IV. ИСПОЛЬЗОВАНИЕ 3D-ОБЪЕКТОВ В СТАТИЧНЫХ СЦЕНАХ

# 4.1. \*. OBJ

Формат файла OBJ является одним из самых важных форматов файлов в 3D-печати и 3D-графике. Это предпочтительный формат для многоцветной 3D-печати, широко используемый 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 3D-редакторами. Он содержит трехмерные объекты, которые включают в себя 3D-координаты, текстурные карты, многоугольные грани и другую информацию об объекте.

Формат файла OBJ хранит информацию о 3D-моделях, может кодировать геометрию поверхности 3D-модели, а также хранить информацию о цвете и текстуре. В этом формате не хранится никакая информация о сцене, такая как положение света или анимация.

Файл OBJ генерируется программным обеспечением CAD (Computer Aided Design) как готовый продукт процесса 3D-моделирования. Расширение файла, соответствующее формату файла OBJ: – "\*. OBJ".

Формат файла OBJ является нейтральным и имеет открытый исходный код, и часто используется для обмена 3D-моделями в графических приложениях, поскольку он обеспечивает хорошую поддержку импорта и экспорта практически из всех программ САПР. Он также используется в качестве формата файлов для многоцветной 3D-печати, поскольку стандартный формат 3D-печати STL не включает информацию о цвете и текстуре.

Формат файла OBJ был первоначально создан компанией Wavefront Technologies в приложении Advanced Visualizer для хранения геометрических объектов, состоящих из линий, многоугольников, кривых и поверхностей произвольной формы. Последняя документированная версия – v3.0, заменяющая предыдущую версию v2.11.

Самым доминирующим форматом в мире 3D-печати является STL. Однако STL – это старый формат файлов, который хоть и очень популярен, но не учитывает современные требования. Точность 3D-печати приближается к разрешению на микронном уровне, а многоцветные узоры становятся все более популярными. Формат файла STL не поддерживает высокое разрешение, так как более высокое разрешение приводит к увеличению размера файла. Также формат STL не подходит для многоцветной 3D-печати, так как не поддерживает информацию о цвете и текстуре. В отличие от этого, формат OBJ может довольно точно аппроксимировать геометрию поверхности, не оказывая существенного влияния на размер файла. Это возможно с помощью кривых Безье и метода, называемого NURBS. Кроме того, формат OBJ имеет встроенную поддержку нескольких цветов и текстур в одном шаблоне.

Таким образом, использование формата OBJ имеет ряд преимуществ по сравнению с использованием формата STL, где необходимы многоцветные модели с высоким разрешением. С другой стороны, формат файла OBJ не так универсален, как формат STL. Практически все 3D-принтеры поддерживают формат STL, чего не скажешь о формате OBJ, хотя он и пользуется поддержкой. Поэтому, если есть необходимость напечатать монохромную 3D-модель на стандартном принтере, формат STL предпочтительнее.

# 4.2. Описание основных объектов и методов в работе с \*. OBJ в онлайн редакторе p5.js

p5.js Library – это JavaScript библиотека, которая облегчает процесс разработки программ с использованием 2D и 3D графики и предназначена как для опытных, так и для начинающих разработчиков. Библиотека p5.js бесплатна и имеет открытый исходный код.

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

**Основные функции:**

*Загрузка 3D-модели из файла OBJ или STL*

**Функция loadModel():** она должна быть размещена внутри функции **preload(),** это позволяет модели полностью загрузиться перед прохождением программы до конца.

Одним из ограничений формата OBJ и STL является то, что он не учитывает масштаб. Это означает, что модели, экспортируемые из разных программ, могут иметь разные размеры. Если модель не отображается, можно вызвать **loadModel()** с параметром нормализации, установленным в true. Это приведет к изменению размера модели до масштаба, подходящего для p5. Дополнительные изменения размера модели также могут быть внесены с помощью **функции** scale().

Кроме того, цветные файлы STL не поддерживаются. Цветные STL-файлы будут отображаться без свойств цвета.

**Синтаксис:**

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

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

где: path String – путь к загруженной модели;

normalize boolean – если это правда – масштабирует модель до стандартного размера при загрузке;

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

failureCallback Function(Event) – вызывается с событием ошибки, если модель не загружается (опционально);

fileType String – расширение файла модели (.stl, .obj) (опционально). Возвращает p5.Geometry – объект типа p5.Geometry.

**Функция Model()** – воспроизводит 3D модель на экране;

**Синтаксис:**

**model(model);**

где: Модель – p5. Геометрия.

Загруженная 3D модель для воспроизведения:

**Sketch Files->Upload file**

# 4.3. Создание 3D статической сцены

Задачей лабораторной работы является проектирование и построение 3D статической сцены с помощью онлайн-редактора p5.js, используя 3D модели объектов, хранящиеся в .OBJ файле, созданного самостоятельно или скаченного из Интернета.

