Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем и программной инженерии

Лабораторная работа № 5

по дисциплине "Программирование графических приложений"

ТЕМА РАБОТЫ:

Плоские объекты в WebGL

Выполнил:

студент гр. ПРИм-124

Парахин К.В.

Принял:

Жигалов И.Е.

Владимир 2024 г.

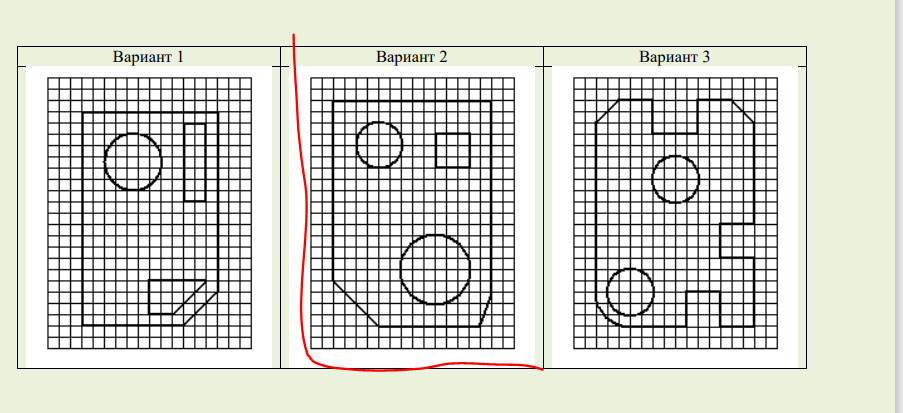
Цель работы:

Изучение способов построения плоских объектов при формировании графических изображений с использованием WebGL

Выполнение работы:

Индивидуальный вариант 2 (14 % 12 = 2)

На основе программ разделов 5.1, 5.2 разработать программу построения двухмерного объекта в соответствие с вариантом задания (табл. 1).



Листинг:

// ���������� ����������

var container, camera, controls, scene, renderer, light;

// �������� �������� ����� ������ �������� ��������

window.onload = function()

{

init();

animate();

}

function init()

{

scene = new THREE.Scene(); //������� �����

AddCamera( 0, 300, 500); //��������� ������

AddLight( 0, 0, 500 ); //������������� ����� ����

//������� ��������

renderer = new THREE.WebGLRenderer( { antialias: true } );

renderer.setClearColor( 0xffffff );

renderer.setSize( window.innerWidth, window.innerHeight );

container = document.getElementById('MyWebGLApp');

container.appendChild( renderer.domElement );

var planeMaterial = new THREE.MeshBasicMaterial({ wireframe: true, color: 0x9999ff });

var planGeo = new THREE.PlaneGeometry(100, 100, 20, 20);

var plane = new THREE.Mesh(planGeo, planeMaterial);

plane.position.set(50,80,50);

plane.rotation.x = Math.PI;

plane.rotation.y = 0;

plane.rotation.z = Math.PI;

// �������������

var material = new THREE.LineBasicMaterial ( { color: 0xcc0000 } );

// отрисовка внешнего контура

var geometry1 = new THREE.Geometry;

geometry1.vertices.push(new THREE.Vector3 (10, 30, 0 ));

geometry1.vertices.push(new THREE.Vector3 (10, 110, 0 ));

geometry1.vertices.push(new THREE.Vector3 (80, 110, 0 ));

geometry1.vertices.push(new THREE.Vector3 (85, 25, 0 ));

geometry1.vertices.push(new THREE.Vector3 (75, 10, 0 ));

geometry1.vertices.push(new THREE.Vector3 (30, 10, 0 ));

geometry1.vertices.push(new THREE.Vector3 (10, 30, 0 ));

// большая окружность

var radiusLarge = 15;

var pi = Math.PI;

var step = 0.01;

var centerXLarge = 55;

var centerYLarge = 35;

var startPoint = pi/2;

var geometry2 = new THREE.Geometry;

for (i = 0; i <= 200; i+=1)

{

    var x = Math.sin(startPoint - pi \* step \* i) \* radiusLarge + centerXLarge;

    var y = Math.cos(startPoint - pi \* step \* i) \* radiusLarge + centerYLarge;

    geometry2.vertices.push(new THREE.Vector3 (x, y, 0 ));

