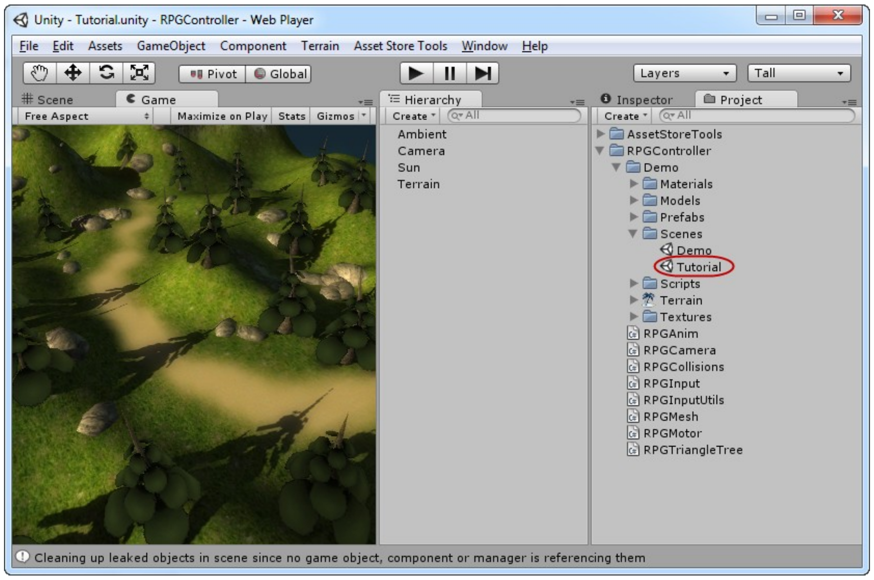
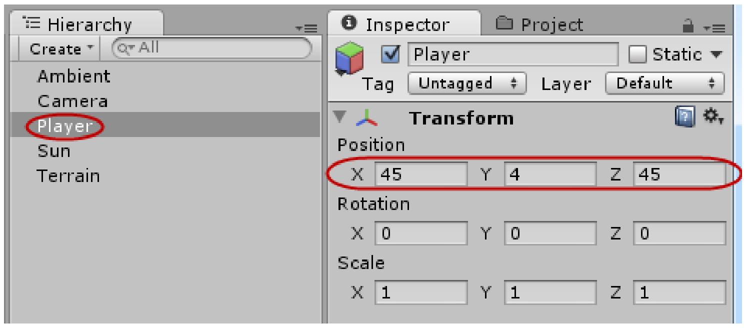
RPG Controller & Camera Manual

To follow along in this manual open the scene located in RPGController/Demo/Scenes/Tutorial, it should be a very basic scene looking like this:

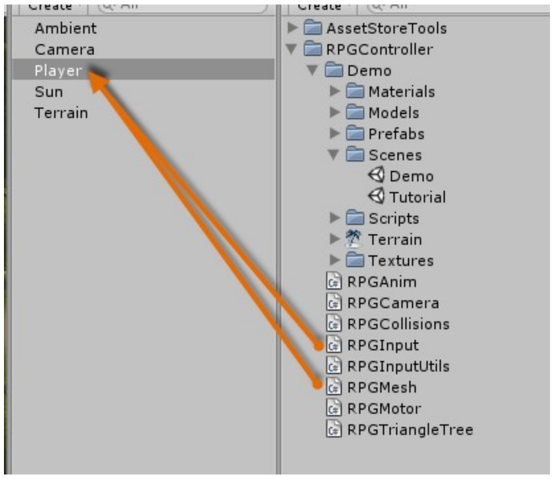


Cleaning up leaked objects in scene since no game object, component or manager is referencing them

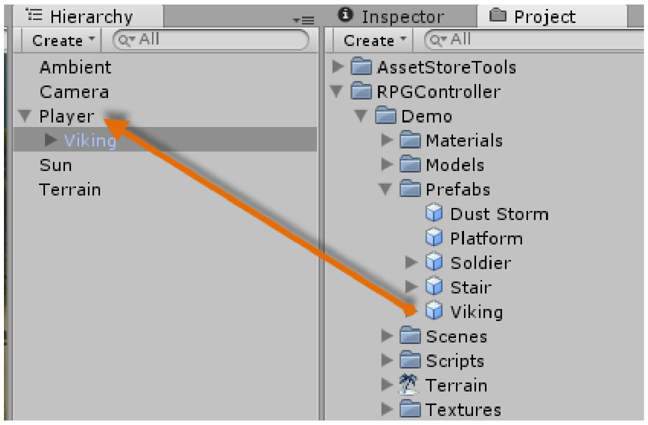
This scene contains just the basics terrain, a camera and two lights. Now it's time to do the basic setup of our controller and camera. Start by creating an empty game object , position it at 45, 4, 45 and call it player.



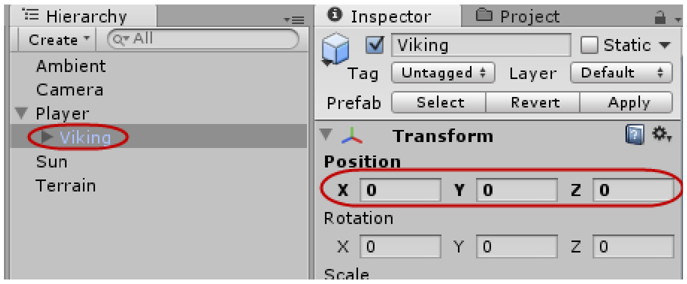
Now take the RPGMotor and RPGInput scripts and drag-n-drop them onto our Player object



Next, drag-n-drop the “Viking” prefab from the RPGController/Demo/Prefabs folder as a child on the new Player object we just created:

