Amazon Lumberyard Editor

Getting Started Guide

Version 1.8

Amazon Lumberyard Editor

Table of Contents

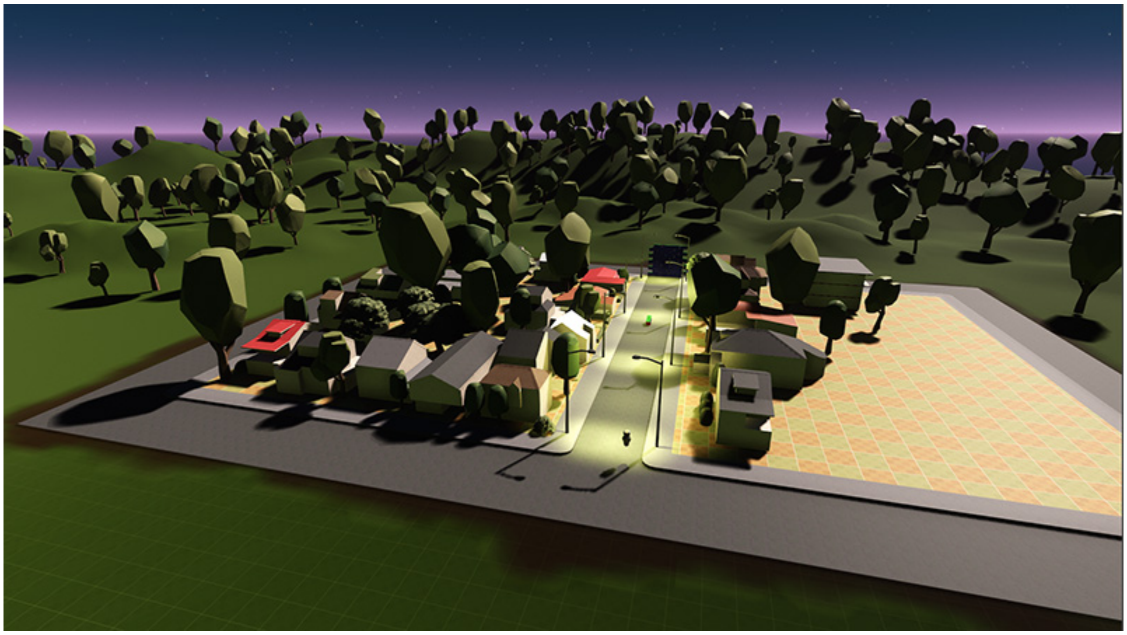
Introduction to Lumberyard Editor

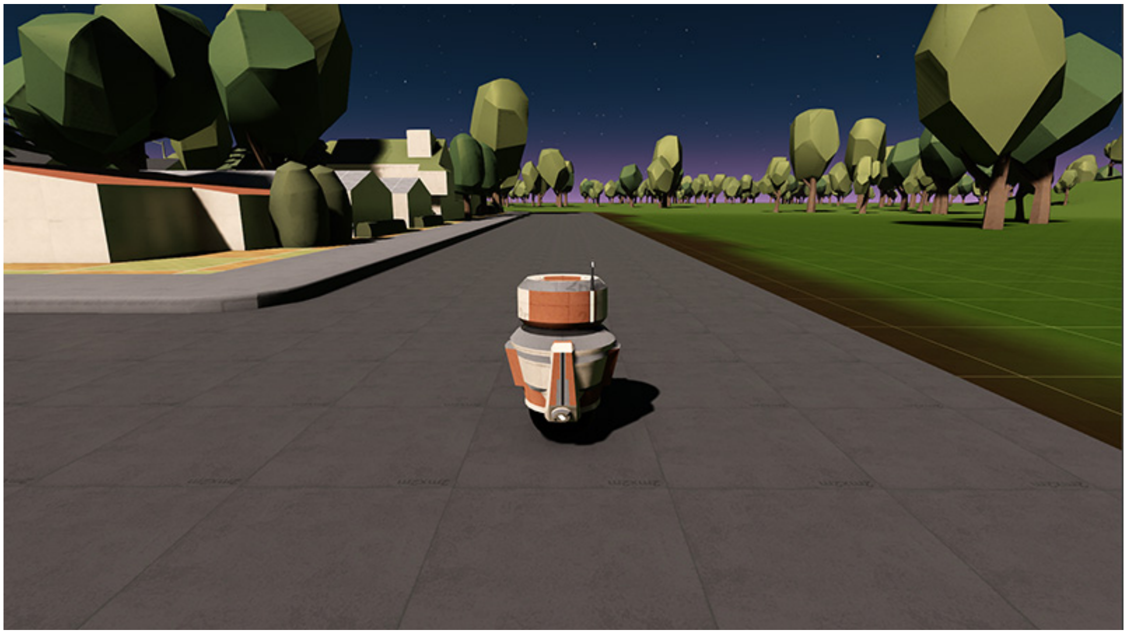
Welcome to the *Getting Started Guide for Amazon Lumberyard Editor*. Lumberyard is a free, cross-platform, 3D game engine for creating high quality games, connecting your games to the vast AWS cloud computing and storage, and engaging fans on Twitch.

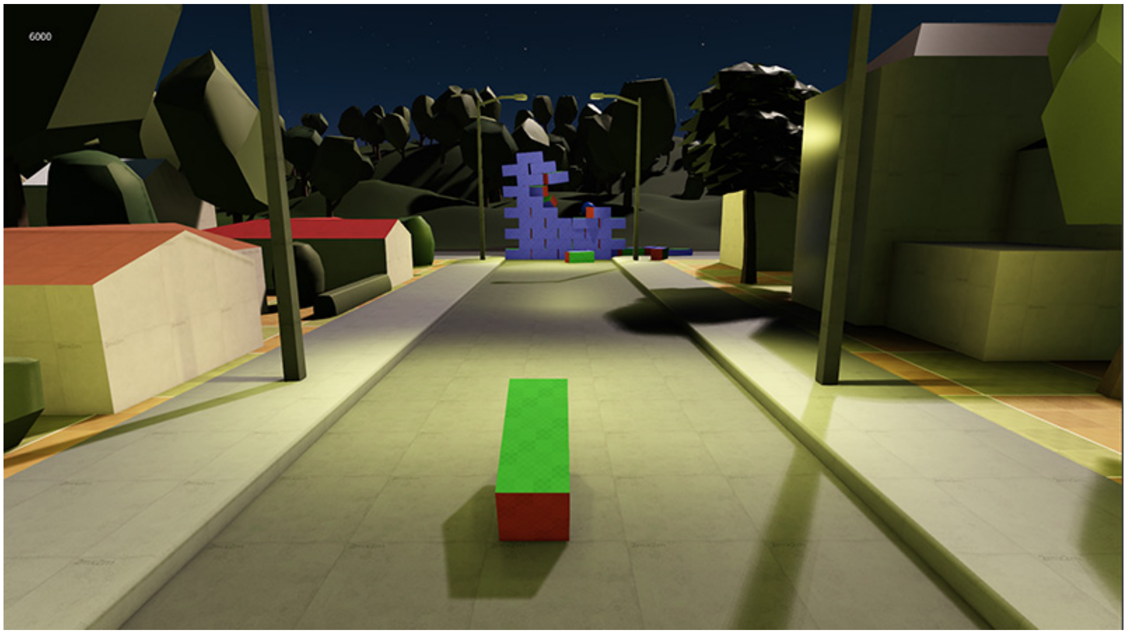
This Getting Started Guide familiarizes you with the basics of Lumberyard Editor. You'll be guided through nine tutorials that describe the most commonly used tools and features of this editor. During this tutorial, you'll create an environment with buildings, trees, and rolling hills. You'll also create a script for a character that can shoot balls to knock over a block wall.

After completing this tutorial, you'll be knowledgeable enough to explore Lumberyard's wide range of tools and features. You can complete additional tutorials to help you learn more about specific tools and features and put you well on your way to building your next game.

To access Lumberyard's library of written and video tutorials, see Amazon GameDev Tutorials






6000

Running Lumberyard Setup Assistant

Use the Lumberyard Setup Assistant application to validate that you have installed the third-party software required to run Lumberyard. For more information, see Using Lumberyard Setup Assistant to Install Third-Party Software in the *Lumberyard User Guide*.

To use Lumberyard Setup Assistant

1. Do one of the following:
   - Double-click the **Setup Assistant** desktop shortcut.
   - Click **Setup Assistant** in the Start menu.
   - Open the directory where you extracted Lumberyard and run `SetupAssistant.bat`.
2. On the **Get started** page, select what you want to do. If you choose a compile option, you may later see the **Install SDKs** page. If so, follow the instructions to obtain each third-party SDK that you do not yet have installed.
3. Select **Visual Studio 2013**, **Visual Studio 2015**, or both.

   Note
   Lumberyard supports Microsoft Visual Studio 2015 Update 3 or later. By default, the Visual Studio 2015 installation does not include C++ as an installed language. In order to build, you must select **C++**, its child options, and **MFC** during the Visual Studio 2015 installation. To verify your current installation, click **Control Panel**, **Programs and Features**, **Microsoft Visual Studio 2015**. Next, select **Modify** to view or add C++ and MFC support.
4. Click **Next**. Follow the instructions on each page. When you have all the required software and SDKs installed for your implementation, click **Launch Editor**.
5. Log in to your existing Amazon account or create a new account to access the editor.

Using the Project Configurator

You can use the Project Configurator to configure a new project or enable gems (packages):

To configure a new project

1. In Lumberyard Setup Assistant, on the **Summary** page, click **Configure project**.
2. In the Project Configurator, click **Create new**.
3. Type your project name, and then click **Create project**.
4. Click on an image to select your project.
5. Click **Set as default**.

To enable gems

1. In Lumberyard Setup Assistant, on the **Summary** page, click **Configure project**.
2. In the Project Configurator, click **Enable Gems** below the project name to which you want to add gems.
3. Select the gems you want to enable, and then click **Save**.
4. From a command prompt, use `lmbr_waf configure` from the `\dev` directory to configure your default project.
5. If the previous step was successful, at the same command line, do one of the following depending on which version of Visual Studio you have installed:

- For Visual Studio 2013, type `lmbr_waf build_win_x64_vs2013_profile -p all`
- For Visual Studio 2015, type `lmbr_waf build_win_x64_vs2015_profile -p all`

To view other build commands or variables to use for this step, see Game Builds.

This step may take some time to complete. A success message at the end indicates a successful completion.

Running Lumberyard Editor

To run Lumberyard Editor, click **Launch Lumberyard** on the **Summary** page of the Lumberyard Setup Assistant application.

You can also access the editor using one of the following:

- Click **Lumberyard Editor** in your Start menu.
- Navigate to `\dev\`*`Bin64vc120`*` or `*`Bin64vc140`*`\editor.exe`.

If you have not yet run Lumberyard Setup Assistant, it will start first to ensure that you install all required third-party software (this only happens the first time you run the editor).

Directory Structure

The Lumberyard directory structure includes the following directories and files:

`dev`

- _WAF_ – Waf build system files
- Bin64 – Binaries directory and configuration files for the resource compiler
- Bin64vc120 – Binaries directory and configuration files for Visual Studio 2013
- Bin64vc140 – Binaries directory and configuration files for Visual Studio 2015
- Code – Source files directory and solution files
- Editor – Editor assets
- Engine – Engine assets
- FeatureTests – Collection of levels to test certain features
- Gems – Optional systems and assets
- MultiplayerProject – Multiplayer sample project for evaluating Amazon GameLift
- MultiplayerSample – Multiplayer sample project that demonstrates how to build a multiplayer game with the new component entity system.
- ProjectTemplates – Configuration files, libraries, and scripts for the empty template
- SamplesProject – Collection of sample levels and code
- Tools – Third-party tools and plugins

`3rdParty`

- Third-party software required to use or compile Lumberyard

`docs`

- Release notes
- *Lumberyard Getting Started Guide*

Accessing Documentation

The Lumberyard Documentation team is continuously writing and improving the official documentation to provide a better help experience:

- Lumberyard online documentation
- Lumberyard tutorials

You can also refer to the `docs` folder in the Lumberyard directory for help topics about using Lumberyard.

Contacting Support

Amazon Web Services provides a combination of tools and expertise to help support your success with Lumberyard. To learn about the variety of resources we offer, see Amazon Lumberyard.

It's day one and we're just getting started. We look forward to your feedback.

Working with Lumberyard Editor

The Lumberyard Editor provides extensive tools for creating and customizing your game environment including levels, objects, terrain, lighting, animation, layers, and much more. Keyboard controls familiar to gamers make navigating your levels in 3d a breeze. Customize your display so that you can focus on what's important to you. Read the topics in this section to learn the Lumberyard Editor's most common and essential features.

Topics

New Level Creation

A level is a world or map that represents the space or area available to the player during the course of completing a discrete game objective. Most games consist of multiple levels. To create a level, you use **Lumberyard Editor**.

To create a level in Lumberyard Editor

1. Start Lumberyard Editor as explained in the Introduction (p. 3).
2. In the **Welcome to Lumberyard Editor** window, you can create a new level, open a recent level (if one exists), or open a level from within the level directory. You can also choose to stop showing this dialog on startup.
3. Click **New Level**.
4. Type a name for your new level.
5. To generate a terrain for your level, you can specify your **Heightmap Resolution** and **Meters Per Texel**. A heightmap is a grayscale image that stores surface height data with high areas in white and low areas in black. For this tutorial, accept the defaults. Click **OK**.
6. In the **Generate Terrain Texture** dialog box, you can control the appearance of your level's terrain. Click **OK** to accept the default settings.

Lumberyard Objects

Lumberyard has three object types, which encompass every object that can be placed in a level:

Entities

Objects with behavior properties. The behavior properties use game scripts or code to enable objects to respond to game events. Entities are subdivided into the following types:

- **Entities** – General objects used to set up and create gameplay conditions or visual settings (such as lights, volumes, cameras, physics objects, and so on).
- **Geometry Entities** – Entities with an attached geometry mesh.
- **Particle Entities** – Particle systems created and placed within a level.
- **Archetype Entities** – Custom entities that the player defines based on existing entity properties.

Brushes

Objects with 3D mesh data only. Brushes do not contain behavior properties of an entity.

Designer objects

Objects that are created with the **Lumberyard Designer** modeling tool. Designer objects are similar to brushes.

Editor Layout

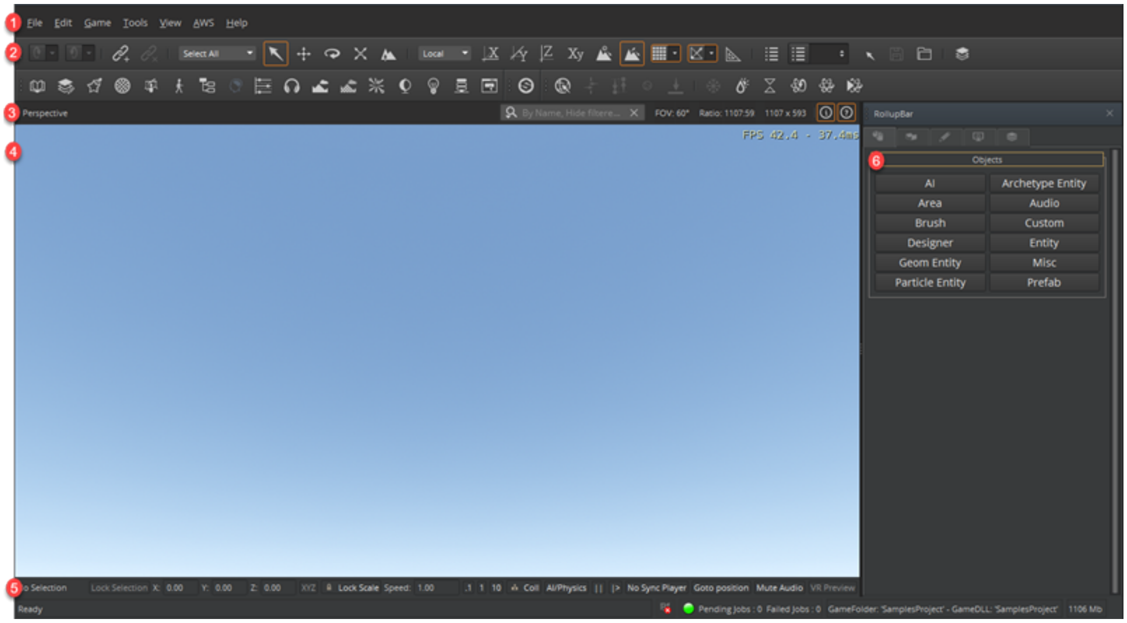
The Lumberyard Editor window comprises the following areas:

1. **Main menu** – All functions and settings
2. **Editor toolbar** – Most commonly used tools and editors
3. **Viewport title bar** – Search bar and display options for **Perspective** viewport
4. **Perspective viewport** – 3D environment view of level
5. **Viewport controls** – Controls for selected objects, options for navigation speeds, and other viewport features
6. **Rollup Bar** – Access to objects or entities and tools for building and managing content in the **Perspective** viewport

The Rollup Bar contains the following tabs:

- **Objects** – Brushes, entities, volumes, prefabs, etc.
- **Terrain** – Terrain, vegetation, and environment tools
- **Modeling** (obsolete)
- **Display** – Render settings, 3D settings, hide settings
- **Layers** – Organize and manage assets by layers

To get the most out of Lumberyard, familiarize yourself with these terms and areas.

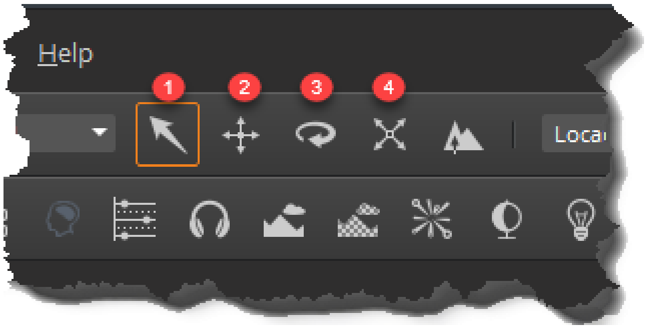


Essential Tools

Lumberyard Editor features many robust tools, settings, and options to help you build high quality games. The most essential tools for manipulating objects are **Select**, **Move**, **Rotate**, and **Scale** tools. Note that you can select objects with any of these tools.

You select these tools either with a keyboard shortcut or from the **Lumberyard Editor** toolbar, as shown in the following image.

1. **Select**
2. **Move**
3. **Rotate**
4. **Scale**



Each tool provides its own unique 3D handle, called a gizmo, on the selected object. This helps you identify the tool that is currently selected.

> **Tip**
> If you don't see the toolbar with these tools, right-click an empty area of the menu or toolbar area and click **EditMode Toolbar**.

To use the keyboard to select a tool

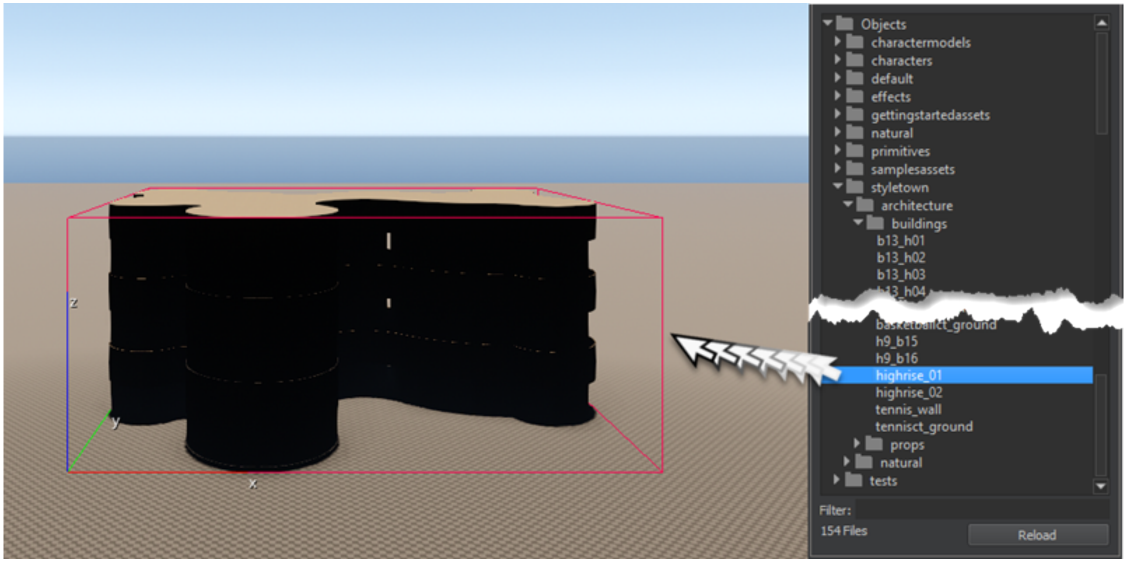
Press any of the following numbers on your keyboard:

- **Select** – **1**
- **Move** – **2**
- **Rotate** – **3**
- **Scale** – **4**

To place an object

Place an object in your level so that you may follow along and test out the essential tools described in the following sections.

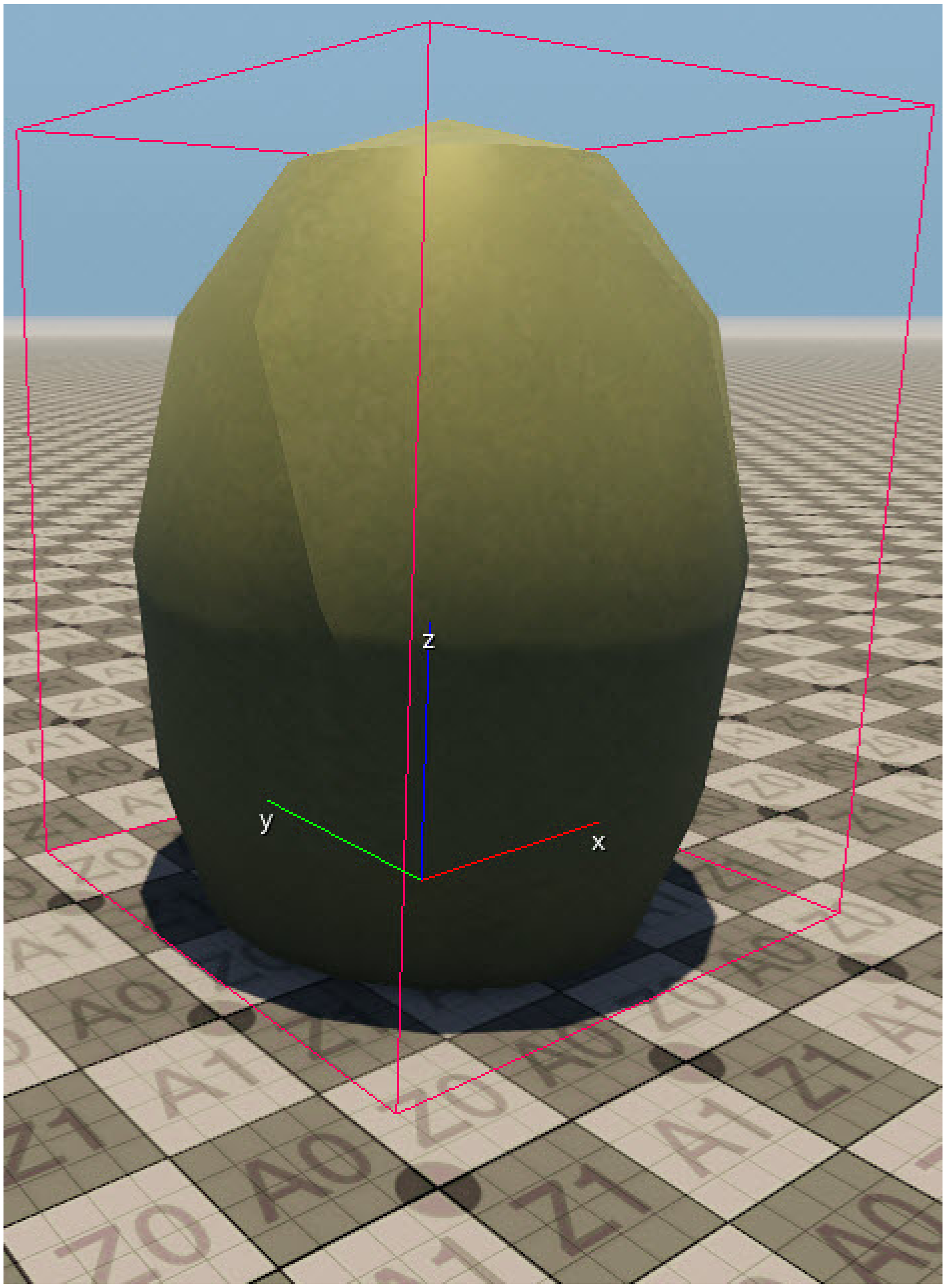
1. In the **Rollup Bar**, click **Brush** to display the assets you can add to the level.
2. Under the **Browser** heading, in the directory tree, expand `objects\styletown\architecture\buildings` and select one of the buildings listed (for example, `highrise_01`).
3. Drag the building you selected (for example, the text of `highrise_01`) into the **Perspective** viewport.
4. You are now ready to test the select, move, rotate, and scale tools.


z
y
x
Objects
charactermodels
characters
default
effects
gettingstartedassets
natural
primitives
samplesassets
styletown
architecture
buildings
b13_h01
b13_h02
b13_h03
b13_h04
basketballct_ground
h9_b15
h9_b16
highrise_01
highrise_02
tennis_wall
tennisct_ground
props
natural
tests
Filter:
154 Files
Reload

Select

Using **Select**, you can choose any object in the **Perspective** viewport. The gizmo for **Select** is a set of three lines—one for each direction: X, Y, Z.

To select, move your pointer over the object you want to select. When the object is highlighted yellow and the pointer changes to a +, left-click to select the object.

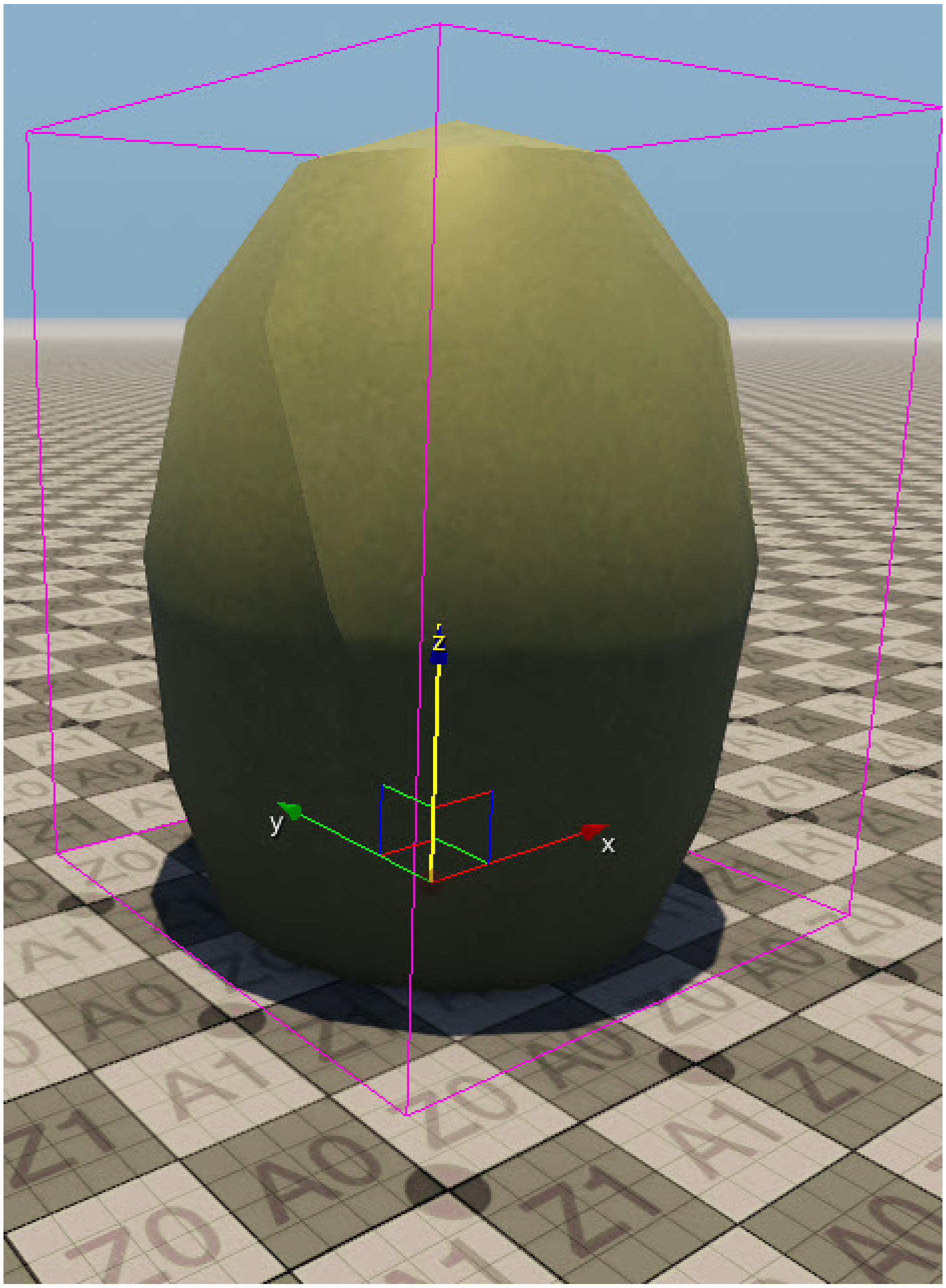


Move

The **Move** tool selects and moves an object within the 3D space of the **Perspective** viewport. The **Move** gizmo is a set of three lines with arrowheads on the X, Y, and Z lines.

To move your selected object along a fixed line, click the X, Y, or Z line, which appears yellow when selected. You can then drag your object along that line.

The **Move** gizmo also features three small right angle squares along the XY, ZY, and XZ planes. To move your object along a plane, click to select one of the small squares. You can then drag your object along that plane.

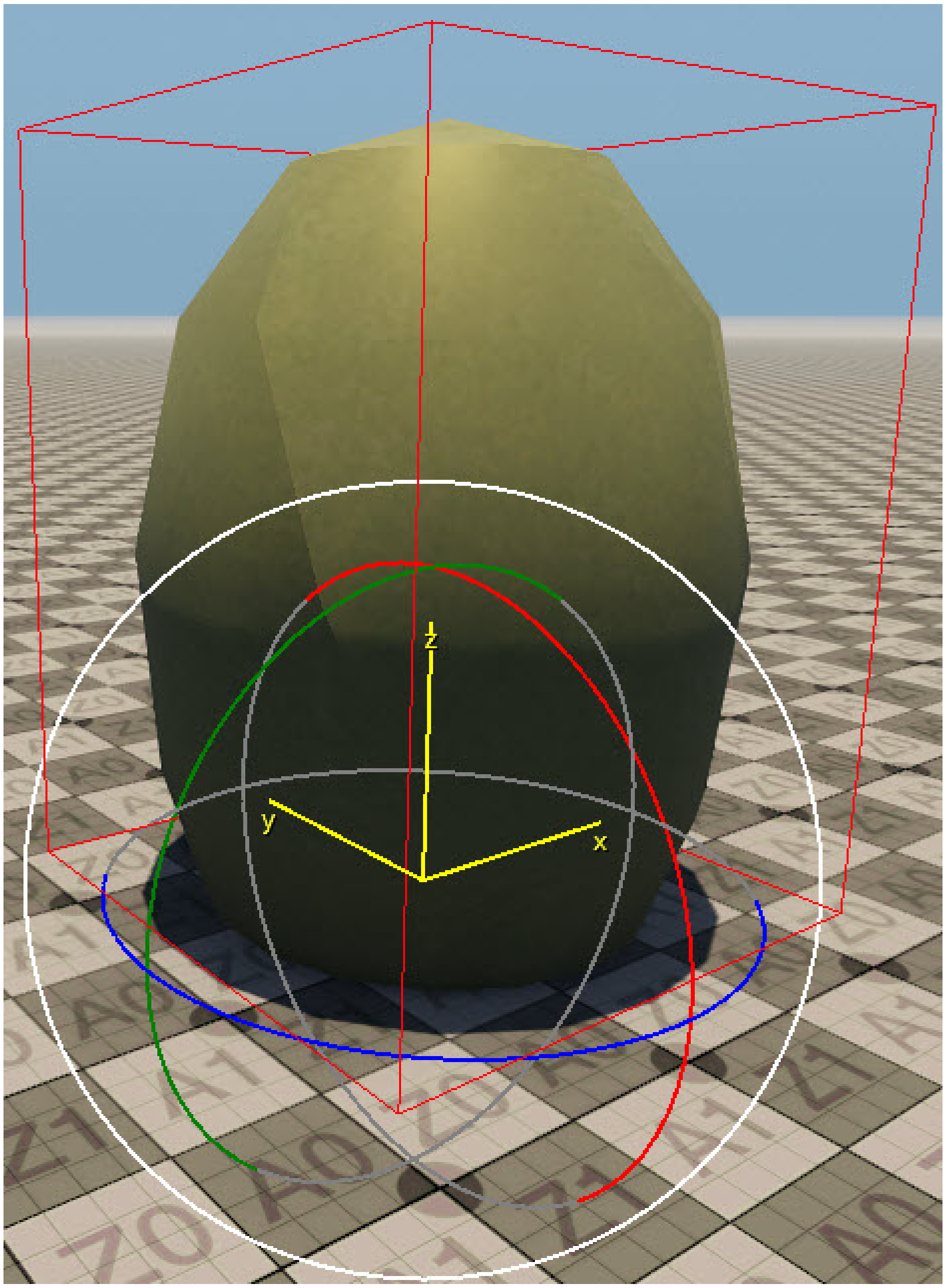


Rotate

The **Rotate** tool selects and rotates an object. The **Rotate** gizmo is a set of circles around the object along the X,Y, and Z axes.

