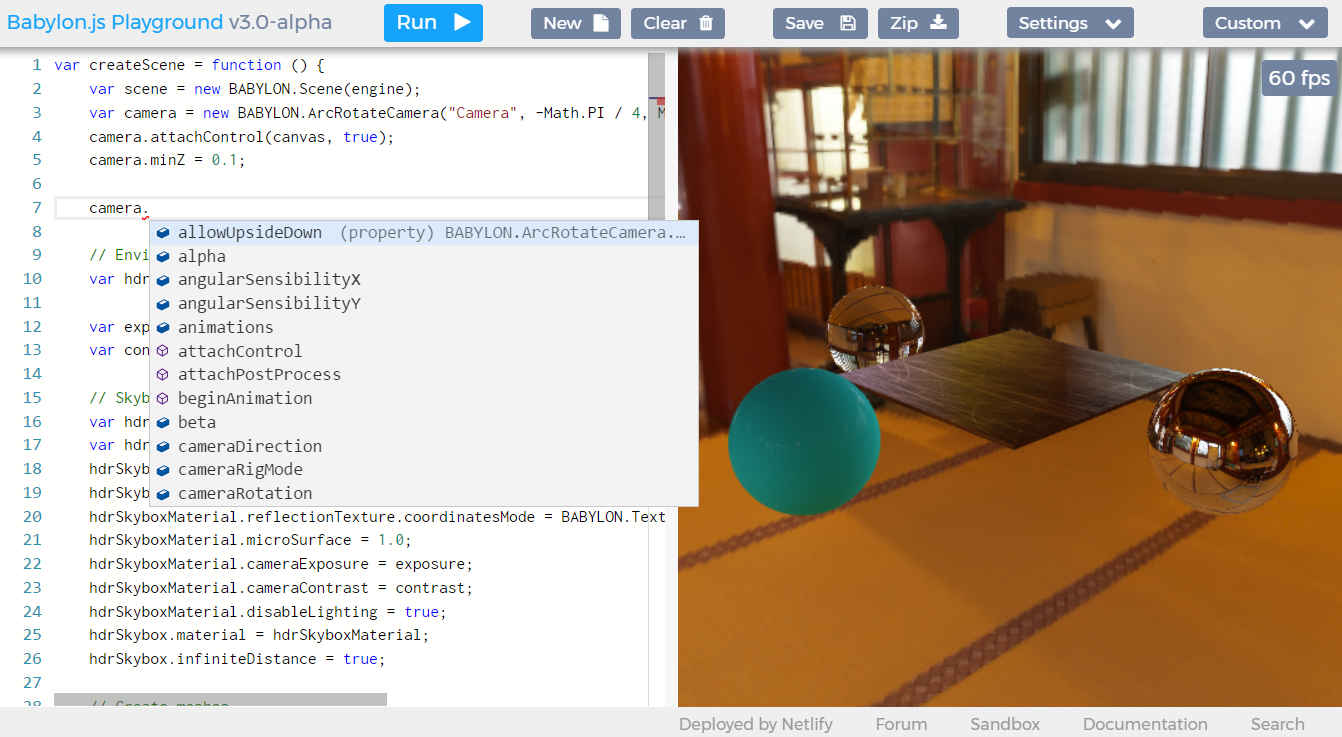
Babylon.js[11] е JavaScript рамка с отворен код, която се използва за разработване на 3D приложения / видео игри за мрежата.

Осветлението е от съществено значение за реализма на сцената. В Babylon.js сенките стават още по-добри с добавената способност да могат да изобразяват меки прозрачни сенки за прозрачни обекти. С въвеждането на PBR филтриране в реално време, усъвършенстваното пречупване и отражение вече е само на няколко реда код.

В Babylon.js е въведен Babylon Native, система за изграждане на собствени приложения със същата мощност на изобразяване и код, използвани за уеб приложения Babylon.js. Тази инициатива прави голяма стъпка напред с Babylon React Native. Този проект използва силата на рамката React Native и я комбинира със силата, красотата и простотата на Babylon.js, за да отключи способността да създава наистина невероятни изживявания на различни платформи.

Babylon.js разполага с богата библиотека от документация, за да помогне да започнете и да овладеете платформата. Babylon.js е с напълно реорганизирана документацията, в сравнение с предишни версии, за да насочват обучението с напредването на знанията. Документацията за Babylon.js ще бъде по-проста и полезна от всякога.

На фигура 1.4 е представен Babylon.js компилатор.



**Фигура 1.4 Babylon.js компилатор**

### Three.js

Повечето проекти на three.js[12], които виждаме, включват 3D графики в реално време, при които взаимодействието на потребителя води до незабавна визуална обратна връзка. Другият тип 3D графики са различни ефекти, изкуствени символи във филми или различни „изобразявания“, които може да видите отпечатани или в уеб каталог.

Подмножество на всичко това е 3D математика. 3D графиките не могат да се правят без математика и компютърните езици не четат 3D концепциите по подразбиране. Тук влиза библиотека, която абстрахира тези математически операции, може би ги оптимизира и излага интерфейс на високо ниво като Matrix4 или .dot ().

Three.js идва със собствена математическа библиотека със специфични класове за 3D математика. Има самостоятелни библиотеки, които се занимават само с тази математика, но с Three.js това е просто подмножество на много по-голяма система.

Предаването е другата голяма отговорност на библиотеката, но и тук нещата стават малко сложни. WebGL е доста специален и се откроява от тази група.

С canvas, svg и CSS отговорността на Three.js е чисто 3D изобразяване. Тези приложни програмни интерфейси имат много други библиотеки, не за да нарисуват 3D обекти, а защото се нуждаят от 3D математика, за да направят 3D визуализация.

3D визуализацията идва най-вече под формата на абстракция на интерфейса. Например, доста е сложно да управлявате 3D състояние на <div> елемент, който е превърнат в 3D чрез CSS. Необходима е много логика, за да се превърне API на платното, за да нарисува нещо, което изглежда като 3D. WebGL е по-ангажиран с порядъци.

Three.js абстрахира всички тези API в нещо толкова просто като render (), но за да го направят, е необходимо общо представяне на това, което е „3D свят“.

### WebGL

WebGL[13] (съкратено от Web Graphics Library) е API на JavaScript за изобразяване на интерактивни 2D и 3D графики във всеки съвместим уеб браузър без използването на приставки. WebGL е напълно интегриран с други уеб стандарти, което позволява ускорено използване на GPU на физика и обработка на изображения и ефекти като част от платното на уеб страницата. WebGL елементите могат да се смесват с други HTML елементи и да се композират с други части на страницата или фона на страницата. Програмите WebGL се състоят от контролен код, написан на JavaScript и шейдър код, написан на OpenGL ES Shading Language(GLSL ES), език, подобен на C или C++ , и се изпълнява на графичния процесор на компютъра (GPU).

#### Помощни програми

Природата на ниското ниво на API на WebGL, която осигурява малко за бързо създаване на желана 3D графика, допринася за създаването на библиотеки, които обикновено се използват за изграждане на нещата в 3D графика. Основни задачи като зареждане на графики, на сцени и 3D обекти в популярните индустриални формати също се абстрахират от библиотеките (някои от които са пренесени в JavaScript от други езици), за да осигурят допълнителна функционалност. Неизчерпателен списък с библиотеки, които предоставят много функции на високо ниво, включва A-Frame (VR) , BabylonJS , PlayCanvas ,three.js , OSG.JS и CopperLicht .

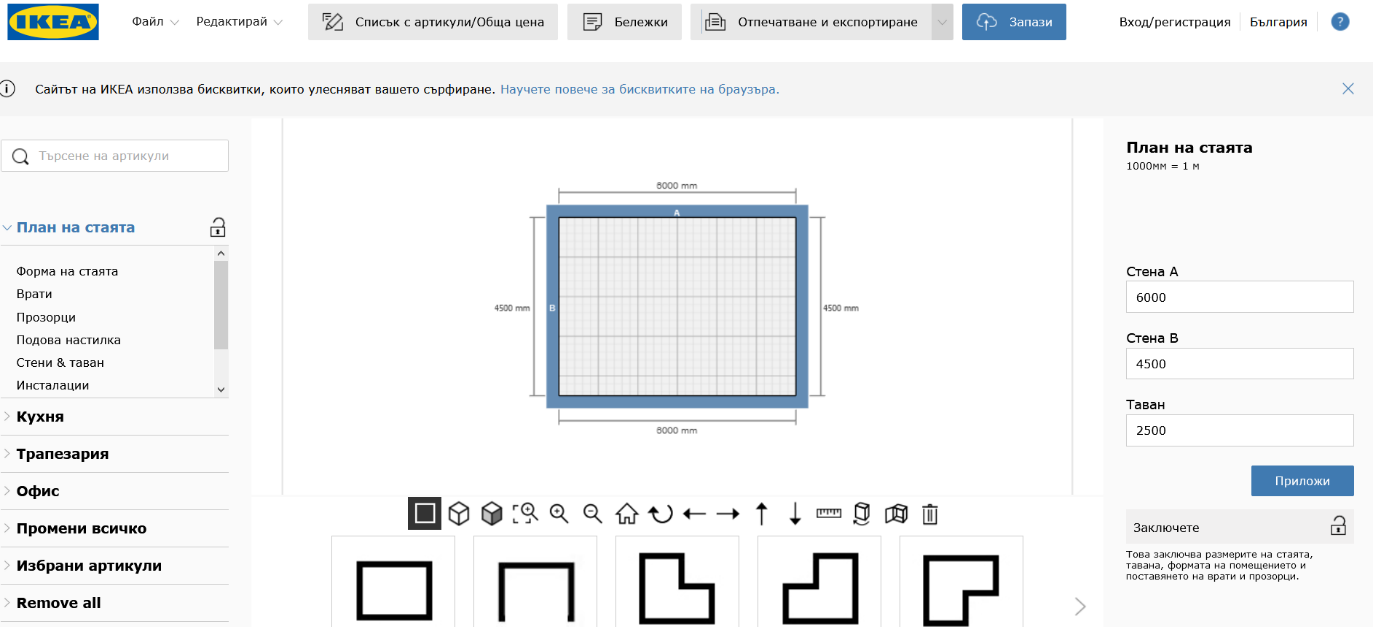
#### Създаване на съдържание

Както за всеки друг графичен API, създаването на съдържание за WebGL сцени изисква използването на обикновен инструмент за създаване на 3D съдържание и експортиране на сцената във формат, който може да се чете от читателя или помощната библиотека. За тази цел може да се използва настолен софтуер за 3D авторство като Blender , Autodesk Maya или SimLab Composer. По-специално, Blend4Web позволява да се създаде сцена на WebGL изцяло в Blender и да се експортира в браузър с едно щракване, дори като самостоятелна уеб страница. Съществуват и някои специфични за WebGL софтуери като CopperCube и онлайн базиран на WebGL редактор Clara.io. Онлайн платформи като Sketchfab и Clara.io позволяват на потребителите директно да качват своите 3D модели и да ги показват с помощта на хоустван WebGL преглед.

