# Практическая работа Основы работы с Unity

**Unity** — кроссплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих 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.

На **Unity** написаны тысячи игр, приложений, визуализации математических моделей, которые охватывают множество платформ и жанров. При этом **Unity** используется как крупными разработчиками, так и независимыми студиями.

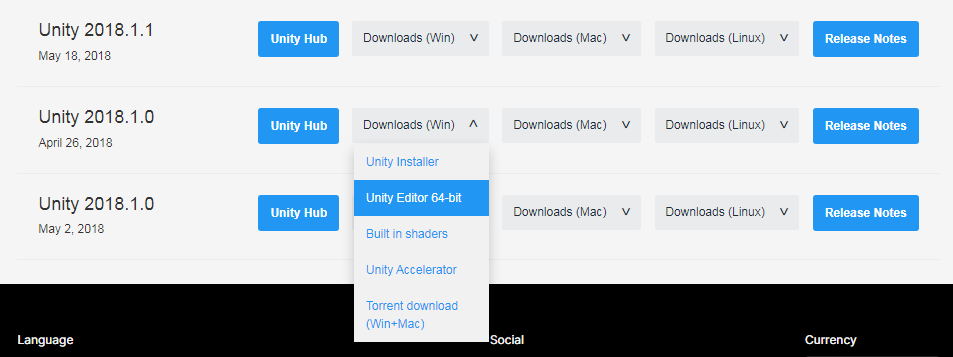
***Цель работы:*** *Создать проект моделирования взлёта и посадки ВС в  
3-мерном пространстве.*

# Порядок выполнения

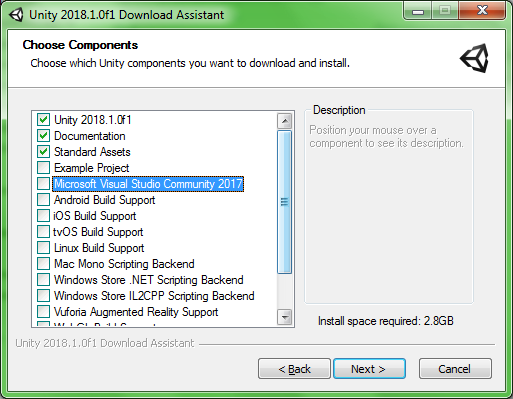
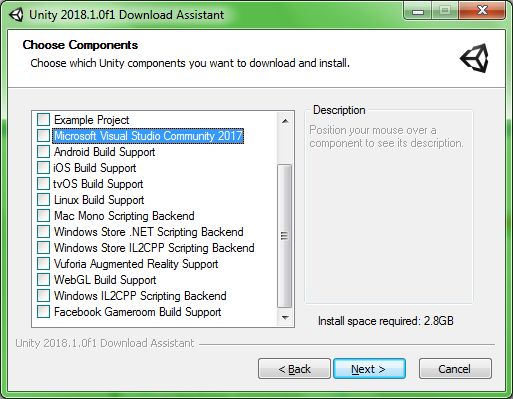
*Примечание*: данные указания подходят только для версии **Unity 2018.1.0f1**. Для других версий возможны расхождения, или могут отсутствовать какие-либо компоненты.

1. Скачиваем с официального сайта установщик Unity

<https://unity.com/releases/editor/archive>



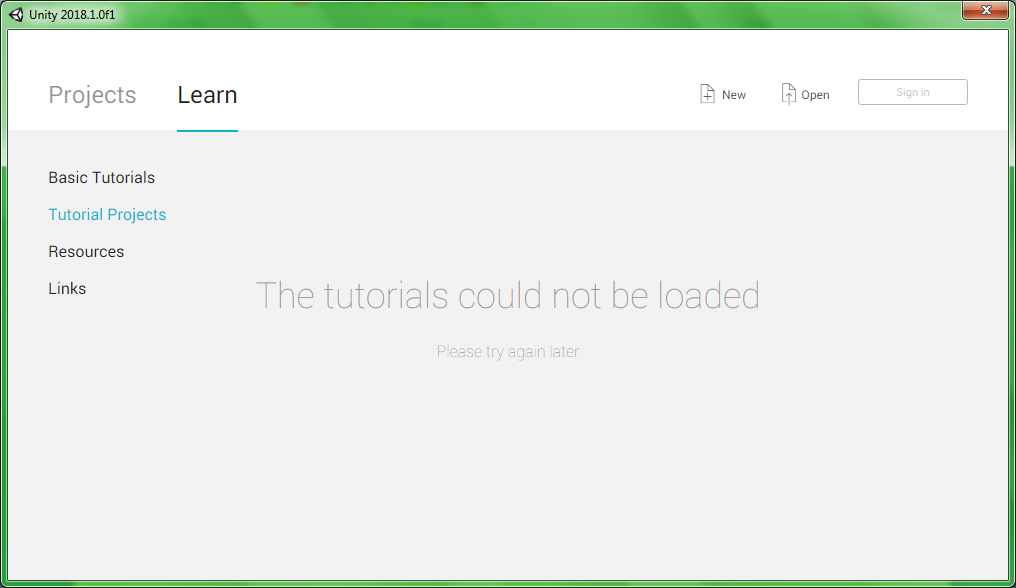
1. В установщике убираем пункт Microsoft Visual Studio Community 2017 и жмём « Next » (см. скриншоты):

После выбираем папку, куда будет производиться установка, и ждем окончания загрузки.

