Лабораторная работа №10. Создание и управление моделью танка в Unity

1. В 3DsMAX создайте на основе примитивов модель танка из трех составляющих частей: корпус, башня, ствол, задавая им соответствующие имена. При создании танка необходимо изменить его составляющим Pivot в соответствии с рисунками 1 и 2, чтобы в дальнейшем направление осей совпадало с осями в редакторе Unity.

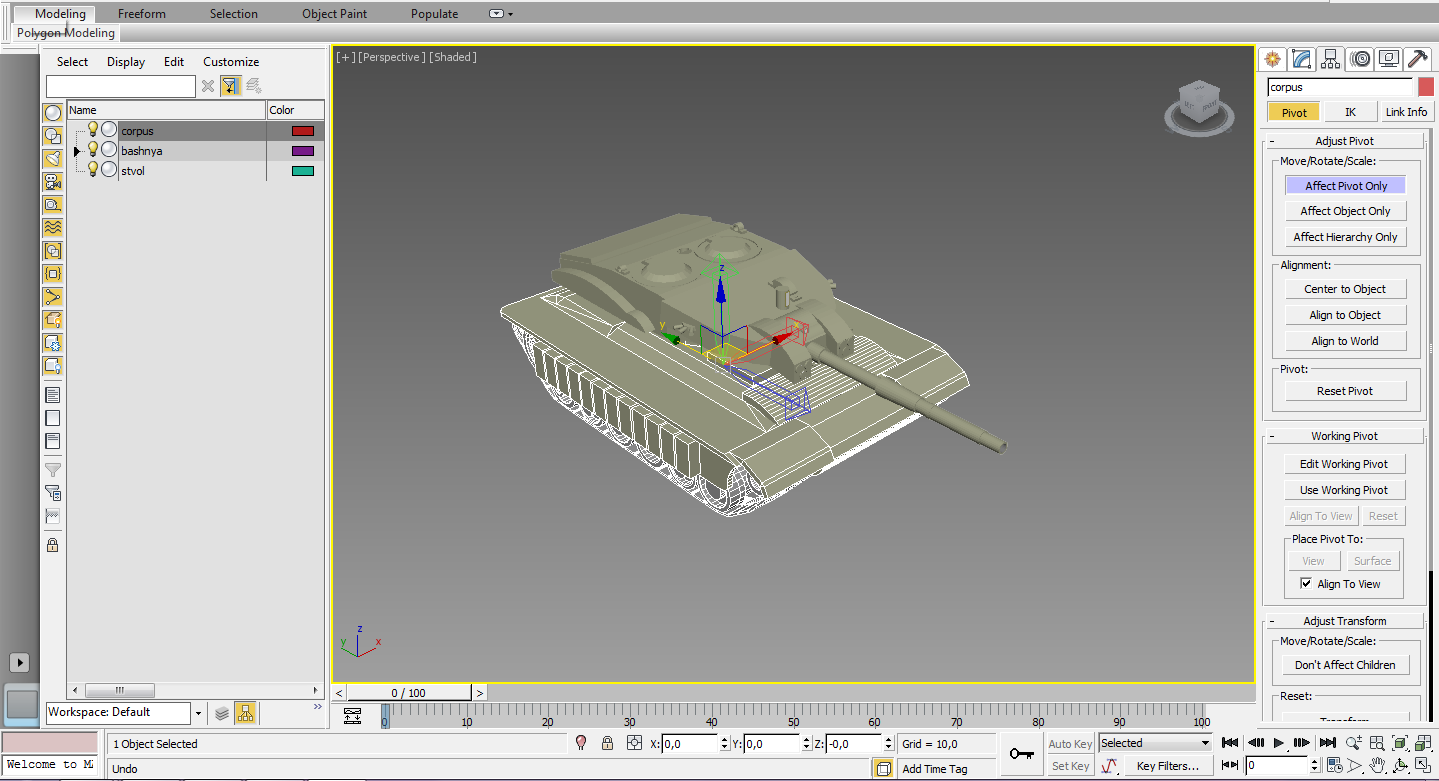


Рисунок 1 – Pivot корпуса танка 3DsMAX

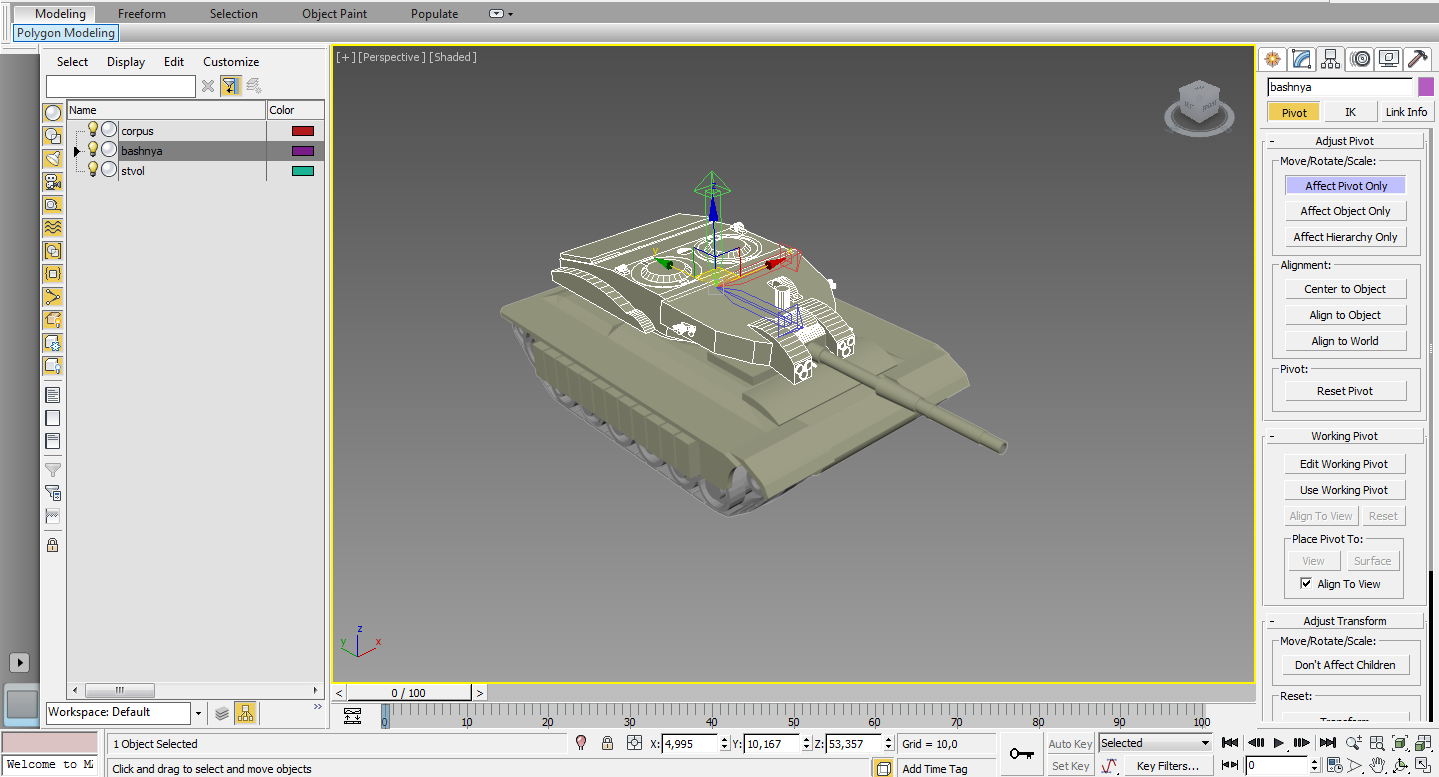
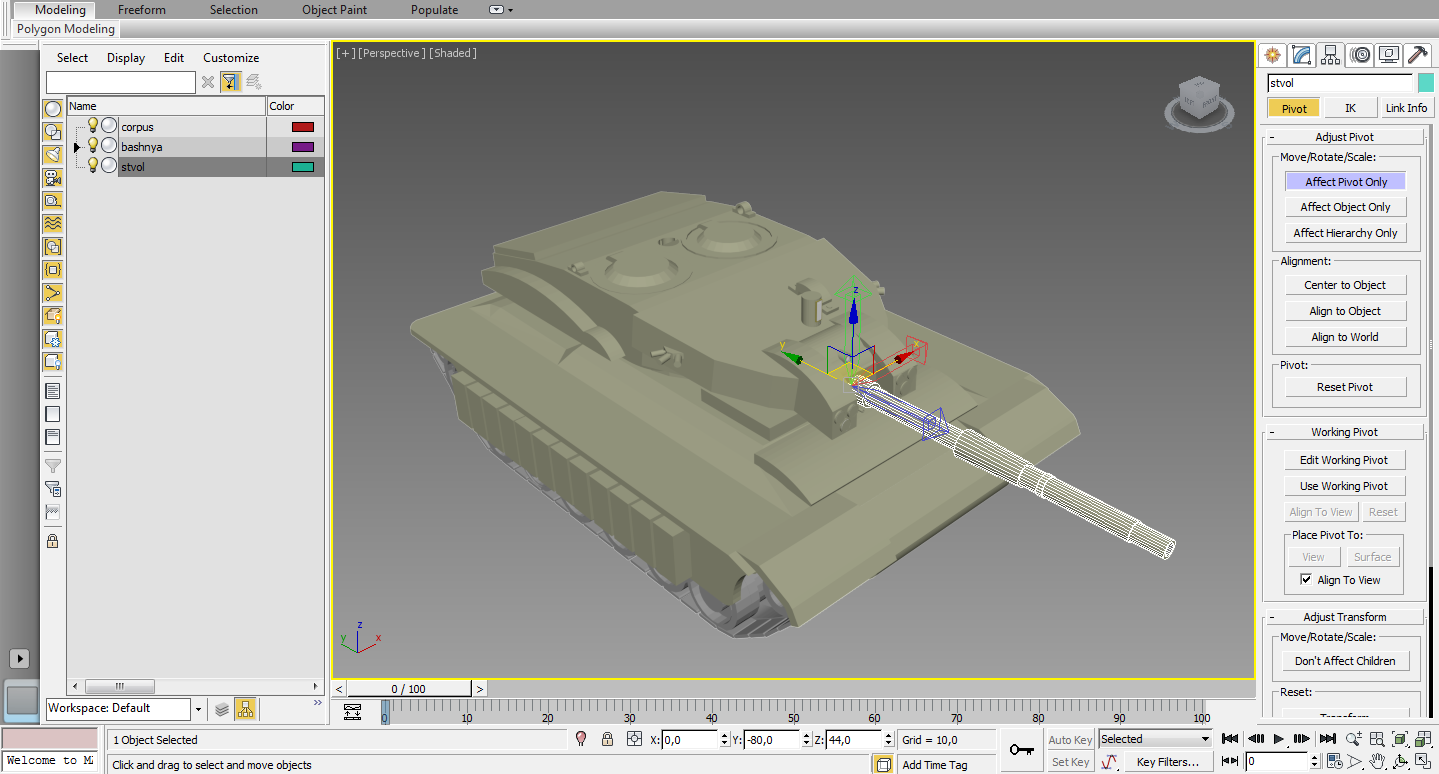
 

