**Оглавление**

[Введение 3](#_Toc27525194)

[1. Проектирование 4](#_Toc27525195)

[1.1 Постановка задачи 4](#_Toc27525196)

[1.2 Проектирование сайта 4](#_Toc27525197)

[2. Реализация 2D анимации 8](#_Toc27525198)

[3. Моделирование в Autodesk 3ds Max 11](#_Toc27525199)

[4. Работа с Unity 12](#_Toc27525200)

[5. Руководство по использованию 18](#_Toc27525201)

[Заключение 24](#_Toc27525202)

[Список используемой литературы 25](#_Toc27525203)

[Приложение А. Листинг скриптов 26](#_Toc27525204)

**Введение**

Важным элементом современных виртуальных коммуникаций, которые во многом определяют, насколько успешно будет воспринята информация, являются мультимедийные технологии и средства. Средства мультимедиа реальны и физически ощутимы в современном мире. Это направление реализует представление информации в рамках традиционных технологий реального мира. Их развитие идет по пути увеличения размерности представляемой информации. Так, например, в повседневную жизнь входят трехмерные принтеры, которые стали доступны широкому потребителю. Сегодня появились компактные модели трехмерных принтеров для решения целого ряда профессиональных задач, которыми заняты инженеры и дизайнеры. Эти устройства умеют делать модели машин, детали макетов, даже зубные слепки из гипса и клея или специальных материалов. Без современных средств мультимедиа человеку сложно, а порой и невозможно воспринимать информацию. Многие технические устройства также нуждаются в некотором особом представлении информации, которое основано па мультимедийных подходах к информационным процессам.

Мультимедийные средства виртуального мира представляют информацию в виртуальном пространстве, и, хотя физически они неощутимы, человек вполне их воспринимает и с ними взаимодействует. Например, известны системы, которые с помощью технологий трехмерного моделирования создают виртуальный объект и обеспечивают с ним взаимодействие человека с помощью специальных средств. В этом случае реализуется интерактивная виртуальная коммуникация в виртуальном пространстве, а человек представляет собой элемент этого виртуального пространства, в котором само действие также виртуально.

Для разработки проекта были использованы следующие программы: Unity, Autodesk 3D’s Max, Visual Studio 2019, Adobe Animate CC.

# Проектирование

## 1.1 Постановка задачи

Целью курсового проекта является разработка сайта, содержащего имитационную модель физической лаборатории по теме «Изучение моментов инерции твердых тел». Данная мультимедийная система должна решать следующие задачи:

* осуществлять диалог с пользователем посредством устной и письменной речи;
* имитировать реальную физическую лабораторию.

## 1.2 Проектирование сайта

В ходе работы был спроектирован сайт, который состоит из четырех страниц:

* Главная страница.
* Теоретическое введение.
* Описание установки.
* Лаборатория онлайн.

Главная страница представлена на рисунке 1.1. На ней пользователь может ознакомится с темой работы и узнать ее цель. Здесь имеется также навигация к другим страницам.

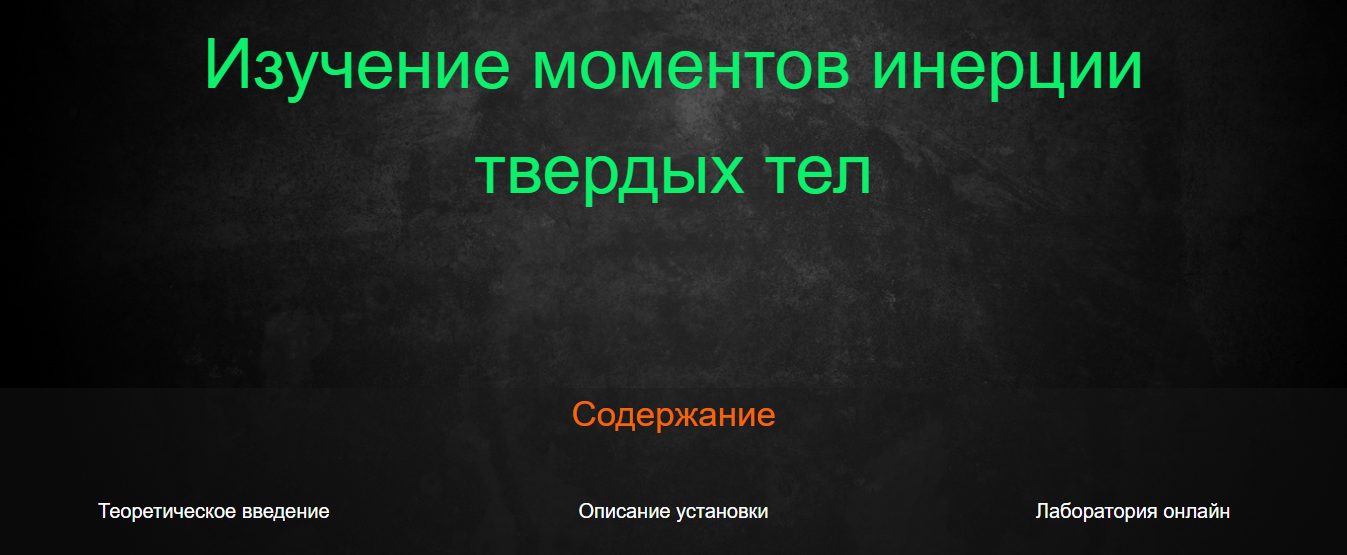


Рисунок 1.1 – Главная страница сайта

Страница с теоретическим введением представлена на рисунке 1.2. Здесь пользователь сможет получить теоретическую базу по теме «Изучение моментов инерции твердых тел.

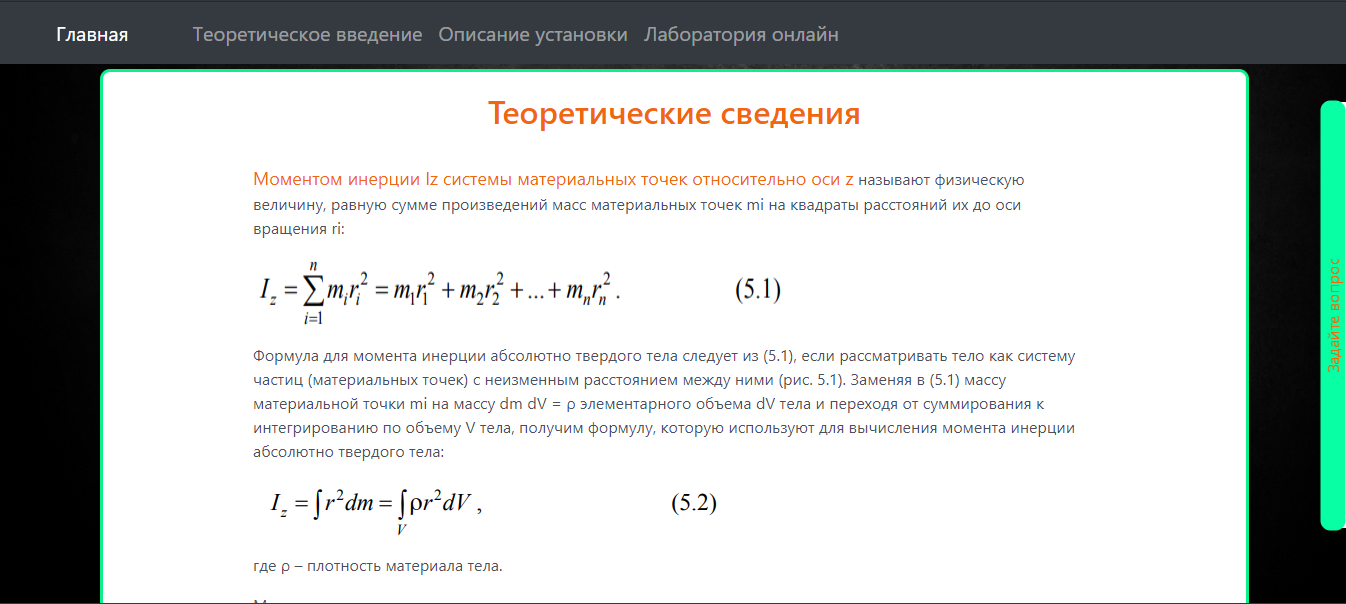


Рисунок 1.2 – Страница сайта «Теоретическое введение»

Страница с описанием установки. На этой странице имеется анимация, позволяющая пользователю получить информацию относительно составляющей части установки. Она изображена на рисунке 1.3.

