МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01-03 «Информационные системы и технологии»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Компьютерные мультимедийные системы в издательском деле»

Тема «Солнечная система»

Исполнитель

студент 3 курса 3 группы Гришанович А. П.

(Ф.И.О.)

Руководитель ассистент Жук Я. А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Руководитель Жук Я. А.

(подпись)

Минск 2018

**Оглавление**

[Введение 4](#_Toc532168464)

[1 Проектирование 5](#_Toc532168465)

[1.1 Постановка задачи 5](#_Toc532168466)

[1.2 Проектирование сайта 5](#_Toc532168467)

[2 Реализация 7](#_Toc532168468)

[2.1 2D-анимация 7](#_Toc532168469)

[2.2 Симулятор Солнечной системы 8](#_Toc532168470)

[3 Руководство по использованию 10](#_Toc532168471)

[Заключение 12](#_Toc532168472)

[Список используемой литературы 13](#_Toc532168473)

[ПРИЛОЖЕНИЕ А. Листинг реализации 2D-анимации 14](#_Toc532168476)

[ПРИЛОЖЕНИЕ Б. Листинг скриптов Unity 16](#_Toc532168539)

# Введение

Все чаще в повседневной жизни стали использовать компьютерные мультимедийные системы. Они находят своё применение в различных областях, включая рекламу, искусство, образование, индустрию развлечений, технику, медицину, математику, бизнес, научные исследования, пространственно-временные приложения и прочие информационные процессы с участием людей. В образовании мультимедиа используется для создания компьютерных учебных курсов и справочников, таких как энциклопедии и сборники. Учебные курсы позволяет пользователю пройти через серию презентаций, тематического текста и связанных с ним иллюстраций в различных форматах представления информации. Теория обучения за последнее десятилетие была значительно развита в связи с появлением мультимедиа. Выделилось несколько направлений исследований, такие как теория когнитивной нагрузки, мультимедийное обучение и другие. Возможности для обучения и воспитания почти бесконечны. Идея медиа-конвергенции также становится одним из важнейших факторов в сфере образования, особенно в сфере высшего образования. Медиа-конвергенция определяется как отдельные технологии, такие как голосовые (и функции телефонии), базы данных (и производные приложения), видео-технологии, которые сейчас совместно используют ресурсы и взаимодействуют друг с другом, комплексно создавая новые оперативности. В образовании медиа-конвергенция – это стремительно меняющийся учебный курс дисциплин, преподаваемых в университетах по всему миру. Газетные компании также пытаются охватить новый феномен путём внедрения его практик в свою работу. Разработчики программного обеспечения могут использовать мультимедиа в компьютерных симуляторах чего угодно: от развлечения до обучения, например: военного или производственного обучения. Мультимедиа для программных интерфейсов часто создаётся как сотрудничество между творческими профессионалами и разработчиками программного обеспечения. Создается более дружественное программное обеспечение, ликвидирующее барьер между пользователем и программой. Средства мультимедиа начинают активно использоваться для разработки систем идентификации в различных сферах: банковской, торговой, охранной, медицинской, исследовательской.

Для разработки данного проекта были использованы следующие программы: Unity, Visual Studio 2017, Sublime Text 3.

# 1 Проектирование

### 1.1 Постановка задачи

Целью курсового проекта является разработка сайта, содержащего имитационную модель Солнечной системы. Данная мультимедийная система должна решать следующие задачи:

* осуществлять диалог с пользователем посредством устной и письменной речи;
* имитировать реальное поведение планетной системы.

### 1.2 Проектирование сайта

В данной работе был спроектирован сайт, показанный на рисунке 1.1.