To rotate an object, select one of the small inner circles. You can then drag to rotate around that rotational plane.

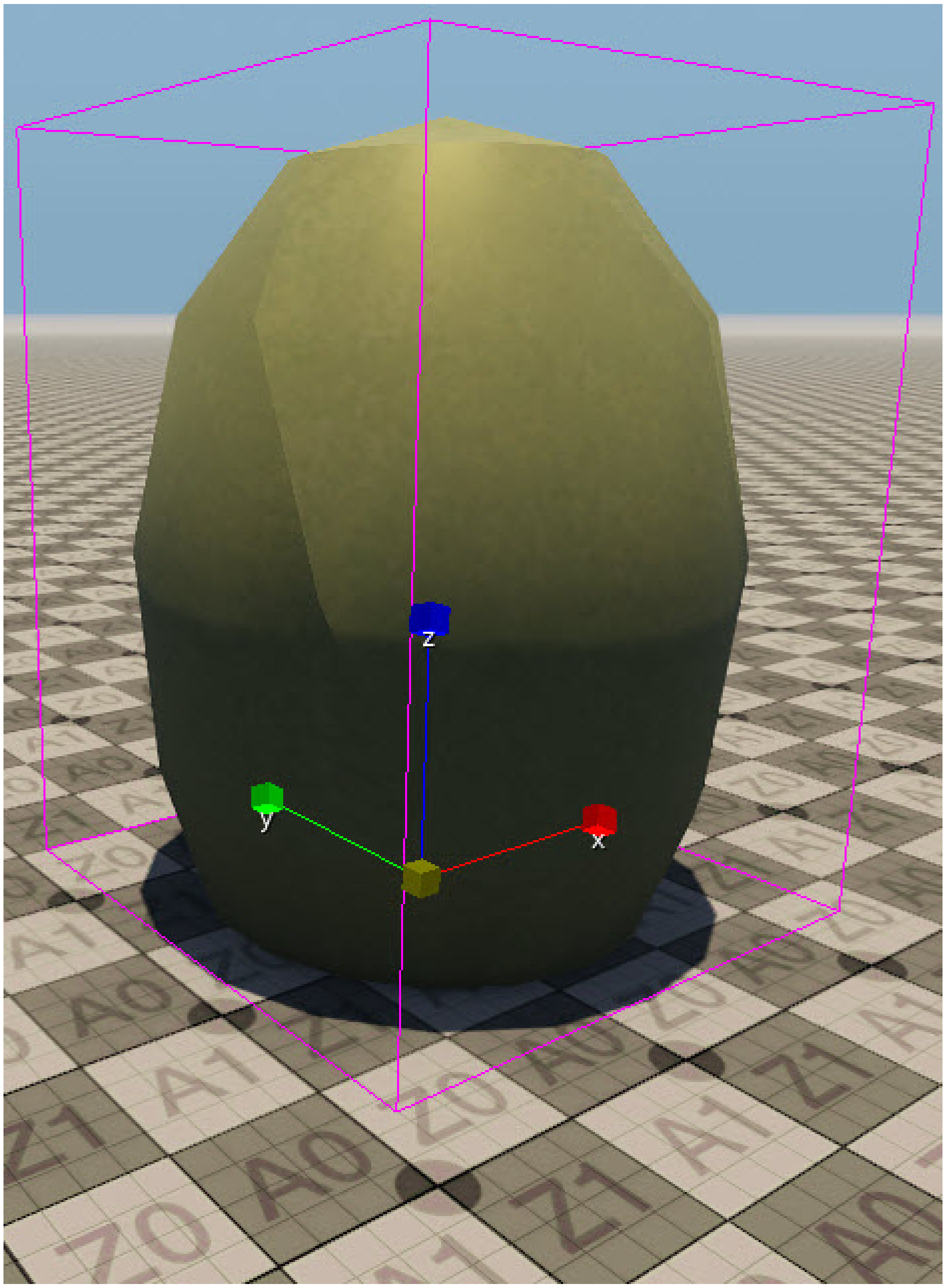
A larger outer circle also surrounds the entire gizmo. Select and drag this circle to rotate the object in relation to the screen display.



Scale

The **Scale** tool can select an object and change its size. The **Scale** gizmo has cubes on the X, Y, and Z lines.

To scale an object, select the X, Y, or Z line, then drag up or down to increase or decrease the scale of the object in the selected direction.



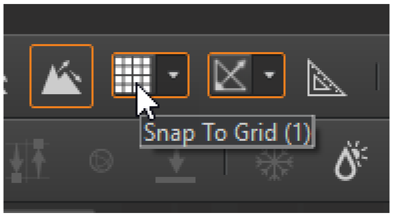
Snap Features

Lumberyard Editor includes snap features to help you precisely position objects.

Topics

Snap to Grid

When you move an object, you can use **Snap to Grid** to attract that object to points along a customizable grid. **Snap to Grid** is on by default.

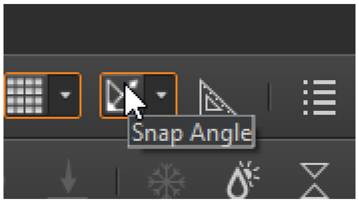


To use snap grid

1. To turn grid snap off or on, click the **Snap to Grid** icon on the toolbar.
2. To customize the size of the snap grid, click the arrow to the right of the **Snap to Grid** icon. Then select the preferred value to modify the distance betwen snap points.

Snap Angle

When you rotate an object, you can use **Snap Angle** (on by default) to attract that object to degrees of angle.



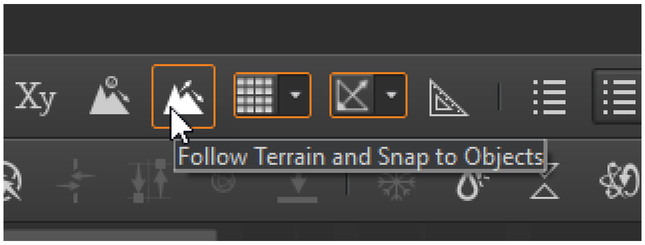
To use snap angle

1. To turn snap angle on or off, click the **Snap Angle** icon on the toolbar.
2. To customize the **Snap Angle**, click the arrow to the right of on the icon. Then select the preferred value to modify the degree of rotation with each snap.

Follow Terrain and Snap to Objects

Use **Follow Terrain and Snap to Objects** to move an object along terrain features rather than along the X, Y, Z axes or planes. With **Follow Terrain and Snap to Objects** on, you can freely move your object in any direction along your terrain, and the object automatically adjusts to terrain features.

In levels with a terrain mesh, this tool can be very useful, as you can easily keep your objects sitting directly on the terrain (or in whatever relation to the terrain you already have it) rather than having to adjust it manually to peaks and valleys.



To use Follow Terrain and Snap to Objects

- To turn terrain snapping on and off, click the **Follow Terrain and Snap to Objects** icon in the toolbar.

3D Level Navigation

The level navigation in the **Perspective** viewport is similar to that of other 3D modeling tools with first person shooter (FPS) controls. If you are familiar with FPS games, you should find it easy to navigate within the **Perspective** viewport.

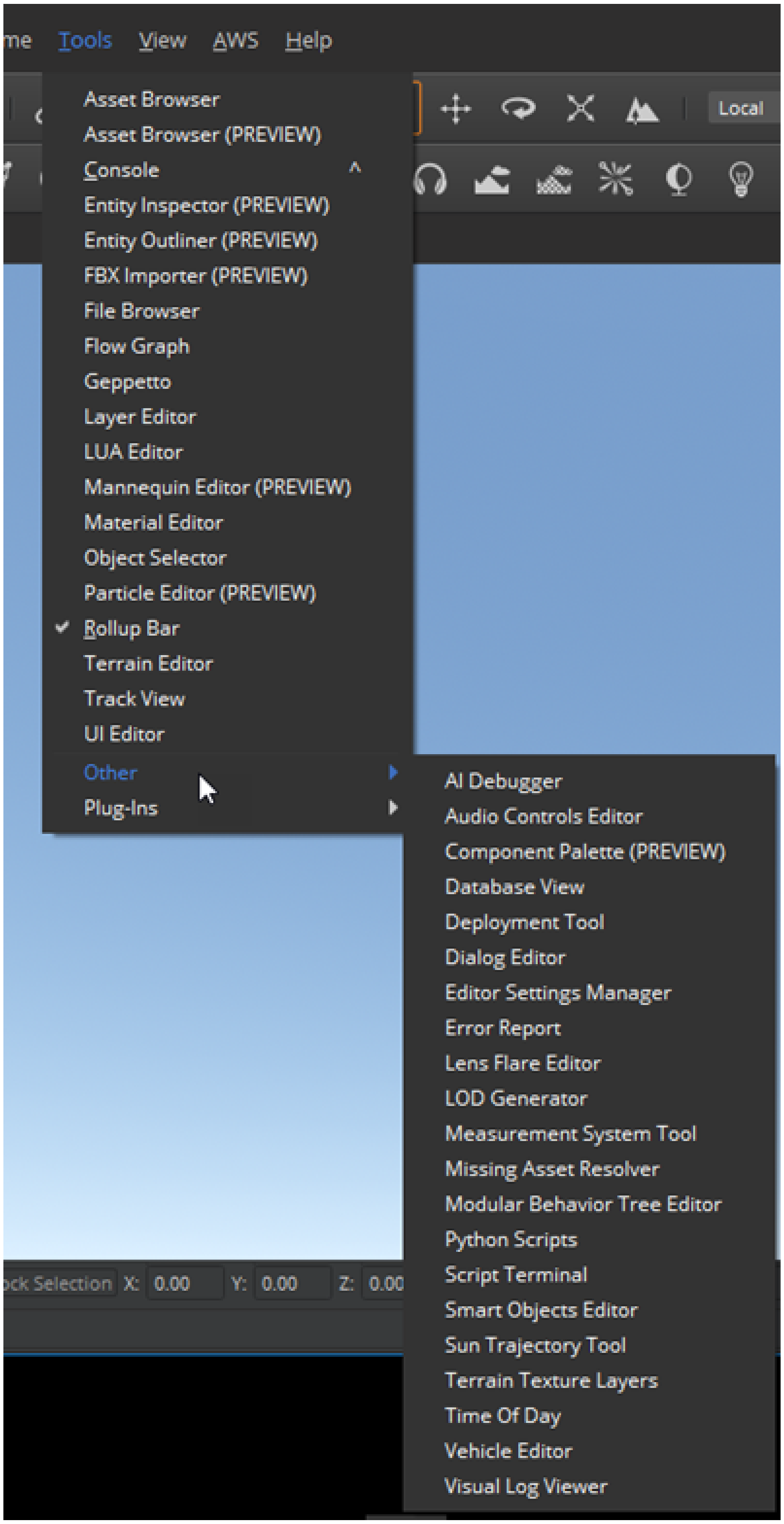
To navigate within your level in the **Perspective** Viewport, use the following click and drag actions:

Action	**Mouse Button(s) (*click and drag*) or Keystroke**
Select multiple objects	Left mouse button
Turn left or right, look up or down	Right mouse button
Pan left or right, pan up or down	Middle mouse button
Zoom in or out	Right mouse + middle mouse button *or* Mouse wheel
Strafe forward	**W**
Strafe backward	**S**
Strafe left	**A**
Strafe right	**D**

Editors

Lumberyard Editor features a collection of editor tools for building specific categories of content.

You can open any editor from the **Tools** menu.



You can also open the most commonly used editors from the editors toolbar.

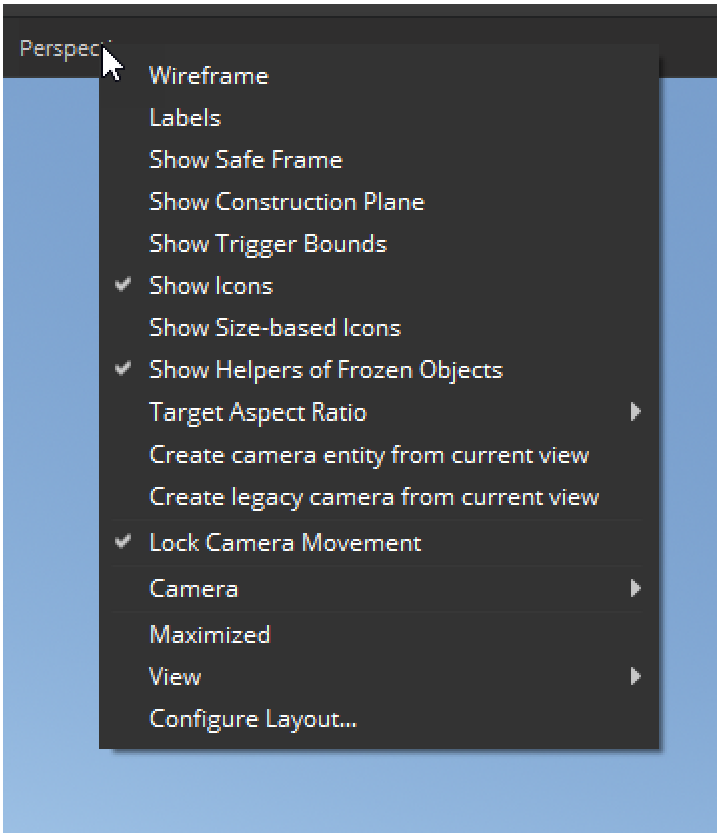
Display Options and Settings

You can use Lumberyard Editor's display options and settings to customize your view to see the most useful tools and options.

Topics

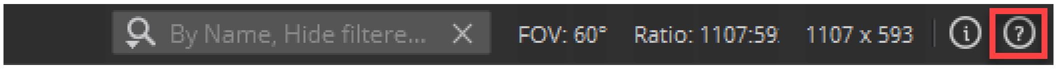
Perspective Viewport Options

To configure display options for the **Perspective** viewport, right-click the viewport title bar. Select or deselect options to suit your individual workflow preferences.



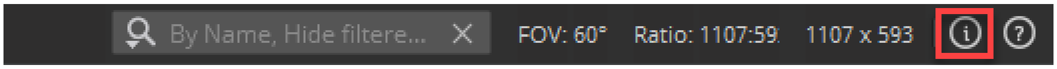
Show/Hide Helpers

The right side of the viewport title bar has additional display settings. Click the **H** (helper) icon to show or hide entity icons and their visual guidelines. Hiding these elements can declutter your view when you want to focus on other components.



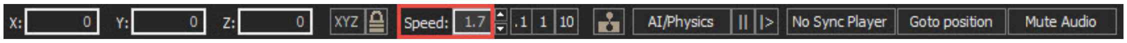
Toggle Display Information

To change the amount of debug/display information that is displayed in the **Perspective** viewport, click **i** (information) icon. Click this icon multiple times to choose the level of information you'd like to see.



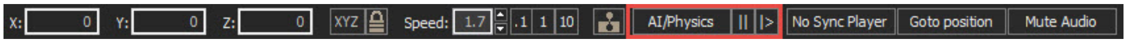
Navigation Speed Settings

You can adjust your **Perspective** viewport navigation speed. The **Speed** setting displays the current movement speed setting. Type a number into the **Speed** field, or click **.1** (slow), **1** (normal), or **10** (fast).



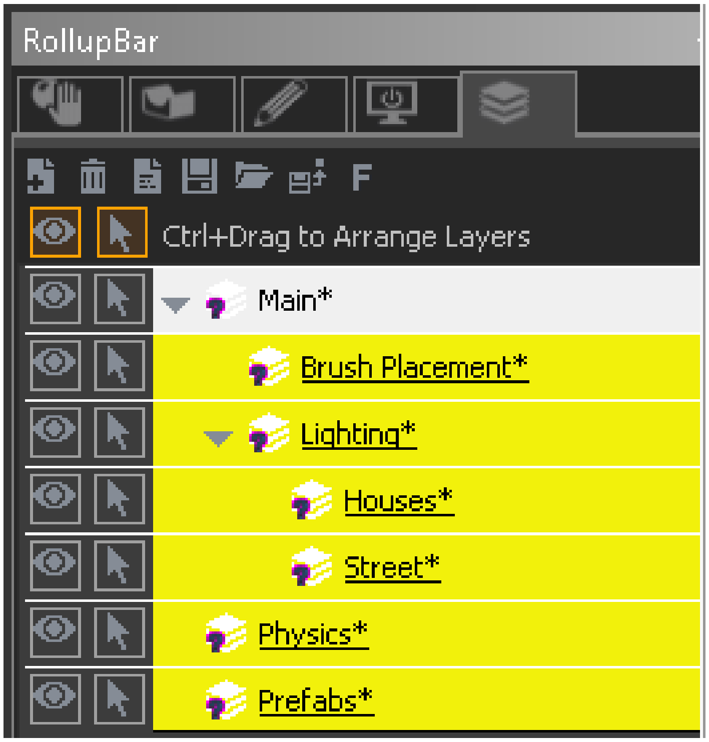
AI/Physics Toggle

AI/Physics turns on and off the movement events for physics, AI (artificial intelligence), and particles in edit mode. With these options, you can test and view these events without entering game mode. For more about modes, see Switching to Game Mode (p. 65).



Layers

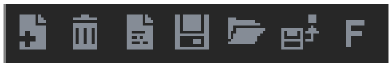
The **Layers** tab in the **Rollup Bar** helps you organize the large amount of content created when building a level.



Topics

- Using Layers Icons (p. 22)
- Working with Layers and Their Files (p. 23)
- Collaborating with Multiple Users (p. 23)
- Moving Assets Between Layers (p. 24)

Using Layers Icons

You can use the toolbar on the **Layers** tab to create new layers or delete, rename, save, and export your existing layers: 

Additionally, each layer has its own eye and arrow icons to help you manage your objects:

- **Eye icon** – Temporarily hides a layer in order to focus on a specific layer. Click the eye icon on each layer that you want to hide. Click it again to make the layer visible.
- **Arrow icon** – Disables the ability to select objects in that layer. This can be useful if you are having trouble selecting an object that is overlapped by objects in other layers.

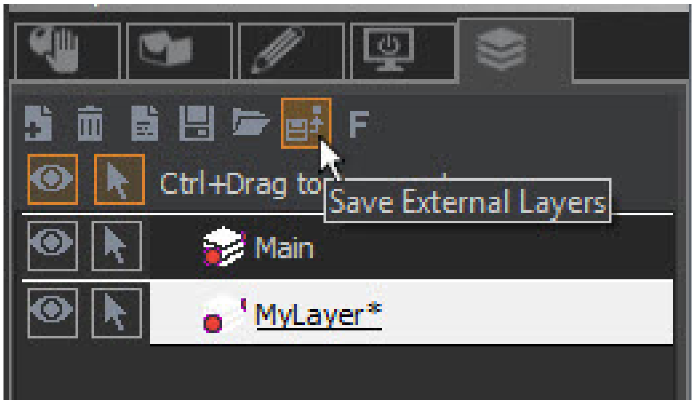
Furthermore, you can organize your layers into nested groups by holding **Ctrl** and dragging each layer to your preferred location.

Working with Layers and Their Files

When you create a new layer in a level, that layer is stored as a file in the `level\layers` directory with the extension `.lyr`.

To work within a specific layer, click the **Layer** tab, and then select the layer. With that layer selected, you can create and add content, all of which are automatically created as a part of that layer.

When working within a specific layer, you don't need to save the level file, but you do need to save the layer file. To save the layer file, click **Save External Layers** icon, as shown in the following image.



Collaborating with Multiple Users

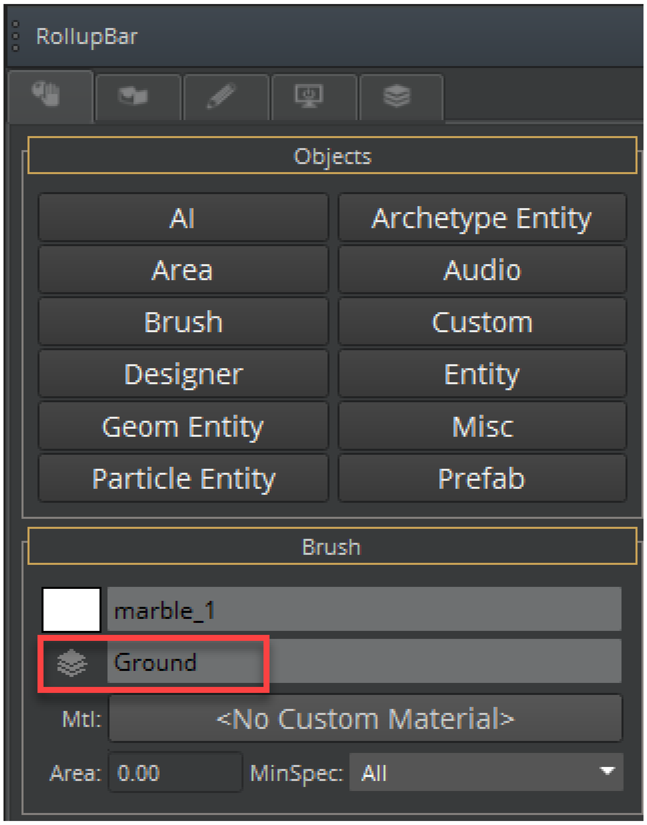
With the **Layers** tab multiple users can work within the same level. To do that, each user can create his or her own layer and build all the content within that layer. Although not required, a source control tool such as Perforce provides a useful way to manage these different layers; users just check files in or out to get the latest updates from other team members.

Moving Assets Between Layers

Each entity, brush, or designer object you place in the level is assigned to the currently selected layer. If you have not created any additional layers, objects are placed in the default main layer.

To assign an object to a different layer

1. Select the object in the **Perspective** viewport.
2. In the **Rollup Bar** click the **Objects** tab to view the name of the layer that contains the object.



marble_1

Ground

Mtl: <No Custom Material>

Area: 0.00 MinSpec: All

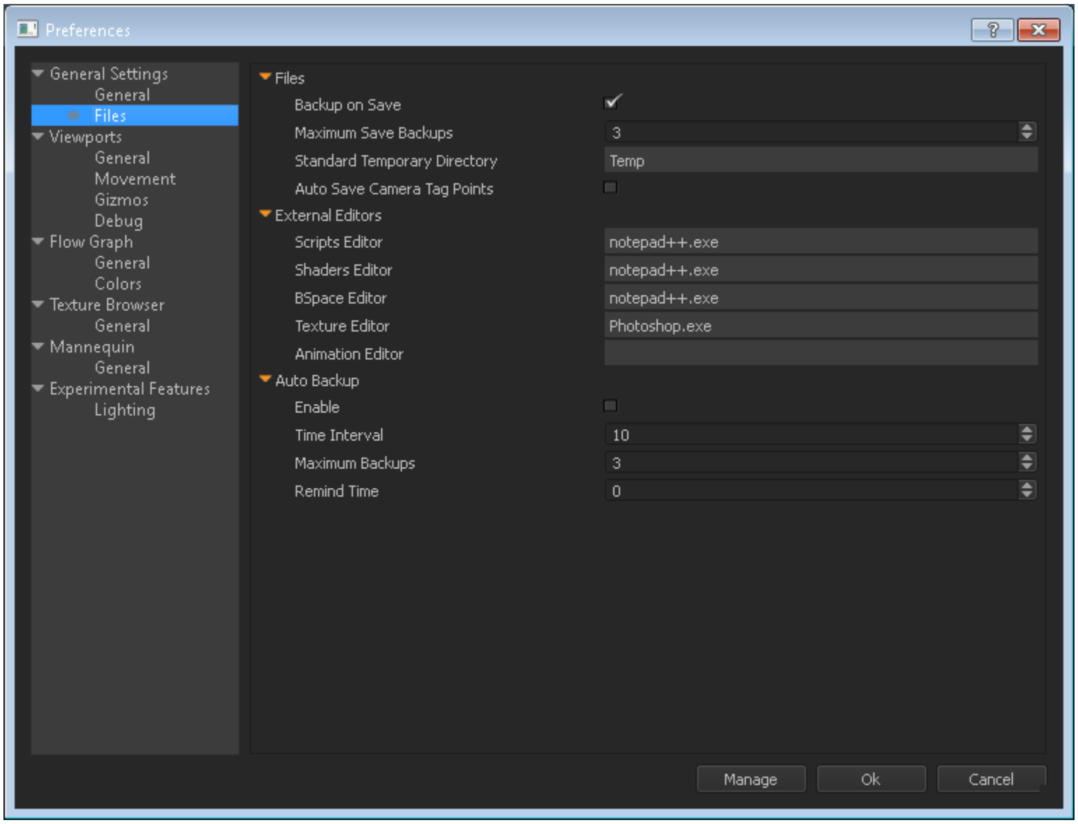
3. Click the **Layers** icon to display a list of the current layers for that level.
4. Select the destination layer from the list.

Auto Backup

Lumberyard's **Auto Backup** feature is on by default. **Auto Backup** saves your level file incrementally. This helps prevent loss of your work in case of unexpected problems.

To customize your Auto Backup settings

1. From the main menu, open **Edit**, **Editor Settings**, **Global Preferences**.
2. Under **General Settings**, click **Files**. From here, you can customize your **Auto Backup** settings.



Files
- Backup on Save ☑
- Maximum Save Backups: 3
- Standard Temporary Directory: Temp
- Auto Save Camera Tag Points ☐

External Editors
- Scripts Editor: notepad++.exe
- Shaders Editor: notepad++.exe
- BSpace Editor: notepad++.exe
- Texture Editor: Photoshop.exe
- Animation Editor:

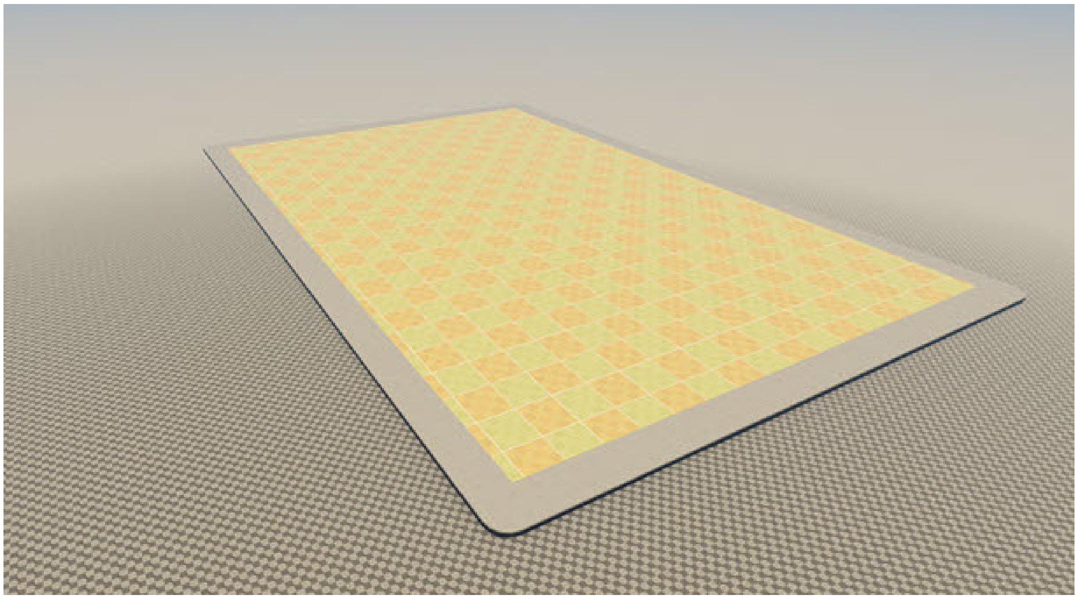
Auto Backup
- Enable ☐
- Time Interval: 10
- Maximum Backups: 3
- Remind Time: 0

Placing Brush Objects

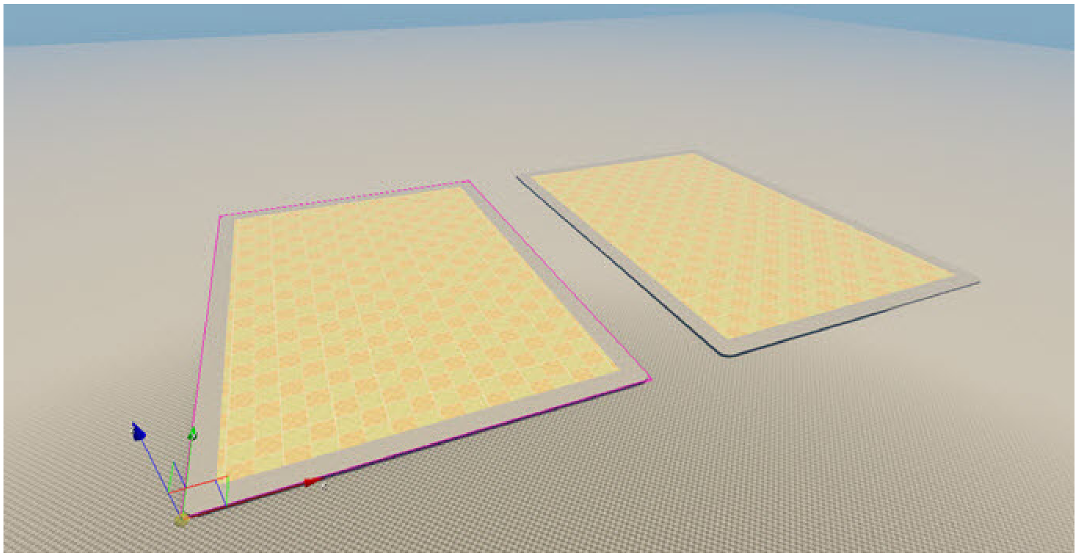
Placing objects in your level adds realism and interest to your environment. This tutorial teaches you how to start building your scene by adding a particular type of static 3D shape known as *brush* objects.

To place a brush

1. In the **Rollup Bar**, on the **Objects** tab ✋ , click **Brush** to display the assets you can add to the level.
2. Under **Browser**, expand `Objects\StyleTown\Natural\Terrain` and select `townblock`.
3. Drag the **townblock** object (the word *townblock*) into the **Perspective** viewport to add it to your level. Click to place your townblock.

4. In the **Perspective** viewport, select the townblock brush. When selected, a gizmo appears at the corner of your brush.
5. Press **Ctrl+D** to copy your townblock, and move your second townblock next to your first townblock. Click to place it.

