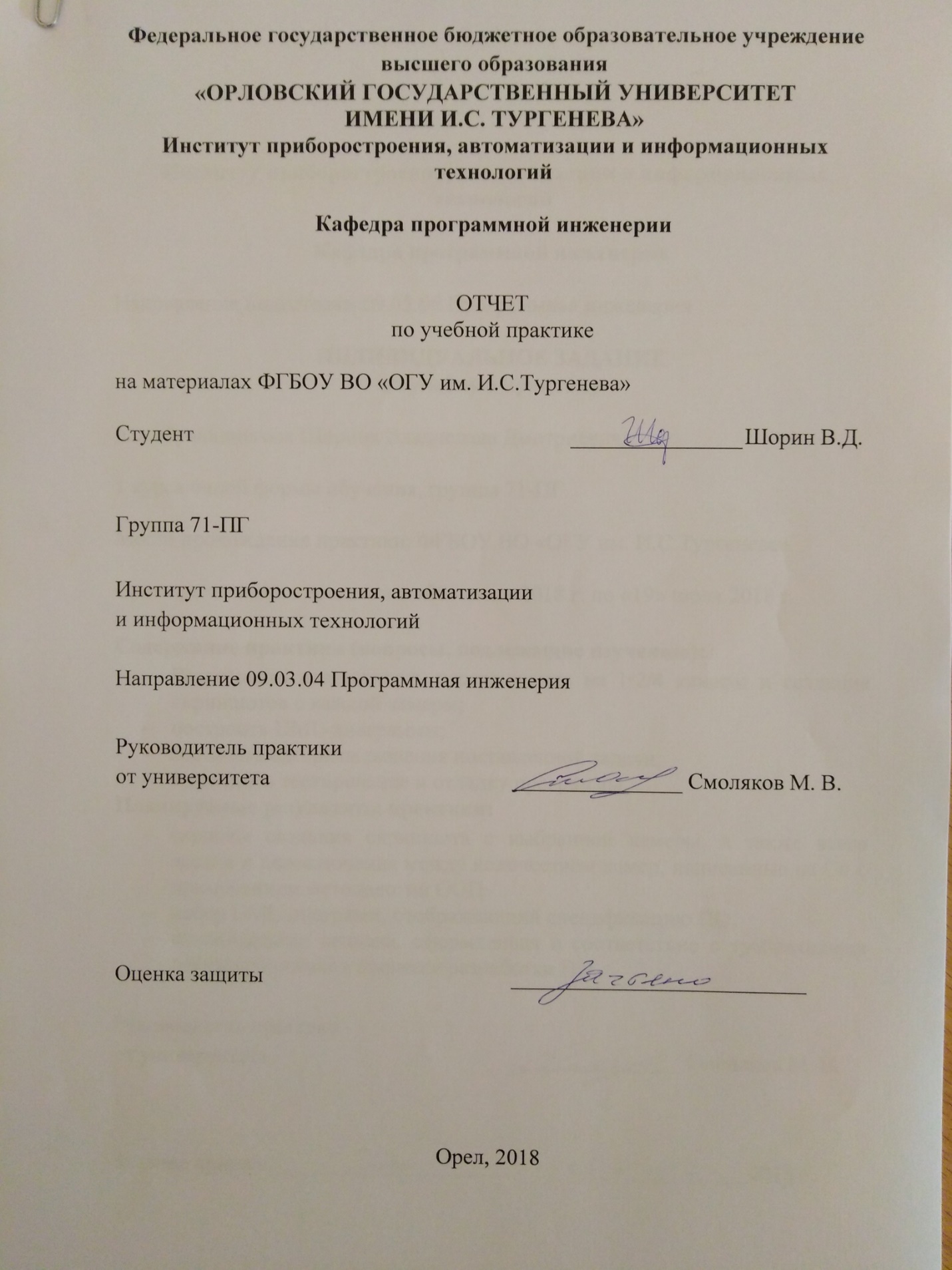
