Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

Работа выполнена в СКБ «Интеллектуальные технологии»

СОГЛАСОВАНО

Начальник отдела ОНиПКРС

_____ Е.М. Димитриади
(подпись)

«21» 06 2023 г.

Декан

_____И.А. Трещёв
(подпись)

«21» 06 2023 г.

УТВЕРЖДАЮ

Проректор по научной работе

_____А.В. Космынин
(подпись)

«21» 06 2023 г.

«Разработка программного обеспечения для системы проецирования стенда изготовления жгутов»

Комплект *проектной* документации

Руководитель СКБ _____ 21.06.2023 *Г.В. Москалец*
(подпись, дата)

Руководитель проекта _____ 21.06.2023 *А.Н Петрова*
(подпись, дата)

Комсомольск-на-Амуре 2023

Карточка проекта

Название	Разработка программного обеспечения для системы проецирования стенда изготовления жгутов
Тип проекта	Тип проекта: техническое творчество (инициативный)
Исполнители	Студент _____ С.В. Якимова – 9ВТб-1
Срок реализации	31.10.2022-10.06.2023

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ЗАДАНИЕ
на разработку

Название проекта: «Разработка программного обеспечения для системы проецирования стенда изготовления жгутов»

Назначение: Автоматизация процесса сборки электрических кабельных жгутов, для сокращения времени, затрачиваемое на выполнение всех технологических операций, разделённых на этапы, а также уменьшение вероятности возникновения ошибок.

Область использования: ПО может использоваться в автомобильная промышленности для оптимизации сборки проводки в автомобилях, в аэрокосмической промышленности точного размещения и сборки проводов, обеспечивая надежное соединение и соответствие техническим требованиям, может быть применено при производстве различных электронных устройств и систем, где требуется сборка сложных кабельных жгутов. Программа поможет оптимизировать процесс сборки, улучшить точность и снизить количество ошибок.

Функциональное описание проекта: Программное обеспечение позволяет проецировать необходимые выбранные схемы жгутов на сборочный стенд упрощающий процесс сборки изделия. В данном программном обеспечении были реализованы следующие функции: проверка правильности вывода проекции с помощью квадратной сетки, параметры которой задаются в приложении, загрузка схем жгута в программное обеспечение, вывод списка схем жгута, выбор определённых схем жгута для их проецирования, расстановка точек в режиме позиционирования для определения плоскости, на которую будет производиться проецирование схем, сохранение и загрузка данных в файлы формата json.

Техническое описание устройства: Файл «SchemeView.cs» содержит класс для размещения выбранной из списка схемы на объекте сцены во время выполнения программы.Файл «Config.cs» содержит класс, записывающий и считывающий данные из json-файлов настроек проекта.Файл «ConfigData.cs»

содержит метод для генерации и присвоения уникальных идентификаторов объектам сцены, записываемых в файлы настроек проекта, а также метод (свойство) для доступа к этим идентификаторам, необходимый для файла «Config.cs».Файл «Scheme.cs» является классом хранения данных для объекта сцены, отвечающего за отображение схем.Файл «TrackerInfo.cs» является классом хранения массива данных, передаваемых трекерами.Файл «NetTracker.cs» содержит вложенный класс udp-сервера, принимающего и обрабатывающего данные от трекеров. Сам же класс позволяет проверять состояние кнопки, устанавливающей точки в режиме позиционирования, а также преобразовывать местоположение объектов сцены, на которых установлен скрипт.Файл «MessageBox.cs» содержит методы и класс для отображения всплывающих окон приложения.Файл «Messages.cs» содержит метод для вызова всплывающего окна, появляющегося на несколько секунд. Скрипт создан для удобства обрабатывания сообщений при взаимодействии с пунктами меню.Файл «ToggleData.cs» содержит публичные поля, необходимые для работы переключателей в меню выбора найденных схем.Файл «ToggleList.cs» позволяет выводить список схем при нажатии на кнопку «Схема» главного меню.Файл «UIView.cs» содержит методы для раскрытия или закрытия главного меню, а также активации меню списка схем.

Требования: Для разработки программного обеспечения по проецированию схем жгутов необходимы следующие технические средства:

- процессор с архитектурой x64;
- оперативная память 8 Гб и больше;
- жесткий диск 128 Гб и больше;
- встроенная видеокарта;
- монитор;
- клавиатура, манипулятор «мышь»;
- два контроллера ViveTracker 2.0 и базовая станция, идущая в комплекте;
- проектор.

План работ:

Наименование работ	Срок
Изучение предметной области	*11.2022*
Изучение и анализ аналогичного ПО	*11.2022*
Разработка требований к ПО	*12.2023*
Проектирование приложения	*03.2023*
Разработка приложения	*05.2023*
Тестирование и отладка	*06.2023*

Комментарии:

Перечень графического материала:

1. Листинги;
2. Изображения;

Руководитель проекта _____ *Г.В. Москалец*

(подпись, дата)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ПАСПОРТ

«Разработка программного обеспечения для системы проецирования стенда изготовления жгутов»

Руководитель проекта _____ *А.Н. Петрова*

(подпись, дата)

Комсомольск-на-Амуре 2023

Содержание

Общие положения

Настоящий паспорт является документом, предназначенным для ознакомления с основными техническими характеристиками, устройством, правилами установки и эксплуатации устройства «Разработка программного обеспечения для системы проецирования стенда изготовления жгутов» (далее «изделие»).

