Использование 3D-объектов в статических сценах
4.1. Файлы OBJ

Формат файла OBJ является одним из наиболее важных форматов файлов в приложениях 3D-печати и 3D-графики. Это предпочтительный формат для разноцветной 3D-печати и широко используется в качестве нейтрального формата обмена для неанимированных 3D-моделей в графических приложениях.

Файл OBJ — это стандартный формат 3D-изображений, который может быть экспортирован и открыт различными программами 3D-редактирования. Он содержит трехмерный объект, который включает в себя 3D-координаты, текстурные карты, полигональные грани и другую информацию об объекте.

Короче говоря, формат файла OBJ хранит информацию о 3D-моделях. Он может кодировать геометрию поверхности 3D-модели, а также хранить информацию о цвете и текстуре. Этот формат не хранит никакой информации о сцене, такой как положение света или анимация.

Файл OBJ обычно генерируется программным обеспечением CAD (Computer Aided Design) в качестве конечного продукта процесса 3D-моделирования. Расширение файла, соответствующее формату файла OBJ, просто ".obj".

Формат файла OBJ нейтрален и является открытым исходным кодом. Он часто используется для обмена 3D-моделями в графических приложениях, поскольку он пользуется хорошей поддержкой импорта и экспорта почти всего программного обеспечения CAD. Он также используется в качестве формата файла для разноцветной 3D-печати, поскольку стандартный формат печати 3D STL не включает информацию о цвете и текстуре.

Формат файла OBJ был первоначально создан Wavefront Technologies для приложения Advanced Visualizer для хранения геометрических объектов, состоящих из линий, полигонов и кривых, а также поверхностей свободной формы. Последняя документированная версия - v3.0, заменившая предыдущую версию v2.11.

Доминирующим форматом в мире 3D-печати является STL. Однако STL – это старый формат файлов, который, хотя и пользуется большой популярностью, не учитывает современных требований. Точность 3D-печати быстро приближается к разрешению на микронном уровне, и многоцветные модели становятся все более популярными. Формат файла STL не поддерживает высокое разрешение очень хорошо, так как более высокое разрешение приводит к увеличению размера файла. Кроме того, формат STL не подходит для разноцветной 3D-печати, поскольку он не поддерживает информацию о цвете и текстуре. Напротив, формат OBJ может довольно точно воспроизвести геометрию поверхности, не оказывая существенного влияния на размер файла. Это возможно с помощью кривых Безье и метода под названием NURBS. Кроме того, формат файла OBJ имеет встроенную поддержку нескольких цветов и текстур одной и той же модели.

Таким образом, использование формата OBJ имеет ряд преимуществ по сравнению с использованием формата STL в случае, если вам нужны многоцветные модели высокого разрешения. С другой стороны, формат файла OBJ не так универсален, как формат STL. Почти все 3D-принтеры поддерживают формат STL. То же самое нельзя сказать о формате OBJ, даже если он поддерживается печатным оборудованием. Поэтому, если нужно напечатать монохромную 3D-модель на стандартном принтере, предпочтительнее формат STL.

Оба формата OBJ и STL имеют широкий спектр использования с большой базой лояльных пользователей и пользуются поддержкой множества сторонних приложений.

Основными конкурентами в случае файловых форматов для 3D-печати являются VRML, AMF и 3MF, которые, однако, не пользуются такой поддержкой и совместимостью и поэтому не являются серьезными альтернативами форматам файлов STL и OBJ.

Формат файла OBJ используется в приложениях 3D-графики. Наиболее широко используемыми форматами файлов в приложениях 3D-графики являются OBJ, FBX и COLLADA.

Различия между форматами файлов OBJ, FBX, COLLADA

Наиболее важным различием между форматом файла OBJ и двумя другими (FBX и COLLADA) является поддержка информации о сцене, такой как источники света и анимации. Формат файла OBJ не содержит информации о сценах и анимациях, в то время как FBX и COLLADA содержат. Поэтому, если вы хотите сохранить поддержку анимации в случае игр или кино, то рекомендуется использовать формат FBX или COLLADA. Если, однако, нет необходимости в сложной сцене или анимации, можно использовать формат файла OBJ.

Преимущества формата obj

Прежде всего, формат файла OBJ является простым и открытым форматом. Он имеет широкую поддержку экспорта и импорта среди САПР. Это значит, что если вы делитесь 3D-моделью в виде файла OBJ, то другие программы САПР будут интерпретировать ее правильно и последовательно. То же самое нельзя сказать о форматах FBX или COLLADA. Формат Collada также является открытым исходным кодом, но он довольно сложен. Различное программное обеспечение САПР интерпретирует его по-разному, и это может привести к ошибкам.

Формат FBX является закрытым форматом, но он предоставляет SDK (software development kit) для преобразования существующих форматов в FBX. Преобразование файла FBX в другой формат, довольно сложно и может сопровождаться появлением ошибок.

Файл OBJ будет намного проще и меньше по размеру по сравнению с файлом FBX или COLLADA, содержащими ту же самую 3D-модель. Это связано с простотой формата файла OBJ по сравнению с другими спецификациями и из-за его собственной двоичной кодировки.

Таким образом, если нет необходимости в сложной сцене или анимации, но гораздо больше нужна поддержка и правильная интерпретация различными программами CAD, формат файла OBJ является предпочтительным форматом. Практически во всех остальных случаях FBX является оптимальным форматом для 3D-графических приложений.

Современные возможности формата файлов obj

С точки зрения современных функций, формат FBX является наиболее прогрессивным форматом, который предлагает множество современных функций, обновлений и регулярных улучшений. Формат файла OBJ занимает второе место с точки зрения функций, в то время как формат COLLADA не так часто обновляется и улучшается.

Геометрия

Основной целью формата файла OBJ является кодирование геометрии поверхности 3D-объекта. Формат файла OBJ довольно универсален в этом отношении. Он допускает несколько вариантов кодирования геометрии поверхности. Ниже приведены несколько методов кодирования, их преимущества и недостатки.

Тесселяция поверхности с полигональными гранями (tessellation - мозаика)

В своей простейшей форме формат файла OBJ позволяет пользователю разбить поверхность 3D-модели в простые геометрические фигуры, такими как треугольники, четырехугольники или более сложные многоугольники. Вершины полигонов и стороны каждого многоугольника затем сохраняются в файле, содержащем геометрию поверхности модели.

Тесселяция с полигональными гранями имеет ряд преимуществ и недостатков. Полигоны — это простые геометрические фигуры, и этот метод на самом деле является самым простым способом описания геометрии поверхности. Однако приближение изогнутой поверхности к многоугольникам приводит к изменению толщины рисунка.

В случае печати 3D-принтер напечатает объект с той же толщиной, указанной файлом. Конечно, делая треугольники все меньше и меньше, приближение может быть сделано все более и более точно, что приводит к напечатанной модели отличного качества. Однако по мере уменьшения размера треугольника уменьшается и количество треугольников, необходимых для описания поверхности. Это приводит к значительному увеличению размера

файла, что и пытаются решить слайсеры 3D-печати. Также становится трудно распространять или управлять такими огромными файлами.

Поэтому очень важно найти правильный баланс между размером файла и качеством печати. Нет смысла уменьшать размеры треугольников до бесконечности, так как в какой-то момент уже не удастся различить невооруженным глазом разницу между качествами печати.