## Сходни Проекти

### Онлайн конфигуратор

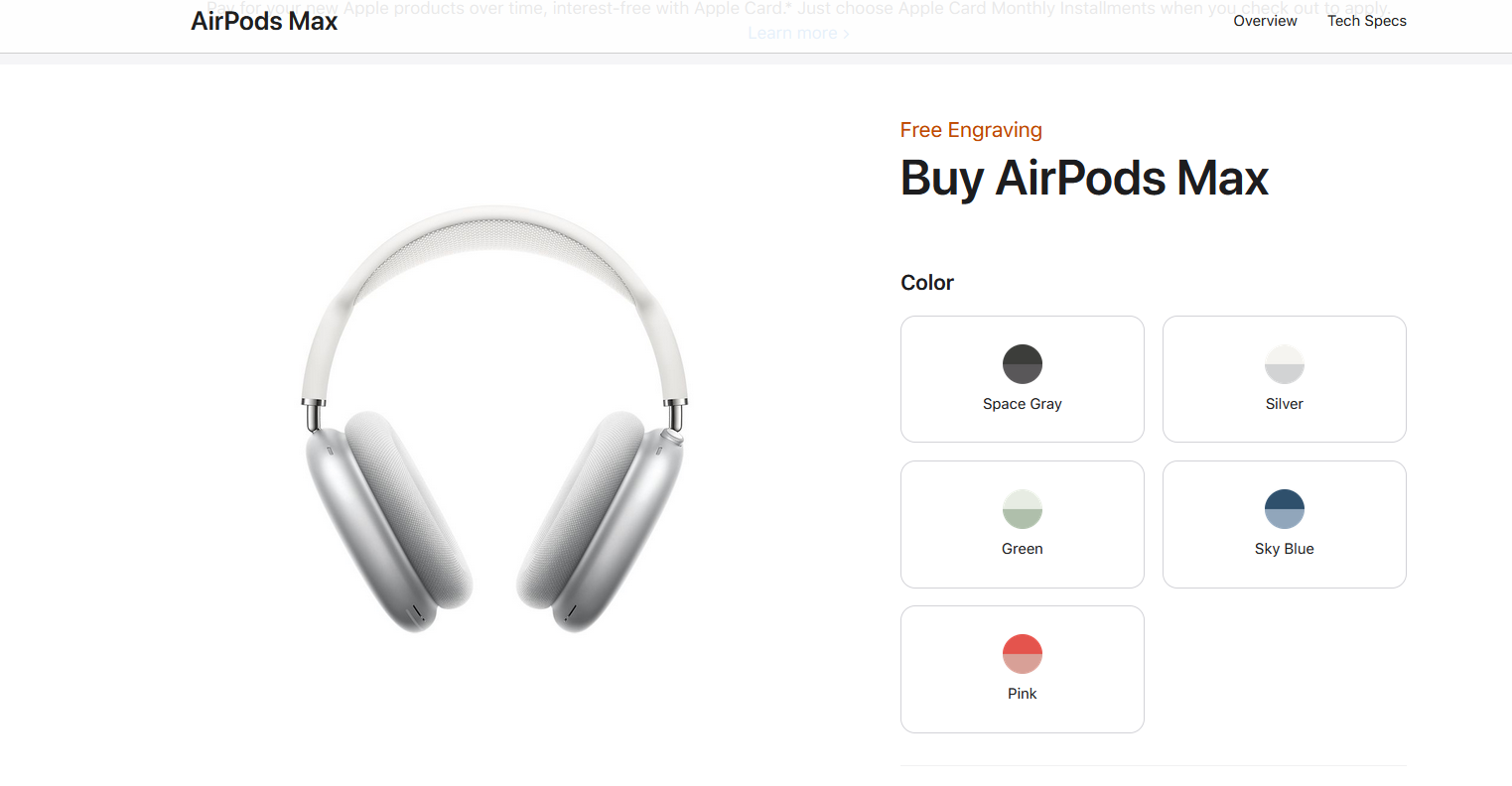
Един примерен онлайн конфигуратор е този на сайта на Ikea. При него може да се аранжира всяка от стаите в дома по избор на клиента. Това помага на използващите го да получат пълна власт над своя проект и след това да запазят крайния продукт на сървъра си. На фигура 1.5 може да се види как изглежда конфигураторът на самия сайт.



**Фигура 1.5 Сайт на Ikea за конфигуриране на стаи**

### Сходен Конфигуратор на продукти

Също така, на проекта, разработен от мен, е персонализиране на слушалки на Apple. Продуктът е представен в 2D среда, може да се избере цвета му и да се гравира лично послание. Това дава голяма възможност на купувача да персонализира продукта, който ще купи. На фигура 1.6 е показан този конфигуратор.



**Фигура 1.6 Сайт за изработка на персонализирани слушалки**

Втора глава

# Изисквания към приложението. Проектиране на структурата на Конфигуратор на продукти

## Функционални изисквания към конфигуратора за продукти

### Качване на 3D модел

Потребителят има възможността да качва 3D обекти в сцената, от файловете на своя компютър. Той има пълната свобода да качи какъвто пожелае файл, който съдържа 3D обект и този обект да бъде инициализиран на сцената.

### Местене на обекта

Използващият конфигуратора може да натисне върху 3D модела на обекта и да го тегли по дължината на подложката и по този начин да сменя положението му в сцената. Също така може да използва плъзгача, за да движи обекта нагоре и надолу.

### Завъртане на обекта

Потребителят има възможността, при натискане върху даден обект, да го завърти.

### Reset на сцената

Чрез натискане на бутона Reset потребителят ще може да се върне в първоначалната сцена, но по този начин ще изтрие напредъка си в разработката си на 3D модел.

### Промяна на светлината

Потребителят може да избира кои от светлините да работят и по този начин да прецени, колко и кои от тях са му нужни за по-добър изглед на модела.

### Промяна на цветовете на средата

Потребителят решава дали иска цвета на средата да остане същият, като на първоначалната сцена или да го промени от дадените избори.

### Смяна на цвета и текстурата на модела

Потребителят ще може да си избере по какъв начин да изглежда моделът му, като смени облика чрез цвят или текстура.

### Смяна на размера на обекта

Потребителят има възможността чрез помощта на плъзгача да променя динамично големината на неговия обект и сам да прецени колко голям да бъде той.

### Настройка за подмяна на обекта

Когато потребителя реши, че иска да смени обекта, който конфигурира, той ще може да подмени досегашния 3D обект със нов. Старият ще бъде изтрит и на негово място ще бъде сложен новия.

### Подреждане на възможните настройки в процес

Настройките в конфигуратора трябват да бъдат подредени по най-добрият начин на използване от страна на потребителя. Настройките трябва да са подредени и систематизирани, за да може потребителят надеждно да мине през всяка стъпка и да може да направи свой собствен продукт.

### Запазване и зареждане след всяка стъпка от редакцията

При решение на потребителят, той ще може да запамети досегашния 3D обект, върху който е работил, и файла с обекта ще бъде свален на компютъра му.

### Експортиране в HTML формат на конфигуратора

Когато потребителят натисне Export, ще се генерира конфигуратор в HTML документ. Този документ съдържа цялата информация за проекта, по който е работил до сега в сайта.

## Аргументация на избора на средите за разработка

### HTML

Едно от най-големите предимства на HTML[14] е, че той е безплатен и няма нужда да се закупува специфичен софтуер. Друго предимство на езика е, че не е нужно използването на допълнителни плъгини. При промени, те могат да се видят мигновено, след като се запази и се презареди HTML страницата. При обработката на кода от браузъра няма никакъв проблем, защото HTML се поддържа от множество браузъри. Друго предимство на езика е, че много лесно може да се интегрира с други езици. При разработката на уебстраница HTML много лесно може да се комбинира с CSS за подобрение на стила и BabylonJS за изобразяване на 3D обектите.

### CSS

Една важна функция на CSS[15], e че при разработката на стил на даден елемент, този стил може да бъде поставен на повече от един елемент, което спестява много време. При прилагането на даден стил, резултатът може да бъде видян мигновено. Освен факта, че CSS има повече опции за форматиране над HTML, като опции за контрол на интервалите и воденето на текст. CSS може също така да определи в какъв ред се показва самата страница.

### JavaScript

Най-голямото предимство на JavaScript е способността да поддържа всички съвременни браузъри и да дава еквивалентен резултат. Независимо къде JavaScript бива хостван, той винаги се изпълнява в клиентска среда, за да спести много честотна лента и да ускори процеса на изпълнение. JavaScript се използва навсякъде в мрежата и това помага за лесното разработване на уебстраница с него. Всеки браузър с активиран JavaScript може да разбере и интерпретира JavaScript кода. Друго голямо предимство е, че JavaScript не изисква процес на компилация, така че не е необходим компилатор. Браузърът интерпретира JavaScript като HTML тагове.

!!!JavaScript е език за програмиране на високо ниво, който отговаря на спецификацията ECMAScript. Това е език, който се характеризира като динамичен, слабо типизиран и многопарадигмен.

Наред с HTML и CSS, JavaScript е една от трите основни технологии на World Wide Web. JavaScript позволява интерактивни уеб страници и по този начин е съществена част от уеб приложенията. По-голямата част от уебсайтовете го използват, а всички основни уеб браузъри имат специален JavaScript механизъм, който да го изпълни.

Много често JavaScript се използва за динамично модифициране на HTML и CSS за обновяване на потребителския интерфейс чрез използването на Document Object Model API.