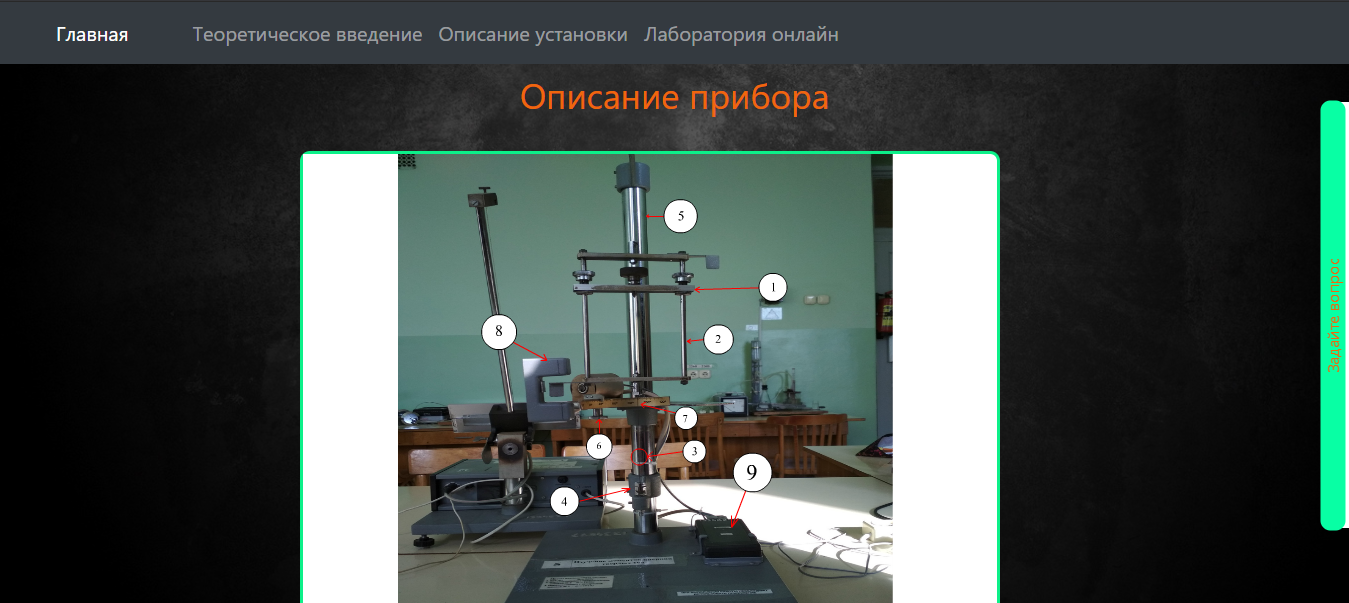


Рисунок 1.3 – Описание прибора на сайте

Также на сайте присутсвует 2D анимация, описывающая принцип работы установки. Она изображена на рисунке 1.4.

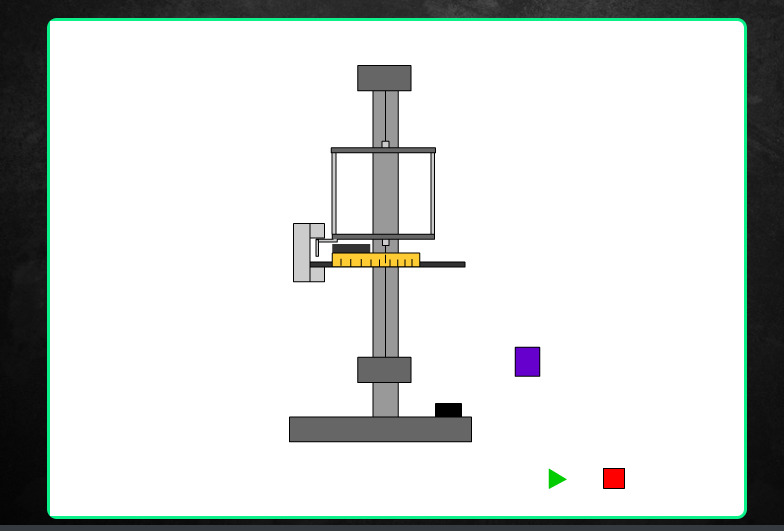


Рисунок 1.4 – 2D анимация на сайте

Страница «Лаборатория онлайн» представлена на рисунке 1.5. Именно здесь содержится виртуальная установка, с которой пользователь будет взаимодействовать для проведения лабораторной работы.

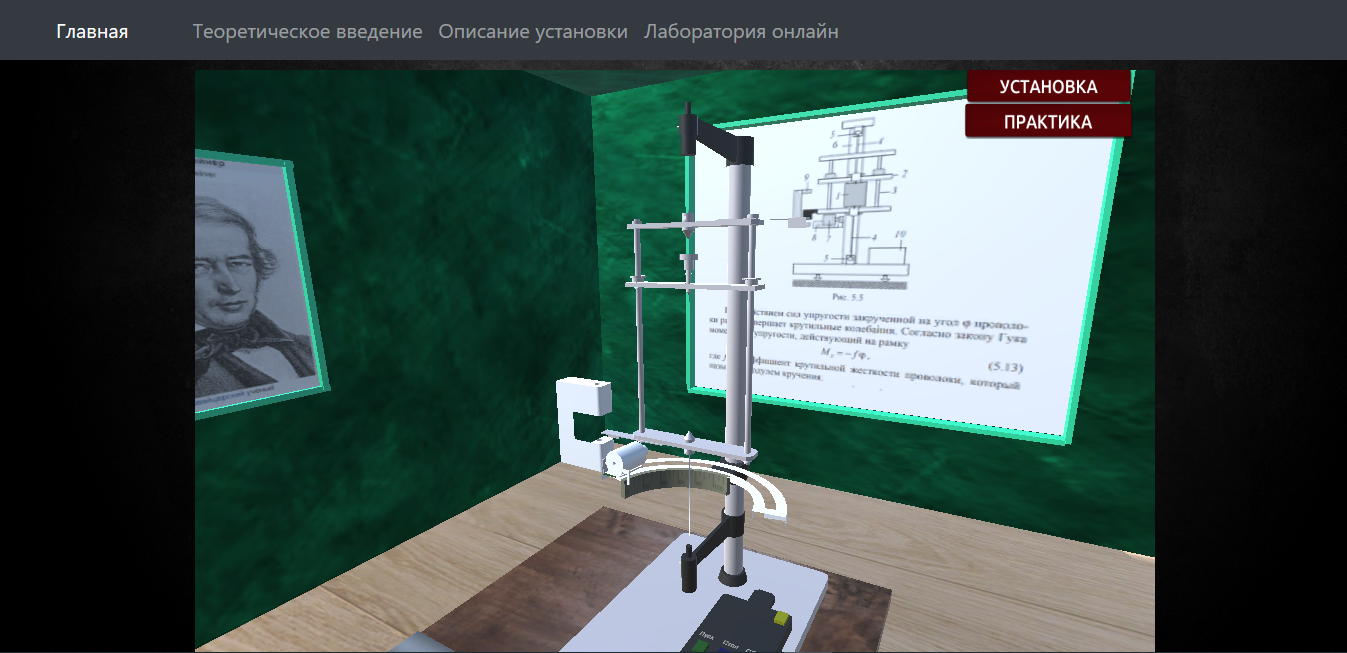


Рисунок 1.5 – страница сайта «Лаборатория онлайн»

Также на некоторых страницах находится диалоговый модуль. Это такой элемент страницы, с помощью которого пользователь сможет получить ответы на интрересующие его вопросы по теме лабораторной работы. Пользователь вводит вопрос в строке ввода и получает соответствующий ответ, если он присутствует в базе. Диалоговый модуль представлен на рисунке 1.6.



Рисунок 1.6 – Диалоговый модуль

Для разработки сайта использовались: HTML, JavaScript, JQuery, CSS.

# Реализация 2D анимации

Для обеспечения работы анимации, изображающей и озвучивающей описание работы с установкой, используемой в лабораторной работе, использовалось программное обеспечение Adobe Animate 2019 и язык программирования ActionScript.

Было разработано два анимированных ролика.

Первый ролик предназначен для описания установки. В анимации можно получить информацию об каждом отдельном элементе. Для этого были наложены невидимые кнопки поверх каждого элемента. При нажатии на эти кнопки воспроизводится соответствующий звук.

Первая анимация изображена на рисунке 2.1.