Расширение файла .obj в первую очередь относится к Wavefront 3D (.obj), разработанному Wavefront Technologies для передового программного обеспечения Advanced Visualizer. Формат OBJ — это текстовый формат для описания геометрии трехмерных тел, который позволяет моделировать сложные объемные формы и применять различные материалы и текстуры. В дополнение к платформе .OBJ, типичный 3D-объект или сцена Wavefront, обычно включает в себя один или несколько файлов библиотеки шаблонов материалов (.mtl), которые определяют материалы объекта со ссылками на внешние растровые текстуры, обычно хранящиеся в отдельном подкаталоге.

Формат OBJ стал одним из самых популярных и поддерживаемых форматов 3D-моделей и функций экспорта/импорта файлов. OBJ присутствуют почти в каждом 3D-редакторе. Довольно много утилит для просмотра 3D-моделей (есть даже веб-приложения, которые позволяют импортировать, просматривать, редактировать, экспортировать и конвертировать файлы. OBJ) предлагает возможность открытия файлов. OBJ и отображение содержащихся моделей с полным воспроизведением, а также серия конверторов позволяющих сохранять модели OBJ в другие форматы. В интернете существуют целые коллекции и библиотеки моделей в этом формате. Формат геометрии OBJ является открытым форматом для файлов описания геометрии.

Спецификация OBJ

Формат файла OBJ является форматом файла ASCII. Его редактирование можно сделать с помощью любого текстового редактора.

В исходной спецификации явно не указано, каким должен быть символ конца строки, поэтому некоторые программы используют \r return de car (CR), а другие используют \n new line (NL или LF). Иногда, требуется преобразовать символы, указывающие на конец строки, если редактор или программное обеспечение не могут прочитать файл.

Первый символ каждой строки очень важен, поскольку он указывает тип команды. Если первый символ #, эта строка представляет комментарий, а все остальное в этой строке игнорируется. Пустые строки также игнорируются.

Команда "комментарий"

строка комментариев

Как упоминалось ранее, символ **#** указывает, что строка представляет собой комментарий и должна быть проигнорирована. Первая строка обычно всегда представляет собой комментарий, который указывает, какая программа создала файл.

Команда "вершина"

vn x y z

В эту упрощенную версию спецификации будут включены только полигональные грани, поэтому для указания вершин полигональных вершин можно использовать команду **vertex-v**. При использовании поверхностей и кривых произвольной формы существует аналогичная команда **vp**, которую можно использовать для указания контрольных точек поверхности или кривой.

Команда **vertex-v** задает вершину через три декартовые координаты **x**, **y** и **z**. Вершине "*i*" автоматически присваивается имя в зависимости от порядка, в котором она находится в файле. Первая вершина в файле получает имя «1», вторая — «2», третья — «3» и так далее.

Команда "нормаль к вершине"

vn x y z

Vertex normal – vn — команда для нормали к вершине. Задает нормальный вектор к поверхности. Параметры **x**, **y** и **z** являются компонентами нормального вектора. Этот нормальный вектор не связан ни с одной вершиной. Позже его нужно будет связать с одной из вершин с помощью другой команды, называемой командой **f**.

Команда normal vertex может быть опущена в наборе файлов, так как при группировании вершин в полигональные грани с **помощью команды f** автоматически определяется нормальный вектор в координатах вершины и порядок, в котором будут появляться вершины.

Команда "текстура вершины"

vt u v [w]

Команда **vertex texture - vt** указывает точку на текстурной карте. Параметры **u** и **v** являются координатами **x** и **y** на текстурной карте. Это будут числа с плавающей запятой между 0 и 1. Они должны быть связаны с вершиной при помощи контрольного **f**, аналогично нормалям к вершинам.

Команда "поверхность"

f v1 [/ vt1] [/ vn1] v2 [/ vt2] [/ vn2] v3 [/ vt3] [/ vn3] ...

Команда **surface - f,** вероятно, самая важная команда. Задает полигональную поверхность, созданную из ранее заданных вершин.

Чтобы ссылаться на вершину, нужно следовать системе нумерации вершин по умолчанию. Например, «f 23 24 25 27» описывает полигональную поверхность, построенную из вершин 23, 24, 25 и 27 по порядку.

Для каждой вершины может быть приписана команда **vn**, которая затем связывает эту нормаль с соответствующей вершиной. Аналогично, команда **vt** может быть связана с вершиной, которая будет определять отображение текстуры, используемой для этой вершины.

Если указана текстура или нормаль для одной вершины, то они должны быть указаны для всех вершин.

Команда "группировать"

g имя

Команда **group - g** указывает, что часть объекта сгруппирована в группу с именем, указанным в качестве параметра. Все следующие команды **f** будут включены в одну группу. Это полезно, если требуется повторно использовать определенную информацию, такую как тип материала, для выбранной части объекта.

Команда "материал"

имя usemtl

Команда **материал – usemtl** позволяет указывать используемый материал. Все последующие команды **f** будут использовать один и тот же материал до тех пор, пока не появится другая команда **usemtl**.

4.2. Библиотека материалов

Информация о внешнем виде объектов (материалов), содержащихся в файле OBJ, находится в отдельных файлах в формате MTL (Material Library). Файл OBJ ссылается на такой файл, если это необходимо, используя директиву:

Mtllib [имя внешнего MTL-файла]

MTL — это стандарт, установленный Wavefront Technologies. Вся информация представлена в форме ASCII и может быть прочитана. Стандарт MTL очень популярен и поддерживается большинством пакетов 3D-графики.

Информация о простых материалах в файле выглядит следующим образом:

```
Newmtl material_name1
# Анонс следующего материала
# Цвета Ka 1.000 1.000 0.000
# Цвет окружающего света (желтый) Kd 1.000 1.000 1.000
# Рассеянный цвет (белый)
# Параметры отражения Ks 0.000 0.000 0.000
# Цвет зеркального отражения (0; 0; 0 - выкл.) Ns 10.000
# Зеркальный коэффициент отражения (от 0 до 1000)
# Параметры прозрачности d 0.9
# Прозрачность задается с помощью директивы d Tr 0.9
# или в других реализациях формата с использованием Tr
# Следующий материал newmtl material_name2 ...
```

Все параметры являются необязательными. При отсутствии какого-либо параметра программа автоматически устанавливает его по умолчанию.

Учитывая все вышесказанное, ниже приведено содержимое простого файла OBJ (CUB), встроенного в редактор Google SketchUp 3D (рис. 4.1) и экспортируемого как фаил .OBJ при помощи плагина *LIPID Lightsolve*, который можно легко найти и установить из меню *Window>Extemsion Warehouse*. Чтобы иметь возможность установить любой плагин из библиотеки расширений, пользователь должен иметь учетную запись Google и войти в систему.