Для иллюстрации выполнения лабораторной работы требуется создать статичную сцену, которая представляла бы собой ветряную мельницу, огороженную забором, во дворе которой будут находиться животные, дерево и стог сена. 3D-модели этих объектов можно создавать или импортировать из хранилища 3D Warehouse, доступ к которому можно получить из основного меню графического редактора Google SketchUp ***Window->3D Warehouse 3D***, а затем редактировать по мере необходимости, как показано на рисунке 4.1.

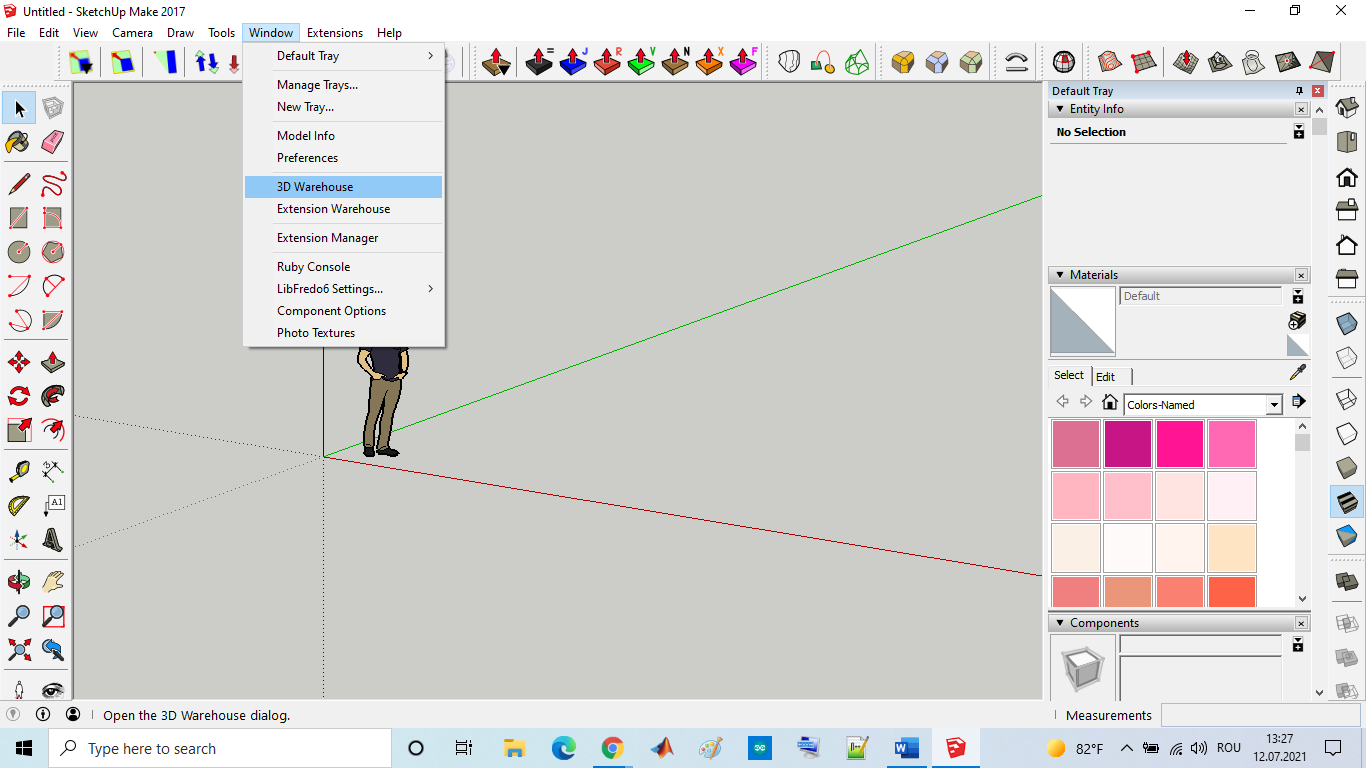


Рисунок 4.1. Рабочее окно хранилища 3D Warehouse

Чтобы импортировать выбранную модель в текущий эскиз, пользователь должен иметь учетную запись Google и войти в хранилище 3D Warehouse. После загрузки необходимых моделей их можно отредактировать по мере необходимости и экспортировать как .OBJ с помощью плагина ***LIPID OBJ Exporter***, который можно установить из меню *Window->Extension Warehouse*. На рисунке 4.2, в поле поиска которого мы указываем ключевое слово OBJ, в ответ отображаются плагины, удовлетворяющие критериям поиска, среди которых можно увидеть искомый плагин *LIPID OBJ Exporter*, как показано на рисунке 4.3.

