5. Преобразования 3D
5.1. Система координат

В геометрии система координат — это способ, при котором любая точка однозначно связана с упорядоченным набором вещественных чисел, называемых координатами этой точки. В евклидовом пространстве требуются три координаты (абсцисса, ордината и аппликата), две (абсцисса и ордината) необходимы на плоскости, и только одна координата требуется для расположения точек на линии. В аналитической геометрии использование систем координат позволяет преобразовывать геометрические задачи в задачи алгебры.

Используются однородные координаты, как и в случае 2D.

В математике однородные координаты, введенные Августом Фердинандом Мёбиусом, позволяют выполнять преобразования, представляя их в виде матрицы. Однородные координаты также позволяют выполнять вычисления в проективных пространствах аналогично тому, как декартовы координаты делают это в евклидовом пространстве.

Однородные координаты точки в проективном пространстве размерности n обычно записываются как ($x: y: z: ...: w$), линейный вектор длины $n + 1$, отличный от (0: 0: 0: .: 0). Два набора координат, которые пропорциональны, обозначают одну и ту же точку в проективном пространстве: для любого ненулевого *скалярного c* в поле, лежащем в основе K, ($cx : cy : cz : ... : cw$) представляет ту же точку. Поэтому эту систему координат можно объяснить следующим образом: если проективное пространство построено из векторного пространства V размерности $n + 1$, то координаты в V вводятся, выбрав основание, и используя их в $P(V)$, пропорциональные классы эквивалентности ненулевых векторов в V.

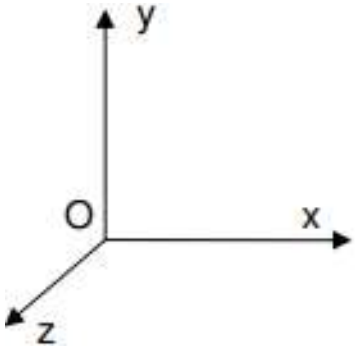

Рис. 5.1. Левосторонняя система координат

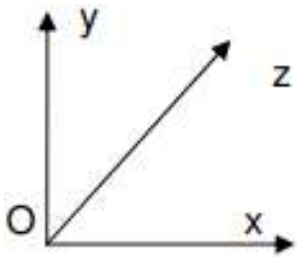

Рис. 5.2. Правосторонняя система координат

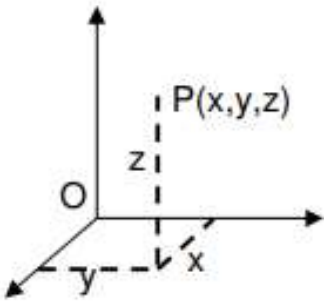

Рис. 5.3. Декартова система координат

Преобразования будут представлены в виде матриц 4x4.

Используется правосторонняя система координат.

Точка. Точка P в трехмерном пространстве может быть определена следующим образом:

- Декартовы координаты: $P(x, y, z)$;
- Однородные координаты: $P(x, y, z, u)$.

5.2. Основные 3D-преобразования

Перенос

$$T(t_x, t_y, t_z) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_y \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Вращение вокруг оси

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Масштабирование

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

5.3. Составные 3D-преобразования

Составление преобразований

Обобщенная матрица

Структура для случая векторно-колоночной нотации:

- матрица 3x3 включает в себя такие преобразования как: локальное масштабирование, наклон (сдвиг – shear), зеркальное отображение и вращение,
- матрица 3x1 представляет собой преобразование переноса,
- матрица 1x3 представляет собой преобразование перспективного проектирования,
- матрица 1x1 представляет собой преобразование общего масштабирования,

$$\begin{bmatrix} & & & \vdots & 3 \\ & 3 \times 3 & & \vdots & \times \\ & & & \vdots & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 3 & \times & 1 & \vdots & 1 \times 1 \end{bmatrix}.$$

Матрица, соответствующая составному преобразованию, получается путем умножения матриц элементарных преобразований. Поскольку умножение матриц не является коммутативным, важен порядок, в котором эти преобразования применяются.

Матрица преобразования, ближайшая к вектору линии (или вектору столбца), соответствует первому примененному преобразованию, в то время как самая дальняя матрица преобразования является последней из примененных.

Математически это выражается следующим образом:

[*M*] [*VC*] = [*Mn*] ..... [*M*3] [*M*2] [*M*1].... [*VC*], для вектора столбца,

или *[VL] [M] =[VL] [M1]^T [M2]^T [M3]^T ... [Mn]^T*, для вектора линии,

где [*Mi*] может быть любая матрица преобразования.

Цель работы: Создание динамической 3D-сцены.

Задача лабораторных работ заключается в задумке и построении динамической 3D-сцены, которая содержала бы преобразования: перенос, вращение и масштабирование. Выполнение сцены с помощью онлайн-редактора p5.js с использованием 3D моделей, хранящихся в файлах .OBJ, которые могут быть созданы индивидуально или загружены из Интернета.