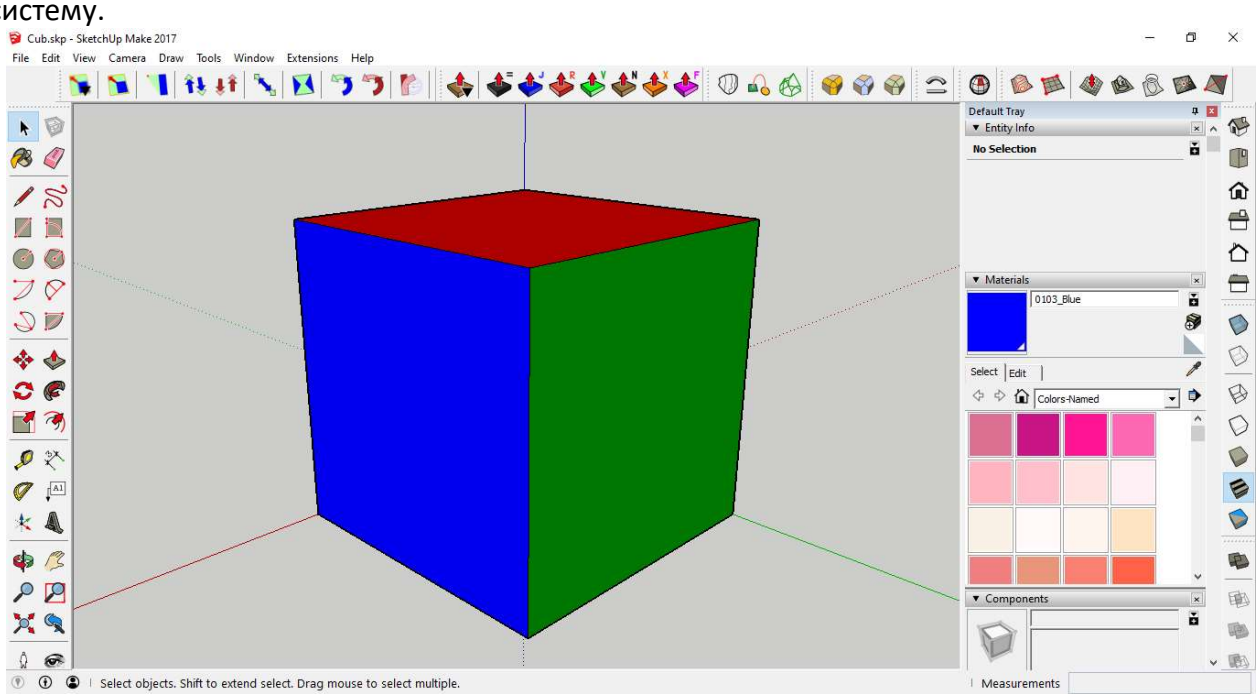


Рис. 4.1. Модель куба, построенная в 3D-редакторе Google SketchUp

Чтобы экспортировать 3D-модель куба в формате OBJ, нужно перейти по ссылке: File-> LIPID OBJ Exporter, (рис. 4.2), а затем в окне LIPIDOBJ установите флажок Ignore Back face.

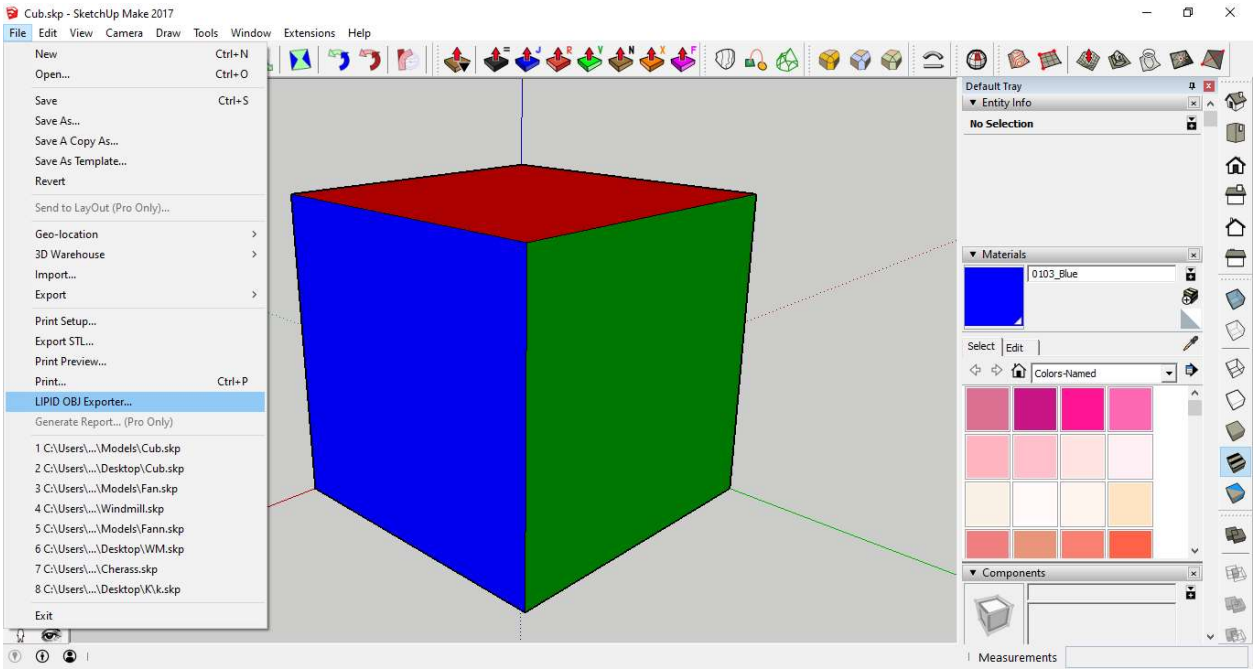


Рис. 4.2. Экспорт модели куба в формате OBJ с помощью плагина LIPIDOBJ

Содержимое файла .OBJ модели куба, показанного на рисунке 4.2:

```
mtllib /Cub.mtl
s 1
o
g
usemtl 0020_Red
v 100 100 0
v 0 0 0
v 0 100 0
vn 0 0 -1
f 1//1 2//1 3//1
v 100 0 0
f 2//1 1//1 4//1
v 100 0 100
v 0 100 100
v 0 0 100
vn 0 0 1
f 5//2 6//2 7//2
v 100 100 100
f 6//2 5//2 8//2
usemtl 0076_Green
vn 0 -1 0
f 5//3 2//3 4//3
f 2//3 5//3 7//3
vn 0 1 0
f 6//4 1//4 3//4
f 1//4 6//4 8//4
usemtl 0103_Blue
vn -1 0 0
f 6//5 2//5 7//5
f 2//5 6//5 3//5
vn 1 0 0
f 1//6 5//6 4//6
f 5//6 1//6 8//6
```

Анализируя содержимое представленного выше файла .OBJ можно увидеть, что в файле Cub.obj используется файл материала Cub.mtl, расположенный в том же файле. В дополнение к этому вы также можете наблюдать описание координат всех 8 вершин куба, указание поверхностей (треугольников), составляющих 6 сторон куба, со спецификацией их нормалей и атрибуцией текстур *0020_Red, 0076_Green* и *0103_Blue* соответствующих поверхностей (треугольников).

Ниже представлено содержимое файла с материалами, используемые для текстурирования 3D-модели куба, показанного на рис. 4.2.

```
#
## Alias MTL Material File
# Converted from SKP by LIPID Lightsolve
newmtl 0020_Red
Ka 0.000000 0.000000 0.000000
Kd 1 0 0
Ks 0.330000 0.330000 0.330000
newmtl 0076_Green
Ka 0.000000 0.000000 0.000000
Kd 0 0.501961 0
Ks 0.330000 0.330000 0.330000
newmtl 0103_Blue
Ka 0.000000 0.000000 0.000000
Kd 0 0 1
Ks 0.330000 0.330000 0.330000
newmtl default_material
Ka 0.000000 0.000000 0.000000
Kd 0.330000 0.330000 0.330000
Ks 0.330000 0.330000 0.330000
```

Анализ содержимого файла .MTL, показанного выше, определяет текстуры *0020_Red, 0076_Green* и *0103_Blue*, используемые в файле Cub.obj для текстурирования граней куба.

Описание основных объектов и методов работы с файлами. OBJ с помощью библиотеки для создания трехмерной WEB графики в онлайн-редакторе p5.js

Библиотека p5.js — это библиотека JavaScript, которая облегчает процесс разработки программ, использующих элементы 2D и 3D графики и предназначена как для опытных разработчиков, так и для начинающих. Библиотека p5.js бесплатна с открытым исходным кодом.

