**Тема 3. Программирование генерации 3D-объектов и обработка щелчков мышью**

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** с 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**».
4. Создать новую сцену с именем «Щелчки мышью». Добавить в окно **Hierarhy** новый объект, отвечающий за обработку событий, выполнив команду **Create/UI/Event System** (объект на сцене невидим)**.**
5. Создать на сцене новый стандартный 3D-объект типа **Cube, д**обавить к объекту компоненту **Script,** задать ему имя и активизировать двойным щелчком мыши редактор кода **C#**.
6. Добавить в сценарий строку использования системы событий **using UnityEngine.EventSystems;**.
7. Добавить к базовому классу **MonoBehaviour** через запятую новый класс интерфейса системы обработки событий щелчков мышью по объектам **IPointerClickHandler**.
8. Вместо имеющихся в исходном коде функций **Start()** и **Update()** вызвать встроенную в **Unity** функцию обработки щелчков мышью по объектам сцены

**public void OnPointerClick(PointerEventData eventData)**

которая обрабатывает щелчки мыши по объектам сцены.

1. Написать для этой функции код задания цвета по схеме **RGB** случайным образом, например:

**float red = Random.Range(.0f, 1.0f);**

а также для **green**, **blue** и задать цвет для игрового объекта (аналогично п. 5) при каждом щелчке по нему мышью:

**gameObject.GetComponent<Renderer>().material.color = col1;**

где **col1** – объектная переменная типа **Color** из одноименного класса **new Color**( **red, green, blue)**.

1. Запустить приложение командой **Play** и проверить работу созданной функции.
2. *(\*, следующие пункты задания выполнить факультативно)*. Для выполнения другого действия – сдвига объекта при щелчке по нему мышью необходимо добавить к игровому объекту **Cube** компоненту **Rigidbody**, а также создать еще один стандартный объект **Plane** для того, чтобы игровой объект опирался на плоскость и не падал с нее под действием гравитации (сдвиньте вверх куб на половину его высоты).
3. Добавить для объекта **Main Camera** компоненту **Physics RayCaster**, которая проецирует щелчки мышью по 2D-экрану в щелчки мышью по 3D-объектам на сцене по всей ее глубине.
4. Добавить в функцию **OnPointerClick(PointerEventData eventData)** код для двух переменных типа **Vector3 -** вектора определяющего направление щелчка мышью по объекту, например **target:**

**Vector3 target=eventData.pointerPressRaycast.worldPosition;**

1. где **pointerPressRaycast –** задает направление на точку удара в глобальных координатах **worldPosition** сцены, и второго вектора, определяющего направление удара, как направление от камеры, например **collid:**

**Vector3 collid = Camera.main.transform.position;**

1. Задать также переменную целого типа, например **forse,** определяющей силу удара (можно для начала задать значение 100, а затем увеличивать его по мере необходимости).
2. Для того, чтобы согласовать направление удара и величину силы, с которой происходит удар в месте щелчка по объекту мышью, необходимо задать новый вектор **Vector3,** например **distance**,как разность двух векторов **target – collid** и нормализовать его (т. е. задать ему единичную длину) с помощью свойства 3D-векторов **normalized**.
3. Переопределить вектор направления удара **collid** через вектор **distance**:

**Vector3 collid=distance\*forse;**

где значение переменной **forse** задает силу удара.

1. Применить для компоненты игрового объекта **Rigidbody** встроенныйметод **AddForceAtPosition** обработки ударас параметрами **collid** и **target**

**gameObject.GetComponent<Rigidbody>().AddForceAtPosition(collid, target);**

1. Запустить приложение командой **Play** и проверить работу добавленного кода функции **OnPointerClick**.
2. Создать **Prefab** для объекта **Cube,** перетащив его из окна **Hierarhy** в окно **Assets** в нижней части экрана.
3. Клонировать куб со всеми его свойствами, включая связанный с ним программный код, перетаскивая на сцену контейнер **Prefab,** создавая тем самым экземпляры куба на сцене. Запустить приложение и, щелкая по объектам мышью, убедиться, что клонированные объекты под действием щелчков мышью ведут себя аналогичным образом.
4. Создать из клонированных кубов стенку из 4х8 (4 в высоту и 8 в ширину) клонированных объектов и, запустив приложение, разрушить ее постепенно щелчками мышью. Сохранить законченную сцену.