**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Лабораторная работа № 6**

Тема: Создание шейдерных анимационных эффектов в OpenGL

Студент: Кондратьев Егор Алексеевич

Группа: 80-306

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

1. Постановка задачи

Для поверхности, созданной в л.р. №4-5, обеспечить выполнение следующего шейдерного эффекта - Анимация. Цветовые координаты изменяются по синусоидальному закону.

1. Описание программы

Программа состоит из окна для взаимодействия с многогранником, из canvas для визуализации, вращения многогранника и изменения некоторых его параметров.

1. Набор тестов

1. Изначальный вид.

2. Вращение многогранника.

3. Увеличение точности аппроксимации.

1. Результаты выполнения тестов

1. Изначальный вид.

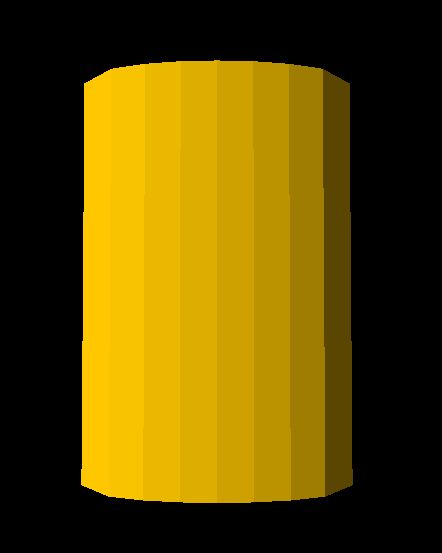


Рис. 4.1 Результат открытия окна

2. Вращение многогранника.

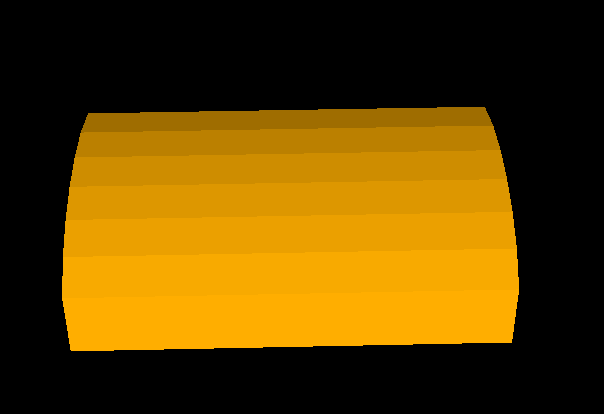


Рис. 4.2 Результат вращения многогранника

3. Увеличение точности аппроксимации.