Библиотека p5.js имеет полный набор графических функций. С помощью этой графической библиотеки вся страница браузера рассматривается как рабочее пространство, где могут использоваться объекты HTML5 для текста, видео, веб-камеры и звука.

Основные функции:

Загрузка в библиотеку p5.js 3D-модели как файл OBJ или STL.

loadModel() должен быть помещен внутрь функции ***preload()***. Это позволяет модели полностью загрузиться перед запуском остальной части программы.

Одним из ограничений формата OBJ и STL является то, что они не учитывают масштаб. Это означает, что модели, экспортируемые из разных программ, могут иметь разные размеры. Если модель не отображается, попробуйте вызвать ***loadModel()*** с параметром нормализации

true. Это изменит размер модели до масштаба, подходящего для p5. Дальнейшие изменения в размере модели также могут быть сделаны с помощью функции ***scale()***.

Также не реализована поддержка цветных STL файлов. Цветные STL-файлы будут отображаться без свойств цвета.

Синтаксис:

loadModel(path, normalize, [successCallback], [failureCallback], [fileType])

loadModel(path, [successCallback], [failureCallback], [fileType])

Параметры:

Path String: путь к загруженной модели;

normalize: если значение равно true, модель масштабируется до стандартного размера при загрузке;

successCallback Function (p5.Geometry): функция, которая должна быть вызвана после загрузки модели. Будет возвращен объект 3D-модели. (Опционально);

failureCallback Function (Event): вызывается в случае появления ошибки, если модель не загружается. (Опционально);

fileType String: расширение файла модели (.stl, .obj). (Опционально);

p5.Geometry: объект типа p5.Geometry.

model() Воспроизводит 3D-модель на экране.

Синтаксис:

model(model)

Параметры:

model p5.Geometry: загруженная 3D-модель для воспроизведения

Sketch Files->Upload file

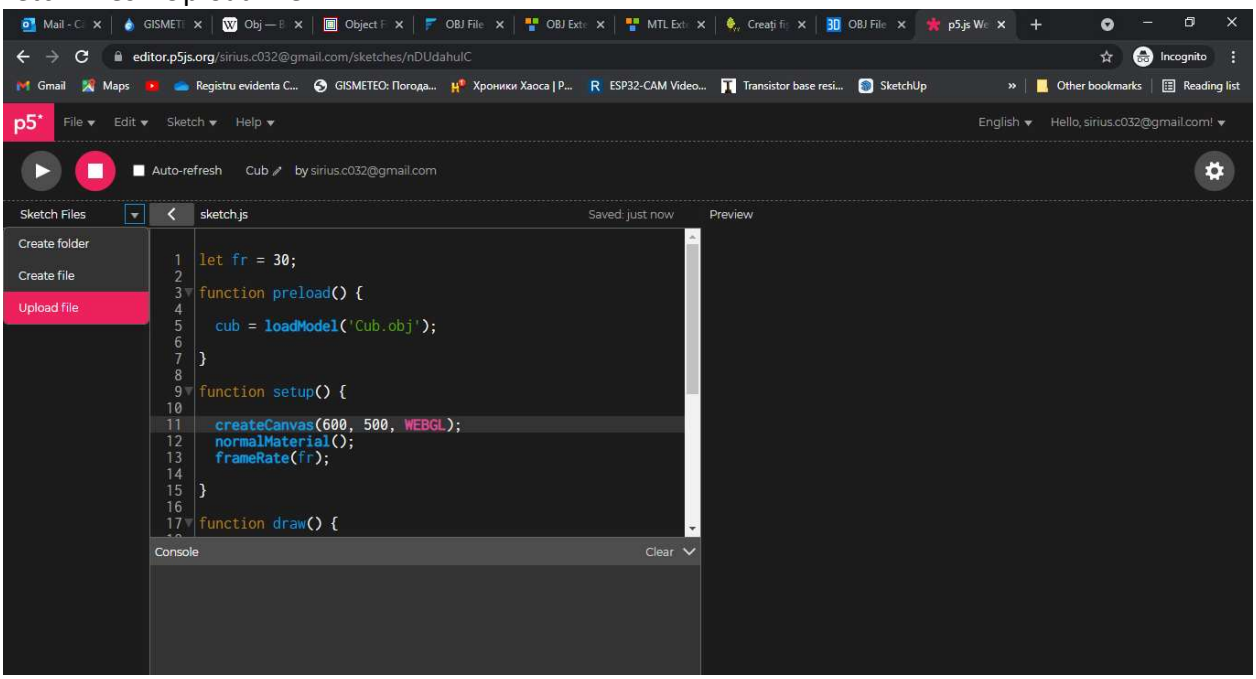


Рис. 4.3. Загрузка файла Cub.obj в каталог проекта

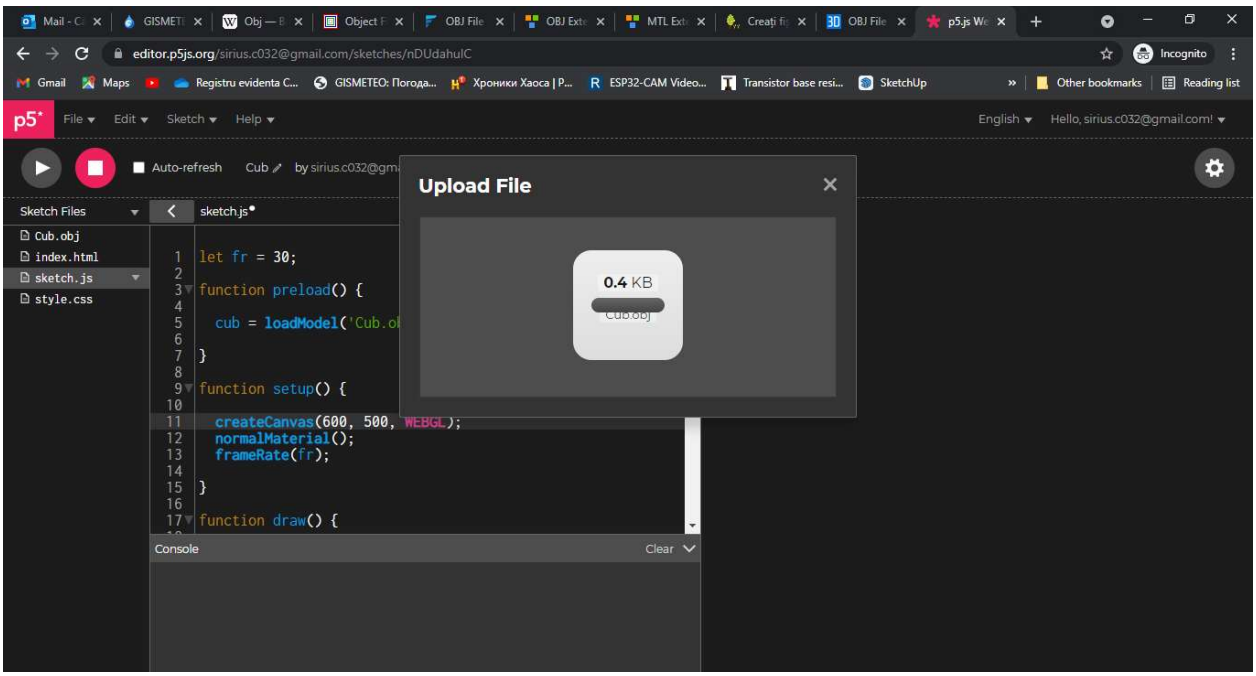


Рис. 4.4. Результат процесса загрузки файла Cub.obj в каталог проекта

```
let fr = 30;
function preload()
{
        cub = loadModel('Cub.obj');
}
function setup()
 {
        createCanvas(600, 500, WEBGL);
        normalMaterial();
        frameRate(fr);
}

function draw()
{
        orbitControl();
        background(50);
        normalMaterial();
        scale(1.0);
        stroke(100, 100, 100);
        strokeWeight(0.3);
        fill(150, 150, 150);
        model(cub);
}
```

