Министерство науки и высшего образования РФ

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образовани**я

**«Томский государственный университет систем управления**

**и радиоэлектроники» (ТУСУР)**

Кафедра автоматизированной обработки информации (АОИ)

**Работа с туманом и прозрачностью**

Отчет о выполнении лабораторной работы

По дисциплине «Компьютерная графика»

Студент гр.\_\_428-2\_\_

\_\_\_\_\_\_\_\_\_Челпанов Д. А.\_\_\_

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

Руководитель

канд. техн. наук, доцент каф.АОИ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.О. Перемитина

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

Томск 2020

# 1 Постановка задачи

**Цель работы:** Сдать долг по лабораторным работам по дисциплине Компьютерная графика.

Выполнение работ: Вариант 20

**Лабораторная работа 7**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Tao.FreeGlut;

using Tao.OpenGl;

using Tao.Platform.Windows;

namespace lr7

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

holst.InitializeContexts();

Glut.glutInit();

}

double pirX = 0, pirY = 0, angle = 90;

double conX = 0, conY = 0, alfa =270;

private void osnova()

{

Gl.glViewport(0, 0, holst.Width, holst.Height);

Gl.glClearColor(0.5f, 0.5f, 0.5f, 1f);

Gl.glClear(Gl.GL\_COLOR\_BUFFER\_BIT);

Gl.glEnable(Gl.GL\_DEPTH\_TEST);

Gl.glClear(Gl.GL\_DEPTH\_BUFFER\_BIT);

Gl.glEnable(Gl.GL\_LIGHTING);

Gl.glEnable(Gl.GL\_LIGHT0);

Gl.glEnable(Gl.GL\_NORMALIZE);

Gl.glLightModelf(Gl.GL\_LIGHT\_MODEL\_TWO\_SIDE, Gl.GL\_TRUE);

Glut.glutInitDisplayMode(Glut.GLUT\_RGB | Glut.GLUT\_DOUBLE | Glut.GLUT\_DEPTH | Glut.GLUT\_ALPHA);

Gl.glMatrixMode(Gl.GL\_PROJECTION);

Gl.glLoadIdentity();

Gl.glFrustum(-0.8, 0.8, -1, 1, 1, 50);

Gl.glMatrixMode(Gl.GL\_MODELVIEW);

Gl.glEnable(Gl.GL\_FOG);

Gl.glLoadIdentity();

Gl.glFogi(Gl.GL\_FOG\_MODE, Gl.GL\_EXP2);

Gl.glFogf(Gl.GL\_FOG\_START, 0.0f);

Gl.glFogf(Gl.GL\_FOG\_END, 1.0f);

float[] fogColor = { 0.5f, 0.5f, 0.5f, 1f };

Gl.glFogfv(Gl.GL\_FOG\_COLOR, fogColor);

Gl.glFogf(Gl.GL\_FOG\_DENSITY, 0.1f);

Gl.glHint(Gl.GL\_FOG\_HINT, Gl.GL\_DONT\_CARE);

}

private void holst\_Load(object sender, EventArgs e)

{

}

private void dodecahedron()

{

Gl.glLoadIdentity();

float[] color\_dod = { 0, 1, 0 };

float[] light0\_dif = { 0.7f, 0.7f, 0.2f };

float[] light0\_pos = { 0.0f, 1.0f, 0.0f, 0.0f };

Gl.glLightfv(Gl.GL\_LIGHT0, Gl.GL\_AMBIENT, light0\_dif);

Gl.glMaterialfv(Gl.GL\_FRONT\_AND\_BACK, Gl.GL\_AMBIENT, color\_dod);

Gl.glLightfv(Gl.GL\_LIGHT0, Gl.GL\_POSITION, light0\_pos);

Gl.glLoadIdentity();

Gl.glTranslated(0, 0, -15);

Gl.glRotated(angle, pirX, pirY, 0);

Gl.glTranslated(0, 0, -8);

Glut.glutSolidDodecahedron();

Gl.glLoadIdentity();

angle += 3;

pirX += 10;

pirY -= 10;

holst.Invalidate();

}

private void conus()

{

Gl.glLoadIdentity();

float[] color\_con = { 0.7f, 0.5f, 1 };

float[] light0\_dif = { 0.7f, 0.7f, 0.2f };

float[] light0\_pos = { 0.0f, 1.0f, 0.0f, 0.0f };

Gl.glLightfv(Gl.GL\_LIGHT0, Gl.GL\_AMBIENT, light0\_dif);

Gl.glMaterialfv(Gl.GL\_FRONT\_AND\_BACK, Gl.GL\_AMBIENT, color\_con);