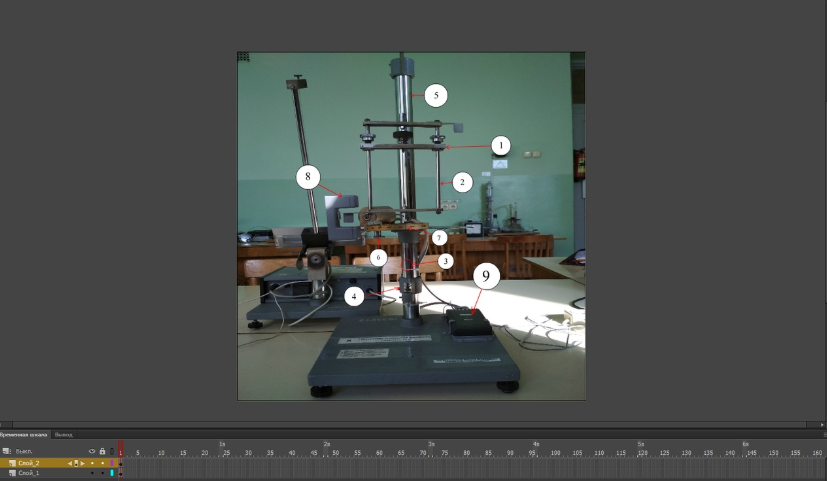


Рисунок 2.1 – Анимация «Описание установки»

Библиотека для первой анимации представлена на рисунке 2.2.

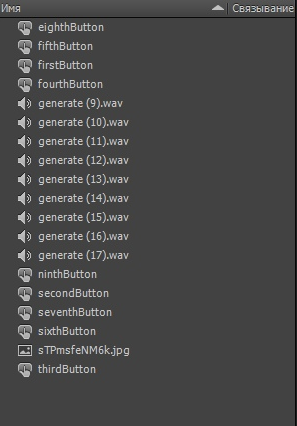


Рисунок 2.2 – Библиотека для анимации «Описание устновки»

Вторая анимация, представленная на рисунке 2.3, описывает принцип работы установки. Здесь присутствуют кнопки манипуляции, которые позволяют запустить или приостановить анимацию. Реализация данной возможности была достигнута за счет ActionScript. Реализация представлена на рисунке 2.4.

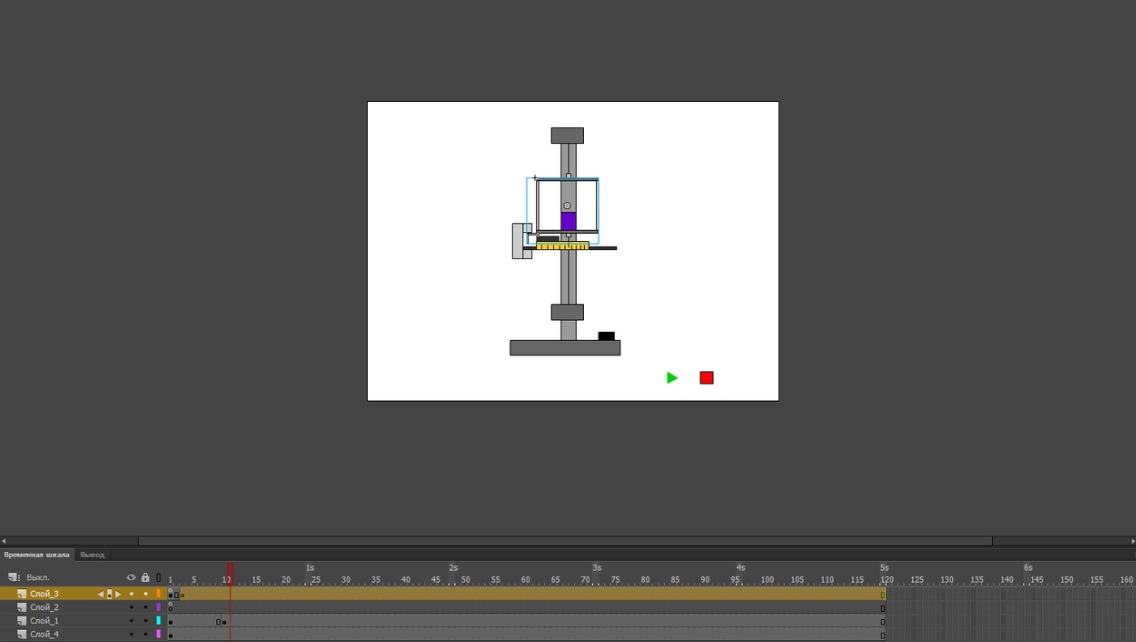


Рисунок 2.3 – Разработка анимации «Принцип работы установки»

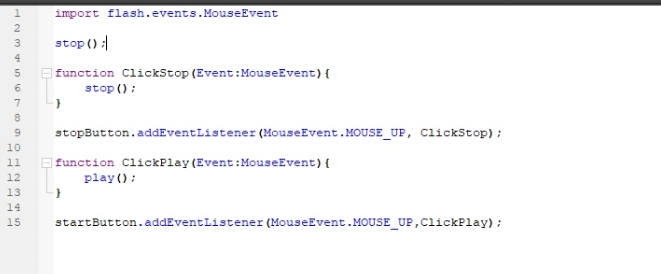


Рисунок 2.4 – Примение ActionScript

# Моделирование в Autodesk 3ds Max

Для создания статической модели установки было применено средство Autodesk 3ds Max. Autodesk 3ds Max — профессиональное программное обеспечение для 3D-моделирования, анимации и визуализации при создании игр и проектировании.

Для создания модели применялись следующие примитивы: Box, Cylinder, Capsule, Torus и др.

Помимо примитивов также были использованы модификаторы. Например, Extract применялся для выдавливания полигонов объекта. Для соединения объектов был применен модификатор Compound objects Boolean. Применялся и такой модификатор как Bevel для скругления углов объектов. В ходе разработки применялись и другие модификаторы.

В результате моделирования была сделана модель установки, изображенная на рисунке 3.1.

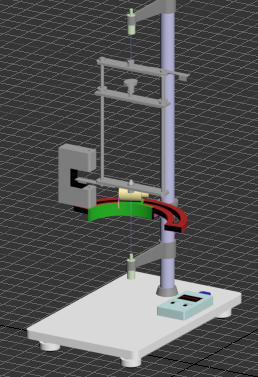


Рисунок 3.1 – Модель установки

В дальнешей в unity на установку были наложены правильные текстуры.

Для экспорта данной модели в Unity была произведена конвертация файлов в формат FBX. Также были настроены оси в соответсвии с особенностями Unity.

# Работа с Unity

Unity — межплатформенная среда разработки компьютерных игр. Unity позволит создавать приложения, работающие под более чем 20 различными операционными системами, включающими персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Выпуск Unity состоялся в 2005 году и с того времени идёт постоянное развитие.

Основными преимуществами Unity являются наличие визуальной среды разработки, межплатформенной поддержки и модульной системы компонентов. К недостаткам относят появление сложностей при работе с многокомпонентными схемами и затруднения при подключении внешних библиотек.

Проект в Unity делится на сцены — отдельные файлы, содержащие свои игровые миры со своим набором объектов, сценариев, и настроек. Сцены могут содержать в себе как объекты (модели), так и пустые игровые объекты — объекты, которые не имеют модели. Объекты, в свою очередь содержат наборы компонентов, с которыми и взаимодействуют скрипты. Также у объектов есть название (в Unity допускается наличие двух и более объектов с одинаковыми названиями), может быть тег (метка) и слой, на котором он должен отображаться. Так, у любого объекта на сцене обязательно присутствует компонент Transform — он хранит в себе координаты местоположения, поворота и размеров объекта по всем трём осям. У объектов с видимой геометрией также по умолчанию присутствует компонент Mesh Renderer, делающий модель объекта видимой.

К объектам можно применять коллизии, которых существует несколько типов.

Также Unity поддерживает физику твёрдых тел и ткани, а также физику типа Ragdoll. В редакторе имеется система наследования объектов; дочерние объекты будут повторять все изменения позиции, поворота и масштаба родительского объекта. Скрипты в редакторе прикрепляются к объектам в виде отдельных компонентов.

В отличие от многих игровых движков, у Unity имеется два основных преимущества: наличие визуальной среды разработки и межплатформенная поддержка. Первый фактор включает не только инструментарий визуального моделирования, но и интегрированную среду, цепочку сборки, что направлено на повышение производительности разработчиков, в частности, этапов создания прототипов и тестирования. Под межплатформенной поддержкой предоставляется не только места развертывания (установка на персональном компьютере, на мобильном устройстве, консоли и т. д.), но и наличие инструментария разработки (интегрированная среда может использоваться под Windows и Mac OS).