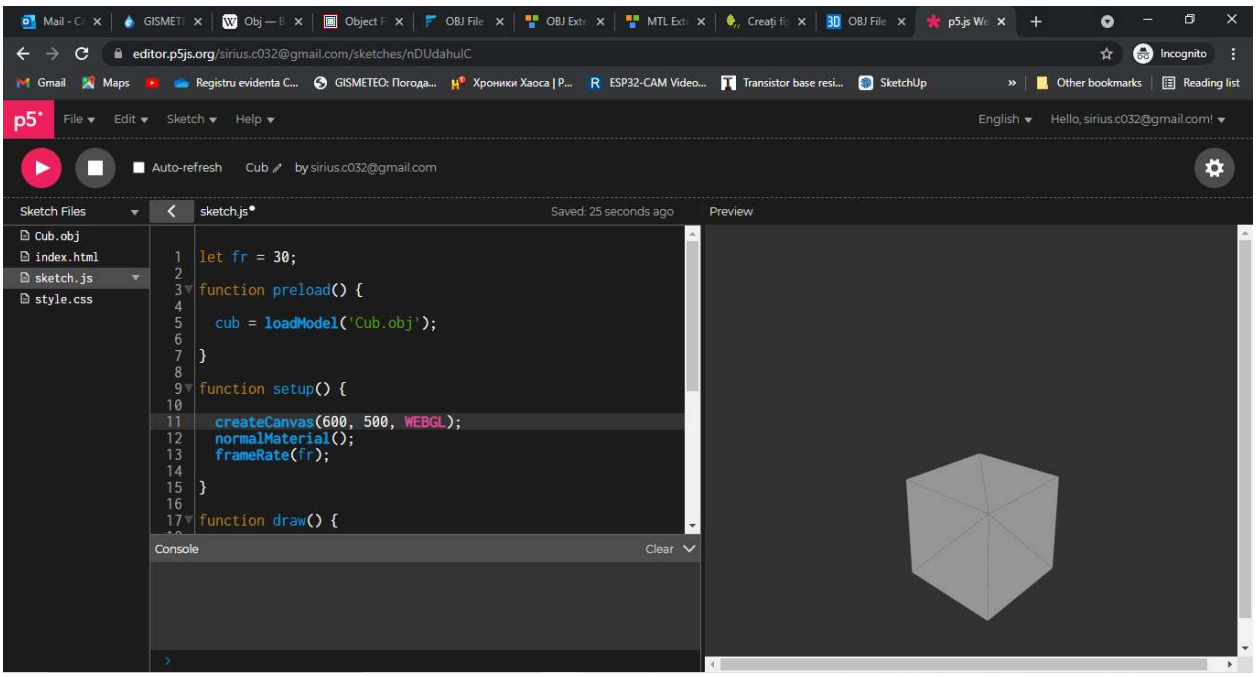


Рис. 4.5. Результат выполнения программы для отображения 3D модели куба

Создание статической 3D-сцены

Задача лабораторных работ заключается в задумке и построении статической 3D-сцены с помощью онлайн-редактора p5.js с использованием моделей 3D-объектов, хранящихся в файлах. OBJ созданных индивидуально или загруженных из Интернета.

Последовательность лабораторной работы №. 4, по созданию статичной сцены, может быть отслежена на примере, который представляет ветряную мельницу, отгороженную забором, во дворе которой будет несколько домашних животных, дерево и стог сена. 3D-модели этих объектов могут быть созданы с нуля или импортированы из репозитория 3D Warehouse, доступ к которому можно получить из базового меню графического редактора Google SketchUp *Window->3D Warehouse*, а затем отредактировать по мере необходимости.