Ход работы:

Как пример дальше показан ход лабораторной работы № 5, где предлагается создать динамическую сцену, которая представляла бы водяной насос, ориентирующий свой пропеллер в зависимости от направления ветра. Модель насоса разделена на три части: основание (Base), пропеллер (Fan) и ориентир (Tail). Каждый компонент помещается в отдельный файл .skp с расположением модели таким образом, чтобы точки привязки (начало системы координат каждой части) и ориентация осей совпадали с началом и ориентацией общей системы координат сцены. 3D-компоненты модели могут быть созданы с нуля или модель может быть импортирована из репозитория 3D Warehouse, доступного в базовом меню графического редактора Google SketchUp Window-> 3D Warehouse, а затем отредактированы соответствующим образом по мере необходимости.

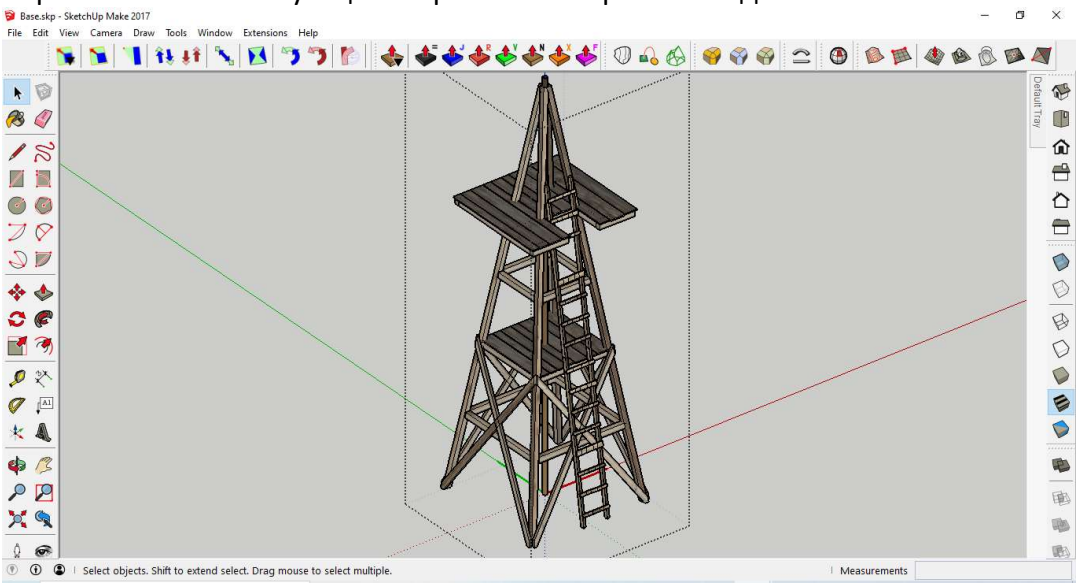

Рис. 5.4. Модель основания водяного насоса (Base)

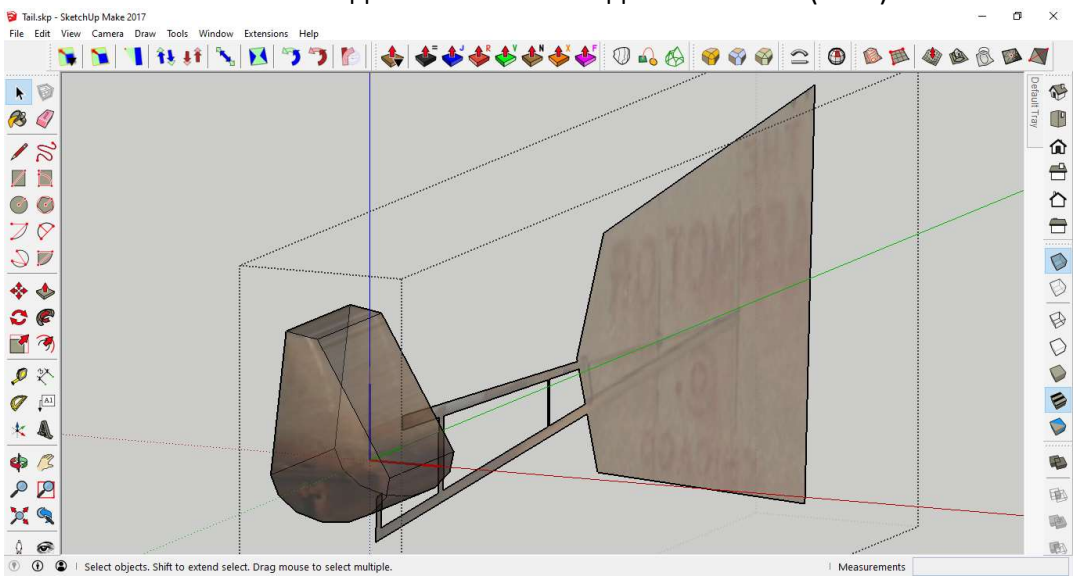

Рис. 5.5. Модель ориентира водяного насоса (Tail)