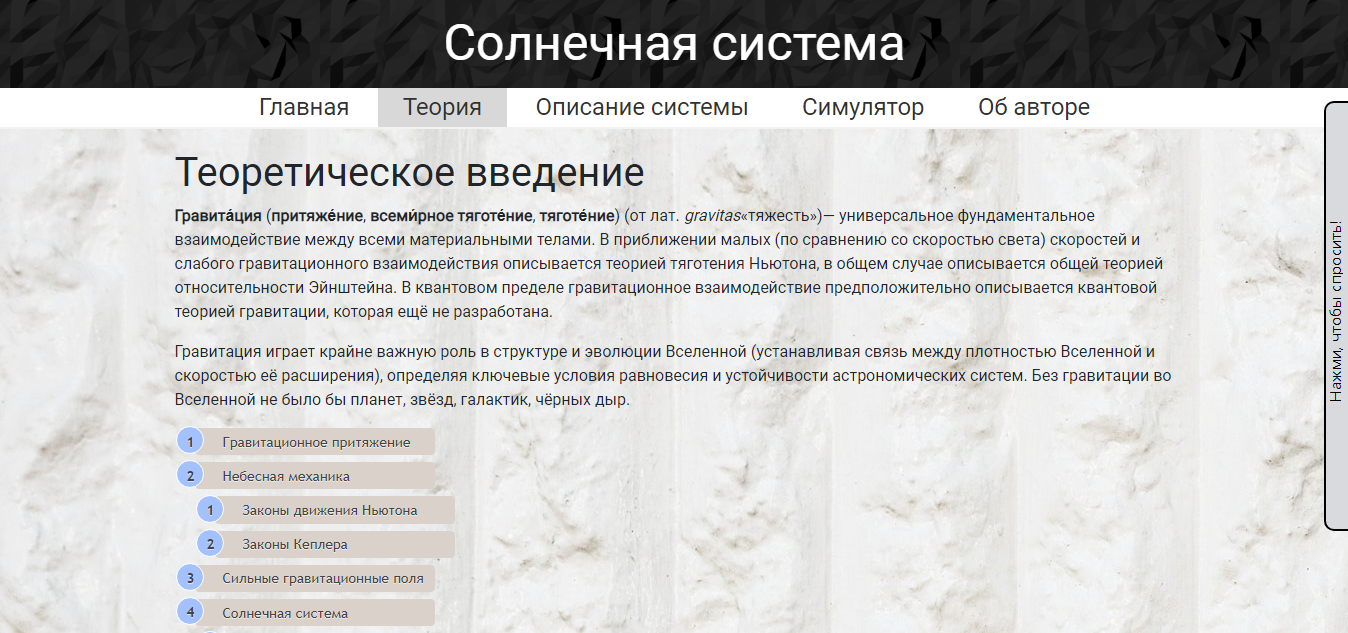


Рисунок 1.1 – Разработанный сайт

Как видно из рисунка, на сайте имеется пять вкладок меню. Более подробно об этих пунктах:

* главная. Данная страница является титульной и служит для перехода на страницу теоретического введения;
* теоретическое введение. Данная вкладка содержит теоретические основы по нашему проекту. На основе этой информации будет строиться устный и письменный диалог с пользователем;
* описание установки. Эта вкладка содержит анимацию, изображающую Солнечную систему, а также озвучивающую её планеты, при клике на них мышью;
* симулятор онлайн. На этой вкладке находится имитационная модель Солнечной системы;
* об авторе. На данной вкладке излагается основная информация о создателе имитационной системы.

Так же на каждой странице сайта присутствует диалоговый модуль, в котором в любой момент можно задать интересующий вас вопрос по теме. Для того чтобы его открыть, требуется нажать на элемент, на котором написано “Нажми, чтобы спросить!”. Он всегда расположен справа на странице. После нажатия выезжает весь диалоговый модуль, который мы можем увидеть на рисунке 1.2. Для того чтобы задать вопрос, его нужно написать в блоке ввода, и далее нажать кнопку “Спросить” или нажать кнопку Enter. Далее вопрос задастся, и озвучит вам ответ. Так же можно ввести вопрос голосовым вводом. Для этого требуется нажать на кнопку с изображением микрофона, и нажать на поле ввода, после этого озвучить свой вопрос. Для того чтобы вернутся к способу ручного ввода вопроса, требуется нажать на кнопку с изображением перечеркнутого микрофона.

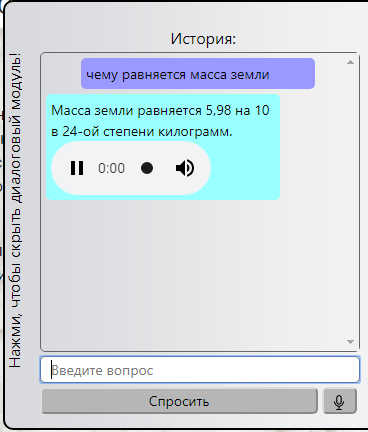


Рисунок 1.2 – Диалоговый модуль

Для создания сайта использовалось программное средство Sublime Text 3, язык разметки HTML с каскадными таблицами стилей CSS, а также язык программирования JavaScript.

# 2 Реализация

### 2.1 2D-анимация

Так как анимация у нас будет находится на сайте, для обеспечения ее работ было решено использовать анимацию CSS. На рисунке 2.1 мы видим фрагмент кода с анимированием HTML компонентов.