Третьим преимуществом называется модульная система компонентов Unity, с помощью которой происходит конструирование игровых объектов, когда последние представляют собой комбинируемые пакеты функциональных элементов. В отличие от механизмов наследования, объекты в Unity создаются посредством объединения функциональных блоков, а не помещения в узлы дерева наследования. Такой подход облегчает создание прототипов, что актуально при разработке игр.

В качестве недостатков приводятся ограничение визуального редактора при работе с многокомпонентными схемами, когда в сложных сценах визуальная работа затрудняется. Вторым недостатком называется отсутствие поддержки Unity ссылок на внешние библиотеки, работу с которыми программистам приходится настраивать самостоятельно, и это также затрудняет командную работу. Ещё один недостаток связан с использованием шаблонов экземпляров. С одной стороны, эта концепция Unity предлагает гибкий подход визуального редактирования объектов, но с другой стороны, редактирование таких шаблонов является сложным. Также, WebGL-версия движка, в силу специфики своей архитектуры (трансляция кода из C# в С++ и далее в JavaScript), имеет ряд нерешённых проблем с производительностью, потреблением памяти и работоспособностью на мобильных устройствах.

На рисунке 4.1 изображена комната с установкой.

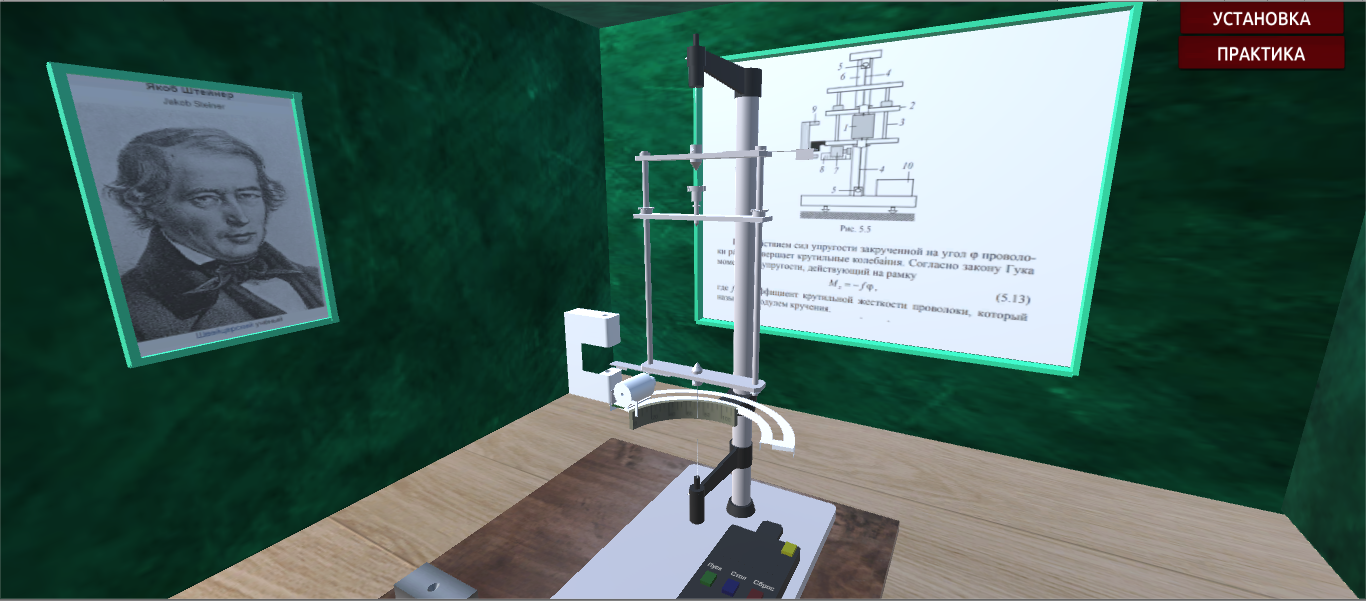


Рисунок 4.1 – Команата с установкой

В верхней части экрана справа находится меню, содержащее две кнопки.

Первая кнопка «Установка» представляет собой раскывающийся список, который представлен на рисунке 4.2.

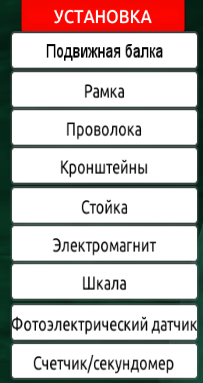


Рисунок 4.2 – Раскрывающийся список «Установка»

С помощью данного списка можно получить теоретические сведения относительно составляющих установки. Демонстрация имеется на рисунке 4.3.



Рисунок 4.3 – Теоретические сведения об элементах установки

В Unity была экспортирована установка и прочие элементы, смоделированные с помощью Autodesk 3ds Max. Элементы, используемые в unity, показаны на рисунке 4.4



Рисунок 4.4 – Объекты сцены Unity

Для того, чтобы запрограммировать какой-либо элемент установки, необходимо добавить к этому элементу новый скрипт, а затем открыть его в Visual Studio. Затем описывается действие этого элемента.

Так, например, если пользователь нажимает на кнопку в выпадающем списке, то камера должна плавно подлететь к элементу. Скрипт показан на рисунке 4.5.

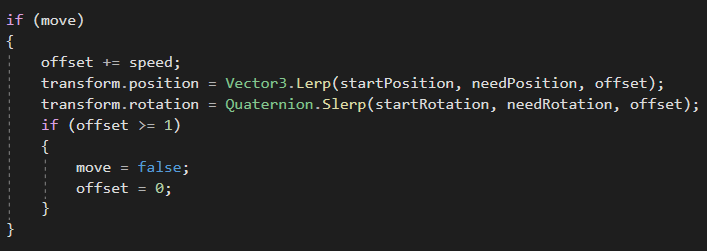


Рисунок 4.5 – Перемещение камеры к объекту

Также можно манипулировать электромагнитом, поворачивая его на определенный угол. Реализация представлена на рисунке 4.6. Здесь с помощью функции OnMouseDrag() отслеживается в какую сторону поворачивается электромагнит.

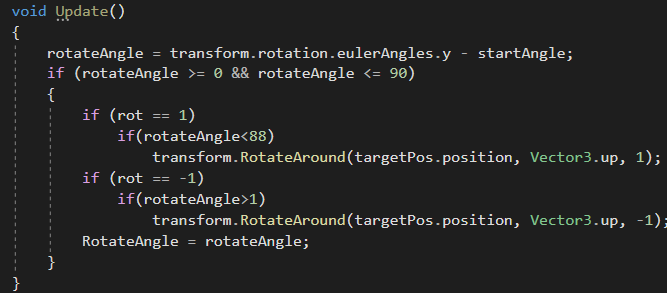
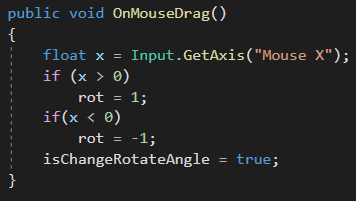


Рисунок 4.6 – Реализация перемещения электромагнита

При наведении на кнопку в выпадающем списке подсвечивается соответсвующий элемент установки. Данная функциональность была реализована с помощью скрипта, показанного на рисунке 4.7.

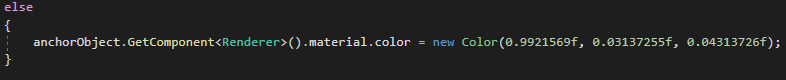
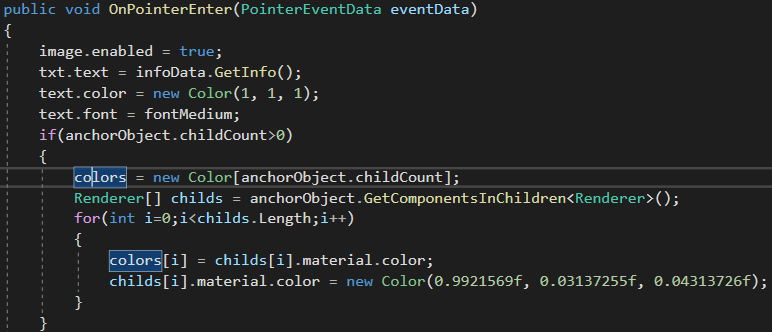


Рисунок 4.7 – Подсветка элементов

Также имеется возможность поместить исследуеме тело на рамку. Для этого был написан следующий скрипт, изображенный на рисунке 4.8.