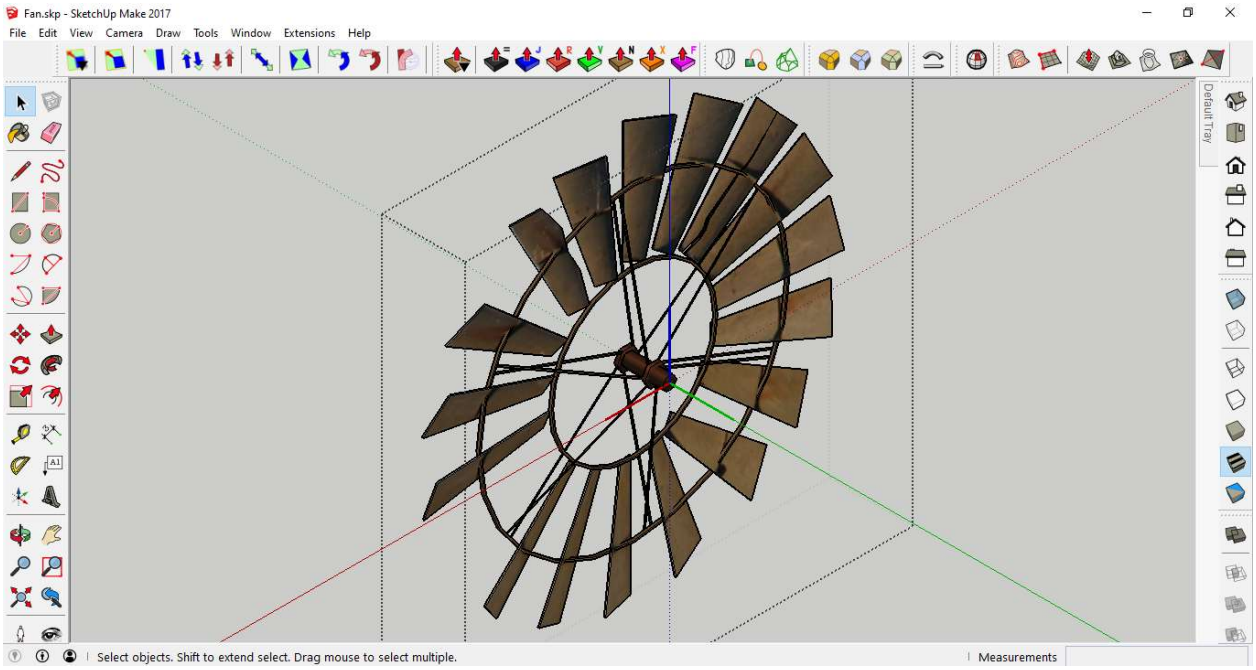

Рис. 5.6. Модель пропеллера водяного насоса (Fan)

Для того, чтобы иметь возможность скачать или импортировать выбранную модель в текущем эскизе, пользователь должен иметь учетную запись Google и войти в среду 3D Warehouse. После того, как каждый компонент модели будет правильно загружен, их можно редактировать по мере необходимости и экспортировать в виде файлов. OBJ с помощью плагина *LIPID OBJ Exporter*, способ установки которого был представлен в лабораторной работе № 4.

Описание основных функций в библиотеке p5.js

preload()

Вызываемая непосредственно перед ***функцией setup(),*** функция ***preload()*** используется для управления асинхронной загрузки внешних файлов. Если функция предварительной загрузки определена, функция ***setup()*** будет ждать завершения любого вызова загрузки. Ничего, кроме вызовов загрузки (*loadImage, loadJSON, loadFont, loadStrings* и т.д.), не должно находиться внутри функции предварительной загрузки. Если предпочтительна асинхронная загрузка, методы загрузки можно вызвать в программе setup () или в любом другом месте с помощью параметра обратного вызова. Ничего другого не должно быть сделано внутри функции ***preload(),*** кроме вызовов загрузок, все остальные инициализации должны быть сделаны внутри функции ***setup()***.

Синтаксис:
preload()

loadModel()

Загружает 3D-модель из файла OBJ или STL. Функция ***loadModel()*** должна быть помещена 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 имеют встроенного масштабирования. Это означает, что модели, экспортируемые из разных графических 3D-редакторов, могут иметь разные размеры. Если модель не отображается, можно вызвать

метод ***loadModel()*** с значением ”true” параметра нормализации. Это изменит размер модели до масштаба, подходящего для p5. Дальнейшие изменения конечного размера модели могут быть сделаны с помощью функции ***scale()***. Еще одним недостатком является то, что библиотека p5.js не обеспечивает поддержку цветных STL и OBJ-файлов, в которых содержится информация об: материалах, текстурах и цветах. 3D модели в p5 будут отображаться без свойств материала, текстур и цветов.