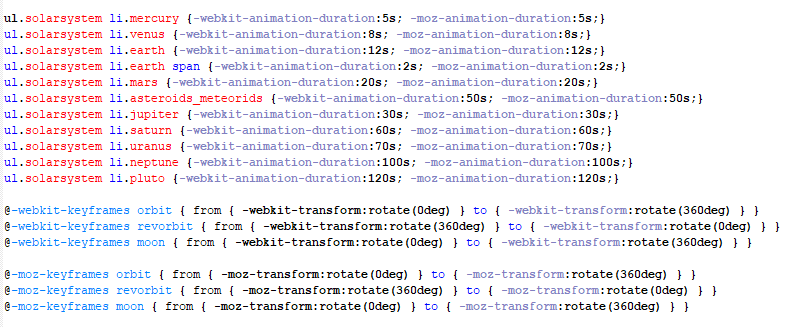


Рисунок 2.1 – Фрагмент кода анимирования компанентов

Данная таблица стилей применяется к HTML-странице, структура которой показана на рисунке 2.2.

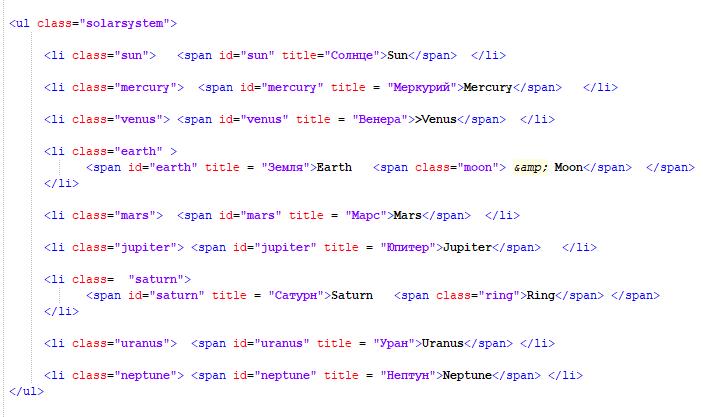


Рисунок 2.2 – Фрагмент HTML страницы

Для озвучиваия названия планет при щелчке на нее, использовались подключенные в HTML аудиофайлы и jquery для запуска аудиопотока при клике на планету.

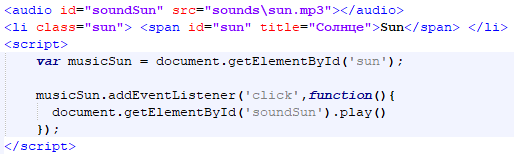


Рисунок 2.3 – Реализация озвучивания компонентов при клики на них

### 2.2 Симулятор Солнечной системы

Для обеспечения работы 3D-симулятора Солнечной системы использовалась среда разработки Unity и язык программирования С#. Объекты сцены представлены на рисунке 2.4.



Рисунок 2.4 – Объекты сцены в Unity

Для того чтобы запрограммировать какой-либо элемент установки,необходимо добавить к этому элементу новый скрипт, а затем открыть его в Visual Studio, где далее описываются действия этого элемента в имитационной модели.

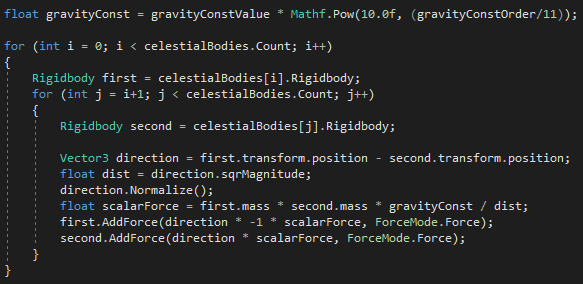


Рисунок 2.5 – Реализация просчета силы притяжения между объектами

Представленная на рисунке 2.5 программная реализация осуществляет просчет силы притяжения между всеми объектами на сцене. Для этого использовалась физичиская формуа гравитации.

Реализация изменения графитацтонной постоянной через графический интерфейс приведена на рисунке 2.6.

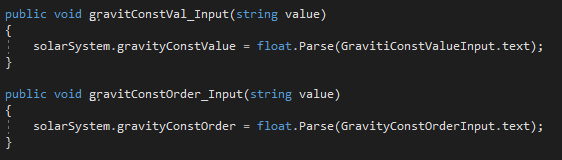


Рисунок 2.6 – Реализация изменения графитацтонной постоянной

Полная реализация всех скриптов проекта представлена вприложении Б.

# 3 Руководство по использованию

В данном руководстве описано, как нужно пользоваться симулятором Солнечной системы. Симулятор представлен на рисунке 3.1.