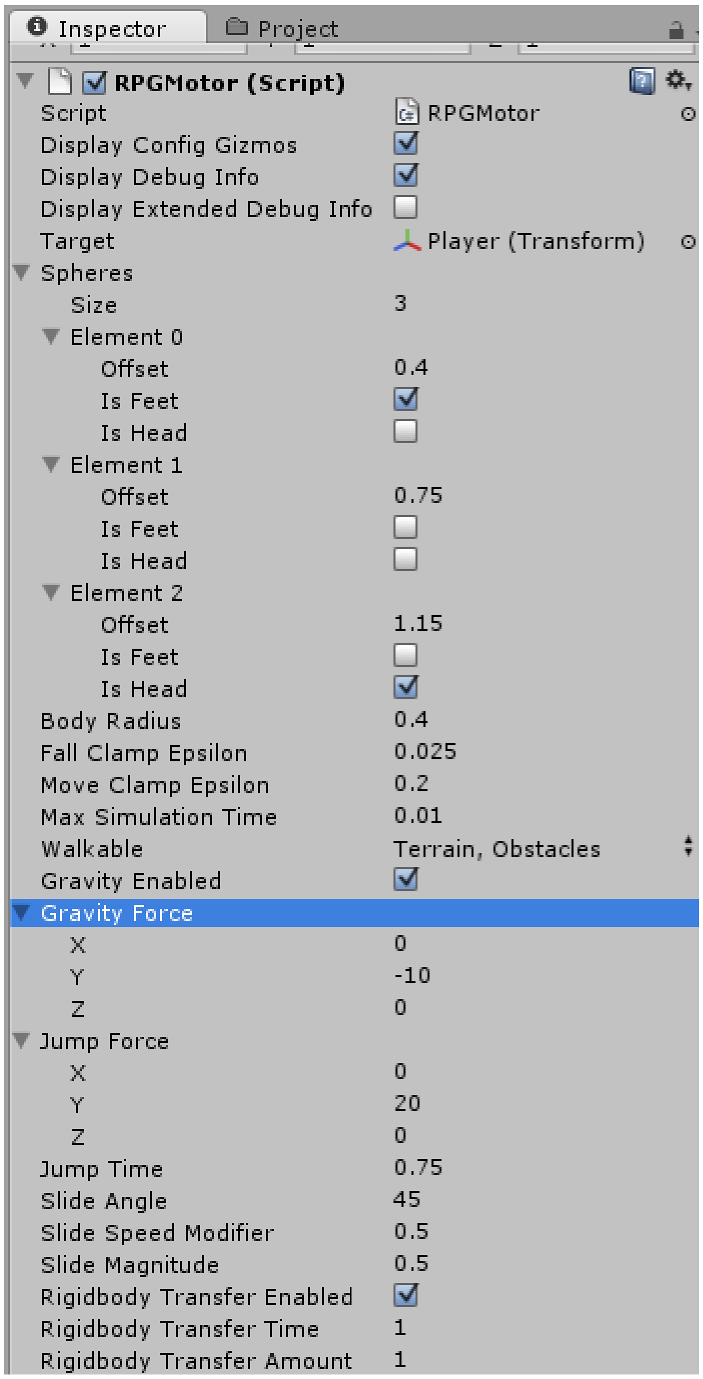


Also make sure that the position of the new Viking object is 0, 0, 0, like this:



Viking
Tag Untagged
Layer Default
Position
X 0 Y 0 Z 0
Rotation
X 0 Y 0 Z 0

Now it's time to configure the scripts we attached to the player object, let's start with the RPGMotor script. First of all I will show a screenshot of all the correct settings for this demo, and then I will walk through them one by one:



RPGMotor (Script)	✓
Script	RPGMotor
Display Config Gizmos	✓
Display Debug Info	✓
Display Extended Debug Info	☐
Target	Player (Transform)
Spheres	
Size	3
Element 0	
Offset	0.4
Is Feet	✓
Is Head	☐
Element 1	
Offset	0.75
Is Feet	☐
Is Head	☐
Element 2	
Offset	1.15
Is Feet	☐
Is Head	✓
Body Radius	0.4
Fall Clamp Epsilon	0.025
Move Clamp Epsilon	0.2
Max Simulation Time	0.01
Walkable	Terrain, Obstacles
Gravity Enabled	✓
Gravity Force	
X	0
Y	-10
Z	0
Jump Force	
X	0
Y	20
Z	0
Jump Time	0.75
Slide Angle	45
Slide Speed Modifier	0.5
Slide Magnitude	0.5
Rigidbody Transfer Enabled	✓
Rigidbody Transfer Time	1
Rigidbody Transfer Amount	1

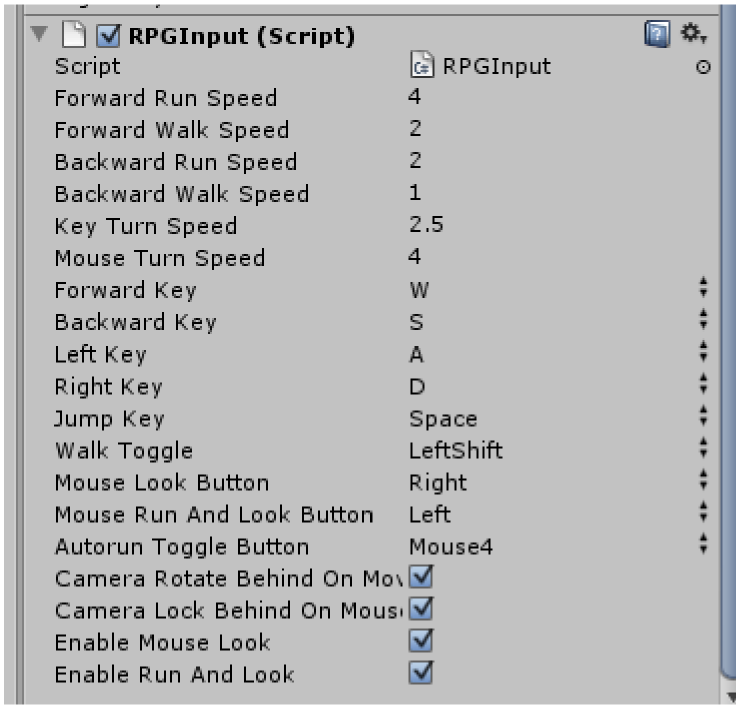
These are the correct settings for this demo, now let's inspect them one by one:

- **Display Config Gizmos –** This will show/hide the circles around the character used for configuring the spheres, leave this on.
- **Display Debug Info –** This will display a bunch of other visual feedback lines and logging in the console for debugging invalid behavior, leave this on.
- **Display Extended Debug Info –** This will display a lot of additional debug logging in the console, this produces a lot of console spam so leave it off.
- **Target –** The target transform we're going to control with our motor, set this to the Player objects transform.
- **Spheres –** These are the three spheres used for colliding with the environment, you can use any amount of spheres you want, but with a few restrictions: Just must have one, and only one sphere that is

marked feet, which is the “lowest” sphere. You must also have one, and only one sphere that is marked “Head” and which is the top one.