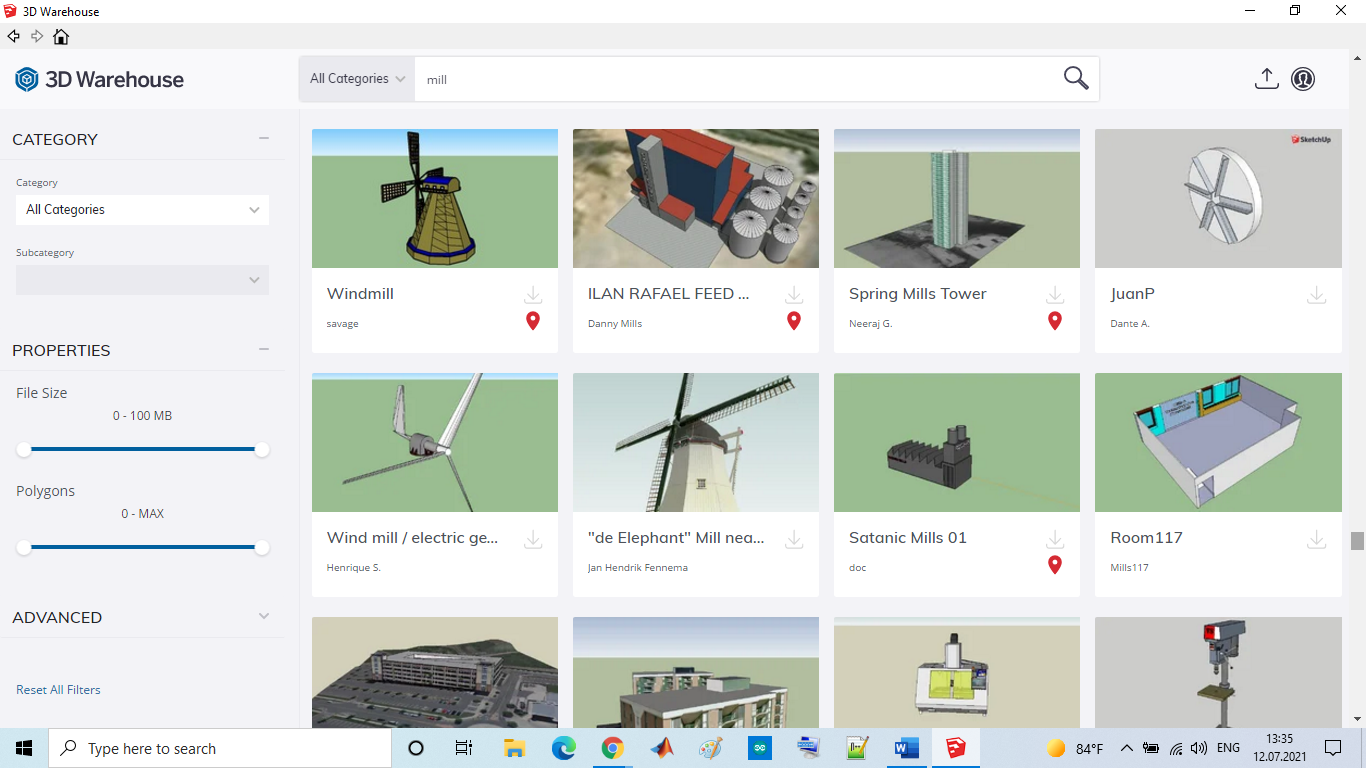


Рисунок 4.2. Поиск 3D-моделей в хранилище 3D Warehouse

Чтобы установить плагины, пользователь должен иметь учетную запись Google и войти в онлайн- хранилище *Extension Warehouse*.