Паспорт входит в комплект поставки изделия. Прежде, чем пользоваться изделием, внимательно изучите правила обращения и порядок работы с ним. В связи с постоянной работой по усовершенствованию изделия, повышающей его надежность и улучшающей условия эксплуатации, в конструкцию могут быть внесены изменения, не отраженные в данном издании.

1.1 Наименование изделия

Полное наименование изделия – «Разработка программного обеспечения для системы проецирования стенда изготовления жгутов».

1.2 Наименования документов, на основании которых ведется проектирование изделия

Проектирвание «Разработка программного обеспечения для системы проецирования стенда изготовления жгутов» осуществляется на основании требований и положений следующих документов:

- задание на разработку;
- календарный план-график выполнения этапов работы.

1.3 Перечень организаций, участвующих в разработке изделия

Заказчиком проекта «Разработка программного обеспечения для системы проецирования стенда изготовления жгутов» является Федеральное государственное бюджетное образовательное учреждение высшего образования «Комсомольский-на-Амуре государственный университет» (далее заказчик), находящийся по адресу: 681013, Хабаровский край, г. Комсомольск-на-Амуре, Ленина пр-кт., д. 17.

Исполнителями проекта «Разработка программного обеспечения для системы проецирования стенда изготовления жгутов» являются участники студенческого конструкторского бюро «Интеллектуальные технологии», студенты групп 0ВТб-1 Покровский Виктор Владимирович, 0ИБ-1 Монастырная Елизавета Игоревна.

1.4 Сведения об использованных при проектировании нормативно-технических документах

При проектировании использованы следующие нормативно-технические документы:

ГОСТ 2.001-2013. Единая система конструкторской документации. Общие положения.

ГОСТ 2.102-2013. Единая система конструкторской документации. Виды и комплектность конструкторских документов.

ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам.

ГОСТ 2.610-2006. Единая система конструкторской документации. Правила выполнения эксплуатационных документов.

ГОСТ 2.004-88. Единая система конструкторской документации. Общие требования к выполнению конструкторских технологических документов на печатающих и графических устройствах вывода ЭВМ.

ГОСТ 2.051-2006. Единая система конструкторской документации. Электронные документы. Общие положения.

ГОСТ 2.052-2006. Единая система конструкторской документации. Электронная модель изделия. Общие положения.

ГОСТ 2.601-2013. Единая система конструкторской документации. Эксплуатационные документы.

2 Назначение и принцип действия

2.1 Назначение изделия

Разработка программного обеспечения для системы проецирования стенда изготовления жгутов

В состав изделия входят:

- Паспорт,
- Программная реализация.

2.2 Области использования изделия

Данное программное обеспечение может использоваться в автомобильная промышленности для оптимизации сборки проводки в автомобилях, в аэрокосмической промышленности точного размещения и сборки проводов, обеспечивая надежное соединение и соответствие техническим требованиям, может быть применено при производстве различных электронных устройств и систем, где требуется сборка сложных кабельных жгутов. Программа поможет оптимизировать процесс сборки, улучшить точность и снизить количество ошибок.

2.3 Принцип действия изделия

Программное обеспечение позволяет проецировать необходимые выбранные схемы жгутов на сборочный стенд упрощающий процесс сборки изделия. В данном программном обеспечении были реализованы следующие функции: проверка правильности вывода проекции с помощью квадратной сетки, параметры которой задаются в приложении, загрузка схем жгута в программное обеспечение, вывод списка схем жгута, выбор определённых схем жгута для их проецирования, расстановка точек в режиме позиционирования для определения плоскости, на которую будет производиться проецирование схем, сохранение и загрузка данных в файлы формата json.

3 Состав изделия и комплектность

В комплект поставки входит:

- Паспорт,
- Программная реализация.

4 Устройство и описание работы изделия

4.1 Описание работы изделия

В ходе разработки программного обеспечения были созданы две сцены: симуляция всех процессов в среде Unity3D и итоговая сцена для работы с реальным оборудованием – с трекерами и базовой станцией. При запуске приложения на сцене находятся стол и проектор, имитирующие реальную комнату. За пределами «комнаты» находится камера, которая направлена на объект, отвечающий за отображение схем, по умолчанию на нём установлено изображение квадратной чёрно-белой сетки. Камера и объект «схема» сохранены в сцене для работы с реальным оборудованием.

После запуска приложения появляется возможность взаимодействия с раскрывающимся главным меню.

С помощью кнопки «Сохранить» данные из проекта сохраняются в файлах формата json. При нажатии на кнопку «Загрузить» данные из json-файлов переносятся в программное обеспечение. Основные настройки для проецирования представлены тремя первыми пунктами меню. Кнопка «Установить» переводит режим работы программы в позиционирование. На сцене симуляции с помощью мыши становится возможным установить три точки для определения плоскости, на которую будет проводиться проецирование схем. На сцене работы с реальным оборудованием в этом режиме тоже ставятся три точки, только с помощью нажатия на кнопку, прикреплённую к трекеру.

При нажатии на кнопку «Схема» открывается дополнительное меню, позволяющее выбрать настройки проецируемой квадратной сетки.Этот режим работы позволяет проверить правильность калибровки проектора, отображающего схемы.