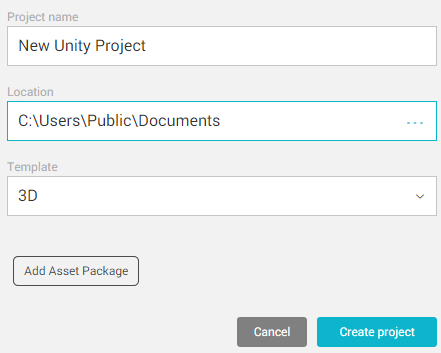
После установки желательно пройти регистрацию на сайте и войти в аккаунт в установленном приложении.

1. После установки запускаем Unity и нажимаем New:

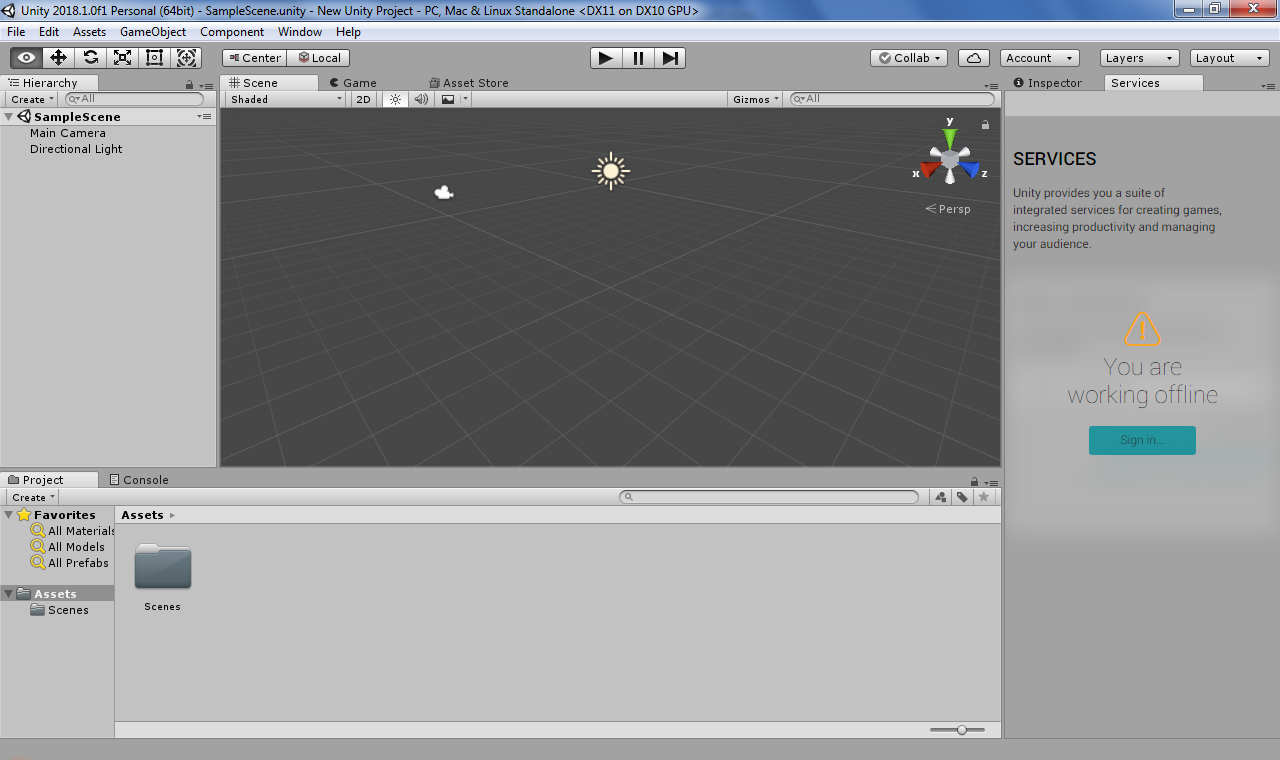
****

Если появится диалоговое окно о «Unity Pro», то включаем галочку «Не имею подписку в Unity Pro», и продолжаем.

1. Зададим «Имя проекта», его расположение, выбираем проект 3D и нажимаем Create Project:



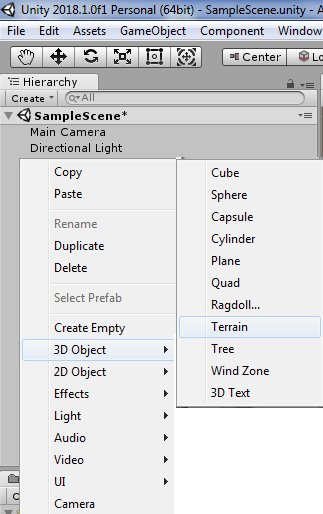
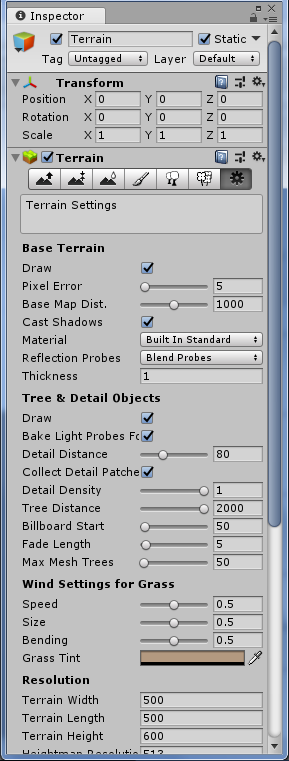
1. Подождем некоторое время, после чего откроется редактор Unity:



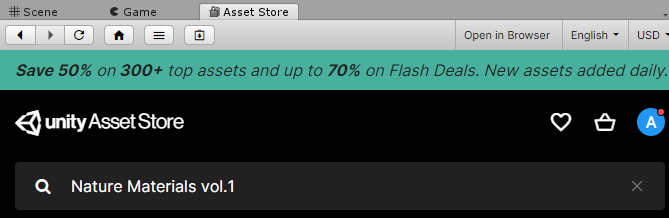
Инструменты перемещения/вращения камеры: 

1. В редакторе есть несколько панелей. Объекты мы будем создавать на панели Scene. В качестве поверхности взлёта/посадки нашего ВС создадим объект **Terrain**(*с англ. «Рельеф»*), он и будет являться поверхностью земли:

Проводим Курсор мыши на панель «**Scene**», ПКМ▬►3D Objects▬►Terrain:

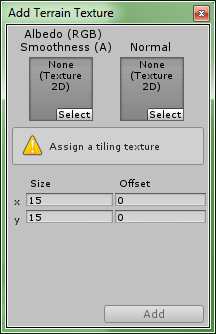


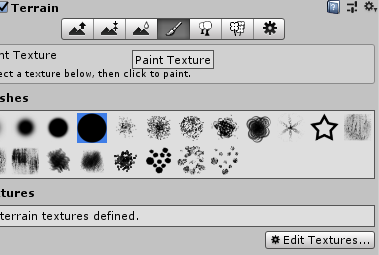
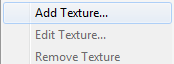
1. Дважды клик на **Terrain** и перейдем на окно **Inspector**. Здесь можно задавать свойства объектов. Нажмем на  и зададим длину и ширину объекта **Terrain**:
2. Добавим текстуру поверхности **Terrain**. В Unity в окне **Asset Store** магазин различных ассетов, где можно скачать различные платные и бесплатные модели, объекты, материалы и текстуры:



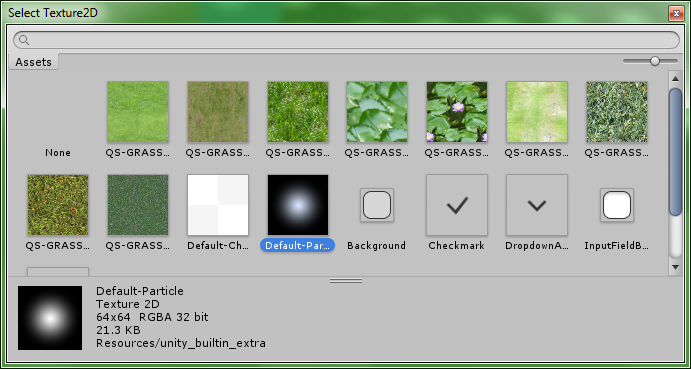
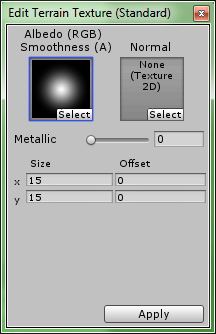
Для удобства его можно открыть в браузере. Скачаем любой набор текстур травы для заполнения объекта **Terrain**, далее импортируем в **Assets**:



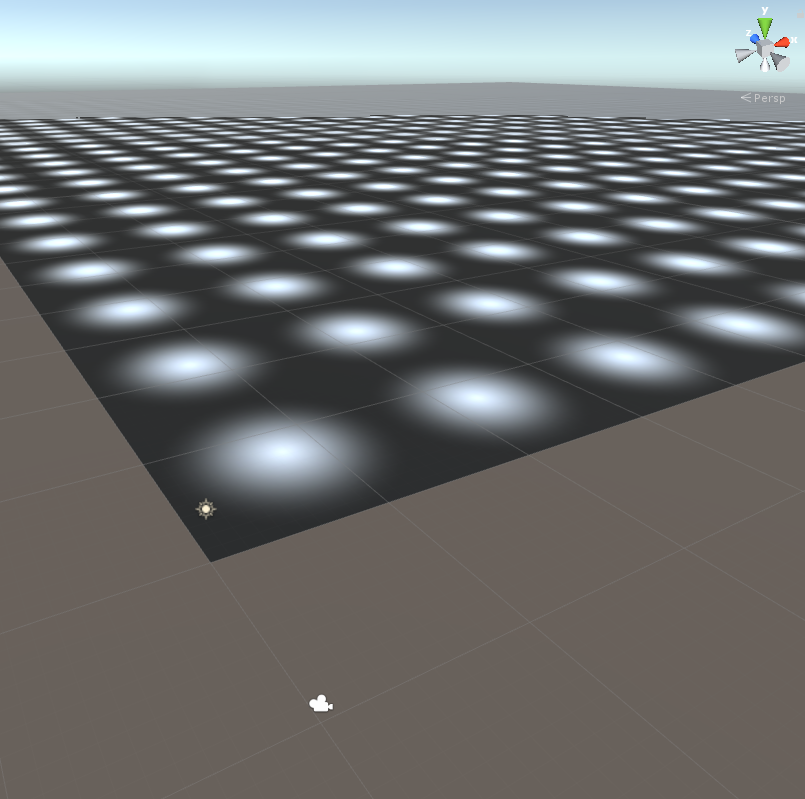
1. После импорта добавим текстуру на объект Terrain: окно **Inspector▬►  
   ▬►Paint Texture▬►Edit Textures▬►Select**



