**МАИ**

**Отчет о проделанной работе**

Елка и Музыка в системе Octave

**Выполнил:**

Студент группы М7О-114БВ-24

Фельдман Лев Борисович

**Проверил:**Заведующий кафедрой 702

Кривилёв Александр Владимирович

Москва 2024

**Задание 1: Создать Новогоднюю елку в системе Octave.**

*Шаг 1. Создание рабочего окружения. Задание первых слоев Елки, состоящей их трех конусов, и ствола елки, в виде из цилиндра.*

clc;

clear;

figure;

hold on;

axis equal;

axis off;

view(3);

az = 45;

el = 30;

set(gca, 'CameraPosition', [3, 3, 3]); % Камера на фиксированном месте

set(gca, 'CameraTarget', [0, 0, 0]); % Точка фокуса (центр сцены)

set(gca, 'CameraUpVector', [0, 0, 1]); % Направление вверх для камеры

set(gca, 'CameraViewAngle', 6); % Установить угол обзора

% Создание группы объектов для елки

treeGroup = hgtransform;

% Слои елки (вверх направленные конусы)

num\_layers = 3; % Количество слоев елки

layer\_height = 1.2; % Высота каждого конуса

base\_height = 0; % Начальная высота основания

% Массив для хранения объектов слоев елки

layerHandles = gobjects(num\_layers, 1);

for i = 1:num\_layers

% Параметры конусов

radius = 1.2 - (i-1) \* 0.4; % Уменьшающийся радиус

height = layer\_height; % Высота каждого конуса

[X, Y, Z] = cylinder([radius, 0], 50); % Конус вверх

Z = Z \* height + base\_height; % Смещение по оси Z для наложения слоев

base\_height = base\_height + height; % Обновление высоты для следующего слоя

h = surf(X, Y, Z, 'EdgeColor', 'none'); % Рисуем конус с плоскими поверхностями

layerHandles(i) = h; % Сохраняем ручку слоя

set(h, 'Parent', treeGroup); % Добавление конуса в группу объектов

end

% Ствол елки (цилиндр)

trunk\_radius = 0.2;

trunk\_height = 1;

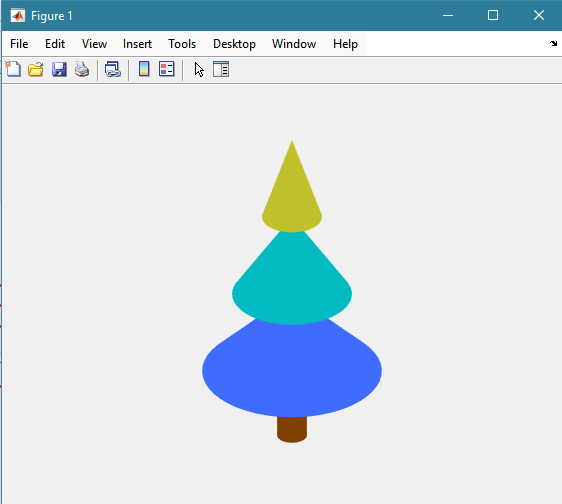
[X, Y, Z] = cylinder(trunk\_radius, 30);

Z = Z \* trunk\_height;

Z = Z - trunk\_height; % Расположение ствола внизу

h = surf(X, Y, Z, 'FaceColor', [0.5, 0.25, 0], 'EdgeColor', 'none'); % Коричневый ствол

set(h, 'Parent', treeGroup); % Добавление ствола в группу объектов



*Шаг 2. Добавляем звезду на вершину елки.*

% Звезда на вершине елки

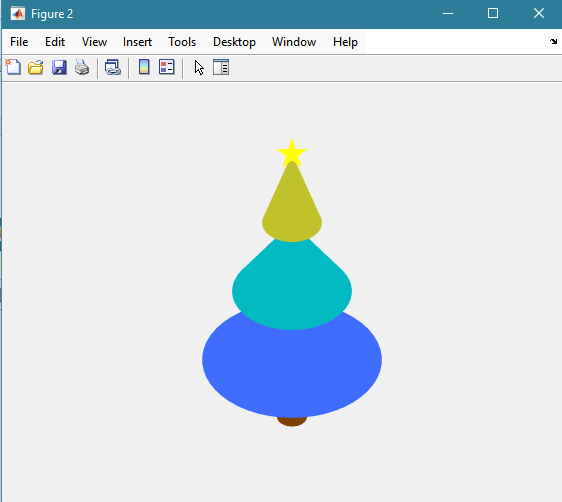
star\_x = 0;

star\_y = 0;

star\_z = base\_height; % Позиция звезды на верхушке

h = scatter3(star\_x, star\_y, star\_z, 600, 'yellow', 'filled', 'Marker', 'p');

set(h, 'Parent', treeGroup); % Добавление звезды в группу объектов



*Шаг 3. Настраиваем отображение и освещение и создаем снежинки.*

% Освещение и настройки отображения

lightangle(-45, 30);

lighting gouraud;

camlight;

title('Новогодняя Елка');