- **Body Radius –** This controls the size of to spheres.
- **Fall Clamp Epsilon –** This controls what values are considered “equal” when looking for a ground when falling, if you put this at a to large value you might experience “snapping” of the player when he's falling and get's close to the ground. For all intents and purposes leave this at the default 0.025
- **Move Clamp Epsilon –** This does the same as Fall Clamp Epsilon but for movement, in essence this controls how high terrains we can “step” to without colliding with them.
- **Max Simulation Time –** Works in the same way as the fixed update rate of unity itself, but for the manual physics for the controller. For all intents and purposes leave this at 0.2, but if you create very small or large characters (in relation to unitys “1.0 units” = “1 meter” standard) you might need to decrease or increase this.
- **Walkable –** The layers our character should collide with, set this to Terrain and Obstacles.
- **Gravity Enabled –** Enabled gravity, leave this on.
- **Gravity Force –** The force that the gravity will pull you with, -10 gives a pretty nice fall speed that fits most RPG games
- **Jump Force –** The power at which we should jump with, as a general rule this should be “abs(GravityForce) * 2”.
- **Jump Time –** The amount of time the Jump Force will be applied over, longer time equals longer jumps.
- **Slide Angle –** The angle at which we should start sliding backwards, this should be somewhere between 45-75 depending on how steep hills you can walk up.
- **Slide Speed Modifier –** The sliding “force” uses the gravity to push you down, this modifier allows you to lower or strengthen how much of the gravity that is applied, usually leave this at 0.5.
- **Slide Magnitude –** This controls how “far” above ground on the angled plane you're standing on you must get before you start sliding. This basically allows your character to walk on small slopes and uneven things on the ground without sliding.
- **Rigidbody Transfer Enabled –** This enables or disables transfer for forces from rigidbodies you collide with.
- **Rigidbody Transfer Time –** How long the transferred force should be applied over
- **Rigidbody Transfer Amount –** How much of the force should be applied, 1.0 equals 100%, 0.5 equals 50%, 2.0 equals 200%, etc.

Now, let's check out the RPGInput script, here's a screenshot of the values for this demo:

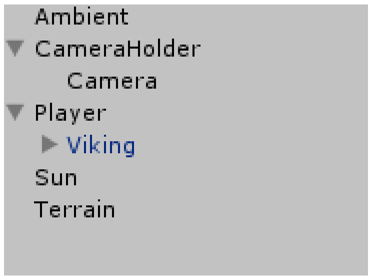

RPGInput (Script)

Script	RPGInput
Forward Run Speed	4
Forward Walk Speed	2
Backward Run Speed	2
Backward Walk Speed	1
Key Turn Speed	2.5
Mouse Turn Speed	4
Forward Key	W
Backward Key	S
Left Key	A
Right Key	D
Jump Key	Space
Walk Toggle	LeftShift
Mouse Look Button	Right
Mouse Run And Look Button	Left
Autorun Toggle Button	Mouse4
Camera Rotate Behind On Mov	✓
Camera Lock Behind On Mous	✓
Enable Mouse Look	✓
Enable Run And Look	✓

Now most of these settings are pretty obvious, and I will only walk you through the ones that are not.

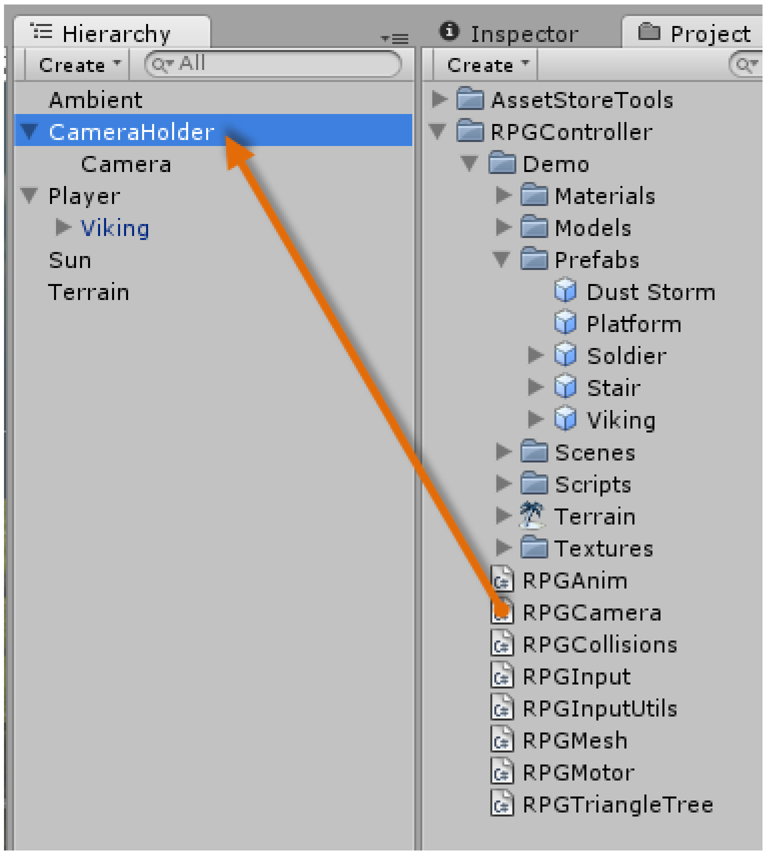
- **Mouse Look Button –** What mouse button to hold down to rotate the character with the mouse
- **Mouse Run And Look Button –** What mouse button to hold down, in combination with Mouse Look Button to both run forward and look around at the same time.
- **Camera Rotate Behind On Move –** If the camera should be forced to rotate behind us when we're moving
- **Camera Lock Behind On Mouse Look –** If the camera should be locked behind us when we're using "Mouse Look".