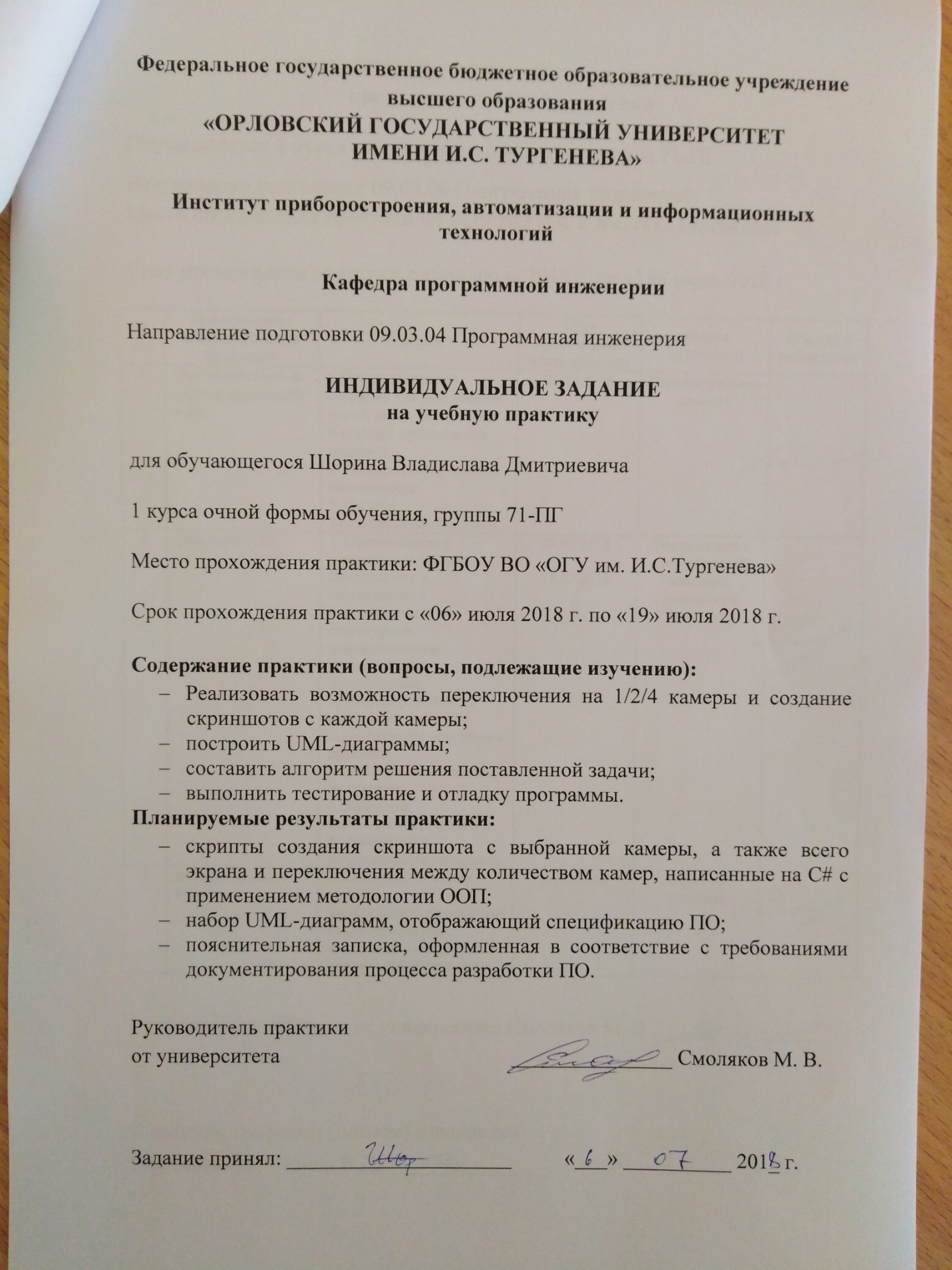