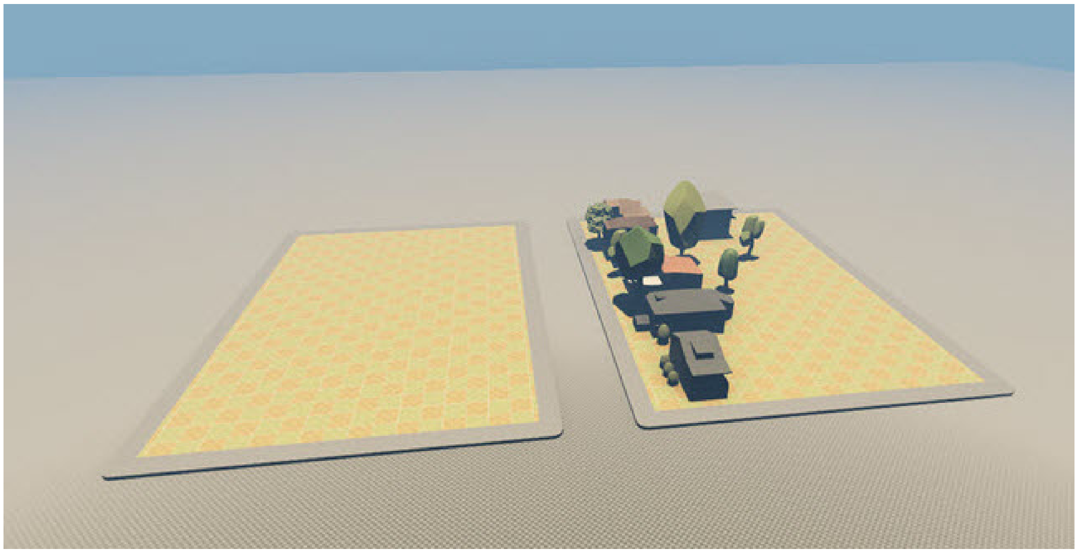


6. Click **Brush** again in the **Rollup Bar**.
7. Expand `StyleTown\Architecture\Buildings`. Drag four buildings onto one of your townblocks.

   Tip

   When dragging, you see only the pointer. After you release the mouse button, your brush appears.
8. Click **Brush** again.
9. Under **Browser**, expand `StyleTown\Natural\Vegetation`. Drag trees and bushes and place them around the buildings. Add as many as you would like.

   Your scene should now look similar to the following:



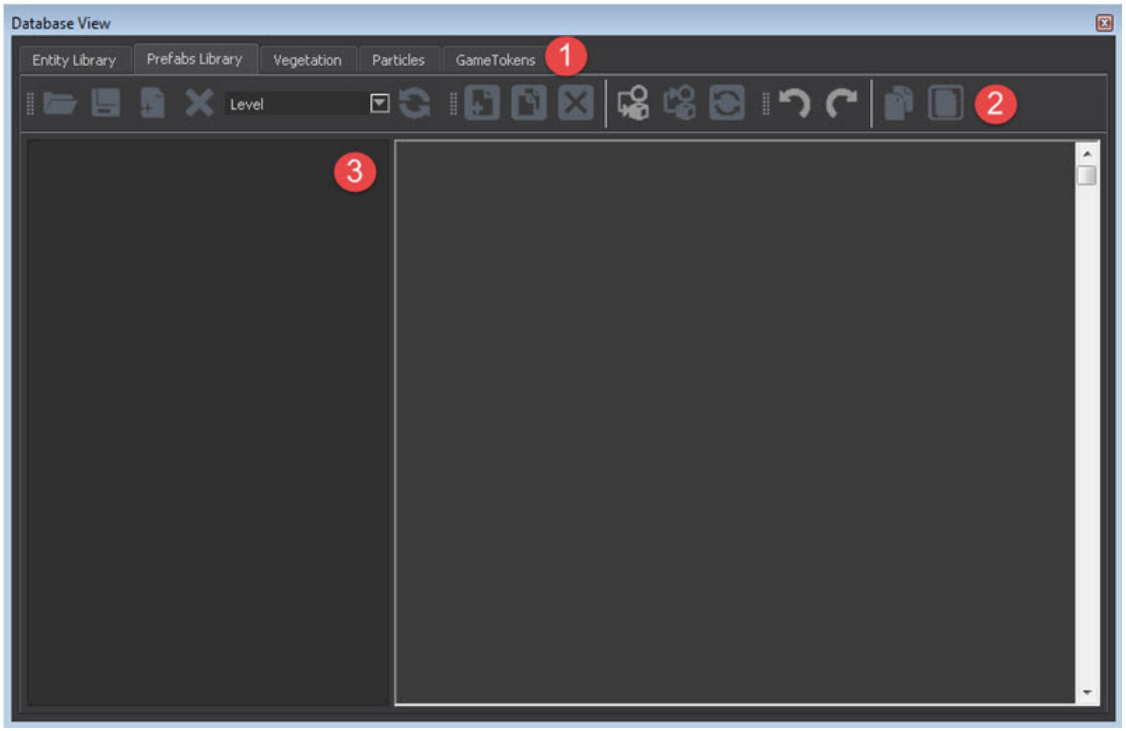
Using Prefabs

Prefabs are combinations of predefined assets that help you create content more quickly.

Using prefabs simplifies and speeds up the building of your environment.

To access previously created prefabs, you use the **Database View** editor. This editor has the following features:

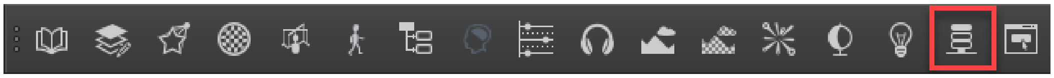
1. **Database tabs** – Database types you can view and manage:
   - **Entity Library**
   - **Prefabs Library**
   - **Vegetation**
   - **Particles**
   - **Game Tokens**
2. **Editor toolbar** – Tools to open, save, add, and remove prefabs
3. **File tree view** – Opened prefabs available in your level. This view is empty at first; the following procedure describes how to open and access prefab libraries so you can see the **NeighborhoodBlock** prefabs as shown in the following image.



To place prefabs

1. Do one of the following to open the **Database View** editor:

   - On the editor toolbar, click **Database View**.

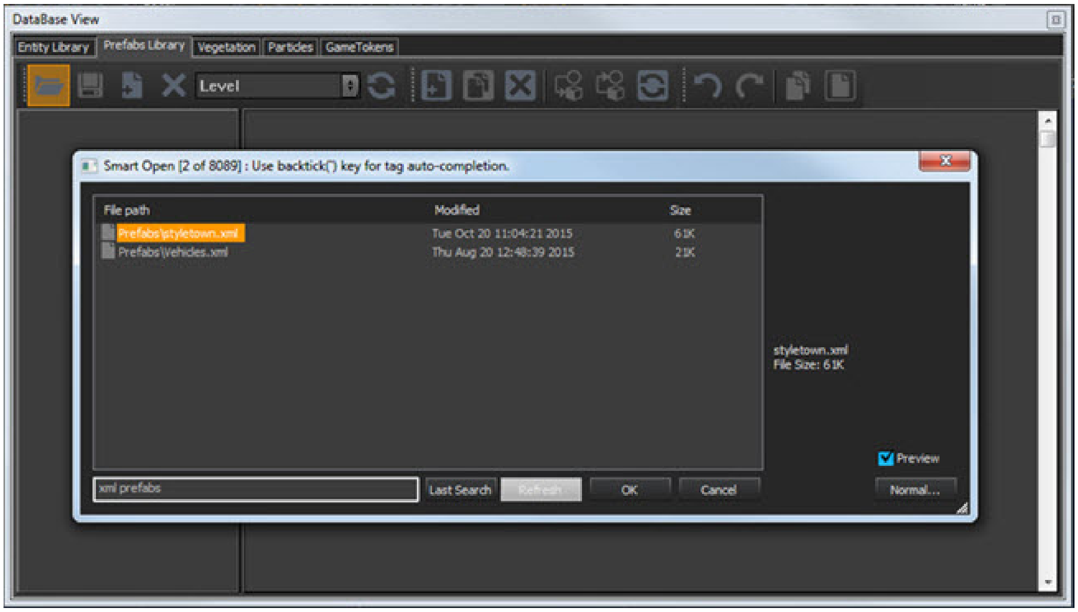
     

   - From the main menu, click **Tools**, **Other**, **Database View**.

2. Click the **Prefabs Library** tab.
3. Click **Load Library**

   .

4. Select `Prefabs\styletown.xml` and click **OK** to display the prefab library **NeighborhoodBlock**.



Smart Open [2 of 8089] : Use backtick(`) key for tag auto-completion.

File path	Modified	Size
Prefabs\styletown.xml	Tue Oct 20 11:04:21 2015	61K
Prefabs\Vehicles.xml	Thu Aug 20 12:48:39 2015	21K

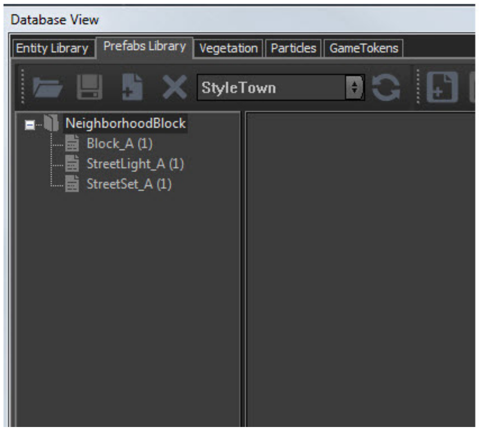
styletown.xml
File Size: 61K

xml prefabs

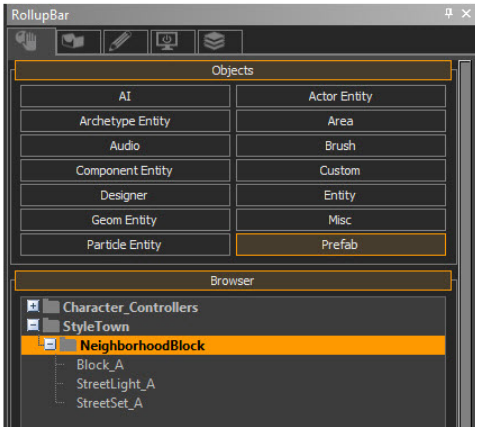
Tip

After you use this method to add a prefab library to a level, these prefabs appear when you click **Prefab** in the **Rollup Bar**. This makes it faster for you to place prefabs in a level.

Database View



Rollup Bar View



5. Drag **StreetSet_A** into the **Perspective** viewport. Position the street so that it fits around the two town blocks.
6. Drag **Block_A** into the **Perspective** viewport.

   Align this block of houses and trees over the empty block.
7. Drag **StreetLight_A** into the **Perspective** viewport. Position it along the center street.
8. Save your level file.

   You now have something that looks like this:

Building the Terrain

You can use Lumberyard Editor to apply materials to the terrain, modify the terrain height, and use the vegetation tool to paint trees.

Topics

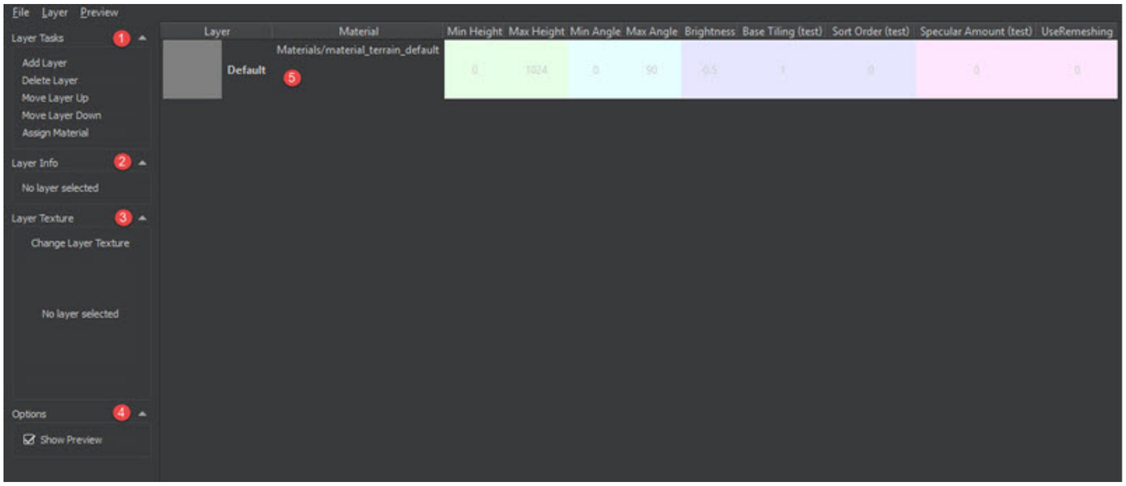
Painting the Terrain

Now that you have the basics of the scene built with brushes, you'll start building the surrounding terrain environment. To do this, you'll use two new editors: The **Terrain Texture Layers** editor and the **Material Editor**.

Terrain Texture Layers Editor

The **Terrain Texture Layers** editor defines the materials used to paint on the level's terrain mesh. The Terrain Texture Layers editor has the following features:

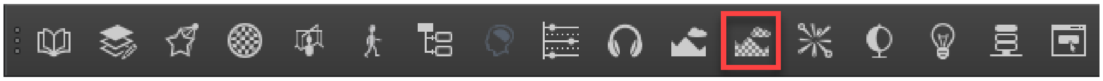
1. **Layer Tasks** – Controls for adding, deleting, and reordering layers in the **Layer** list
2. **Layer Info** – Information about the selected layer, including the layer size and surface type count
3. **Layer Texture** – Low-detail texture swatch; displays distant textures and color information for the surface texture
4. **Options** – Settings related to the **Layer** list
5. **Layer list** – Layer textures available for painting onto the terrain (such as dirt, grass, rocks, and more)



Layer	Material	Min Height	Max Height	Min Angle	Max Angle	Brightness	Base Tiling (test)	Sort Order (test)	Specular Amount (test)	UseRemeshing
Default	Materials/material_terrain_default	0	1024	0	90	0.5	1	0	0	0

To open the **Terrain Texture Layers** editor

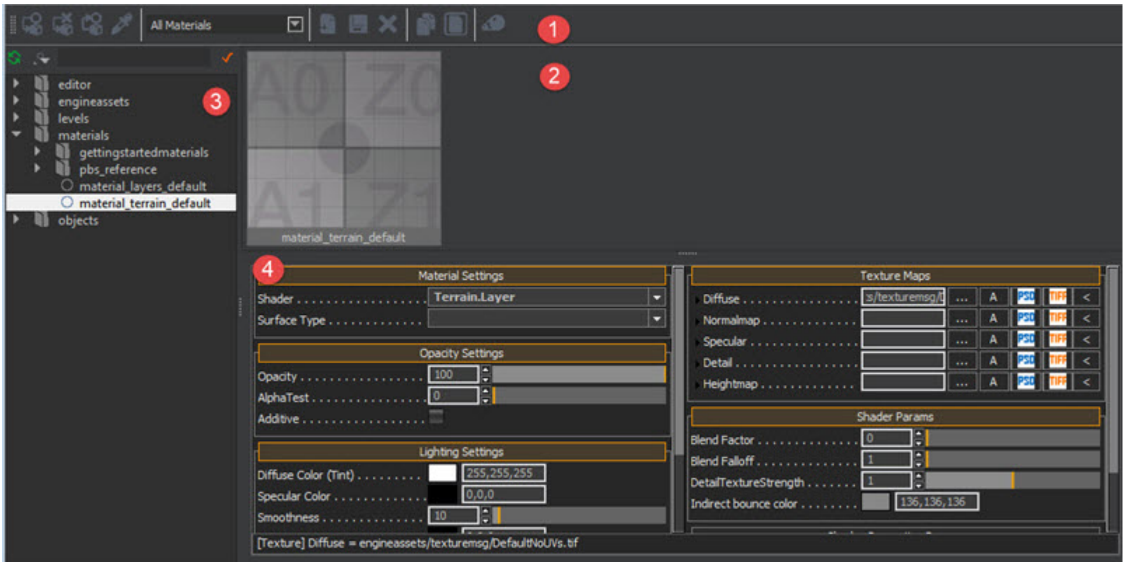
Do one of the following:

- On the editor toolbar, click the **Terrain Textures Layers Editor** icon

- From the main menu, open **Tools**, **Other**, **Terrain Texture Layers**

Material Editor

The **Material Editor** has the following features:

1. **Editor toolbar** – Tool list for applying, deleting, saving, and creating materials
2. **Material preview** – Display for the selected material's appearance
3. **Material folder directory** – Folder tree to navigate through the materials available for use in the level
4. **Material properties and settings** – Options for defining the material's appearance

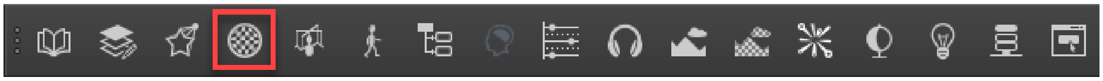


Shader . . . . Terrain.Layer
Surface Type . . . .
Opacity 100
AlphaTest 0
Additive . . . .
Diffuse Color (Tint) 255,255,255
Specular Color 0,0,0
Smoothness 10
Diffuse . . . . s/texturemsg/
Normalmap . . . .
Specular . . . .
Detail . . . .
Heightmap . . . .
Blend Factor 0
Blend Falloff 1
DetailTextureStrength 1
Indirect bounce color 136,136,136
[Texture] Diffuse = engineassets/texturemsg/DefaultNoUVs.tif

To open the **Material Editor**

Do one of the following:

- On the editor toolbar, click the **Material Editor** icon



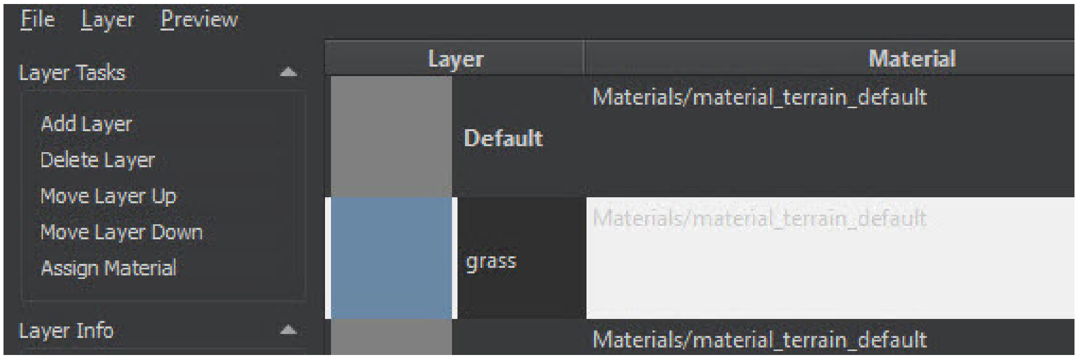
- From the main menu, open **Tools**, **Material Editor**

Building the Surrounding Environment

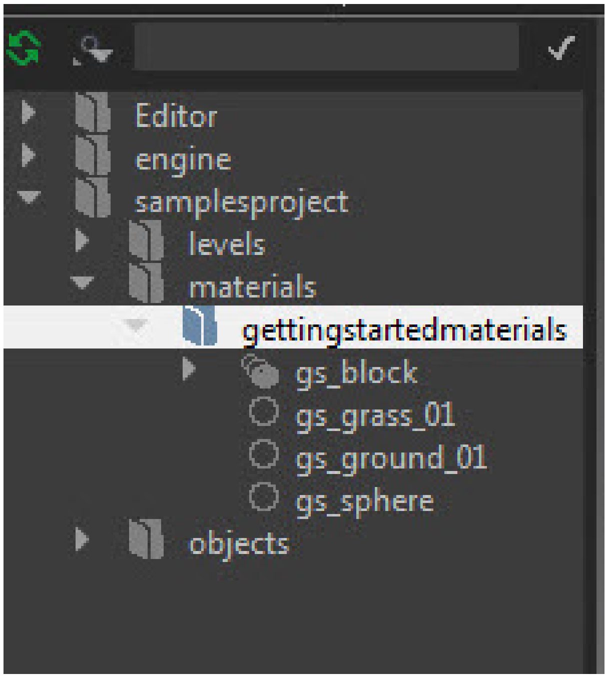
You are ready to assign materials to your terrain mesh.

To build the surrounding terrain environment

1. Open the **Terrain Texture Layers** editor and in the **Layer Tasks** area, click **Add Layer** twice to add two new layers.
2. In the first new layer, double-click **NewLayer** and rename it `grass`.

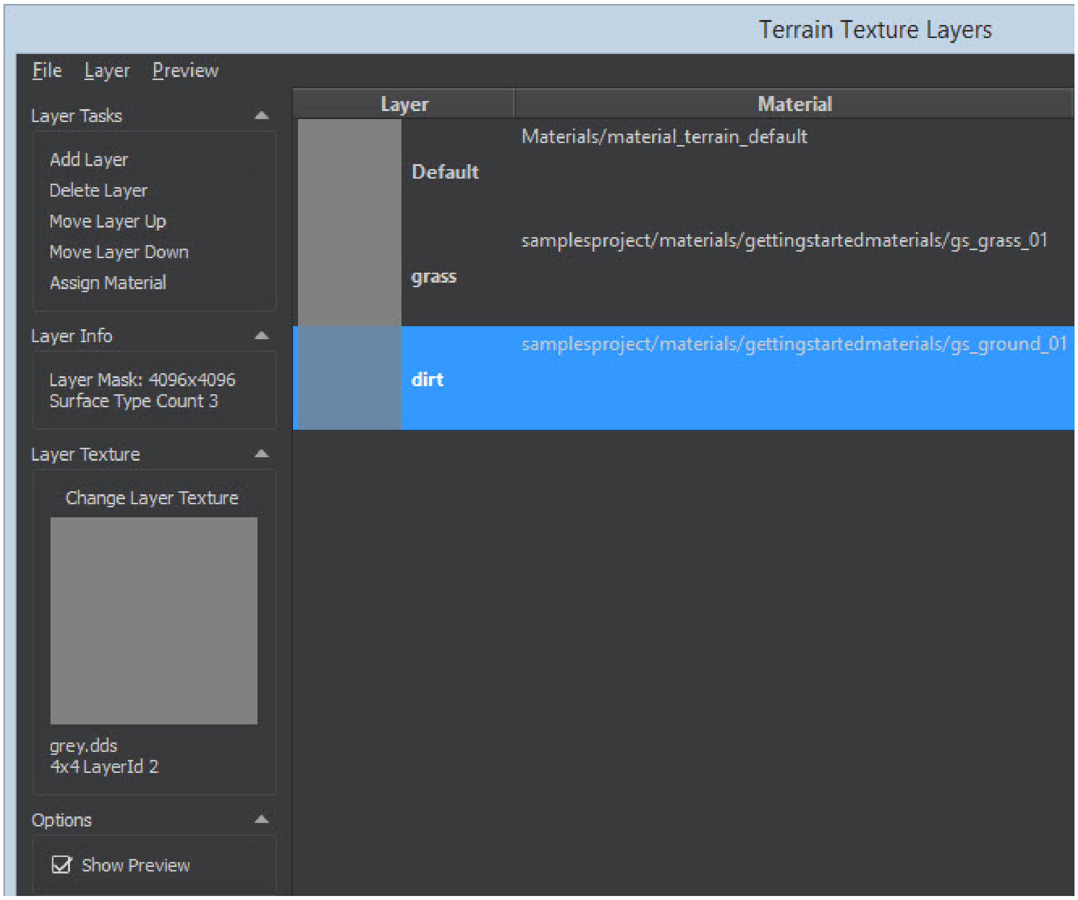


3. In the second new layer, double-click **NewLayer** and rename it **`dirt`**.
4. Click the **grass** layer to select it.
5. Open the **Material Editor** and select the material **gs_grass_01** located in the following path: `materials\gettingstartedmaterials\gs_grass_01`.



6. In the **Terrain Texture Layers** editor, the grass layer should still be selected. In the **Layer Tasks** area, click **Assign Material**.
7. Switch back to the **Material Editor** and select **gs_ground_01** in the `gettingstartedmaterials` folder.
8. Switch to the **Terrain Texture Layers** editor. Select the **dirt** layer and click **Assign Material** in the **Layer Tasks** area.

> **Note**
> In each of the terrain material layers is a small material preview box. This material preview box displays the assigned layer texture, not the material assigned from the material editor. For this tutorial, we are using the default `grey.dds` file, so both the grass and dirt layers appear with the gray layer texture.



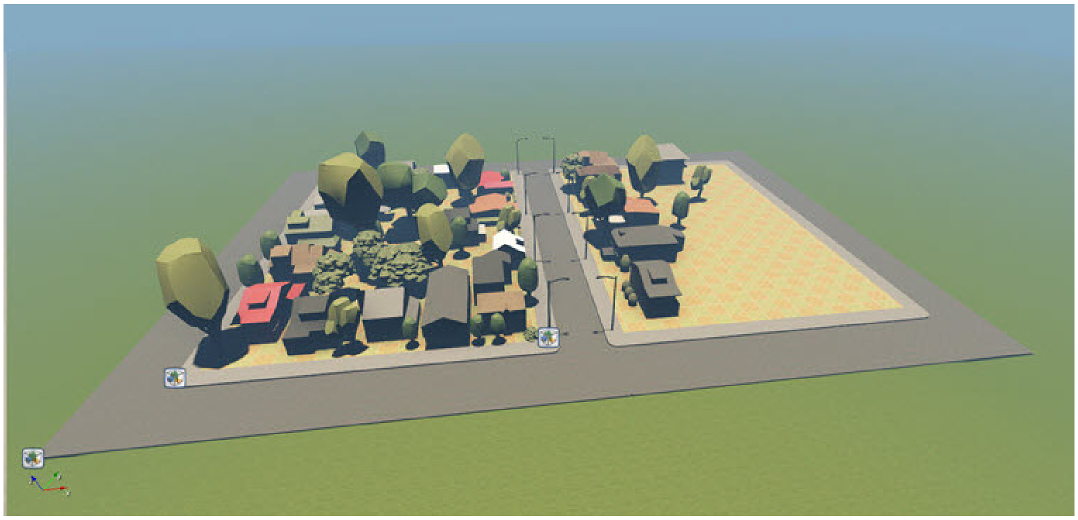
Layer	Material
Default	Materials/material_terrain_default
grass	samplesproject/materials/gettingstartedmaterials/gs_grass_01
dirt	samplesproject/materials/gettingstartedmaterials/gs_ground_01

Layer Mask: 4096x4096
Surface Type Count 3

grey.dds
4x4 LayerId 2

9. Close the **Material Editor** and **Terrain Texture Layer** editors.

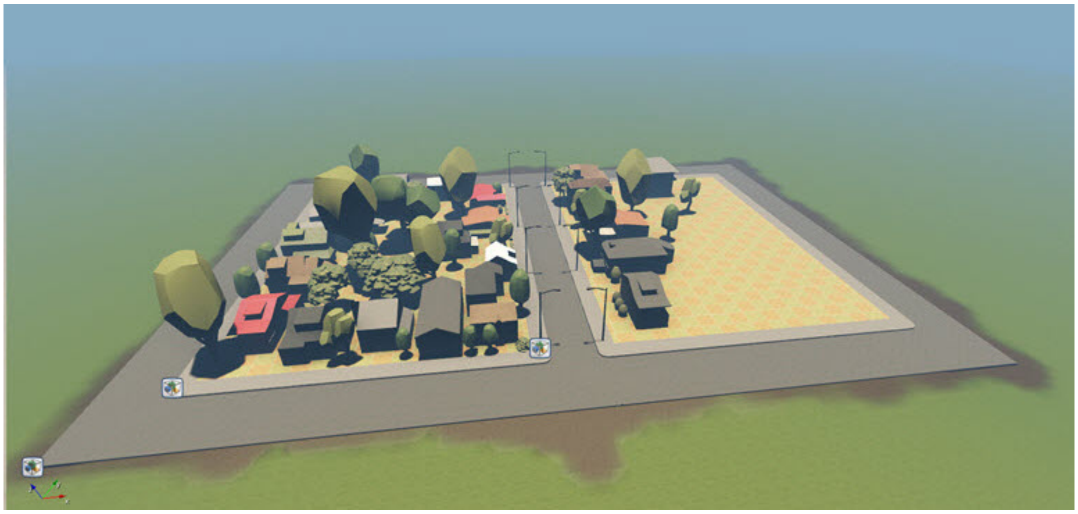
   You are now ready to paint grass and dirt textures onto the terrain.
10. In the **Rollup Bar**, click the **Terrain** tab, and then click **Layer Painter** to display the terrain layer painting tools.
11. The bottom of the **Layer Painter** section shows a list of the terrain materials that you have created: grass and dirt. Select **grass**.
12. Just above that list is **Vertex Coloring** with a **Color** box (white is the default). Click the color box and change the RGB color values to **`145`**, **`180`**, **`75`** for a grass-green color. Click **OK**.
13. Click **Flood** at the bottom of the **Layer Painter** section. The terrain is now covered in the grass texture and looks similar to this:



You can now paint some dirt into the scene around the perimeter of the street.

14. Select the **dirt** material at the bottom of the **Layer Painter** section.
15. Adjust the **Color** box to a brown tone: RGB **`115`**, **`95`**, **`50`**. Click **OK**.
16. For **Brush Settings**, set the **Radius** to **`5`** and the **Hardness** to **`0.5`**.
17. Click in the **Perspective** viewport. Drag to paint the dirt texture around the perimeter of the street. Do as little or as much as you like.

    When you are finished, you will have a town similar to the following:



18. Save your level file.

Adjusting Terrain Height

After you paint your terrain, you can manipulate its height with the **Modify Terrain** settings.

The **Modify Terrain** section features the following:

Brush Settings

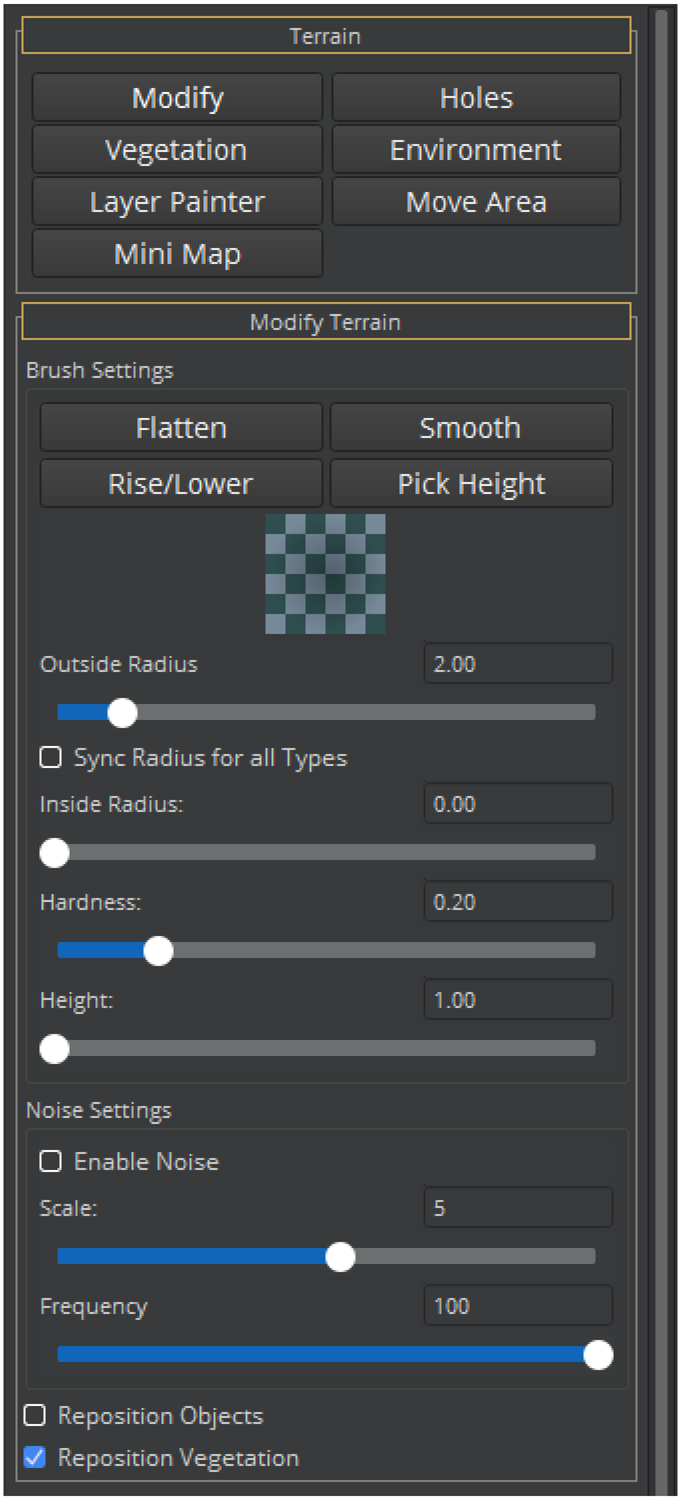
Use the following parameters to adjust the rise/lower, smooth, and flatten brushes.