That was the RPGInput controller, now let's setup the camera and we're almost done with the basic setup! Create a new game object called "Camera Holder" and drag the Camera object as a child to it:



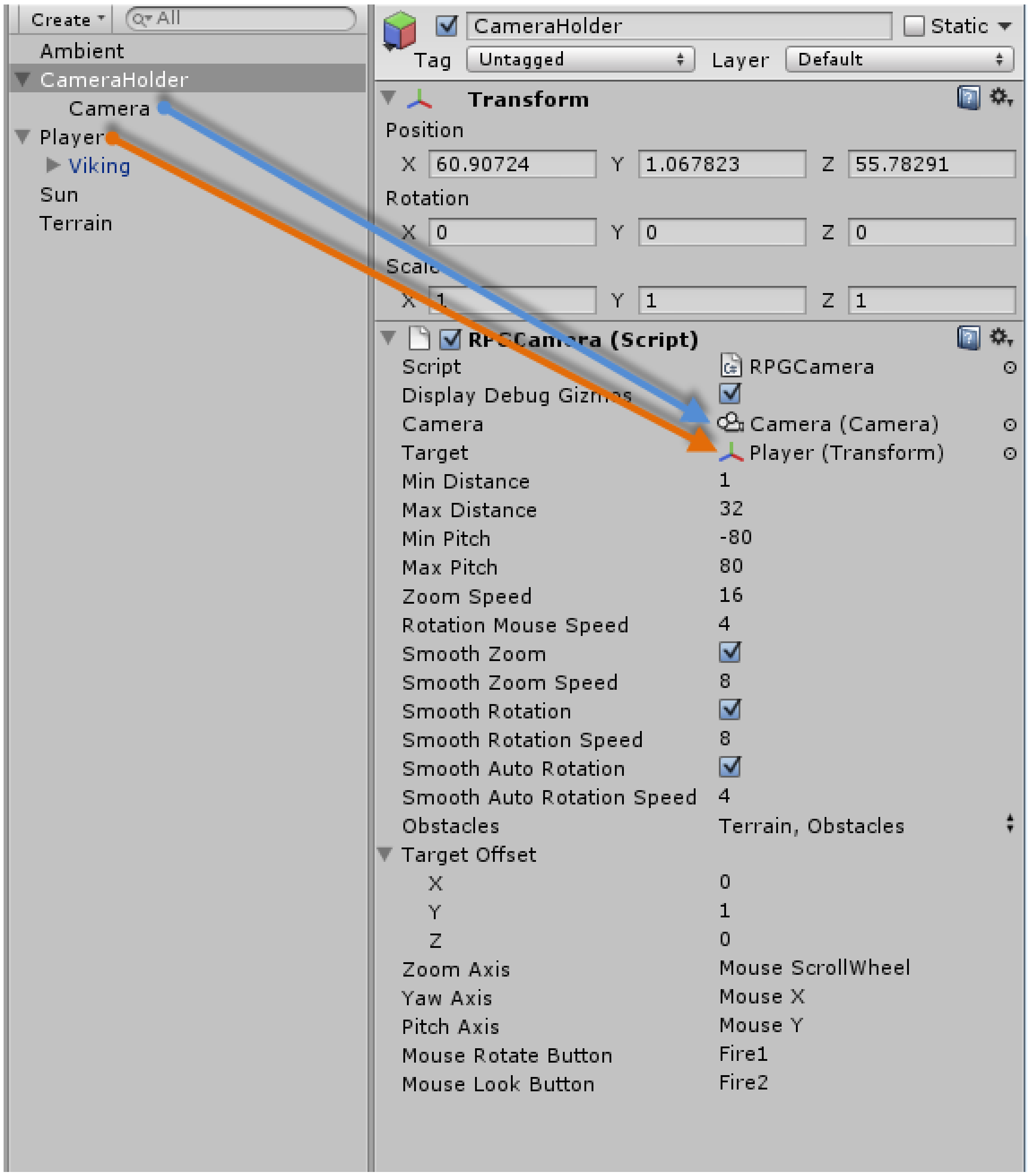
```
  Ambient
▼ CameraHolder
      Camera
▼ Player
    ▶ Viking
  Sun
  Terrain
```

Now drag-n-drop the "RPGCamera" script onto the CameraHolder object:



```
  Ambient                    ▶ AssetStoreTools
▼ CameraHolder               ▼ RPGController
      Camera                     ▼ Demo
▼ Player                             ▶ Materials
    ▶ Viking                         ▶ Models
  Sun                                ▼ Prefabs
  Terrain                                  Dust Storm
                                           Platform
                                         ▶ Soldier
                                         ▶ Stair
                                         ▶ Viking
                                     ▶ Scenes
                                     ▶ Scripts
                                     ▶ Terrain
                                     ▶ Textures
                                     RPGAnim
                                     RPGCamera
                                     RPGCollisions
                                     RPGInput
                                     RPGInputUtils
                                     RPGMesh
                                     RPGMotor
                                     RPGTriangleTree
```

Now drag-n-drop the "Camera" object into the "Camera" property on the RPGCamera script (blue arrow), and then drag-n-drop the "Player" object into the "Target" property on the RPGCamera script (orange arrow):