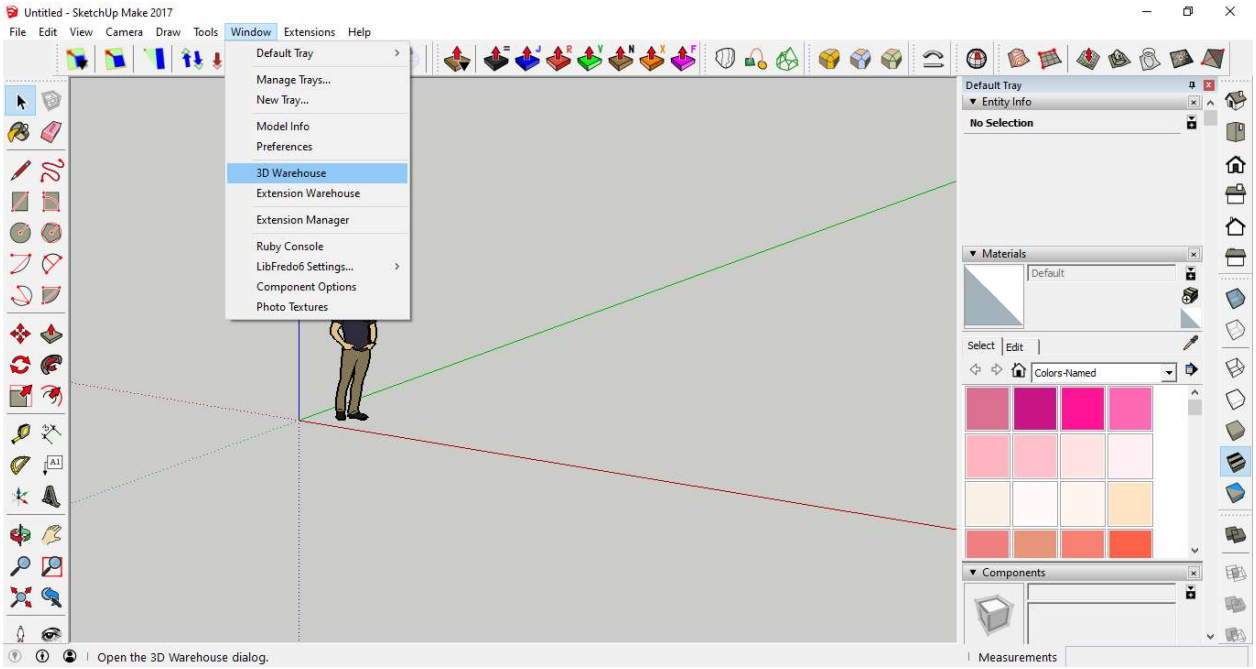


Рис. 4.6. Доступ к репозиторию 3D Warehouse

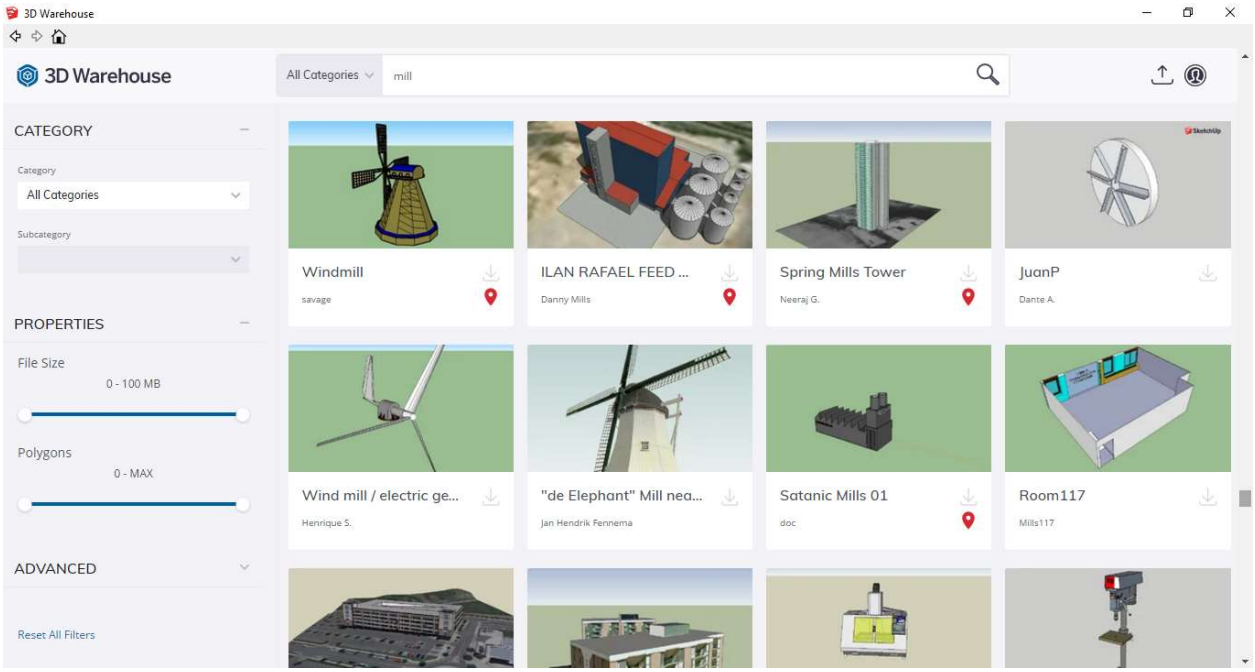


Рис. 4.7. Поиск 3D-моделей в репозитории 3D Warehouse

Для того, чтобы иметь возможность скачать или импортировать выбранную модель в текущем файле, пользователь должен иметь учетную запись Google и войти в среду 3D Warehouse. После загрузки требуемых моделей их можно редактировать по мере необходимости и экспортировать в виде файлов. OBJ через плагин *LIPID OBJ Exporter*, который можно установить из меню *Window->Extension Warehouse* (рис. 4.8), в поле поиска которого мы указываем ключевое слово OBJ и в ответ отображаются плагины, удовлетворяющие критериям поиска, среди которых можно наблюдать поисковый плагин *LIPID OBJ Exporter* (рис. 4.9). Чтобы иметь возможность установить плагины, пользователь должен иметь учетную запись Google и войти в онлайн-репозиторий *Extension Warehouse*.

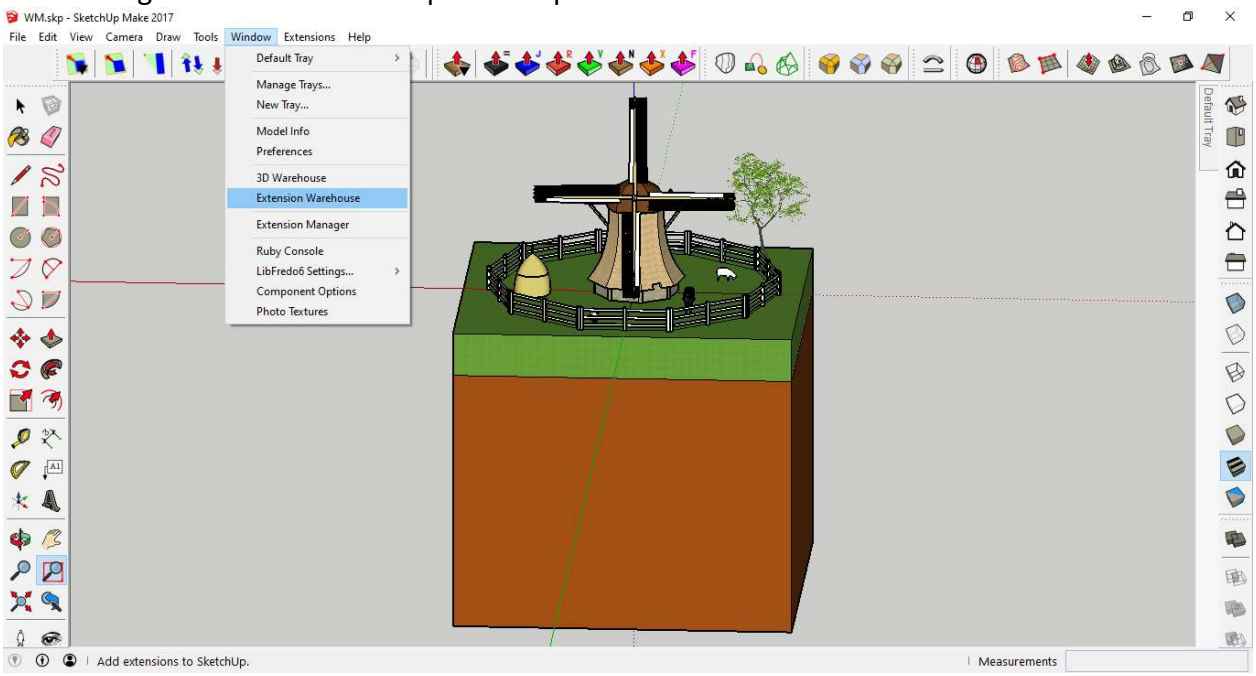


Рис. 4.8. Доступ к «*Extension Warehouse*»

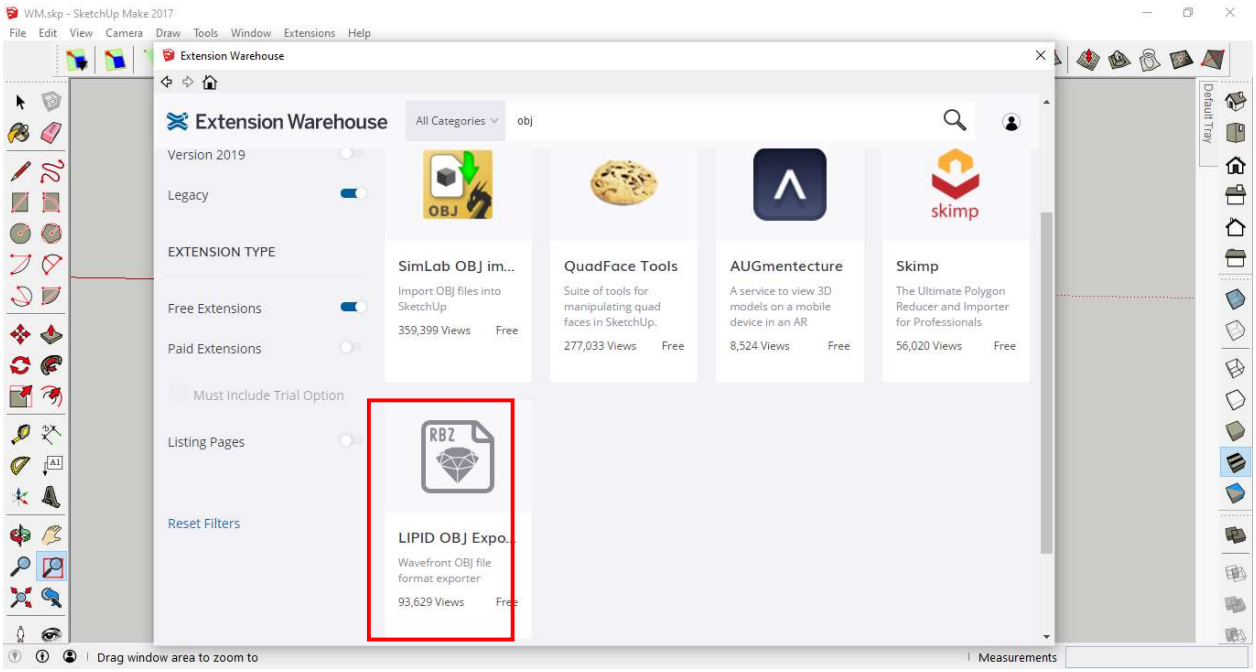


Рис. 4.9. Окно поиска и установки плагина из онлайн-репозитория *Extension Warehouse*