Рисунок 3.1 – Симулятор Солнечной системы

У симулятора есть главная панель управления (рисунок 3.2), она находится слева и управляет скоростью времени в системе, значение гравитационной постоянной, а также присутствуют кнопки для остановки время и перезапуска Солнечной системы. При изменении значения гравитационной постоянной можно в реальном времени увидеть изменение траектории полета планеты. Также на панели присутствует название объекта, за которым сейчас следит камера.

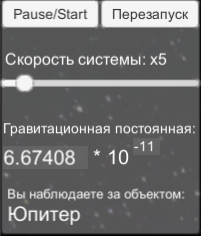


Рисунок 3.2– Левая панель

Также есть вторая панель (рисунок 3.3). Она находится справа. С помощью полей ввода на данной панели мы можем изменять значение масс планет, и видеть изменения их поведения в реальном времени. Также нажав на кнопку в виде глаза рядом с названием планеты, можно переключить вид камеры на данный объект.

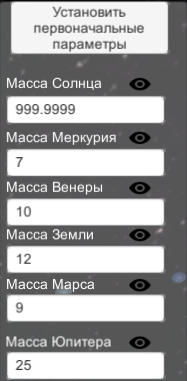


Рисунок 3.3 – Правая панель

Для скрытия или отображения всех панелей требуется нажать на клавитуре кнопку «Q». Для управления камерой используеться мышь. Для приближения камеры к объекту требуется прокрутить колесико мышки либо воспользоваться ползунком слева снизу экрана, что показанно на рисунке 3.4. Для управления громкостью можно воспользоваться палзунком который находиться слева снизу экрана.

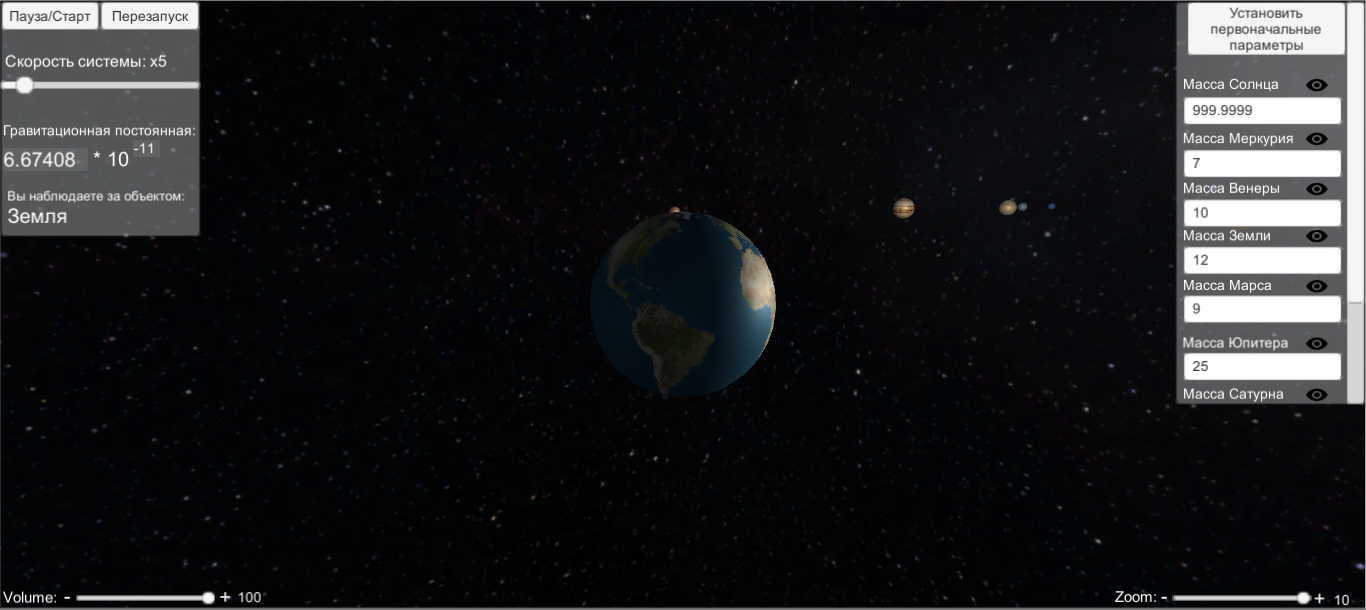


Рисунок 3.4 – Приблежение камеры к объекту

Для того чтобы переключится на наблюдение за другим объектом (планетой), нужно нажать на данную планету левой клавишей мыши или нажать соответствующую кнопку в виде глаза на правой панели. Для поворота камеры вокруг наблюдаемого объекта требуется с зажатой правой клавишей мыши передвигать ее. Для выхода из симулятора требуется нажать кнопку «Esc» на клавиатуре и подтвердить действие, нажав в открывшейся панели на кнопку «Ага».