Gl.glLightfv(Gl.GL\_LIGHT0, Gl.GL\_POSITION, light0\_pos);

Gl.glLoadIdentity();

Gl.glTranslated(0, 0, -15);

Gl.glRotated(alfa, conX, conY, 0);

Gl.glTranslated(0, 0, -8);

Glut.glutSolidCone(1.5, 2, 30, 30);

Gl.glLoadIdentity();

alfa += 3;

conX += 10d;

conY -= 10d;

holst.Invalidate();

}

private void tor()

{

Gl.glEnable(Gl.GL\_ALPHA\_TEST);

Gl.glEnable(Gl.GL\_BLEND);

Gl.glBlendFunc(Gl.GL\_SRC\_COLOR, Gl.GL\_ONE\_MINUS\_SRC\_COLOR);

Gl.glLoadIdentity();

float[] color\_con = { 0.5f, 0, 1, 0.5f };

float[] light0\_dif = { 0.7f, 0.7f, 0.2f, 0.9f };

float[] light0\_pos = { 0.0f, 1.0f, 0.0f, 0.0f };

Gl.glLightfv(Gl.GL\_LIGHT0, Gl.GL\_AMBIENT, light0\_dif);

Gl.glMaterialfv(Gl.GL\_FRONT\_AND\_BACK, Gl.GL\_AMBIENT, color\_con);

Gl.glLightfv(Gl.GL\_LIGHT0, Gl.GL\_POSITION, light0\_pos);

Gl.glLoadIdentity();

Gl.glTranslated(0, 0, -10);

Gl.glRotated(30, 0, 1, 1);

Glut.glutSolidTorus(1, 2, 20, 20);

Gl.glLoadIdentity();

Gl.glDisable(Gl.GL\_ALPHA\_TEST);

Gl.glDisable(Gl.GL\_BLEND);

holst.Invalidate();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void holst\_MouseClick\_1(object sender, MouseEventArgs e)

{

timer1.Start();

timer1.Enabled = true;

}

private void timer1\_Tick(object sender, EventArgs e)

{

osnova();

dodecahedron();

conus();

tor();

holst.Invalidate();