- **Flatten** – Flatten the terrain to the desired height setting
- **Smooth** – Soften the terrain down to a smoother surface
- **Rise/Lower** – Raise or lower the terrain based on brush size settings
- **Pick Height** – Find and set height based on existing terrain geometry
- **Outside Radius** – Set brush size for painting
- **Sync Radius for all Types** – Set the same outer radius value across the flatten, smooth, and rise/lower brushes
- **Inside Radius** – Set how round or flat the brush is in relation to the outside radius setting
- **Hardness** – Soften or harden the outer brush settings
- **Height** – Set the brush height

Noise Settings

Use noise settings to add random terrain variances to the brush.

- **Scale** – Modify the strength of the noise effect. Higher values produce more noise
- **Frequency** – Set how often the effect is applied



Outside Radius 2.00

□ Sync Radius for all Types

Inside Radius: 0.00

Hardness: 0.20

Height: 1.00

□ Enable Noise

Scale: 5

Frequency 100

□ Reposition Objects

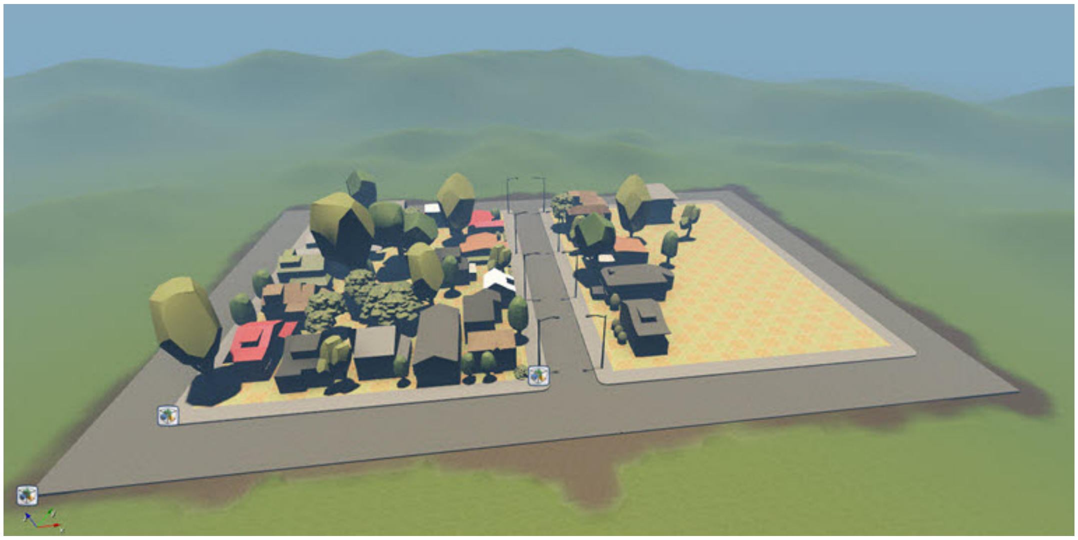
☑ Reposition Vegetation

To modify terrain height

1. In the **Rollup Bar**, click the **Terrain** tab and then click **Modify**.
2. Under **Modify Terrain**, click **Rise/Lower** and use the following settings to create gentle hills in your scene:

- **Outside Radius** = `25`
- **Inside Radius** = `1`
- **Hardness** = `0.25`
- **Height** = `3`

3. In the **Perspective** viewport, navigate towards the outer perimeter of the terrain map and click or drag to paint on the terrain. Experiment with clicking and dragging along the terrain to manipulate the terrain to different heights. Build some larger hills of different sizes and shapes.
4. Modify the brush settings to the following:
   - **Inside Radius** = `20`
   - **Hardness** = `1`
   - **Height** = `1`
5. Paint again on the terrain. Notice how the terrain rises up straight and rigid.
6. Click the **Smooth** tool, and use the following settings:
   - **Outside Radius** = `25`
   - **Hardness** = `0.2`
7. Paint with the smooth brush over the last area of terrain you created. Notice the smoothing of the terrain.
8. Click **Pick Height** and click on a high point on the terrain. Notice that the **Height** setting in the tool adjusts to the height you clicked on.
9. Select a point on the terrain where the height was unchanged. The **Height** setting in the **Modify Terrain** tool changes to 32 (the default terrain height). This tool does not affect your terrain directly, but simply adjusts settings for the next step.
10. Click **Flatten**, and use the following settings:
    - **Outside Radius** = `25`
    - **Inside Radius** = `0`
    - **Hardness** = `1`
11. Paint with the **Flatten** tool over the area you just smoothed. The terrain flattens to the same height as the rest of the default terrain height.
12. Using the terrain height tools, create a range of high and long hills in the distant background beyond the neighborhood block area that you created.
13. Adjust your brush settings to create smaller rolling hills closer to the neighborhood block. Use the **Smooth** tool to soften where you like. With a few minutes' work, you have something like this:



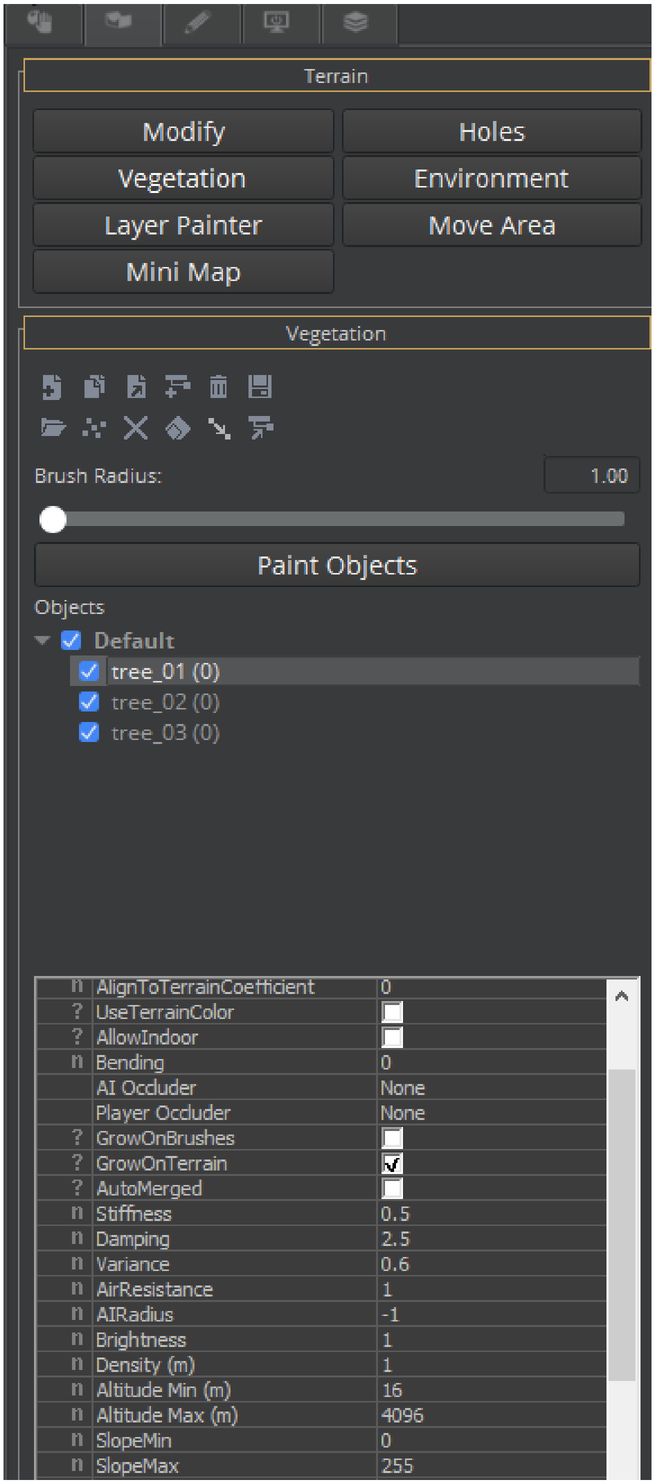
14. Save your level file.

Adding Terrain Vegetation

You can use the **Vegetation** tool to paint 3D mesh objects like trees, shrubs, and grasses onto the terrain. Various settings help you to build organic environments using any type of 3D models you define.

The **Vegetation** tool offers the following settings:

- **Toolbar** – Tools to create, modify, and organize vegetation types
- **Brush Radius** – Size of the brush used to paint vegetation into the level
- **Paint Objects** – Switch that enables painting in the level
- **Objects** – List of vegetation objects
- **Table of attributes** – List of attributes that can be modified for each vegetation object



Brush Radius: 1.00

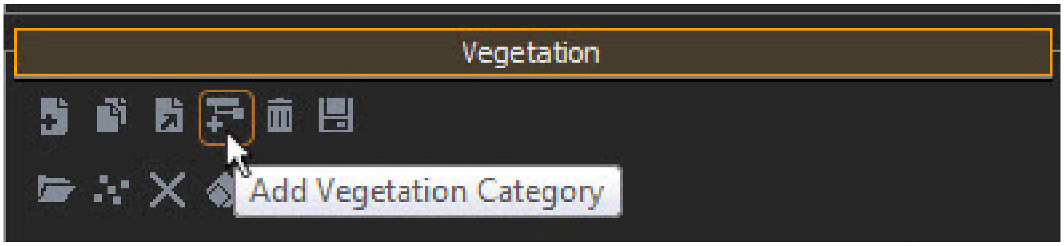
- ☑ Default
  - ☑ tree_01 (0)
  - ☑ tree_02 (0)
  - ☑ tree_03 (0)

n	AlignToTerrainCoefficient	0
?	UseTerrainColor	☐
?	AllowIndoor	☐
n	Bending	0
	AI Occluder	None
	Player Occluder	None
?	GrowOnBrushes	☐
?	GrowOnTerrain	☑
?	AutoMerged	☐
n	Stiffness	0.5
n	Damping	2.5
n	Variance	0.6
n	AirResistance	1
n	AIRadius	-1
n	Brightness	1
n	Density (m)	1
n	Altitude Min (m)	16
n	Altitude Max (m)	4096
n	SlopeMin	0
n	SlopeMax	255

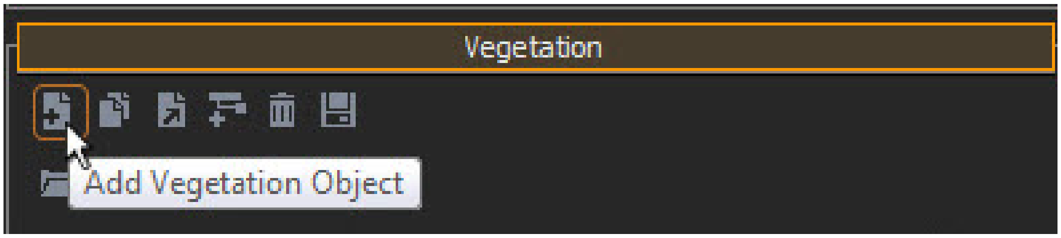
In the previous steps, you textured and modified your terrain. Now you'll use the **Vegetation** tool to add some trees.

To add trees

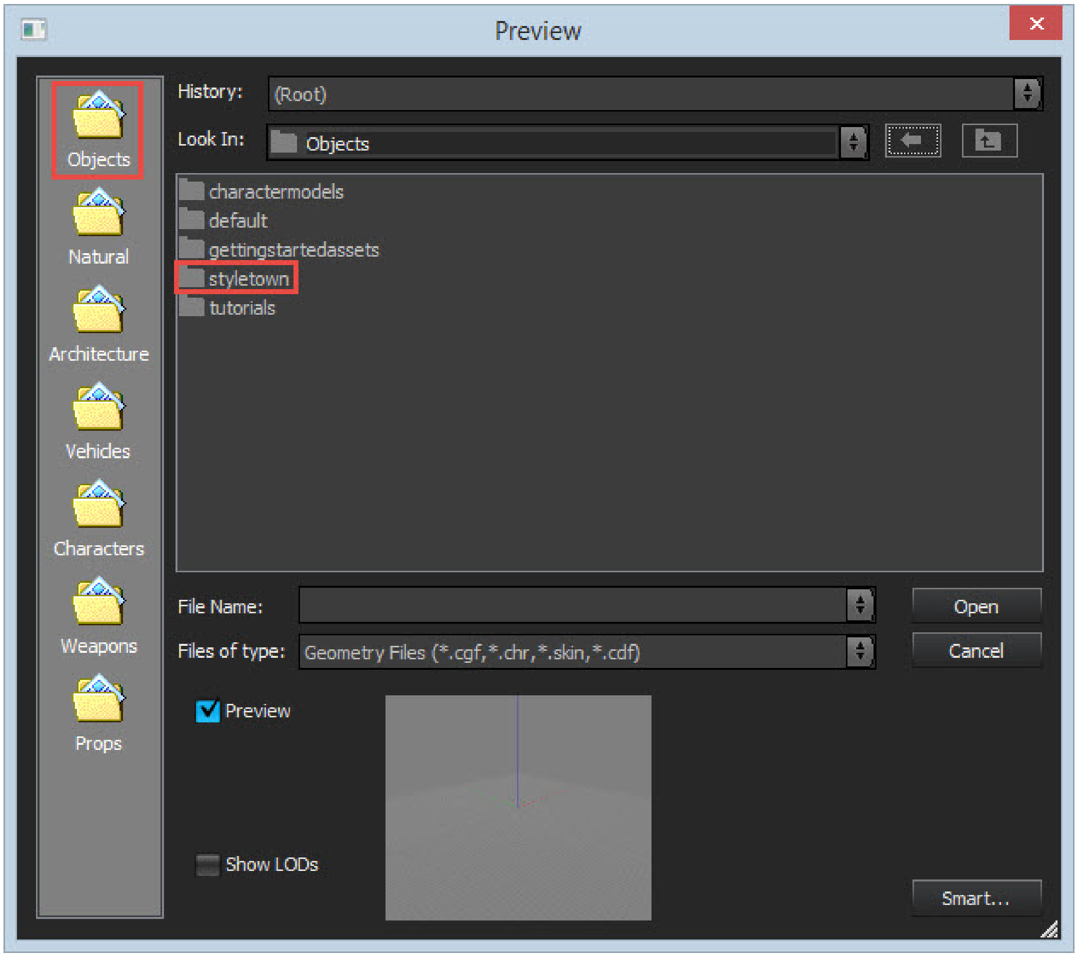
1. In the **Rollup Bar**, on the **Terrain** tab, click **Vegetation**.
2. In the tools list, click **Add Vegetation Category**.



3. When prompted to name the **New Category**, type `Trees`. Click **OK**.
4. In the **Objects** list, select the **Trees** category you just created.
5. In the tool list, click **Add Vegetation Object**.



6. In the **Preview** dialog, click the `Objects` folder and then open the `StyleTown` folder.



History: (Root)

Look In: Objects

Files of type: Geometry Files (*.cgf,*.chr,*.skin,*.cdf)

7. Open the `Natural` folder, and then `Vegetation`.
8. In the list of .cgf files, use **Ctrl+click** to select the following files: `tree_01`, `tree_02`, `tree_04`, and `tree_06`. Click **Open**.

   If you paint trees into the environment at this point, every tree would appear with the default brush settings, providing no variation on size, rotation, or spread of the trees.
9. To create variation in tree size, rotation, and spread, select all the trees in the list and change the following settings in the attributes list.

   +- Size Var = `0.2`

   Amount of variation in tree size

   Random Rotation = Selected (checked)

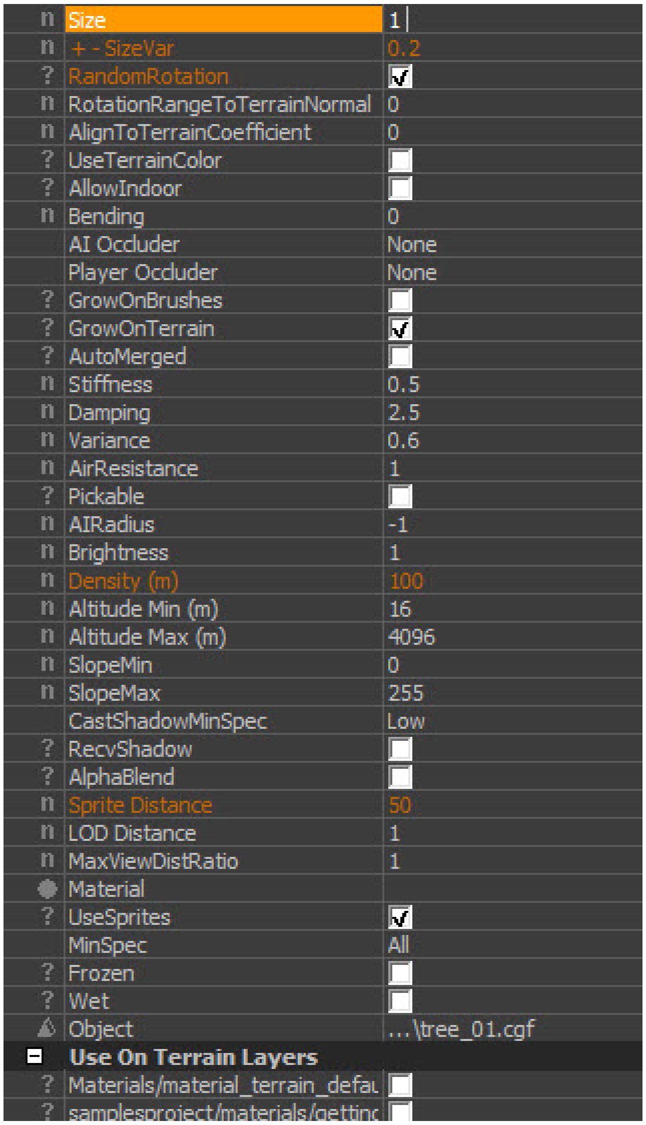
   Random rotation of trees as they're placed

   Density = `100`

   Density of the trees

   Sprite Distance = `50`

   Distance from the camera view that vegetation transitions from a mesh to a sprite of that object

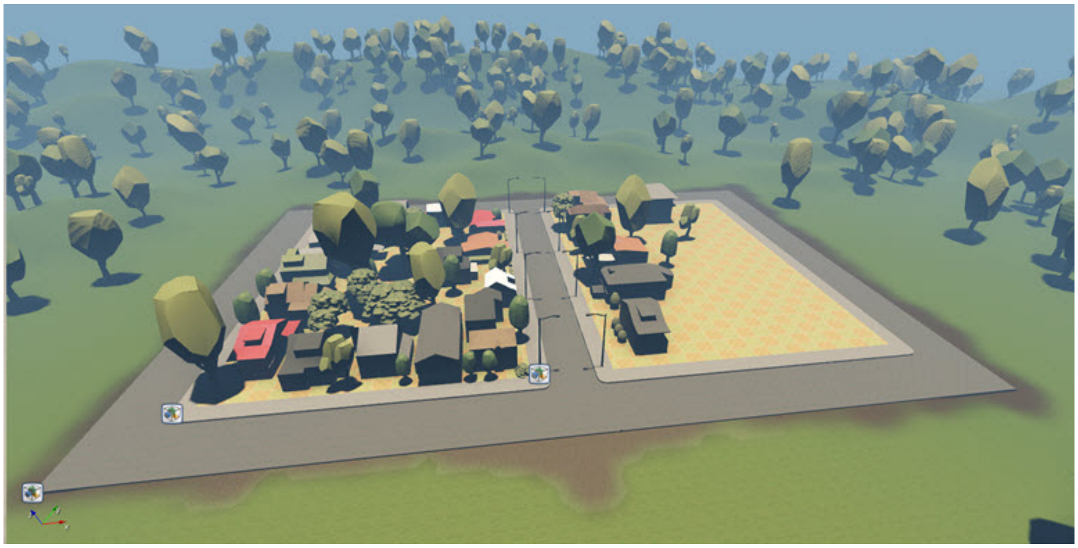


Size	1
+ - SizeVar	0.2
RandomRotation	☑
RotationRangeToTerrainNormal	0
AlignToTerrainCoefficient	0
UseTerrainColor	□
AllowIndoor	□
Bending	0
AI Occluder	None
Player Occluder	None
GrowOnBrushes	□
GrowOnTerrain	☑
AutoMerged	□
Stiffness	0.5
Damping	2.5
Variance	0.6
AirResistance	1
Pickable	□
AIRadius	-1
Brightness	1
Density (m)	100
Altitude Min (m)	16
Altitude Max (m)	4096
SlopeMin	0
SlopeMax	255
CastShadowMinSpec	Low
RecvShadow	□
AlphaBlend	□
Sprite Distance	50
LOD Distance	1
MaxViewDistRatio	1
Material	
UseSprites	☑
MinSpec	All
Frozen	□
Wet	□
Object	...\tree_01.cgf
Use On Terrain Layers	
Materials/material_terrain_defau	□
samplesproject/materials/gettinc	□

10. Click the **Trees** group name. Adjust the **Brush Radius** to `50` (this size is appropriate for filling the terrain space quickly).
11. Click **Paint Objects**, then place your pointer in the **Perspective** viewport and click or drag to paint your trees.

    Depending on the tree density you want, you can click once and place a random group of trees, or you can drag through the space and paint them along a path. Adjust the **Brush Radius** and **Density** settings to change the number of trees painted.

    Your neighborhood scene now looks similar to the following:



12. Save your level file.

    You have created your first level environment with a small neighborhood between rolling hills amid a green forest.

Lighting the Scene

Amazon Lumberyard offers the tools and features used to light a scene, including environment probes, time of day settings, and lights. This tutorial walks you through the use of each.

Topics

Placing Environment Probes

Before you add lights, it's helpful to learn about environment probes and time of day settings, which affect lighting and appearance in your level. Environment probes are critical in achieving great-looking lighting. For example, you may have noticed that the shadows cast by objects are very dark. Adding an environment probe helps to produce more realistic ambient shadow intensity and reflections.

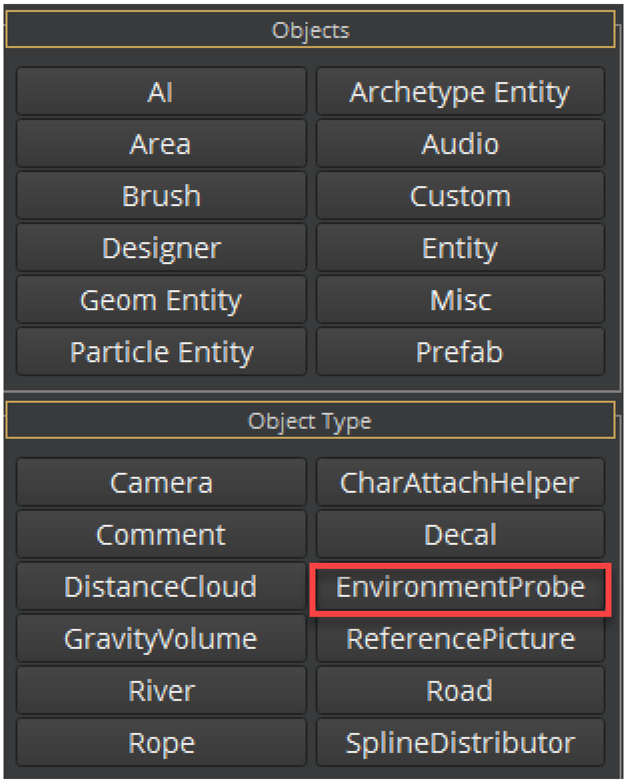
Environment probes are important for a variety of features including reflections, ambient diffuse values, particle diffuse values, and shadow colors. When building a level file, start by placing a global environment probe, and later place multiple environment probes to achieve the right visual quality for the space. After you place an environment probe, you will use the **Generate Cubemap** function which creates three textures in `textures\cubemaps\your_level`—one for the diffuse map, one for the specular map, and one for the source .dds file. A cubemap is a set of six squares that represent reflections from the environment. The six squares form the faces of an imaginary cube that surrounds an object. This step add realism to your level by incorporating object reflections.

This tutorial helps you set up the global probe.

To add a global probe

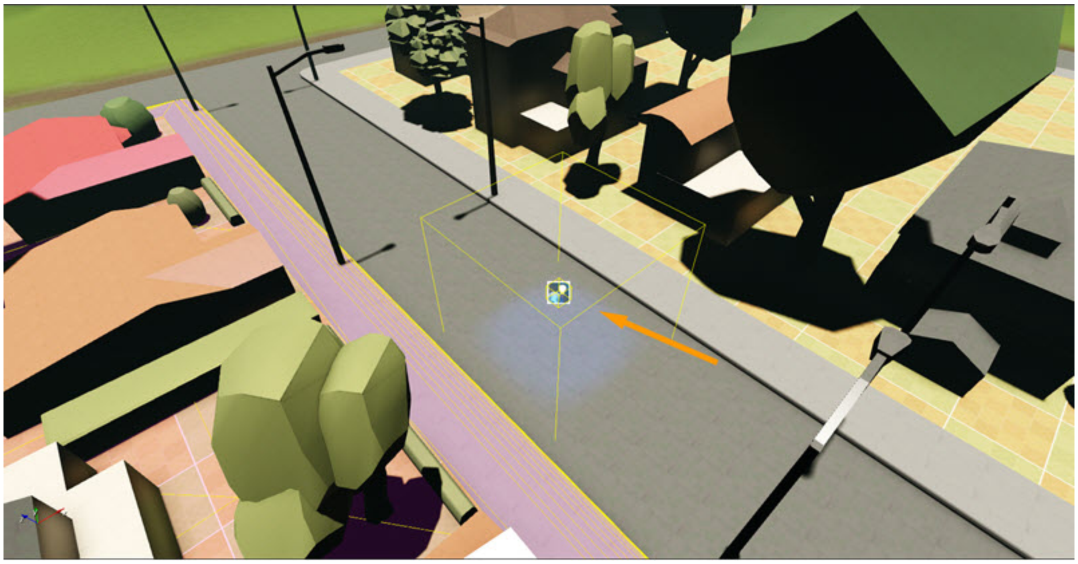
1. In the **Rollup Bar**, click the **Objects** tab

   and then click **Misc**.
2. Under **Object Type**, click **EnvironmentProbe**. Move the pointer into the **Perspective** viewport and then click to place the probe.

   The environment probe appears as a square volume entity.

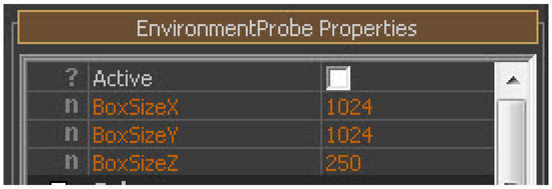
   > **Tip**
   > If you cannot see the environment probe as a square volume entity, click the **H** icon in the upper right corner of your viewport to display objects.
3. Position the probe approximately in the center of the level.



4. Under **EnvironmentProbe Properties**, apply the following settings:

   - **BoxSizeX** = **1024**
   - **BoxSizeY** = **1024**
   - **BoxSizeZ** = **250**

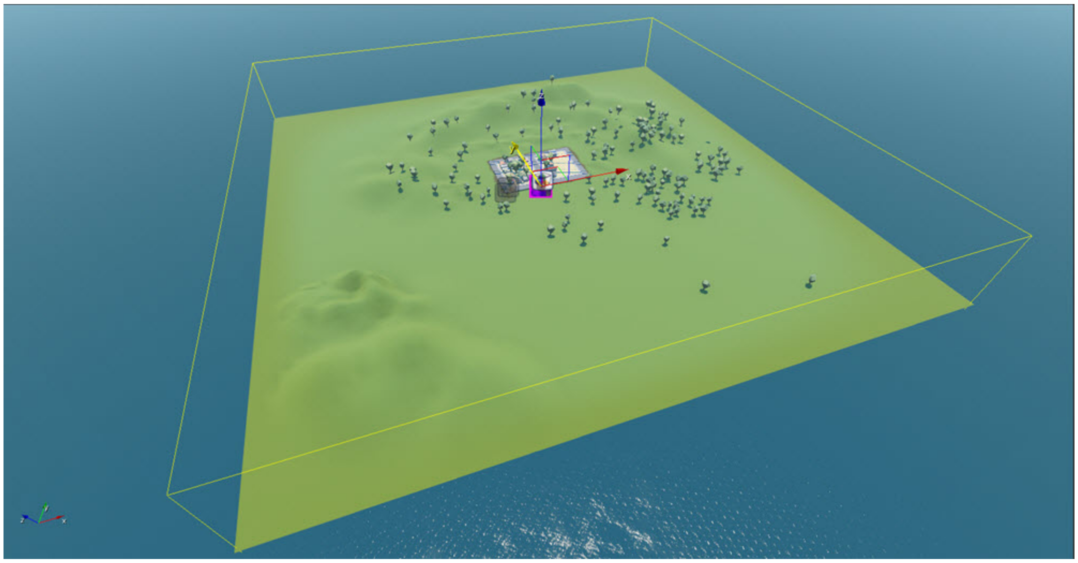
   These settings sets size of the probe to the size of the entire map.



EnvironmentProbe Properties

?	Active	□
n	BoxSizeX	1024
n	BoxSizeY	1024
n	BoxSizeZ	250

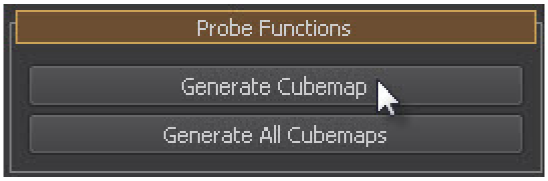
5. With the probe still selected, zoom out from the level so that you can see the entire map. If needed, adjust the position of the probe so that its volume covers the entire level.



6. To turn on the probe, select **Active** in the **EnvironmentProbe Properties**.

7. To generate the cubemap, under **Probe Functions** (above **EnvironmentProbe Properties**), click **Generate Cubemap**.

   The scene is briefly colored magenta as the probe renders the scene. When finished, the most visible change is that the shadows appear less dark.

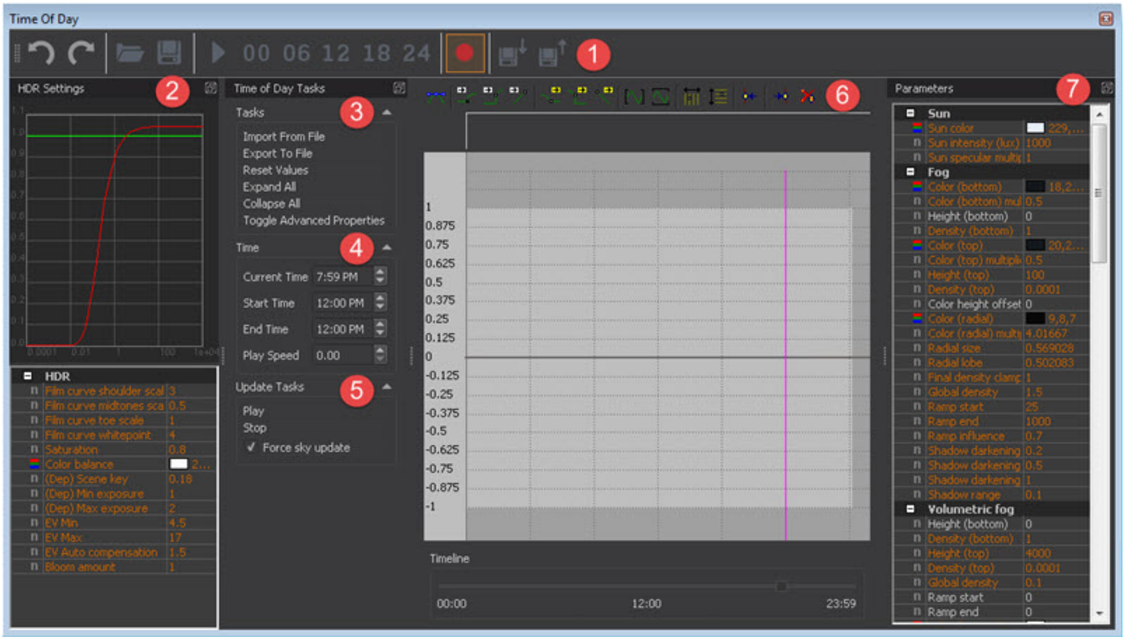
   > **Tip**
   > Adjust the values for **Color: Diffuse** (RGB values) and **Color: DiffuseMultiplier** to see how it affects your scene.