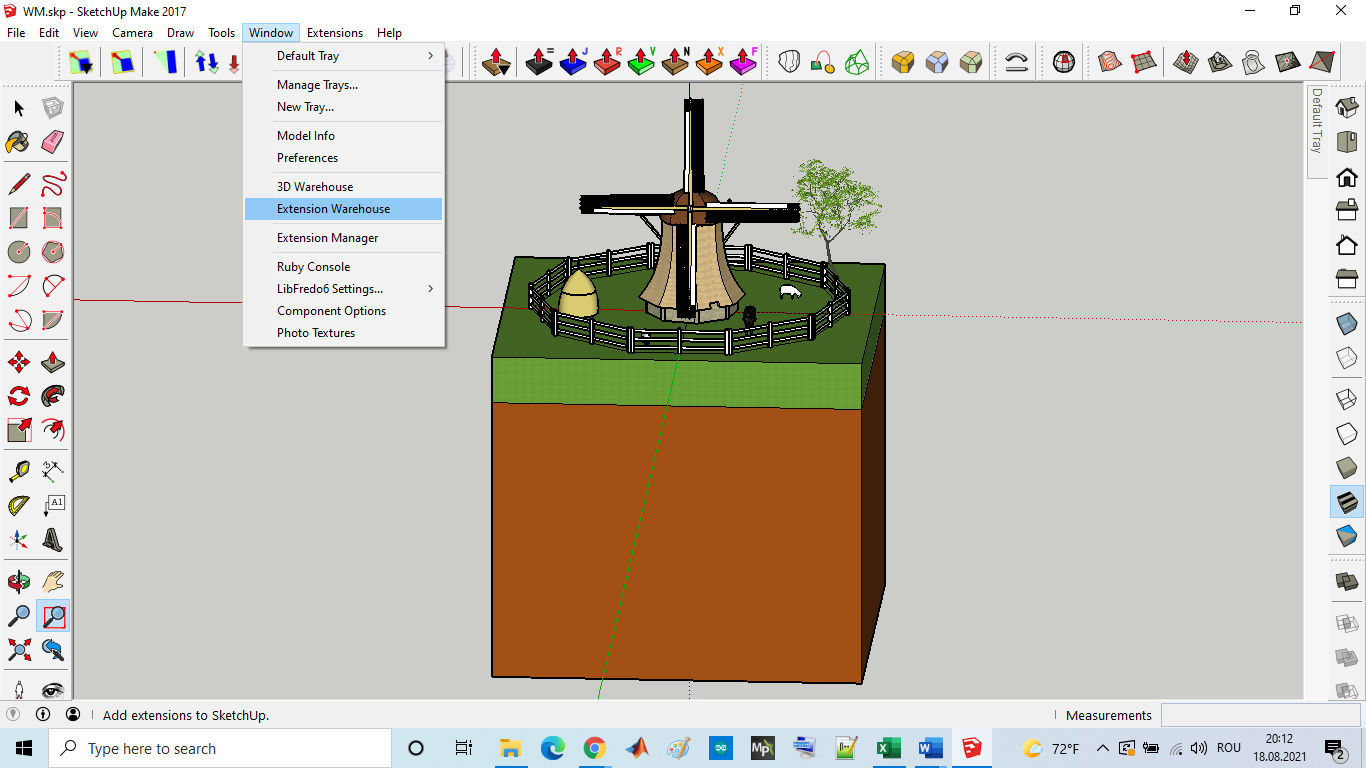


Рисунок 4.3. Доступ к опции «Extension Warehouse» в строке меню

На рисунках ниже показаны шаги, которые необходимо выполнить для создания модели \*.obj.