Заключение

В ходе выполнения данной работы подробно были изучены возможности среды разработки Unity. В данной среде есть возможность создавать приложения для запуска в браузерах с помощью реализации технологии WebGL, также можно разрабатывать приложения и игры для таких платформ как Windows, MacOS, Android, IOS. Проект в Unity делится на сцены — отдельные файлы, содержащие свои игровые миры со своим набором объектов, сценариев, и настроек. Объекты, в свою очередь содержат наборы компонентов, с которыми и взаимодействуют скрипты. Также у объектов есть название, может быть тег (метка) и слой, на котором он должен отображаться. Итогом работы является сайт с готовой имитационной моделью Солнечной системы, которая может быть использовано людьми для того, чтобы упростить изучение астрономии, в частности Солнечной системы, входящих в нее планет, их физических свойств, и поведения при различных условиях, которые можно изменять собственноручно, и наблюдать за изменениями в реальном времени. В данной имитационной модели также можно изучать свойства гравитации, а именно изменение ее свойств при изменении массы объектов либо при изменении гравитационной постоянной. У всех объектов данной модели присутствует такой компонент как «Rigibody». Данный компонент позволяет добавить объекту физические свойства, такие как масса, сила притяжения. Имитационные модели могут применяться в абсолютно различных сферах, например, развлекательной, обучающей, развивающей, и множестве других. Созданная мультимедийная система включает в себя такие возможности, как:

* устный и письменный диалог с пользователем;
* имитация реальной Солнечной системы.

# Список используемой литературы

# Гурин Н. И. Компьютерные обучающие системы в издательском деле. В 2 ч. Ч. 1. Разработка компьютерной обучающей среды: учеб. пособие / Н. И. Гурин, О. В. Герман. – Минск: БГТУ, 2015. – 258 с.