Рисунок 2 – Pivot башни и ствола танка 3DsMAX

1. Отредактируйте поверхность модели танка для придания реалистичности и наложите на них соответствующие текстуры. Сохранить модель танка в формате 3DS или FBX.
2. Создать в среде Unity холмистую поверхность Terrain и наложить соответствующие текстуры. Импортируйте модель танка в среду Unity. При импорте танка его Pivot должен быть в соответствии с рисунками 3 и 4:

Рисунок 3 – Pivot башни и ствола танка Unity

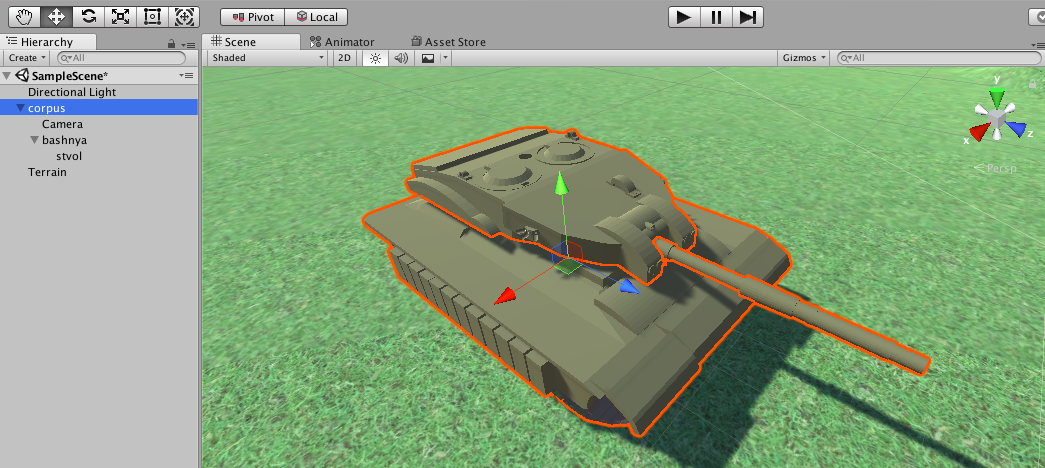


Рисунок 4 – Pivot ствола танка Unity

1. Создать иерархию подчиненности частей танка в Unity: корпус->башня->ствол (рисунок 5). То есть в танк должен быть вложен корпус, а затем башня, а в башню вложен ствол. Кроме того, при программном управлении движением 3D-объектов необходимо учесть разную направленность осей координат в Unity (вертикальная ось – Y, а в горизонтальной плоскости – оси X-Z) и 3DsMAX (вертикальная ось – Z, а в горизонтальной плоскости – оси X-Y), что, как правило, приводит к необходимости использовать требуемое направление (Vector3.forward, Vector3.right).