Кнопка «Схема» открывает дополнительное меню со списком найденных схем. После выбора схемы из списка изображение с диска переносится в программу и устанавливается на объекте «схема» как главная

текстура. Квадратная сетка также устанавливается как главная текстура на этот объект при выборе соответствующего пункта меню.

5 Условия эксплуатации

Условия эксплуатации программного обеспечения для проецирования схем определяются условиями эксплуатации тех технических средств, на которых запускается программное обеспечение. Ограничений по времени эксплуатации программное обеспечение для системы проецирования стенда изготовления жгутов не имеет.

Для безотказной работы программного обеспечения необходимо произвести ряд действий:

- во-первых, расположить схемы жгутов и настройки программного обеспечения в файлах формата json по определённому пути;
- во-вторых, провести калибровку проекторов и осуществить контроль калибровки с помощью проецирования квадратной сетки в масштабе 1:1;
- в-третьих, спозиционировать и зафиксировать схемы жгутов на плазе.

5.1 Правила и особенности размещения изделия

Изделие должно быть расположено на расстоянии не менее 1 м от нагревательных приборов.

ВНИМАНИЕ! При эксплуатации изделия запрещается проводить самостоятельно какие-то либо работы по извлечению и установке внутренних компонентов изделия.

5.2 Меры безопасности

Необходимо соблюдать требования техники безопасности и следующие меры предосторожности:

- Диагностирование неисправностей программного обеспечения должно производиться при соблюдении условий эксплуатации.
- Требования безопасности при эксплуатации и обслуживании программного обеспечения «Стенд проецирования жгутов» устанавливаются в соответствии с СанПиН 2.2.2.542-96. Гигиенические требования к видеодисплейным терминалам, персональным электронно-вычислительным машинам и организации работы, введенным 14.07.96.

5.3 Правила хранения и транспортирования

Для хранения программного обеспечения «Стенд проецирования жгутов» применяются любые носители ёмкостью не менее 300 Мб.

Требования к транспортированию и хранению определяются соответствующими характеристиками транспортирования и хранения для выбранного типа носителя данных и используемого компьютерного оборудования.

ПРИЛОЖЕНИЕ А

(обязательное)