Синтаксис:
loadModel(path, normalize, [successCallback], [failureCallback], [fileType])
loadModel(path, [successCallback], [failureCallback], [fileType])
Параметры:

Path	String: путь к загруженной модели
normalize	boolean: если значение true, масштабируется модель до стандартизированного размера при загрузке
функция successCallback	(p5. Geometry): функция, которая должна быть вызвана после загрузки модели. Будет передан объект 3D-модели. (опционально)
failureCallback	(Event): вызываемая с ошибкой события, если модель не загружается. (опционально)
fileType	String: расширение файла модели (.stl, .obj). (опционально)

setup()

Функция ***setup()*** вызывается лишь однажды после запуска программы. Он используется для определения свойств исходной среды, таких как размер экрана, цвет фона, а также для загрузки мультимедиа, такого как изображения и шрифты, во время выполнения программы. В каждой программе может быть только одна функция ***setup(),*** и она не должна вызываться повторно после ее первоначального выполнения.
Примечание: Переменные, объявленные в функции ***setup(),*** недоступны в других функциях, включая функцию ***draw(***).
Синтаксис:
setup()

creatCanvas()

Создает элемент canvas (полотно) в HTML-документе и задает его размеры в пикселях. Этот метод следует вызывать только один раз в начале начального параметра. Вызов метода createCanvas() несколько раз в проекте приведет к непредсказуемому поведению. Для создания нескольких холстов рисунка можно использовать функцию createGraphics (по умолчанию эта функция скрыта, но может и отображаться).

В режиме 2D (т.е. когда p5.Renderer не настроен) начало координат (0.0) расположено в левом верхнем углу экрана. В режиме 3D (т.е. когда p5.Renderer установлен как WEBGL), начало координат расположено в центре холста.

Системные переменные width и height задаются параметрами функции creatCanvas(). Если createCanvas() не используется, окно получит размер по умолчанию 100x100 пикселей.
Синтаксис:
createCanvas (w, h, [renderer])
Параметры:

w	Значение: ширина холста
h	Значение: высота холста
renderer:	P2D или WEBGL (опционально)

normalMaterial()

normalMaterial (нормальный материал) для геометрии – это материал, на который не влияет свет. Он не является отражающим и является замещающим материалом, часто используемым для отображения на экране. Поверхности, ориентированные по оси X, становятся красными, поверхности, ориентированные по оси Y, становятся зелеными, а ориентированные по оси Z — синими.

Синтаксис:

normalMaterial()

frameRate()

Задает количество кадров, отображаемых каждую секунду. Например, вызов функции ***frameRate(30)*** попытается перерисовать сцену 30 раз в секунду. Если процессор недостаточно оптимизирован для поддержания заданного параметра, частота кадров не сможет быть обеспечена. Рекомендуется устанавливать частоту кадров внутри функции ***setup(***). Частота кадров по умолчанию основана на частоте кадров дисплея (также называемой «частотой обновления»), которая на большинстве компьютеров установлена на уровне 60 кадров в секунду. Для плавной анимации будет достаточно частоты 24 кадра в секунду (обычно для фильмов) или выше. Эта функция аналогична функции ***setFrameRate (значение)***.

Вызов функции frameRate() без аргументов возвращает текущую частоту кадров. Функция ***draw()*** должна выполняться хотя бы один раз, прежде чем функция ***frameRate()*** вернет значение. Эта функция аналогична функции ***getFrameRate()***. Вызов ***функции frameRate()*** с аргументами, которые не являются числовыми или не являются положительными, также возвращает текущую частоту кадров.

Синтаксис:

frameRate(fps)

frameRate()

Параметры:

fps: количество кадров, отображаемых каждую секунду

angleMode()

Задает в p5 текущий режим представления единицы измерения углов в радианах или градусах. Режим представления единицы измерения угла по умолчанию — в радианах.

Синтаксис:

angleMode (mode)

Параметры:

mode constant: RADIANS, или ГРАДУСЫ

draw()

Вызываемая непосредственно после ***setup()***, функция ***draw()*** непрерывно выполняет строки кода, содержащиеся в ее блоке, пока программа не будет остановлена или не будет вызван ***noLoop()***. Если внутри функции setup() вызывается функция ***noLoop(),*** функция ***draw()*** будет выполнена еще раз перед остановкой. Функция ***draw()*** вызывается автоматически и не нуждается в вызове по умолчанию. Вызов функции ***draw()*** всегда должен управляться с помощью ***noLoop()***, ***redraw()*** и ***loop()***. После того, как ***noLoop()*** перестанет выполнять код в ***draw()***, ***redraw()*** заставит код в ***draw()*** выполниться только один раз, и вызов функции ***loop()*** возобновит циклическое выполнение в качестве кода в функции ***draw().*** Количеством выполнений функции ***draw()*** каждую секунду можно управлять с помощью функции ***frameRate()***. Для каждой сцены в каждом проекте может быть только одна функция ***draw()***.

Функция ***draw()*** должна существовать, чтобы код выполнялся непрерывно или обрабатывал события, такие как ***mousePressed()***.