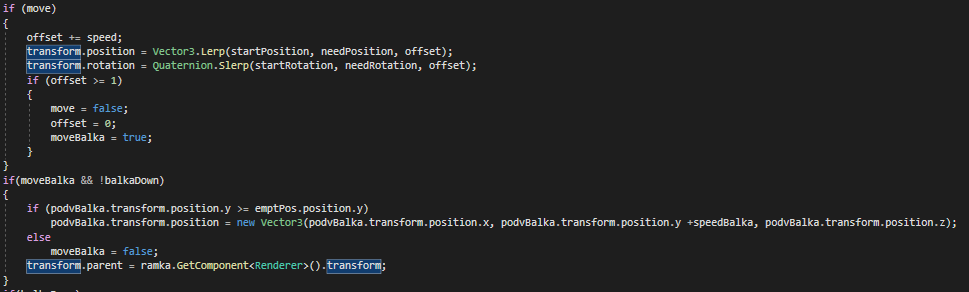


Рисунок 4.8 – Перемещение тела на рамку

Полная реализация всех скриптов предложена в приложении А.

# Руководство по использованию

Данная глава предназначена для предоставления помощи при использовании вышеописанной системы.

Работа начинается с того, что нужно включить установку. Кнопка располагается в правом верхнем углу секундомера. При нажатии на кнопку она загорится желтым цветом. При этом загораются индикаторы для подсчета времени и числа колебаний. Демонстрация данного этапа представлена на рисунке 5.1.

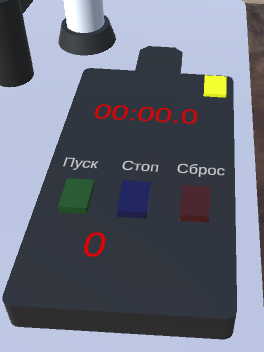


Рисунок 5.1 – Включение установки

Далее пользователь должен выбрать угол отклонения рамки. Для этого нужно с помощью мышки перетащить электромагнит относительно шкалы. Это продемонстрировано на рисунке 5.2.



Рисунок 5.2 – Выбор угла отклонения рамки

Затем необходимо нажать на кнопку пуск секундомера (кнопка зеленого цвета). Рамка начнет вращаться, при этом начнется отсчет времени и числа колебаний. Демонстрация на рисунке 5.3



Рисунок 5.3 – Вращание рамки после нажатия кнопки пуск

После прохождения определенного числа колебани пользователь должен остановить вращение рамки, нажав на кнопку стоп секундомера (синяя кнопка). Рамка вернется в исходное положение и при этом в таблицу занесутся данные периода колебания, число колбаний и другое. Демонстрация показана на рисунке 5.4.



Рисунок 5.4 – Нажатие кнопки стоп секундомера

Затем пользователь должен сбросить старые данные для проведения нового замера. Для этого нужно нажать красную кнопку стоп на секундомера. При этом сбросятся таймер и счетчик числа колебаний. Это показано на рисунке 5.5.



Рисунок 5.5 – Сброс показаний

После проведения пяти таких замеров нужно поместить исследуемое тело на рамку. Для этого необходимо щелкнуть на исследуемое тело, и оно перенесется на рамку. Это продемонстрировано на рисунке 5.6.

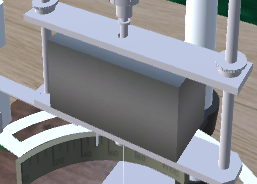


Рисунок 5.6 – Помещение тела на рамку

Затем проделываются такие же действия, что и с пустой рамкой. После этого тело снимается с рамки, для этого просто надо щелкнуть на тело. Затем выключается установка, нажатием на кнопку включения установки (желтая кнопка, была описана выше).

Также пользователь может просмотреть данные, которые заносятся в таблицу в ходе выполения работы. Для этого нужно нажать на соответсвующую кнопку, показанную на рисунке 5.7. При этом появляется таблица, показанная на рисунке 5.8. Чтобы скрыть таблицу, необходимо еще раз нажать на эту же кнопку.



Рисунок 5.7 – Кнопка для показа/скрытия таблицы



Рисунок 5.8 – Таблица со значениями

Также для сайта была разработана база знаний, с помощью которой пользователь модет получить ответы на интересующие его вопросы. Она реализована в виде диалога, представленного на рисунке 5.9.

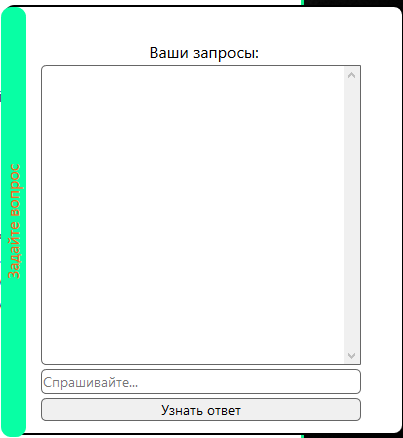


Рисунок 5.9 – Диалоговый модуль

С помощью WebGL установка была встроена на сайт. Симуляция установки на сайте представлена на рисунке 5.10.

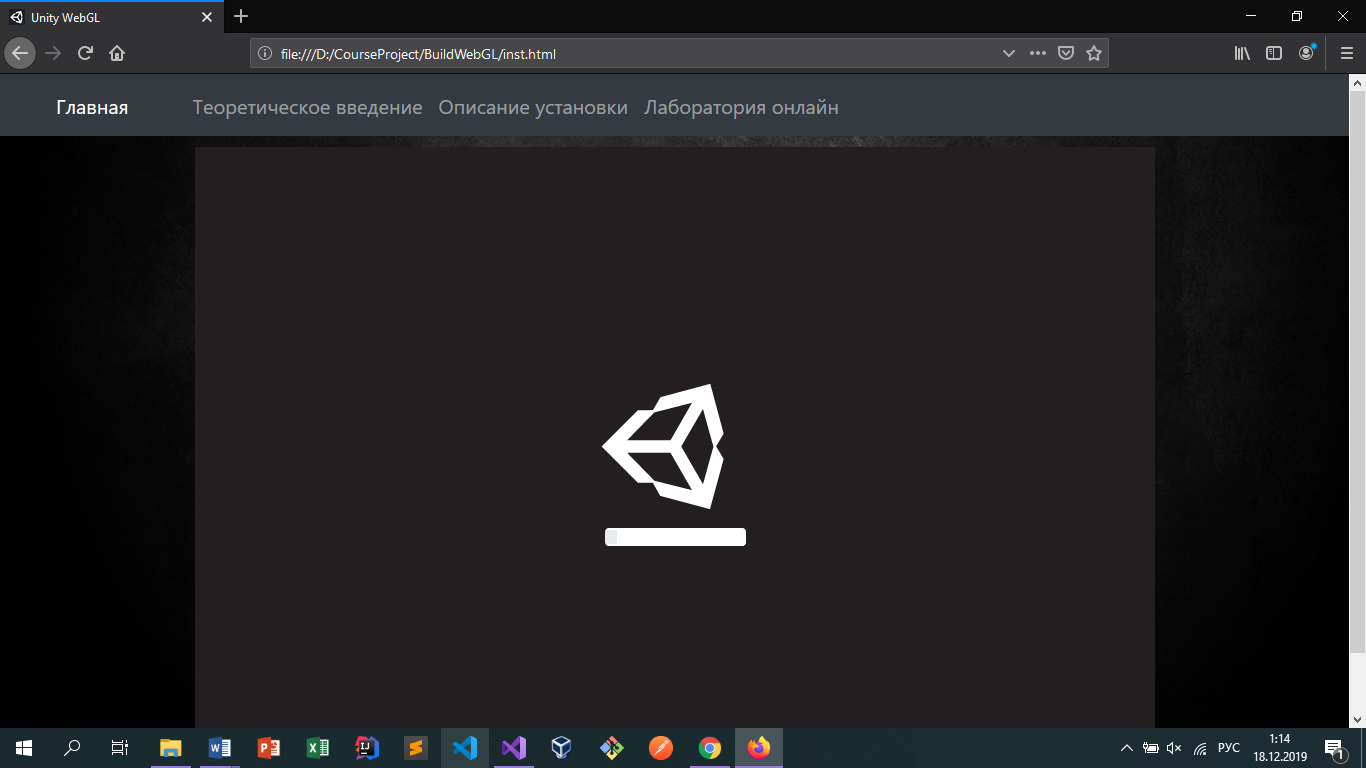


Рисунок 5.10 – Симуляция установки на сайте

# Заключение

При разработке данной курсового проекта были изучены функциональные возможности таких программ, как Unity, Autodesk 3D’s Max, Visual Studio 2019, Adobe Animate CC.

