Tutorial – Scripting

For this tutorial we will start creating the Tank game that forms your first piece of assessment for this course.

The project we make in this session will start of simple, and we'll add more functionality to it over the next several lessons until we have a finished tank game.

There will probably be a few times where you want to expand on what is presented in these tutorials and do your own thing. We encourage you to explore Unity and customise the game you make, but we also recommend keeping a version of your project that strictly follows the tutorial. That way if your own game suddenly breaks you'll always have a version that matches the tutorial and can be used to pass the assignment.

With that said, let's start making our tank game…

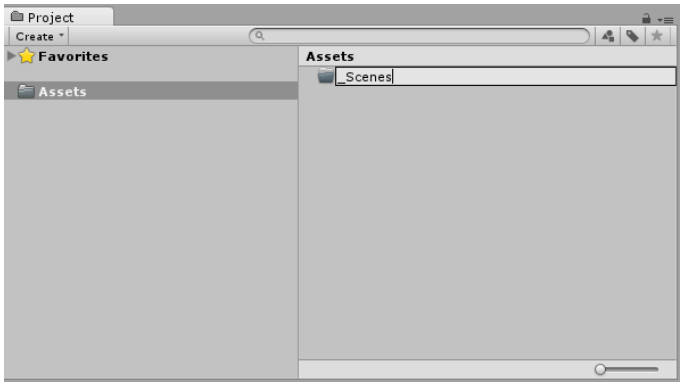
Creating a Simple Scene:

The first thing we need is a new project and scene.

Create a new project and give it a name. (We suggest "Tanks!"). You should now see an empty project.

The first thing we will so is save the scene. In the Project Hierarchy window, right-click on the *Assets* folder and select **Create** -> **Folder**. This will create a new folder in the Assets folder. Call the new folder "*_Scenes*".

Tip: By calling your Scenes folder "_Scenes" it will always appear at the top of the hierarchy, making your Unity scenes easy to find.



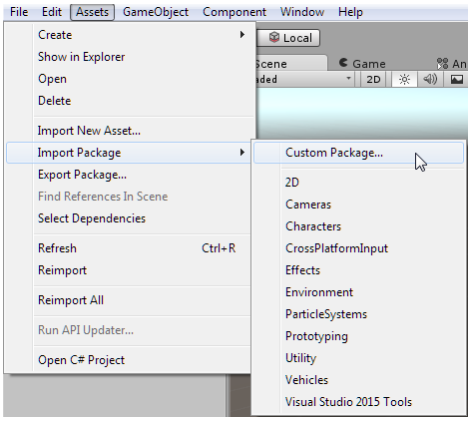
Now select **File** -> **Save Scene**. A dialog box will appear asking for the location and name for your new scene. Make sure to save it under the *_Scenes* folder, and call it something like *Game*.

Your scene should now appear in the hierarchy window.

Importing Assets:

Download the file "*Tank Game Models.unitypackage*" from the Portal page. This file contains some models we can use to make our tank game.

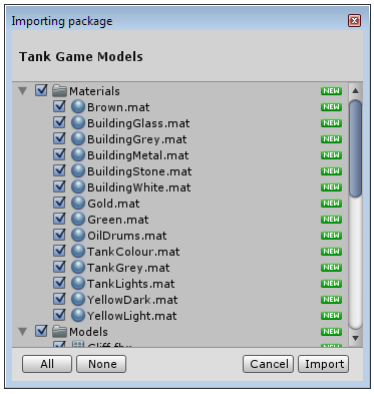
Select **Assets** -> **Import Package** -> **Custom Package**



Locate the file *Tank Game Models.unitypackage* and click **Open**.

After a few seconds you will see a pop-up dialog open asking you which assets you would like to import into your project.

Make sure all the assets are selected and press the **Import** button.

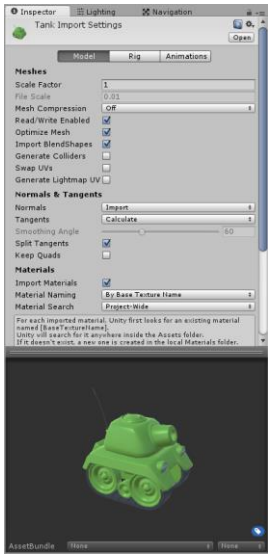


Once the import process has finished, you will see two new folders in the Hierarchy window. The first folder is *Materials*.

Materials are definitions of how a surface should be rendered, including references to textures used, tiling information, colour tints and more. Each file in this folder contains information about how a model will be drawn (for example, what colour to use when drawing a specific part of a model). Without the files in this folder, all the models in our game would be a dull grey colour.