В окне **Select** **Texture2D** Выбираем рисунок▬►Закрываем **Select** **Texture2D**▬►

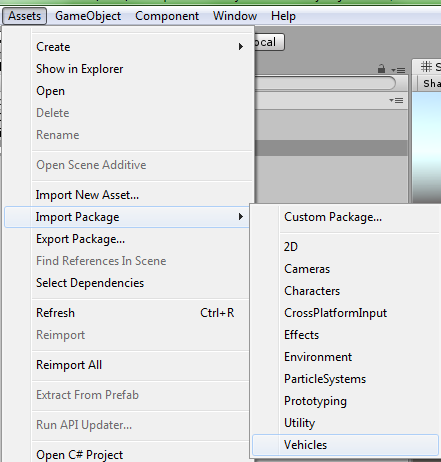
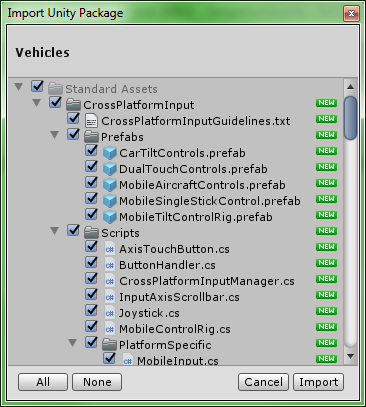
 ▬►**Apply**

Теперь объект **Terrain** имеет текстуру:

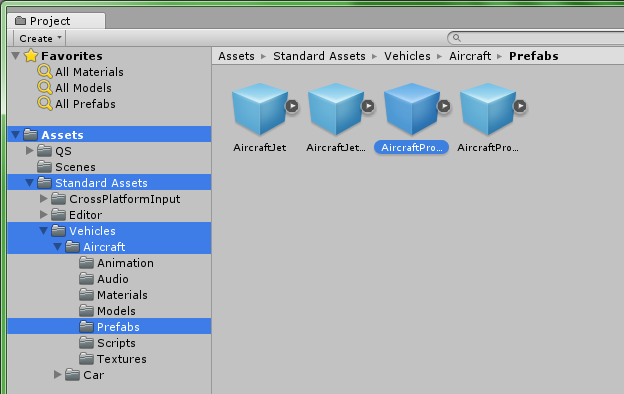


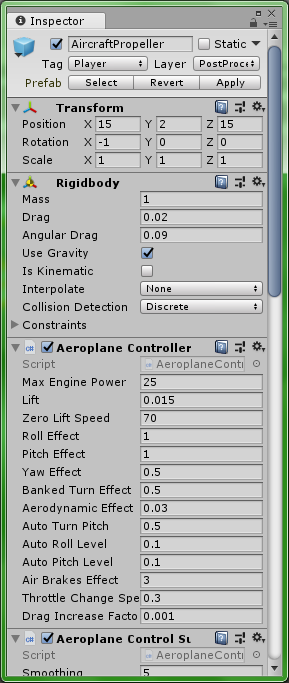
1. Импортируем объекты Vehicles, чтобы добавить ВС на текстуру:

Раздел **Assets▬►Import Package▬►Vehicles▬►Import**:

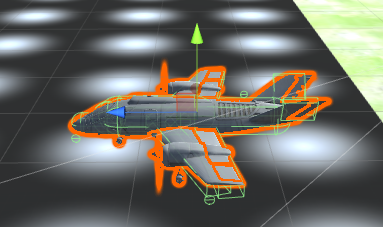
 

1. В окне Project выбираем: **Assets**▬►**Standard** **Assets**▬►**Vehicles**▬► ▬►**Aircraft** ▬►**AircraftPropeller** и ЛКМ перетащим на окно **SampleScene**:

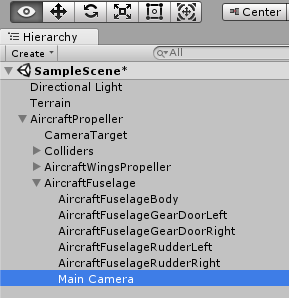
В окне **Inspector** напишем позицию ВС

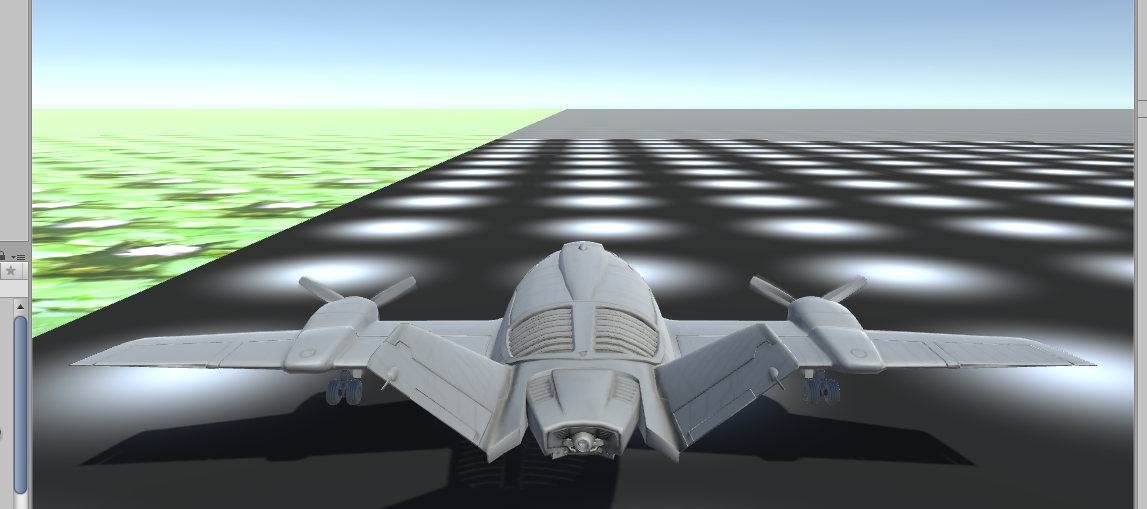


Или нажимаем на кнопку перемещения , и переместим ВС с помощью направляющих **x**(Вправо/влево), **z**(Вперёд/назад), **y**(Вверх/вниз):

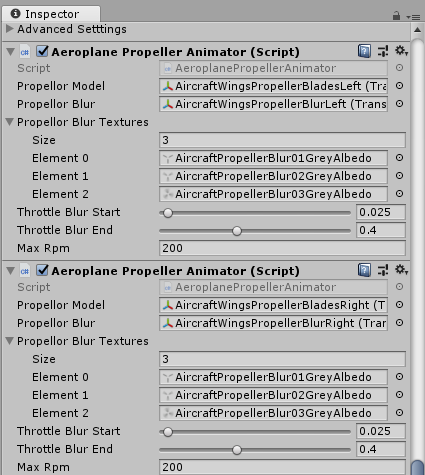


1. В окне **SampleScene** объект **Main Camera** переместим в **Aircraft** ▬►**AircraftPropeller** чтобы камера при движении самолета следовала за ним. С помощью направляющих осей отрегулируем камеру, чтобы она располагалась за самолетом. Контролировать положение камеры относительно самолета можно на вкладке **Game** или в окошке Camera Preview снизу:



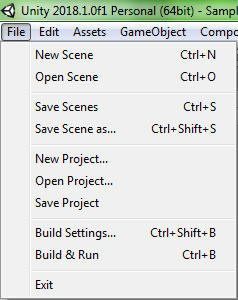
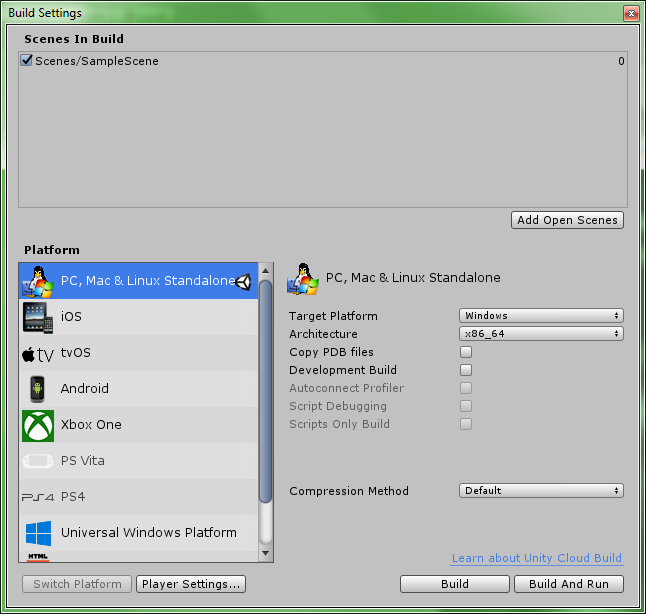
1. Добавить текстуры пропеллеров:
2. Выбираем объект **AircraftPropeller** в **SampleScene**
3. На окне **Inspector** ▬► **Aeroplane Propeller** **Animator(script)** *(их 2 свойства)* ▬► **Propeller Blur Textures** ▬►
4. Element 0 – выбираем **AircraftPropellerBlue01CreyAldebo**
5. Element 1 – выбираем **AircraftPropellerBlue02CreyAldebo**
6. Element 2 – выбираем **AircraftPropellerBlue03CreyAldebo**



1. Запустим сцену в окне **Game**, нажав .

Управление самолетом стрелками или WASD, отклонения в стороны производится перемещением курсора мыши.