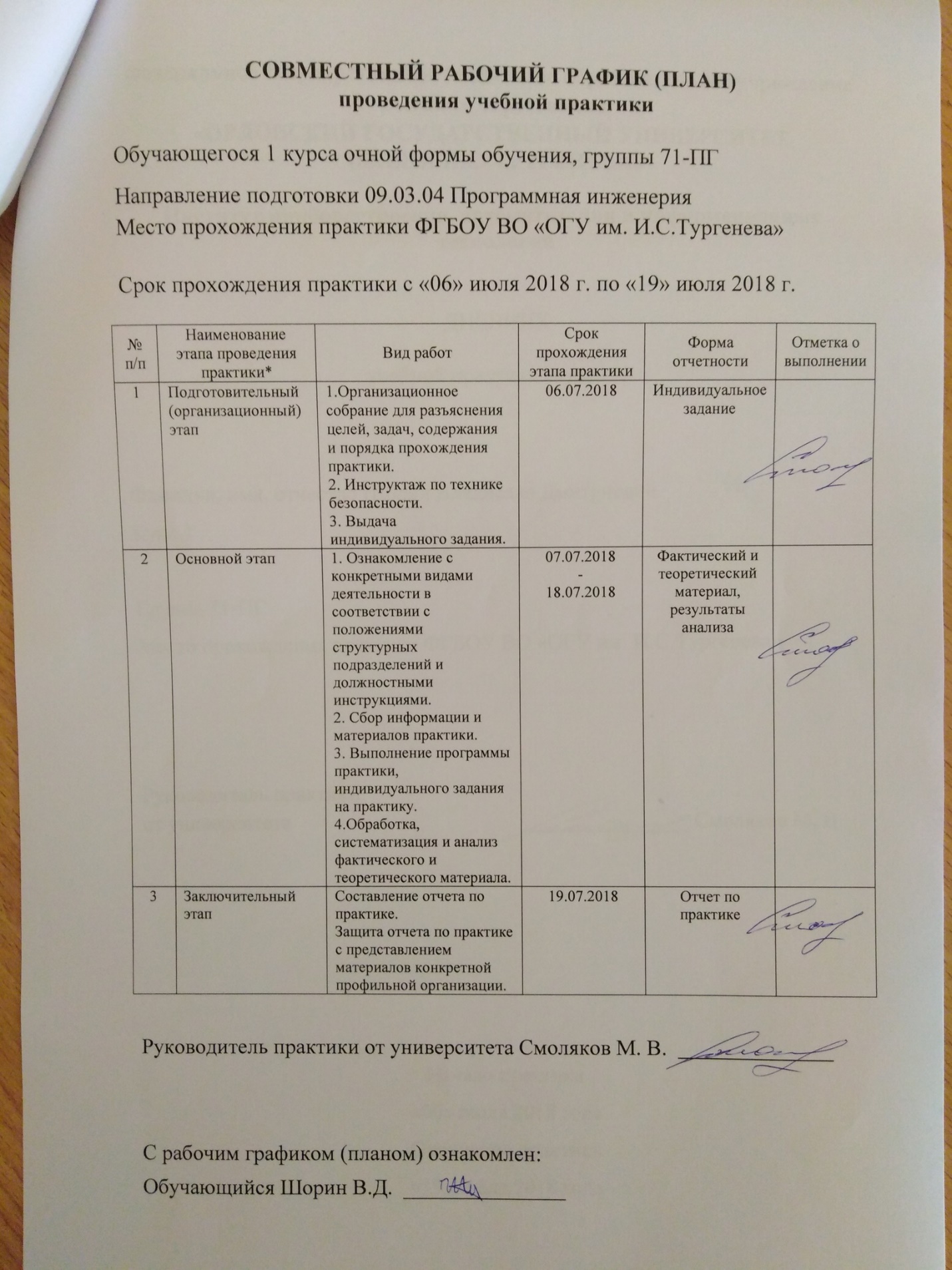
**