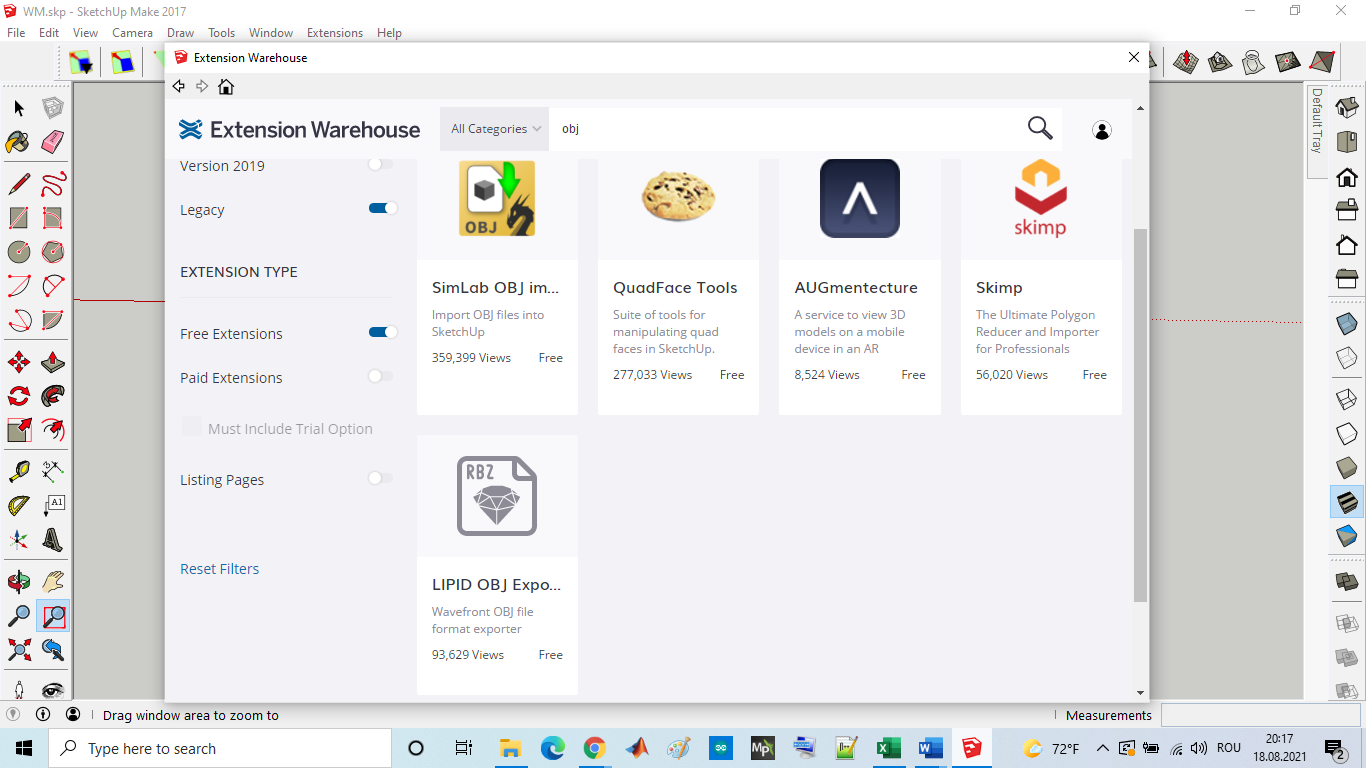


Рисунок 4.4. Окно поиска и установки плагинов из онлайн- хранилища Extension Warehouse