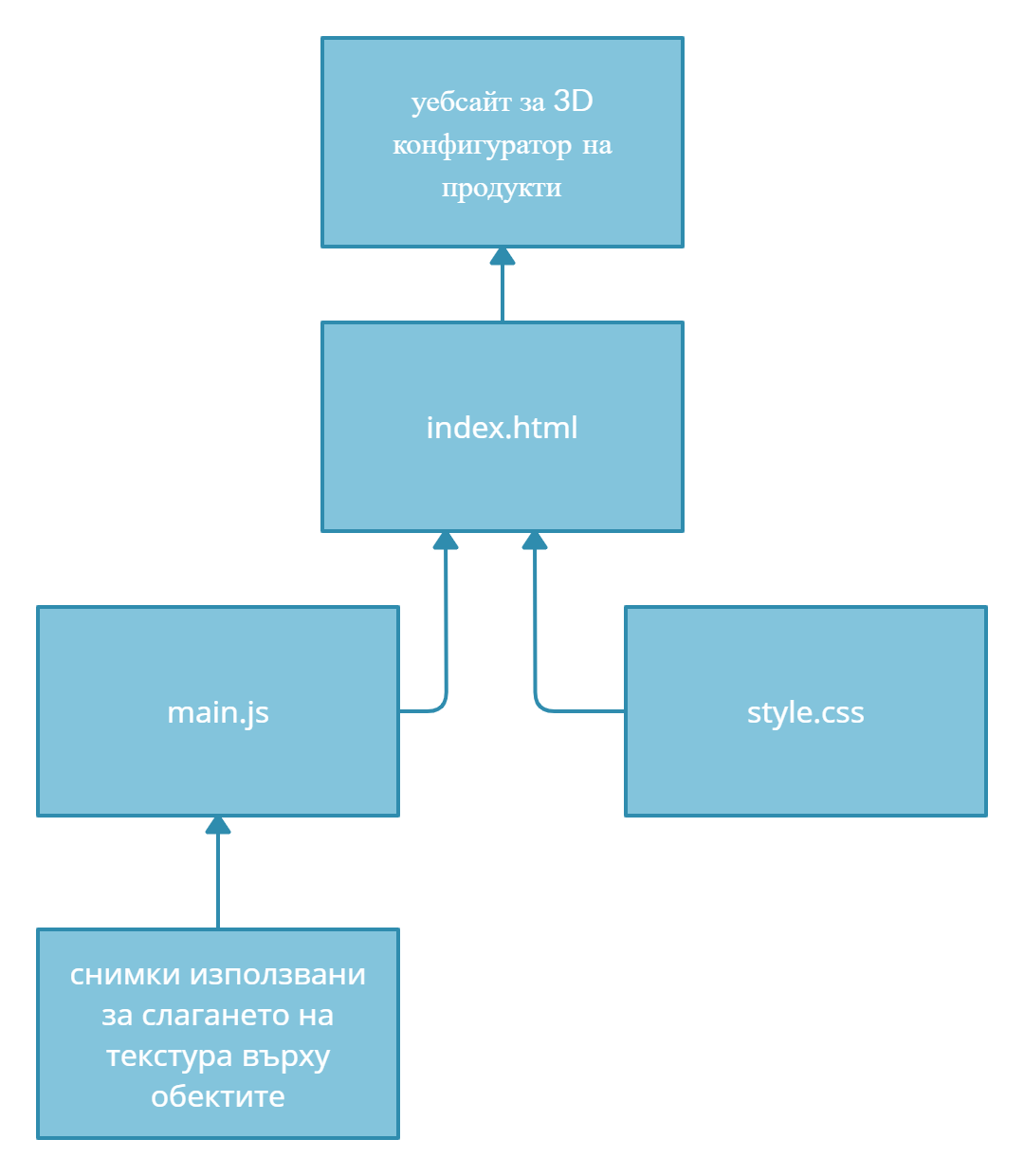
### BabylonJS

Едно значително предимство на Babylon.js е че е един от малкото безплатни за използване компилатори на 3D сцени. Чрез него много лесно може да се интерпретира 3D сцена.

Като 3D двигател той разполага с инструментите за създаване, показване и текстуриране на мрежи в пространството и за добавяне на източници на светлина и камери. Тъй като е фокусиран върху сцената, Babylon.js има някои допълнителни функции, които обикновеният 3D двигател не изисква.

Babylon се използва и за направата на игри. Той има естествена поддръжка за откриване на сблъсък, гравитация на сцената, ориентирани към игра камери (например следяща камера, която проследява движещ се обект), както и естествена поддръжка за Oculus Rift и други устройства за виртуална реалност (VR). Той разполага с приставка за физически двигател, вградена аудио поддръжка, потребителски мениджър на действията и много други.

## Схема на приложението



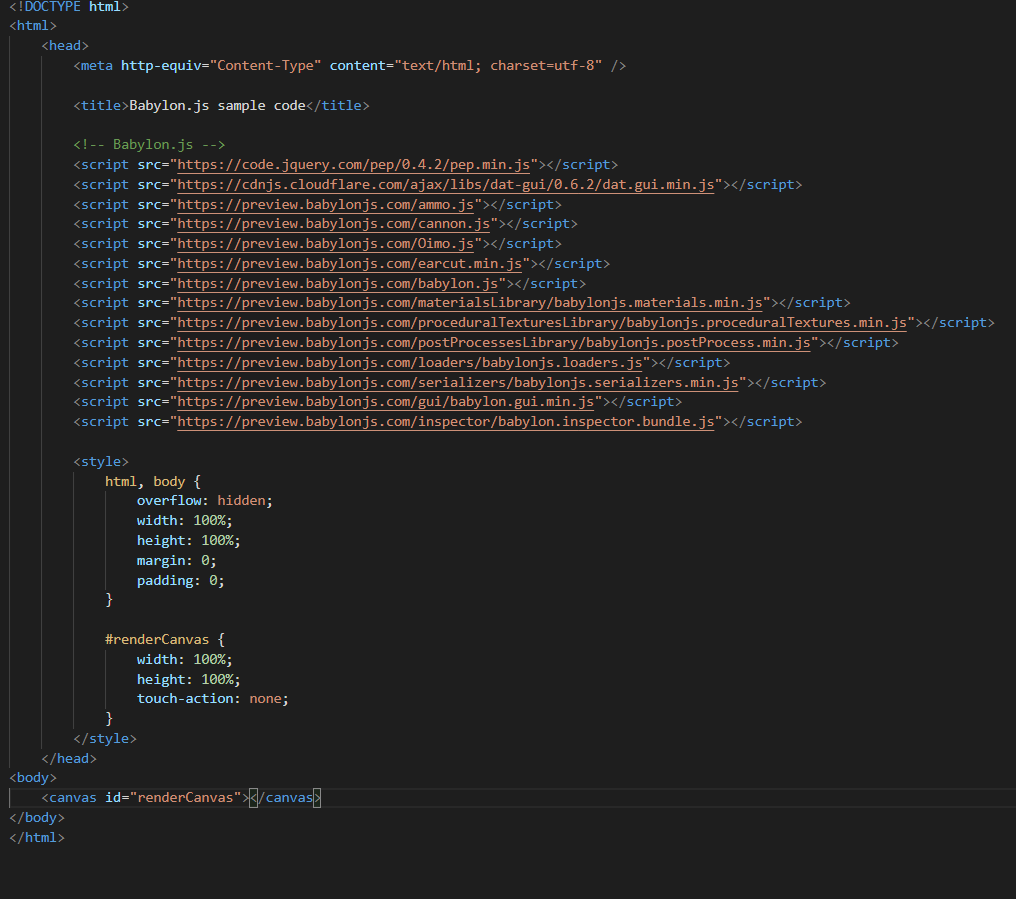
**Фигура 2.1 Схема на приложението**

Трета глава

# Разработка на уеб приложението

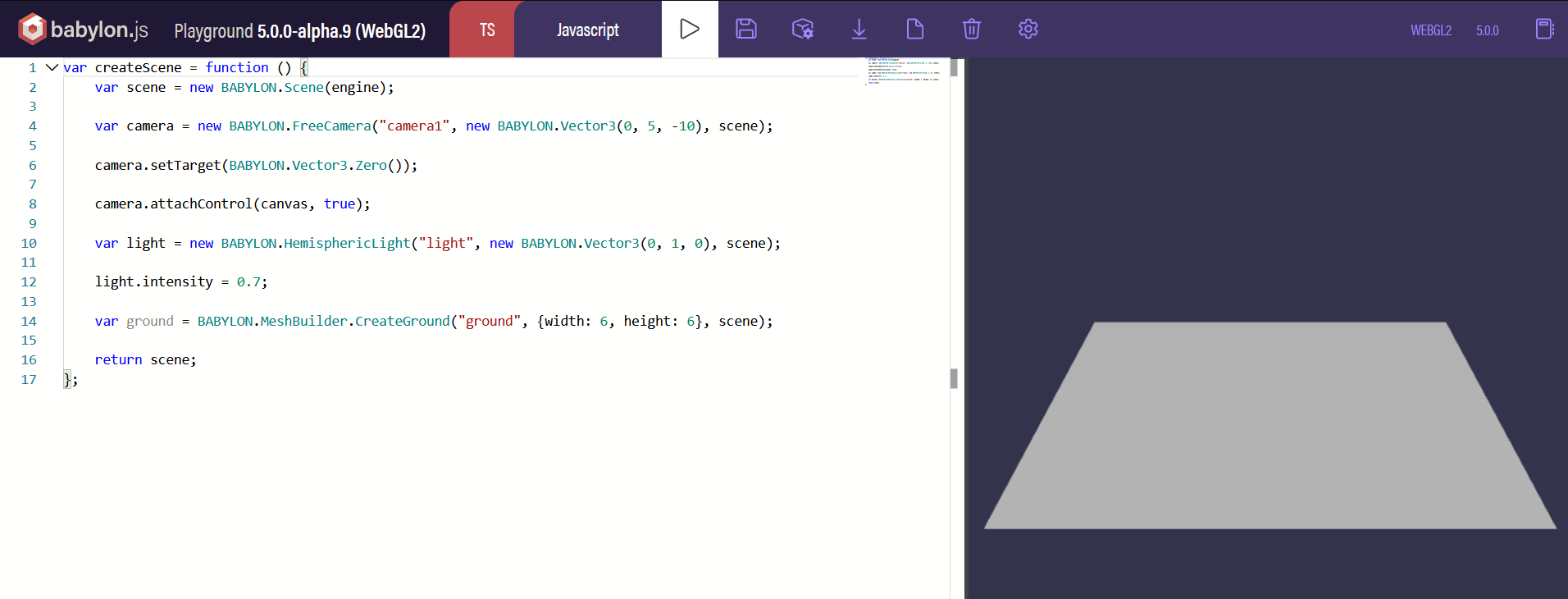
## Начална сцена

В началото правим базов HTML файл, с включен към него CSS. Това е онагледено на фигура 3.1.



**Фигура 3.1 База на кода**

Към него слагаме Babylon частта, която е елементарна сцена взета от Babylon.js Playground(фигура 3.2). На нея са изобразени фона, който е едноцветен, поставка, на която ще бъде поставен самият 3D модел, камера, която е насочена към тази поставка, и единствена светлина насочена отново към поставката.