The second folder is the *Models* folder. This folder contains the models we need to draw our game (like our tank, for example).

Take a moment to explore the different models in this folder. As you click on each model file you should see a preview of that asset appear in the Inspector window.



> Once you get a bit more experienced using Unity you may want to find your own models and put them into the scene. (The Unity Asset Store is a good place to start looking. From the **Window** menu, select **Asset Store**.

Create a Basic Scene:

For this tutorial we will only need two things – the tank model and a 3D plane. Once you've completed the tutorial you can go back add more models to the scene, but for now let's just focus on getting some basic tank movement.

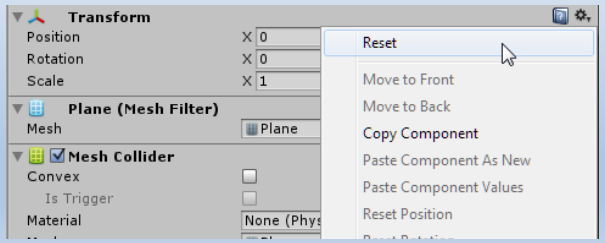
Select **Game Object** -> **3D Object** -> **Plane**. Reset the transform (set the x,y,z position to 0,0,0; do the same for the rotation, and ensure the scale is set to 1,1,1).

Set the scale of the plane to (10, 1, 10).

This plane will form the ground of our level.

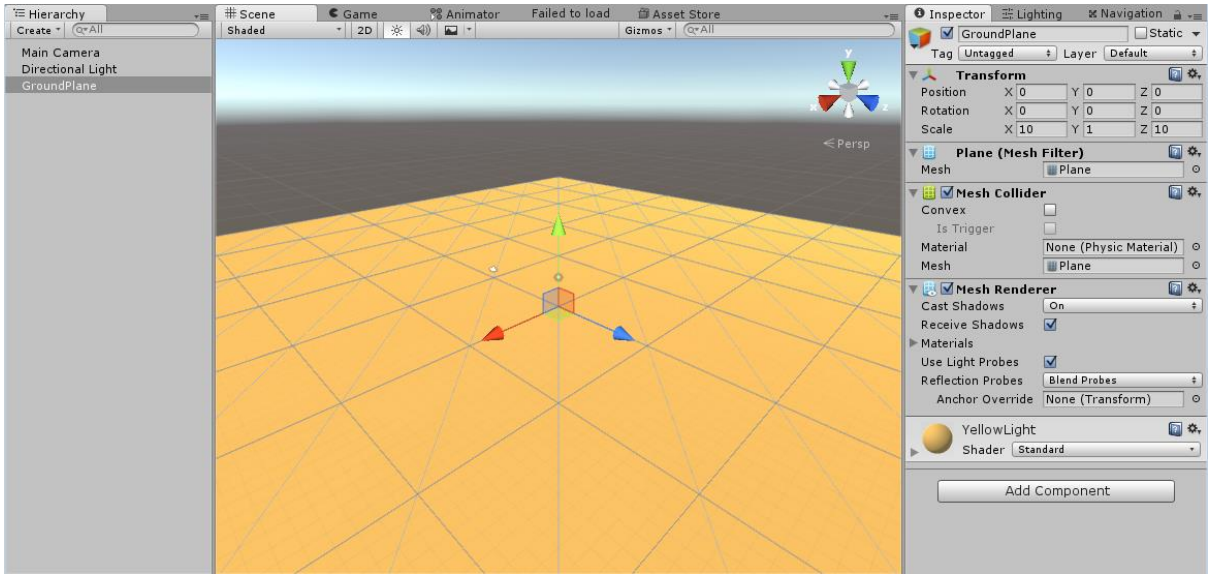
You may want to rename your plane and give it a more descriptive name. Rename it to *GroundPlane*.

A quick way to reset the transform (or any component) is to click the options icon on the upper-right of the component and select **Reset**.



Lastly, let's make the ground look like the ground by giving it a colour. In the *Project* window fine and open the *Materials* folder. Drag the *YellowLight* material onto the plane. (You can either drag it onto the plane in the scene, or drag it onto the plane's properties in the *Inspector* window. Both ways will work).

Your scene should look something like this:



This is all we need for now. Later, you can add more assets to make a complete level.

Add a Tank:

In the *Models* folder, find the *Tank* model and drag it into the scene. Reset its transform.

Before we begin scripting the tank, we need to add a few components to the model. These components will influence how the tank behaves in the game.