**

**

**Содержание**

[Введение 7](#_Toc522970758)

[1 Анализ задачи 8](#_Toc522970759)

[2 Построение диаграмм 9](#_Toc522970760)

[2.1 Диаграмма вариантов использования 9](#_Toc522970761)

[3 Реализация приложения 11](#_Toc522970762)

[Список литературы 15](#_Toc522970763)

[Приложение А(обязательное) Листинги программы 16](#_Toc522970764)

# **Введение**

В ходе прохождения учебной практики мною были получено задание разработки функций переключения между 1/2/4 камерами и создания скриншотов с выбранной камеры, а также всего экрана.

Суть задания заключается в следующем: имеется приложение, в котором необходимо реализовать переключение между 1/2/4 камерами (камеры показывают с разных ракурсов) по нажатию на кнопку. Также, в процессе работы приложения будет возникать необходимость получения изображения с определенной камеры или со всего экрана, что также необходимо реализовать кнопкой.

Целью практики является разработка функций переключения между 1/2/4 камерами и создания скриншотов с выбранной камеры, а также всего экрана.

Задачами, которые необходимо выполнить для достижения поставленной цели, являются:

1. анализ поставленной задачи;
2. разработка набора UML-диаграмм;
3. разработка алгоритма решения поставленной задачи;
4. тестирование и отладка функций.

# **1 Анализ задачи**

Задача: разработка функций переключения между 1/2/4 камерами и создания скриншотов с выбранной камеры, а также всего экрана.

Для выполнения поставленной задачи будет использоваться методология объектно-ориентированного программирования (ООП), так как реализация данных скриптов значительно упрощается при создании классов, что и позволяет сделать ООП[2]. Для написания программы будет использоваться язык С#[3], поддерживающий методологию ООП.

При анализе функциональных и структурных требований будут построены UML-диаграммы, которые позволяют наглядно изобразить функционал и состав системы. Будет разработана диаграмма вариантов использования, отображающая возможные действия в системе.

# **2 Построение диаграмм**

Для построения проекта программного обеспечения (ПО) необходимо разработать последовательный набор UML-диаграмм, каждая из которых будет отображать некоторую особенность спецификации ПО.

# **2.1 Диаграмма вариантов использования**

Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Разрабатывается данная диаграмма для:

• Определения общих границ и контекста моделируемой предметной области на начальных этапах проектирования;

• Формулировки общих требований к функциональному поведению проектируемой системы.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования[1].

На Рисунке 1 изображена диаграмма вариантов использования, построенная для решения поставленной задачи.

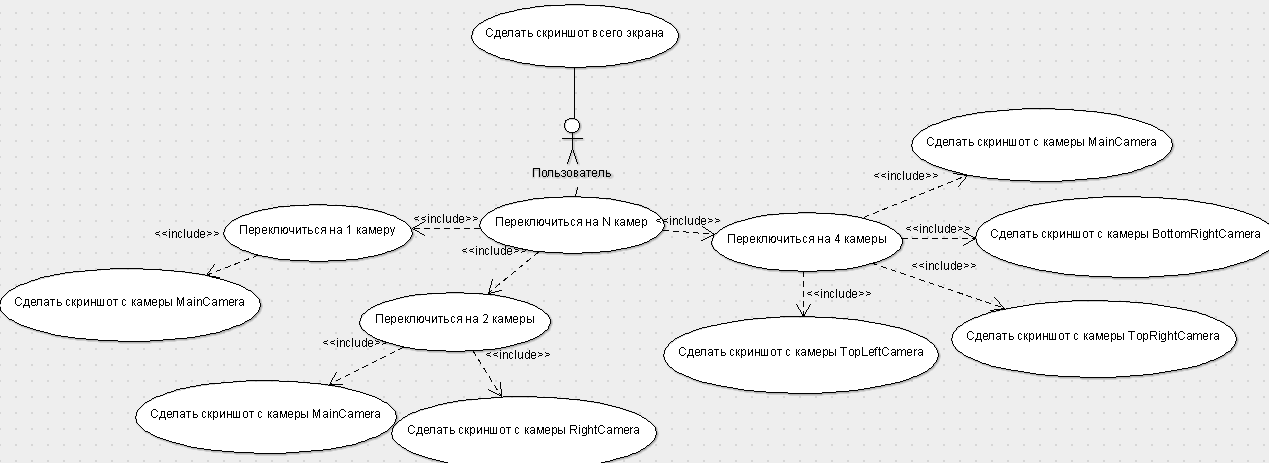


Рисунок 1 – Диаграмма вариантов использования

Пользователь, войдя в приложение, может переключиться на 1/2/4 камеры и, в зависимости от количества текущих камер, может сделать скриншот с любой из них. Также, он в любой момент может сделать скриншот всего экрана.