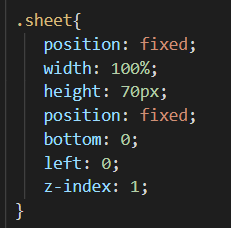


**Фигура 3.2 База на Babylon.js**

## Поставяне на бутони

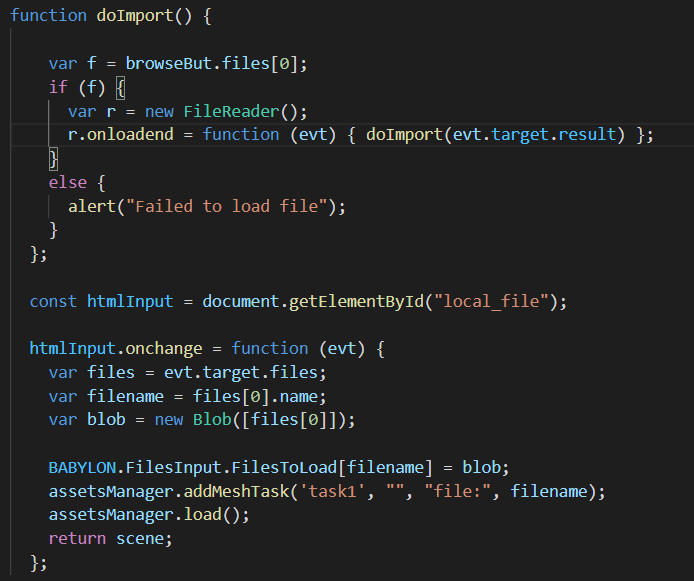
Първоначалната идея за направата на бутоните беше да бъдат сложени в Babylon.js, с помощта на специална библиотека, и те да бъдат инициализирани и самите функции, които ще изпълняват ще бъдат на едно място. Но след дълго проучване на метода за правене на бутоните, установих, че те се правят твърде сложно. От там ми дойде идеята да го направя в HTML частта и функциите да си останат в java script.

Правя един div и го разполагам върху цялата повърхност на екрана. После чрез помощта на CSS го правя прозрачен и в него мога да разполагам бутоните по цялата повърхност на сайта. Чрез тази идея мога по лесно да контролирам бутоните, да променям местоположението и формата им. CSS който прави „невидимия“ div разположен върху уебстраницата е онагледен на фигура 3.3.



**Фигура 3.3 CSS на div върху когото се поставят бутоните**

## Поставяне на бутон за качване на 3D обект

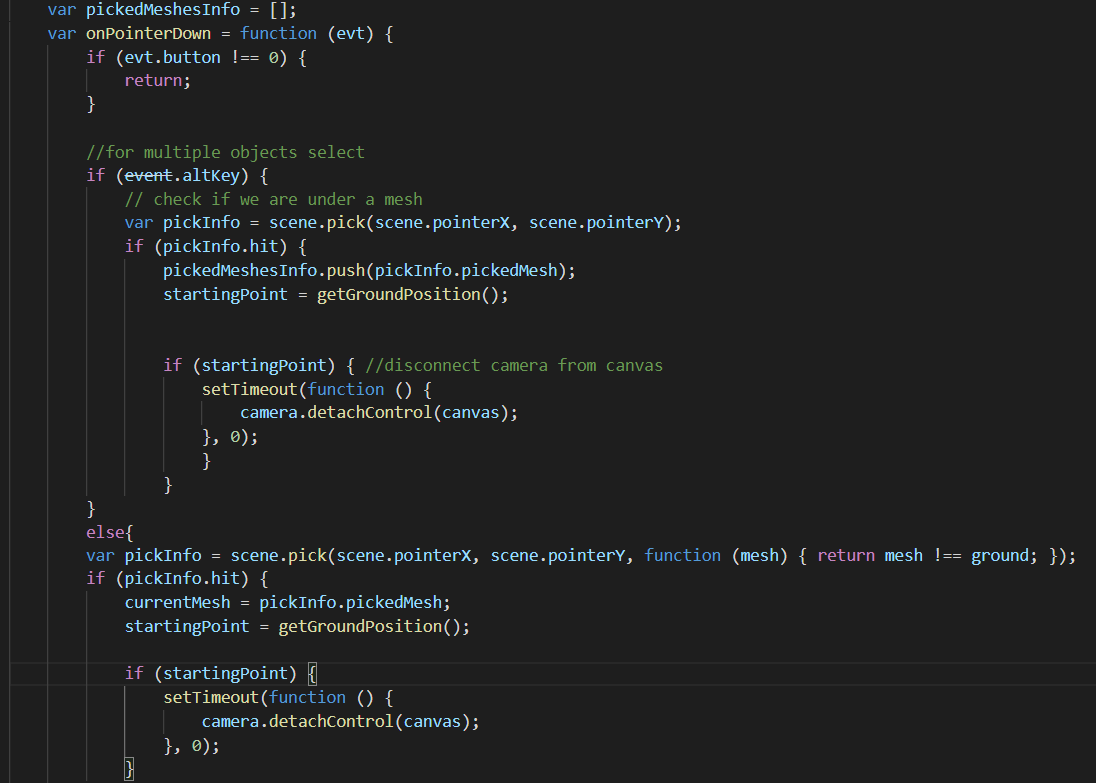
Първо се взима файловете и се запаметява в files , после се взима името на първия файл. След това се използва blob, за да можем да запаметим първия файл. След това се взимат запаметения файл в blob и се подава на Babylon, за да може да бъде зареден в сцената. След това чрез помощта на assetsManager зареждаме обекта и пускаме да се зарежда процеса. Накрая връщаме сцената. Това е изобразено на фигура 3.4. 

**Фигура 3.4 Код за качване на 3D обект**

## Мърдане на 3D обекта

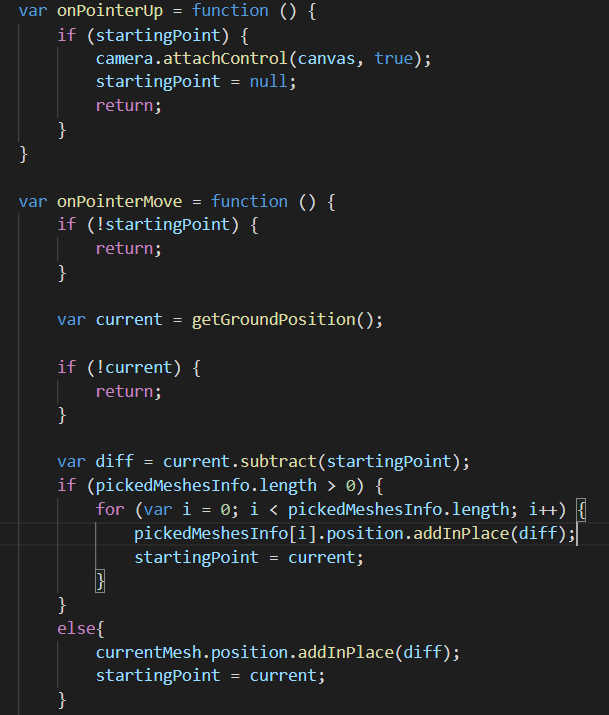
Първо си правим функция getGroundPosition, в която гледаме къде по-точно е поставен нашият обект и връщаме координатите му.

При натискане върху обекта се изпълнява функцията onPointerDown, при която проверяваме дали е натиснат бутон на мишката. Ако самата мишка е натиснала 3D обект взимаме обекта и координати. След това разскачаме камерата от canvas. При задържане на бутона Alt и селектирането на обекти в сцената те се запазват в списък. При местенето на някой от обектите в този списък, всички те се местят с него. Същото става и при смяна на цвета и на текстурата. Това е показано на фигура 3.5.



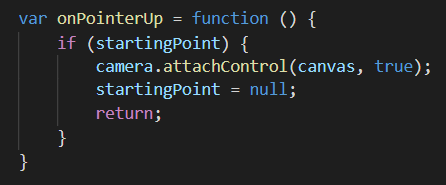
**Фигура 3.5 onPointerDown функция**

При местене на обекта се изпълнява onPointerMove. При тази функция сменяме позицията на обекта с новата му и задаваме началната позиция да е настоящата му. Това е показано на фигура 3.6.



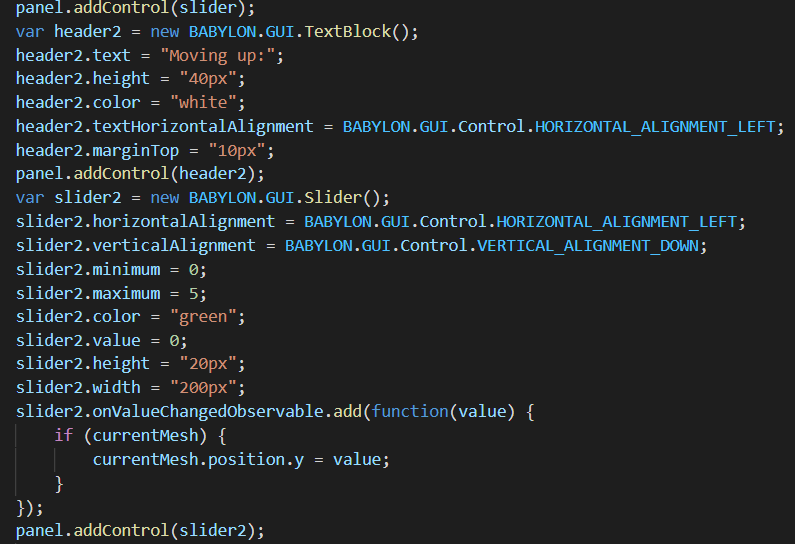
**Фигура 3.6 onPointerMove функция**

При пускане на обекта се използва функцията onPointerUp. При нея свързваме камерата към canvas и махаме началната позиция на обекта. Онагледено е на фигура 3.7.



**Фигура 3.7 onPointerUp функция**

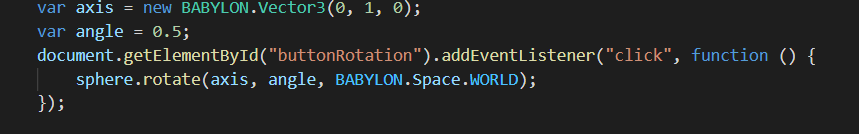
Ако потребителят жела да вдигне обекта нагоре или да го смъкне надолу, той може да използва плъзгачa и да промени положението на обекта спрямо ординатната му ос. Това е показано на фигура 3.8.



**Фигура 3.8 Код на плъзгач за промяна на координатите на обекта по ординатната ос**

## Завъртане

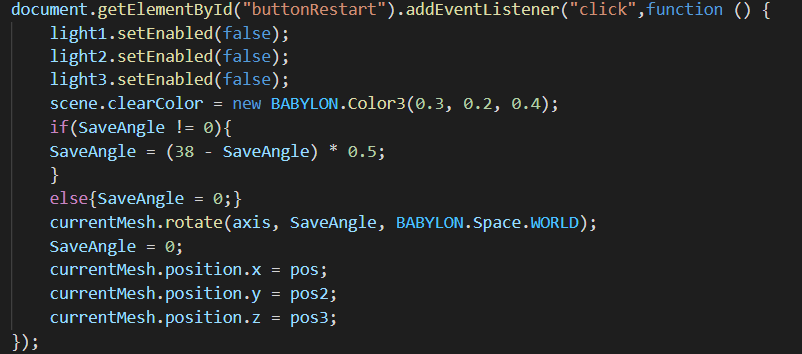
Първият аргумент ни показва по коя от осите ще се върти 3D обектът, след това показва с колко ще се завърти самият обект и с последния показваме, че ние се намираме в 3D свят и че има 3 оси, а не както в 2D света да има 2. Това е онагледено на фигура 3.9.