}

// малая окружность

var radiusLarge = 10;

var centerXLarge = 30;

var centerYLarge = 90;

var startPoint = pi/2;

var geometry3 = new THREE.Geometry;

for (i = 0; i <= 200; i+=1)

{

    var x = Math.sin(startPoint - pi \* step \* i) \* radiusLarge + centerXLarge;

    var y = Math.cos(startPoint - pi \* step \* i) \* radiusLarge + centerYLarge;

    geometry3.vertices.push(new THREE.Vector3 (x, y, 0 ));

}

// квадрат внутри

var geometry4 = new THREE.Geometry;

geometry4.vertices.push(new THREE.Vector3 (60, 80, 0 ));

geometry4.vertices.push(new THREE.Vector3 (60, 95, 0 ));

geometry4.vertices.push(new THREE.Vector3 (75, 95, 0 ));

geometry4.vertices.push(new THREE.Vector3 (75, 80, 0 ));

geometry4.vertices.push(new THREE.Vector3 (60, 80, 0 ));

// Отрисовка полученного линиями

var line1 = new THREE.Line( geometry1, material );

var line2 = new THREE.Line( geometry2, material );

var line3 = new THREE.Line( geometry3, material );

var line4 = new THREE.Line( geometry4, material );

scene.add(plane);

scene.add(line1);

scene.add(line2);

scene.add(line3);

scene.add(line4);

}

function animate()

{

requestAnimationFrame(animate);

render();

}

function render()

{

controls.update();

renderer.render(scene, camera);

}

function AddCamera(X,Y,Z)

{

camera = new THREE.PerspectiveCamera( 45, window.innerWidth / window.innerHeight, 1, 10000 );

camera.position.set(X,Y,Z);

controls = new THREE.TrackballControls( camera, container );

controls.rotateSpeed = 0;

controls.noZoom = false;

controls.zoomSpeed = 1.2;

controls.staticMoving = true;

}

function AddLight(X,Y,Z)

{

light = new THREE.DirectionalLight( 0xffffff );

light.position.set(X,Y,Z);

scene.add( light );

}

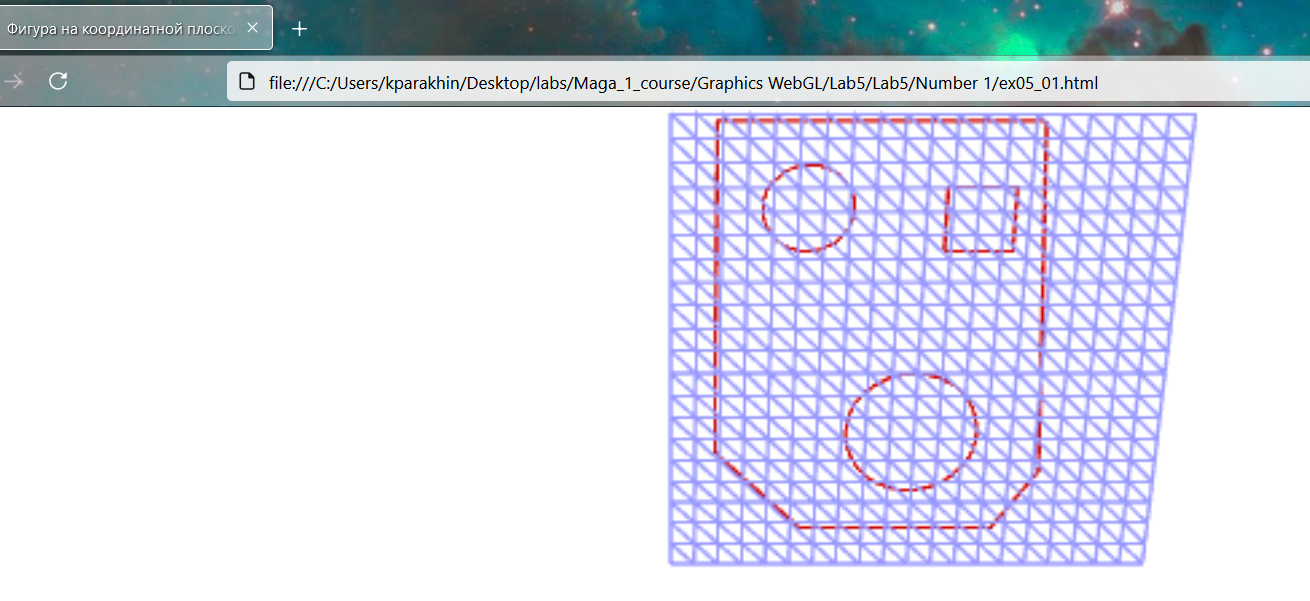
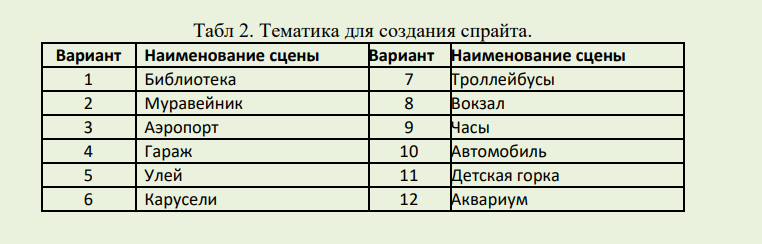


