**Тема 11. Программирование трансформации и генерации 3D-объектов**

**Раздел 1. Программирование трансформации 3D-объектов**

*Почитайте и используйте в дальнейшем книгу о программировании в Unity* ***Дж. Хокинг «Unity в действии»****, которую вы получаете вместе с* ***Темой 11***.

1. Создать новый проект **Unity** с ресурсами, присоединяемыми по умолчанию.
2. Создать на сцене 3D-объект типа **Cube**. Добавить к кубу компоненту **Script**, выполнив для этого на панели **Inspector** команду **Add Component/Script/New Script**, и задать ему имя **ScriptPosition**.
3. Активизировать двойным щелчком мыши по имени скрипта в добавленной компоненте редактор кода C# **MonoDevelop** (*или* ***Visual Studio*** *в зависимости от настройки среды Windows*) и создать для методе **Update()** программный код, выполняющий перемещение объекта по оси X (через свойство **transform.position** и вектор **Vector3(x,y,z)** ) при запуске сцены на выполнение **Play**.

*Выполнение действий кода в этой сцене и в последующих заданиях проверяйте каждый раз запуском сцены на выполнение* ***Play****!!!*

* Замедлить (ускорить) движение, изменив шаг добавления координаты с каждым кадром так, чтобы объект исчез за край экрана примерно через 2 секунды.
* Добавить в код перемещение объекта по другим осям координат с разными скоростями и в разных направлениях.
* Сохранить сцену в проекте под именем «**Движение**».

1. Создать новую *вторую сцену с именем «****Вращение****»* и на ней 3D-объект типа **Cube**, расположить его в левой части сцены относительно камеры и растянуть в горизонтальной плоскости.
2. Добавить к новому объекту компоненту **Script,** задать ему имя **Rotation\_Rotate,** создать для метода **Update()** программный код, выполняющий вращение этого объекта вокруг вертикальной оси **Y** с использованием метода **Rotate()** при запуске сцены на выполнение **Play.**
3. Создать на этой же сцене второй 3D-объект **Cube,** расположить его в левой части сцены и растянуть по вертикальной оси. Создать для этого объекта в новой компоненте **Rotation\_Euler** программный код, выполняющий вращение объекта вокруг двух осей **X** и **Z** одновременно, но вращение будет зависеть от последовательности предыдущих поворотов по осям **X** и **Z**.
4. Создать на этой же сцене третий 3D-объект **Cube,** расположить его в правой части сцены и растянуть по вертикальной оси. Создать для этого объекта в новой компоненте **Rotation\_Quaternion** программный код, выполняющий вращение объекта вокруг двух осей **X** и **Z** одновременно и независимо:

* в новом классе перед методом **Start()** определить переменную типа **Quaternion** для фиксации начального поворота (угол и ось поворота) и переменную типа **float** для значения угла поворота;
* в методе **Start()** определить переменную типа **Quaternion** задав ей начальный поворот объекта (через свойство **transform.rotation)**;
* в методе **Update()** задать приращение угла поворота, определить последовательно углы поворота вокруг обеих осей, используя ключевые вектора **Vector3.right** и соответственно **Vector3.forward**, и, наконец, задать общий угол поворота объекта вокруг двух осей за каждый такт выполнения методе **Update().**
* Расположить рядом два объекта, которые вращаются вокруг осей **X** и **Z** одновременно с одинаковыми скоростями, но с использованием разных методов: свойства **eulerAngles** и **Qauternion**, задать для них одинаковые пропорции и начальное расположение на сцене, и после запуска **Play** убедиться, что их вращение различаются.
* Создать с помощью кватерниона вращение вокруг произвольной оси в 3D-пространстве, используя полную конструкцию 3D-вектора **Vector3(x,y,z),** где параметры задают проекции вектора на оси координат (не угол поворота!).
* Сохранить сцену в проекте.

1. Создать *третью сцену с именем «****Масштабирование****»* и на ней 3D-объект типа **Cube**, расположить его в левой части сцены относительно камеры, передвигая его вдоль оси **X** влево. В правой части сцены раположить 3D-объект типа **Sphere**, передвинув его вдоль оси **X** вправо. Добавить для куба скрипт с именем **ScriptScale** и добавить в него для метода **Update()** код, выполняющий масштабирование объекта по оси **X** (через свойство **transform.localScale** и вектор **Vector3(x,y,z)** с указанием в нем приращения по оси **X**).

* Добавить к обоим объектам на сцене компоненту **Physics/Rigidbody** и снять галочку со свойства **Gravity**, чтобы объекты не «падали» вниз – куб будет толкать шар.
* Расположить куб в центре сцены. Создать 3 копии объекта Sphere (копированием их в Буфер обмена и вставляя их из него) и расположить сферы вокруг куба по всем его 4-м сторонам, передвигая их по осям в плоскости, перпендикулярной оси зрения из камеры (удобнее выставить соответствующим образом координаты объектов в **Inspector** на панели **Transform/Position**).
* Добавить масштабирование куба по вертикальной оси **Y** – при запуске сцены на выполнение **Play** расширяющийся куб будет раздвигать сферы в стороны.
* Сохранить сцену в проекте.