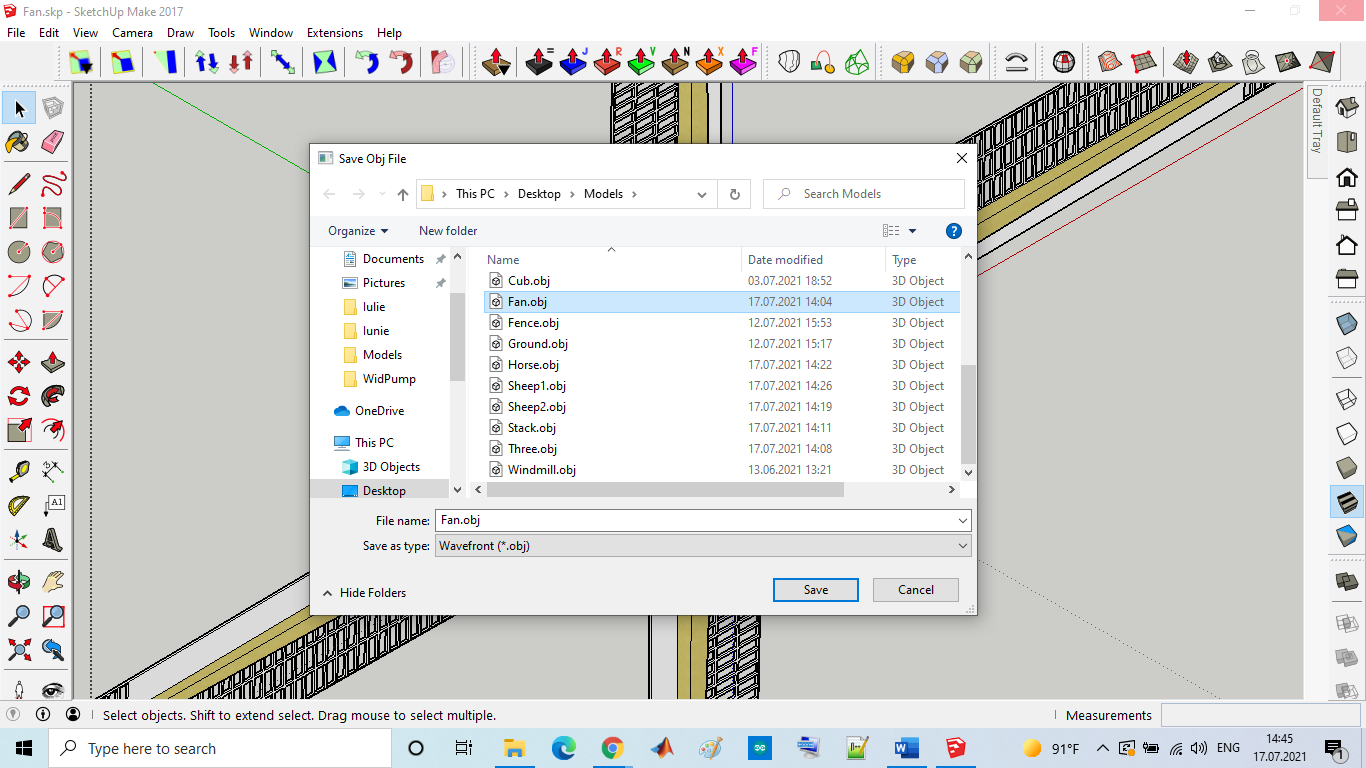


Рисунок 4.5. Окно для сохранения 3D моделей объектов в сцене в формате .obj

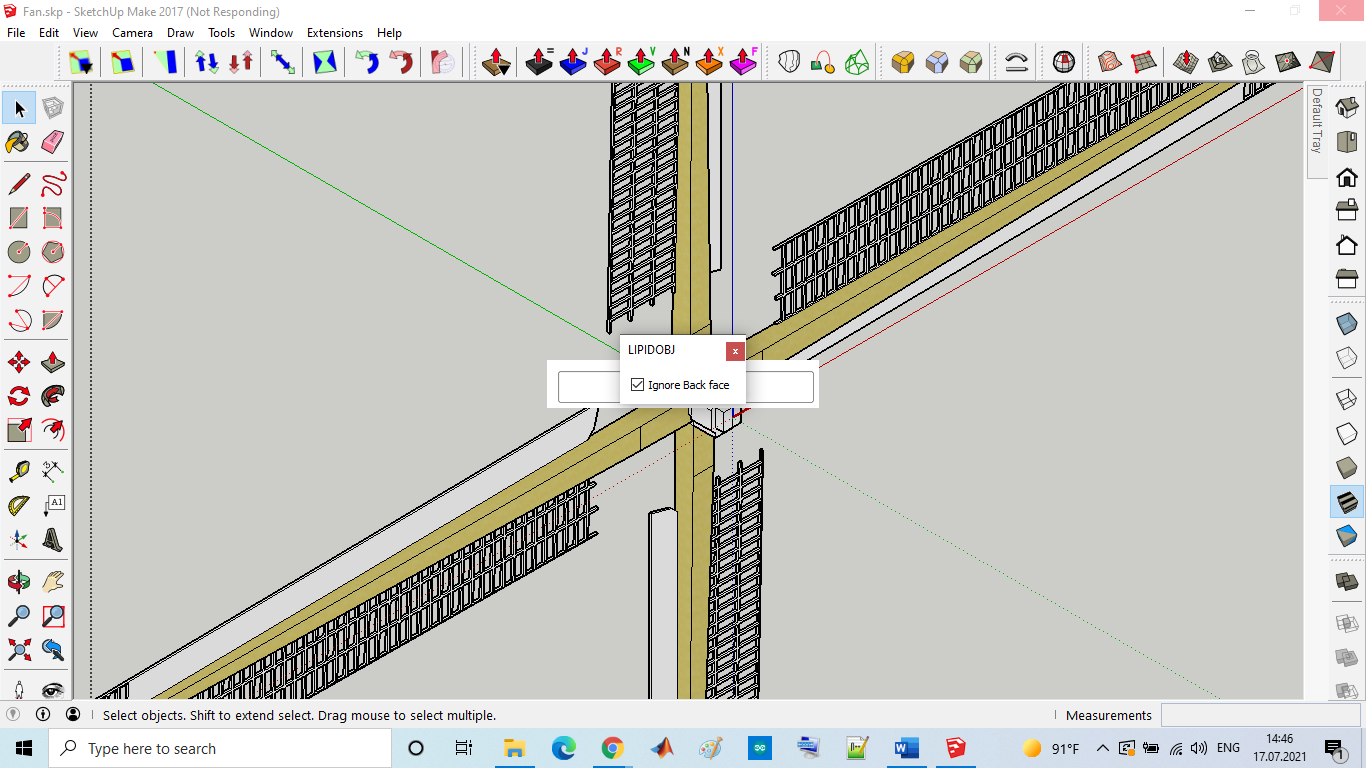


Рисунок 4.6. Игнорирование задних поверхностей плагина LIPIDOBJ при экспорте 3D моделей в формат .obj