Следует уточнить, что все преобразования, выполненные по системе координат проекта, будут сбрасываться в начале каждого вызова функции ***draw()***. Если преобразования выполняются внутри функции ***draw()*** (например, масштабирование, поворот, перенос), их эффекты будут отменены в начале функции ***draw(),*** поэтому преобразования не будут накапливаться с течением времени. С другой стороны, примененный стиль (например, заливка, линия и т. д.) останется неизменым.

Синтаксис:

draw()

orbitControl()

Данная функция позволяет перемещаться по 3D-сцене с помощью мыши или трекпада. Щелчок левой кнопкой и перемещение мыши будут поворачивать положение камеры вокруг центра сцены, щелчок правой кнопкой и перемещение мыши будут перемещать положение камеры без поворота, а использование колеса мыши (прокрутка) переместит камеру ближе или дальше от центра сцены. Эта функция может быть вызвана с параметрами, определяющими чувствительность к движению мыши по осям X и Y. Вызов этой функции без параметров эквивалентен вызову функции ***orbitControl(1, 1).*** Для того чтобы изменить направление движения по обеим осям, необходимо ввести отрицательное значение чувствительности.

Синтаксис:

orbitControl([sensitivityX], [sensitivityY], [sensitivityZ])

Параметры:

sensitivityX	Значение: чувствительность к движению мыши по оси X (опционально)
sensitivityY:	Значение: чувствительность к движению мыши по оси Y (опционально)
sensitivityZ (опционально)	Значение: чувствительность к движению прокрутки вдоль оси Z

background()

Функция ***background()*** задает цвет, используемый для фона сцены, в p5.js. Фон по умолчанию прозрачный. Эта функция обычно используется внутри функции ***draw()*** для удаления окна отображения каждого кадра, но она также может быть использована внутри функции ***setup()*** для установки фона первого кадра анимации или если фон необходимо установить только один раз.

Цвет указывается в одном из форматов RGB, HSB или HSL, в зависимости от текущего цветового режима. По умолчанию используется формат RGB, каждое значение которого находится в диапазоне от 0 до 255. По умолчанию *альфа-диапазон* находится в интервале от 0 до 255.

Если указан только один аргумент, поддерживаются цветовые форматы RGB, RGBA и Hex CSS, а также все именованные цветовые форматы. В этом случае значение *параметра alpha* не принимается в качестве второго аргумента, необходимо использовать форму RGBA.

Цветной объект p5. также можно использовать для установки цвета фона. В p5.js также можно использовать существующее изображение для установки цвета фона.

Синтаксис:

background(color)
background(colorstring, [a])
background(gray, [a])
background(v1, v2, v3, [a])

background(values)
background(image, [a])

Параметры:

color	p5.Color: любое значение, созданное функцией color()
colorstring	String: цветовая строка, возможные форматы включают: целочисленный rgb() или rgba(), процент rgb() или rgba(), 3-значный шестнадцатеричный, 6-значный шестнадцатеричный
a	Значение: непрозрачность фона по отношению к текущей цветовой гамме (по умолчанию 0-255) (опционально)
gray	Значение: задает значение оттенка между черным и белым
v1	Значение: красный или оттенок (в зависимости от текущего цветовой модели)
v2	Значение: зеленый или насыщенность (в зависимости от текущего цветовой модели)
v3	Значение: синий или значение яркости (в зависимости от текущего цветовой модели)
values	Number[]: массив, содержащий красный, зеленый, синий и альфа-компоненты цвета
image	p5.Image: изображение, созданное с помощью loadImage() или createImage() для установки в качестве фона (изображение должно быть того же размера, что и сцена)

nostroke()

Отключает рисование обводки. Если вызываются функции ***noStroke()*** и ***noFill(),*** на экране ничего не отображается.

Синтаксис:

noStroke ()

push() и pop()

Функция ***push()*** сохраняет текущие настройки и преобразования стиля рисования, а ***pop()*** восстанавливает эти настройки. Следует отметить, что эти функции всегда используются вместе. Эти функции позволяют изменять стиль и преобразования для повторного использования в предыдущих настройках. Когда новое состояние запускается с ***push()***, оно основано на текущем стиле и преобразует информацию. Функции ***push()*** и ***pop()*** могут быть включены для обеспечения большего контроля.

Функция ***push()*** хранит информацию, связанную с текущими преобразованиями и настройками стиля, управляемыми следующими функциями: ***fill(), noFill(), noStroke(), stroke(), tint(), noTint(), strokeWeight(), strokeCap(), strokeJoin(), imageMode(), rectMode(), ellipseMode(), colorMode(), colorMode(), colorMode(), textAlign(), textFont(), textSize(), textLeading(), applyMatrix(), resetMatrix(), rotate(), scale(), shearX(), shearY(), translate(), noiseSeed()***.

Дополнительные настройки стиля хранятся в режиме WEBGL. Они управляются следующими функциями: ***setCamera(), ambientLight(), directionalLight(), pointLight(), texture(), specularMaterial(), shininess(), normalMaterial()*** и ***shader()***.