**Фигура 3.9 Код за завъртане на 3D обект**

## Reset на сцената

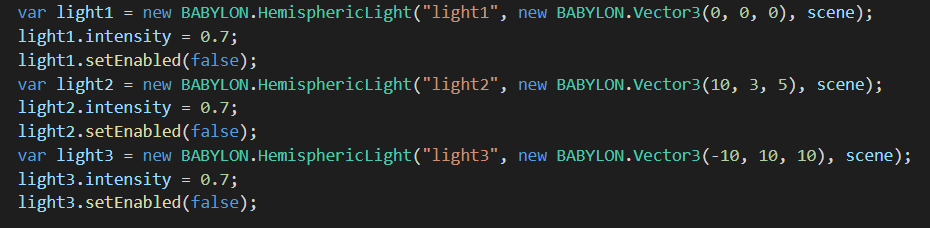
При рестартиране на сцената първо се изгасят всички светлини, без първоначално имплементираната светлина. След това се слага първоначалният цвят на фона. После ако обекта е бил завъртян, бива върнат в първоначалния му вид. Последно слагаме обекта на първоначалното му място, чрез заместване на x, y и z координатите му, който са взети още при натискане на върху самия 3D обект. Това е изобразено на фигура 3.10.



**Фигура 3.10**

## Светлини

На сцената взета от Babylon.js Playground имплементирана светлина свети през цялото време. След това имплементираме 3 светлини(фигура 3.11), всяка на различно място в пространството, както е показано на фигура, за да могат да осветяват обекта от различни ъгли.



**Фигура 3.11 Имплементиране на светлините**

След това при натискане на бутоните, от падащото меню Lights, се разглежда дали променливите са натиснати за четен или нечетен път и така се определят дали трябва да се спре или да се пусне светлината. Това може да бъде видяно на фигура .

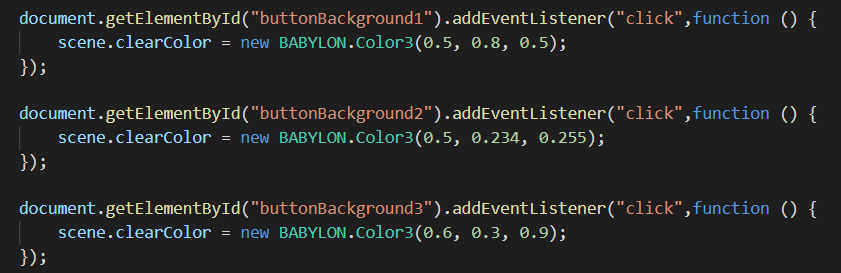
Направено е падащо меню, от което може да се изберат една от трите дадени опции. При повторно натискане на бутона светлината се изключва. По този начин може да се използват всичките светлини, ако потребителят иска или само избраните от него. Това е показано на фигура 3.12.Не може да бъде изключвана светлината, която е имплементирана от началото, защото без светлина ще се вижда черен екран.



**Фигура 3.12 Код за смяна на светлините**

## Смяна на цвета на сцената

При натискането на бутоните, от падащото меню Background, сменяме цвета на фона чрез функцията показана на фигура 3.13 . По този начин можем да видим обекта в различни цветове и по този начин да си го представим по-реалистично.



**Фигура 3.13 Код за смяна на фона**

## Смяна на цвета на обекта

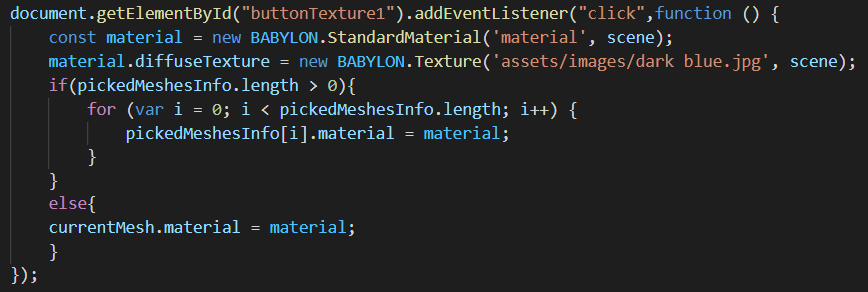
За всеки един от бутоните имплементираме материал с име и казваме, че той принадлежи към сцената, която показваме на екрана. След това се задава стойности за червено, зелено и синьо и по този начин определяме какъв ще е цвета. После слагаме този цвят на дадения 3D обект, както е показано на фигура 3.14.



**Фигура 3.14 Код за смяна на цвета на 3D обект**

## Смяна на текстурата на обекта

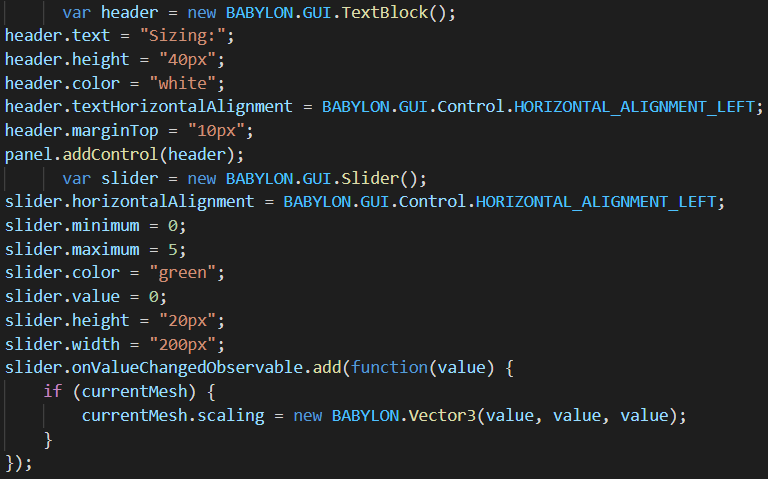
За всеки един от бутоните се имплементира материал с име и втория аргумент казва, че той принадлежи към сцената, която показваме на екрана. След това се показва какъв е пътят на снимката, която ще е текстурата сложена върху обекта и че принадлежи към сцената. После слагаме тази текстура на дадения 3D обект. Това е изобразено на фигура 3.15.



**Фигура 3.15 Код за смяна на текстурата на 3D обект**

## Смяна на размера на обекта

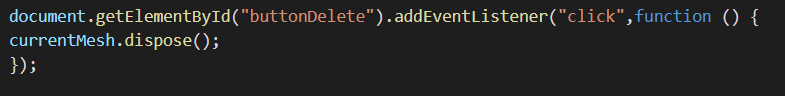
Чрез използването на плъзгача, който е направен на Babylon.js, потребителят ще има възможността да смени размера на натиснатия от него обект, като може да избира между стойност 0 и 5 за натиснатия от него обект. Това е представено на фигура 3.16.



**Фигура 3.16 Код за смяна на размера на 3D обект**

## Настройка за смяна на обекта

При селектиране на 3D обекта, който искаме да бъде махнат и при натискането на бутона Delete, се премахва обекта от сцената(кода е представен на фигура 3.17) и след това на негово място може да се качи от компютъра нов обект.



**Фигура 3.17 Код за изтриване на обект**

## Експортиране на сцената

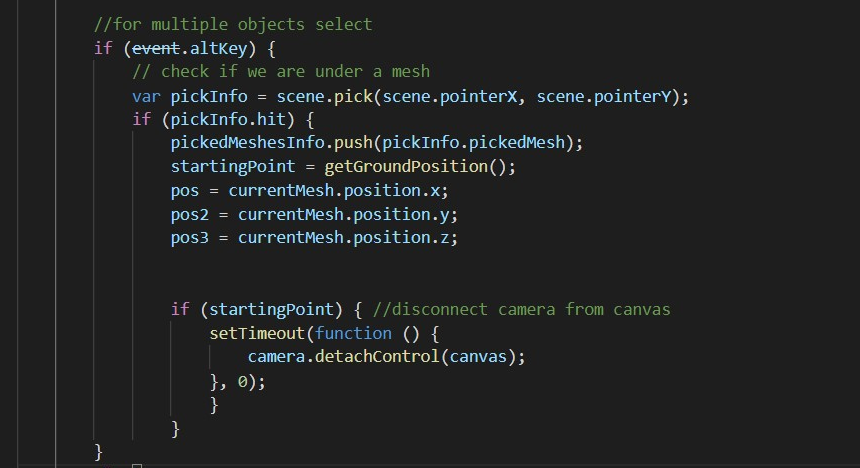
Първо се осигурява, че променливата objectUrl е празна. После се взема сцената на части и тези части се превръщат в JSON. Добавя се разширението .babylon в края на името на обекта, който ще се експортира. След това се прави blob от съдържанието на сцената, който да е stream. Blob се превръща в url. Прави се линк, който да има url-ът на линка и да го тегли. Накрая се прави ивент, който при избиране може да изтегли файла. Това е онагледено на фигура 3.18.



**Фигура 3.18 Код за Експортиране**

## Едновременно местене на обекти

При задържане на бутона Alt и селектирането на няколко обекта на сцената, потребителят има възможността да мърда едновременно обектите, които е селектирал. Процеса е същия, като местенето на един обект, но тук новото е, че запазваме информацията на обектите не в променливата currentMesh, а в масива pickedMeshesInfo. Това е онагледено на фигура 3.19.