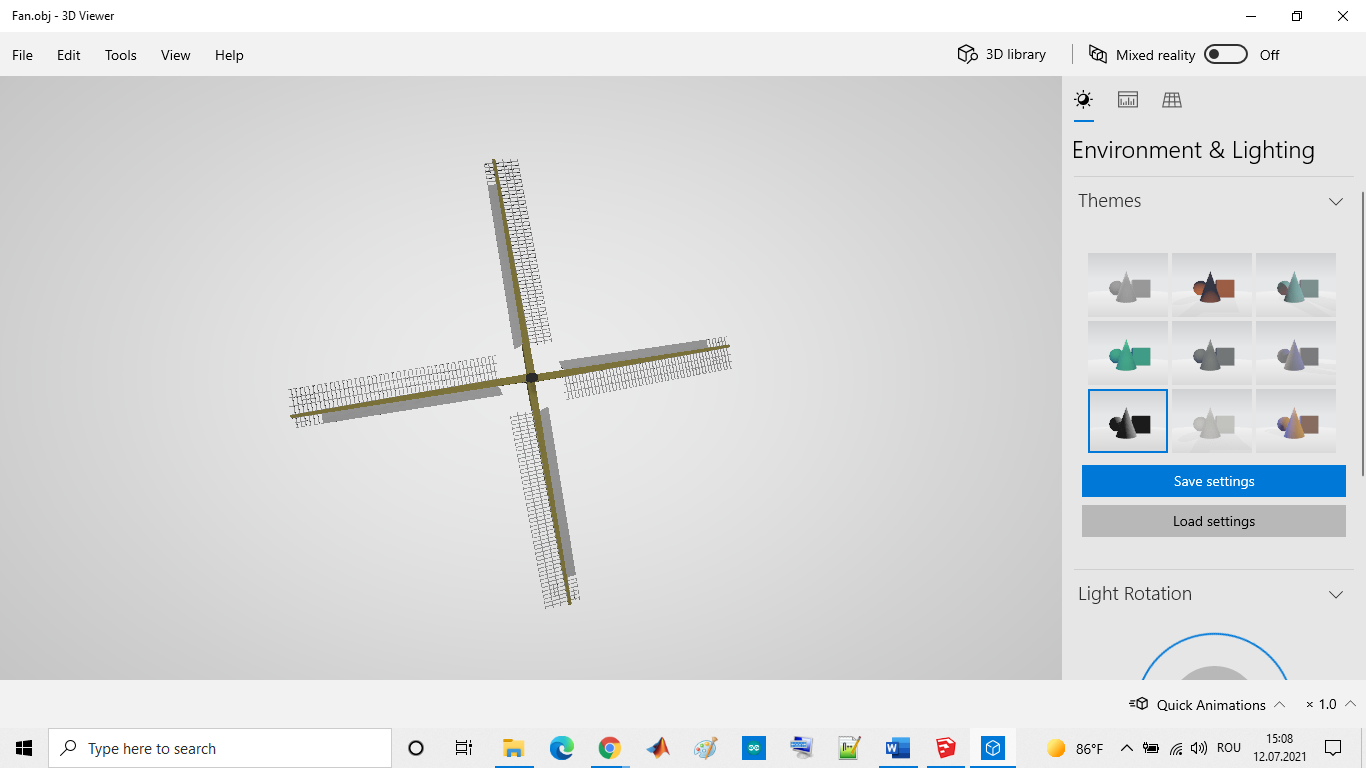


Рисунок 4.7. Ознакомление с содержимым файла .OBJ с помощью стандартного приложения Windows 10 – 3D Viewer

### Лабораторная работа No 4

### Тема: СОЗДАНИЕ 3D СТАТИЧНОЙ СЦЕНЫ

**Задачи работы:**

**1. Создание статичной 3D модель** – используйте графический редактор, совместимый с экспортом в .obj формат (например, Blender, SketchUp или Warehouse).

**2. Правильный экспорт модели** – в .obj формат и импорт ее в p5.js среду программирования.

**3. Добавление 3D графических примитивов** – в p5.js для доработки и настройки 3D сцены.

**4. Исследование функций p5.js для работы с 3D-объектами** путем добавления цветовых и световых атрибутов.

**5. Применение концепций компьютерной графики** – позиционирование и структурирование трехмерной сцены.

**6. Создание финальной сцены**, которая сочетает в себе импортированную 3D-модель и графику, добавленную непосредственно в p5.js для создания целостного визуального эффекта.

**Количество часов, необходимое для выполнения – 4 академических часа.**

**Цель работы:** получить практические знания по синтезу статичных 3D графических сцен, используя стандартные функции представления 3D моделей в библиотеке p5.js. Развить навыки моделирования и визуализации 3D объектов за счет использования графических редакторов и интеграции моделей в формате \*.obj в интерактивные среды на базе JavaScript.

**Задача работы:**

1. Разработать объект с расширением **\*.obj** в котором создать статичную 3D сцену по варианту, указанному в таблице 4.1. Для создания сцены можно использовать 3D-графику из хранилища 3D Warehouse.

2. Разработайте программу, в которой вы синтезируете статичную 3D сцену, которая бы содержала не менее 5 объектов, используя стандартные функции представления 3D модели в библиотеке p5.js, в созданной сцене импортируйте объект \*.obj созданный на шаге 1 задания.

Таблица 4.1. Варианты для лабораторной работы

| № | 3D-объект | Рисунок | № | 3D-объект | Рисунок |
| --- | --- | --- | --- | --- | --- |
| 1 | Карусель |  | 2 | Маяк |  |
| 3 | Броневик |  | 4 | Водяная мельница |  |
| 5 | Винтовой самолет |  | 6 | Ветряк |  |
| 7 | Ветряная турбина |  | 8 | Вертолёт |  |
| 9 | Подводная лодка |  | 10 | Требушет |  |
| 11 | Автомобиль |  | 12 | Моторная лодка |  |
| 13 | Детская горка |  | 14 | Стол со стульями |  |
| 15 | Декоративный мост |  | 16 | Детский велосипед |  |

**Пример:**

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();