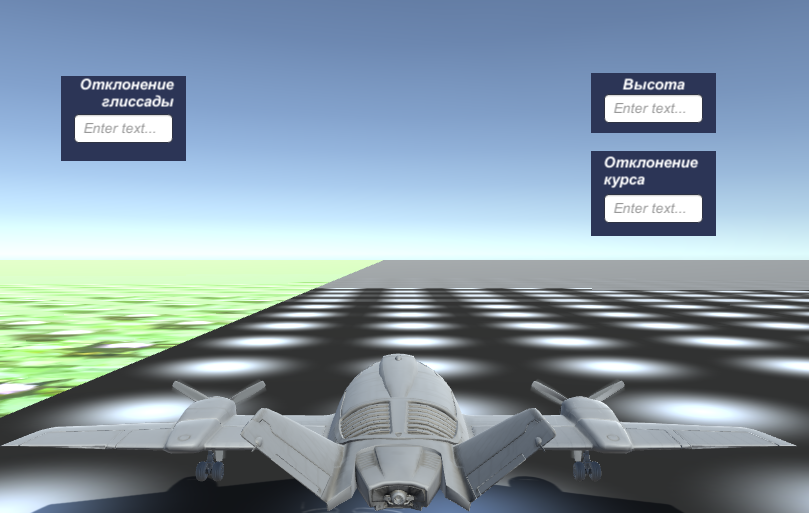
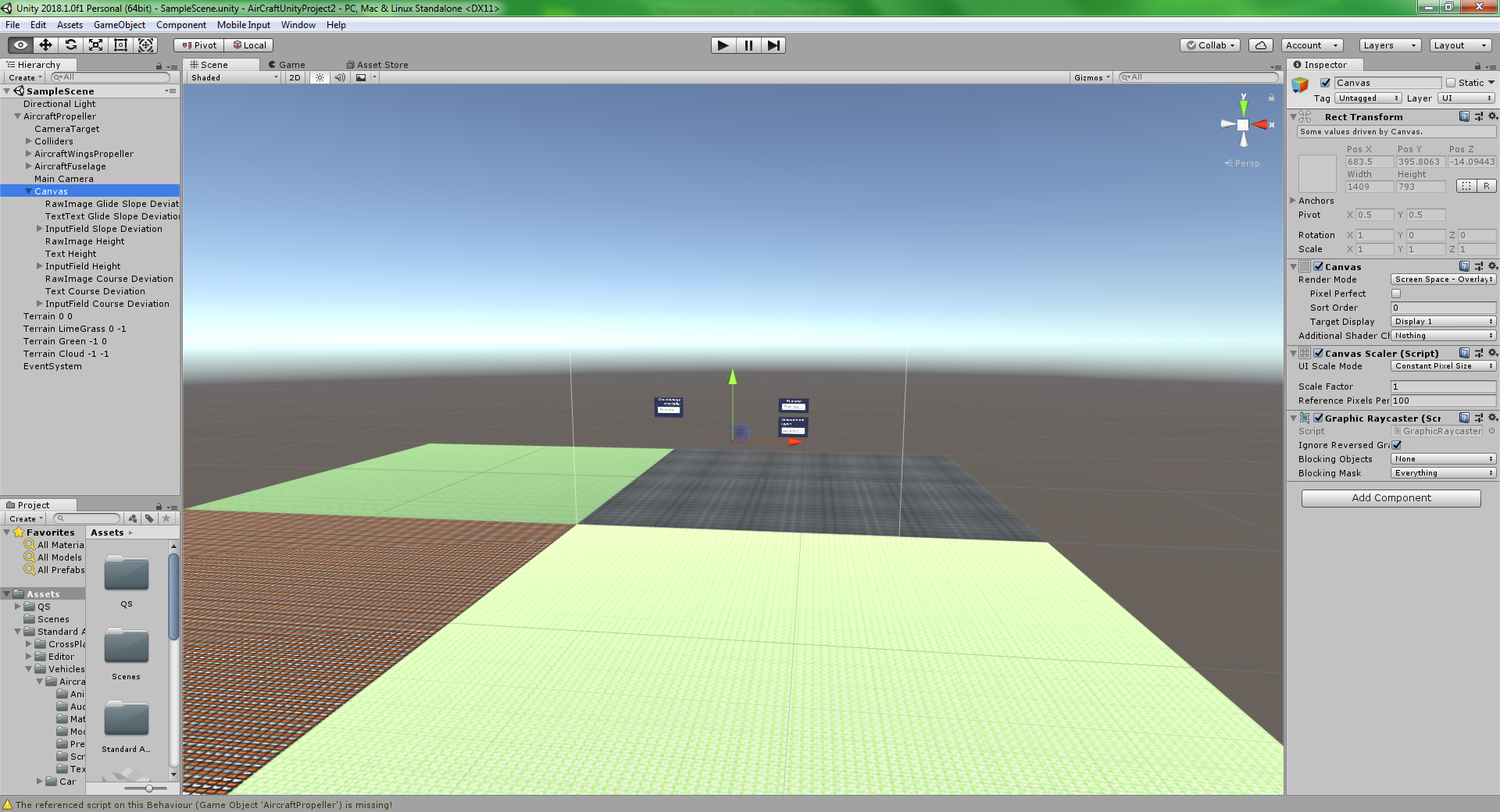
1. Сохраняем готовый проект в роле исполняемого файла (.exe): (Ctrl+Shift+B) или File ▬►Build Settings ▬►(Выберем платформу) ▬►Build