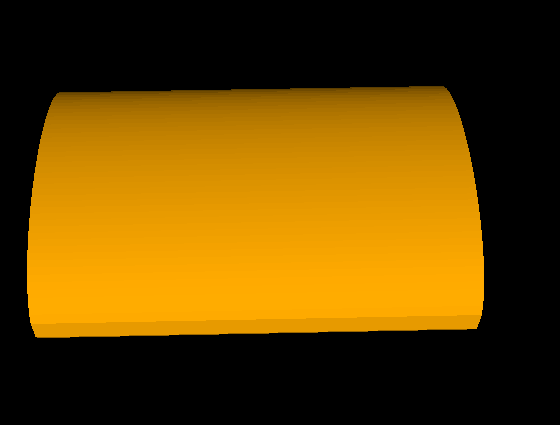


Рис. 4.3 Результат увеличение точности аппроксимации

1. Листинг программы

**<!-- Кондратьев Егор Алексеевич М8О-308Б -->**

**<!-- Вариант: Прямой эллиптический цилиндр. -->**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="utf-8">**

**<link rel='stylesheet' href='style.css'>**

**<title>Elliptical cylinder - Кондратьев Егор</title>**

**</head>**

**<body onLoad="onLoad();">**

**<div id="container" style="width:600px; height:600px; background-color:#000000">**

**</div>**

**<div class='controls'>**

**<div class='note'>**

**<p>**

**Зажмите левую кнопку мыши и передвигайте курсор для вращения октаэдра**

**</p>**

**<p>**

**Используйте колесико мыши для изменения масштаба**

**</p>**

**</div>**

**<label for="approx">Точность аппроксимации</label>**

**<div class='range'>**

**<input type="range" id="approx" name="volume" min="6" max="60" step="2" value="8">**

**</div>**

**<label for="intens">Интенсивность света</label>**

**<div class='range'>**

**<input type="range" id="intens" name="volume" min="0" max="1" step="0.01" value="0.8">**

**</div>**

**</div>**

**<script src="Three.js"></script>**

**<script src="OrbitControls.js"></script>**

**<script type="main.js"></script>**

**<script src="cylinder.js"></script>**

**</body>**

**</html>**

**var camera, scene, renderer, controls, geom, mesh, light;**

**var delta = 0, colorFlag = true;**

**function init() {**

**// Grab our container div**

**var container = document.getElementById("container");**

**const approx = document.getElementById("approx");**

**const intens = document.getElementById("intens");**

**// Create the Three.js renderer, add it to our div**

**renderer = new THREE.WebGLRenderer();**

**renderer.setPixelRatio(window.devicePixelRatio);**

**renderer.setSize(container.offsetWidth, container.offsetHeight);**

**container.appendChild( renderer.domElement );**

**// Create a new Three.js scene**

**scene = new THREE.Scene();**

**// Create a camera and add it to the scene**

**camera = new THREE.PerspectiveCamera( 45, container.offsetWidth / container.offsetHeight, 1, 500 );**

**camera.position.set( 0, 0, 200 );**

**scene.add( camera );**

**controls = new THREE.OrbitControls( camera, renderer.domElement );**

**controls.enableDamping = true; // an animation loop is required when either damping or auto-rotation are enabled**

**controls.dampingFactor = 0.25;**

**controls.screenSpacePanning = false;**

**controls.minDistance = 10;**

**controls.maxDistance = 600;**

**// controls.maxPolarAngle = Math.PI;**

**scene.add( new THREE.HemisphereLight( 0x606060, 0x404040 ) );**

**light = new THREE.DirectionalLight(0xffffff);**

**light.intensity = intens.value;**

**light.reflectivity = intens.value;**

**light.position.set( 600, 600, 600 ).normalize();**

**scene.add( light );**

**geom = initGeom();**

**// var color = new THREE.Color("rgb(0, 255, 255)");**

**var color = new THREE.Color("hsl(0, 100%, 50%)");**

**var material = new THREE.MeshLambertMaterial( { color: color } );**

**mesh = new THREE.Mesh(geom, material);**

**scene.add( mesh );**

**}**

**const A = 20, B = 10, H = 60;**

**let num = approx.value;**

**approx.onchange = function (e) {**

**num = approx.value;**

**scene.remove(mesh);**

**geom = initGeom();**

**var cyan = new THREE.Color("rgb(0, 255, 255)");**

**var material = new THREE.MeshLambertMaterial( { color: cyan } );**

**mesh = new THREE.Mesh(geom, material);**

**scene.add( mesh );**

**render();**

**}**

**intens.onchange = function(e) {**

**light.intensity = intens.value;**

**}**

**function fplus(x) {**

**return Math.sqrt((1 - (x \* x) / (A \* A)) \* B \* B);**

**}**

**function fminus(x) {**

**return -Math.sqrt((1 - (x \* x) / (A \* A)) \* B \* B);**

**}**

**function initGeom() {**

**//аппроксимируем выпуклое тело треугольниками**

**var geometry = new THREE.BufferGeometry();**

**var vertices = [];**

**var step = 2 \* A / num;**

**//лицевые треугольники**

**var z = H / 2;**

**var prevX = -A, prevY = fplus(prevX);**

**for (var x = -A + step; Math.floor(x) < A; x += step) {**

**var y = fminus(x);**

**vertices.push( 0, 0, z );**

**vertices.push( prevX, prevY, z);**

**vertices.push( x, y, z );**

**geometry.addAttribute( 'position', new THREE.Float32BufferAttribute( vertices, 3 ) );**

**vertices.push( x, y, -z );**

**vertices.push( x, y, z );**

**vertices.push( prevX, prevY, z );**

**geometry.addAttribute( 'position', new THREE.Float32BufferAttribute( vertices, 3 ) );**

**prevX = x;**

**prevY = y;**

**}**

**var x = A, y = fminus(x);**

**vertices.push( 0, 0, z );**

**vertices.push( prevX, prevY, z);**

**vertices.push( x, y, z );**

**geometry.addAttribute( 'position', new THREE.Float32BufferAttribute( vertices, 3 ) );**

**vertices.push( x, y, -z );**

**vertices.push( x, y, z );**

**vertices.push( prevX, prevY, z );**

**/////////////////////////////////////////////**

**prevX = -A, prevY = fminus(prevX);**

**for (var x = -A + step; x < A; x += step) {**

**var y = fplus(x);**

**vertices.push( x, y, z );**

**vertices.push( prevX, prevY, z);**

**vertices.push( 0, 0, z );**

**geometry.addAttribute( 'position', new THREE.Float32BufferAttribute( vertices, 3 ) );**

**vertices.push( prevX, prevY, -z);**

**vertices.push( prevX, prevY, z);**

**vertices.push( x, y, z );**

**geometry.addAttribute( 'position', new THREE.Float32BufferAttribute( vertices, 3 ) );**

**prevX = x;**

**prevY = y;**

**}**

**var x = A, y = fplus(x);**

**vertices.push( x, y, z );**

**vertices.push( prevX, prevY, z);**

**vertices.push( 0, 0, z );**

**geometry.addAttribute( 'position', new THREE.Float32BufferAttribute( vertices, 3 ) );**

**vertices.push( prevX, prevY, -z);**

**vertices.push( prevX, prevY, z);**

**vertices.push( x, y, z );**

**geometry.addAttribute( 'position', new THREE.Float32BufferAttribute( vertices, 3 ) );**

**// задние треугольники**

**z = -H / 2;**

**var prevX = -A, prevY = fplus(prevX);**

**for (var x = -A + step; x < A; x += step) {**

**var y = fplus(x);**

**vertices.push( x, y, z );**

**vertices.push( 0, 0, z );**

**vertices.push( prevX, prevY, z);**

**geometry.addAttribute( 'position', new THREE.Float32BufferAttribute( vertices, 3 ) );**

**vertices.push( x, y, z);**

**vertices.push( prevX, prevY, z);**

**vertices.push( x, y, -z );**

**geometry.addAttribute( 'position', new THREE.Float32BufferAttribute( vertices, 3 ) );**

**prevX = x;**

**prevY = y;**

**}**

**var x = A, y = fplus(x);**

**vertices.push( x, y, z );**

**vertices.push( 0, 0, z );**

**vertices.push( prevX, prevY, z);**

**geometry.addAttribute( 'position', new THREE.Float32BufferAttribute( vertices, 3 ) );**

**vertices.push( x, y, z);**

**vertices.push( prevX, prevY, z);**

**vertices.push( x, y, -z );**

**geometry.addAttribute( 'position', new THREE.Float32BufferAttribute( vertices, 3 ) );**

**/////////////////////////////////////////////**

**prevX = -A, prevY = fminus(prevX);**

**for (var x = -A + step; x < A; x += step) {**

**var y = fminus(x);**

**vertices.push( prevX, prevY, z);**

**vertices.push( 0, 0, z );**

**vertices.push( x, y, z );**

**geometry.addAttribute( 'position', new THREE.Float32BufferAttribute( vertices, 3 ) );**

**vertices.push( prevX, prevY, -z);**

**vertices.push( prevX, prevY, z);**

**vertices.push( x, y, z);**

**geometry.addAttribute( 'position', new THREE.Float32BufferAttribute( vertices, 3 ) );**

**prevX = x;**

**prevY = y;**

**}**

**var x = A, y = fminus(x);**

**vertices.push( prevX, prevY, z);**

**vertices.push( 0, 0, z );**

**vertices.push( x, y, z );**

**geometry.addAttribute( 'position', new THREE.Float32BufferAttribute( vertices, 3 ) );**

**vertices.push( prevX, prevY, -z);**

**vertices.push( prevX, prevY, z);**

**vertices.push( x, y, z);**

**geometry.addAttribute( 'position', new THREE.Float32BufferAttribute( vertices, 3 ) );**

**geometry.computeVertexNormals();**

**return geometry;**

**}**

**function changeColor(x) {**

**// scene.children[3].material.color.setHSL(0.7, 1, 0,5);**

**scene.children[3].material.color = new THREE.Color("hsl(" + x + ", 100%, 50%)");**

**}**

**function animate() {**

**requestAnimationFrame( animate );**

**scene.children[3].material.color = new THREE.Color("hsl(" + delta%360 + ", 100%, 50%)");**

**if (delta >= 360) {**

**colorFlag = false;**

**}else if (delta <= 0) {**

**colorFlag = true;**

**}**

**if (colorFlag) {**

**delta += 1;**

**} else {**

**delta -= 1;**

**}**

**controls.update();**

**render();**

**}**

**function render() {**

**renderer.render( scene, camera );**

**}**

**function onLoad()**

**{**

**init();**

**render();**

**animate();**

**}**

ЛИТЕРАТУРА

1. Введение в 3D: основы Three.js [Электронный ресурс]URL: <https://habr.com/ru/post/494810/> (Дата обращения: 10.11.2021).

2. Three.js - импорт орбитальных контроллеров [Электронный ресурс]URL: <https://themightyprogrammer.dev/snippet/orbit-controls-webpack> (Дата обращения: 10.11.2021).

3. Орбитальные контроллеры - Three.js [Электронный ресурс]URL: <https://threejs.org/docs/#examples/en/controls/OrbitControls> (Дата обращения: 10.11.2021).