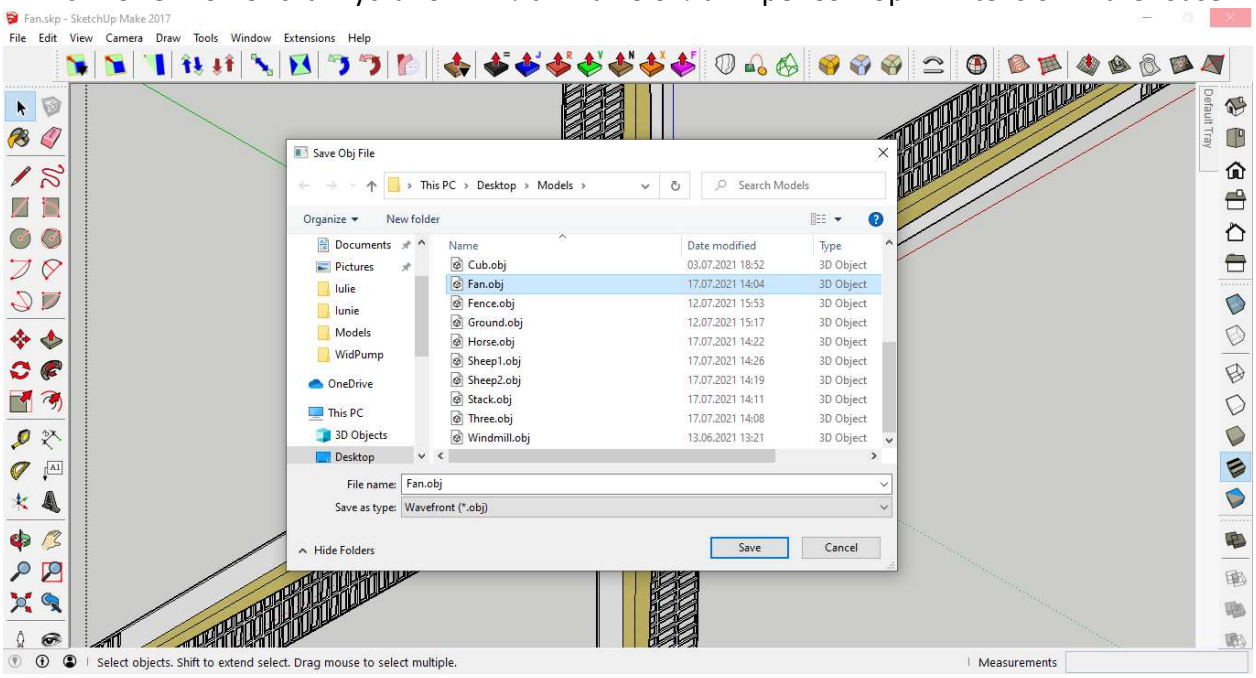


Рис. 4.10. Окно сохранения 3D моделей объектов в формате .obj

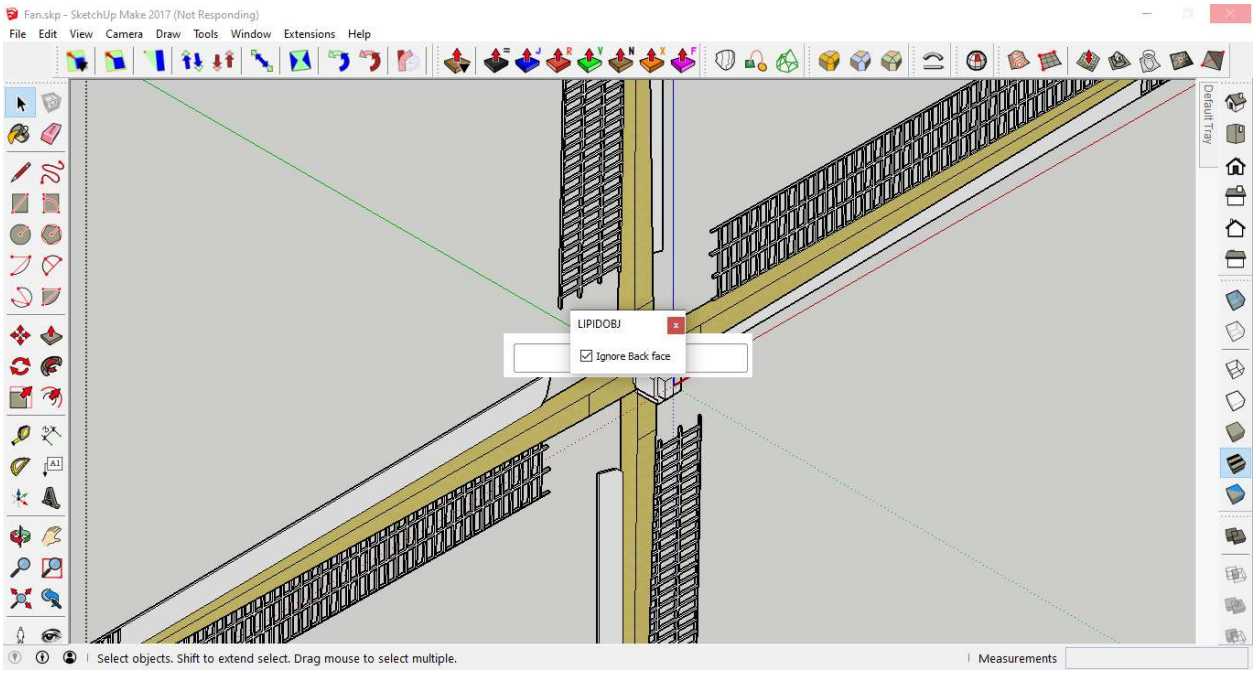


Рис. 4.11. Опция игнорирования невидимых поверхностей плагина LIPIDOBJ при экспорте 3D-моделей в формате .obj