# Unity [Электронный ресурс]. – Режим доступа: https://unity3d.com/ru. – Дата доступа: 01.11.2018.

# MSDN [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/. – Дата доступа: 23.11.2018.

# ПРИЛОЖЕНИЕ А. Листинг реализации 2D-анимации

# var musicSun = document.getElementById('sun');

# musicSun.addEventListener('click',function(){

# document.getElementById('soundSun').play()

# })

# var musicVenus = document.getElementById('venus');

# var musicMerury = document.getElementById('mercury');

# var musicEarth = document.getElementById('earth');

# var musicMars = document.getElementById('mars');

# var musicJupiter = document.getElementById('jupiter');

# var musicSaturn = document.getElementById('saturn');

# var musicUranus = document.getElementById('uranus');

# var musicNeptune = document.getElementById("neptune");

# document.getElementById('mercury').addEventListener('click',function(){

# document.getElementById('soundMercury').play()

# })

# document.getElementById('venus').addEventListener('click',function(){

# document.getElementById('soundVenus').play()

# })

# document.getElementById('earth').addEventListener('click',function(){

# document.getElementById('soundEarth').play()

# })

# document.getElementById('mars').addEventListener('click',function(){

# document.getElementById('soundMars').play()

# })

# document.getElementById('jupiter').addEventListener('click',function(){

# document.getElementById('soundJupiter').play()

# })

# document.getElementById('saturn').addEventListener('click',function(){

# document.getElementById('soundSaturn').play()

# })

# document.getElementById('uranus').addEventListener('click',function(){

# document.getElementById('soundUranus').play()

# })

# document.getElementById("neptune").addEventListener('click',function(){

# document.getElementById('soundNeptune').play()

# })

# ul.solarsystem li.mercury {-webkit-animation-duration:5s; -moz-animation-duration:5s;}

# ul.solarsystem li.venus {-webkit-animation-duration:8s; -moz-animation-duration:8s;}

# ul.solarsystem li.earth {-webkit-animation-duration:12s; -moz-animation-duration:12s;}

# ul.solarsystem li.earth span {-webkit-animation-duration:2s; -moz-animation-duration:2s;}

# ul.solarsystem li.mars {-webkit-animation-duration:20s; -moz-animation-duration:20s;}

# ul.solarsystem li.asteroids\_meteorids {-webkit-animation-duration:50s; -moz-animation-duration:50s;}

# ul.solarsystem li.jupiter {-webkit-animation-duration:30s; -moz-animation-duration:30s;}

# ul.solarsystem li.saturn {-webkit-animation-duration:60s; -moz-animation-duration:60s;}

# ul.solarsystem li.uranus {-webkit-animation-duration:70s; -moz-animation-duration:70s;}

# ul.solarsystem li.neptune {-webkit-animation-duration:100s; -moz-animation-duration:100s;}

# ul.solarsystem li.pluto {-webkit-animation-duration:120s; -moz-animation-duration:120s;}

# @-webkit-keyframes orbit { from { -webkit-transform:rotate(0deg) } to { -webkit-transform:rotate(360deg) } }

# @-webkit-keyframes revorbit { from { -webkit-transform:rotate(360deg) } to { -webkit-transform:rotate(0deg) } }

# @-webkit-keyframes moon { from { -webkit-transform:rotate(0deg) } to { -webkit-transform:rotate(360deg) } }

# @-moz-keyframes orbit { from { -moz-transform:rotate(0deg) } to { -moz-transform:rotate(360deg) } }

# @-moz-keyframes revorbit { from { -moz-transform:rotate(360deg) } to { -moz-transform:rotate(0deg) } }

# @-moz-keyframes moon { from { -moz-transform:rotate(0deg) } to { -moz-transform:rotate(360deg) } }

# 

# <audio id="soundSun" src="sounds\sun.mp3"></audio>

# <audio id="soundMercury" src="sounds\mercury.mp3"></audio>

# <audio id="soundVenus" src="sounds\venus.mp3"></audio>

# <audio id="soundEarth" src="sounds\earth.mp3"></audio>

# <audio id="soundMars" src="sounds\mars.mp3"></audio>

# <audio id="soundJupiter" src="sounds\jupiter.mp3"></audio>

# <audio id="soundSaturn" src="sounds\saturn.mp3"></audio>

# <audio id="soundUranus" src="sounds\uranus.mp3"></audio>

# <audio id="soundNeptune" src="sounds\neptune.mp3"></audio>

# ПРИЛОЖЕНИЕ Б. Листинг скриптов Unity

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class UI\_SolarSystem : MonoBehaviour {

public SolarSystem solarSystem;

public CameraRotateAround camera;

public Text targetName;

public InputField mp;

public Text SpeedValue;

public Slider slider;

public InputField GravitiConstValueInput, GravityConstOrderInput;

public void ResetWorld\_Click()

{

solarSystem.ResetWorld();

}

public void SpeedSystem\_Input()

{

solarSystem.TimeSpeed = slider.value;

SpeedValue.text = "Скорость системы: x" + slider.value.ToString();

}

public void mass\_Input(string mass)

{

}

public void gravitConstVal\_Input(string value)

{

solarSystem.gravityConstValue = float.Parse(GravitiConstValueInput.text);

}

public void gravitConstOrder\_Input(string value)

{

solarSystem.gravityConstOrder = float.Parse(GravityConstOrderInput.text);

}

public void Time\_Pause()

{

if (Time.timeScale != 0)

Time.timeScale = 0;

else Time.timeScale = slider.value;

}

public void Start()

{

SpeedValue.text = "Скорость системы: x" + slider.value.ToString();

GravitiConstValueInput.text = solarSystem.gravityConstValue.ToString();

GravityConstOrderInput.text = solarSystem.gravityConstOrder.ToString();

}

void FixedUpdate () {

targetName.text = camera.target.name;

}

}

using UnityEngine;

using System.Collections;

using System.Collections.Generic;

[RequireComponent(typeof(Rigidbody))]

public class CelestialBody : MonoBehaviour {

private new Rigidbody rigidbody;

public string name;

public Rigidbody Rigidbody

{

get

{

return rigidbody;

}

}

void Awake()

{

rigidbody = GetComponent<Rigidbody>();

}

void FixedUpdate () {

}

}

using UnityEngine;

using System.Collections;

using System.Collections.Generic;

public class SolarSystem : MonoBehaviour {

public Rigidbody sun;

public Rigidbody mercury;//

public Rigidbody venus;//

public Rigidbody earth;

public Rigidbody mars;

public Rigidbody jupiter;

public Rigidbody saturn;

public Rigidbody uranus;

public Rigidbody neptune;

public float gravityConstOrder = -11.0f;

public float gravityConstValue = 6.67408f;

public float startMercuryImpulse = 48;//

public float startVenusImpulse = 25;

public float startEarthImpulse = 25;

public float startMarsImpulse = 25;

public float startJupiterImpulse = 25;

public float startSaturnImpulse = 25;

public float startUranusImpulse = 25;

public float startNeptuneImpulse = 25;

public float TimeSpeed= 1f;//

protected static List<CelestialBody> celestialBodies = new List<CelestialBody>();

protected List<Vector3> defaultPositions;

protected List<Quaternion> defaultRotations;

void Awake()