With the tank selected in the *Hierarchy* press the **Add Component** button in the *Inspector* window, and select **Physics** -> **Rigidbody**. Set the *Mass*, *Drag* and *Angular Drag* to 1.

The *Mass* value here is relative to other objects in the scene, and is not very important for our purposes. Setting a higher mass value will mean that our tank reacts less when it is hit by a missile (we'll get to that in a later lesson).

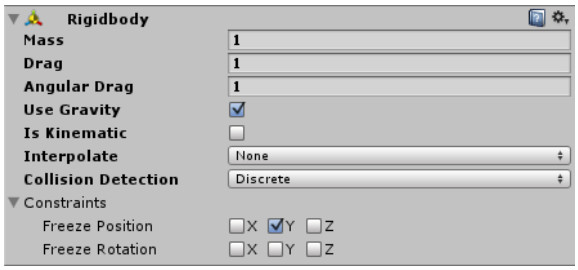
The *Drag* and *Angular Drag* values are important. These values will make our tank gradually stop moving after we stop applying a force to it (i.e., drag will slow the tank down after we 'take our foot off the accelerator'). So if you don't want your tank sliding all over the level we'll need to set these values to 1. (A higher value will make the tank stop faster, but it also means the tank will move slower.)

We also want to use gravity, so ensure *Use Gravity* is checked.

Finally we want to add a freeze constraint. We want to limit the tank's movement so that it never moves along the *Y* axis (i.e., the tank cannot jump or fly). Under *Constraints*, for the *Freeze Position* ensure *Y* is checked.

When we add collision objects to the level you may see why this is necessary. Basically, without this freeze constraint when the tank hits something in the level it can bounce a little. It's possible for the tank to bounce so much that it starts flying into the air. We want our tank to react a little more believably, so we make sure the tank can never fly into the air after a collision.

The properties for the *Rigidbody* should look like this:



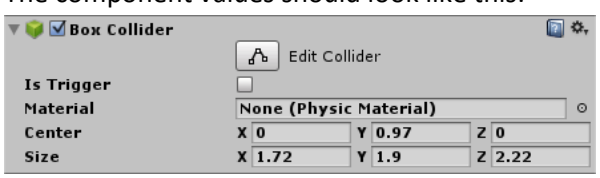
Next add a **Box Collider** component. Click **Add Component** -> **Physics** -> **Box Collider**.

This will add a collision box around our tank. When the box collider is added, it won't be big enough to cover the whole tank, so we need to enlarge it.

You can try to do this yourself by modifying the values in the *Box Collide* component, or you can enter the following values:

Center	x: 0	y: 0.97	z: 0
Size	x: 1.72	y: 1.9	z: 2.22

The component values should look like this:



We now have everything set up for the tank. Now it's time to do some scripting.

Scripting the Tank Movement:

In the *Project* window, under the *Assets* folder make a new folder called "*Scripts*". Under this folder, make a new folder called "*Tank*".

We'll put all the scripts that control the tank under this *Assets/Scripts/Tank* folder. (We will be writing a lot of scripts, so it's a good idea to keep everything ordered).

In the *Assets/Scripts/Tank* folder, make a new *C# Script* and call it "*TankMovement*"

When you create a new script, make sure there are no spaces in the name. Use "**TankMovement"** instead of "Tank Movement"

Double-click the *TankMovement* script. Either *MonoDevelop* or *Visual Studio* will open for you to start editing your script. (Your teacher will discuss with you the differences between these two programs, and how to set up your preferred editor).

Enter the following code into this script (we'll discuss its meaning next):

```
using UnityEngine;
using System.Collections;

public class TankMovement : MonoBehaviour
{

    public float m_Speed = 12f;         // How fast the tank moves forward and back
    public float m_TurnSpeed = 180f;    // How fast the tank turns in degrees per second

    private Rigidbody m_Rigidbody;
    private float m_MovementInputValue; // The current value of the movement input
    private float m_TurnInputValue;     // the current value of the turn input

    private void Awake()
    {
        m_Rigidbody = GetComponent<Rigidbody>();
    }

    private void OnEnable()
    {
        // when the tank is turned on, make sure it is not kinematic
        m_Rigidbody.isKinematic = false;

        // also reset the input values
        m_MovementInputValue = 0f;
        m_TurnInputValue = 0f;
    }
```