Adjusting Time of Day

Use lighting tools to adjust and animate the time of day. The **Time of Day** editor features a variety of tools to adjust and manage time of day settings. This tutorial focuses only on changing the time of day.

The **Time of Day** editor has the following features:

1. **Editor toolbar** – Icon toolbar for most common functions: undo, redo, import, export
2. **HDR Settings** – Settings to manage HDR (high dynamic range) lighting
3. **Time of Day Tasks** – Management of basic tasks within the **Time of Day** editor
4. **Current Time** – Display of start and end times as well as play speed
5. **Update Tasks** – Controls to update the play or stop of time of day, based on play speed setting
6. **Timeline** – Management of light settings along a 24-hour time cycle
7. **Parameters** – Lighting settings to adjust time of day conditions



Current Time 7:59 PM
Start Time 12:00 PM
End Time 12:00 PM
Play Speed 0.00

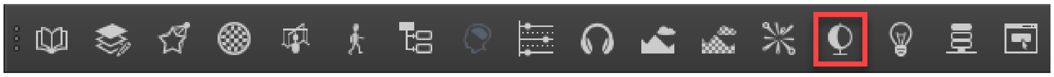
HDR	
Film curve shoulder scal	3
Film curve midtones sca	0.5
Film curve toe scale	1
Film curve whitepoint	4
Saturation	0.8
Color balance	2...
(Dep) Scene key	0.18
(Dep) Min exposure	1
(Dep) Max exposure	2
EV Min	4.5
EV Max	17
EV Auto compensation	1.5
Bloom amount	1

Sun	
Sun color	229,...
Sun intensity (lux)	1000
Sun specular multi	1
Fog	
Color (bottom)	18,2...
Color (bottom) mul	0.5
Height (bottom)	0
Density (bottom)	1
Color (top)	20,2...
Color (top) multipli	0.5
Height (top)	100
Density (top)	0.0001
Color height offset	0
Color (radial)	9,8,7
Color (radial) multi	4.01667
Radial size	0.569028
Radial lobe	0.502083
Final density clamp	1
Global density	1.5
Ramp start	25
Ramp end	1000
Ramp influence	0.7
Shadow darkening	0.2
Shadow darkening	0.5
Shadow darkening	1
Shadow range	0.1
Volumetric fog	
Height (bottom)	0
Density (bottom)	1
Height (top)	4000
Density (top)	0.0001
Global density	0.1
Ramp start	0
Ramp end	0

To adjust time of day

1. Do one of the following to open the **Time of Day** editor:

   - On the editor toolbar, click the **Time of Day** editor icon.

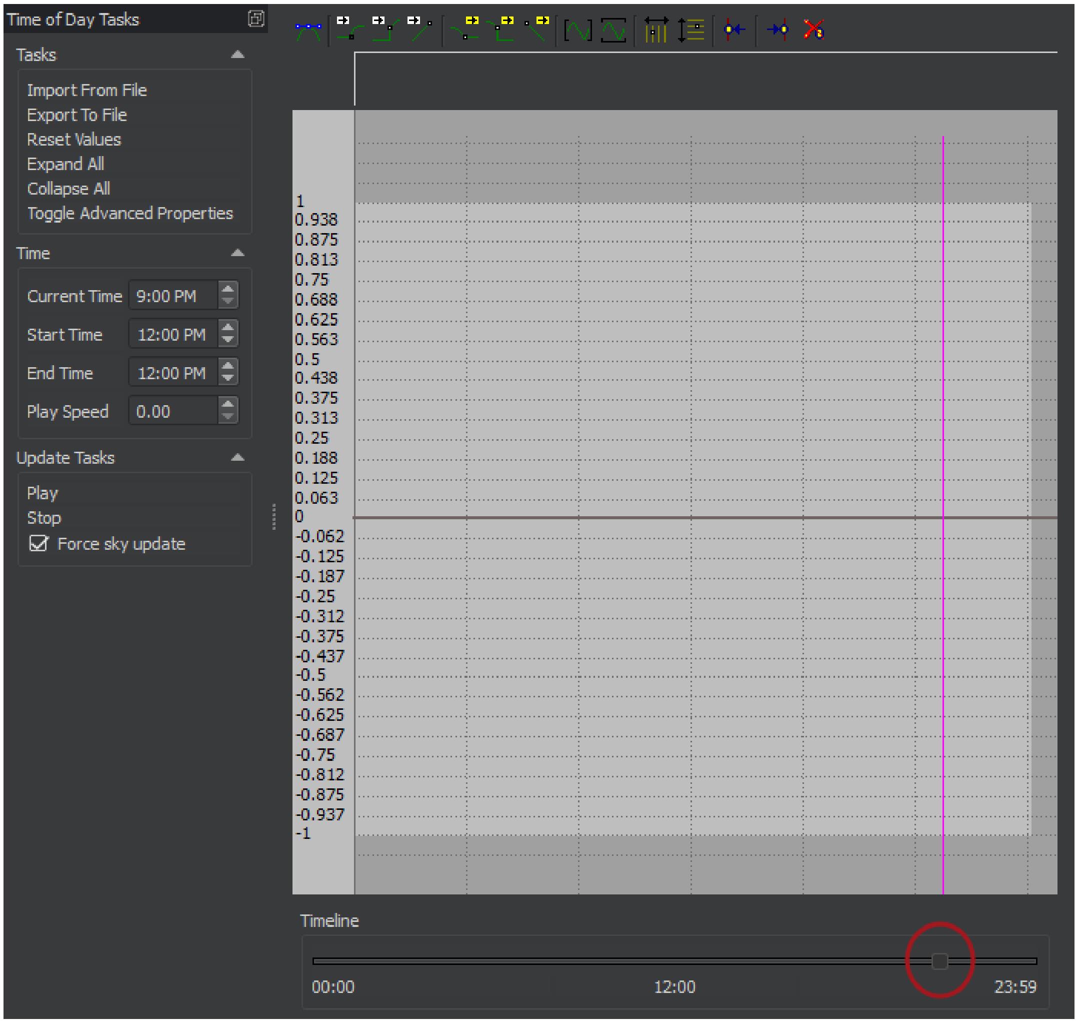
   

   - From the main menu, click: **Tools**, **Other**, **Time of Day**.

2. In the **Tasks** area, choose **Import from File**. Navigate to `SamplesProject\Levels\GettingStartedFiles` and open `TimeOfDay.xml`.

   This loads a set of time of day settings created for this tutorial. Notice the changes in light, fog, and sky colors.

3. In the timeline, drag the timeline marker (gray bar) to the number **21** (indicating 21:00 or 9:00 p.m.). As you drag the timeline marker, you can watch your scene in the **Perspective** viewport change from day to night.



Current Time 9:00 PM
Start Time 12:00 PM
End Time 12:00 PM
Play Speed 0.00
☑ Force sky update

4. Adjust the timeline marker to different times of day to see the lighting changes in your scene. Observe how this time adjustment also changes the settings in the **Parameters** area.

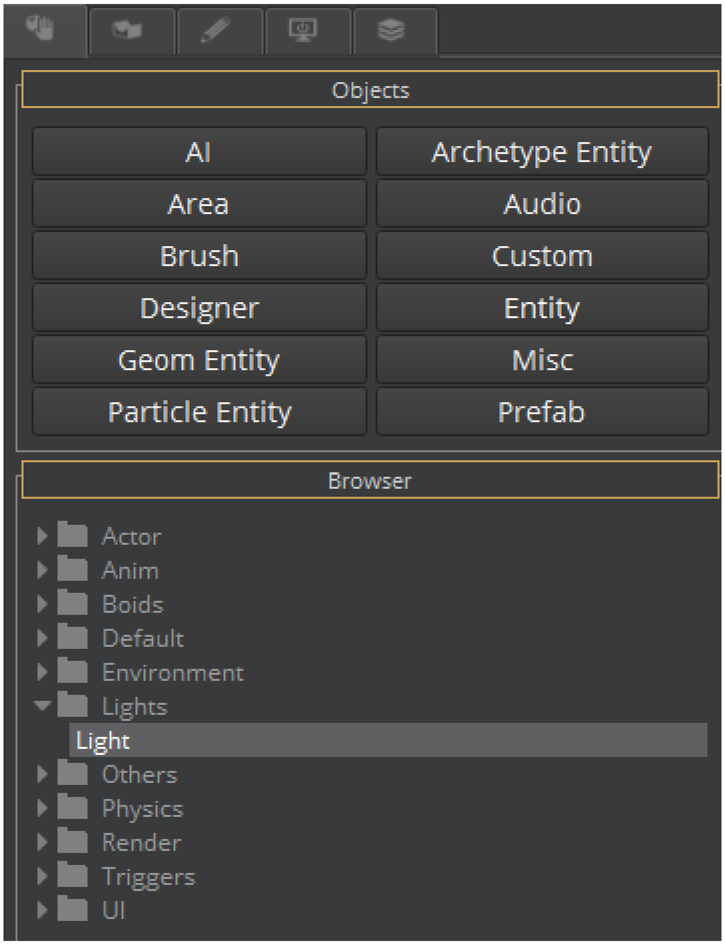
5. Close the **Time of Day** editor.

Adding Lights

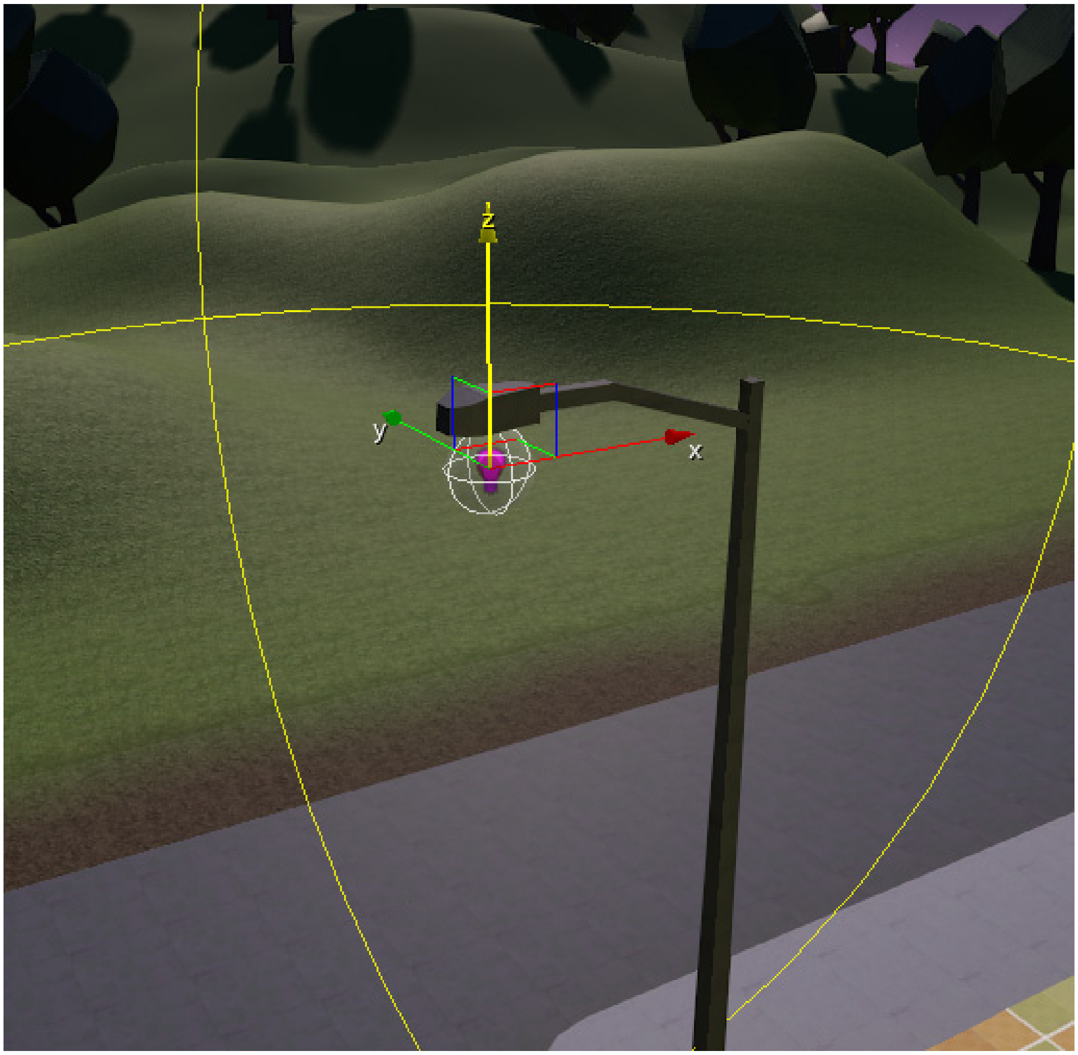
Now that your scene is set to night time, you can more clearly see the lights that you are about to place.

To add lights

1. In the **Rollup Bar**, click the **Objects** tab and then click **Entity**.
2. Under **Browser**, expand **Lights**. Select **Light** and drag it into the **Perspective** viewport.

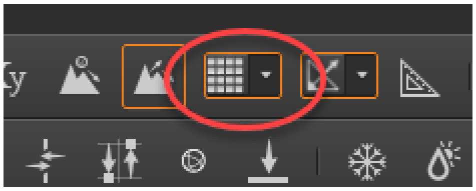


3. Position the light beneath one of the street lamps.

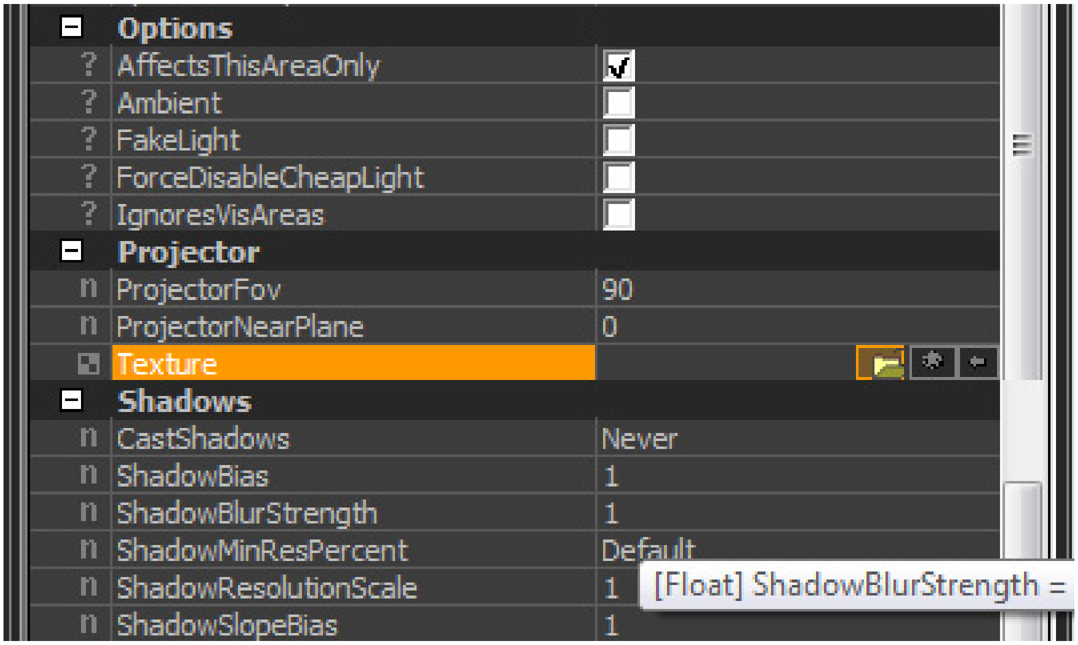


Tip

If you have trouble with precisely aligning the light, it is likely snapping to the grid. Turn off **Snap to Grid** in the editor toolbar, as shown in the following image, to move your light more precisely.



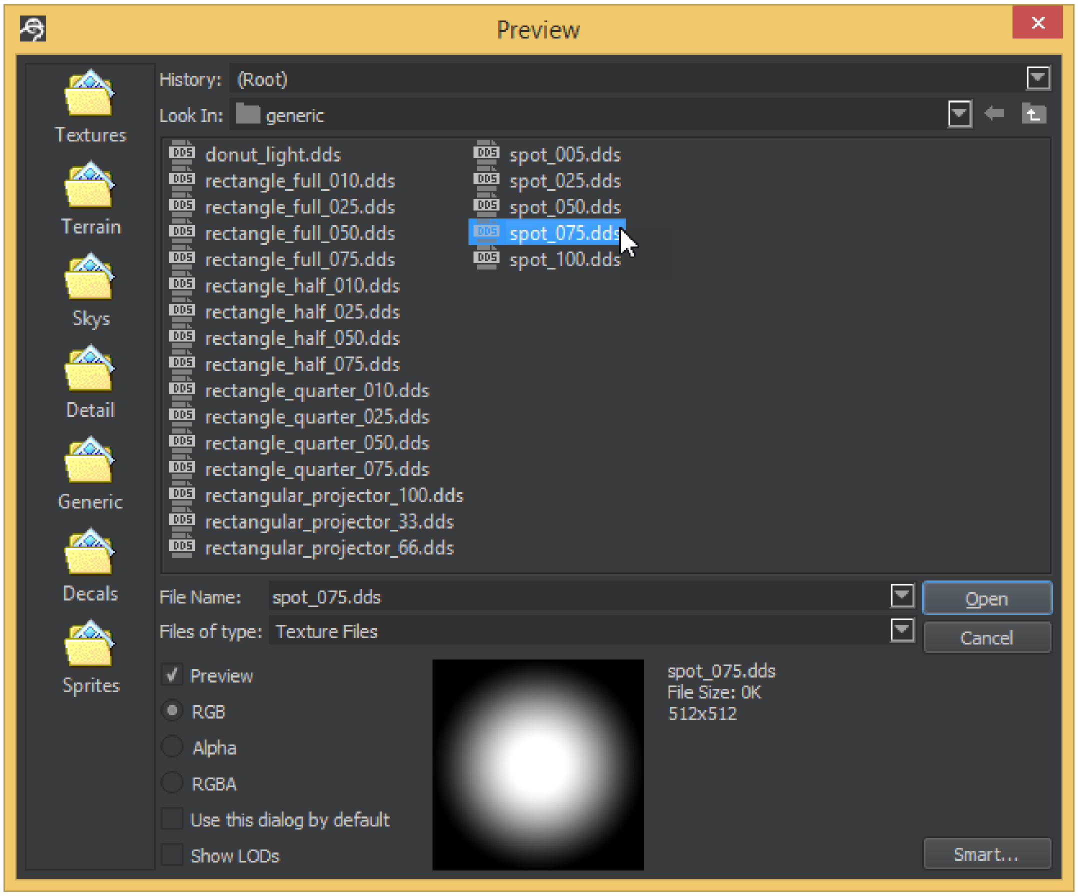
4. Because the light source is a street lamp, it should be a spot light (rather than a fill light). To change the light from fill to spot: Under the **Entity Properties** heading, find the **Projector** section. Under **Projector**, select **Texture**, and then click the folder icon.



Options	
AffectsThisAreaOnly	☑
Ambient	□
FakeLight	□
ForceDisableCheapLight	□
IgnoresVisAreas	□
Projector	
ProjectorFov	90
ProjectorNearPlane	0
Texture	
Shadows	
CastShadows	Never
ShadowBias	1
ShadowBlurStrength	1
ShadowMinResPercent	Default
ShadowResolutionScale	1
ShadowSlopeBias	1

[Float] ShadowBlurStrength =

5. Open the directory `\SamplesProject\textures\lights\generic`. Open `spot_075.dds`.



Preview

History: (Root)

Look In: generic

donut_light.dds
rectangle_full_010.dds
rectangle_full_025.dds
rectangle_full_050.dds
rectangle_full_075.dds
rectangle_half_010.dds
rectangle_half_025.dds
rectangle_half_050.dds
rectangle_half_075.dds
rectangle_quarter_010.dds
rectangle_quarter_025.dds
rectangle_quarter_050.dds
rectangle_quarter_075.dds
rectangular_projector_100.dds
rectangular_projector_33.dds
rectangular_projector_66.dds
spot_005.dds
spot_025.dds
spot_050.dds
spot_075.dds
spot_100.dds

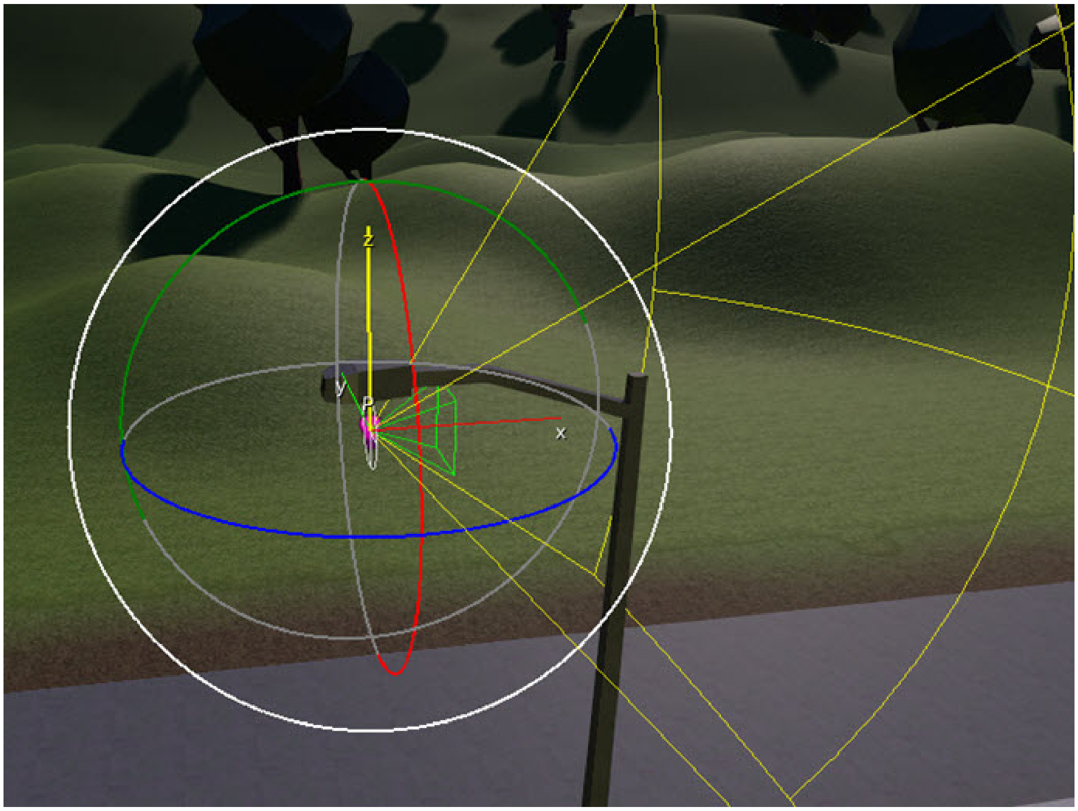
File Name: spot_075.dds

Files of type: Texture Files

☑ Preview
● RGB
○ Alpha
○ RGBA
□ Use this dialog by default
□ Show LODs

spot_075.dds
File Size: 0K
512x512

6. The light changes to a spotlight and is oriented sideways. Use the rotate tool to select and rotate the light so that it points down.



7. In the attributes table under **Entity Properties**, you can modify a variety of settings to customize the light. Several of these settings affect the light. For this tutorial, use the following settings:

   AttenuationBulbSize = `6`
   The size of the light bulb. This is the starting point where light begins to fall off exponentially. A value of 1 sets the light at full intensity for one yard before it begins to fall off. Adjust this size in relation to the **DiffuseMultiplier** to manage the brightness of the light source without using unmanageable numbers.

   Radius = `20`
   Surrounding area that the light affects as measured by the distance from the source.

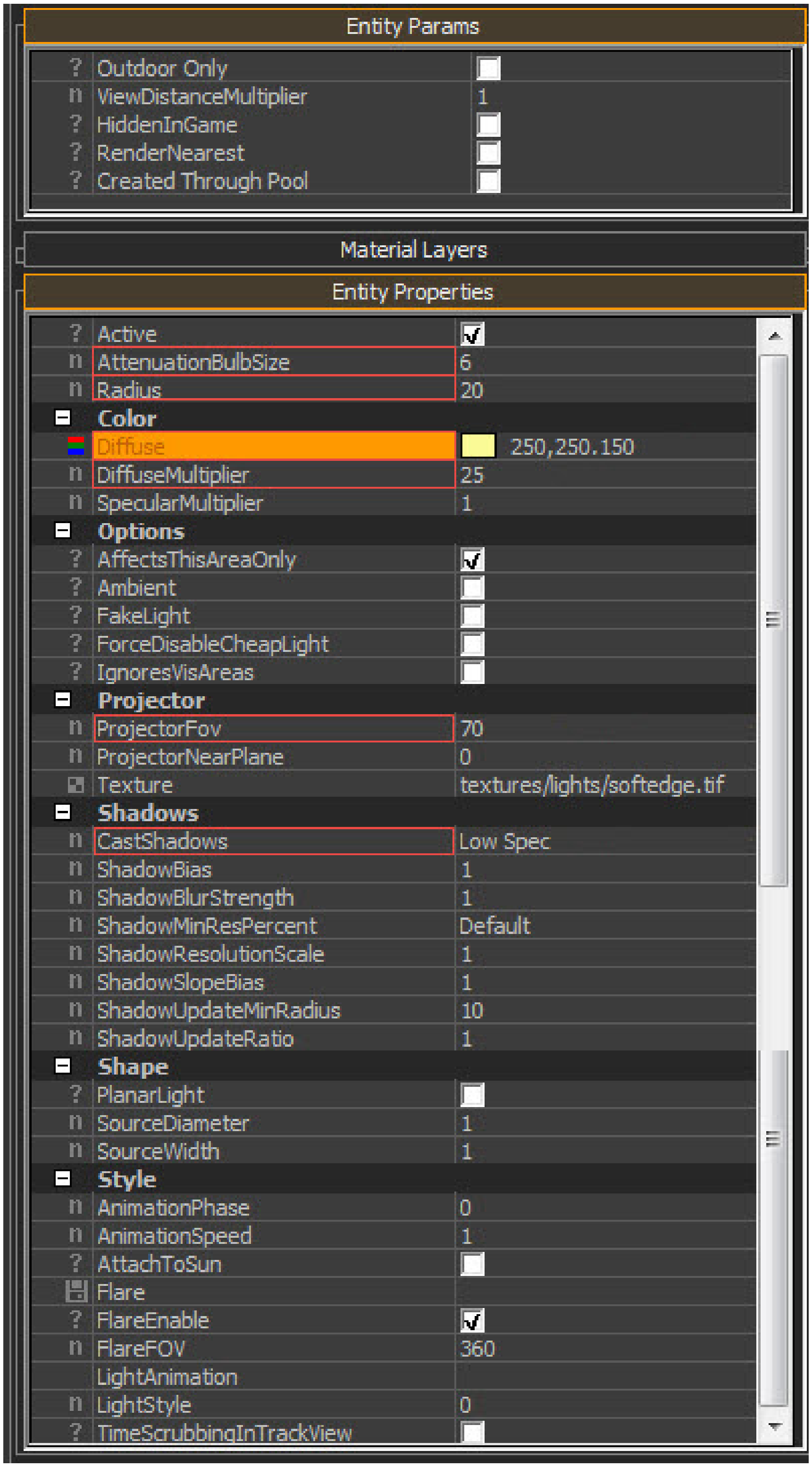
   Diffuse = `250,250,150`
   Color (RGB value) of the light.

   DiffuseMultiplier = `25`
   Intensity of the diffuse color. Balance this value with the **AttenuationBulbSize** to define the balance of natural light levels.

   ProjectorFov = `70`
   Field of view for the projection light.

   CastShadows = **Lowspec**
   Setting that makes the light cast a shadow based on the minimum selected configuration specification (config spec; predefined quality setting). If you use a setting of **High Spec**, for example, the shadow won't appear when running the game on a machine with a config spec of **Low Spec** or **Medium Spec**. To ensure shadows are always cast, set this to **Low Spec**. Note that the **CastShadows** setting does not affect visual quality; use it to adjust performance based on the type of machine running the level.

   Experiment with these settings to see how the bulb size, radius, and diffuse multiplier change based on the input values.



Entity Params

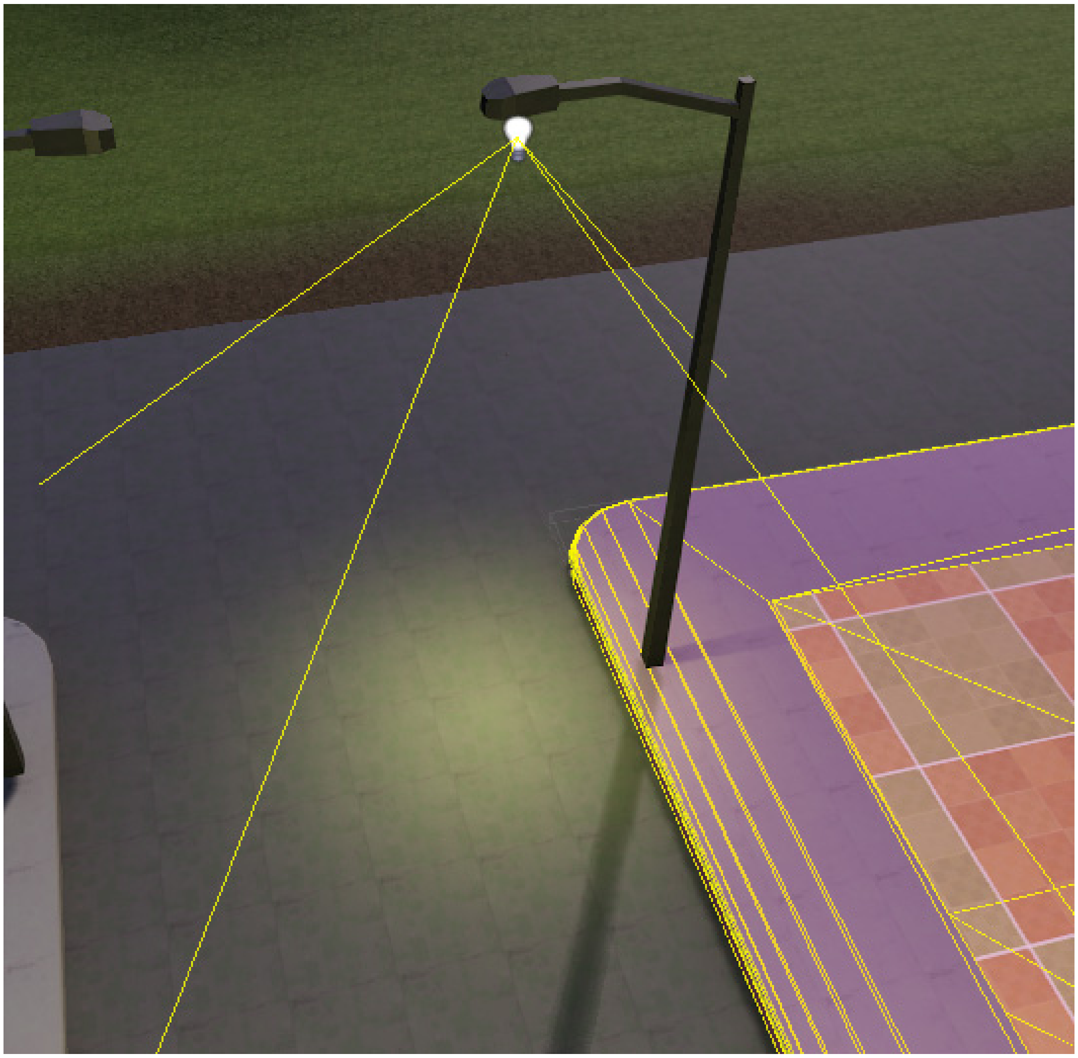
Outdoor Only	□
ViewDistanceMultiplier	1
HiddenInGame	□
RenderNearest	□
Created Through Pool	□

Material Layers

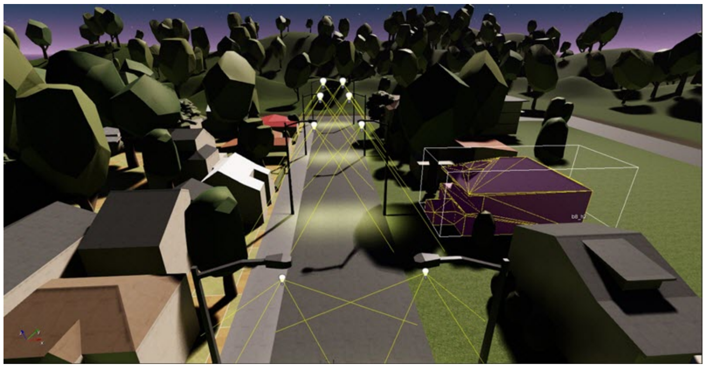
Entity Properties

Active	☑
AttenuationBulbSize	6
Radius	20
Color	
Diffuse	250,250.150
DiffuseMultiplier	25
SpecularMultiplier	1
Options	
AffectsThisAreaOnly	☑
Ambient	□
FakeLight	□
ForceDisableCheapLight	□
IgnoresVisAreas	□
Projector	
ProjectorFov	70
ProjectorNearPlane	0
Texture	textures/lights/softedge.tif
Shadows	
CastShadows	Low Spec
ShadowBias	1
ShadowBlurStrength	1
ShadowMinResPercent	Default
ShadowResolutionScale	1
ShadowSlopeBias	1
ShadowUpdateMinRadius	10
ShadowUpdateRatio	1
Shape	
PlanarLight	□
SourceDiameter	1
SourceWidth	1
Style	
AnimationPhase	0
AnimationSpeed	1
AttachToSun	□
Flare	
FlareEnable	☑
FlareFOV	360
LightAnimation	
LightStyle	0
TimeScrubbingInTrackView	□

Your light should look something like this:



8. With the light selected, press **Ctrl+D** and drag your pointer to copy the light. Position the light under the next street light. Repeat this for the rest of the street lamps in the scene.