{

celestialBodies = new List<CelestialBody>(FindObjectsOfType<CelestialBody>());

}

void Start()

{

SaveDefaults();

ResetWorld();

}

public void SpeedSystem(int speed)

{

TimeSpeed = (float)speed;

}

protected void SaveDefaults()

{

defaultPositions = new List<Vector3>();

defaultRotations = new List<Quaternion>();

for (int i = 0; i < celestialBodies.Count; i++)

{

defaultPositions.Add(celestialBodies[i].transform.position);

defaultRotations.Add(celestialBodies[i].transform.rotation);

}

}

public void ResetWorld()

{

for (int i = 0; i < celestialBodies.Count; i++)

{

celestialBodies[i].transform.position = defaultPositions[i];

celestialBodies[i].transform.rotation = defaultRotations[i];

celestialBodies[i].Rigidbody.AddForce( -celestialBodies[i].Rigidbody.velocity, ForceMode.VelocityChange);

celestialBodies[i].Rigidbody.AddRelativeTorque(-celestialBodies[i].Rigidbody.angularVelocity, ForceMode.VelocityChange);

}

mercury.AddForce(Vector3.forward \* startMercuryImpulse, ForceMode.Impulse);

mercury.AddRelativeTorque(Vector3.up \* 1, ForceMode.Impulse);

venus.AddForce(Vector3.forward \* startVenusImpulse, ForceMode.Impulse);

venus.AddRelativeTorque(Vector3.up \* 1, ForceMode.Impulse);

earth.AddForce(Vector3.forward \* startEarthImpulse, ForceMode.Impulse);

earth.AddRelativeTorque(Vector3.up \* 1, ForceMode.Impulse);

mars.AddForce(Vector3.forward \* startMarsImpulse, ForceMode.Impulse);

mars.AddRelativeTorque(Vector3.up \* 1, ForceMode.Impulse);

jupiter.AddForce(Vector3.forward \* startJupiterImpulse, ForceMode.Impulse);

jupiter.AddRelativeTorque(Vector3.up \* 1, ForceMode.Impulse);

saturn.AddForce(Vector3.forward \* startSaturnImpulse, ForceMode.Impulse);

saturn.AddRelativeTorque(Vector3.up \* 1, ForceMode.Impulse);

uranus.AddForce(Vector3.forward \* startUranusImpulse, ForceMode.Impulse);

uranus.AddRelativeTorque(Vector3.up \* 1, ForceMode.Impulse);

neptune.AddForce(Vector3.forward \* startNeptuneImpulse, ForceMode.Impulse);

neptune.AddRelativeTorque(Vector3.up \* 1, ForceMode.Impulse);

/\*\*

ForceMode.Force == Force per second

ForceMode.Impulse == Force per frame

\*/

//earth.AddForce(Vector3.forward \* 1000, ForceMode.Force);

}

void FixedUpdate () {

Time.timeScale = TimeSpeed;

sun.transform.position = Vector3.zero;

float gravityConst = gravityConstValue \* Mathf.Pow(10.0f, (gravityConstOrder/11));

for (int i = 0; i < celestialBodies.Count; i++)

{

Rigidbody first = celestialBodies[i].Rigidbody;

for (int j = i+1; j < celestialBodies.Count; j++)

{

Rigidbody second = celestialBodies[j].Rigidbody;

Vector3 direction = first.transform.position - second.transform.position;

float dist = direction.sqrMagnitude;

direction.Normalize();

float scalarForce = first.mass \* second.mass \* gravityConst / dist;

first.AddForce(direction \* -1 \* scalarForce, ForceMode.Force);

second.AddForce(direction \* scalarForce, ForceMode.Force);

}

}

}

}

using UnityEngine;

using UnityEngine.UI;

public class UI\_Camera : MonoBehaviour {

public CameraRotateAround camera;

public Slider zoomSlider;

public Text zoomValue;

public Text zoomValueRealTime;

// Use this for initialization

void Start () {

zoomSlider.value = camera.offset.z;

zoomValue.text = zoomSlider.value.ToString();

}

public void ZoomSlider\_Input ()

{

zoomValue.text = zoomSlider.value.ToString();

camera.offset.z = -zoomSlider.value;

}

// Update is called once per frame

void Update () {

//zoomValueRealTime.text = camera.offset.z.ToString();