# **3 Реализация приложения**

На рисунке:

2 – изображено состояние экрана при 1 камере (нажатие на кнопку 1);

3 – изображено состояние экрана при 2 камерах (нажатие на кнопку 2);

4 – изображено состояние экрана при 4 камерах (нажатие на кнопку 3);

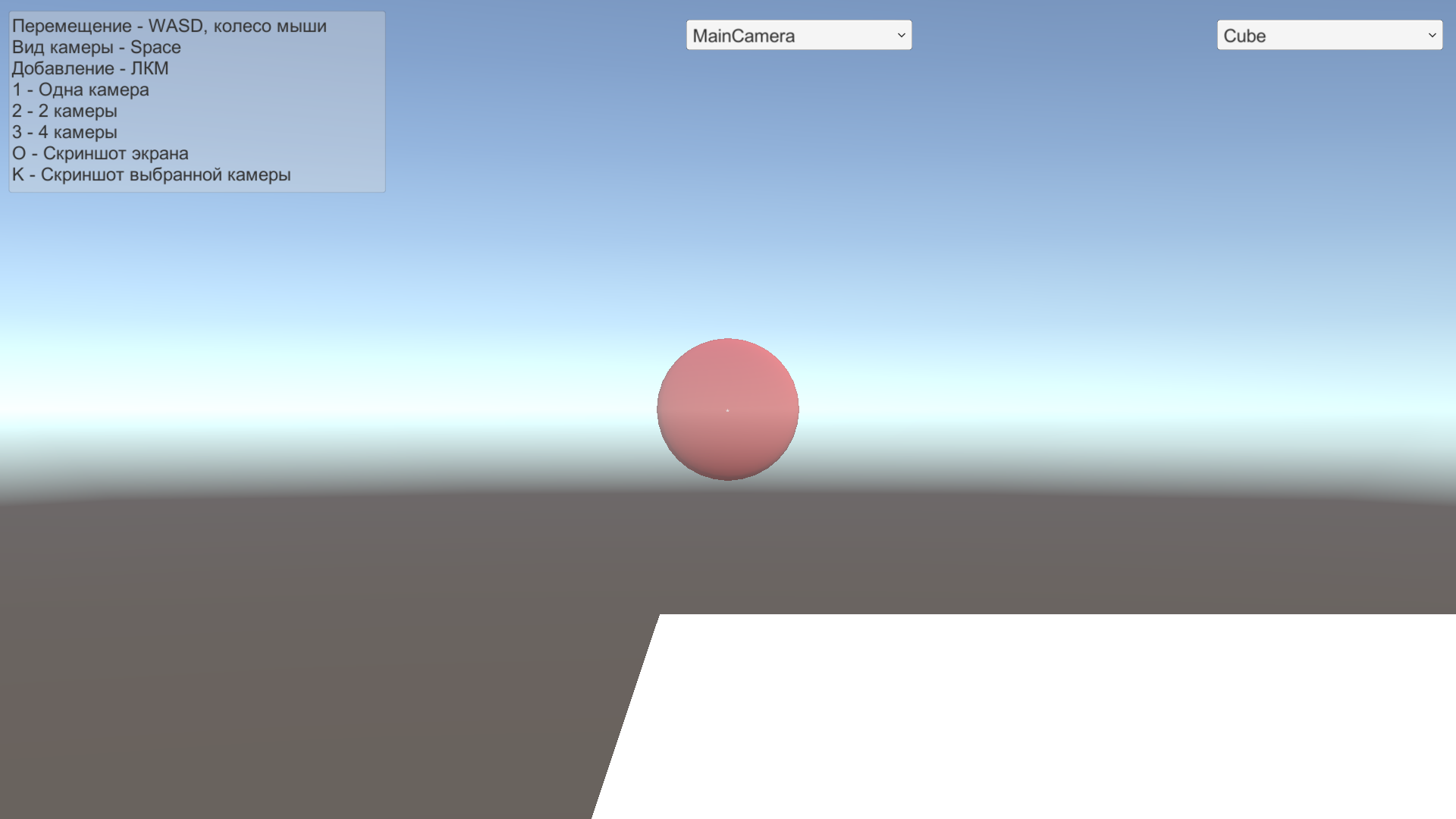


Рисунок 2 - Состояние экрана при 1 камере

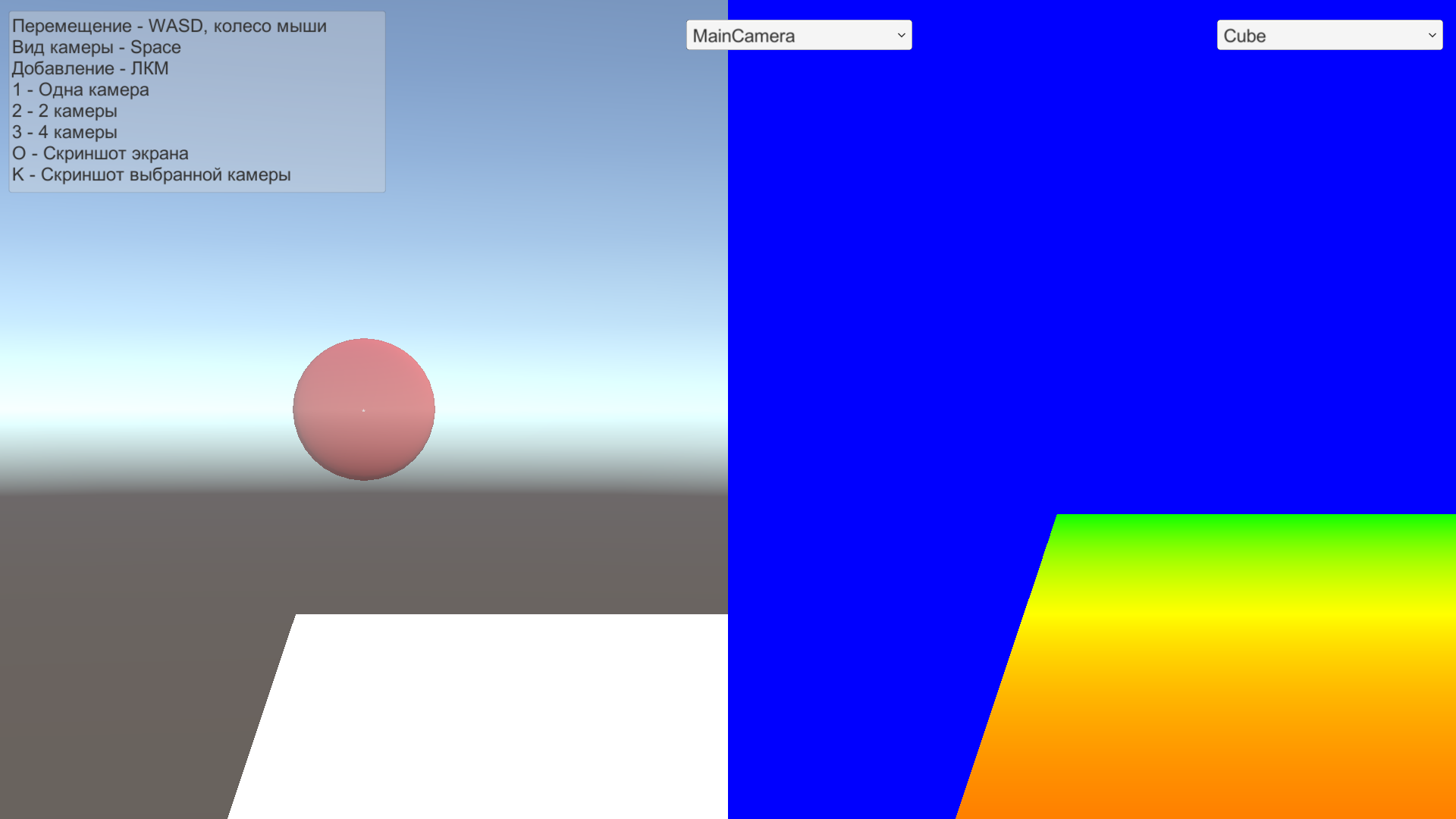


Рисунок 3 - состояние экрана при 2 камерах

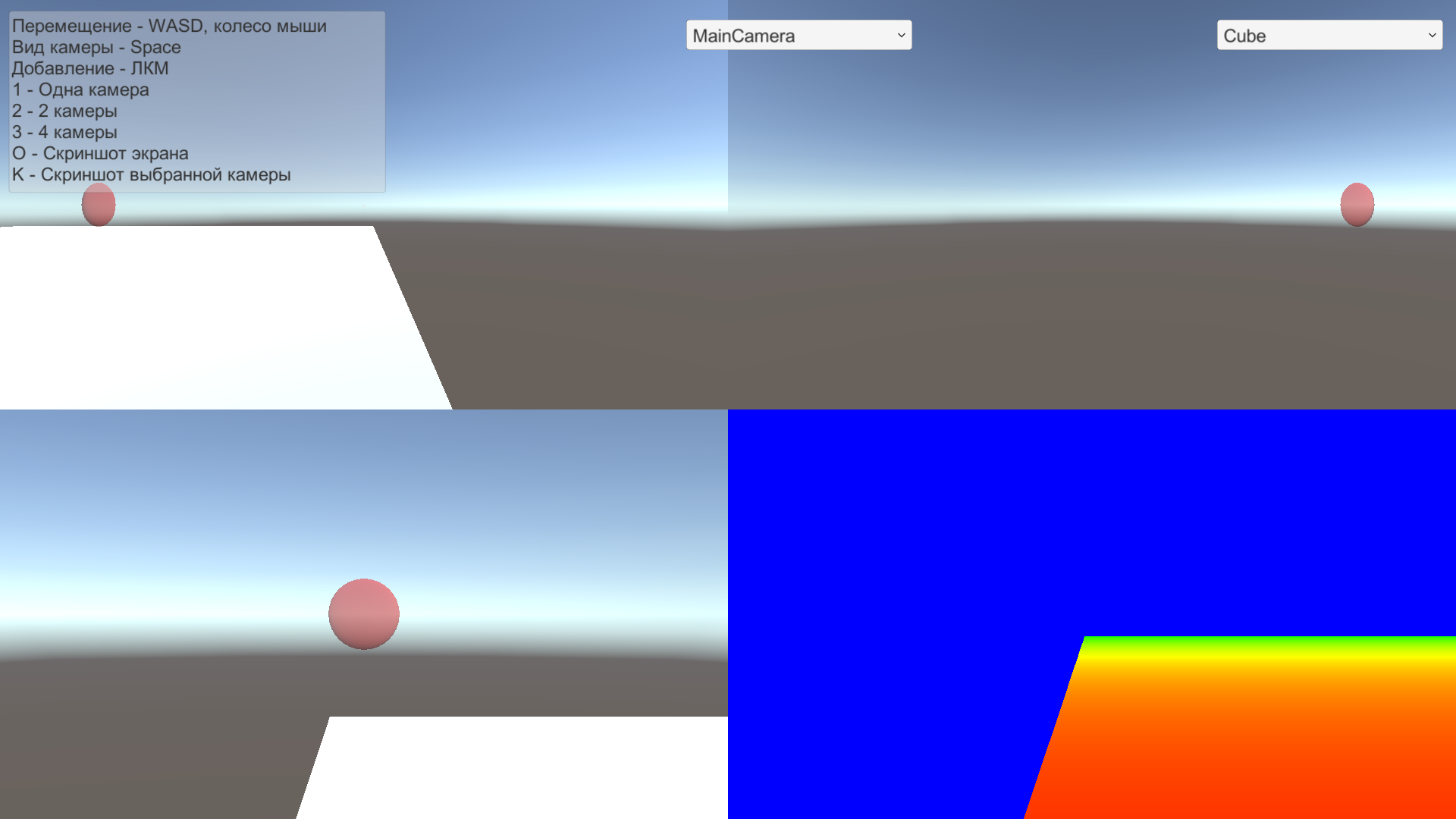


Рисунок 4 - состояние экрана при 4 камерах

На каждом рисунке можно увидеть выпадающий список активных в данный момент камер, с которых можно сделать скриншот.

Также, в любой момент можно сделать скриншот всего экрана по нажатию на кнопку «О».

Таким образом, программа полностью удовлетворяет условиям поставленной задачи, поэтому цель практики можно считать достигнутой.

Листинги программы на языке С# приведены в Приложении А.

# **Заключение**

Поскольку анализ поставленной задачи выполнен, диаграммы языка UML построены, алгоритм решения поставленной задачи составлен и реализован в виде программного кода на языке С#, программа протестирована и отлажена, а также соответствует поставленной цели, поставленную цель практики можно считать достигнутой.

В ходе прохождения практики были закреплены знания по объектно- ориентированной методологии программирования, закреплены навыки разработки проекта ПО как набора последовательных UML-диаграмм, разработан алгоритм решения поставленной задачи и реализован в виде кода на языке программирования С#.