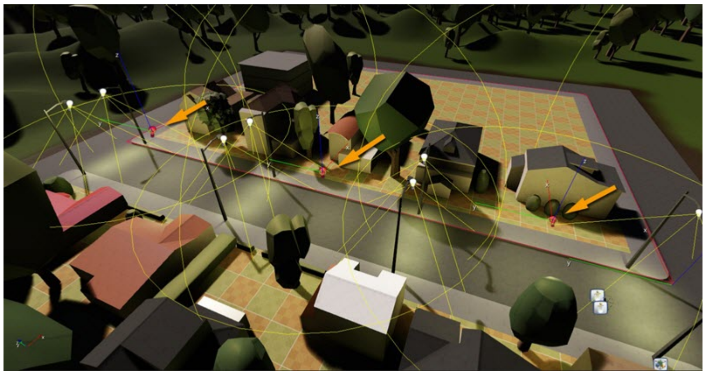


9. You can place additional fill lights to help light the scene, which is still somewhat dark. To do this, repeat the previous steps and place lights between the street lamps, as in the following image.

   For these lights, use the following settings:

   - **AttenuationBulbSize** = **5**

- **Radius** = **`25`**
- **Diffuse Color** = **`250,250,150`**
- **Diffuse Multiplier** = **`8`**
- **CastShadows** = **Never**



10. Experiment with these settings and observe how the lights affect the scene. Find a setting you like, or keep the suggested values above.

    Tip

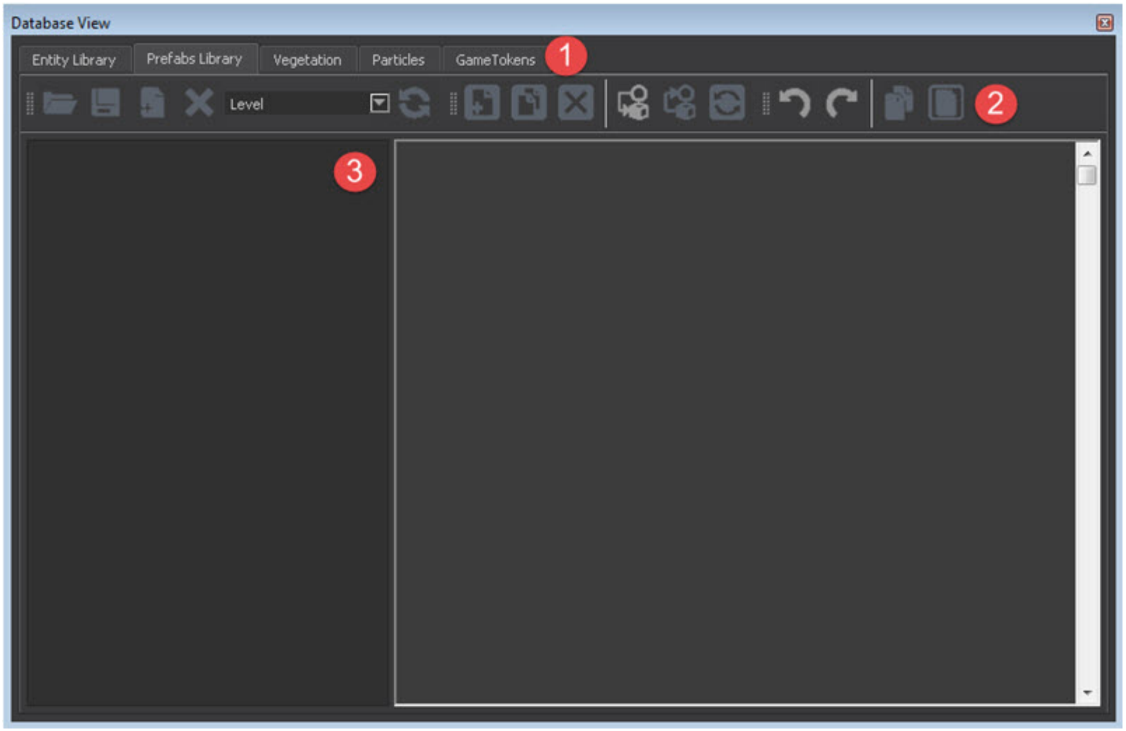
    Try using the **Hide** option (**H** icon on the right side of the **Perspective** viewport title bar) to hide the entity icons. This allows you to view the scene without the visual clutter of entity icons.
11. Save your level file.

Placing a Game Camera and Character

To enable game play in a level, you will create and place a game play camera that is attached to a robot character that you control. The location you place your robot character also specifies the starting point of your level. To save you time and get you to prototyping levels sooner, Lumberyard has created a game play camera and robot character (with supporting input controls) to use as a quick start to building a game play level.

The **Database View** editor for prefabs has the following areas:

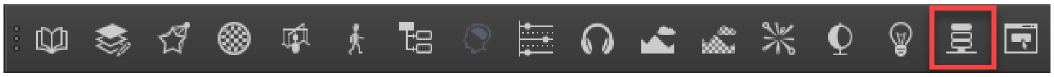
1. **Database tabs** – Database types to view and manage
2. **Editor toolbar** – Tools to open, save, add, remove Prefabs
3. **File tree view** – View of the Prefabs available for use



To place the game camera and robot character in the level

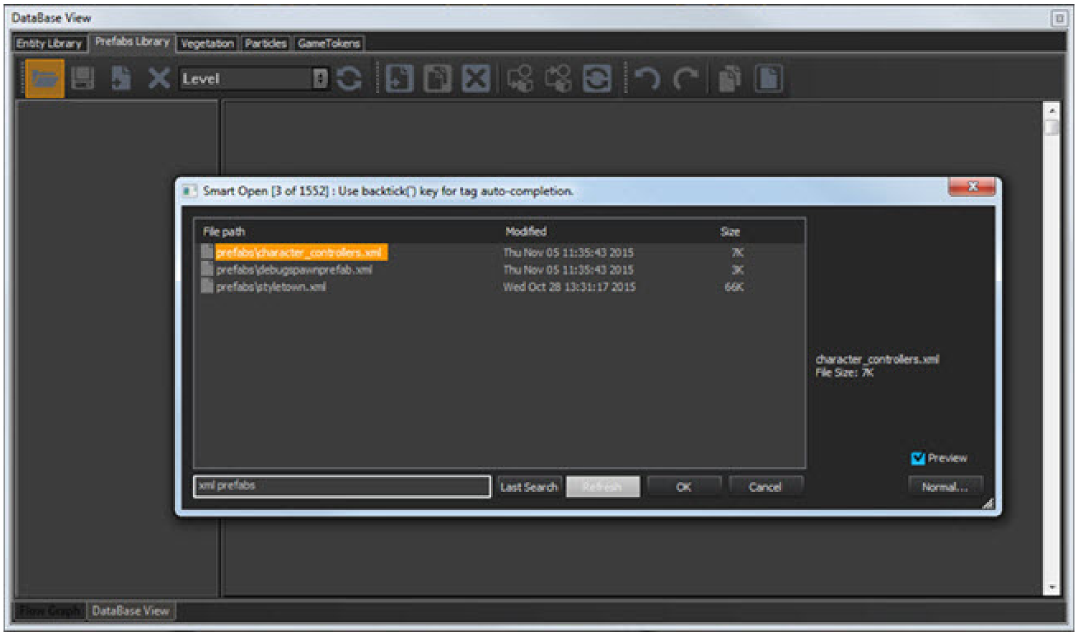
1. Do one of the following to open the **Database View** editor:

   - On the editor toolbar, click **Database View**.



   - From the main menu, click **Tools**, **Other**, **Database View**.

2. In the **Database View** editor, click the **Prefabs Library** tab.
3. Click **Load Library** .
4. In the prefabs directory listing, select `prefabs\character_controllers.xml`.



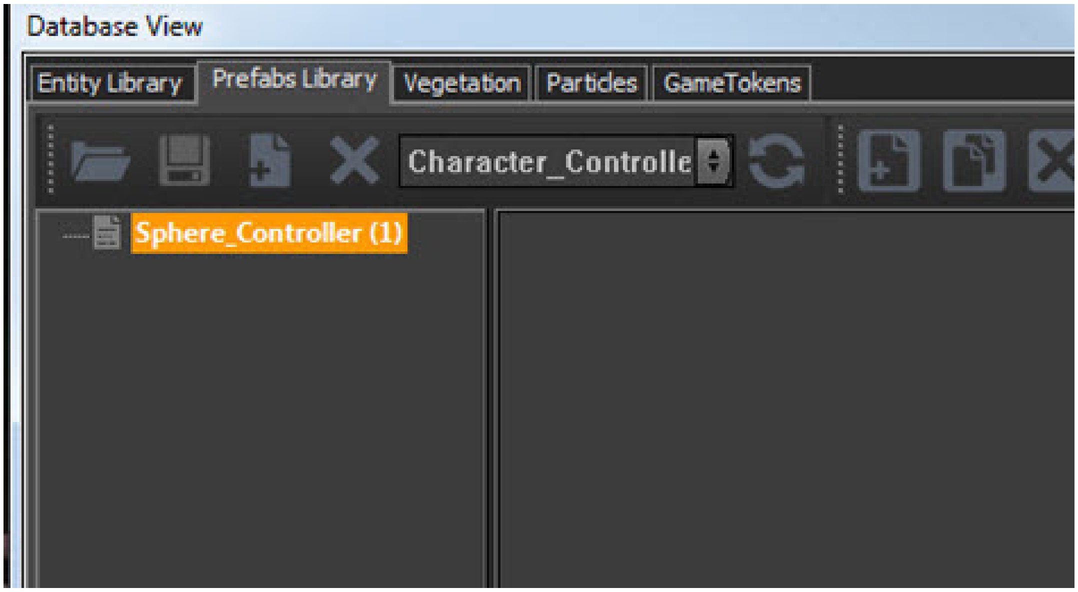
Smart Open [3 of 1552] : Use backtick(`) key for tag auto-completion.

File path	Modified	Size
prefabs\character_controllers.xml	Thu Nov 05 11:35:43 2015	7K
prefabs\debugspawnprefab.xml	Thu Nov 05 11:35:43 2015	3K
prefabs\styletown.xml	Wed Oct 28 13:31:17 2015	66K

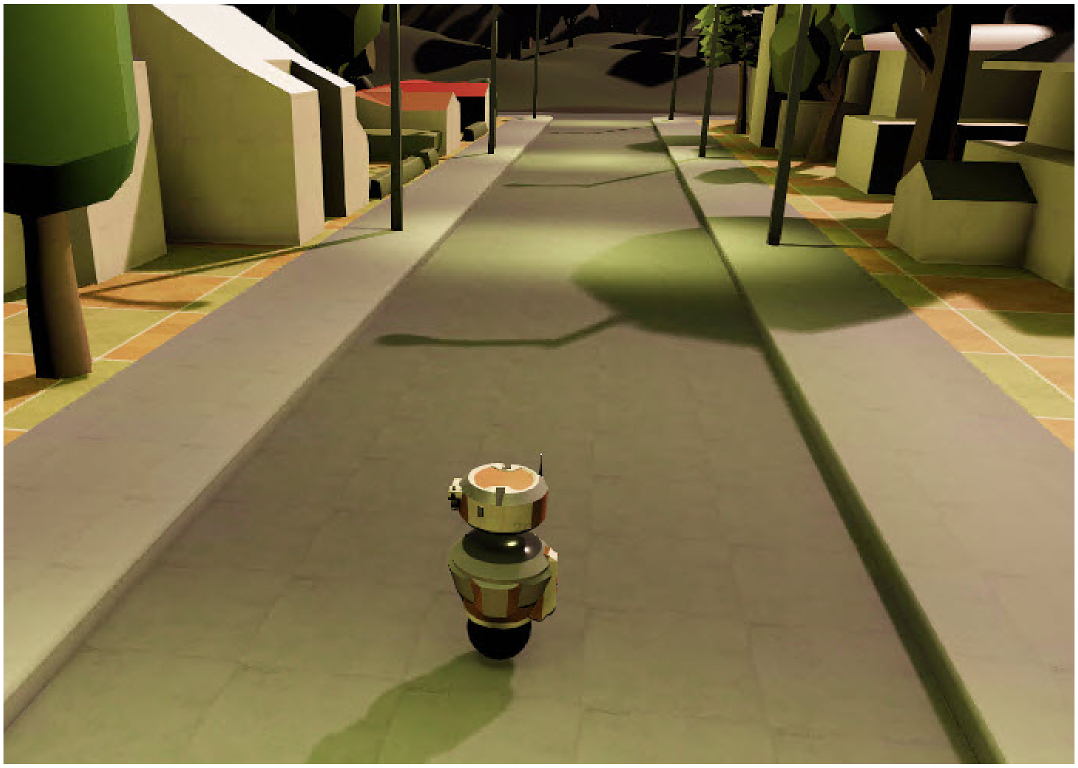
character_controllers.xml
File Size: 7K

xml prefabs

5. From the **Prefab Library** file tree view, drag **Sphere_Controller(1)** into the **Perspective** viewport.



This provides the level with a third-person camera as well as the robot character and input control. The location where you place it also determines the start point of the level.



6. Save your level file.

Switching to Game Mode

Now that you have placed the camera in the level, you can switch to game mode and move around the level in third-person mode.

To switch to game mode from edit mode

Do one of the following:

- In the main menu, choose **Game**, **Play Game**.
- On your keyboard, press **Ctrl+G**.

You are now running your first Lumberyard level.

To navigate, use the following controls:

- **Move forward or back** – Press **`W`** and **`S`** keys
- **Strafe left or right** – Press **`A`** and **`D`** keys
- **Turn left or right** – Move pointer left or right
- **Jump** – Press **space bar**
- **Camera Look** – Move pointer

To exit game mode and return to edit mode

Press **Esc** on your keyboard.

Creating Designer Objects

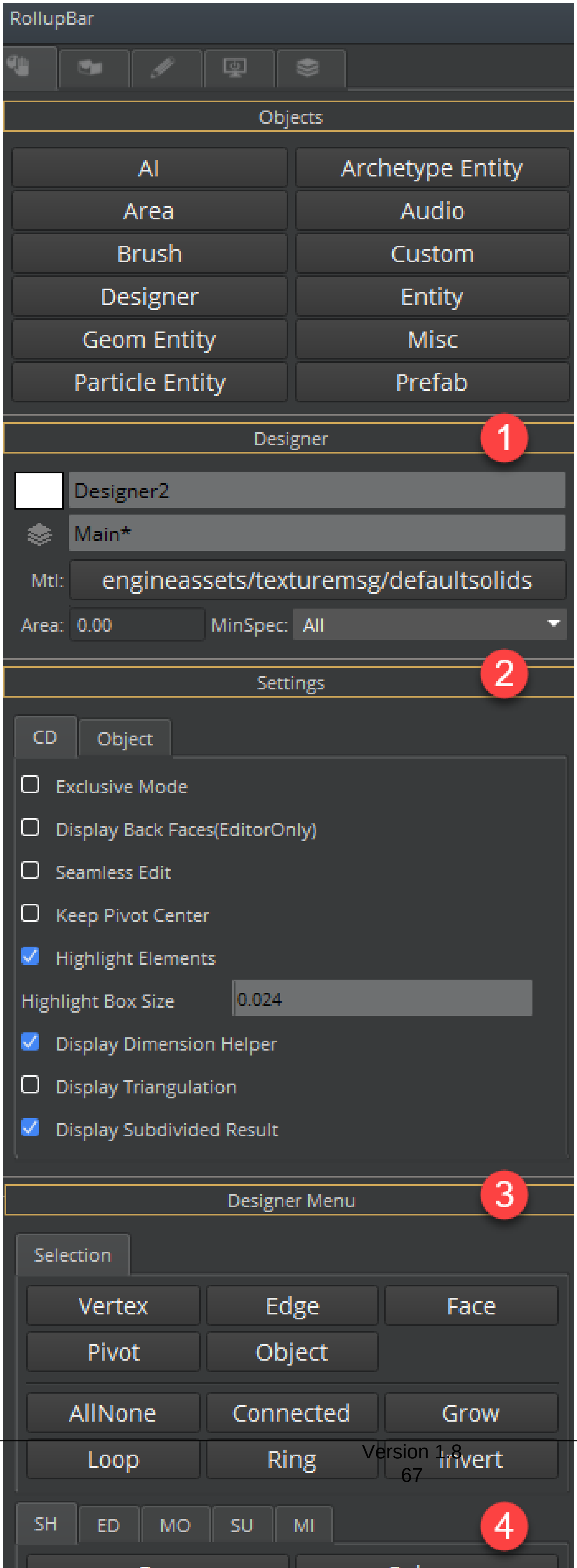
With the **Designer** tool, you can create, edit, and export meshes. You can also use it to project texture coordinates onto a 3D surface, a process known as UV mapping.

A mesh is a collection of vertices, edges, and faces that describe the shape of a 3D object:

- A vertex is a single point.
- An edge is a straight line segment connecting two vertices.
- A face is a flat surface enclosed by edges.

The Designer tool has the following areas:

1. Standard parameters such as name and layer location.
2. Settings for toggling various object creation conditions. You can typically use the default settings.
3. Selection options (vertex, edge, or face) as well as secondary selection methods.
4. Controls for selecting and editing shapes, modifying objects, managing surface material, and exporting objects.
5. Controls for specifying the dimensions and subdivision of faces.



Designer2

Main*

Mtl: engineassets/texturemsg/defaultsolids

Area: 0.00 MinSpec: All

- □ Exclusive Mode
- □ Display Back Faces(EditorOnly)
- □ Seamless Edit
- □ Keep Pivot Center
- ☑ Highlight Elements

Highlight Box Size 0.024

- ☑ Display Dimension Helper
- □ Display Triangulation
- ☑ Display Subdivided Result

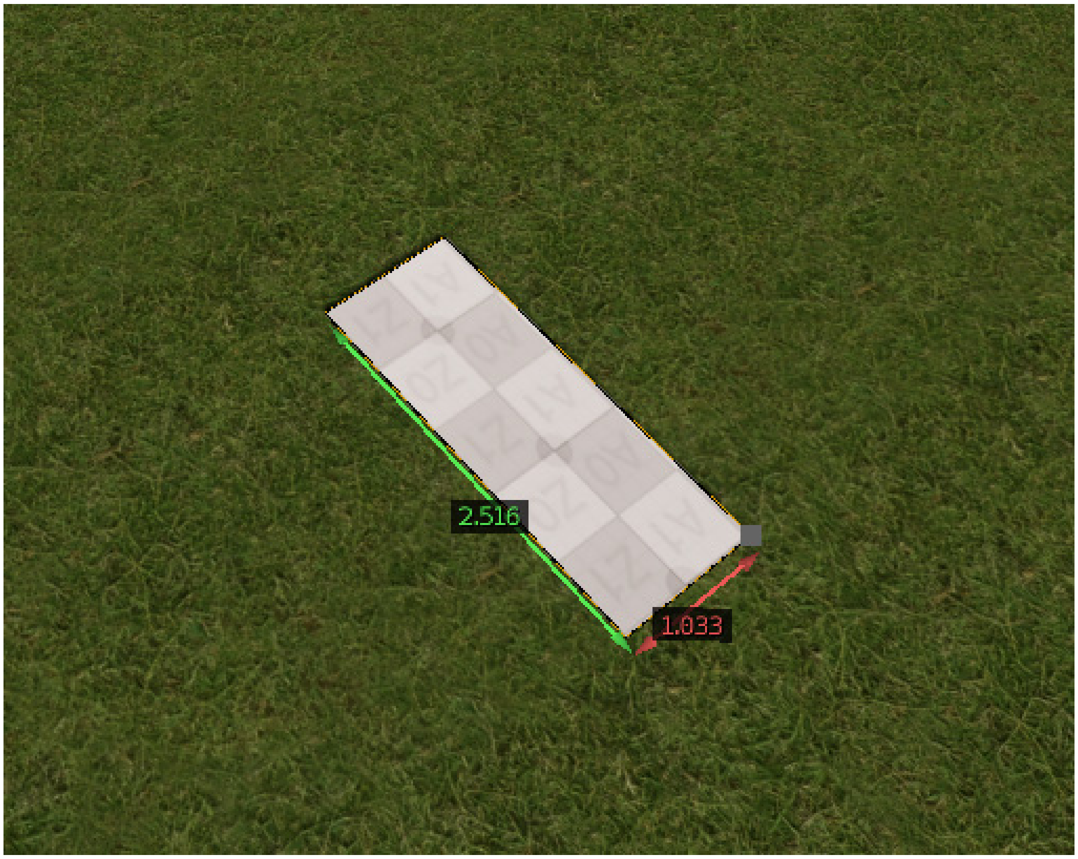
Objects you create with the **Designer** tool are a type of mesh object within the editor. Though similar to brushes and entities, which are also mesh objects, designer objects have their own unique category. Designer objects do not have the same capabilities as entities, but they work just as well as brush objects when building a level. You can block out entire levels with the Designer tool; this is an effective method for blocking out game play environments.

For this tutorial, you'll create two designer objects: A box and a sphere.

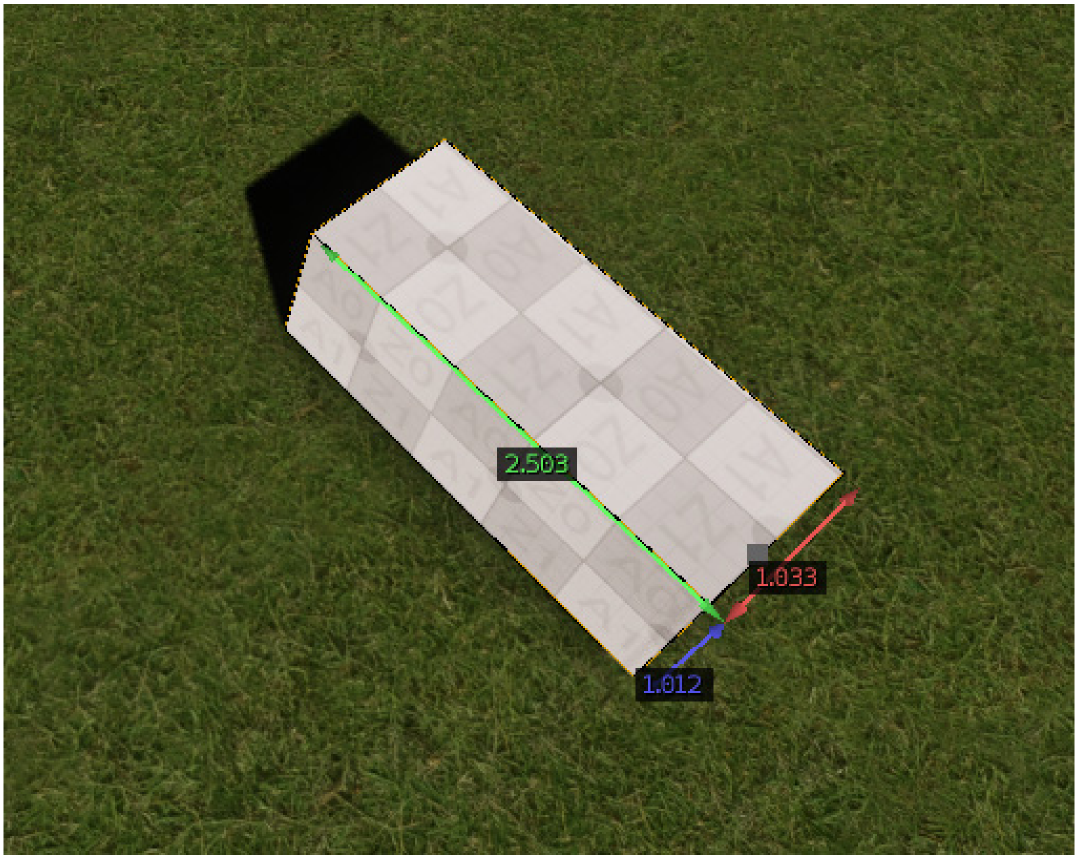
To create designer objects

1. In the **Rollup Bar**, click the **Objects** tab. Click **Designer**.
2. Turn off **Snap to Grid** in the editor toolbar, or press **G** on your keyboard.
3. Under **Designer Menu** (in the section with the tabs **SH**, **ED**, **MO**, **SU**, **MI**), click **Box**.
4. In the **Perspective** viewport, drag to create a rectangle. When you release the mouse button, the length and width are locked in, and the height control appears. Create a box that is about 2.5 units long by 1 unit wide. The size doesn't need to be exact, but try to come close to that size range for this tutorial.

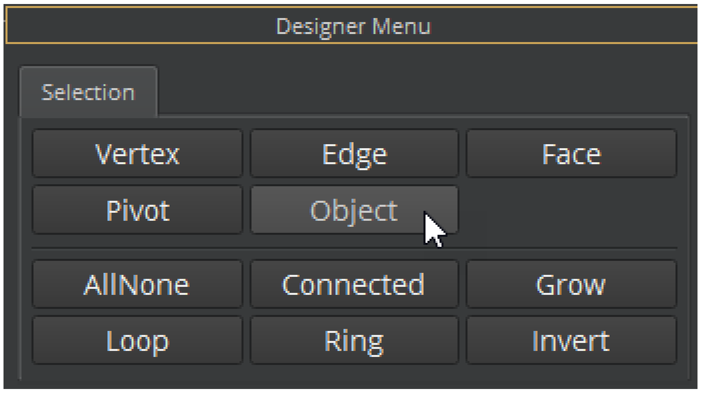
   Tip
   It is easier to create the box if you zoom in close to the ground.

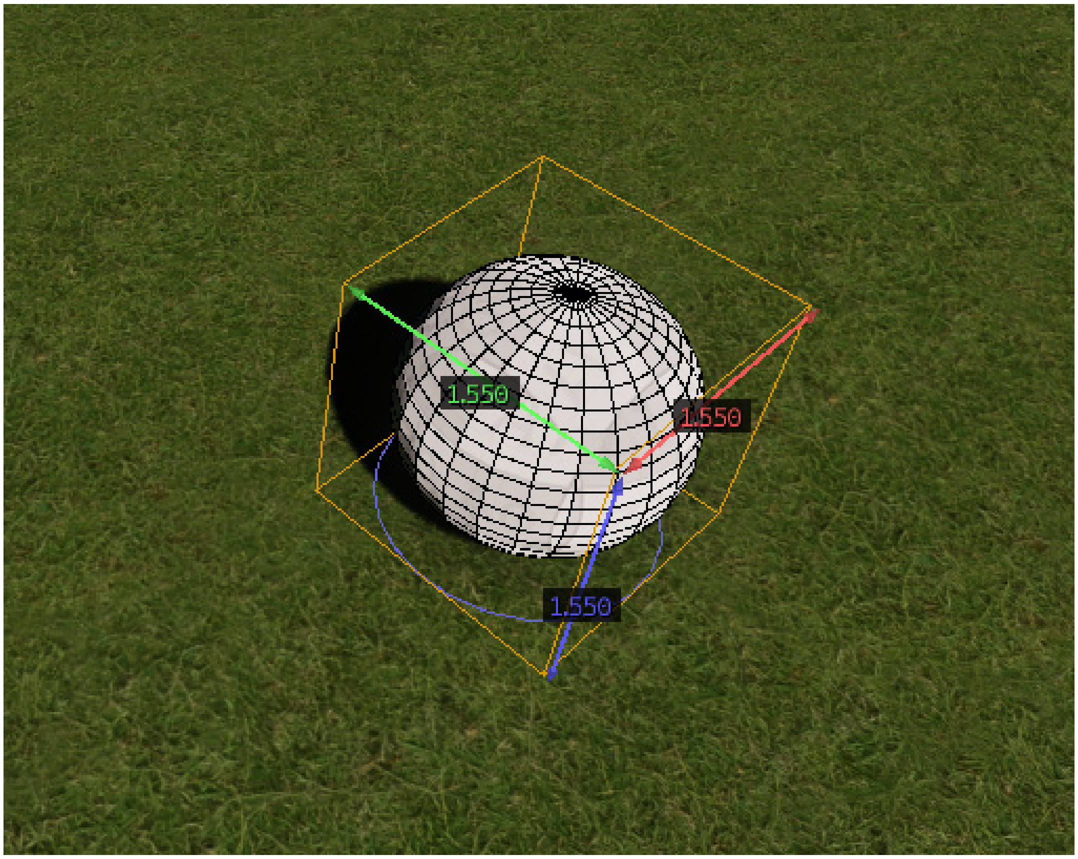
5. Move your pointer up to define the height at about 1 unit high. Click the left mouse button to lock the height.



6. Under **Designer Menu**, in the **Selection** area, click **Object**. This exits the edit mode on the box you just created.



7. Next, create a sphere. Under **Designer Menu**, click **Sphere**.
8. In the **Perspective** viewport, drag the mouse to create a sphere that is about 1.5 units in diameter.



Tip
As with the box, zoom in close to the ground to make it easier to create a sphere with these dimensions.

9. Now exit creation mode: Under **Designer Menu**, in the **Selection** area, click **Object**.

   Now that you have these two objects created, you'll apply textures to them and then export them.

Using Materials

With Lumberyard, you can create materials for your level and apply those materials to a designer object. You can also export that object. The tutorials in this section show you how.

Topics

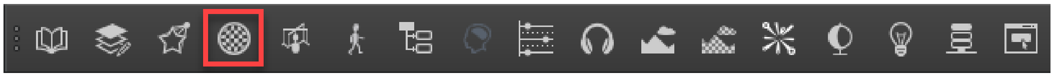
Creating a New Material

To create a new material, you use the Material Editor (p. 35). The **Material Editor**'s features are described in Painting the Terrain (p. 32).