}

}

using UnityEngine;

using UnityEngine.UI;

public class UI\_Planets : MonoBehaviour {

public SolarSystem solarSystem;

public InputField sunMassInput, mercuryMassInput,

venusMassInput, earthMassInput, marsMassInput,

jupiterMassInput, saturnMassInput, uranusMassInput, neptuneMassInput;

public void SetMass\_Default()

{

solarSystem.sun.mass = 1000;

sunMassInput.text = solarSystem.sun.mass.ToString();

solarSystem.mercury.mass = 7;

mercuryMassInput.text = solarSystem.mercury.mass.ToString();

solarSystem.venus.mass = 10;

venusMassInput.text = solarSystem.venus.mass.ToString();

solarSystem.earth.mass = 12;

earthMassInput.text = solarSystem.earth.mass.ToString();

solarSystem.mars.mass = 9;

marsMassInput.text = solarSystem.mars.mass.ToString();

solarSystem.jupiter.mass = 25;

jupiterMassInput.text = solarSystem.jupiter.mass.ToString();

solarSystem.saturn.mass = 24;

saturnMassInput.text = solarSystem.saturn.mass.ToString();

solarSystem.uranus.mass = 15;

uranusMassInput.text = solarSystem.uranus.mass.ToString();

solarSystem.neptune.mass = 13;

neptuneMassInput.text = solarSystem.neptune.mass.ToString();

}

public void SunMass\_Input()

{

solarSystem.sun.mass = float.Parse(sunMassInput.text);

}

public void MercuryMass\_Input()

{

solarSystem.mercury.mass = float.Parse(mercuryMassInput.text);

}

public void VenusMass\_Input()

{

solarSystem.venus.mass = float.Parse(venusMassInput.text);

}

public void EarthMass\_Input()

{

solarSystem.earth.mass = float.Parse(earthMassInput.text);

}

public void MarsMass\_Input()

{

solarSystem.mars.mass = float.Parse(marsMassInput.text);

}

public void JupiterMass\_Input()

{

solarSystem.jupiter.mass = float.Parse(jupiterMassInput.text);

}

public void SaturnMass\_Input()

{

solarSystem.saturn.mass = float.Parse(saturnMassInput.text);

}

public void UranusMass\_Input()

{

solarSystem.uranus.mass = float.Parse(uranusMassInput.text);

}

public void NeptuneMass\_Input()

{

solarSystem.neptune.mass = float.Parse(neptuneMassInput.text);

}

// Use this for initialization

void Start () {

sunMassInput.text = solarSystem.sun.mass.ToString();

mercuryMassInput.text = solarSystem.mercury.mass.ToString();

venusMassInput.text = solarSystem.venus.mass.ToString();

earthMassInput.text = solarSystem.earth.mass.ToString();

marsMassInput.text = solarSystem.mars.mass.ToString();

jupiterMassInput.text = solarSystem.jupiter.mass.ToString();

saturnMassInput.text = solarSystem.saturn.mass.ToString();

uranusMassInput.text = solarSystem.uranus.mass.ToString();

neptuneMassInput.text = solarSystem.neptune.mass.ToString();

}

// Update is called once per frame

void Update () {

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class UI\_SolarSystem : MonoBehaviour {

public SolarSystem solarSystem;

public CameraRotateAround camera;

public Text targetName;

public InputField mp;

public Text SpeedValue;

public Slider slider;

public InputField GravitiConstValueInput, GravityConstOrderInput;

public void ResetWorld\_Click()

{

solarSystem.ResetWorld();

}

public void SpeedSystem\_Input()

{

solarSystem.TimeSpeed = slider.value;

SpeedValue.text = "Скорость системы: x" + slider.value.ToString();

}

public void mass\_Input(string mass)

{

}

public void gravitConstVal\_Input(string value)

{

solarSystem.gravityConstValue = float.Parse(GravitiConstValueInput.text);

}

public void gravitConstOrder\_Input(string value)

{

solarSystem.gravityConstOrder = float.Parse(GravityConstOrderInput.text);

}

public void Time\_Pause()

{

if (Time.timeScale != 0)

Time.timeScale = 0;

else Time.timeScale = slider.value;

}

public void Start()

{

SpeedValue.text = "Скорость системы: x" + slider.value.ToString();

GravitiConstValueInput.text = solarSystem.gravityConstValue.ToString();

GravityConstOrderInput.text = solarSystem.gravityConstOrder.ToString();

}

void FixedUpdate () {

targetName.text = camera.target.name;

}

}