С помощью данных приложений удалось разработать сайт, содержащего имитационную модель физической лаборатории по теме «Изучение моментов инерции твердых тел». Помимо 3D модели, сайт содержал в себе, как и 2D модели, так и диалоговый модуль. Все это позволит как ознакомиться с данной темой, так и разобраться в ней с помощью проведенного эксперимента над имитационной установкой.

# Список используемой литературы

1. Конспект лекций по дисциплине: Компьютерные мультимедийные системы в издательском деле. Преподаватель: Гурин Н. И.
2. Интернет ресурс https://unity3d.com/ru
3. Интернет ресурс https://www.autodesk.com
4. Интернет ресурс https://helpx.adobe.com/ru/animate/user-guide

# Приложение А. Листинг скриптов

**CameraRotate.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.EventSystems;

using System.Threading;

public class cameraRotate : MonoBehaviour

{

float scrollSpeed = 10f;

int sensivity = 3;

int maxdistance = 6;

int mindistance = 2;

bool move = false;

Vector3 startPosition;

Vector3 needPosition;

Quaternion startRotation;

Quaternion needRotation;

float speed = 0.025f;

float offset = 0;

public Transform targetPos;

public Image image;

void Start()

{

image.enabled = false;

}

void FixedUpdate()

{

float x = Input.GetAxis("Horizontal");

float y = Input.GetAxis("Vertical");

if(x!=0 || y!=0)

{

Vector3 newpos = transform.position + (transform.TransformDirection(new Vector3(x, 0, 0)) +

Vector3.up \* y) / sensivity;

if (ControlDistanse(Vector3.Distance(newpos, targetPos.position)))

transform.position = newpos;

}

if(Input.GetAxis("Mouse ScrollWheel")!=0)

{

Vector3 newpos = transform.position + transform.TransformDirection(Vector3.forward \* Input.GetAxis("Mouse ScrollWheel") \*

scrollSpeed);

if (ControlDistanse(Vector3.Distance(newpos, targetPos.position))) transform.position = newpos;

}

if (Input.GetMouseButton(1))

{

transform.RotateAround(targetPos.position, Vector3.up, Input.GetAxis("Mouse X") \* sensivity);

transform.Rotate(Vector3.left, Input.GetAxis("Mouse Y") \* sensivity);

}

if (move)

{

offset += speed;

transform.position = Vector3.Lerp(startPosition, needPosition, offset);

transform.rotation = Quaternion.Slerp(startRotation, needRotation, offset);

if (offset >= 1)

{

move = false;

offset = 0;

}

}

}

public void MoveTo(Transform t)

{

startPosition = transform.position;

startRotation = transform.rotation;

needPosition = t.transform.position;

needRotation = t.transform.rotation;

offset = 0;

move = true;

}

bool ControlDistanse(float distanse)

{

return (distanse > mindistance && distanse < maxdistance);

}

void Update()

{

}

}

**Dropdown.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.EventSystems;

using UnityEngine.UI;

public class Dropdown : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler {

public GameObject panelTrigger;

RectTransform triggerTransfrom;

public Button button;

RectTransform buttonTransfrom;

Vector2 buttonSize;

public GameObject dropdownList;

RectTransform dropdownTransfrom;

Vector2 dropdownSize;

public Sprite buttonNormalState;

public Sprite buttonHighlightedState;

void Start()

{

if (dropdownList != null)

{

triggerTransfrom = panelTrigger.GetComponent<RectTransform>();

buttonTransfrom = button.GetComponent<RectTransform>();

dropdownTransfrom = dropdownList.GetComponent<RectTransform>();

buttonSize = buttonTransfrom.sizeDelta;

dropdownSize = dropdownTransfrom.sizeDelta;

dropdownTransfrom.sizeDelta = new Vector2(0, 0);

triggerTransfrom.sizeDelta = buttonSize;

dropdownList.SetActive(false);

}

}

public void OnPointerEnter(PointerEventData eventData)

{

button.GetComponent<Image>().sprite = buttonHighlightedState;

dropdownTransfrom.sizeDelta = dropdownSize;

triggerTransfrom.sizeDelta = buttonSize + dropdownSize;

dropdownList.SetActive(true);

}

public void OnPointerExit(PointerEventData eventData)

{

dropdownList.SetActive(false);

if (dropdownList != null)

{

dropdownTransfrom.sizeDelta = new Vector2(0, 0);

triggerTransfrom.sizeDelta = buttonSize;

button.GetComponent<Image>().sprite = buttonNormalState;

ChangeButtonTextOnHover[] mas = dropdownList.GetComponentsInChildren<ChangeButtonTextOnHover>();

foreach (var item in mas)

{

item.PointerOut();

}

dropdownList.SetActive(false);

}

}

}

**EkeltrpMagnRotate.cs**

using System.Collections;

using System.Collections.Generic;

using TMPro;

using UnityEngine;

public class ElektrpMagnRotate : MonoBehaviour

{

public Transform targetPos;

double startAngle;

double rotateAngle;

double rot;

public double RotateAngle { get; private set; }

public bool isChangeRotateAngle { get; private set; }

void Start()

{

rot = 0;

startAngle = transform.rotation.eulerAngles.y;

isChangeRotateAngle = false;

}

void Update()

{

rotateAngle = transform.rotation.eulerAngles.y - startAngle;

if (rotateAngle >= 0 && rotateAngle <= 90)

{

if (rot == 1)

if(rotateAngle<88)

transform.RotateAround(targetPos.position, Vector3.up, 1);

if (rot == -1)

if(rotateAngle>1)

transform.RotateAround(targetPos.position, Vector3.up, -1);

RotateAngle = rotateAngle;

}

}

public void OnMouseUp()

{

rot = 0;

isChangeRotateAngle = false;

}

public void OnMouseDrag()

{

float x = Input.GetAxis("Mouse X");

if (x > 0)

rot = 1;

if(x < 0)

rot = -1;

isChangeRotateAngle = true;

}

}

**Info.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

[CreateAssetMenu(fileName = "Info data", menuName = "ScriptInfo", order = 51)]

public class Info : ScriptableObject {

[SerializeField]

private string information;

[SerializeField]

private Sprite image;

public string GetInfo()

{

return information;

}

public Sprite GetImage()

{

return image;

}

}

**LabTasks.cs**

using System;

using System.Collections;

using System.Collections.Generic;

using TMPro;

using UnityEngine;

using UnityEngine.UI;

public class LabTasks : MonoBehaviour

{

public GameObject elektroMagn;

public GameObject startButton;

public GameObject stopButton;

public GameObject resetButton;

public GameObject onnOffButton;

public GameObject telo1;

public GameObject ramka;

public Text taskText;

public Text avT1;

public Text avT2;

public Text pogT1;

public Text pogT2;

public TextMeshPro textN;

public Button resetBut;

public delegate void task();

public static task taskContainer;

int exp;

double completeIz;

double completedIz;

double e;

public void Start()

{

taskContainer = Task1;

exp = 0;

}

void Update()

{

taskContainer();

if (textN.text == "0")

stopButton.GetComponent<BoxCollider>().enabled = false;

else

stopButton.GetComponent<BoxCollider>().enabled = true;

}

void Task1()

{

taskText.text = "Задание 1: Включите установку (для этого нажмите кнопку, которая расположена в верхнем правом углу секундомера)";

if(onnOffButton.GetComponent<OnnOffButton>().Active)

{

onnOffButton.GetComponent<BoxCollider>().enabled = false;

taskContainer -= Task1;

taskContainer += Task2;

}

}

void Task2()

{

taskText.text = "Задание 2: Выберите угол отклонения рамки, для этого перетащите электромагнит по шкале";

elektroMagn.GetComponent<ElektrpMagnRotate>().enabled = true;

startButton.GetComponent<BoxCollider>().enabled = false;

stopButton.GetComponent<BoxCollider>().enabled = false;

resetButton.GetComponent<BoxCollider>().enabled = false;

telo1.GetComponent<BoxCollider>().enabled = false;

if (elektroMagn.GetComponent<ElektrpMagnRotate>().RotateAngle>=1

&& !elektroMagn.GetComponent<ElektrpMagnRotate>().isChangeRotateAngle)

{

taskContainer -= Task2;

startButton.GetComponent<BoxCollider>().enabled = true;

taskContainer += Task3;