Ambient
CameraHolder
Camera
Player
Viking
Sun
Terrain

CameraHolder ✓ — Static
Tag: Untagged — Layer: Default

Transform

	X	Y	Z
Position	60.90724	1.067823	55.78291
Rotation	0	0	0
Scale	1	1	1

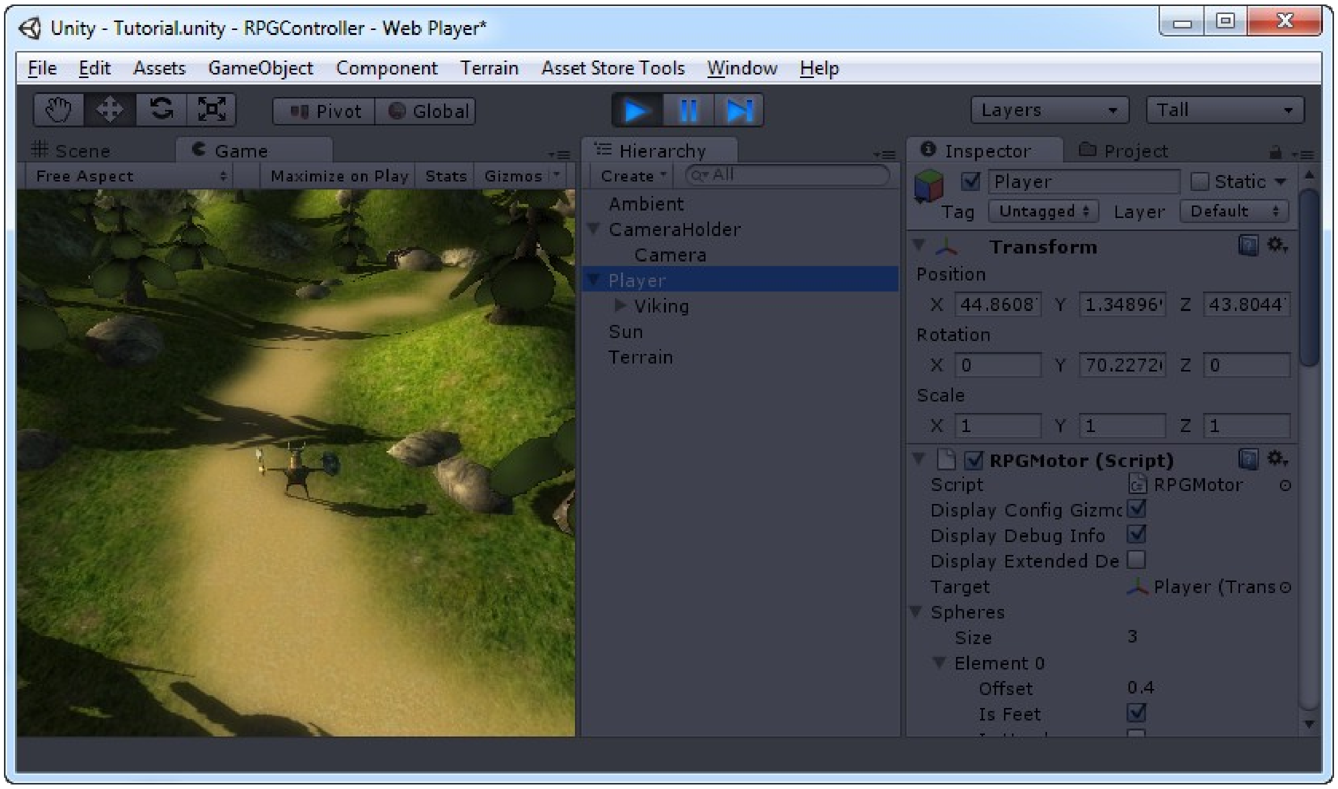
RPGCamera (Script) ✓

Script	RPGCamera
Display Debug Gizmos	✓
Camera	Camera (Camera)
Target	Player (Transform)
Min Distance	1
Max Distance	32
Min Pitch	-80
Max Pitch	80
Zoom Speed	16
Rotation Mouse Speed	4
Smooth Zoom	✓
Smooth Zoom Speed	8
Smooth Rotation	✓
Smooth Rotation Speed	8
Smooth Auto Rotation	✓
Smooth Auto Rotation Speed	4
Obstacles	Terrain, Obstacles
Target Offset	
X	0
Y	1
Z	0
Zoom Axis	Mouse ScrollWheel
Yaw Axis	Mouse X
Pitch Axis	Mouse Y
Mouse Rotate Button	Fire1
Mouse Look Button	Fire2

In the image above you will also see all the correct settings for the RPGCamera. I will walk through the ones that are not obvious:

- **Obstacles –** These are the layers our camera will collide with, set this to Terrain and Obstacles
- **Target Offset –** Offsets the cameras “look at point” in relation to the target.

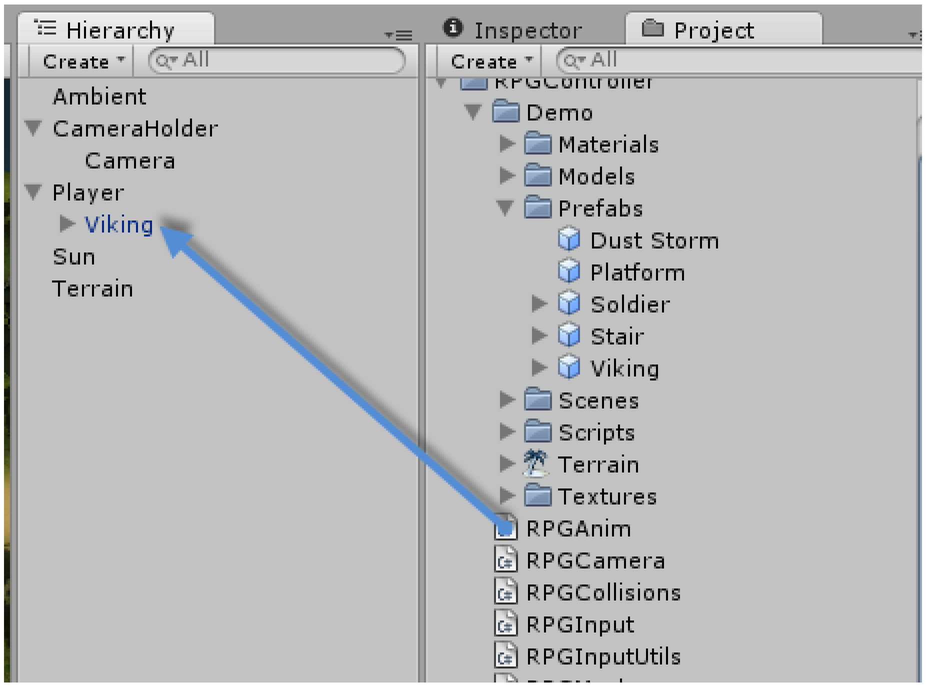
You should now be able to play and move the character around, but as you will notice he doesn't play any animations correctly:


Unity - Tutorial.unity - RPGController - Web Player*
File Edit Assets GameObject Component Terrain Asset Store Tools Window Help
Pivot Global
Layers Tall
Scene Game
Free Aspect Maximize on Play Stats Gizmos
Hierarchy
Create All
Inspector Project

Ambient
CameraHolder
Camera
Player
Viking
Sun
Terrain

Player Static
Tag Untagged Layer Default
Transform
Position
X 44.8608 Y 1.34896 Z 43.8044
Rotation
X 0 Y 70.2272 Z 0
Scale
X 1 Y 1 Z 1
RPGMotor (Script)
Script RPGMotor
Display Config Gizmo ✓
Display Debug Info ✓
Display Extended De ☐
Target Player (Trans
Spheres
Size 3
Element 0
Offset 0.4
Is Feet ✓

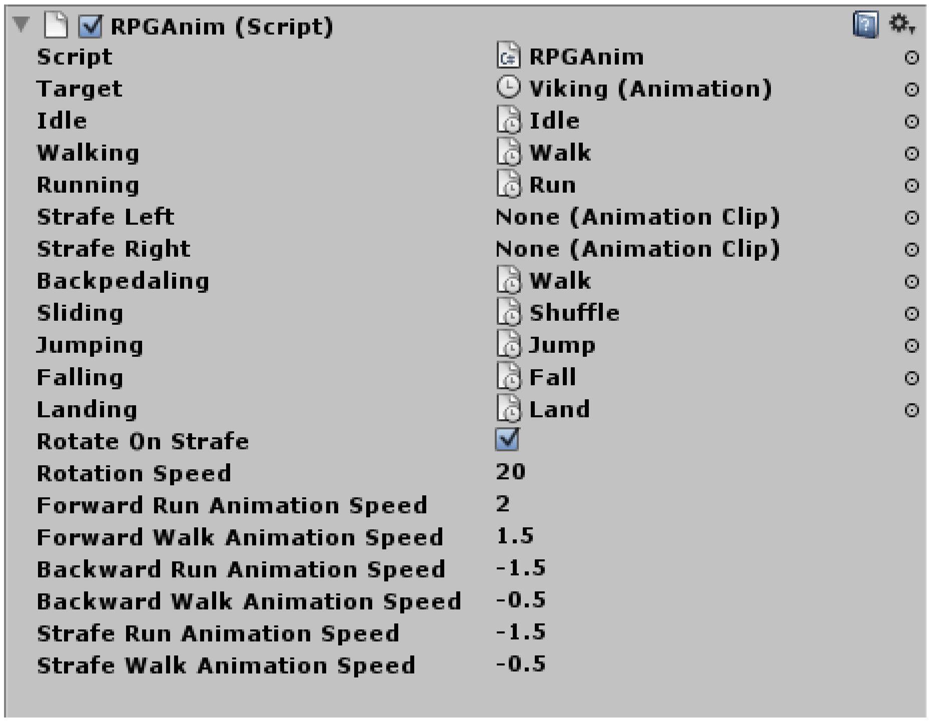
Time to fix that. Stop the game and drag the “RPGAnim” script to the “Viking” object that is a child to the “Player” object, like this:



Ambient
CameraHolder
Camera
Player
Viking
Sun
Terrain

RPGController
Demo
Materials
Models
Prefabs
Dust Storm
Platform
Soldier
Stair
Viking
Scenes
Scripts
Terrain
Textures
RPGAnim
RPGCamera
RPGCollisions
RPGInput
RPGInputUtils

Now click on the Viking object so we can configure the RPGAnim script:

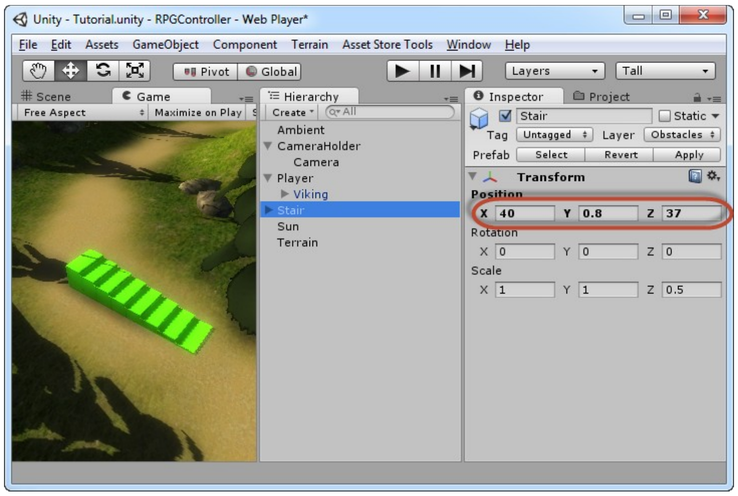


RPGAnim (Script)

Script	RPGAnim
Target	Viking (Animation)
Idle	Idle
Walking	Walk
Running	Run
Strafe Left	None (Animation Clip)
Strafe Right	None (Animation Clip)
Backpedaling	Walk
Sliding	Shuffle
Jumping	Jump
Falling	Fall
Landing	Land
Rotate On Strafe	✓
Rotation Speed	20
Forward Run Animation Speed	2
Forward Walk Animation Speed	1.5
Backward Run Animation Speed	-1.5
Backward Walk Animation Speed	-0.5
Strafe Run Animation Speed	-1.5
Strafe Walk Animation Speed	-0.5