1. Создать в проекте четвертую сцену с именем «**Движ\_клавишами**», разместить на ней 3D-объект типа **Capsule** и добавить к нему скрипт **Move\_Key**, в котором записать программный код в методе **Update(),** который обеспечит возможность перемещение по всем осям координат **X, Y, Z** при нажатии на соответствующие клавиши клавиатуры с помощью конструкции **Input.GetKey(KeyCode.Q)**, когда, например, происходит проверка нажатия на клавишу **Q**. Используйте для перемещения вверх-вниз (по оси **Y**) объекта клавиши **W** и **S**, влево-вправо **A** и **D** (по оси **X**)**,** а для приближения и отдаления (по оси **Z**) объекта клавиши **Q** и **E.**
2. Создать в проекте пятую сцену «**Константы\_движения**», в которой разместить куб и создать для него программный код в скрипте **Input\_GetAxis**, обеспечивающий:

* его перемещение по горизонтали и вглубь с помощью клавиш клавиатуры с использованием конструкции **transform.Translate (x, 0, z);**
* значения **x** и **z** заданы с помощью созданных в классе двух переменных типа **float**, которые определяются в методе **Update()** конструкциями **Input.GetAxis("Horizontal")** и **Input.GetAxis("Vertical")** соответственно.
* его вращение в горизонтальной и вертикальной плоскости с помощью мыши с использованием **transform.Rotate(x,y,0);**
* значения **x** и **y** заданы с помощью созданных в классе двух переменных типа **float**, которые определяются в методе **Update()** конструкциями **Input.GetAxis("Mouse Y")** и **Input.GetAxis("Mouse X")** соответственно.
* ограничить вращение объекта мышью по вертикальной оси от 0 до 90 градусов с помощью конструкции **Mathf.Clamp(угол, нижн. предел, верх. предел),** переопределив с ее помощью угол для вращения по вертикали.

**Раздел 2. Программное создание объектов на сцене**

1. Создать новую сцену и добавить на нее 3D-объект типа **Plane** – горизонтальную плоскость.
2. Добавить для плоскости компоненту **Rigidbody**, убрать в ней гравитацию **Gravity** и установить режим **Is Kinematic** (режим необходим для управления из программного кода объекта);
3. Добавить для объекта **Plane** компоненту **Script** с программным кодом, генерирующим по нажатию на клавишу клавиатуры рождение на сцене нового объекта (например, кубика) над плоскостью и падающего на нее с некоторой высоты (используйте материалы **Лекции 11** о программной генерации объектов):

* определить границы плоскости и координаты точки на сцене для генерации объекта;
* задать в функции **Update()** условие нажатия заданной клавиши на клавиатуре, например, **Q** для генерации объекта;
* сгенерировать объект (куб) при выполнении условия в указанной точке с компонентой **Rigidbody**.

1. Запустить приложение и сгенерировать нажатием заданной клавиши несколько десятков кубов, при этом некоторые из них после столкновений будут «сваливаться» с плоскости.
2. Добавить в функцию **Update()** запускнаклона плоскости вокруг оси **Z**, например при нажатии клавиши **W**, и «сбросить» с нее оставшуюся на плоскости горку кубиков.
3. Сохранить сцену под именем «**кубики**».
4. Создать на сцене 3D-объект типа **Sphere** идобавить к немукомпоненту **Rigidbody**, оставив все другие параметры по умолчанию.
5. Перетяните объект **Sphere** из окна **Hierarchy** в окно **Project** в папку **Assets**, таким образом у вас получится префаб (**Prefab),** другими словами шаблон сферы (используйте в дальнейшем материалы **Лекции 11** о создании и использовании префабов).
6. Из окна **Hierarchy** удалите объект **Sphere** – его не должно быть на сцене, он будет рождаться каждый раз на сцене программно из префаба (шаблона) при нажатии на заданную клавишу.
7. Добавить в программный код скрипта для плоскости следующие действия (координаты точки генерации сферы и размеры плоскости, куда она будет падать используем такие же как и у кубиков):

* определить публичную переменную **public GameObject prefub1**, в которой будет храниться префабсферы (поскольку публичная переменная **prefab1** объектная, то в компоненте **Plane(Script)** Инспектора необходимо указать в строке **Prefab 1,** с каким объектом на сцене эта переменная связана – нажмите кружочек справа и в появившемся окне в списке **Assets** выберите созданный префаб **Sphere**);
* в методе **Update()** создать условный оператор, проверяющий событие нажатия на клавишу, например, пробела **Space**, в котором задан метод **Instantiate()**, генерирующий экземпляр нового объекта из префаба с указанием в своих 3-х аргументах: ссылки на префаб, место создания экземпляра и угол его поворота, относительно исходного положения.

1. Запустить приложение и сгенерировать нажатием заданной клавиши **Space** «шарики», падающие на плоскость.
2. Сгенерируйте поочередно кубики и сферы с падением их на плоскость, наклоните плоскость нажав клавишу **W** для сброса с нее всех оставшихся на ней объектов.
3. Сохраните конечную сцену под именем «**Генерация объектов**».