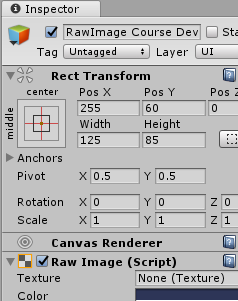
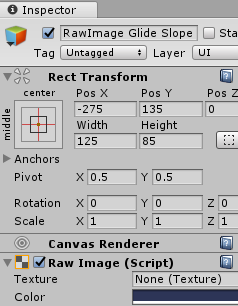
После, в папке проекта запустим исполняемый файл с именем проекта выбрав разрешение окна *[насладитесь своим творением)]*

1. Теперь для вывода информаций на экран нашей камеры(**Main Camera**) (Высота, Отклонение курса, Отклонение Глиссады)   
   добавим(создадим) объект **Canvas** внутри объекта **AircraftPropeller**: ПКМ на **AircraftPropeller**▬►UI▬►Canvas. Далее внутри Canvas создадим 9 объектов:
2. **RawImage** (*переименуем на* **RawImage Glide Slope Deviation** *по желанию*)
3. **TextText** (*переименуем на* **TextText Glide Slope Deviation** *по желанию*)
4. **Input Field** (*переименуем на* **InputField Glide Slope Deviation** *по желанию*)
5. **RawImage** (*переименуем на* **RawImage Height** *по желанию*)
6. **TextText** (*переименуем на* **TextText Height** *по желанию*)
7. **Input Field** (*переименуем на* **InputField Height** *по желанию*)
8. **RawImage** (*переименуем на* **RawImage Course Deviation** *по желанию*)
9. **TextText** (*переименуем на* **TextText Course Deviation**  *по желанию*)
10. **Input Field** (*переименуем на* **InputField Course Deviation** *по желанию*)

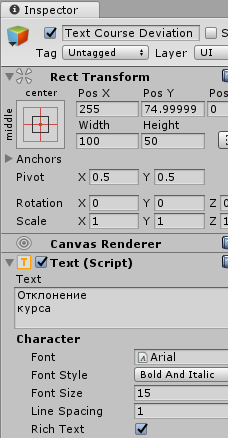
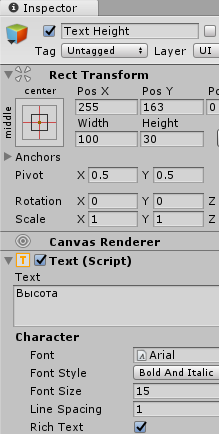
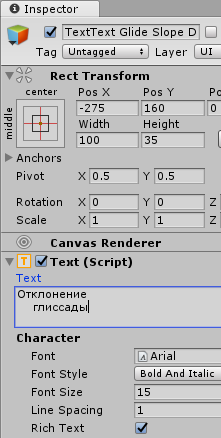
Далее в окне **Inspector** для каждого объекта отредактируем **координаты и цвета** (см. на скриншоты):



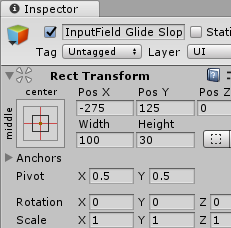
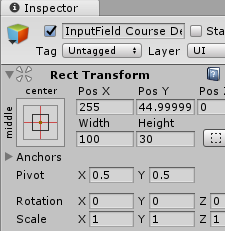
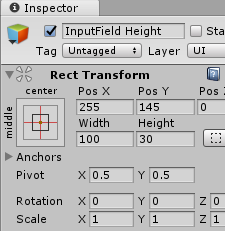
**RawImage 1, 2, 3**



**Text 1, 2, 3**



**Input Field 1, 2, 3**

1. Для осуществления процедур взаимодействия добавим 3 скрипты обработки информации. В Unity скрипты выполняются на языке C#.

На окне Project▬►Assets▬►ПКМ на папку Scenes▬►Create▬►C# Script

Зададим имя (*по желанию*) и откроем созданный скрипт в блокноте(или Notepad++) (или в IDE: VS, VS Code и т.п.) и напишем коды:

1. **GSD\_Output.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class GSD\_Output : MonoBehaviour

{

public Transform MyObject; //Обязательно, и во всех 3х файлах одинаково называется

private double Heihgt; //Расстояние по вертикали

private double GR\_GSD; //Коэффициент Глиссады

private InputField GD\_GSD; //Глиссадная дистанция

// Use this for initialization

void Start ()

{

GD\_GSD = GetComponent<InputField>();

}

// Update is called once per frame

void Update ()

{ GR\_GSD = 5.25; // По стандарту

Heihgt = Mathf.Round(MyObject.position.y);

GD\_GSD.text = (GR\_GSD \* Heihgt).ToString();

}

}

1. **Height\_Output.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class Height\_Output : MonoBehaviour

{

public Transform MyObject; //Обязательно, и во всех 3х файлах одинаково называется

private InputField Height; //Расстояние по вертикали

// Use this for initialization

void Start()

{

Height = GetComponent<InputField>();