- **Target –** Set this to the Viking object itself.
- **Idle, Walking, Running, Strafe Left, Strafe Right, Backpedaling, Sliding, Jumping, Falling, Landing –** These are the animations we should use, fill them in like in the image above. Notice that we're not using strafing. If you want an example on how to setup strafing check out the “Player_Solider” object in the RPGController/Demo/Scenes/Demo scene
- **Rotate On Strafe –** As we're not using strafing, we want our character to rotate and play our “Walking/Running” animations when we're strafing.
- **Rotation Speed –** How fast the character should rotate, leave at 20.
- **The next 6 settings controls the speed of the animations –** This is the speed our animations will play at, leave the the defaults.

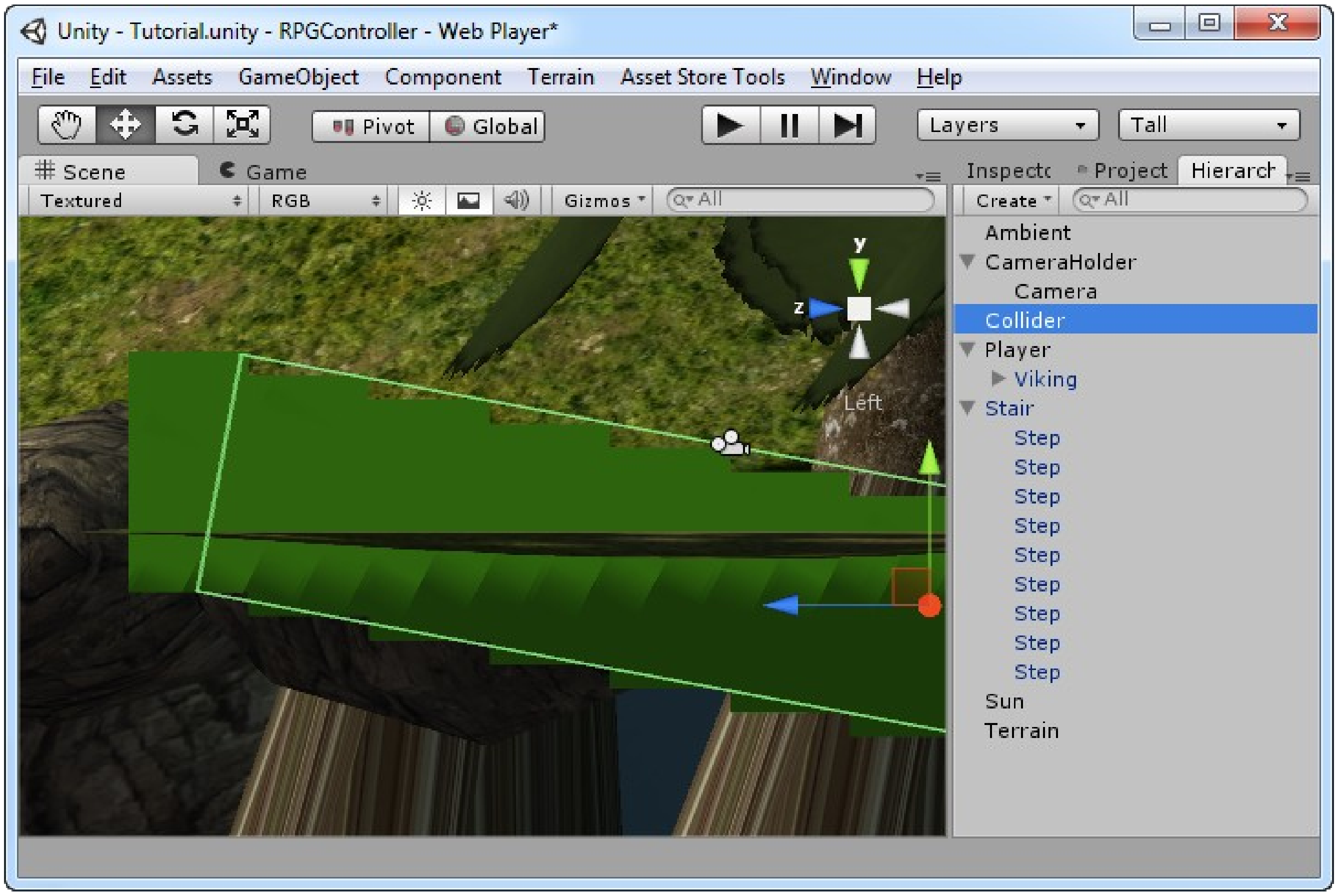
If you hit play now, everything will work as you would want it to! There's two more things to show, the first one is how to setup stairs and have the controller work with them. Drag the “Stair” prefab from RPGController/Demo/Prefabs into the scene and position it at 40, 0.8, 37, like this:


Unity - Tutorial.unity - RPGController - Web Player*
File Edit Assets GameObject Component Terrain Asset Store Tools Window Help
Pivot Global
Layers Tall
Scene Game
Free Aspect Maximize on Play
Hierarchy
Create All
Ambient
CameraHolder
Camera
Player
Viking
Stair
Sun
Terrain
Inspector Project
Prefab Select Revert Apply
Transform

Stair Static
Tag Untagged Layer Obstacles
Position X 40 Y 0.8 Z 37
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 0.5