Рисунок 1. Изображение детали на координатной плоскости

В любом растровом графическом редакторе подготовить атлас спрайтов из 10-12 изображений для анимации двухмерного объекта в соответствие с заданной тематикой (табл. 2) с использованием программы раздела 5.3.



Листинг:

<!doctype html>

<html lang="en">

<head>

<title>Муравейник</title>

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

<meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1.0, maximum-scale=1.0">

</head>

<body>

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

<script src="Detector.js"></script>

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

<script src="THREEx.KeyboardState.js"></script>

<script src="THREEx.FullScreen.js"></script>

<script src="THREEx.WindowResize.js"></script>

<div id="infoButton"></div>

<div id="ThreeJS" style="position: absolute; left:0px; top:0px"></div>

<script>

var container, scene, camera, renderer, controls;

var keyboard = new THREEx.KeyboardState();

var clock = new THREE.Clock();

var annie;

init();

animate();

function init()

 {

  scene = new THREE.Scene();

  var SCREEN\_WIDTH = window.innerWidth, SCREEN\_HEIGHT = window.innerHeight;

  var VIEW\_ANGLE = 45, ASPECT = SCREEN\_WIDTH / SCREEN\_HEIGHT, NEAR = 0.1, FAR = 20000;

  camera = new THREE.PerspectiveCamera( VIEW\_ANGLE, ASPECT, NEAR, FAR);

  scene.add(camera);

  camera.position.set(0,10,200);

  camera.lookAt(scene.position);

  if ( Detector.webgl )   renderer = new THREE.WebGLRenderer( {antialias:true} );

  else  renderer = new THREE.CanvasRenderer();

  renderer.setSize(SCREEN\_WIDTH, SCREEN\_HEIGHT);

  container = document.getElementById( 'ThreeJS' );

  container.appendChild( renderer.domElement );

  THREEx.WindowResize(renderer, camera);

  THREEx.FullScreen.bindKey({ charCode : 'm'.charCodeAt(0) });

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

  var light = new THREE.PointLight(0xffffff);

  light.position.set(0,250,0);

  scene.add(light);

  var runnerTexture = new THREE.ImageUtils.loadTexture( 'marsch.png' );

  annie = new TextureAnimator( runnerTexture, 10, 3, 10, 200 );

  var runnerMaterial = new THREE.MeshBasicMaterial( { map: runnerTexture, side:THREE.DoubleSide } );

  var runnerGeometry = new THREE.PlaneGeometry(70, 70, 1, 1);

  var runner = new THREE.Mesh(runnerGeometry, runnerMaterial);

  runner.position.set(-100,25,0);

  scene.add(runner);

 }

function animate()

{

  requestAnimationFrame( animate );

  render();

  update();

}

function update()

{

  var delta = clock.getDelta();

  annie.update(800 \* delta);

  controls.update();

}

function render() { renderer.render( scene, camera ); }

function TextureAnimator(texture, tilesHoriz, tilesVert, numTiles, tileDispDuration)

{

 this.tilesHorizontal = tilesHoriz;

 this.tilesVertical = tilesVert;

 this.numberOfTiles = numTiles;

 texture.wrapS = texture.wrapT = THREE.RepeatWrapping;

 texture.repeat.set( 1 / this.tilesHorizontal, 1 / this.tilesVertical );

 this.tileDisplayDuration = tileDispDuration;

 this.currentDisplayTime = 0;

 this.currentTile = 0;

 this.update = function( milliSec )