Рисунок 5 – Иерархия составляющих танка в Unity

1. Обратите внимание на данные моменты при импорте:

* Если импортированный танк не соответствует размерам созданного Terrain, необходимо уменьшить **Scale Factor** танка. Не рекомендуется его уменьшать с помощью инструмента масштабирования в Unity!
* Если текстура объекта легла неправильно (например, повторяется много раз), необходимо изменить значение **Tilling** в материале текстуры.

1. Камеру расположите за танком, внутри самого корпуса (рисунок 6).



Рисунок 6 – Расположение камеры в Unity

1. Таким образом, у вас получится иерархия в соответствии с рисунком 7 («Tank» – это объект «corpus», «Turret» – это объект «bashnya», «Cannon» – это объект «stvol», они были переименованы для удобства).



Рисунок 7 – Иерархия, включающая камеру, камерой составляющих танка в Unity

1. Добавьте компонент **Rigidbody** танку и задайте ему массу(Mass) около 1000.
2. Добавьте компонент **Box Collider** на корпус танка (объект Tank на рисунке 7) и на башню (объект Turret на рисунке 7).
3. Создайте скрипт **«Tank»** и добавьте его в качестве компонента модели танка (то есть на верхний объект иерархии танка, в нашем случае это его объект «Tank» на рисунке 7).
4. Создайте переменные типа **float** для скорости **MoveSpeed** и поворота танка **RotateSpeed**.
5. Добавьте в метод **Update()** движение танка вперед-назад, каждый кадр прибавляя к  текущему положению танка смещение в сторону направления его локальной оси Z **(Vector3.forward)**, с применением преобразования из локальных координат в глобальные **(TransformDirection)**, скорость **(MoveSpeed)** и ввод с клавиатуры **Input.GetAxis("Vertical")**. В итогеперемещение реализуется следующим образом:

transform.position += transform.TransformDirection (Vector3.forward \* MoveSpeed \* Input.GetAxis("Vertical"));

1. Добавьте в метод **Update()** выполнение вращения танка вокруг вертикальной оси Y влево-вправо при условии нажатия соответствующих клавиш через метод **Rotate()**:

transform.Rotate(0f, Input.GetAxis("Horizontal") \* RotateSpeed, 0f);

1. Создайте скрипт **«Turret»** для вращения башни вокруг вертикальной оси, добавив переменную для хранения скорости (**float turretSpeed**), а в метод **Update()** добавьте для вращения по оси Y метод **Rotate()**, с использованием перемещения мыши влево-вправо по оси Х и скорости.
2. Поместите скрипт **«Turret»** в качестве компонента на башню (объект «Turret» на рисунке 7).
3. Создайте скрипт для вращения ствола **«Cannon»**, который нужен для реализации перемещения мыши вверх-вниз с ограничением по диапазону угла поворота.
4. Объявите переменные **float min** (минимальный угол) и **float max** (максимальный угол), которые определите самостоятельно, а также **float current\_angle** – текущий угол смещения в градусах. Также создайте переменную типа **Quaternion** с именем **startRotation** и инициализируйте её в методе **Start()** записав в неё текущий локальный поворот ствола.
5. В метод **Update()** прибавляйте к текущему углу смещение мыши по вертикали, затем с помощью метода **Clamp()** проверьте, попадает ли ваш угол в указанный диапазон, после создайте новое вращение в кватернионах по оси X, указав, на какой угол необходимо повернуть и примените данное вращение. Таким образом, это получатся следующие условные конструкции:

     current\_angle += Input.GetAxis("Mouse Y");  
        current\_angle = Mathf.Clamp(current\_angle, min, max);  
        Quaternion newRotation = Quaternion.AngleAxis(current\_angle, new Vector3(1, 0, 0));  
        transform.localRotation = startRotation \* newRotation;

1. Поместите скрипт **«Cannon»** как компонент на ствол (объект «Cannon» на рисунке 7).
2. Запустите проект и проверьте, что реализовано перемещение танка, поворот его башни по горизонтальной оси и вращение ствола по вертикальной оси.