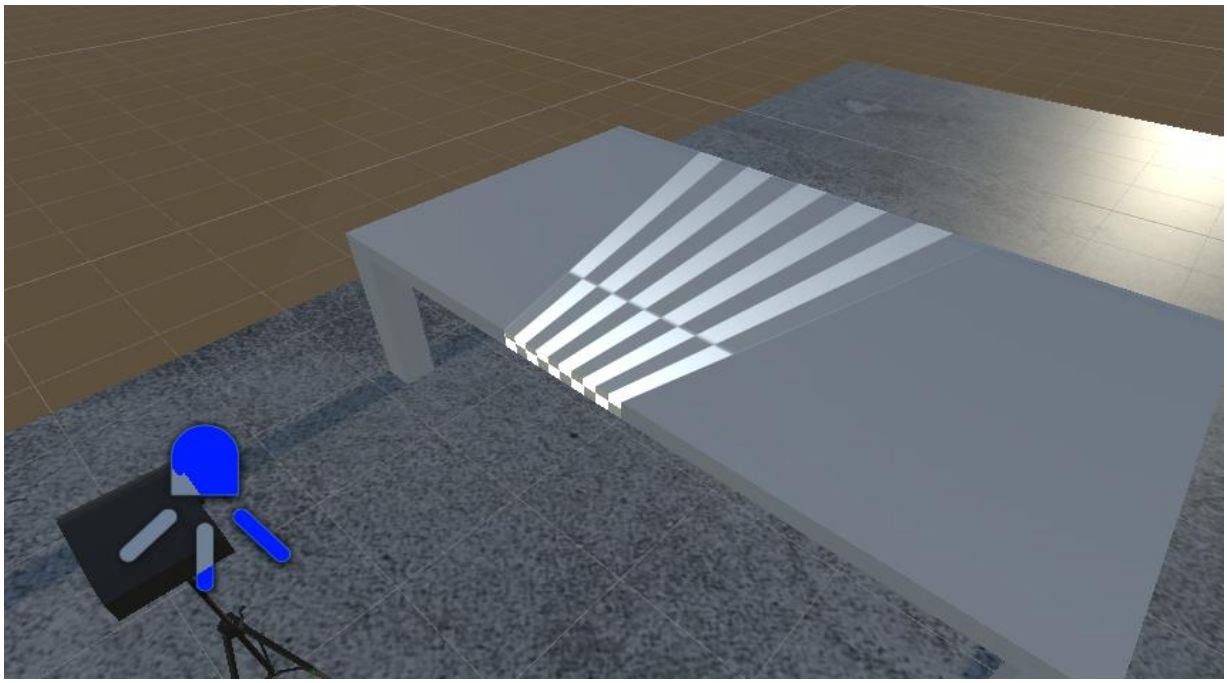

Рисунок А.1 – Проектор и стол на сцене симуляции

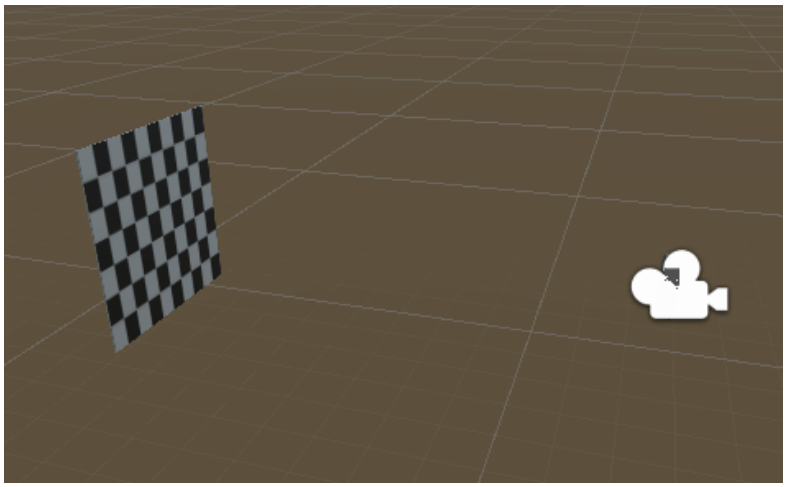

Рисунок А.2 – Камера, направленная на объект, отвечающий за схемы

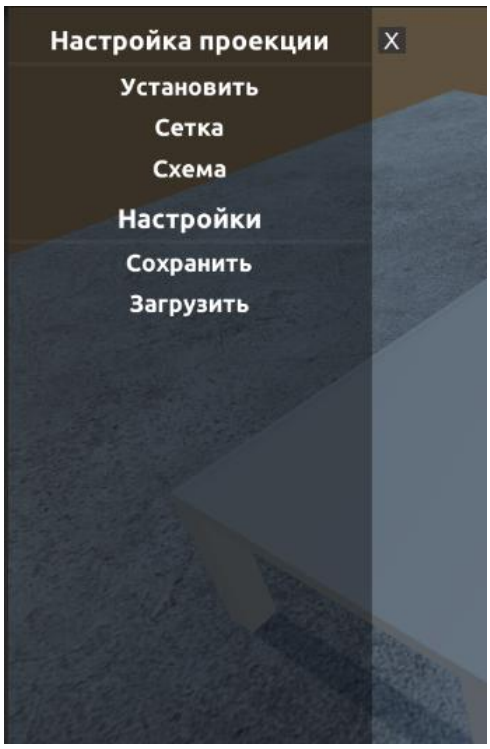

Рисунок А.3 – Главное меню для работы с приложением

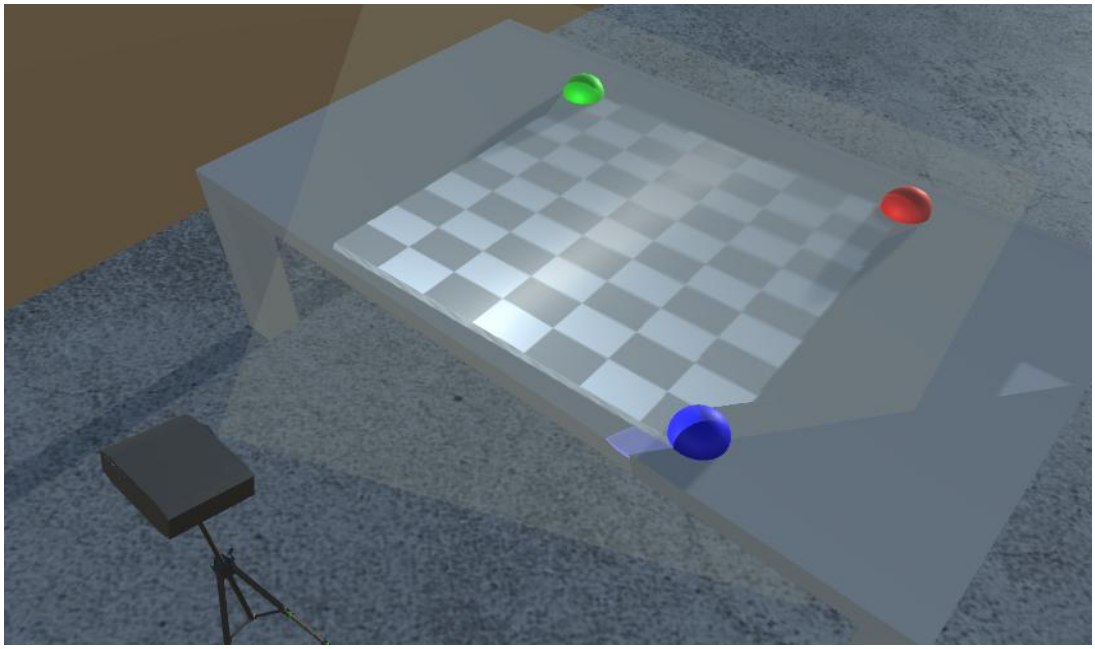

Рисунок А.4 – Установка точек в режиме позиционирования

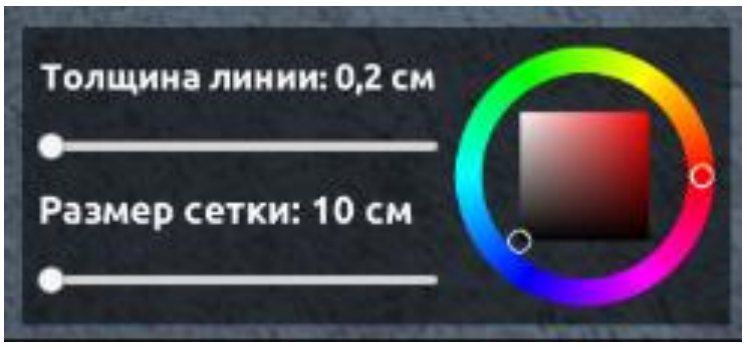

Рисунок А.5 – Открывающееся подменю для настройки квадратной сетки

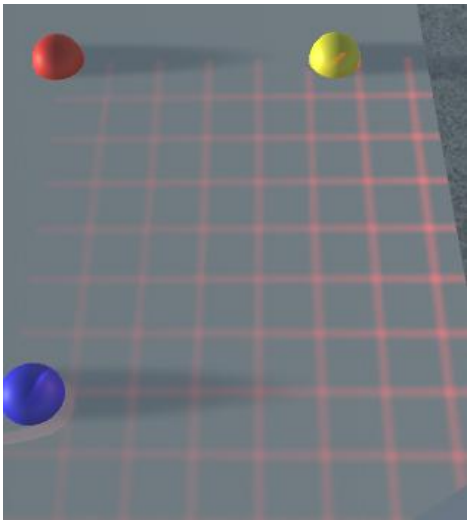

Рисунок А.6 – Пример проецирования на стол квадратной сетки

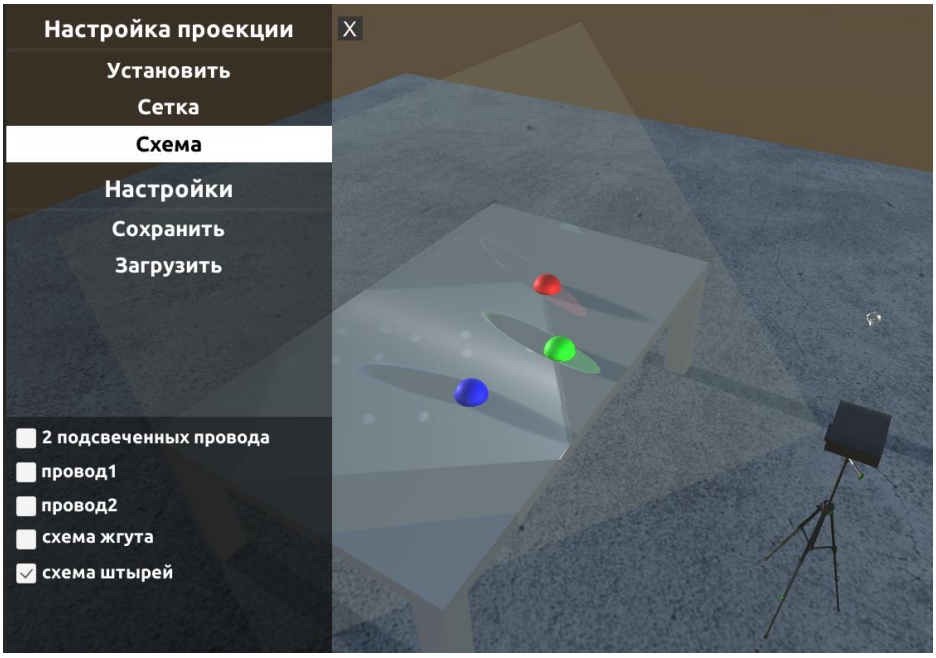

Рисунок А.7 – Список найденных схем и проецирование схемы на стол

ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг Б.1 – SchemeView.cs

```
using System.Collections;
using System.Collections.Generic;
using System.IO;
using UnityEngine;

//класс размещает изображение на объекте cube в runtime

public class SchemeView : MonoBehaviour
{
    private Renderer rend;

    private Texture2D texture;

    private RenderTexture rendText;

    void Awake()
    {
        //получаем доступ к компоненту Renderer у объекта cube
        rend = transform.GetComponent<Renderer>();

        //создаём новые текстуры с указанным разрешением
        texture = new Texture2D(1000, 1000);

        rendText = new RenderTexture(texture.width, texture.height,
0);

        //устанавливаем изображение как главную текстуру
        rend.material.mainTexture = rendText;