}

}

public void Task3()

{

if(exp==0)

{

taskText.text = $"В следующих заданиях: 3,4,5 необходимо будет произвести 5 замеров.\nЗадание 3 (Замер {exp+1}):" +

$"Нажмите кнопку старт на секундомере";

if (startButton.GetComponent<StartSecundomer>().StartSecund)

{

taskContainer -= Task3;

startButton.GetComponent<BoxCollider>().enabled = false;

stopButton.GetComponent<BoxCollider>().enabled = true;

elektroMagn.GetComponent<ElektrpMagnRotate>().enabled = false;

taskContainer += Task4;

}

}

else

{

taskText.text = $"Задание 3 (Замер {exp+1}): Нажмите кнопку старт на секундомере";

if (startButton.GetComponent<StartSecundomer>().StartSecund)

{

taskContainer -= Task3;

startButton.GetComponent<BoxCollider>().enabled = false;

stopButton.GetComponent<BoxCollider>().enabled = true;

taskContainer += Task4;

}

}

}

void Task4()

{

taskText.text = $"Задание 4 (Замер {exp+1}): Дождитесь, когда пройдет N количество колебаний и зафиксируйте время, нажав на кнопку 'стоп' " +

"секундомера";

if(stopButton.GetComponent<StopSecundomer>().ActiveButton)

{

taskContainer -= Task4;

stopButton.GetComponent<BoxCollider>().enabled = false;

resetButton.GetComponent<BoxCollider>().enabled = true;

resetButton.GetComponent<Renderer>().enabled = true;

taskContainer += Task5;

}

}

void Task5()

{

taskText.text = $"Задание 5 (Замер {exp+1}): Сбросьте показатели секундомера, нажав кнопку 'сброс'";

ramka.GetComponent<RotateRamka>().CounterRotateRamka = 0;

if(resetButton.GetComponent<ResetSecundomer>().ActiveButton)

{

exp++;

if(exp>4)

{

taskContainer -= Task5;

stopButton.GetComponent<BoxCollider>().enabled = false;

resetButton.GetComponent<BoxCollider>().enabled = false;

startButton.GetComponent<BoxCollider>().enabled = false;

telo1.GetComponent<BoxCollider>().enabled = true;

exp = 0;

taskContainer += Task6;

}

else

{

taskContainer -= Task5;

resetButton.GetComponent<BoxCollider>().enabled = false;

startButton.GetComponent<BoxCollider>().enabled = true;

taskContainer += Task3;

}

}

}

void Task6()

{

taskText.text = "Задание 6: Поместите исследуемое тело на рамку, нажав на него";

if(ramka.GetComponent<RotateRamka>().I1z> 6.87 \* Math.Pow(10, -4))

{

taskContainer -= Task6;

telo1.GetComponent<BoxCollider>().enabled = false;

startButton.GetComponent<BoxCollider>().enabled = true;

taskContainer += Task7;

}

}

void Task7()

{

if (exp == 0)

{

taskText.text = $"В следующих заданиях: 7,8,9 необходимо будет произвести 5 замеров.\nЗадание 7 (Замер {exp + 1}):" +

$"Нажмите кнопку старт на секундомере";

if (startButton.GetComponent<StartSecundomer>().StartSecund)

{

taskContainer -= Task7;

startButton.GetComponent<BoxCollider>().enabled = false;

stopButton.GetComponent<BoxCollider>().enabled = true;

elektroMagn.GetComponent<ElektrpMagnRotate>().enabled = false;

taskContainer += Task8;

}

}

else

{

taskText.text = $"Задание 7 (Замер {exp + 1}): Нажмите кнопку старт на секундомере";

if (startButton.GetComponent<StartSecundomer>().StartSecund)

{

taskContainer -= Task7;

startButton.GetComponent<BoxCollider>().enabled = false;

stopButton.GetComponent<BoxCollider>().enabled = true;

taskContainer += Task8;

}

}

}

void Task8()

{

taskText.text = $"Задание 8 (Замер {exp + 1}): Дождитесь, когда пройдет N количество колебаний и зафиксируйте время, нажав на кнопку 'стоп' " +

"секундомера";

if (stopButton.GetComponent<StopSecundomer>().ActiveButton)

{

taskContainer -= Task8;

stopButton.GetComponent<BoxCollider>().enabled = false;

resetButton.GetComponent<BoxCollider>().enabled = true;

taskContainer += Task9;

}

}

void Task9()

{

taskText.text = $"Задание 9 (Замер {exp + 1}): Сбросьте показатели секундомера, нажав кнопку 'сброс'";

ramka.GetComponent<RotateRamka>().CounterRotateRamka = 0;

if (resetButton.GetComponent<ResetSecundomer>().ActiveButton)

{

exp++;

if (exp > 4)

{

taskContainer -= Task9;

stopButton.GetComponent<BoxCollider>().enabled = false;

resetButton.GetComponent<BoxCollider>().enabled = false;

startButton.GetComponent<BoxCollider>().enabled = false;

telo1.GetComponent<BoxCollider>().enabled = true;

exp = 0;

taskContainer += Task10;

}

else

{

taskContainer -= Task9;

resetButton.GetComponent<BoxCollider>().enabled = false;

startButton.GetComponent<BoxCollider>().enabled = true;

taskContainer += Task7;

}

}

}

void Task10()

{

taskText.text = "Задание 10. Снимите тело с рамки (щелкните по нему)";

if (telo1.GetComponent<TeloScript>().transform.position== telo1.GetComponent<TeloScript>().tablePosition

&& telo1.GetComponent<TeloScript>().transform.rotation == telo1.GetComponent<TeloScript>().tableRotation)

{

taskContainer -= Task10;

telo1.GetComponent<BoxCollider>().enabled = false;

onnOffButton.GetComponent<BoxCollider>().enabled = true;

resetButton.GetComponent<ResetSecundomer>().OffButton();

taskContainer += Task11;

}

}

void Task11()

{

taskText.text = "Задание 11. Выключите установку";

if(!onnOffButton.GetComponent<OnnOffButton>().Active)

{

taskContainer -= Task11;

onnOffButton.GetComponent<BoxCollider>().enabled = false;

taskContainer += Task12;

}

}

void Task12()

{

double t1 = double.Parse(avT1.text);

double t2 = double.Parse(avT2.text);

double pogt1 = double.Parse(pogT1.text);

double pogt2 = double.Parse(pogT2.text);

completeIz = ramka.GetComponent<RotateRamka>().I1z \* (((Math.Pow(double.Parse(avT2.text), 2)) / (Math.Pow(double.Parse(avT1.text), 2))) + 1);

e = Math.Sqrt(Math.Pow((ramka.GetComponent<RotateRamka>().dI1z / ramka.GetComponent<RotateRamka>().I1z), 2)

+ ((4 \* Math.Pow(t2, 4)) / (Math.Pow(Math.Pow(t2, 2) - Math.Pow(t1, 2), 2)))

\* (Math.Pow(pogt1 / t1, 2) + Math.Pow(pogt2 / t2, 2)));

completedIz = e \* completeIz;

taskText.text = $"Iz тела = {Math.Round(completeIz,5).ToString()}, e={Math.Round(e,5).ToString()}, dIz={Math.Round(completedIz,5).ToString()}";

}

}

**OnnOffButton.cs**

using System.Collections;

using System.Collections.Generic;

using TMPro;

using UnityEngine;

public class OnnOffButton : MonoBehaviour

{

public TextMeshPro textSec;

public TextMeshPro textN;

public GameObject startButton;

public GameObject stopButton;

public GameObject resetButton;

public GameObject telo;

public GameObject magn;

Vector3 positionButton;

public bool Active { get; private set; }

void Start()

{

positionButton = transform.position;

GetComponent<Renderer>().material.color = new Color(0.5f, 0.5f, 0);

OnnOffFunc(false);

Active = false;

}

void Update()

{

}

void OnnOffFunc(bool onn)

{

startButton.GetComponent<BoxCollider>().enabled = onn;

stopButton.GetComponent<BoxCollider>().enabled = onn;

resetButton.GetComponent<BoxCollider>().enabled = onn;

telo.GetComponent<BoxCollider>().enabled = onn;

magn.GetComponent<ElektrpMagnRotate>().enabled = onn;

}

void OnMouseDown()

{

if(!Active)

{

GetComponent<Renderer>().material.color = new Color(1, 1, 0);

transform.position = new Vector3(positionButton.x, positionButton.y - 0.015f, positionButton.z);

textN.text = "0";

textSec.text = "00:00.0";