Синтаксис:

push()
pop()

scale()

Функция ***scale()*** увеличивает или уменьшает размер объекта путем расширения или сжатия вершин. Объекты всегда масштабируются относительно начала системы координат. Значения масштаба задаются в виде десятичных процентов. Например, вызов функции масштабирования ***(2,0)*** увеличивает размеры объекта на 200%.

Преобразования применяются ко всему, что происходит после и последующие вызовы функции умножает их эффект. Например, вызов ***функции scales(2.0),*** а затем ***scales(1.5)*** эквивалентен ***scales(3.0)***. Если функция ***scale()*** вызывается внутри функции ***draw(),*** преобразование сбрасывается при каждом вызове этой функции.

Использование этой функции с параметром ***z*** доступно только в ***режиме WEBGL***. Этой функцией можно дополнительно управлять с помощью функций ***push()*** и ***pop()***.

Синтаксис:

scale(s, [y], [z])
scale(scales)

Параметры:

s	Значение: | p5.Vector | Number[]: процент для масштабирования объекта или процент для масштабирования объекта по оси x, если задано несколько аргументов
y	Значение: процент масштабирования объекта по оси y (опционально)
z	Значение: процент масштабирования объекта по оси z (только в режиме webgl) (опционально)
scales	p5.Vector | Number[]: процент масштабирования объекта

translate()

Задает значение для перемещения объектов в сцене. Параметр ***x*** указывает перенос влево/вправо, параметр ***y*** указывает перенос вверх/вниз.

Преобразования являются кумулятивными и применяются ко всему, что происходит после и повторные вызовы данной функции накапливают эффект. Например, вызов метода ***translate(50, 0),*** а затем ***translate(20, 0)*** эквивалентен вызову ***translate(70, 0)***. Если внутри функции ***draw()*** вызывается метод ***translate(***), преобразование сбрасывается в момент повторного вызова этой функции. Этой функцией можно управлять с помощью ***push()*** и ***pop()***.

Синтаксис:

translate(x, y, [z])
translate(vector)

Параметры:

x	Значение: число пикселей переноса влево/вправо
y	Значение: число пикселей переноса вверх / вниз
z	Значение: число пикселей переноса вперед/назад (только webgl) (опционально)
vector	p5.Vector: вектор, с которым он переносится

rotate()

Поворачивает объект на величину, заданной параметром angle. Эта функция представляет angleMode, поэтому углы могут быть вставлены либо в радианах (RADIANS), либо в градусах (DEGREES).

Объекты всегда поворачиваются вокруг своего положения относительно начала координат, а положительные числа вращают объекты по часовой стрелке. Преобразования применяются ко всему, что происходит после и повторные вызовы этой функции накапливают свой эффект. Например, вызов ***rotate(HALF_PI),*** а затем ***rotate(HALF_PI)*** эквивалентен вызову ***rotate(PI)***. Все преобразования сбрасываются к повторному вызову функции draw().

Технически функция ***rotate()*** выполняет умножение текущей матрицы преобразования на матрицу вращения. Этой функцией можно управлять с помощью ***push()*** и ***pop()***.

Синтаксис:

rotate(angle, [axis])

Параметры:

angle	Значение: угол поворота, заданный в радианах или градусах, в зависимости от текущего режима представления углов
axis	p5.Vector | Număr []: (в 3D-режиме) ось вращения (опционально)

rotateX(), rotateY(), rotateZ()

Поворачивает объект вокруг оси ***X*** со значением угла, указанным в параметре ***angle***. Углы могут быть заданы либо в радианах (RADIANS), либо в градусах (DEGREES).

Объекты всегда поворачиваются вокруг своего положения относительно начала координат, а положительные числа вращают объекты по часовой стрелке. Все преобразования сбрасываются перед новым вызовом функции draw().

Синтаксис:

rotateX(angle)
rotateY(angle)
rotateZ(angle)

Параметры:

angle	Значение: угол поворота, заданный в радианах или градусах, в зависимости от текущего режима представления углов

model()

Воспроизводит 3D-модель в сцене.

Синтаксис:

model(model)

Параметры:

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

Содержание программы:

```
let fr = 30;
let angle = 0;
function preload()
{
  base = loadModel('Base.obj');
  fan = loadModel('Fan.obj');
  tail = loadModel('Tail.obj');
}
function setup()
{
  createCanvas(400, 400, WEBGL);
  normalMaterial();
  frameRate(fr);
  angleMode(DEGREES);
}
function draw()
{
  angle = 45*sin(millis()/50);
```

```
  orbitControl();
  background(50);
  noStroke();
  push();
  scale(0.025);
  translate(0, 0, -4000);
  model(base);
  pop();
  push();
  scale(0.025);
  rotateZ(angle);
  translate(0, -200, 4650);
  rotateY(millis()/10);
  model(fan);
  pop();
  push();
  scale(0.025);
  translate(0, 0, 4650);
  rotateZ(angle);
  model(tail);
  pop();
}
```

Описание программы:

Переменная ***fr*** сохраняет количество воспроизводимых кадров в секунду, количество вызовов функции draw() для воспроизведения сцены.