    }

    //функция по загрузке изображения в текстуру
    public void LoadImage(string path)
    {
        //считываем данные из файла в массив байтов
        byte[] imageData = File.ReadAllBytes(path);

        //загружаем картинку в текстуру
        texture.LoadImage(imageData);

        merge(texture);
    }


    //функция для объединения текстур
    private void merge(Texture2D texture)
    {
```

```
        Graphics.Blit(texture, rendText);}}
```

Листинг Б.2 – Config.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Text;
using System.Text.RegularExpressions;
using UnityEngine;
using UnityEngine.Events;

//класс позволяет записывать и считывать данные из config.json

public class Config : MonoBehaviour
{
    public string configFile;

    public ConfigData[] entries;

    [SerializeField]
    private bool autoLoad;

    public UnityEvent OnSaved;
    public UnityEvent OnLoaded;

    private void Start()
    {
        if (autoLoad)
        {
            Load();
        }
    }

    private string getConfigPath()
    {
        return Path.Combine(Application.streamingAssetsPath,
configFile);
    }

    public void Save()
    {
        List<string> lines = new List<string>();

        foreach (ConfigData entry in entries)
        {

          string line = "\n" + string.Format("\"{0}\" : {1}", en-
try.UniqueID, toJson(entry));
          lines.Add(line);
```

```
        }

        string json_text = "{\n" + string.Join(",", lines) + "\n}";