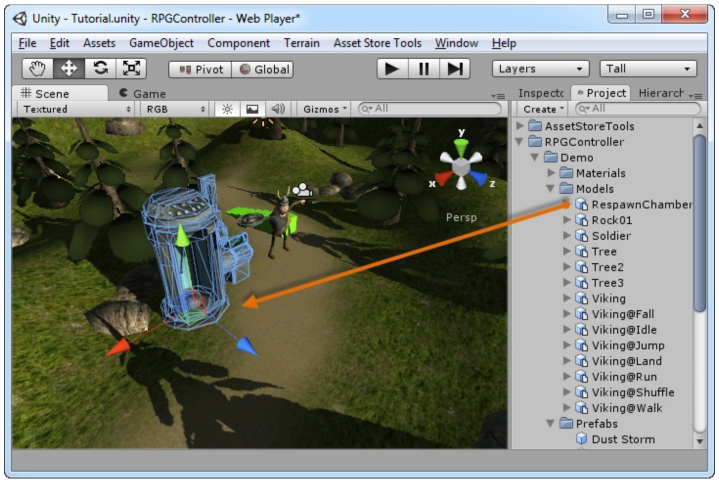
Now if you just walk up and down on this stair, it will work because of the “Move Clamp Epsilon” setting on the RPGMotor, but as you can see it will “snap” you up and down, and it doesn't look that good, what we really want to do is have a box collider which is angled with the stair steps.

What you should do is take a box collider, fit it to the width, height and length of the stairs and then angle it so it lies right on top of the stair steps, like this:

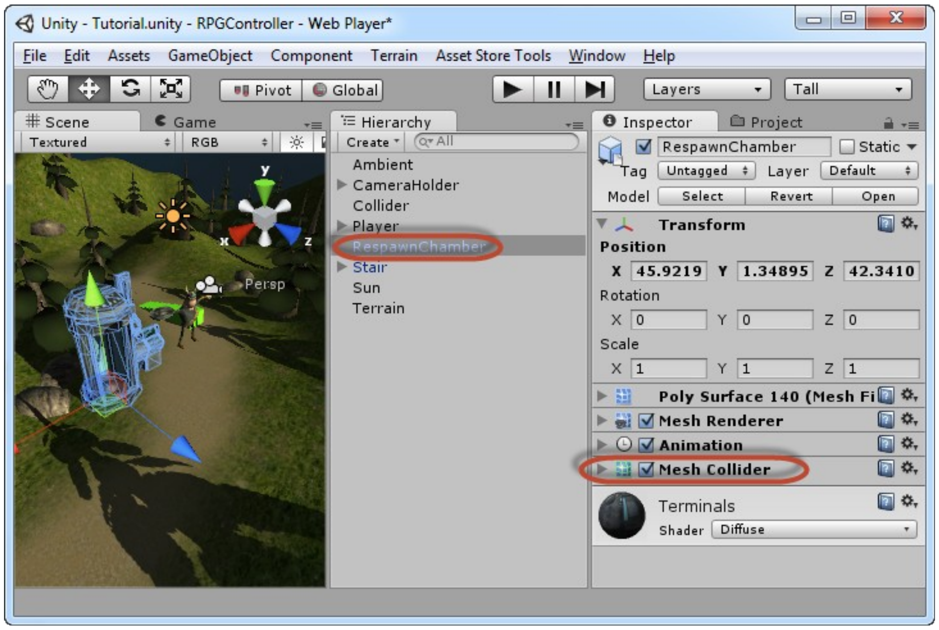


Also make sure that the collider object is in the “Obstacles” layer. If you play now and walk on the stairs you will see now that you get the nice smooth walk up/down the stairs that most RPG/MMOs have.

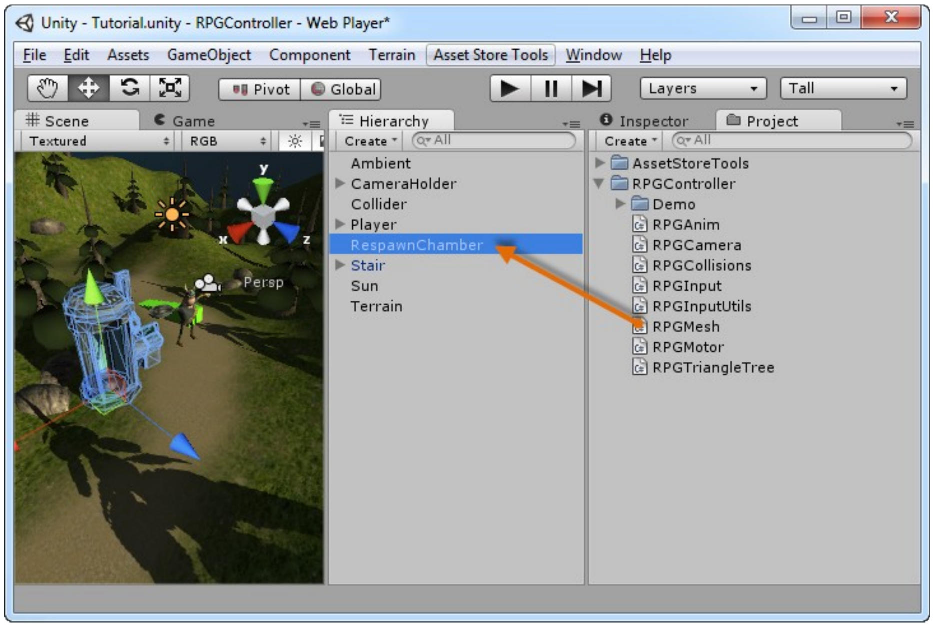
The last thing I want to show you how to setup is mesh collisions, first we need mesh to try this out on, drag-n-drop the RespawnChamber model from the RPGController/Demo/Models folder into the scene:



The first thing we're going to do is to attach a “Mesh Collider” to the chamber in our scene, select the RespawnChamber object in the scene and then go to Component → Physics → Mesh Collider:



Next, drag-n-drop a “RPGMesh” script on top of our RespawnChamber object:



The RPGMesh object doesn't have any public settings at all. Last of all, make sure that our RespawnChamber object is in the Obstacle layer. You should now be able to collide with chamber, walk inside of it, jump on-top of it, etc.

I hope you enjoyed this manual, any questions, bug reports or feature requests are directed to the thread in the Assets and Asset Store forum on http://forums.unity3d.com