To create a new material

1. Do one of the following to open the **Material Editor**:

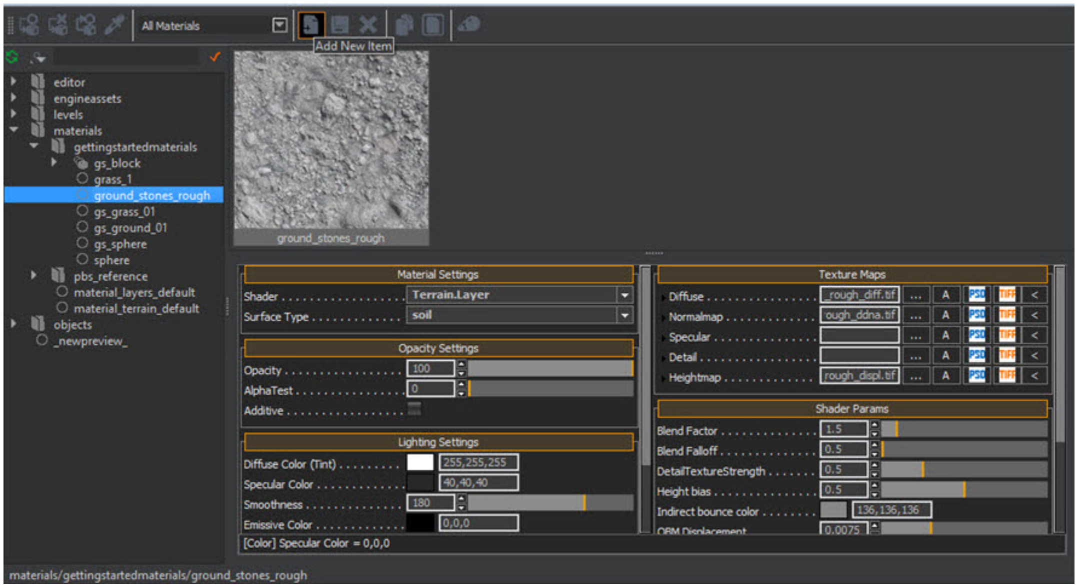
   - On the editor toolbar, click the **Material Editor** icon

   - From the main menu, open **Tools**, **Material Editor**

2. Expand `materials\GettingStartedMaterials`.

3. Select `gettingstartedmaterials`. In the toolbar, click **Add New Item**.

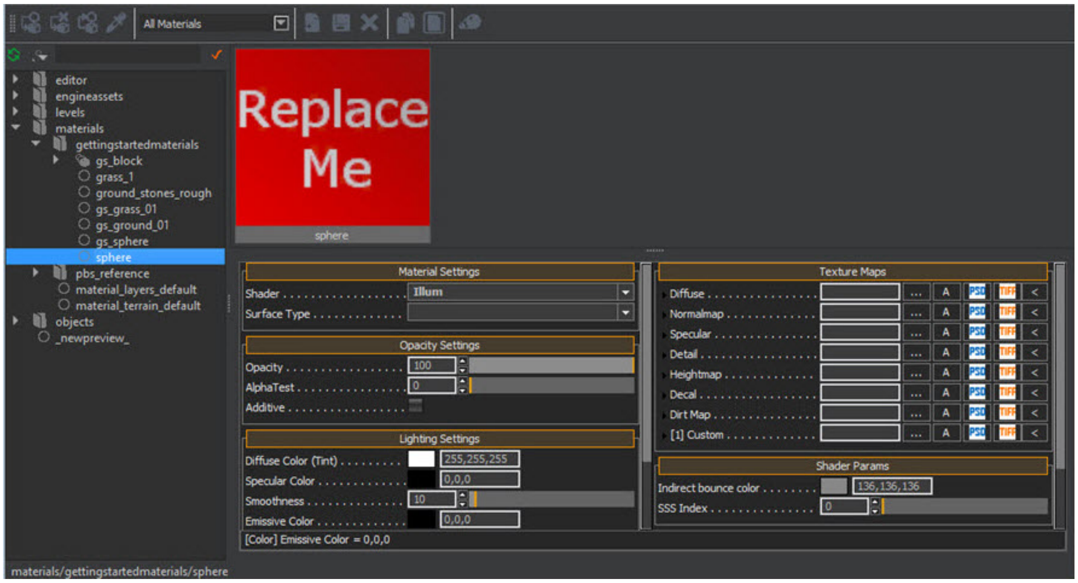
   

   Shader . . . . Terrain.Layer
   Surface Type . . . . soil
   Opacity 100
   AlphaTest 0
   Additive . . . .
   Diffuse Color (Tint) 255,255,255
   Specular Color 40,40,40
   Smoothness 180
   Emissive Color 0,0,0
   [Color] Specular Color = 0,0,0
   Diffuse . . . . _rough_diff.tif
   Normalmap . . . . ough_ddna.tif
   Specular . . . .
   Detail . . . .
   Heightmap . . . . rough_displ.tif
   Blend Factor 1.5
   Blend Falloff 0.5
   DetailTextureStrength 0.5
   Height bias 0.5
   Indirect bounce color 136,136,136
   OBM Displacement 0.0075
   materials/gettingstartedmaterials/ground_stones_rough

4. Navigate to `Materials\GettingStartedMaterials`. For **File name**, type **`sphere`**. Click **Save**.

5. Click **Add New Item** again. For **File name**, type **`block`**. Click **Save**.

6. In the **Material Editor**, select the sphere material you just created. Your **Material Editor** should look similar to the following:



Material Settings

Shader . . . . Illum

Surface Type . . . .

Opacity Settings

Opacity 100

AlphaTest 0

Additive . . . .

Lighting Settings

Diffuse Color (Tint) 255,255,255

Specular Color 0,0,0

Smoothness 10

Emissive Color 0,0,0

[Color] Emissive Color = 0,0,0

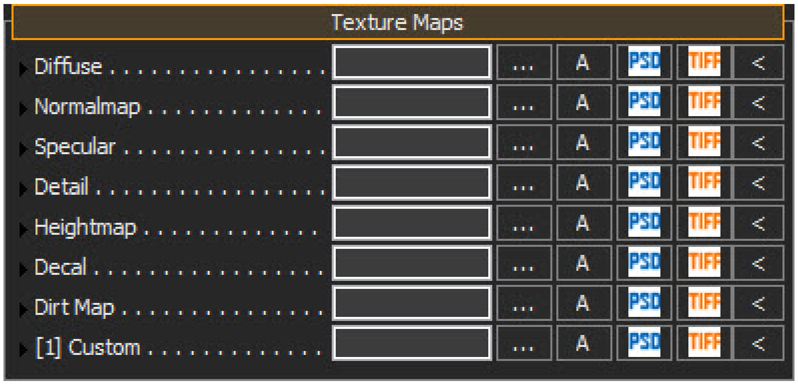
Shader Params

Indirect bounce color 136,136,136

SSS Index 0

materials/gettingstartedmaterials/sphere

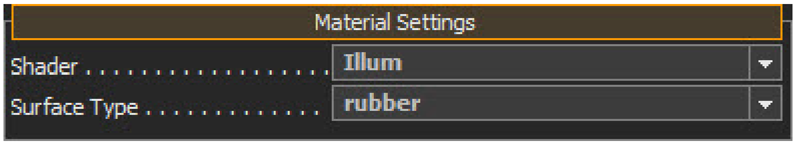
7. Under **Texture Maps** (lower right area), find the **Diffuse** line and click the ellipsis (**...**).


Texture Maps
Diffuse . . . .
Normalmap . . . .
Specular . . . .
Detail . . . .
Heightmap . . . .
Decal . . . .
Dirt Map . . . .
[1] Custom . . . .
...
A
PSD
TIFF
<

8. Navigate to `SamplesProject/textures/GettingStartedTextures`. Open `white.dds`.

9. Under **Materials Settings**, for **Surface Type**, select **rubber** from the list.

   This gives the material object the properties of rubber; objects mapped with rubber bounce when hitting another surface.

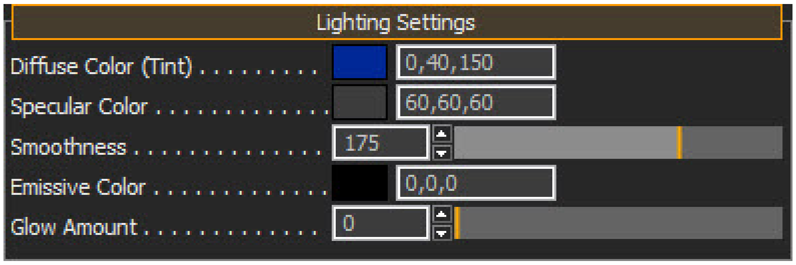


Material Settings

Shader . . . . Illum

Surface Type . . . . rubber

10. Under **Lighting Settings**, set **Diffuse Color** (Tint) to **`0,40,155`**.
11. Set **Specular Color** to **`60,60,60`**.
12. Set **Smoothness** to **`175`**.



Lighting Settings

Diffuse Color (Tint) 0,40,150

Specular Color 60,60,60

Smoothness 175

Emissive Color 0,0,0

Glow Amount 0

13. Save your material setting. To do this, in the toolbar, click the **Save item** icon.
14. Experiment with these various settings to see how adjusting **Diffuse color**, **Specular color**, and **Smoothness** quickly changes your material's appearance.

    If you are continuing to the next section, leave the Material Editor open.

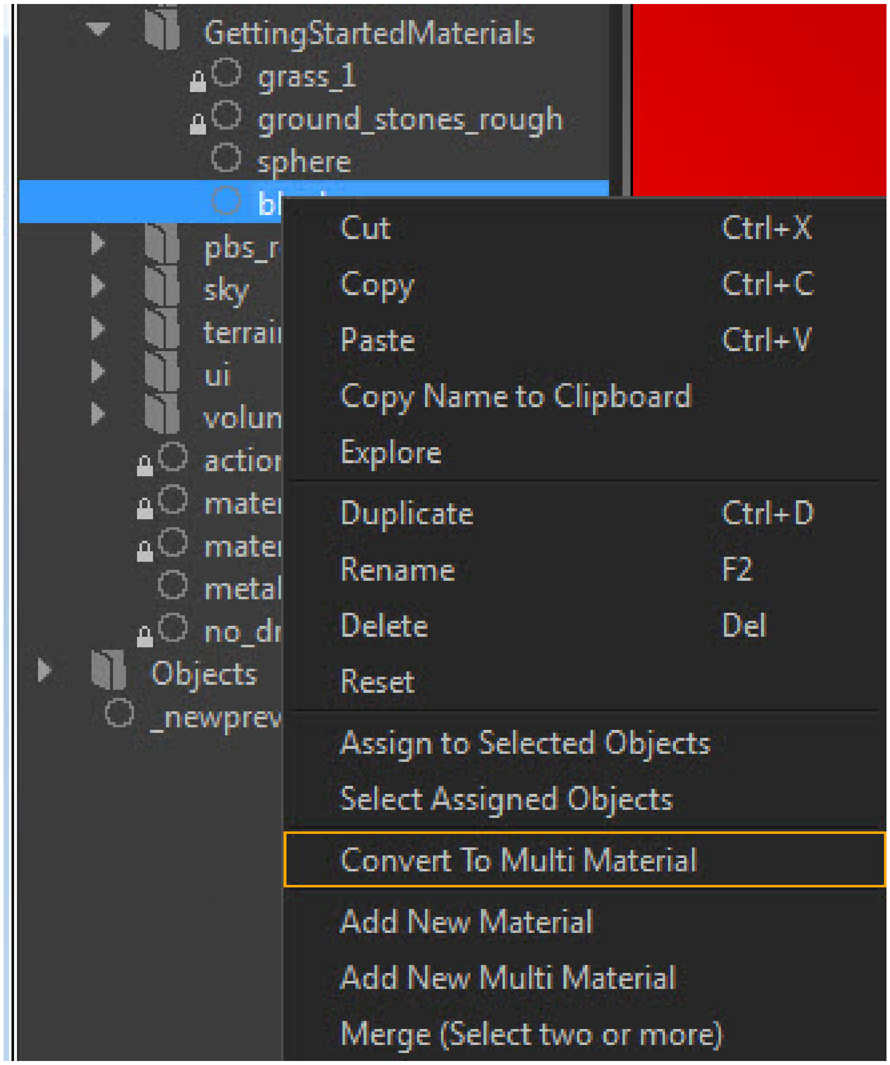
Creating Multimaterial

A multimaterial is a single material file that contains multiple materials. With multimaterial you can map multiple materials onto a single object (for example, a block). Multimaterials are useful for prototyping the internal structure of a scene. For example, if you need to build a neighborhood full of houses, you can build a box with a peaked roof and then cut faces on the side for windows and doors. With a multimaterial, you can create the siding, roof, window, and door textures and then assign the correct material ID to each component to represent a house.

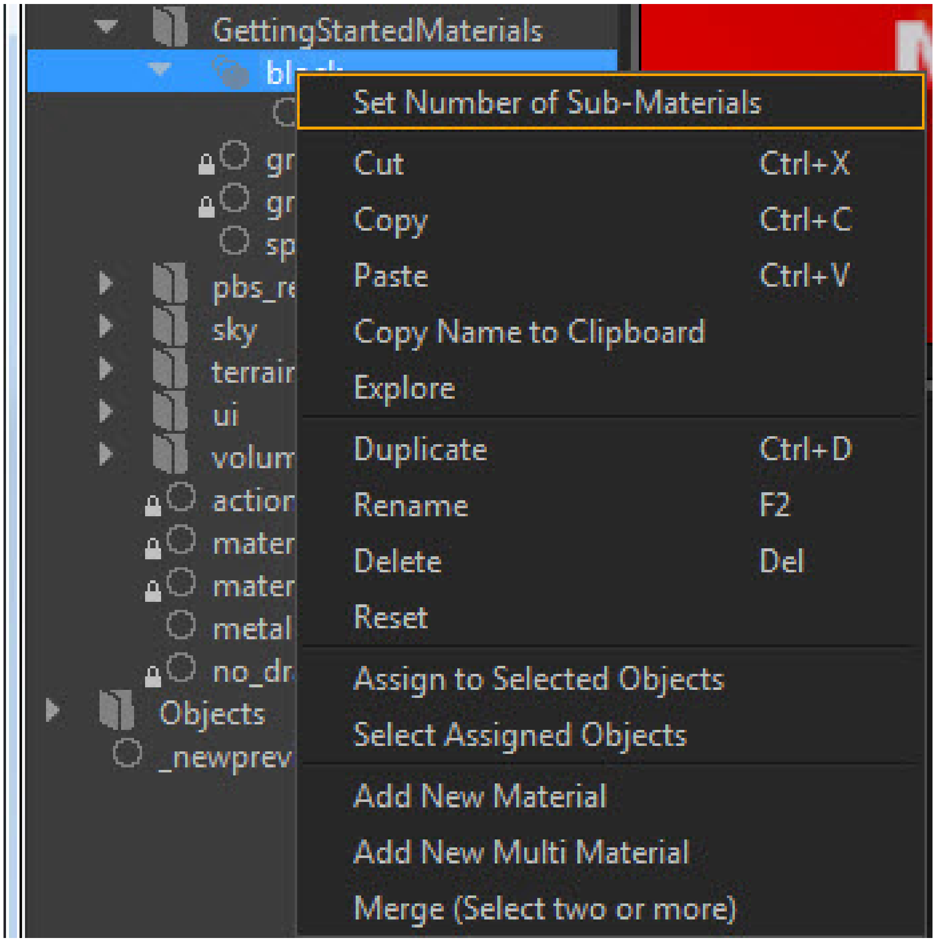
In this tutorial, you'll turn the previously created block material into a multimaterial.

To create a multimaterial

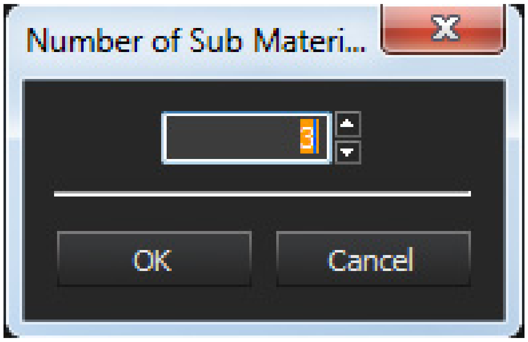
1. Open the **Material Editor** if it is not already open.
2. Right-click the name of the **block** material that you previously created.
3. Click **Convert to Multi Material**.



4. Select **block** (not the newly-created **[1] block**). Right-click the word **block** and select **Set Number of Sub-Materials**.

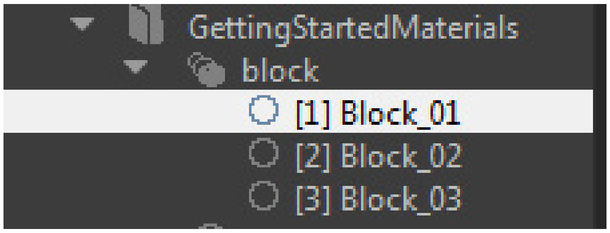


5. When prompted for the number of submaterials, type **3**. Click **OK**.



Three new submaterials appear under **block**.

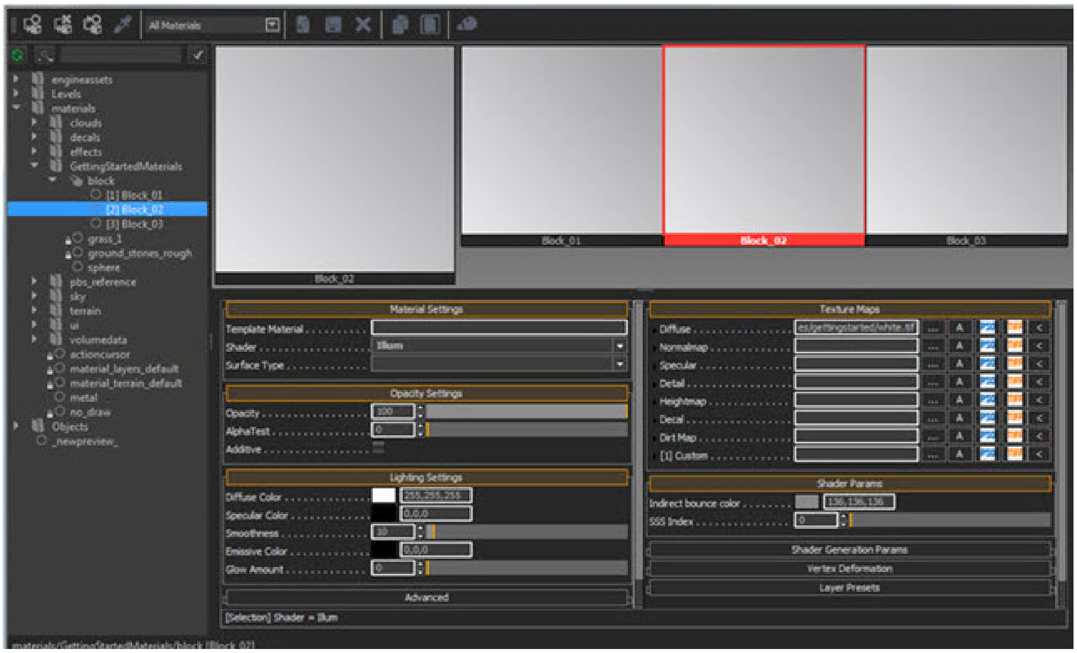
6. Right-click each submaterial and rename them **`block_01`**, **`block_02`**, and **`block_03`**.



7. Hold down **Ctrl** and select **block_01**, **block_02**, and **block_03**. On the lower right, under **Texture Maps**, find **Diffuse** and click the ellipsis (**...**).
8. Navigate to `SamplesProject\textures\GettingStartedTextures`. Open `white.dds`.

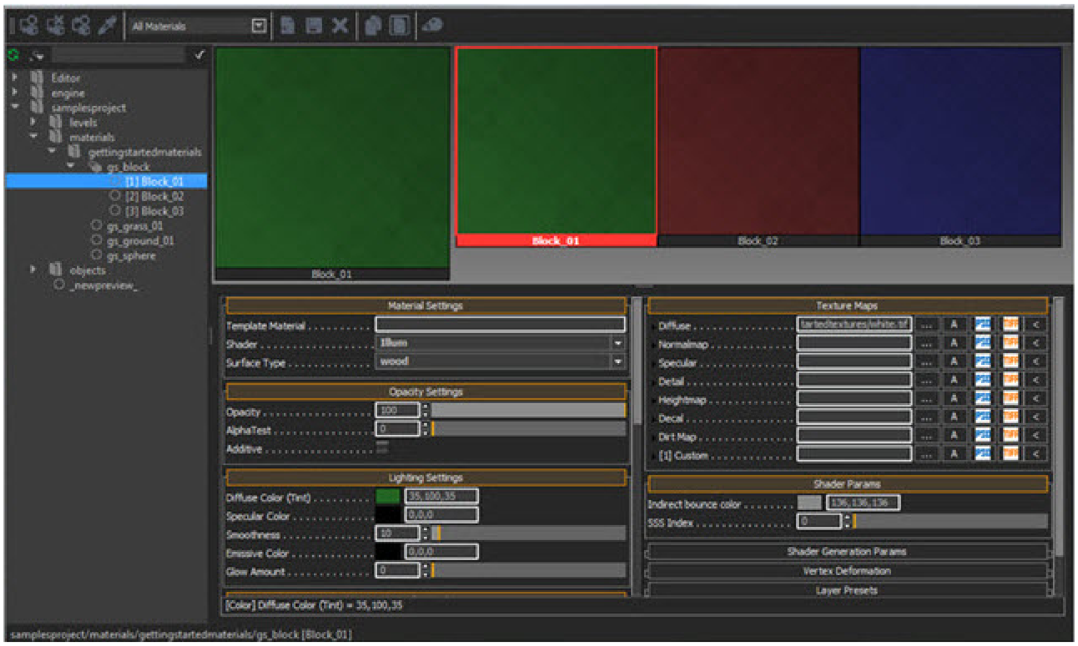
All submaterials now have a diffuse texture map assigned to them.

The material preview window shows the three submaterials on the right; the selected submaterial appears slightly larger on the left.



Template Material
Shader: Illum
Surface Type
Opacity: 100
AlphaTest: 0
Additive
Diffuse Color: 255,255,255
Specular Color: 0,0,0
Smoothness: 10
Emissive Color: 0,0,0
Glow Amount: 0
Diffuse: es/gettingstarted/white.tif
Normalmap
Specular
Detail
Heightmap
Decal
Dirt Map
[1] Custom
Indirect bounce color: 136,136,136
SSS Index: 0
[Selection] Shader = Illum
materials/GettingStartedMaterials/block [Block_02]

9. Select submaterial **block_01** and, under **Lighting Settings**, set the **Diffuse** color to **35,100,35**.
10. Select submaterial **block_02** and set **Diffuse** color to **100,35,35**.
11. Select submaterial **block_03** and set **Diffuse** color to **35,35,100**.
12. Under **Material Settings**, set **Surface Type** to wood for all three submaterials.



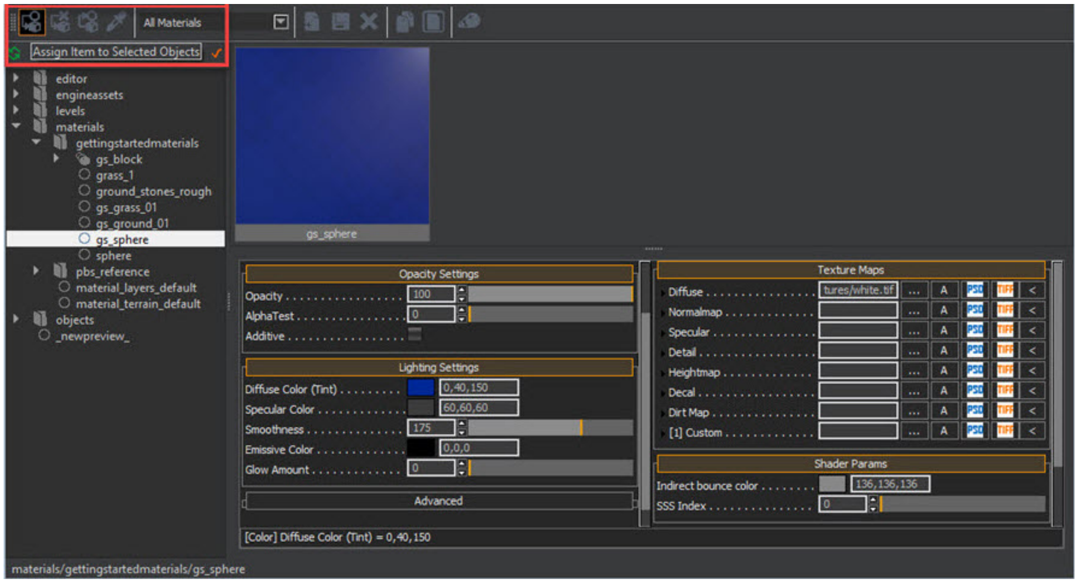
Template Material
Shader: Illum
Surface Type: wood
Opacity: 100
AlphaTest: 0
Additive
Diffuse Color (Tint): 35,100,35
Specular Color: 0,0,0
Smoothness: 10
Emissive Color: 0,0,0
Glow Amount: 0
Diffuse: tartedtextures/white.tif
Normalmap
Specular
Detail
Heightmap
Decal
Dirt Map
[1] Custom
Indirect bounce color: 136,136,136
SSS Index: 0
[Color] Diffuse Color (Tint) = 35,100,35
samplesproject/materials/gettingstartedmaterials/gs_block [Block_01]

Assigning Material to Objects

Now that you have made your materials, you can assign them to the designer objects you created in the previous tutorial.

To assign materials to objects

1. First, assign the **sphere** material to the **sphere** object you created. To do this, in the **Perspective** viewport, select the sphere object you created.
2. In the **Material Editor**, select the **sphere** material that you created.
3. In the **Material Editor** toolbar, click **Assign Item to Selected Objects**. You can also right-click the material name and select **Assign to Selected Objects**.



Opacity 100
AlphaTest 0
Additive . . . .

Diffuse Color (Tint) 0,40,150
Specular Color 60,60,60
Smoothness 175
Emissive Color 0,0,0
Glow Amount 0

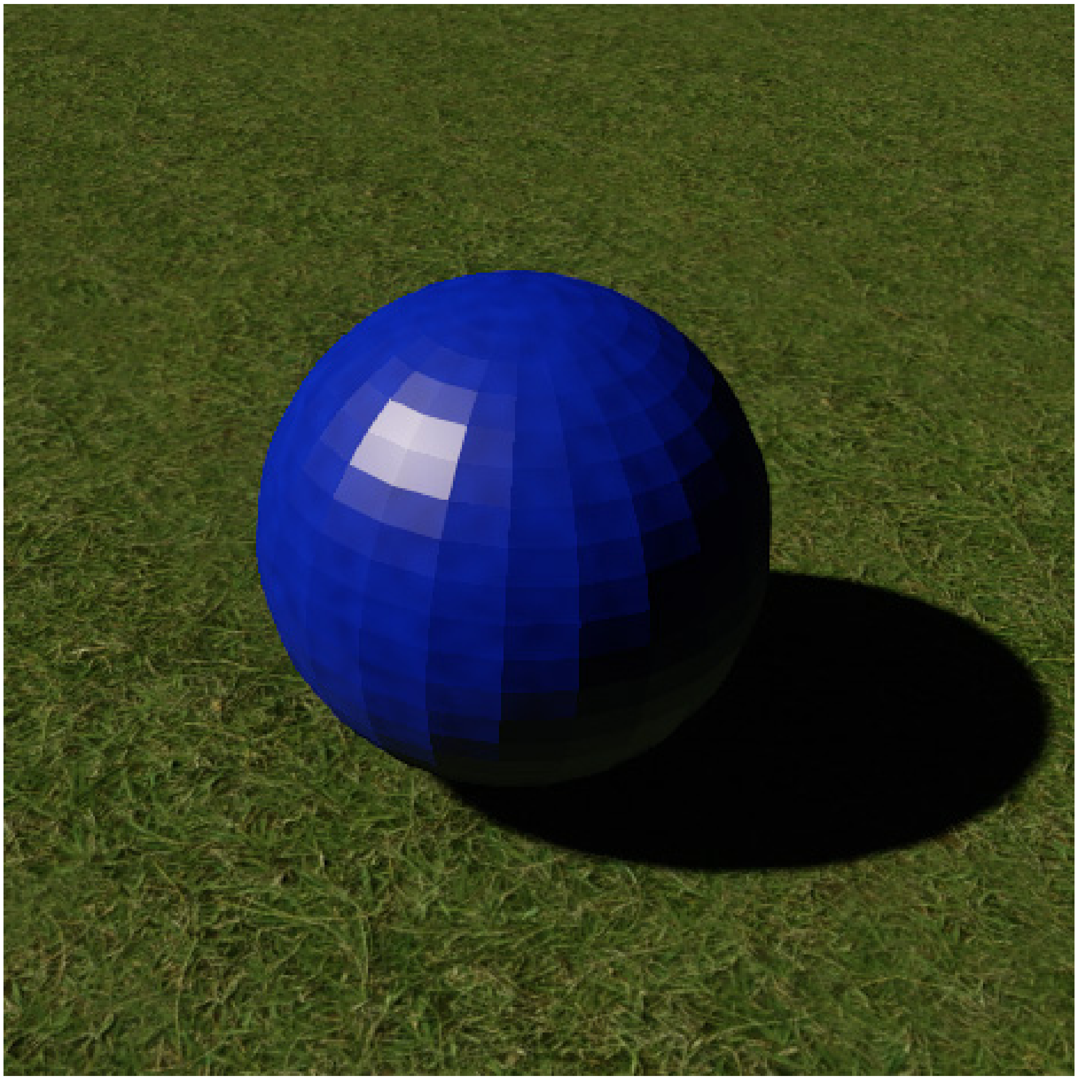
[Color] Diffuse Color (Tint) = 0,40,150

Diffuse . . . . tures/white.tif
Normalmap . . . .
Specular . . . .
Detail . . . .
Heightmap . . . .
Decal . . . .
Dirt Map . . . .
[1] Custom . . . .

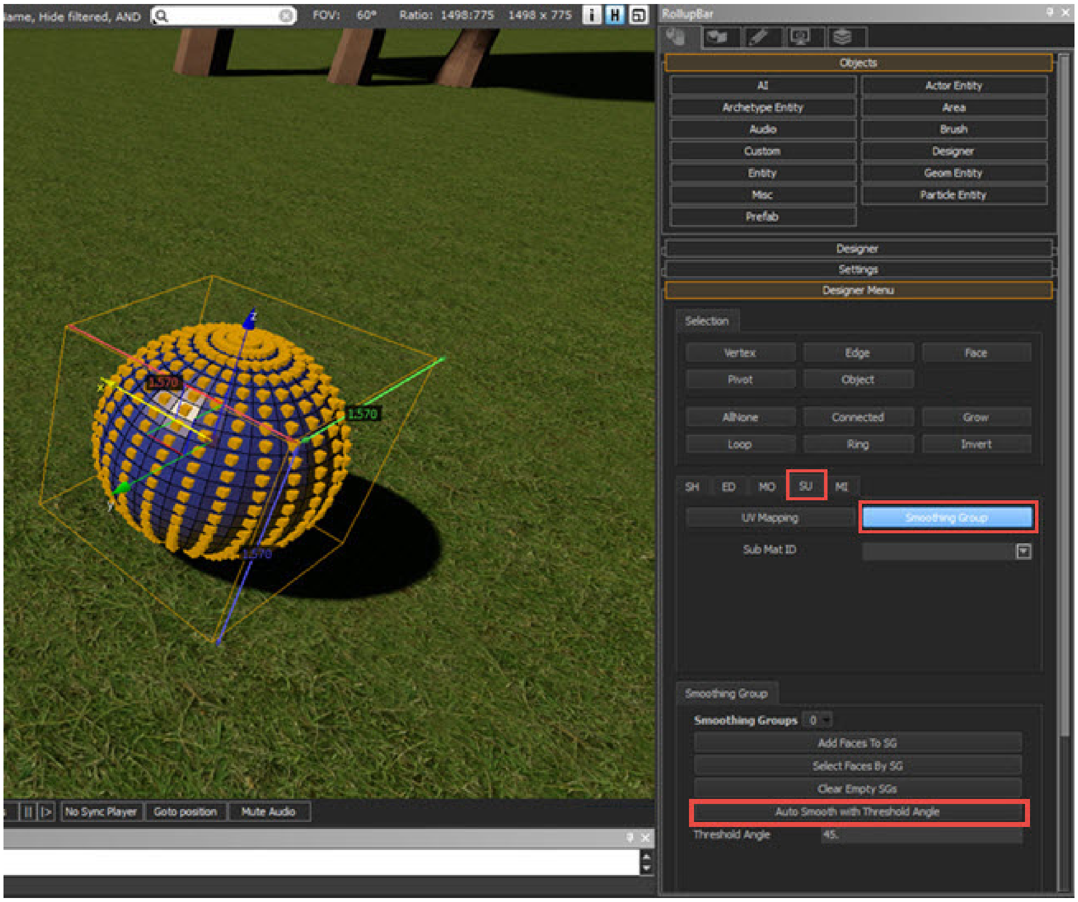
Indirect bounce color 136,136,136
SSS Index 0

materials/gettingstartedmaterials/gs_sphere

The sphere you created should now look something like this.