}

}

}

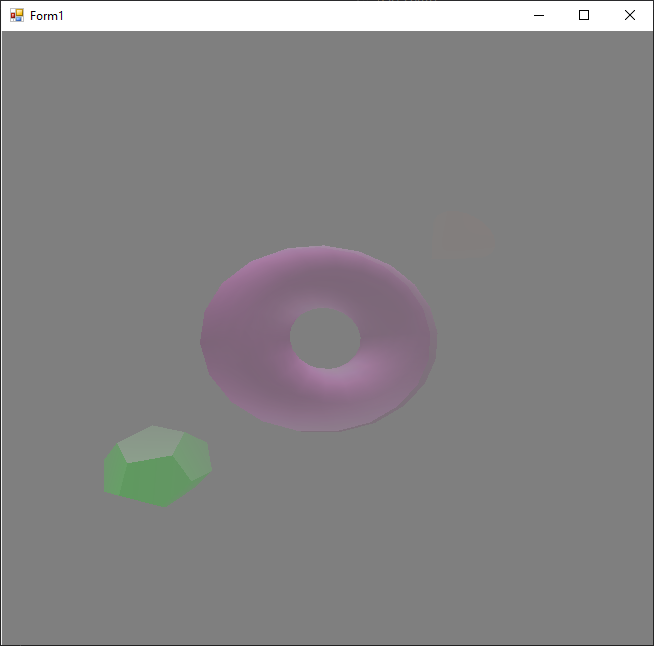


Рис 1

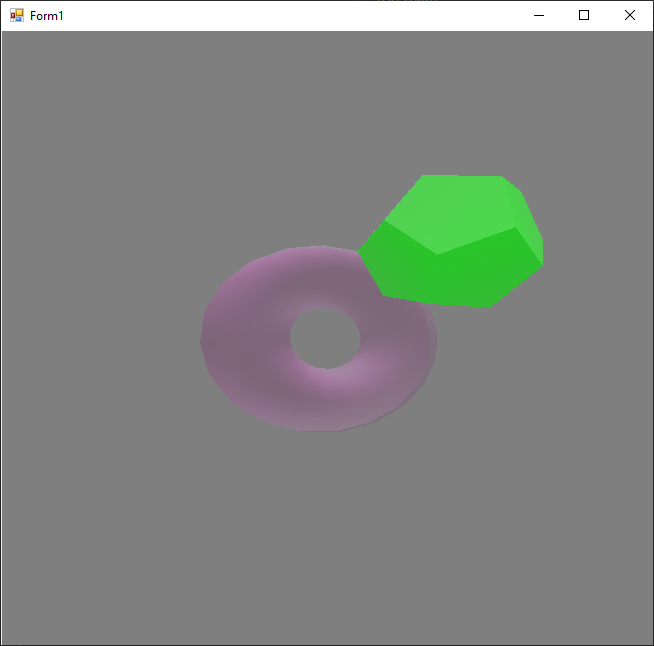


Рис 2

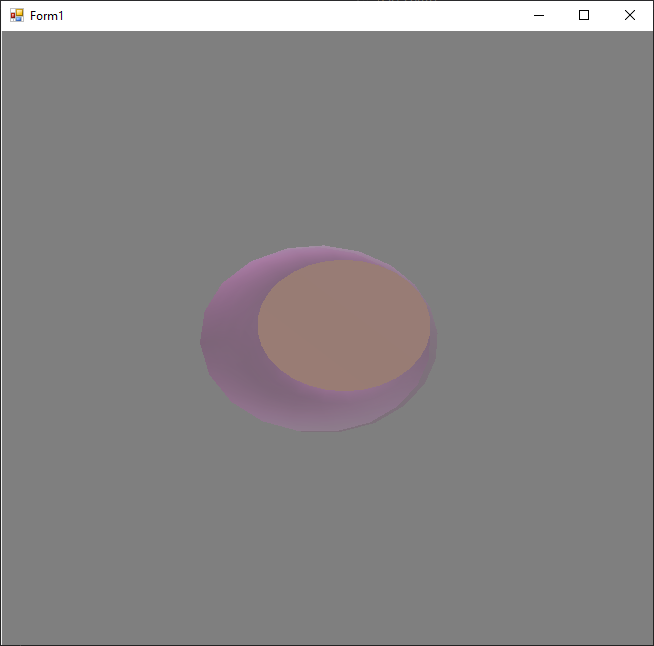


Рис 3

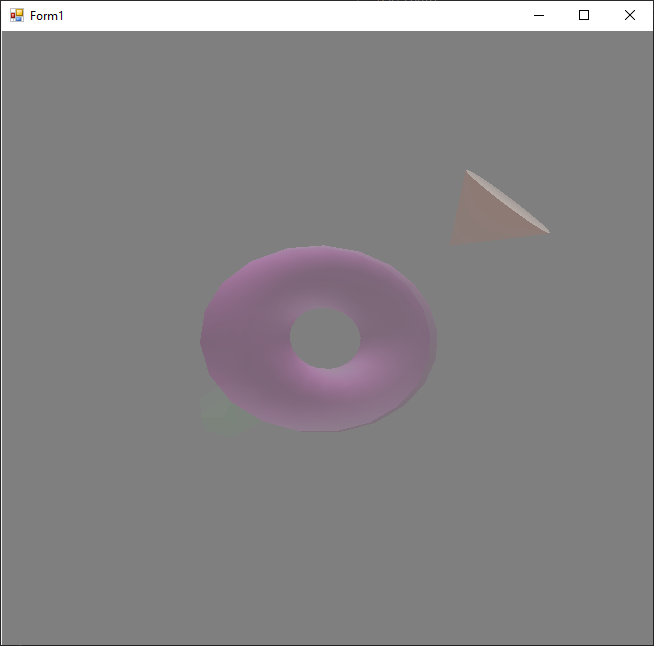


Рис 4

Лабораторная работа 8

Скрипт для движения игрока

using UnityEngine;

using System.Collections;

public class MouseLook : MonoBehaviour

{

public enum RotationAxes

{

MouseXAndY = 0,

MouseX = 1,

MouseY = 2

}

public RotationAxes axes = RotationAxes.MouseXAndY;

public float sensitivityHor = 9.0f;

public float sensitivityVert = 9.0f;

public float minimumVert = -45.0f;

public float maximumVert = 45.0f;

private float \_rotationX = 0;

void Start()

{

Rigidbody body = GetComponent<Rigidbody>();

if (body != null)

body.freezeRotation = true;

}

void Update()

{

if (axes == RotationAxes.MouseX)

{

transform.Rotate(0, Input.GetAxis("Mouse X") \* sensitivityHor, 0);

}

else if (axes == RotationAxes.MouseY)

{

\_rotationX -= Input.GetAxis("Mouse Y") \* sensitivityVert;

\_rotationX = Mathf.Clamp(\_rotationX, minimumVert, maximumVert);

float rotationY = transform.localEulerAngles.y;

transform.localEulerAngles = new Vector3(\_rotationX, rotationY, 0);