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(); }

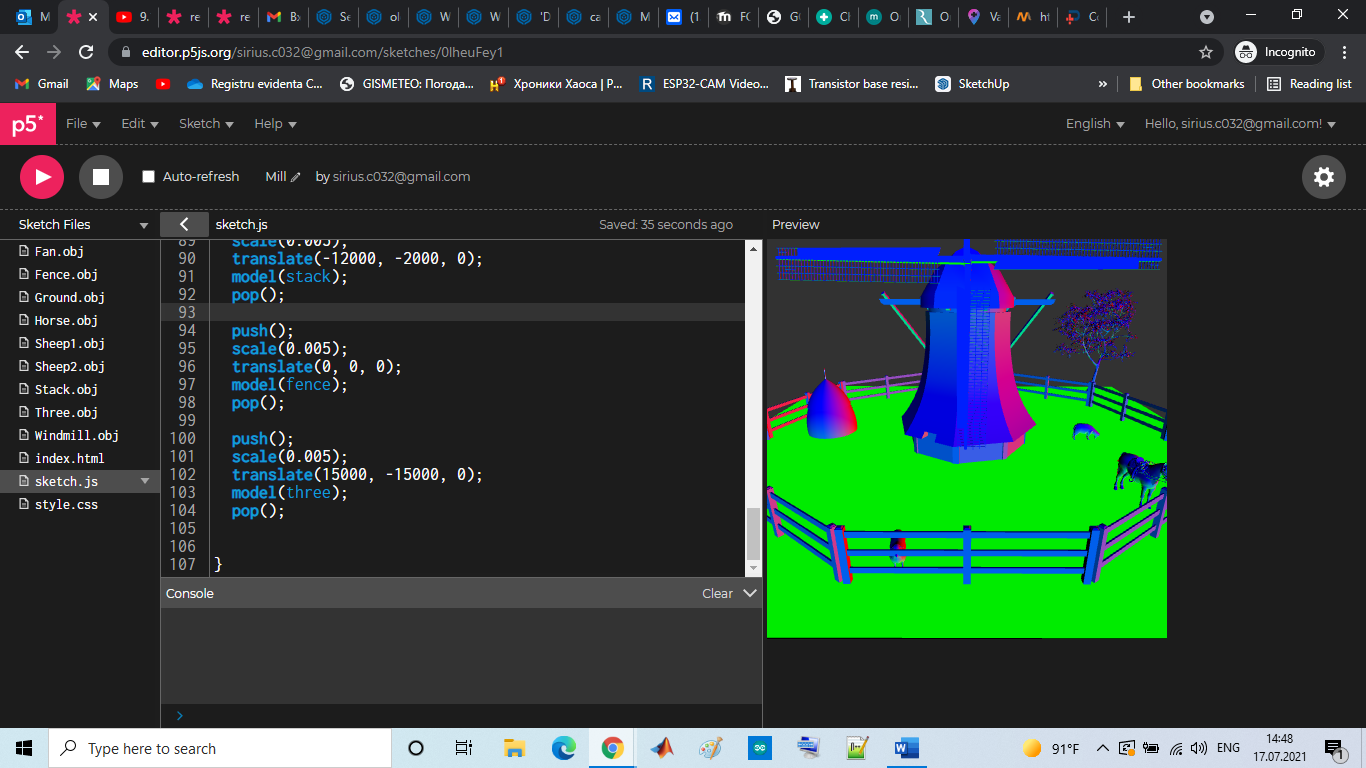


Рисунок 4.8. Результат выполнения программы

**Критерии оценки**

**1. Правильность 3D моделирования в графическом редакторе** (20%) – если созданный 3D объект соответствует требованиям и хорошо проработан, если экспорт в формат .obj выполнен правильно и без потери данных.

**2. Интеграция 3D модели в p5.js** (25%) – если импортируемая модель функционирует в p5.js без ошибок отображения, если позиционирование модели в 3D сцене корректно и уместно.

**3. Добавление и использование графических примитивов в p5.js** (25%) – если в сцену были добавлены 3D-графические примитивы, которые вносят свой вклад в общую композицию, если примитивам были назначены соответствующие места размещения и размеры для создания целостной сцены.

**4. Соответствие требованиям и ясность кода** (10%) – если код четкий, хорошо организованный и документированный, соответствующий заданным техническим требованиям. Пояснительные комментарии включаются там, где это необходимо для уточнения функциональности. Будь то эстетичная и хорошо структурированная сцена, со сбалансированной композицией или основные графические концепции (освещение, тени, перспектива) были применены для приятного визуального впечатления.

**5. Оценка знаний** (10%) – пояснения к процессу выполнения работы, которые могут включать в себя описание основных функций и используемой логики.

**6. Соблюдение требований и срок финальной презентации проекта** – (10%).

**Вопросы для самостоятельной проверки знаний**

1. Что такое статичная 3D модель и чем она отличается от анимированной?

2. Объясните шаги, необходимые для экспорта 3D-модели в формат .obj и импорта ее в проект p5.js.

3. С какими примитивами 3D графики p5.js вы знакомы? Перечислите несколько примеров и возможных применений.

4. Как вращать, переносить и масштабировать 3D объект в p5.js?

5. В чем преимущества использования p5.js для рендеринга статичных 3D-сцен?

6. Как можно настроить освещение в 3D-сцене p5.js чтобы добиться реалистичного эффекта?

7. Какие функции и методы вы использовали для добавления 3D-графики в p5.js и как они повлияли на внешний вид финальной сцены?