(continued on next page)

```
    private void OnDisable()
    {
        // when the tank is turned off, set it to kinematic so it stops moving
        m_Rigidbody.isKinematic = true;
    }

    // Update is called once per frame
    private void Update()
    {
        m_MovementInputValue = Input.GetAxis("Vertical");
        m_TurnInputValue = Input.GetAxis("Horizontal");
    }

    private void FixedUpdate()
    {
        Move();
        Turn();
    }

    private void Move()
    {
        // create a vector in the direction the tank is facing with a magnitude
        // based on the input, speed and time between frames
        Vector3 movement = transform.forward * m_MovementInputValue * m_Speed *
Time.deltaTime;

        // Apply this movement to the rigidbody's position
        m_Rigidbody.MovePosition(m_Rigidbody.position + movement);
    }

    private void Turn()
    {
        // determine the number of degrees to be turned based on the input,
         // speed and time between frames
        float turn = m_TurnInputValue * m_TurnSpeed * Time.deltaTime;

        // make this into a rotation in the y axis
        Quaternion turnRotation = Quaternion.Euler(0f, turn, 0f);

        // apply this rotation to the rigidbody's rotation
        m_Rigidbody.MoveRotation(m_Rigidbody.rotation * turnRotation);
    }
}
```

Let's discuss the code you just wrote.

The variables *m_Speed* and *m_TurnSpeed* are public variables, meaning you will be able to modify their initial value in the Editor. These variables control how fast your tank will move and turn. Think of these values as '12 units' forward per second, and 180 degrees of rotation per second.

The private variable *m_Rigidbody* will allow us to modify the movement of the tank via the rigid body component that we placed on the tank earlier. (The rigid body component will control anything to do with physics – so that means all the movement of the tank and any collisions are handled by this component).

The next private variables *m_MovementInputValue* and *m_TurnInputValue* allow us to store the movement and turn information we get from keyboard presses and use it later in our script to move the tank.

Awake:
The awake function runs when the object this script is attached to is added to the scene. Here we can initialize any variables in our script. Here we get the rigid body component attached to our tanke and set the m_Rigidbody variable to point to this component.

OnEnable:
This function is executed when we 'turn on' the tank. Later on, when we add the 'Game Over' part of the game, we'll turn all the tanks in our scene off (so that they will freeze in their current positions).

When we start a new game we'll need to turn the tanks on again. This function handles what happens when we do that.

Here, we make sure that the 'isKinematic' option is turned off. (Kinematic objects can't move – so by turning this off we allow our tank to move again). Finally we reset the input values.

OnDisable:
This is the opposite of the *OnEnable* function described above. When the game ends we 'freeze' all tanks by making them Kinematic. (This effectively turns their physics processing off, turning the moving tanks into static objects).

Update:
This function runs once every frame. It is the function responsible for updating the movement of our tank in the game.

Here we get the Axis input values. Imagine you are playing with a gamepad and one of the joysticks is controlling the movement of the tank. When you move the joystick on the horizontal axis the tank turns, and when you move it on the vertical axis the tank moves forward.

We won't be using a gamepad to control our tank. Instead Unity maps keyboard keys to these axes. The up and down arrow keys (or 'W' and 'S') are mapped to the vertical axis, and the left and right keys (or 'A' and 'D') to the horizontal.

This function checks the value of this input every frame and stores the value in the member variables for us to use later.

FixedUpdate:
The *FixedUpdate* function is similar to the *Update* function, but with one important difference. The *Update* function runs once every frame (how often the *Update* function runs per second can vary depending on the speed of your computer, CPU load, etc), but the *FixedUpdate* function runs at a fixed interval (a set number of times per second).

The *FixedUpdate* function is used specifically to update any physics that happens in the game. Processing physics actually takes a bit of time, so it is done less often than anything in the *Update* function.

In this function we take the input values we read in the *Update* function and modify the movement of our tank via the *Rigidbody* component.

You should never do any physics processing in the *Update* function, or else you can get unpredictable results. All physics processing needs to happen in the *FixedUpdate* function, but you should only do physics processing in this function – which is why we've put the keyboard event handling in the *Update* function and the tank movement updating in the *FixedUpdate* function.

To find out more about Unity's physics handling and the *FixedUpdate* function, you can refer to the Unity documentation: https://unity3d.com/learn/tutorials/modules/beginner/scripting/update-and-fixedupdate

Move:
Called by the *FixedUpdate* function to move the tank according to the keyboard presses.

Turn:
Called by the *FixedUpdate* function to turn the tank according to the keyboard presses.

Testing the Tank Movement:

Return to the Unity editor. Select the Tank in the *Hierarchy* and in the *Inspector* window press **Add Component** -> **Scripts** -> **Tank Movement**.

The script is now attached to the tank object, meaning that the script will execute while the tank is in the level.

Press the *Play* button now to test the movement of your tank.