 {

   this.currentDisplayTime += milliSec;

   while (this.currentDisplayTime > this.tileDisplayDuration)

   {

    this.currentDisplayTime -= this.tileDisplayDuration;

    this.currentTile++;

    if (this.currentTile == this.numberOfTiles)

    this.currentTile = 0;

    var currentColumn = this.currentTile % this.tilesHorizontal;

    texture.offset.x = currentColumn / this.tilesHorizontal;

    var currentRow = Math.floor( this.currentTile / this.tilesHorizontal );

    texture.offset.y = currentRow / this.tilesVertical;

   }

 };

}

</script>

</body>

</html>

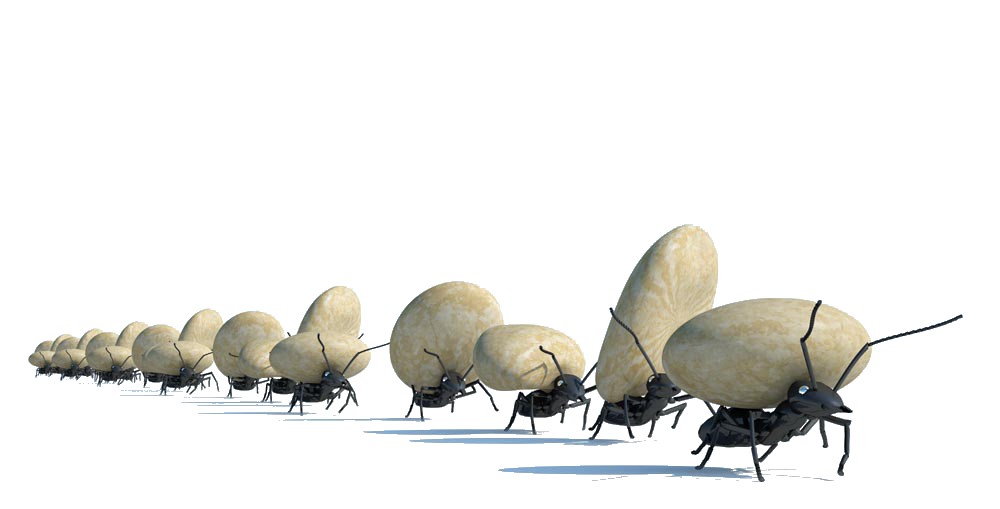


Рисунок 2. Изображение с атласом спрайта муравейника

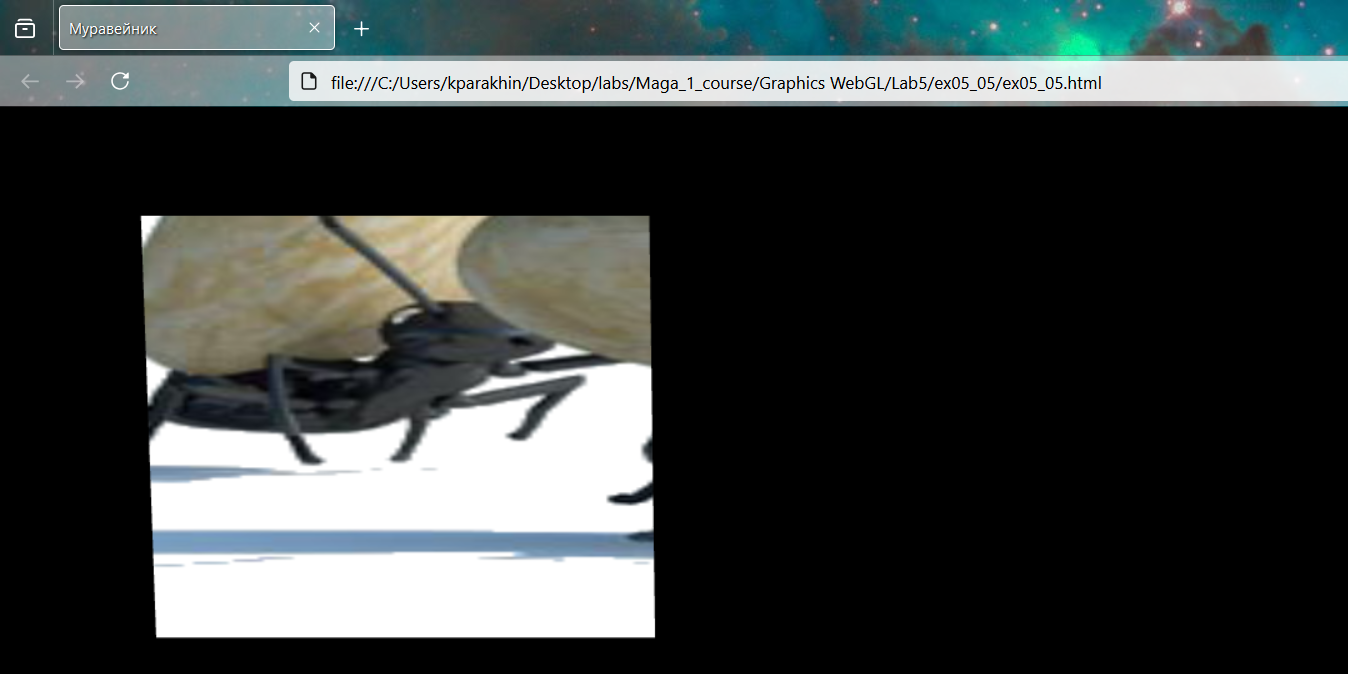


Рисунок 2.1. Один из кадров полученного спрайта

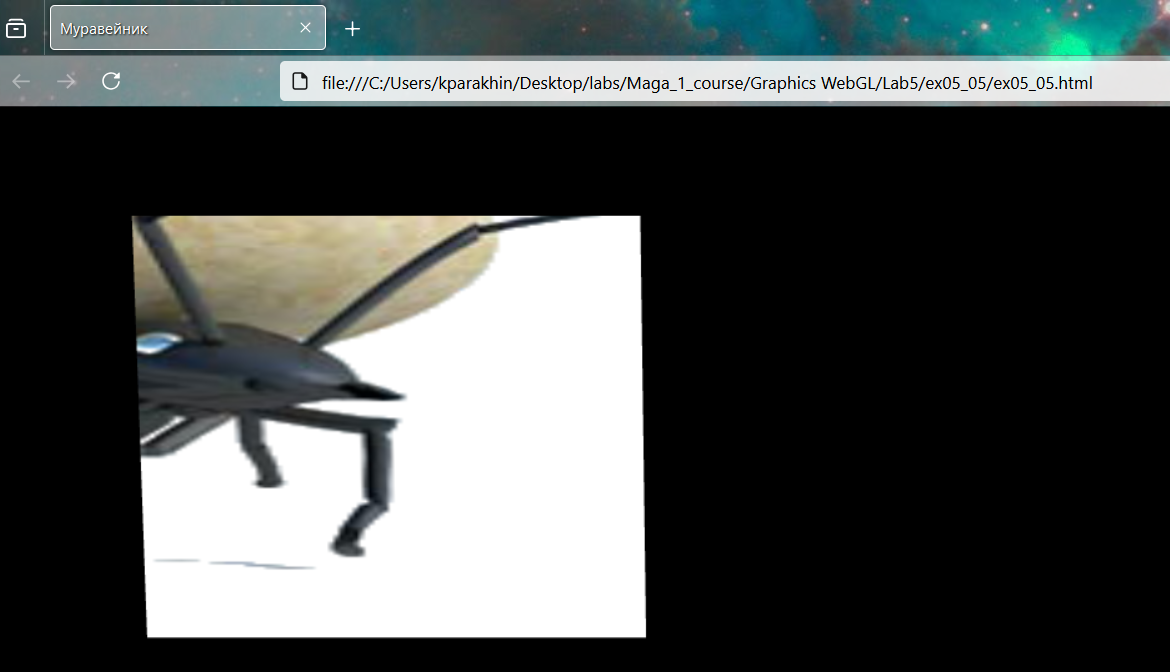


Рисунок 2.2. Еще один кадр из полученного спрайта

Вывод

В результате выполнения работы я провел изучение способов построения плоских объектов при формировании моделей графических объектов с использованием WebGL.