}

else

{

\_rotationX -= Input.GetAxis("Mouse Y") \* sensitivityVert;

\_rotationX = Mathf.Clamp(\_rotationX, minimumVert, maximumVert);

float delta = Input.GetAxis("Mouse X") \* sensitivityHor;

float rotationY = transform.localEulerAngles.y + delta;

transform.localEulerAngles = new Vector3(\_rotationX, rotationY, 0);

}

}

}

using UnityEngine;

using System.Collections;

[RequireComponent(typeof(CharacterController))]

[AddComponentMenu("Control Script/FPS Input")]

public class FPSInput : MonoBehaviour

{

public float speed = 6.0f;

public float gravity = -9.8f;

private CharacterController \_charController;

void Start()

{

\_charController = GetComponent<CharacterController>();

}

void Update()

{

float deltaX = Input.GetAxis("Horizontal") \* speed;

float deltaZ = Input.GetAxis("Vertical") \* speed;

Vector3 movement = new Vector3(deltaX, 0, deltaZ);

movement = Vector3.ClampMagnitude(movement, speed);

movement.y = gravity;

movement \*= Time.deltaTime;

movement = transform.TransformDirection(movement);

\_charController.Move(movement);

}

}

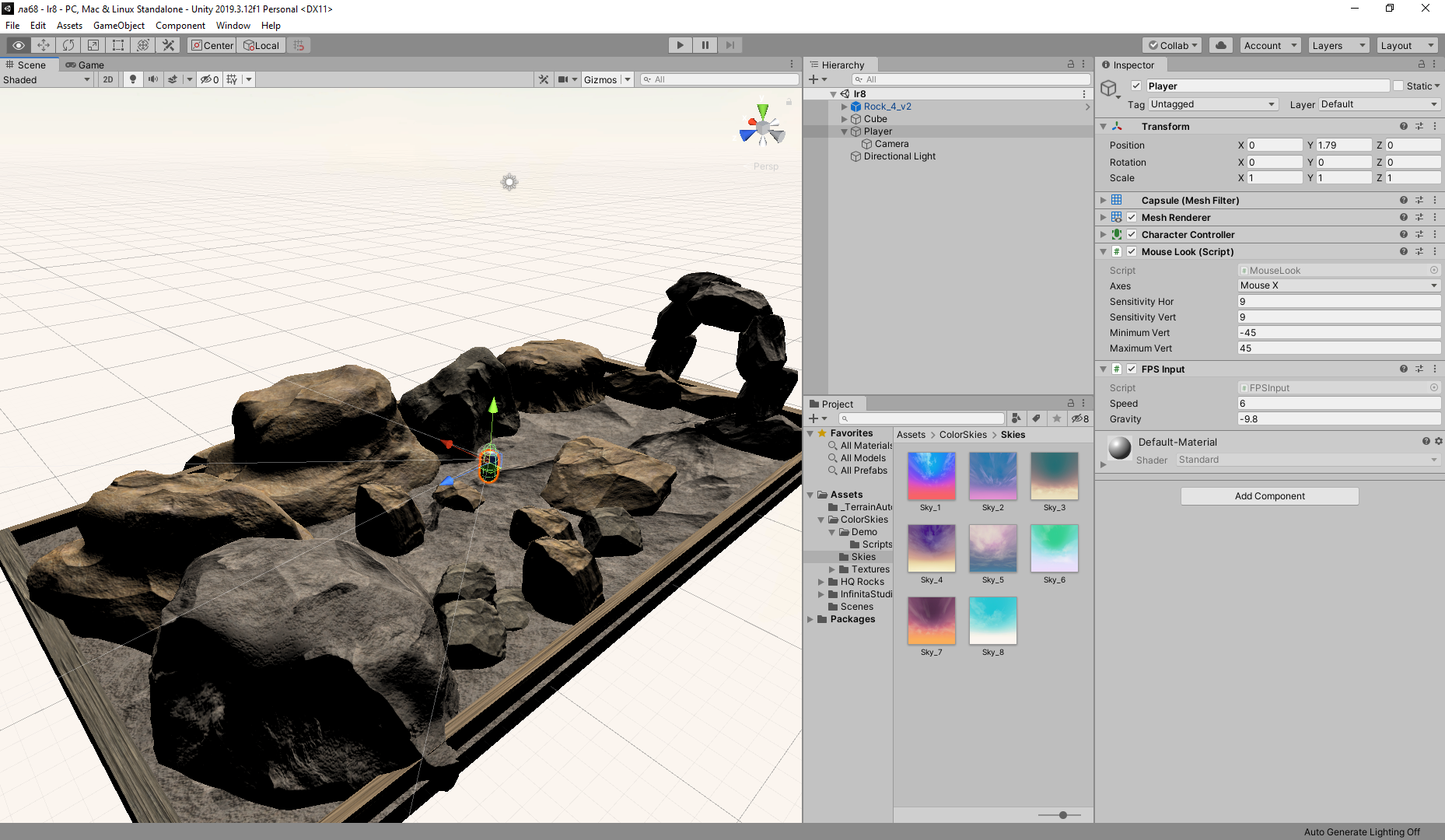


Рис 5РРИсвыа



Рис 6

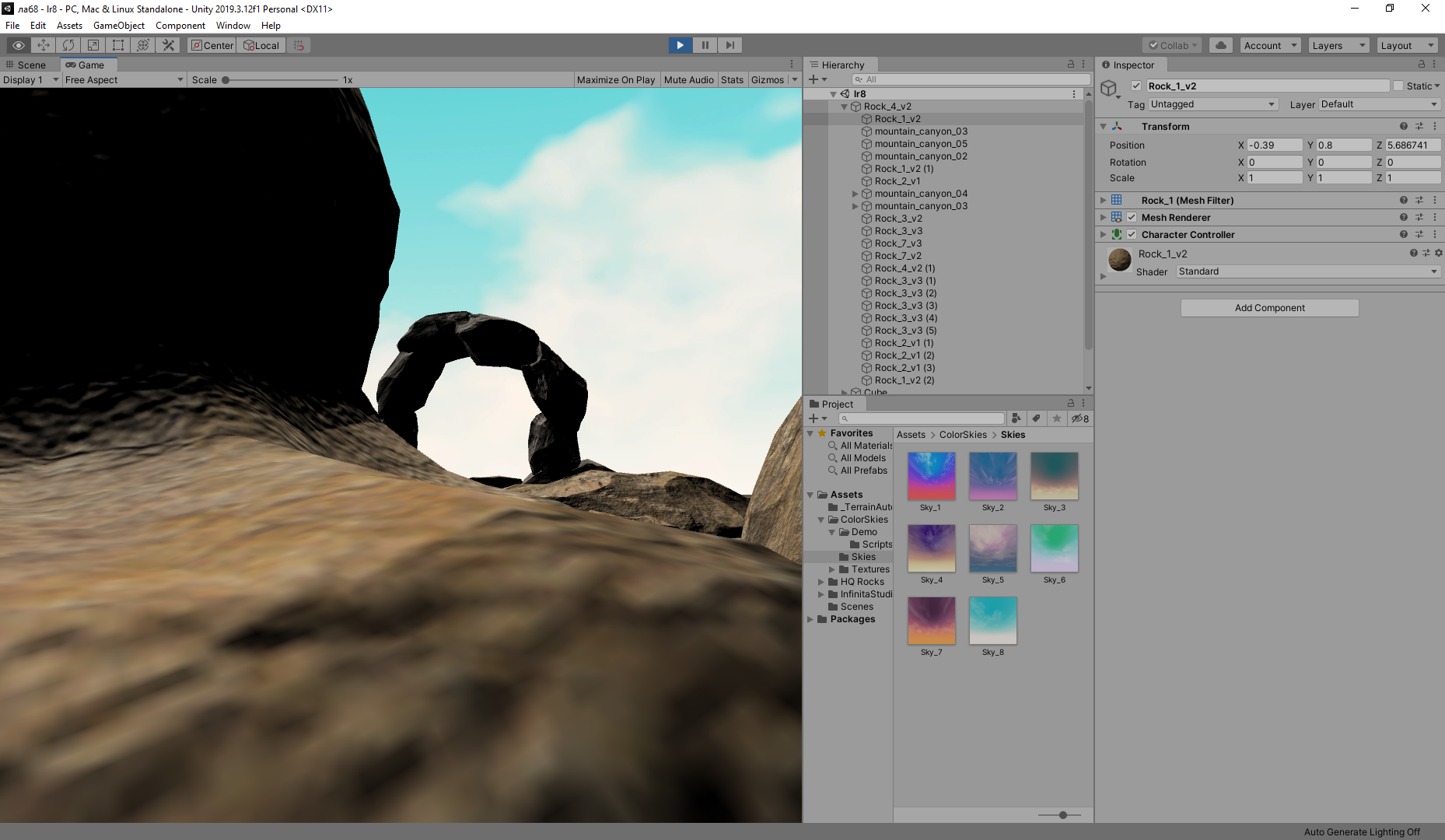


Рис 7

Лабораторная работа 1