Переменная ***angle*** предназначена для удержания угла отклонения ориентира ветрового насоса и пропеллера.

В функции ***preload(),*** предназначенной для загрузки 3D моделей основания (***Base.obj***), пропеллера (***Fan.obj***) и ориентира (***Tail.obj***) модели водяного насоса ветрового действия хранятся в файлах. OBJ. Способ добавления в проект .obj файлов, содержащих геометрию 3D объектов, описан в лабораторной работе № 4.

В функции ***setup()***, которая вызывается только один раз при запуске программы, выполняются все необходимые конфигурации для работы в режиме 3D, такие как вызов функции ***createdCanvas (400, 400, WEBGL),*** создающей 3D-сцену, отрисованную с помощью холста с размерами 400 на 400 пикселей.

Вызывается функция ***normalMaterial()*** при помощи которой устанавливается материал нормальной геометрии, который является материалом, на который не влияет свет, не является отражающим, таким образом поверхности ориентированные по оси X становятся красными, ориентированные по оси Y становятся зелеными, а ориентированные по оси Z - синими.

С помощью функции ***frameRate(fr)*** устанавливается частота кадров, равная 30 кадрам в секунду.

С помощью функции ***angleMode(DEGREES)*** задается режим представления углов поворота в градусах.

Впоследствии функция ***draw(),*** которая рисует сцену, вызывается многократно с частотой 30 раз в секунду. Переменная угла сохраняет значение в градусах угла поворота ориентира. Это значение изменяется в зависимости от времени, прошедшего с начала запуска программы, представленной в миллисекундах. Значение угла изменяется со временем в

диапазоне от -45 до +45 градусов. Это изменение достигается с помощью инструкции ***angle = 45*sin(millis())/50)*** которая обеспечивает анимацию внутри сцены.

Используется функция ***orbitControl()***, которая обеспечивает управление и позиционирование камеры внутри сцены. С помощью функции фона ***(50)*** устанавливается цвет фона. С помощью функции ***noStroke()*** графический режим рендеринга краев задается так, чтобы они не были видны. Для индивидуального применения преобразований к каждому объекту каждый из них попадает в ***push()***, ***pop()***. Внутри этой конструкции сделаны все необходимые преобразования для каждого графического объекта. Для загрузки объектов в сцену все объекты масштабируются с помощью функции ***scale(0,025),*** которая уменьшает размеры объектов на соответствующий коэффициент. Модель основания водяного насоса переносится по оси ***Z*** вниз на 4000 единиц для центрирования в видоискателе камеры. Базовая модель воспроизводится в сцене с помощью вызова функции ***model(base).***

Для представления пропеллера над ним выполняется ряд преобразований, таких как масштабирование, вращение масштабированной модели вокруг оси ***Z*** со значением угла, сохраненным в переменной угла, затем пропеллер переносится при помощи функции ***translate(0, -200, 4650)*** на 200 единиц вперед по оси ***Y*** и 4650 единиц вверх по оси ***Z***. Впоследствии, используя функцию ***rotateY(millis()/10),*** пропеллер вращается вокруг оси Y положения относительно начала и имитирует влияние ветра на пропеллер.

Для представления ориентира насоса его размеры масштабируются с тем же коэффициентом, а затем переносятся вверх по оси ***Z*** при помощи ***translate(0, 0, 4650)***, затем поворачиваются вокруг оси ***Z*** при помощи функции ***rotateZ(angle).***

Все модели воспроизводятся в сцене с помощью вызываемой функции ***model(),*** каждая в своем ***push()***, ***pop()***.