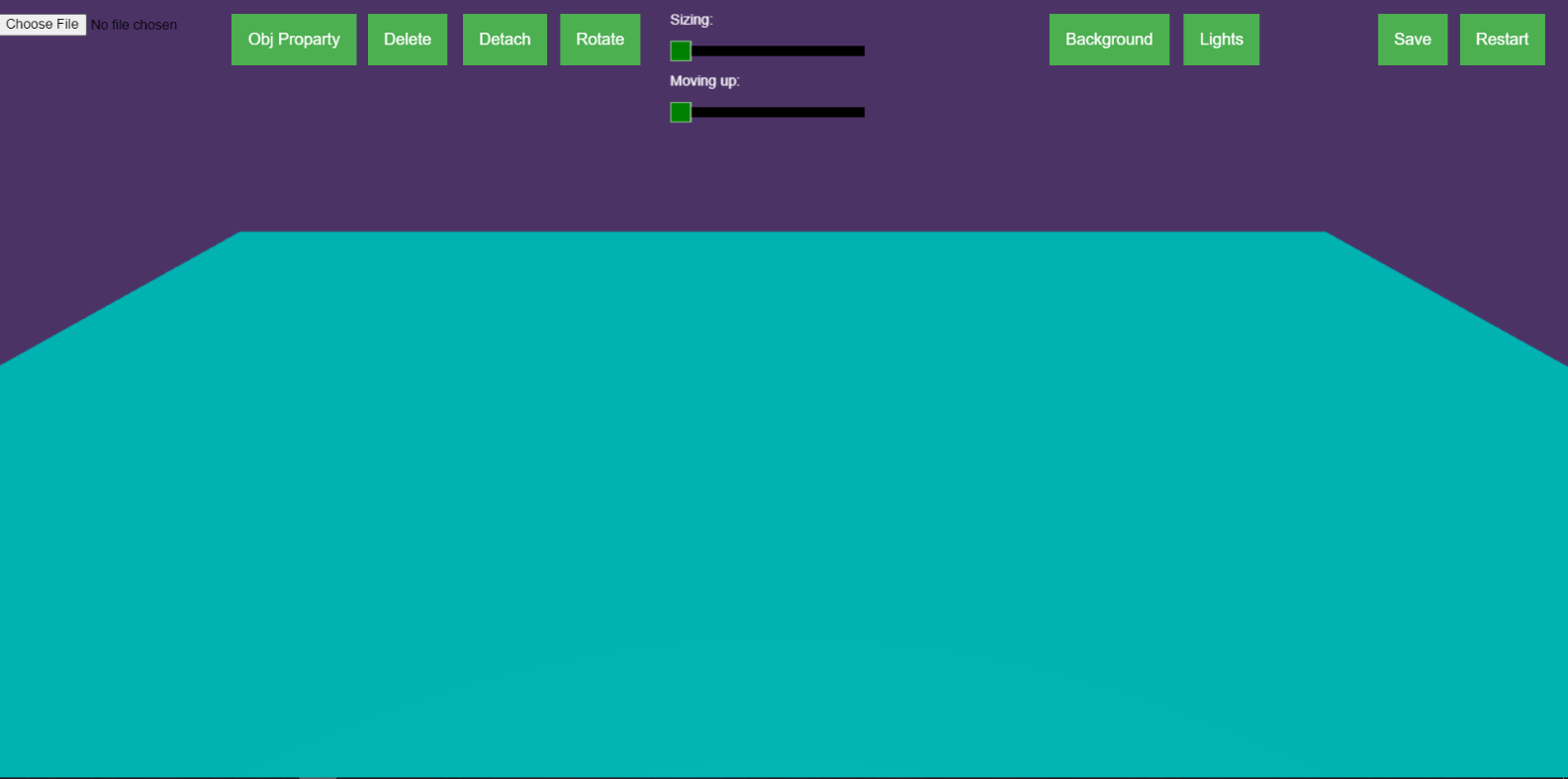
**Фигура 3.19 Код за селектиране на няколко обекта**

Четвърта глава

# Ръководство на потребителя

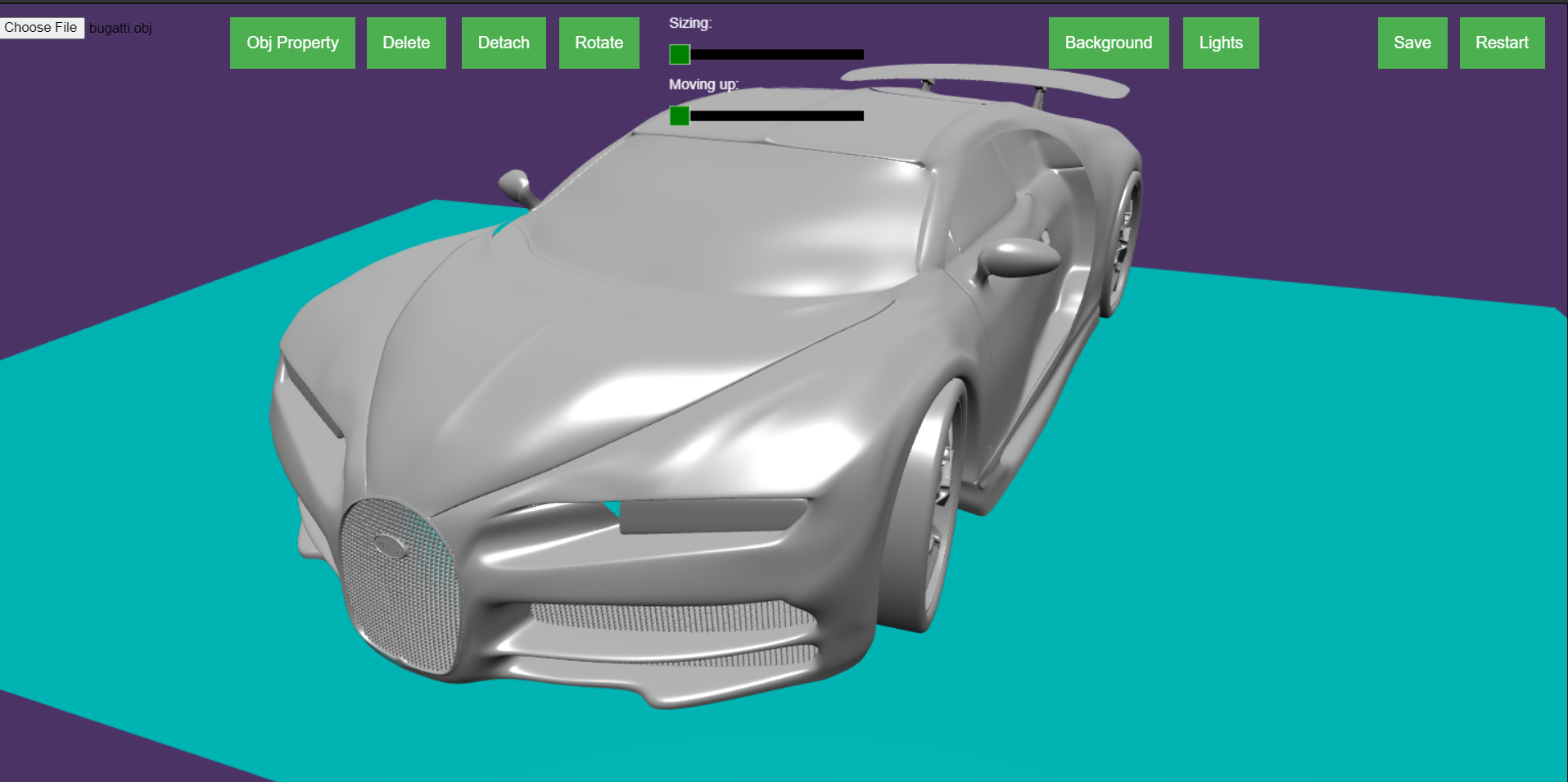
## Използване на уебстраницата

От началната страница (показана на фигура 4.1 отдолу) потребителят може да качи 3D обект, който е запазен някъде в директорията му, като използва Choose File бутонът, който е разположен най-горе в ляво.



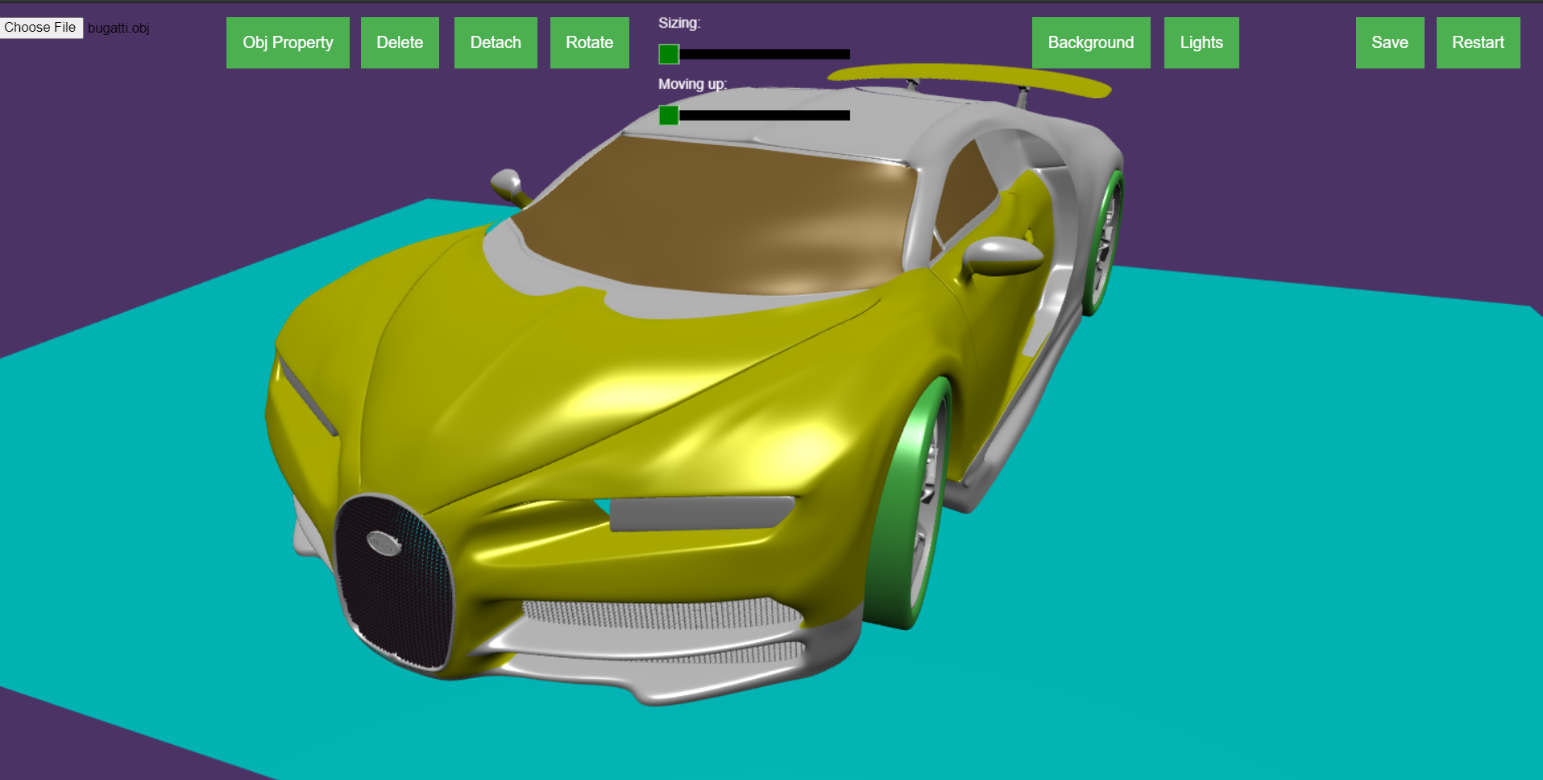
**Фигура 4.1 Първоначална страница**

От там след като потребителят е качил свой 3D обект (пример на фигура 4.2), той може да почне да го персонализира.