        File.WriteAllText(getConfigPath(), json_text);

        OnSaved?.Invoke();
    }

    public void Load()
    {
        string filePath = getConfigPath();

        if (!File.Exists(filePath)) Save();

        string raw_text = File.ReadAllText(filePath);
        Dictionary<string, string> map = parseJson(raw_text);

        foreach(ConfigData data in entries)
        {
            string id = data.UniqueID;
            if (map.ContainsKey(id))
            {
                fromJson(map[id], data);

            }
        }

        StartCoroutine(notifyListeners());
    }

    IEnumerator notifyListeners()
    {
        yield return new WaitForSeconds(0);
        OnLoaded?.Invoke();
    }

    #region SIMPLE JSON PARSER

    private string toJson(object obj, bool removeUnityObjects=true)
    {
        string json_txt = JsonUtility.ToJson(obj, true);

        if (!removeUnityObjects) return json_txt;

        // Search Text "Name":{"instanceID":#},
        string pattern = @"\s*""[^""]+""[^""]+""instanceID""[^}]+}";
        Regex reg = new Regex($"(,{pattern})|({pattern}\\s*,)");

        // And remove it from json_txt
        return reg.Replace(json_txt, "");
    }
```

```
    private void fromJson(string json_text, object obj)
    {
        JsonUtility.FromJsonOverwrite(json_text, obj);
    }

    private Dictionary<string, string> parseJson(string data)
    {
        Dictionary<string, string> map = new Dictionary<string,
string>();

        int index = 0;
        while (index < data.Length)
        {
            string key = parseString(data, ref index);
            if (key == "") break;

            string value = parseObject(data, '{', '}', ref index);

            map.Add(key, value);
        }

        return map;
    }

    private string parseString(string data, ref int index)
    {
        // Is there a variable ?
        int i = data.IndexOf('"', index);
        if (i == -1) return "";
        index = i;

        StringBuilder sb = new StringBuilder();
        while (index < data.Length)
        {
            index++;

            char ch = data[index];
            if (ch == '"')
            {
                return sb.ToString();
            }

            // Restricted tokens
            else if ("\n:".Contains(ch)) break;

            sb.Append(ch);
        }

        throw new Exception("Failed to parse JSON string. Token \"
not found!");
    }

    private string parseObject(string data, char openedToken, char
```

```
closedToken, ref int index)
    {
        StringBuilder sb = new StringBuilder();

        // Is there an object ?
        int i = data.IndexOf(openedToken, index);
        if (i == -1) return "";
        index = i;

        int links = 0;

        while (index < data.Length)
        {
            char ch = data[index];
            sb.Append(ch);

            if (ch == openedToken) links++;
            else if (ch == closedToken) links--;

            if (links == 0) return sb.ToString();
            index++;
        }

        throw new Exception("Failed to parse JSON object. Token } not
found!");
    }
    #endregion SIMPLE JSON PARSER
}
```

Листинг Б.3 – ConfigData.cs

```
using System;
using UnityEngine;
using UnityEngine.Subsystems;

//при смене id с помощью кода изменения не сохраняются
//необходимо самостоятельно сохранить сцену
//Simulator >> Save Scene

public class ConfigData : MonoBehaviour
{
    [SerializeField, HideInInspector]
    private int uniqueID = generateID();

    public string UniqueID
    {
        get
        {
            return GetType().Name + '_' + uniqueID.ToString();
        }

    }
```

```
    private static int generateID()
    {
        var rand = new System.Random();
        int value = rand.Next(100,999);
        return value;
    }
}
```

Листинг Б.4 – Scheme.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Scheme : ConfigData
{
    public float width;
    public float height;
    public string path;
    private Vector3 scale;

    public void Init(GameObject scheme)
    {
        scale = scheme.transform.localScale;
        scale.x = width;
        scale.y = height;
        scheme.transform.localScale = scale;
    }
}
```

Листинг Б.5 – TrackerInfo.cs

```
using System;

public class TrackerInfo
{
    [Serializable]
    public class ViveTracker
    {
        public string serial;
        public bool isConnected;
        public bool isValid;
        public int TrackingResult;
        public int buttonState;

        public int role;
        public float x;
        public float y;
        public float z;
```

```
        public float qw;
        public float qx;
        public float qy;
        public float qz;
    }

    public ViveTracker[] vr_trackers;
}
```