# **Список литературы**

1. Леоненков А.В. Самоучитель UML. – 2-е изд., перераб. и доп.[Текст] / А.В. Леоненков – СПб.: БХВ-Петербург, 2004. – 432 с.
2. Радченко, Г.И. Объектно-ориентированное программирование [Электронный ресурс] / Г.И. Радченко, Е.А. Захаров. – Челябинск: Издательский центр ЮУрГУ, 2013. – 167 с. – Режим доступа: http://glebradchenko.susu.ru/courses/bachelor/oop/OOP-PrePrint.pdf (дата обращения 10.07.2018).
3. Джон Скит. C# для профессионалов: тонкости программирования, 3-е издание, новый перевод = C# in Depth, 3rd ed.. — М.: «Вильямс», 2014. — 608 с.

# **Приложение А(обязательное)**

# **Листинги программы**

**CameraCount.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CameraCount : MonoBehaviour {

public Camera MainCamera;

public Camera RightCamera;

public Camera BottomRightCamera;

public Camera TopLeftCamera;

public Camera TopRightCamera;

void Update () {

if (Input.GetKeyDown(KeyCode.Alpha1)) {OneCamera(); }

if (Input.GetKeyDown(KeyCode.Alpha2)) {TwoCameras();}

if (Input.GetKeyDown(KeyCode.Alpha3)) {FourCameras();}

}

public void OneCamera() {

MainCamera.rect = new Rect(0, 0, 1.0f, 1.0f);

RightCamera.gameObject.SetActive(false);

BottomRightCamera.gameObject.SetActive(false);

TopLeftCamera.gameObject.SetActive(false);

TopRightCamera.gameObject.SetActive(false);

}

public void TwoCameras() {

MainCamera.rect = new Rect(0, 0, 0.5f, 1.0f);

RightCamera.gameObject.SetActive(true);

BottomRightCamera.gameObject.SetActive(false);

TopLeftCamera.gameObject.SetActive(false);

TopRightCamera.gameObject.SetActive(false);

}

public void FourCameras() {

MainCamera.rect = new Rect(0, 0, 0.5f, 0.5f);

BottomRightCamera.gameObject.SetActive(true);

TopLeftCamera.gameObject.SetActive(true);

TopRightCamera.gameObject.SetActive(true);

RightCamera.gameObject.SetActive(false);

}

}

**SubCameraScreen.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class SubCameraScreen : MonoBehaviour {

public int resWidth = 2550;

public int resHeight = 3300;

public float saveTime = 1.0f;

private float curTime;

public bool AutoScreen = false;

private bool takeHiResShot = false;

public string cameraName;

private DropdownCameraSelector Selector;

private void Awake()Selector = FindObjectOfType<DropdownCameraSelector>(); }

void OnEnable() {

Debug.Log("en");

Selector.cameras.Add(this);

Selector.ResetDropdownOptions ();

}

void OnDisable(){Selector.cameras.Remove (this);Selector.ResetDropdownOptions ();

}

private void Update() {

if (AutoScreen == true){

curTime -= Time.deltaTime;

if (curTime <= 0) {

TakeScreenShot();

Debug.Log("AutoSaveScreen");

curTime = saveTime;

}

}

}

public static string ScreenShotName(int width, int height, string name)

{

return string.Format("C:/Users/Lenovo/Desktop/Project exe/Screenshots/{0}/screen\_{1}x{2}\_{3}\_{0}.png",

name,

width, height,

System.DateTime.Now.ToString("yyyy-MM-dd\_HH-mm-ss"));

}

public void TakeHiResShot(){ takeHiResShot = true;}

void LateUpdate() {

takeHiResShot |= Input.GetKeyDown(KeyCode.Keypad1);

if (takeHiResShot) {TakeScreenShot();takeHiResShot = false; }

}

public void TakeScreenShot() {

RenderTexture rt = new RenderTexture(resWidth, resHeight, 24);

GetComponent<Camera>().targetTexture = rt;

Texture2D screenShot = new Texture2D(resWidth, resHeight, TextureFormat.RGB24, false);

GetComponent<Camera>().Render();

RenderTexture.active = rt;

screenShot.ReadPixels(new Rect(0, 0, resWidth, resHeight), 0, 0);

GetComponent<Camera>().targetTexture = null;

RenderTexture.active = null;

Destroy(rt);

byte[] bytes = screenShot.EncodeToPNG();

string filename = ScreenShotName(resWidth, resHeight, cameraName);

System.IO.File.WriteAllBytes(filename, bytes);

Debug.Log(string.Format("Took screenshot to: {0}", filename));

}

}