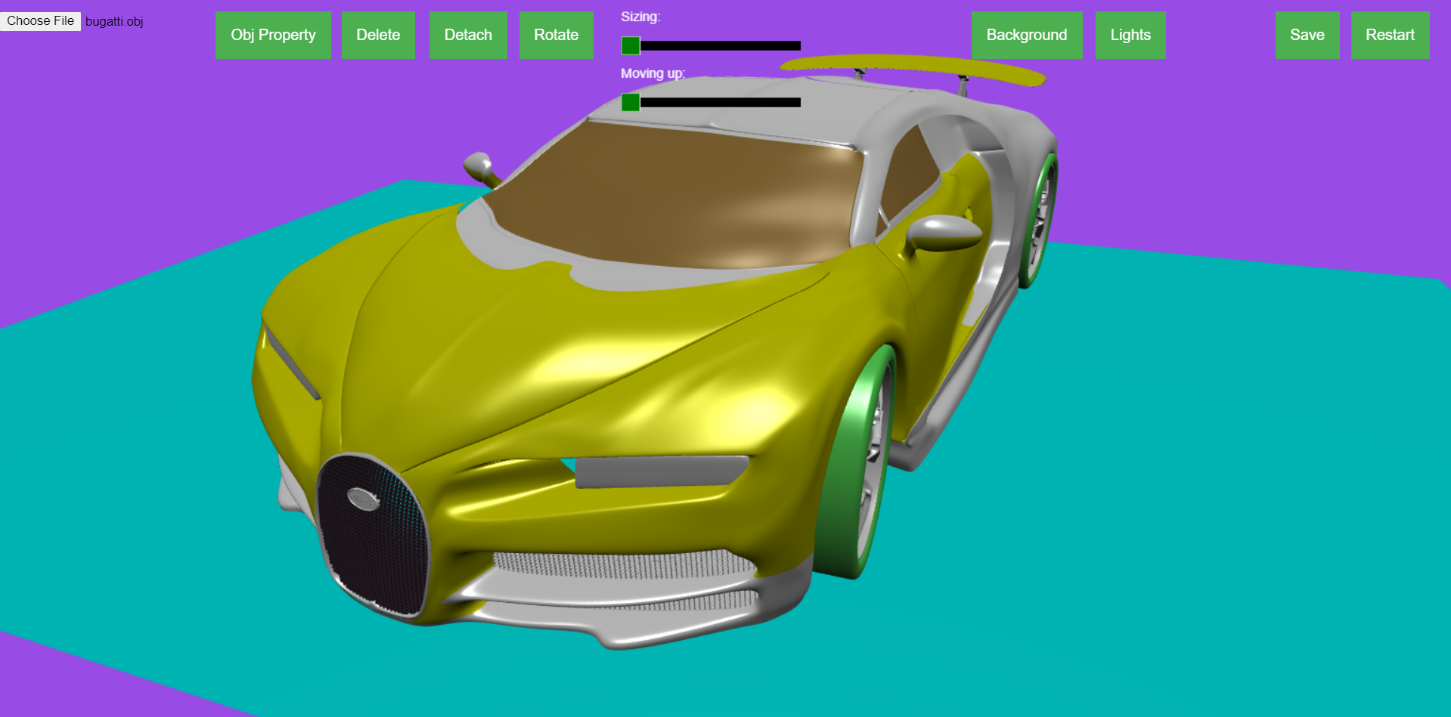
**Фигура 4.2 Качен примерен 3D обект на сцената**

После от падащото меню Obj Property потребителят може да смени текстурата или цвета на всяка част от обекта, както е показано на фигура 4.3.



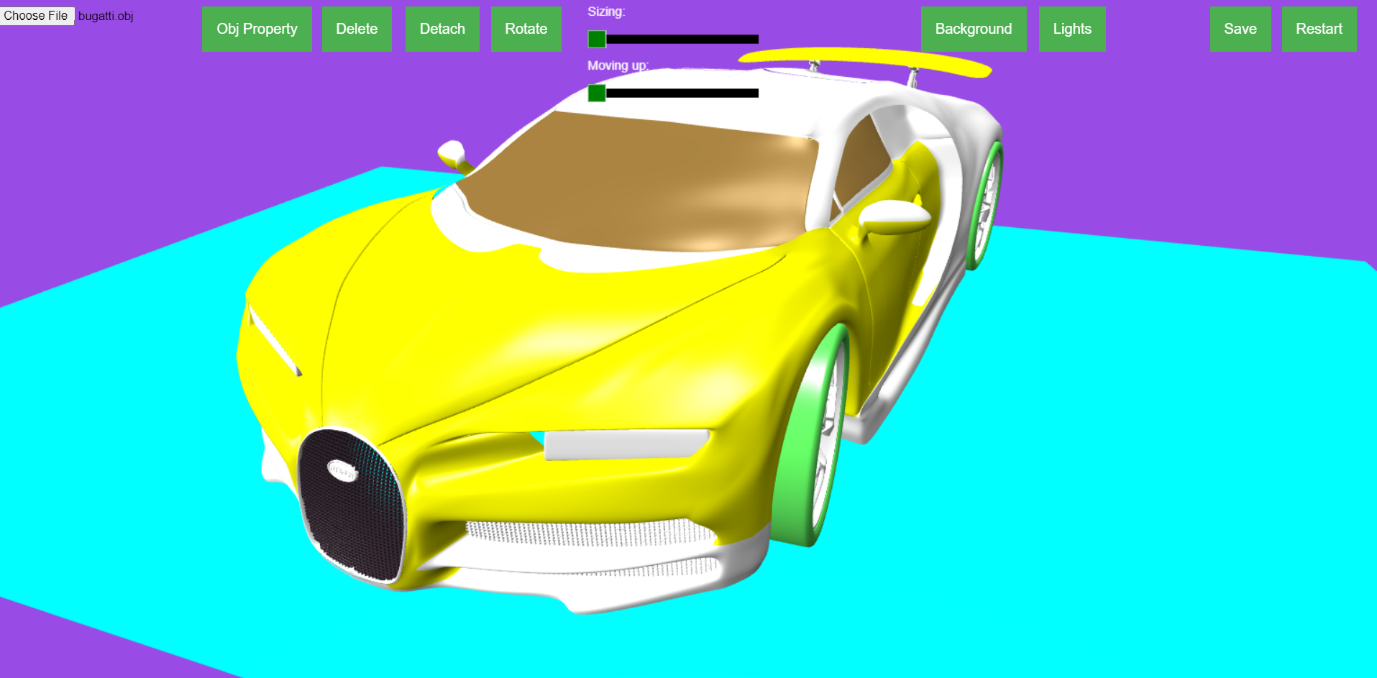
**Фигура 4.3 Пример за прилагане на текстура и цвят**

После може да бъде променен цветът на фона чрез падащото меню на Background (онагледено на фигура 4.4).



**Фигура 4.4 Смяна на цвета на фона**

От падащото меню Lights може да бъдат избрани, кои от светлините да светят, за да може обекта да изглежда по-добре. Това е показано на фигура 4.5.



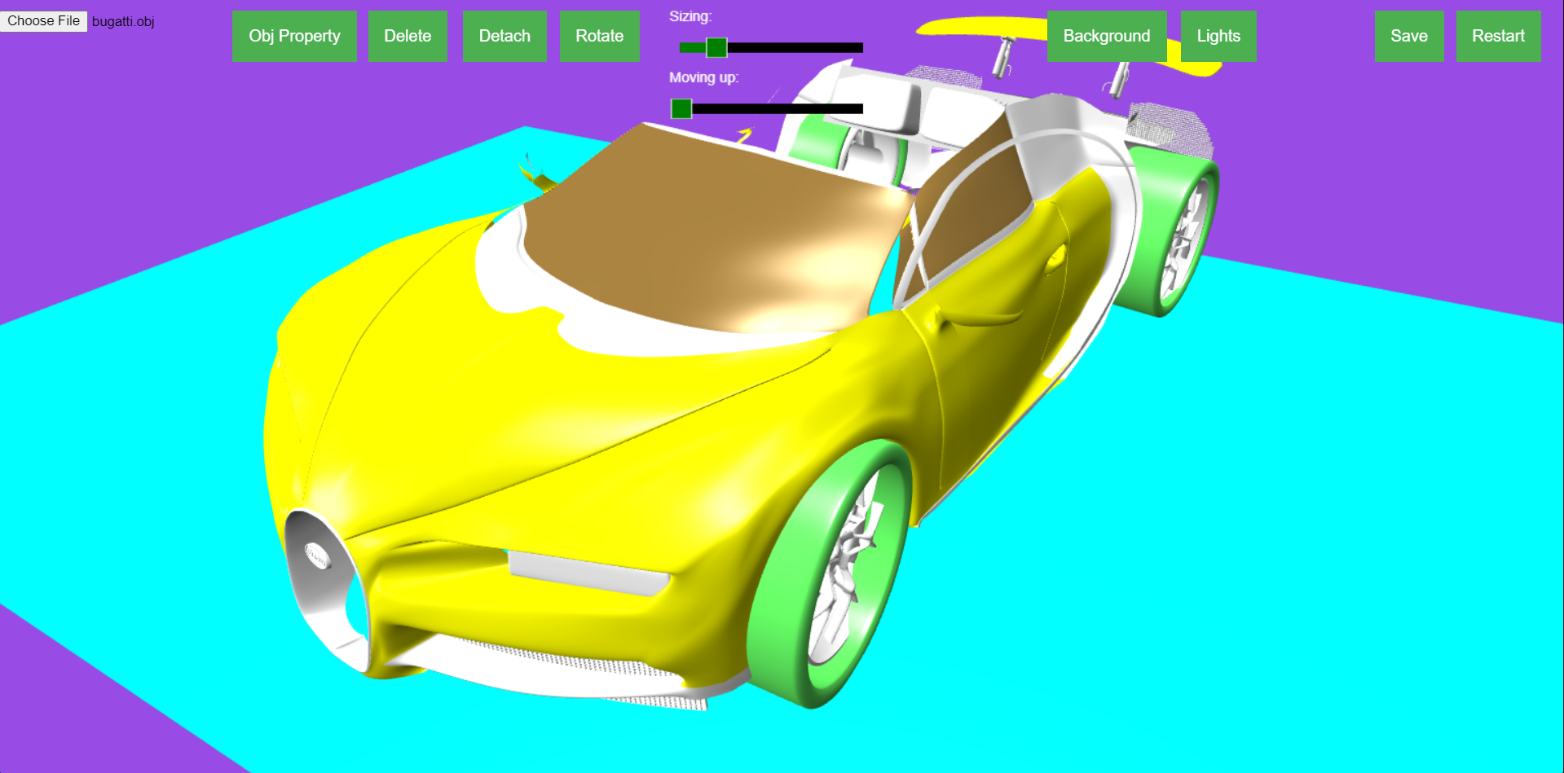
**Фигура 4.5 Избор на светлинен източник**

Чрез задържане на клавиша на клавиатурата Alt и селектиране на някои от обектите, може те да се движат заедно и да сменят цвета и текстурата си едновременно. После чрез плъзгачите Sizing и Мoving up можем да променим големината на обекта и положението му по нагоре и надолу. Това е показано на фигура 4.6.