}

// Update is called once per frame

void Update()

{

Height.text = Mathf.Round(MyObject.position.y).ToString();

}

}

1. **CourseD\_Output.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

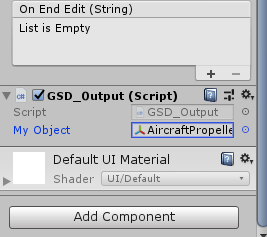
using UnityEngine.UI;

public class CourseD\_Output : MonoBehaviour

{

public Transform MyObject; //Обязательно, и во всех 3х файлах одинаково называется

private InputField CourseD; //Расстояние по вертикали

 // Use this for initialization

void Start ()

{

CourseD = GetComponent<InputField>();

}

// Update is called once per frame

void Update ()

{

//D=(Xm−Xt)/Xt\*100

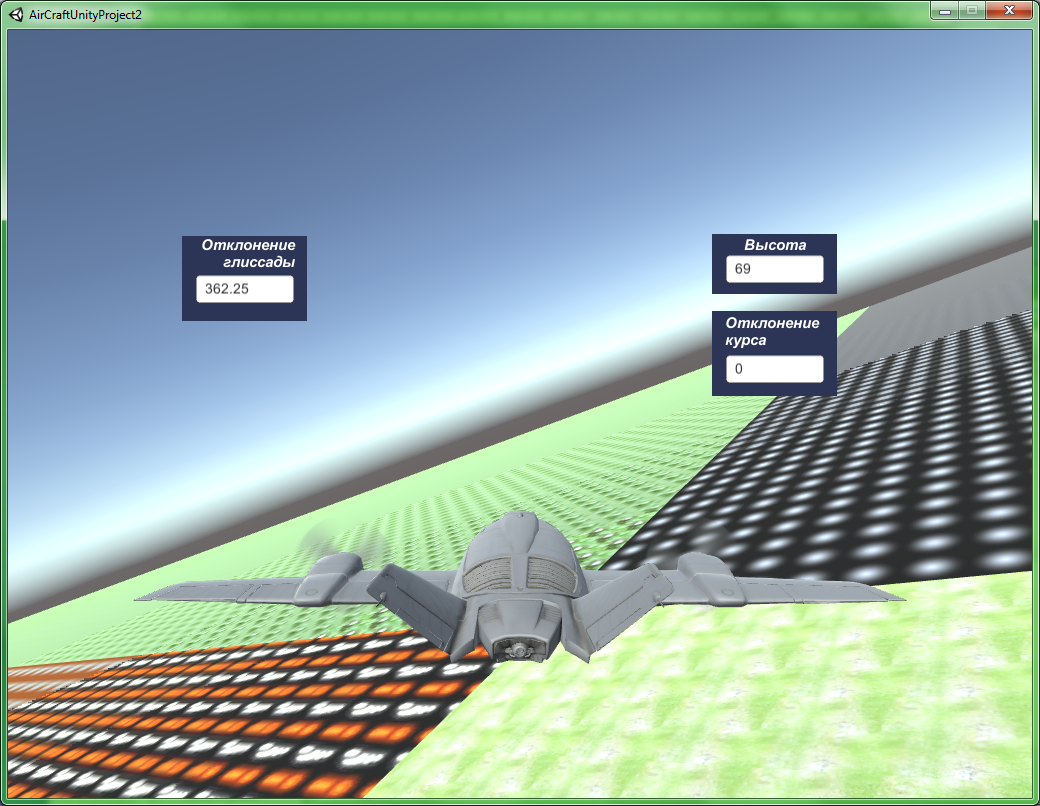
CourseD.text = "";

}

}

1. Теперь «.cs» файлы(скрипты) перетаскиваем на объекты **InputField** на окне **SampleScene**, далее в окне **Inspector** для каждого **InputField** в свойстве **MyObject** укажем **AircraftPropeller** (см. на верхний скриншот).

Теперь запустим и проверим отображение значений:



*Если не отображаются, то необходимо пересоздать объекты* ***InputField****.*

Похожим образом создаются другие скрипты для обработки различных событий (отображение элементов на Canvas, воспроизведение звуков, перемещение элементов на карте, изменение цветов элементов, свойств и многого другого). Подробнее о скриптах: <https://docs.unity3d.com/ru/530/Manual/UnityManual.html>

# Задание:

* 1. Изучить материалы по работе в Unity и по скриптам для Unity на C# (документация, статьи в Интернете, видеоуроки или видеоразборы, форумы и т. д.)
  2. Смоделировать в Unity работу какой-либо системы, используемой на ВС в гражданской авиации (система ILS, огни PAPI, светосигнальные системы, система предупреждения столкновения с землей, автоматический радиокомпас и т. д. ; Интернет в помощь); можно использовать модель самолета из стандартных ассетов

Ссылки в помощь: <https://docs.unity3d.com/ru/530/Manual/UnityManual.html> <https://itproger.com/course/unity-games/4> <https://habr.com/ru/articles/437898/> <https://itvdn.com/ru/blog/article/create-skill-bar-part1> <https://habr.com/ru/articles/246737/> <https://jwinters.ru/unity3d/unity-ui/> <https://skillbox.ru/media/gamedev/chto_takoe_assety_unity/>