OnnOffFunc(true);

Active = true;

}

else

{

GetComponent<Renderer>().material.color = new Color(0.5f, 0.5f, 0);

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

textN.text = textSec.text = "";

OnnOffFunc(false);

Active = false;

}

}

}

**PructButt.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.EventSystems;

public class PructButt : MonoBehaviour, IPointerClickHandler

{

public Transform panel;

public GameObject onnOffButton;

Vector3 startPanel;

bool enableOnnOff;

void Start()

{

startPanel = panel.localScale;

panel.localScale = new Vector3(0, 0, 0);

onnOffButton.GetComponent<BoxCollider>().enabled = false;

enableOnnOff = false;

}

public void OnPointerClick(PointerEventData eventData)

{

if (panel.localScale == startPanel)

panel.localScale = new Vector3(0, 0, 0);

else

panel.localScale = startPanel;

if (enableOnnOff)

{

onnOffButton.GetComponent<BoxCollider>().enabled = false;

enableOnnOff = !enableOnnOff;

}

else

{

onnOffButton.GetComponent<BoxCollider>().enabled = true;

enableOnnOff = !enableOnnOff;

}

}

}

**RotateRamka.cs**

using System;

using System.Collections;

using System.Collections.Generic;

using TMPro;

using UnityEngine;

public class RotateRamka : MonoBehaviour

{

public GameObject magn;

public GameObject startButton;

public Transform emptyArround;

public TextMeshPro textDisplayN;

double rotateAngle;

double startAngle;

DateTime startTime;

TimeSpan timeSpan;

Vector3 startPosition;

Quaternion startRotation;

bool boolStartTime;

public int CounterRotateRamka { get; set; }

bool canIncrem;

bool rot;

double k;

public double I1z { get; set; }

public double dI1z { get; set; }

void Start()

{

startPosition = transform.position;

startRotation = transform.rotation;

CounterRotateRamka = 0;

rot = true;

canIncrem = false;

startAngle = transform.rotation.eulerAngles.y;

k = 161;

I1z = 6.87 \* Math.Pow(10, -4);

dI1z = 0.05 \* Math.Pow(10, -4);

}

void Update()

{

if (startButton.GetComponent<StartSecundomer>().StartSecund)

{

rotateAngle = transform.rotation.eulerAngles.y - startAngle;

if (rot)

{

if (rotateAngle < magn.GetComponent<ElektrpMagnRotate>().RotateAngle)

{

transform.RotateAround(emptyArround.position, Vector3.up, float.Parse(GetFi(I1z).ToString()) \* Time.deltaTime);

if (canIncrem)

{

CounterRotateRamka++;

textDisplayN.text = CounterRotateRamka.ToString();

canIncrem = false;

}

}

else

rot = false;

canIncrem = false;

}

if (!rot)

{

if (rotateAngle > -magn.GetComponent<ElektrpMagnRotate>().RotateAngle)

{

transform.RotateAround(emptyArround.position, Vector3.up, -float.Parse(GetFi(I1z).ToString()) \* Time.deltaTime);

}

else

rot = true;

canIncrem = true;

}

}

}

public double GetFi(double Iz)

{

double fi;

double w0 = Math.Sqrt(k / Iz);

fi = magn.GetComponent<ElektrpMagnRotate>().RotateAngle \* Math.Cos(w0 \* 1);

return fi;

}

public void ReturnPositionRamka()

{

transform.position = startPosition;

transform.rotation = startRotation;

CounterRotateRamka = 0;

rot = true;

}

}

**TableScript.cs**

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class TableScript : MonoBehaviour

{

public GameObject n1s;

public GameObject t1s;

public GameObject T1s;

public GameObject avT1s;

public GameObject pogrT1s;

public GameObject n2s;

public GameObject t2s;

public GameObject T2s;

public GameObject avT2s;

public GameObject pogrT2s;

public Transform telo;

public GameObject ramka;

public Text textHeader;

void Start()

{

WriteStartValues(telo, ramka);

}

void Update()

{

}

void WriteStartValues(Transform telo, GameObject rmk)

{

textHeader.text = $"m={telo.GetComponent<Rigidbody>().mass / 1000} кг a={Math.Round(telo.localScale.x/10,3)} м" +

$" b={Math.Round(telo.localScale.y / 10, 3)} м c={Math.Round(telo.localScale.z / 10, 3)} м" +

$" I1z={rmk.GetComponent<RotateRamka>().I1z} кг⋅м2 ΔI1z =0,00000{rmk.GetComponent<RotateRamka>().dI1z\*1000000} кг⋅м2";

}

public void WriteVal(GameObject stolb, string value)

{

bool stop = false;

int num = 0;

Text[] elements = stolb.GetComponentsInChildren<Text>();

foreach (Text txt in elements)

{

if (txt.text != "")

stop = true;

else

stop = false;

}

if(!stop)

{

for (int i = elements.Length - 1; i >= 0; i--)

{

if (elements[i].text == "")

num = i;

}

elements[num].text = value;

}

if (num==elements.Length-1 && (stolb.tag=="T1s"||stolb.tag=="T2s"))

WriteAverageT(stolb);

}

void WriteAverageT(GameObject stolb)

{

double average = 0;

if(stolb.tag=="T1s")

{

Text[] elem = T1s.GetComponentsInChildren<Text>();

foreach (Text child in elem)

average += double.Parse(child.text.ToString());

average = average / elem.Length;

Text txt = avT1s.GetComponentInChildren<Text>();

txt.text = average.ToString();

pogrT1s.GetComponentInChildren<Text>().text = GetPogr(T1s, average, elem.Length).ToString();

}

if (stolb.tag == "T2s")

{

Text[] elem = T2s.GetComponentsInChildren<Text>();

foreach (Text child in elem)

average += double.Parse(child.text.ToString());

average = average / elem.Length;

Text txt = avT2s.GetComponentInChildren<Text>();

txt.text = average.ToString();

pogrT2s.GetComponentInChildren<Text>().text = GetPogr(T2s, average, elem.Length).ToString();

}

}

double GetPogr(GameObject T, double averageT, double N)

{

double deltaT;

double t = 0;

for(int i=0;i<T.GetComponentsInChildren<Text>().Length;i++)

{

t += Math.Pow(double.Parse(T.GetComponentsInChildren<Text>()[i].text)-averageT, 2);

}

deltaT = Math.Sqrt((1 / (N \* (N - 1))) \* t);

return Math.Round(deltaT,3);

}

public void WriteWithoutTelo(string[] values)

{

WriteVal(n1s, values[0]);

WriteVal(t1s, values[1]);

WriteVal(T1s, values[2]);

}

public void WriteWithTelo(string[] values)

{

WriteVal(n2s, values[0]);

WriteVal(t2s, values[1]);

WriteVal(T2s, values[2]);

}

}

**StartSecundomer.cs**

using System;

using System.Collections;

using System.Collections.Generic;

using TMPro;

using UnityEngine;

using UnityEngine.EventSystems;

public class StartSecundomer : MonoBehaviour

{

Vector3 positionButton;

public TextMeshPro secDisplay;

public GameObject resetButton;

public GameObject stopButton;

public bool StartSecund { get; set; }

DateTime startTime;

public TimeSpan TimeSpan { get; private set; }

void Start()

{

StartSecund = false;

positionButton = GetComponent<Renderer>().transform.position;

GetComponent<Renderer>().material.color = new Color(0, 0.2f, 0);

}

void Update()

{

if(StartSecund)

{

TimeSpan = DateTime.Now.Subtract(startTime);

secDisplay.text=TimeSpan.ToString("mm")+":"+TimeSpan.ToString("ss")+"."+TimeSpan.ToString("%f");

}

}

public void OnMouseDown()

{

this.enabled = true;

GetComponent<Renderer>().transform.position = new Vector3(positionButton.x, positionButton.y-0.015f, positionButton.z);

resetButton.GetComponent<ResetSecundomer>().OffButton();

resetButton.GetComponent<ResetSecundomer>().enabled = false;

stopButton.GetComponent<StopSecundomer>().OffButton();

stopButton.GetComponent<StopSecundomer>().enabled = false;

GetComponent<Renderer>().material.color = new Color(0, 1, 0);

StartSecund = true;

startTime = DateTime.Now;

}

public void ChangeBoolStart()

{

StartSecund = false;

}

public void OffButton()

{

GetComponent<Renderer>().transform.position = positionButton;

GetComponent<Renderer>().material.color = new Color(0, 0.2f, 0);

}

}