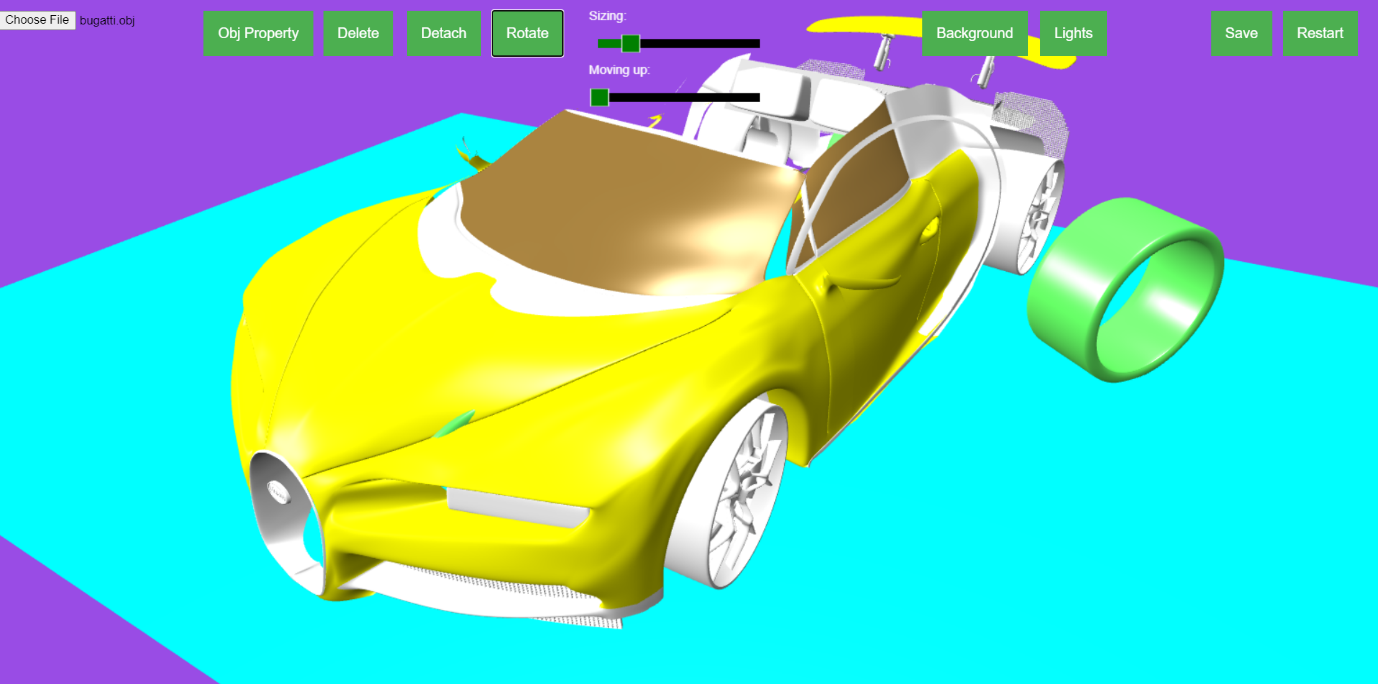
**Фигура 4.6 Промяна на размера и позицията на гумите на Бугатито**

3D обекти могат да бъдат махнати с помощта на бутона Delete, както е показано на фигура 4.7.



**Фигура 4.7 Изтриване на тавана на това Бугати**

Чрез бутонът Rotate може да се завърти всеки един обект на колата (показано на фигура 4.8).



**Фигура 4.8 Завъртане на гумите на колата**

Чрез бутона рестарт може да се рестартира сцената.

Накрая след като потребителят е готов може да натисне Save бутона и да запамети своя персонализиран 3D обект.

# Заключение

В настоящата дипломна работа успешно бе разработено уебсайт, който предоставя възможност на потребителите да персонализира 3D обекти. Приложението не покрива всички заложени функционални изисквания.

Потребителят може да качи 3D обект от своя компютър и да слага върху частите му - текстури и цветове. Обектът може да бъде движен по повърхността на подложката, както и нагоре и надолу.

Големината на обекта може да бъде ръчно избрана от плъзгача.

Потребителят може да нагласи сцената по негово усмотрение, като променя цвета на фона и избира коя от светлините да работи, за да може неговият краен продукт да изглежда най-добре.

При необходимост може да бъде изтрита всяка една от частите на обекта, както може и да бъде завъртяна.

Ако е необходимо сцената може да бъде рестартирана, което означава, че светлините ще бъдат изгасени, фона ще върне първоначалния си цвят и обекта, който е бил местен и завъртан ще си върне предишната позиция.

# Използвана литература и ресурси

1. <https://help.vwo.com/hc/en-us/articles/360020735334>
2. [https://graphics.fandom.com/wiki/3D\_computer\_graphics - Bing](https://graphics.fandom.com/wiki/3D_computer_graphics%20-%20Bing)
3. <https://3dbgprint.com/geometrichnata-teoriq-pri-3d-modeliraneto/>
4. <https://www.techopedia.com/definition/9163/rendering>
5. <https://bul.comparisonnetwork123.com/trehmernaya-grafika-eto-chto-takoe-read-743860>
6. <https://www.reviversoft.com/bg/file-extensions/stl>
7. <https://www.reviversoft.com/bg/file-extensions/obj?ncr=1>
8. <https://www.computerhope.com/jargon/h/html.htm>
9. <https://skillcrush.com/blog/css/>
10. <https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript>
11. <https://www.babylonjs.com/>
12. <https://medium.com/@pailhead011/what-is-three-js-7a03d84d9489>
13. [WebGL: 2D and 3D graphics for the web - Web APIs | MDN (mozilla.org)](https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API)
14. <https://www.educba.com/advantages-of-html/>
15. <https://www.geeksforgeeks.org/advantages-and-disadvantages-of-css/>

# Съдържание

[Увод 4](#_Toc66814004)

[1 Методи и технологии за реализиране на уеб приложения 5](#_Toc66814005)

[1.1 Уебстраница 5](#_Toc66814006)

[1.2 3D компютърната графика 5](#_Toc66814007)

[1.2.1 Моделиране 6](#_Toc66814008)

[1.2.2 Рендериране 7](#_Toc66814009)

[1.2.3 Материали и текстури 7](#_Toc66814010)

[1.3 Формати 8](#_Toc66814011)

[1.3.1 STL 8](#_Toc66814012)

[1.3.2 OBJ 9](#_Toc66814013)

[1.4 Езици за програмиране на уебстраница 9](#_Toc66814014)

[1.4.1 HTML 9](#_Toc66814015)

[1.4.2 CSS 10](#_Toc66814016)

[1.4.3 JavaScript 10](#_Toc66814017)

[1.4.4 BabylonJS 11](#_Toc66814018)

[1.4.5 Three.js 13](#_Toc66814019)

[1.4.6 WebGL 14](#_Toc66814020)

[1.5 Сходни Проекти 15](#_Toc66814021)

[1.5.1 Онлайн конфигуратор 15](#_Toc66814022)

[1.5.2 Сходен Конфигуратор на продукти 16](#_Toc66814023)

[2 Изисквания към приложението. Проектиране на структурата на Конфигуратор на продукти 17](#_Toc66814024)

[2.1 Функционални изисквания към конфигуратора за продукти 17](#_Toc66814025)

[2.1.1 Качване на 3D модел 17](#_Toc66814026)

[2.1.2 Местене на обекта 17](#_Toc66814027)

[2.1.3 Завъртане на обекта 17](#_Toc66814028)

[2.1.4 Reset на сцената 17](#_Toc66814029)

[2.1.5 Промяна на светлината 18](#_Toc66814030)

[2.1.6 Промяна на цветовете на средата 18](#_Toc66814031)

[2.1.7 Смяна на цвета и текстурата на модела 18](#_Toc66814032)

[2.1.8 Смяна на размера на обекта 18](#_Toc66814033)

[2.1.9 Настройка за подмяна на обекта 18](#_Toc66814034)

[2.1.10 Подреждане на възможните настройки в процес 18](#_Toc66814035)

[2.1.11 Запазване и зареждане след всяка стъпка от редакцията 19](#_Toc66814036)

[2.1.12 Експортиране в HTML формат на конфигуратора 19](#_Toc66814037)

[2.2 Аргументация на избора на средите за разработка 19](#_Toc66814038)

[2.2.1 HTML 19](#_Toc66814039)

[2.2.2 CSS 19](#_Toc66814040)

[2.2.3 JavaScript 20](#_Toc66814041)

[2.2.4 BabylonJS 20](#_Toc66814042)

[2.3 Схема на приложението 21](#_Toc66814043)

[3 Разработка на уеб приложението 22](#_Toc66814044)

[3.1 Начална сцена 22](#_Toc66814045)

[3.2 Поставяне на бутони 23](#_Toc66814046)

[3.3 Поставяне на бутон за качване на 3D обект 24](#_Toc66814047)

[3.4 Мърдане на 3D обекта 25](#_Toc66814048)

[3.5 Завъртане 27](#_Toc66814049)

[3.6 Reset на сцената 27](#_Toc66814050)

[3.7 Светлини 28](#_Toc66814051)

[3.8 Смяна на цвета на сцената 29](#_Toc66814052)

[3.9 Смяна на цвета на обекта 30](#_Toc66814053)

[3.10 Смяна на текстурата на обекта 30](#_Toc66814054)

[3.11 Смяна на размера на обекта 31](#_Toc66814055)

[3.12 Настройка за смяна на обекта 32](#_Toc66814056)

[3.13 Експортиране на сцената 32](#_Toc66814057)

[4 Ръководство на потребителя 33](#_Toc66814058)

[4.1 Използване на уебстраницата 33](#_Toc66814059)

[Заключение 38](#_Toc66814060)

[Използвана литература и ресурси 39](#_Toc66814061)

[Съдържание 40](#_Toc66814062)