Листинг Б.6 – NetTracker.cs

```
using System.Collections.Generic;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using UnityEngine;
using UnityEngine.Events;

//класс позволяет получать данные о трекерах

public class NetTracker : MonoBehaviour
{
    #region UDP Server
    public class UDPServer
    {
        private int port = 2020;
        private UdpClient udpClient;
        private Thread threadObject;
        private IPEndPoint IPEndPoint = new IPEndPoint(IPAddress.Any,
0);
        private TrackerInfo trackerInfo = new TrackerInfo();

        private List<NetTracker> trackers = new List<NetTracker>();

        static private UDPServer instance;

        private UDPServer()
        {
            udpClient = new UdpClient(port);
            threadObject = new Thread(thread);
            threadObject.Start();
        }

        public static UDPServer GetInstance()
        {
            if (instance == null) instance = new UDPServer();
            return instance;
        }

        public void Register(NetTracker netTracker) { track-
```

```
ers.Add(netTracker); }
        public void Unregister(NetTracker netTracker) { track-
ers.Remove(netTracker); }

        private void thread()
        {
            while (true)
            {
                byte[] data = udpClient.Receive(ref IPEndPoint);
                dataProcessing(data);
            }
        }

        private void dataProcessing(byte[] data)
        {
            var message = Encoding.UTF8.GetString(data);

            trackerInfo = JsonUtility.FromJson<TrackerInfo>(message);


            Debug.Log("Данные о трекерах перезаписаны в класс хране-
ния");
            SetTrackerData();
        }

        public void SetTrackerData()
        {
            foreach(NetTracker netTracker in trackers)
            {
                foreach(TrackerInfo.ViveTracker viveTracker in
trackerInfo.vr_trackers)
                {
                    if (netTracker.serialNumber ==
viveTracker.serial)
                    {
                        netTracker.tracker = viveTracker;
                    }
                }
            }
        }

        public void Stop()
        {
            if (threadObject.IsAlive) threadObject.Abort();
            if (udpClient != null) udpClient.Close();
        }
    }
    #endregion UDP Server

    public string serialNumber = "LHR-756EF5A2";

    private Quaternion swapDirection = Quaterni-
on.LookRotation(Vector3.forward, Vector3.up) *  // direction for role
```

```
"held in hand"
                                        Quaterni-
on.LookRotation(Vector3.up, Vector3.forward);   // direction for oth-
er roles

    private Vector3 position;
    private Quaternion rotation;

    private TrackerInfo.ViveTracker tracker;
    private int button = 0;

    public UnityEvent OnClick;

    private void Start()
    {
        tracker = new TrackerInfo.ViveTracker();
    }

    private void Update()
    {
        if (button != tracker.buttonState)
        {
            button = tracker.buttonState;
            if (button != 0)
            {
                OnClick?.Invoke();
                Debug.Log("кнопка нажата: " + tracker.buttonState);
            }
        }

        position = new Vector3(tracker.x, tracker.y, tracker.z);
        transform.position = position;

        //                          Any hand           Left hand
right hand
        bool held_in_hand = (tracker.role == 0 || tracker.role == 1
|| tracker.role == 2);
        rotation = new Quaternion(-tracker.qx, -tracker.qy, track-
er.qz, tracker.qw);
        if (held_in_hand )
        {
            rotation = rotation * swapDirection;
        }
        transform.rotation = rotation;
    }

    private void OnEnable()
    {
        UDPServer server = UDPServer.GetInstance();
        server.Register(this);
    }

    private void OnDisable()
```

```
    {
        UDPServer.GetInstance().Unregister(this);
    }

    private void OnDestroy()
    {
        UDPServer.GetInstance().Stop();
    }
}
```

Листинг Б.7 – MessageBox.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;
using UnityEngine.UI;
using Unity.VisualScripting;

//Добавляет на сцену окно подсказок
public class MessageBox : MonoBehaviour
{
    //для окна сообщений
    private static MessageBox instance;
    //для префаба
    public GameObject Template;


    private void Awake()
    {
        instance = this;
    }

    public static void show(string text, int life_time)
    {
        GameObject messageBox = instance.Show(text, life_time);

    }

    public static void show(string text)
    {
        GameObject messageBox = instance.Show(text);
    }


    public GameObject Show(string text)
    {
        if (GameObject.Find("UIMessage(Clone)") != null)
        {

            Text textchanger =
```

```
GameObject.Find("ThreePointsText").GetComponent<Text>();

            //вносим изменения в панель с текстом
            textchanger.text = text;
            StopAllCoroutines();
            return null;
        }
        else
        {
            //Instantiate создаёт существующий объект и добавляет его
на сцену
            GameObject messageBox = Instantiate(instance.Template);

            //для объекта панель
            Transform panel = messageBox.transform.Find("Message");

            //для объекта текст
            Text messageBoxtext = pan-
el.Find("ThreePointsText").GetComponent<Text>();

            //вносим изменения в панель с текстом
            messageBoxtext.text = text;

            //для кнопки чтобы закрыть панель
            Button CloseButton = pan-
el.Find("CloseButton").GetComponent<Button>();


            //событие, закрывающее окно по нажатию
            CloseButton.onClick.AddListener(() =>
            {
                Destroy(messageBox);

            });


            return messageBox;
        }

    }

    public GameObject Show(string text, int life_time)
    {
        if (GameObject.Find("UIMessage(Clone)") != null)
        {
            GameObject exist = GameObject.Find("UIMessage(Clone)");
            Text textchanger =
GameObject.Find("ThreePointsText").GetComponent<Text>();

            //вносим изменения в панель с текстом
            textchanger.text = text;
            //функция, закрывающая окно по таймеру
            StartCoroutine(destroyObj(exist, life_time));
```

```
            return null;
        }
        else
        {
            //Instantiate клонирует существующий объект и добавляет
его на сцену
        GameObject messageBox = Instantiate(instance.Template);

        //для объекта панель
        Transform panel = messageBox.transform.Find("Message");

        //для объекта текст
        Text messageBoxtext = pan-
el.Find("ThreePointsText").GetComponent<Text>();

        //вносим изменения в панель с текстом
        messageBoxtext.text = text;

        //для кнопки чтобы закрыть панель
        Button CloseButton = pan-
el.Find("CloseButton").GetComponent<Button>();

        //функция, закрывающая окно по таймеру
        StartCoroutine(destroyObj(messageBox, life_time));