% Параметры снежинок

num\_snowflakes = 50;

snowflakeX = rand(1, num\_snowflakes) \* 4 - 2; % Случайные X- позиции

snowflakeY = rand(1, num\_snowflakes) \* 4 - 2; % Случайные Y- позиции

snowflakeZ = rand(1, num\_snowflakes) \* 2 + 2; % Случайные Z- позиции (высота)

snowflakes = scatter3(snowflakeX, snowflakeY, snowflakeZ, 50, 'w', 'filled'); % Снежинки

% Скорость падения снежинок

fallingSpeed = rand(1, num\_snowflakes) \* 0.03 + 0.02;

*Шаг 4. С помощью цикла while анимируем елку и снежинки.*

% Анимация бесконечного вращения с изменением цвета градиента и падением снега

angle = 0; % Начальный угол

while true

% Определяем матрицу вращения

rotationMatrix = makehgtform('zrotate', deg2rad(angle));

% Применяем вращение к группе объектов елки

set(treeGroup, 'Matrix', rotationMatrix);

% Каждые 60 кадров (примерно каждые 2 секунды) меняем цвет градиента

if mod(angle, 60) == 0

% Генерация нового цветового градиента для всех слоев

for i = 1:num\_layers

% Генерация случайных цветов для слоев

color = rand(1, 3); % Случайный цвет

set(layerHandles(i), 'FaceColor', color); % Обновление цвета слоя

end

end

% Обновляем позиции снежинок

snowflakeZ = snowflakeZ - fallingSpeed; % Падение снежинок по оси Z

% Проверка, если снежинка выходит за пределы, сбрасываем ее наверх

for i = 1:num\_snowflakes

if snowflakeZ(i) < 0

snowflakeZ(i) = rand \* 2 + 2; % Сброс на верхнюю позицию

snowflakeX(i) = rand \* 4 - 2; % Случайная позиция X

snowflakeY(i) = rand \* 4 - 2; % Случайная позиция Y

end

end

% Обновление позиций снежинок

set(snowflakes, 'XData', snowflakeX, 'YData', snowflakeY, 'ZData', snowflakeZ);

% Увеличиваем угол

angle = angle + 1;

if angle >= 360

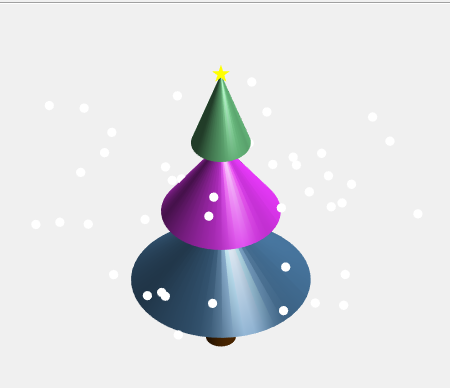
angle = 0;

end

pause(0.03);

end

**Задание 2. Сделать Песню ‘Jingle Bells’ в Octave**

*Шаг 1. Создаем таблицу частоты нот в герцах и таблицу Мелодии Jingle Bells*

noteFreqs = struct('C4', 261.63, 'D4', 293.66, 'E4', 329.63, 'F4', 349.23, 'G4', 392.00, 'A4', 440.00, 'B4', 493.88, 'C5', 523.25, 'D5', 587.33, 'E5', 659.25, 'F5', 698.46);

song = { ...

'E4', 0.5; 'E4', 0.5; 'E4', 1.0; ...

'E4', 0.5; 'E4', 0.5; 'E4', 1.0; ...

'E4', 0.5; 'G4', 0.5; 'C4', 0.5; 'D4', 0.5; 'E4', 2.0; ...

'F4', 0.5; 'F4', 0.5; 'F4', 0.5; 'F4', 0.5; 'F4', 0.5; 'E4', 0.5; 'E4', 0.5; 'E4', 0.5; ...

'E4', 0.5; 'E4', 0.5; 'D4', 0.5; 'D4', 0.5; 'E4', 0.5; 'D4', 0.5; 'G4', 2.0; ...

};

fs = 44100; % Частота дискретизации

songWave = [];

*Шаг 2. С помощью цикла for генерируем звук для каждой ноты.*

for i = 1:size(song, 1)

note = song{i, 1};

duration = song{i, 2};

if strcmp(note, 'pause')

t = 0:1/fs:duration;

wave = zeros(size(t));

else

freq = noteFreqs.(note);

t = 0:1/fs:duration - 1/fs;

wave = sin(2 \* pi \* freq \* t);

end

songWave = [songWave, wave];

end

while true

sound(songWave, fs);

pause(length(songWave) / fs);

end