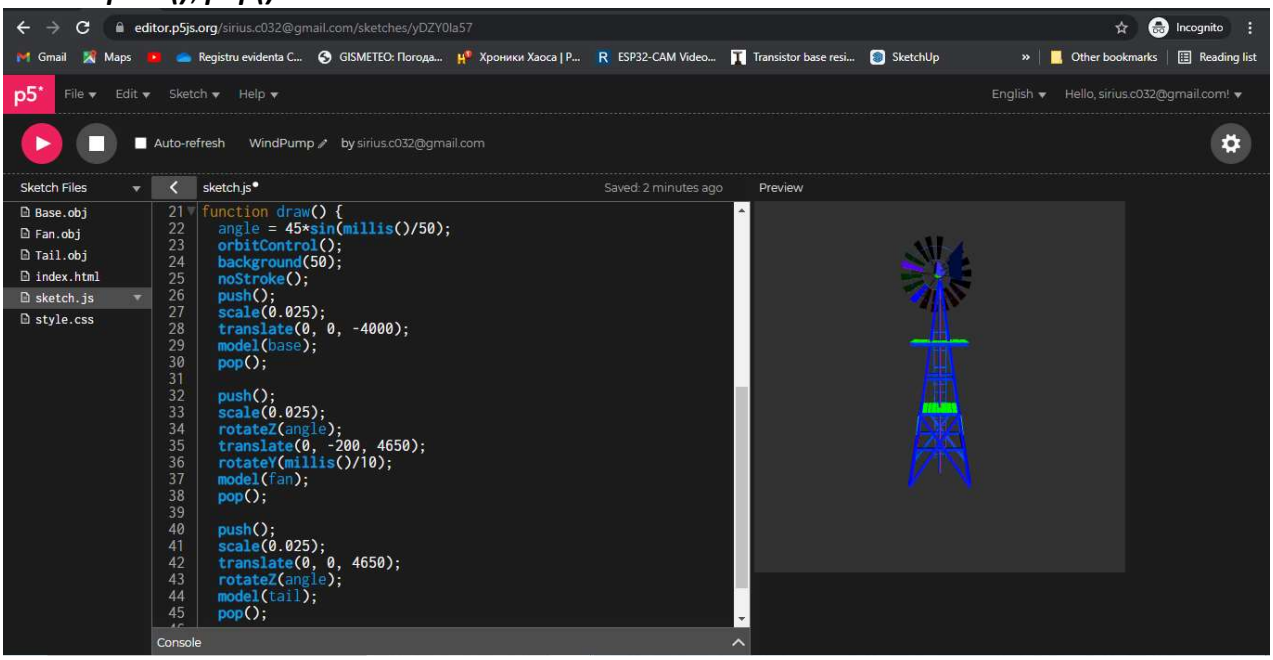

Рис. 5.7. Модель водяного насоса в редакторе p5.js

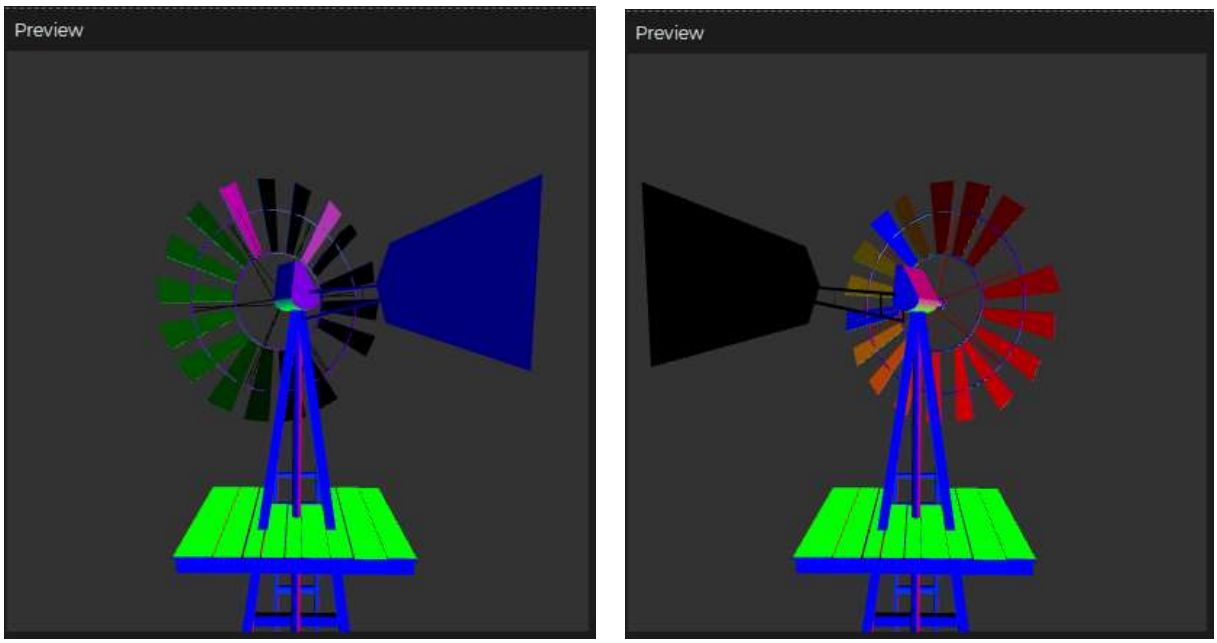

Рис. 5.8. Результат выполнения программы в редакторе p5.js

Лабораторная работа № 5
Тема: 3D-преобразования

Цель работы: Получение практических знаний в области синтеза динамических 3D графических сцен с использованием стандартных функций переноса, вращения и масштабирования в библиотеке p5.js.

Задачи работы:

1. Разработать программу для синтеза динамической 3D-сцены с использованием стандартных функций переноса, поворота и масштабирования в библиотеке p5.js.

2. Разработать программу, которая создает динамическую 3D-сцену по варианту, указанному в таблице 4.1. Для создания сцены могут быть использованы существующие 3D-графические объекты из репозитории 3D Warehouse или 3D объекты смоделированные самостоятельно.