        //событие, закрывающее окно по нажатию
        CloseButton.onClick.AddListener(() =>
        {
            Destroy(messageBox);

        });


        return messageBox;
        }

    }

    //функция для скрытия окна messageBox
    public IEnumerator destroyObj(GameObject box, int time)
    {
        yield return new WaitForSeconds(time);
        Destroy(box);
    }

}
```

Листинг Б.8 – Messages.cs

```
using System.Collections;
```

```
using System.Collections.Generic;
using UnityEngine;

public class Messages : MonoBehaviour
{
    public void showMessage(string message)
    {
        MessageBox.show(message,2);
    }
}
```

Листинг Б.9 – ToggleData.cs

```
using UnityEngine;

//скрипт содержит поля, необходимые для toggle у схем в UI

public class ToggleData : MonoBehaviour
{
    public string nameScheme;
    public string pathToScheme;
}
```

Листинг Б.10 – ToggleList.cs

```
using UnityEngine;
using UnityEngine.UI;
using System.IO;
using UnityEngine.Events;

//скрипт позволяет выводить список схем в UI при нажатии кнопки "схе-
ма"

public class ToggleList : MonoBehaviour
{
    [Header("Настройка элементов списка")]

    [SerializeField]
    private GameObject content;

    [SerializeField]
    private GameObject prefab;

    private Scheme schemeEntity;

    public UnityEvent<string> pathToScheme;

    //инициализация элементов списка и их контейнера
    void Awake()
```

```
    {
        content = GameObject.Find("Viewport/Content");

        prefab = (GameObject)Resources.Load("Prefabs/Toggle",
typeof(GameObject));

        schemeEntity =
GameObject.Find("Config").GetComponent<Scheme>();
    }

    public void OnEnable()
    {
        clear();

        ListOfSchemes(schemeEntity.path);
    }

    //метод, подсчитывающий сколько было создано переключателей
    public void ListOfSchemes(string path)
    {
        DirectoryInfo directoryInfo = new DirectoryInfo(path);
        int count = 0;
        foreach (var file in directoryInfo.GetFiles())
        {
            if (Path.GetExtension(file.FullName) == ".png")
            {
                createToggleItem(file);

                count++;
            }
        }

        if (count == 0) MessageBox.show("В указанном каталоге не было
найдено ни одной схемы!", 3);
        else MessageBox.show("Схем найдено: " + count, 3);
    }

    //метод, создающий префаб переключателя и заполняющий его данными
    void createToggleItem(FileInfo file)
    {
        GameObject toggleItem = Instantiate(prefab);
        toggleItem.transform.SetParent(content.transform, false);

        ToggleData toggleDataItem =
toggleItem.GetComponent<ToggleData>();
        string fileName =
Path.GetFileNameWithoutExtension(file.Name);

        toggleDataItem.nameScheme = fileName;
        toggleDataItem.pathToScheme = file.FullName;
        toggleItem.name = fileName;
        toggleItem.GetComponentInChildren<Text>().text = fileName;
```

```
        Toggle toggle = toggleItem.GetComponent<Toggle>();

        toggle.onValueChanged.AddListener((value) => OnToggle(value,
toggleDataItem));
    }

    void OnToggle(bool isChecked, ToggleData toggleDataItem)
    {
        if (isChecked == true)
        {
            pathToScheme.Invoke(toggleDataItem.pathToScheme);
        }
    }

    //метод, удаляющий элементы списка, находившиеся в нём ранее (в
UI)
    void clear()
    {
        if (content.transform.childCount > 0)
        {
            foreach (Transform child in content.transform) De-
stroy(child.gameObject);
        }
    }
}
```

Листинг Б.11 – UIView.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class UIView : MonoBehaviour
{
    public GameObject panel;
    public GameObject Button;
    public GameObject scrollview;

    public void closePanel()
    {
        panel.SetActive(false);
        Button.SetActive(true);
    }

    public void openPanel()
    {
        panel.SetActive(true);
        Button.SetActive(false);
    }

    public void openSchemeList()
```

```
    {
        scrollview.SetActive(true);
    }
}
```

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

СОГЛАСОВАНО

Начальник отдела ОНиПКРС
_____ Е.М. Димитриади
(подпись)
«21» 06 2023 г.

Декан _____
_____ *И.А. Трещёв*
(подпись)

УТВЕРЖДАЮ

Проректор по научной работе
_____ А.В. Космынин
(подпись)
«21» 06 2023 г.

АКТ
о приемке в эксплуатацию проекта «Разработка программного обеспечения для системы проецирования стенда изготовления жгутов»

г. Комсомольск-на-Амуре «21» 06 2023 г.

Комиссия в составе представителей:

со стороны заказчика

- *Г.В. Москалец* – руководитель СКБ,
- *И.А. Трещёв* – декан ФКТ

со стороны исполнителя

- *А.И. Петрова* – руководитель проекта,
- *С.В. Якимова – 9ВТб-1*
- составила акт о нижеследующем:

«Исполнитель» передает проект «Разработка программного обеспечения для системы проецирования стенда изготовления жгутов», в составе:

1.Паспорта

2. Програмной реализации

Руководитель проекта _____ 21.06.2023 *А.Н. Петрова*

(подпись, дата)

Исполнители проекта _____ 21.06.2023 *С.В. Якимова*

(подпись, дата)