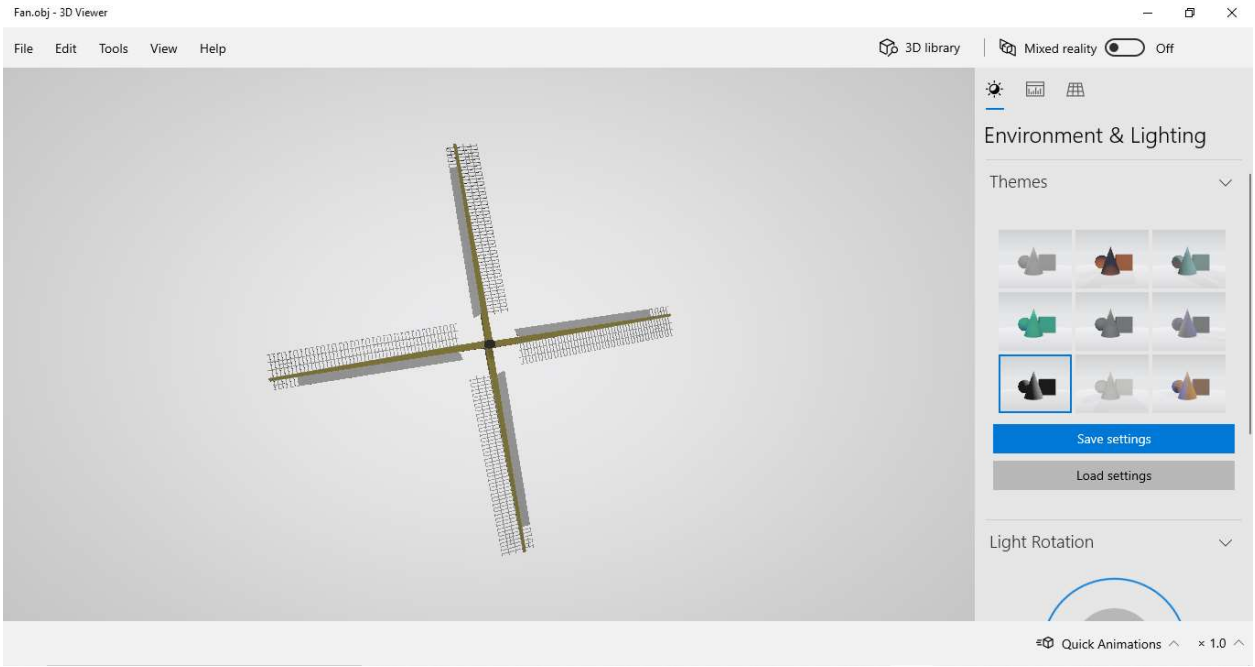


Рис. 4.12. Просмотр содержимого файлов OBJ с помощью стандартного приложения в рамках операционной системы Windows 10 – 3D Viewer

Ниже приведен пример кода библиотеки p5.js с импортированными 3D моделей объектов.

```
let fr = 30;
let obj;

function preload() {

  mill = loadModel('Windmill.obj');
  fan = loadModel('Fan.obj');
  gnd = loadModel('Ground.obj');
  sheep1 = loadModel('Sheep1.obj');
```

```
  sheep2 = loadModel('Sheep2.obj');
  horse = loadModel('Horse.obj');
  stack = loadModel('Stack.obj');
  fence = loadModel('Fence.obj');
  three = loadModel('Three.obj');
}

function setup() {

  createCanvas(400, 400, WEBGL);
  normalMaterial();
  //ambientMaterial(150, 150, 150);
  frameRate(fr);
  angleMode(DEGREES);
}

function draw() {
  orbitControl();
  background(50);
  pointLight(255, 255, 255, 100000, 100000, -100000);
  noStroke();

  push();
  scale(0.005);
  translate(0, 0, 0);
  model(mill);
  pop();

  push();
  scale(0.005);
  translate(0, 3000, 13000);
  rotateX(15);
  model(fan);
  pop();

  push();
  scale(0.005);
  model(gnd);
  pop();

  push();
  scale(0.005);
  translate(-3000, 15000, 0);
  rotateZ(-10);
  model(sheep1);
  pop();

  push();
  scale(0.005);
  translate(10000, 0, 0);
  rotateZ(180);
  model(sheep2);
  pop();

  push();
```

```
  scale(0.005);
  rotateZ(-15);
  translate(7000, 12000, 0);
  model(horse);
  pop();

  push();
  scale(0.005);
  translate(-12000, -2000, 0);
  model(stack);
  pop();

  push();
  scale(0.005);
  translate(0, 0, 0);
  model(fence);
  pop();

  push();
  scale(0.005);
  translate(15000, -15000, 0);
  model(three);
  pop();
}
```

На рис. 4.13 показан ррезультат выполнения программы отображения статической сцены в p5.js.

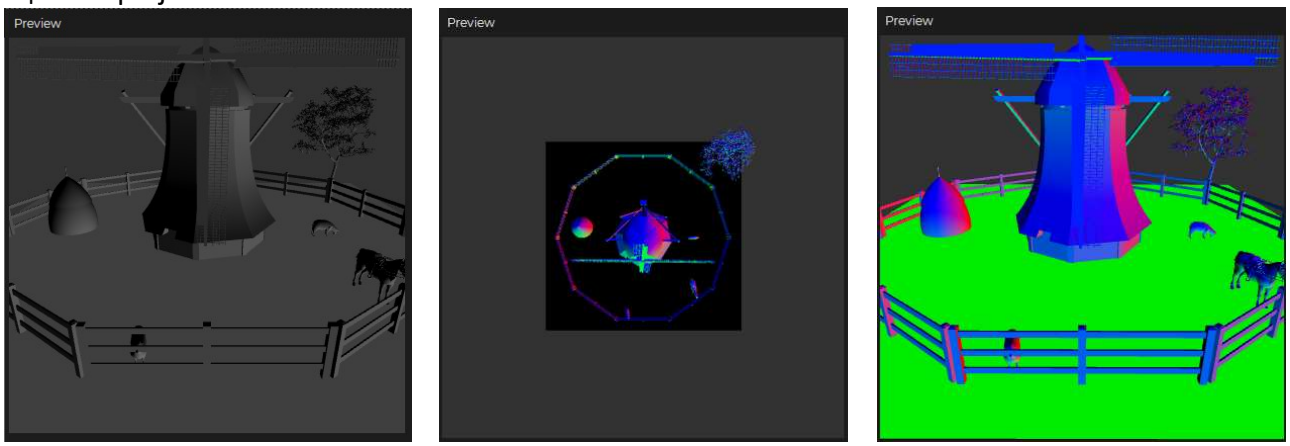


Рис. 4.13. Результат выполнения программы отображения статической сцены

Лабораторной работы № 4
Тема: Статические 3D-сцены

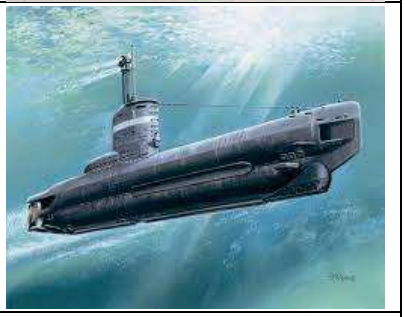
Цель работы: Получить практические знания в области синтеза статических 3D графических сцен, используя стандартные функции представления 3D-моделей из библиотеки p5.js.

Задача работы:

1. Разработать программу для синтеза статической 3D-сцены, которая содержала бы не менее 5 объектов, используя стандартные функции представления 3D-моделей в библиотеке p5.js.

2. Разработайте программу, создающую статическую 3D-сцену по варианту, указанному в таблице 4.1. Для создания сцены можно использовать существующие 3D-графические объекты из репозитории 3D Warehouse.

Таблица 4.1. Варианты проведения лабораторных работ

Вариант	3D объект	Рисунок	Вариант	3D объект	Рисунок
1	Карусель		6	Вертушка	
2	Маяк		7	Ветряная турбина	
3	Броневик		8	Винтовая плоскость	
4	Водяная мельница		9	Подводная лодка	
5	Самолет		10	Требушет