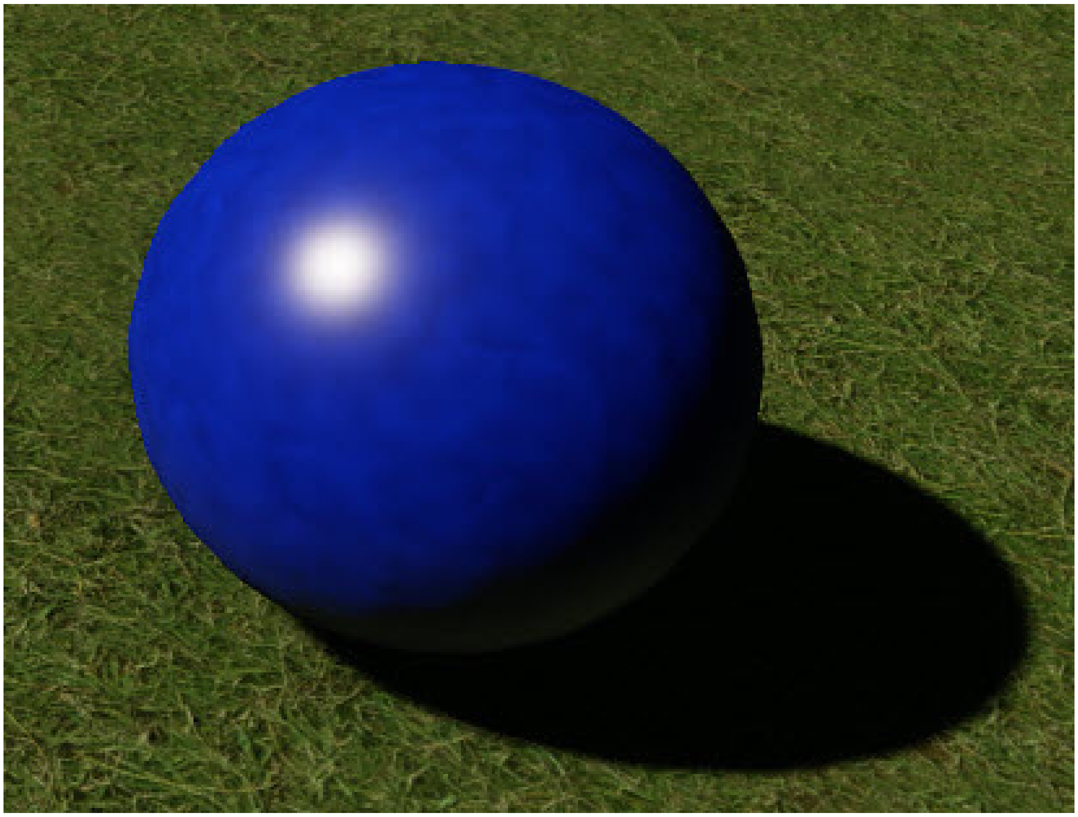


4. The designer sphere's faces are faceted. You can smooth the faces of the sphere. Select the sphere and, in the **Rollup Bar**, open the **Designer** tool.
5. Under **Designer Menu**, select the **SU** tab.
6. Click **Smoothing Group**, and drag a selection box over the sphere to select all its faces:



7. Below the **SU** tab, a **Smoothing Group** tab appears. Click **Auto Smooth with Threshold Angle**. (To see the result, click **Object** in the **Selection** tab to close the **Smoothing Group**.)

   The **Smoothing Group** smooths out the sphere so that it appears smooth and round.

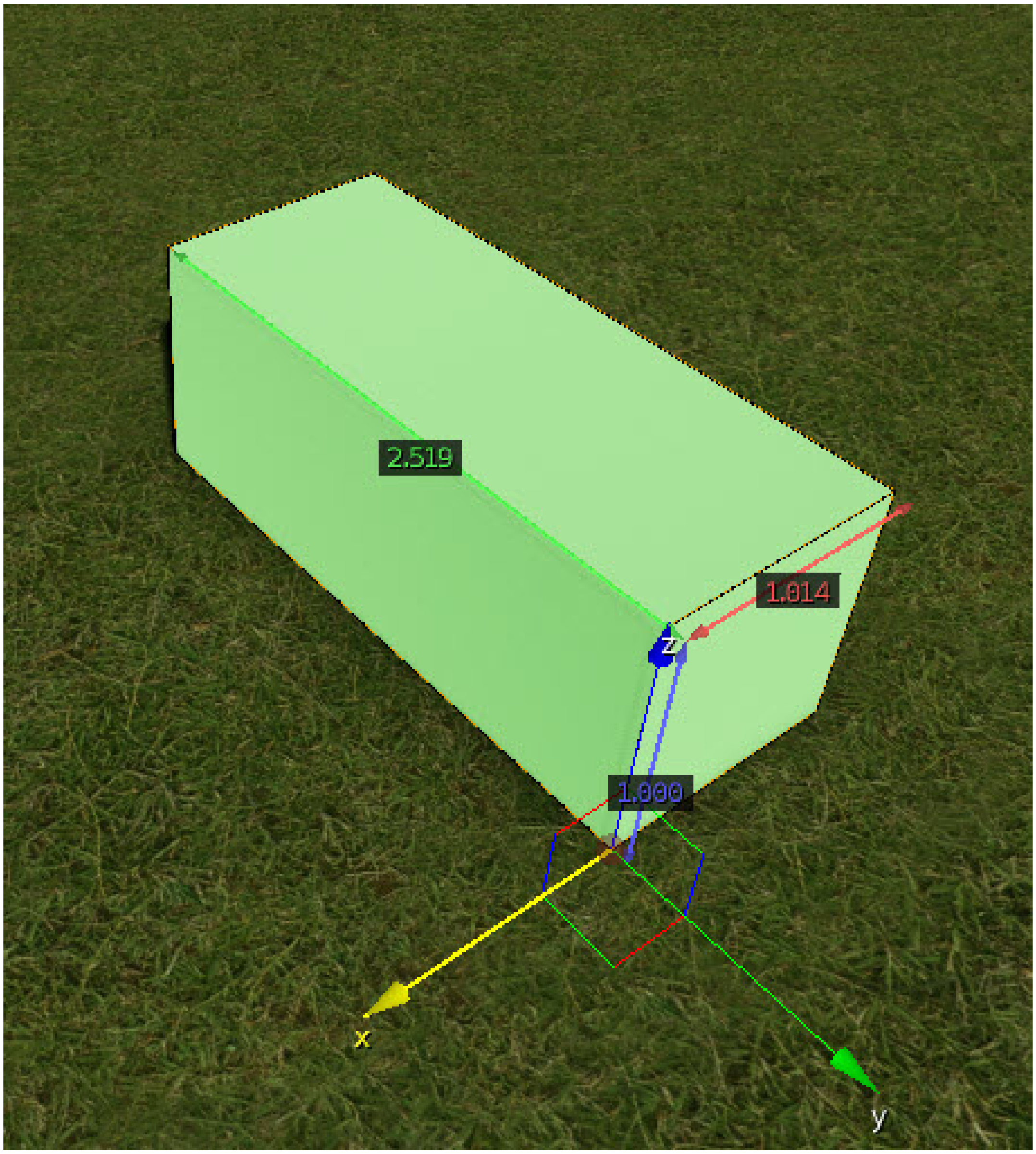


Applying a Multimaterial to an Object

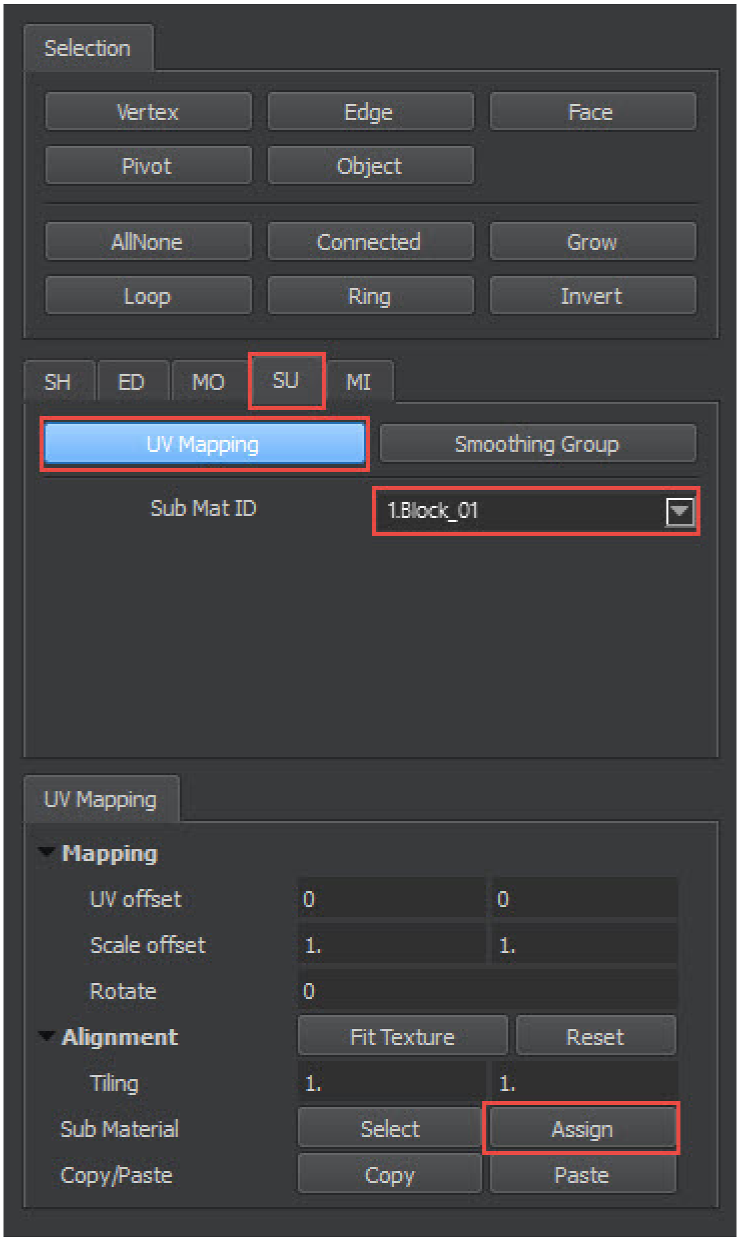
You can also use the **Material Editor** to apply multimaterial to an object.

To apply the multimaterial to the block that you created:

1. Select the block that you created.
2. Assign the **block** material to the block. Your block should look like the following image:



3. Now you need to assign different material IDs to the faces of the box. In the **Rollup Bar**, click the **Designer** tool.
4. Under **Designer Menu**, click the **SU** tab. Click **UV Mapping Button**.



Sub Mat ID: 1.Block_01

UV offset: 0 | 0

Scale offset: 1. | 1.

Rotate: 0

Tiling: 1. | 1.

Sub Material

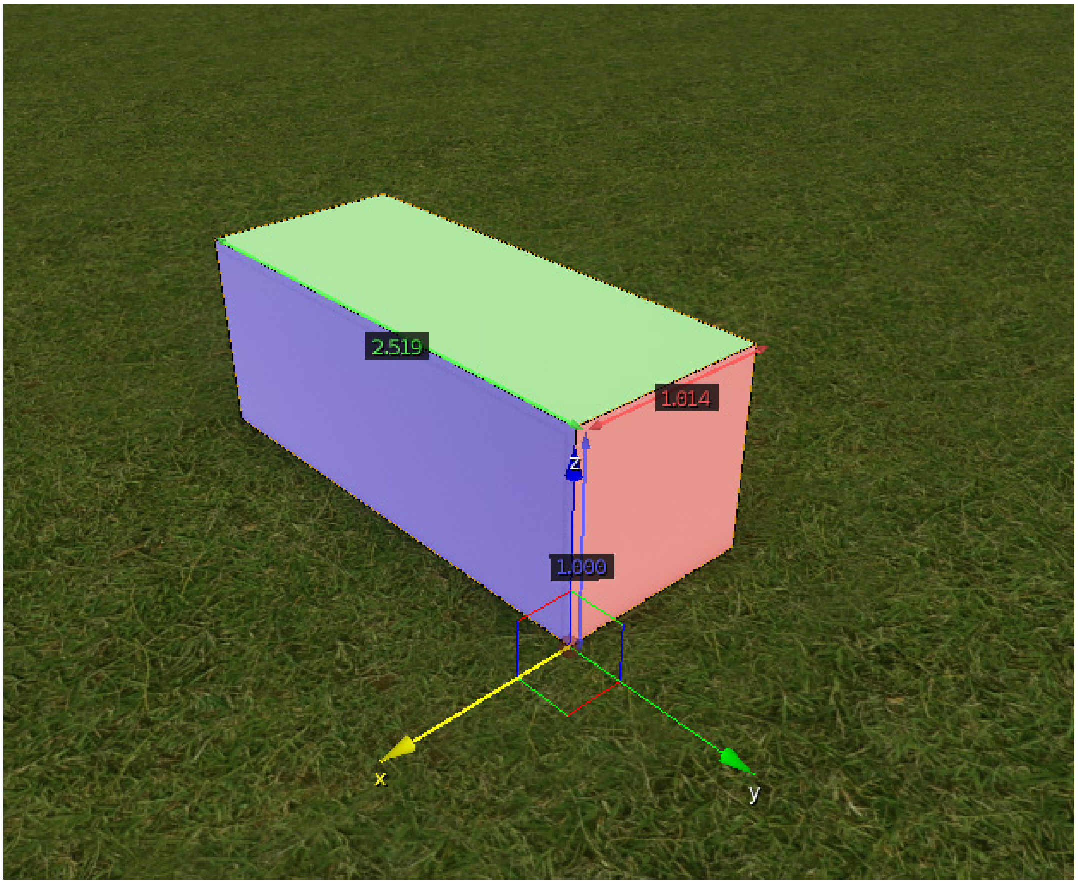
Copy/Paste

5. The faces are now selectable. Select the faces on each end of the block. To select more than one face, click on the first face, then hold down the **Ctrl** key to select additional faces.
6. With the end faces selected, under **UV Mapping**, click **Sub Mat ID** and then click **2.Block_02**.
7. In the **UV mapping** tab at the bottom, click **Assign**.

   The block_02 submaterial is now assigned to the ends of the block.
8. Select the two sides of the box.
9. Click **Sub Mat ID** and then click **3.Block_03**. Click **Assign**.
10. Click **Object** in the **Selection** tab to close the **UV Mapping** tool.

    You now have a block with multi-materials assigned to it.

    Close the **Material Editor**.

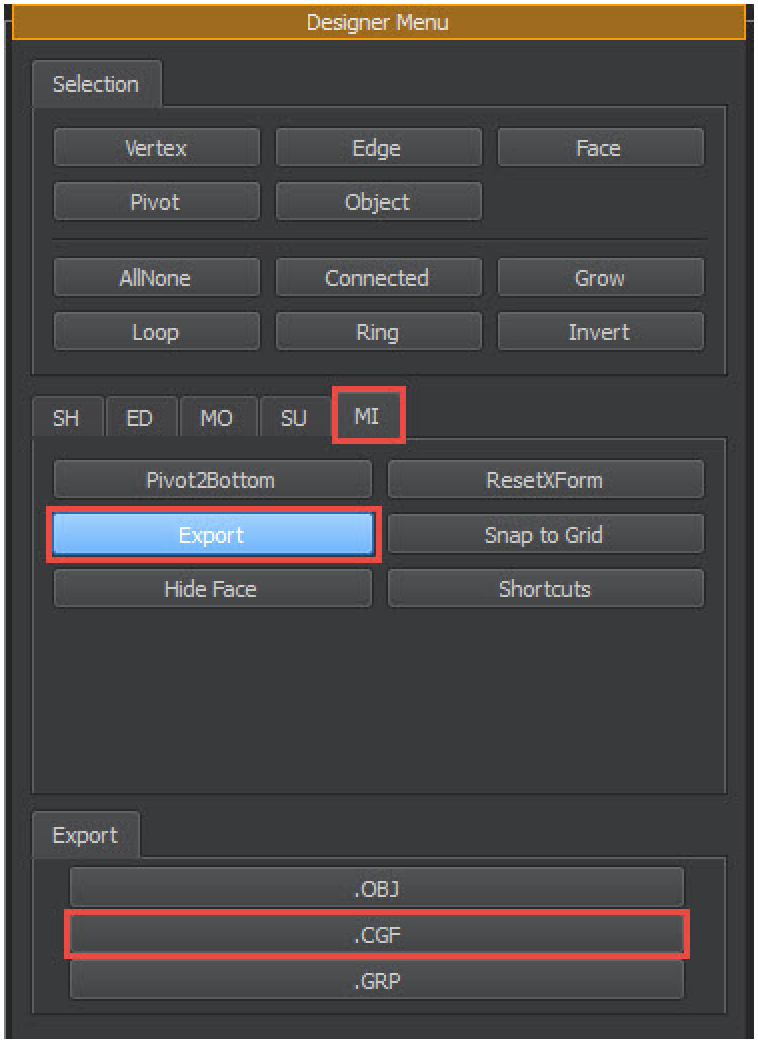

2.519
1.014
z
1.000
x
y

Exporting Objects

Now that you have created objects and applied material, you can export these objects. You will use them for the tutorials on physics entities in the next section.

To export the objects

1. In the **Perspective** viewport select the block that you created in Creating Designer Objects (p. 66).
2. In the **Rollup Bar**, click **Designer**. Under **Designer Menu**, clicks the **MI** tab.
3. Click **Export**.
4. In the **Export** tab, click **.CGF**.



5. Navigate to the directory `SamplesProject\Objects\GettingStartedAssets`. In the **Save As** dialog box, for **File name**, type **`block`**. Click **Save**.
6. In the **Selection** tab, click **Object** to close the **Export** tool.
7. Repeat these steps for the sphere, but name it **`sphere`**.

Creating Physics Objects

This tutorial steps you through how to to set up a physics object to work in the level and then activate during a game scripted event.

Topics

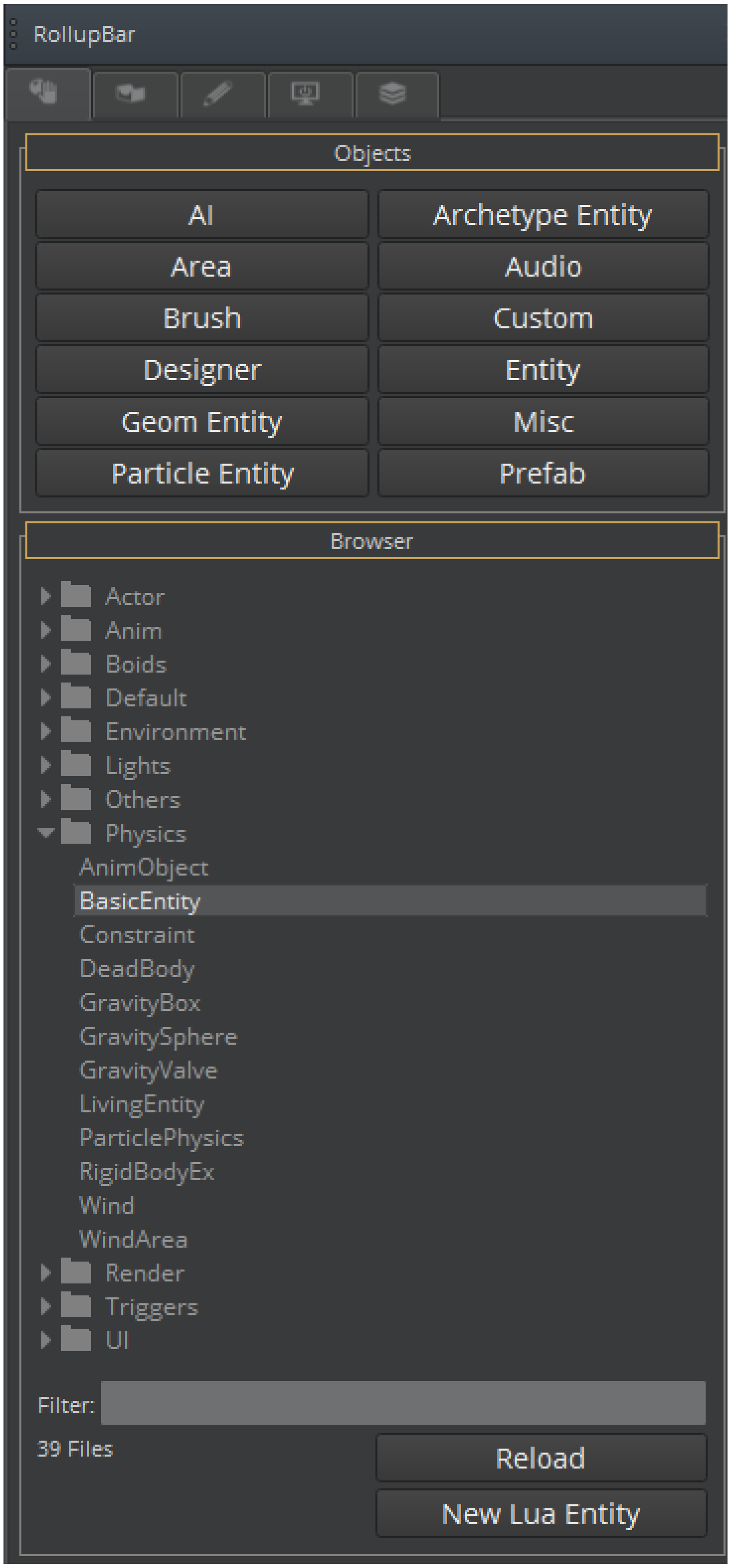
Creating a Physics Block

Physics entities are used to simulate physical events such as explosions, gravity fields, or wind, or to physicalize objects such as cloth, breakable entities, or ropes.

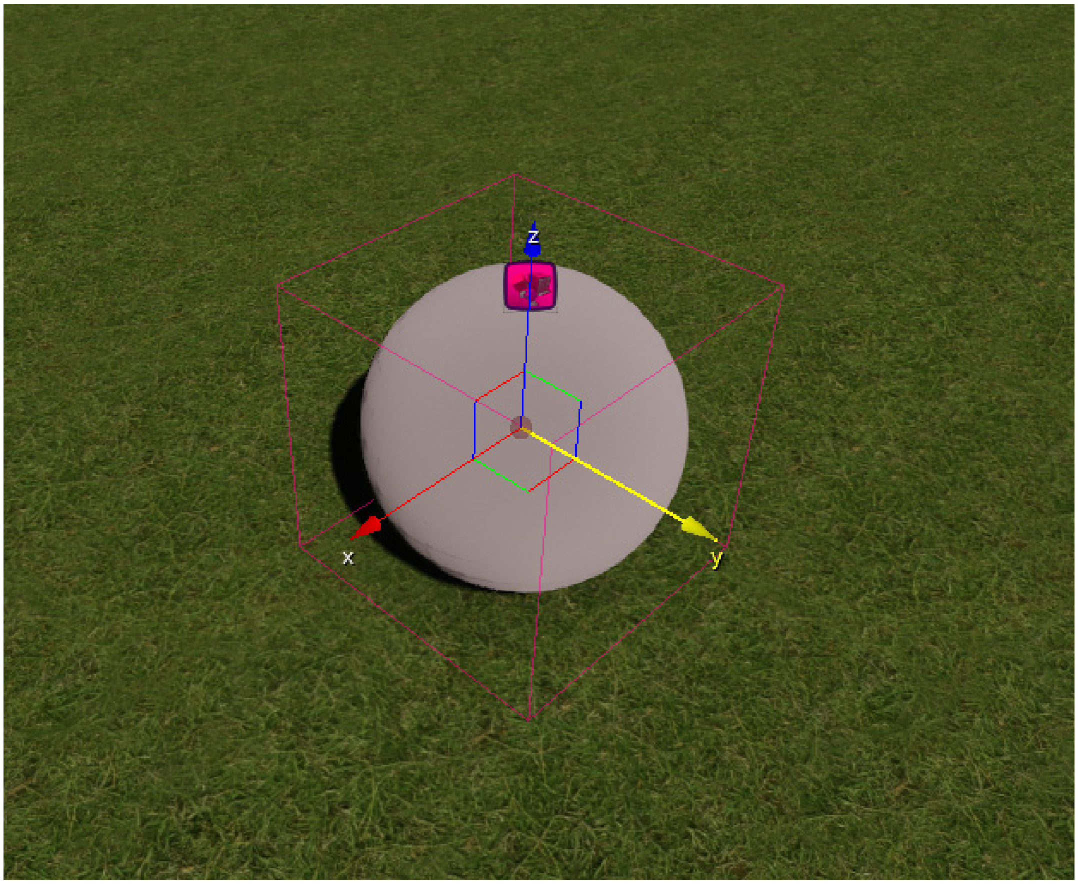
In this tutorial, you will add a physics entity to the level, assign it the block mesh created in previous tutorials, and give it mass, effectively turning it on. You will later place multiple physics blocks to build a wall to shoot cannons at in the game.

To add a physics entity

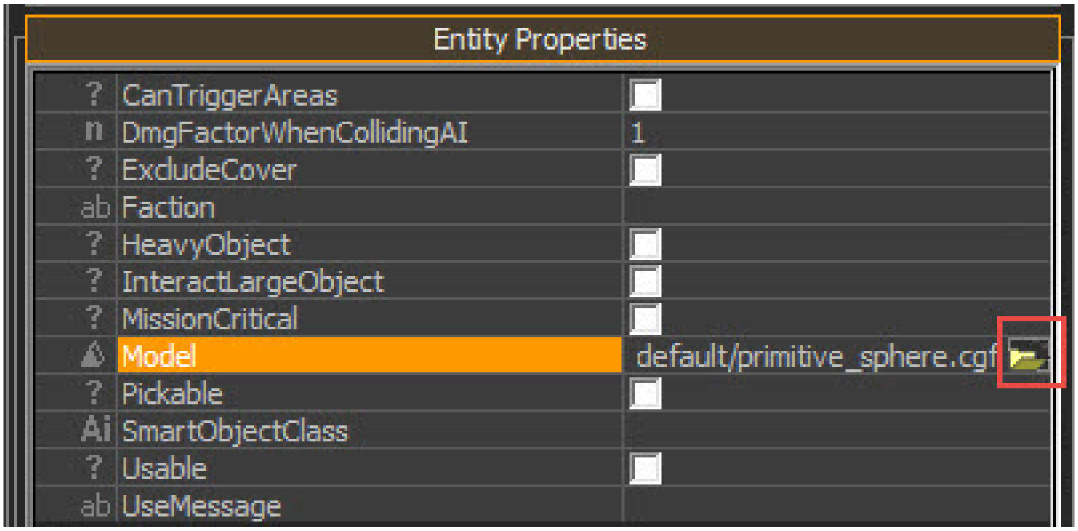
1. In the **Rollup Bar**, on the **Objects** tab, click **Entity**.
2. Under **Browser**, expand **Physics**. Drag **BasicEntity** into the **Perspective** viewport.


RollupBar
Objects
AI
Archetype Entity
Area
Audio
Brush
Custom
Designer
Entity
Geom Entity
Misc
Particle Entity
Prefab
Browser
Actor
Anim
Boids
Default
Environment
Lights
Others
Physics
AnimObject
BasicEntity
Constraint
DeadBody
GravityBox
GravitySphere
GravityValve
LivingEntity
ParticlePhysics
RigidBodyEx
Wind
WindArea
Render
Triggers
UI
Filter:
39 Files
Reload
New Lua Entity

The **BasicEntity** object appears as a sphere.



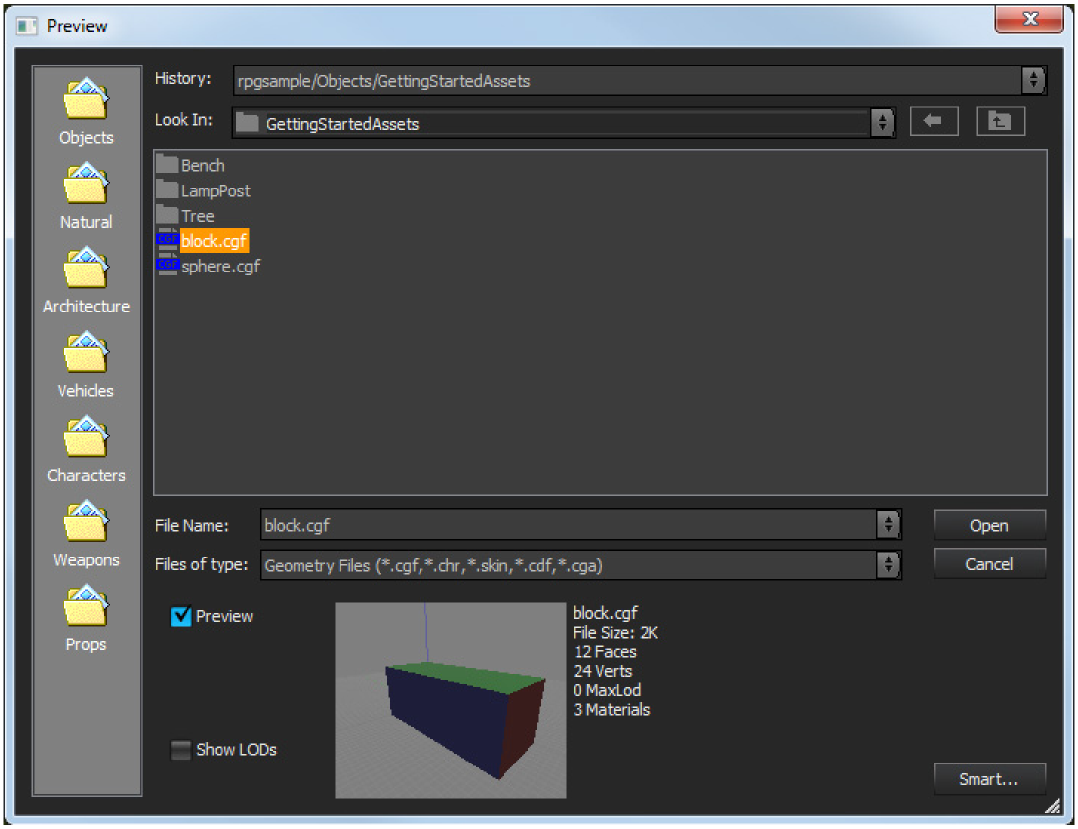
3. Under **Entity Properties**, select the attribute **Model** and click the folder icon.



Entity Properties

?	CanTriggerAreas	☐
n	DmgFactorWhenCollidingAI	1
?	ExcludeCover	☐
ab	Faction	
?	HeavyObject	☐
?	InteractLargeObject	☐
?	MissionCritical	☐
	Model	default/primitive_sphere.cgf
?	Pickable	☐
Ai	SmartObjectClass	
?	Usable	☐
ab	UseMessage	

4. Navigate to `SamplesProject\Objects\GettingStartedAssets` and open `block.cgf`.



History: rpgsample/Objects/GettingStartedAssets

Look In: GettingStartedAssets

Bench
LampPost
Tree
block.cgf
sphere.cgf

File Name: block.cgf

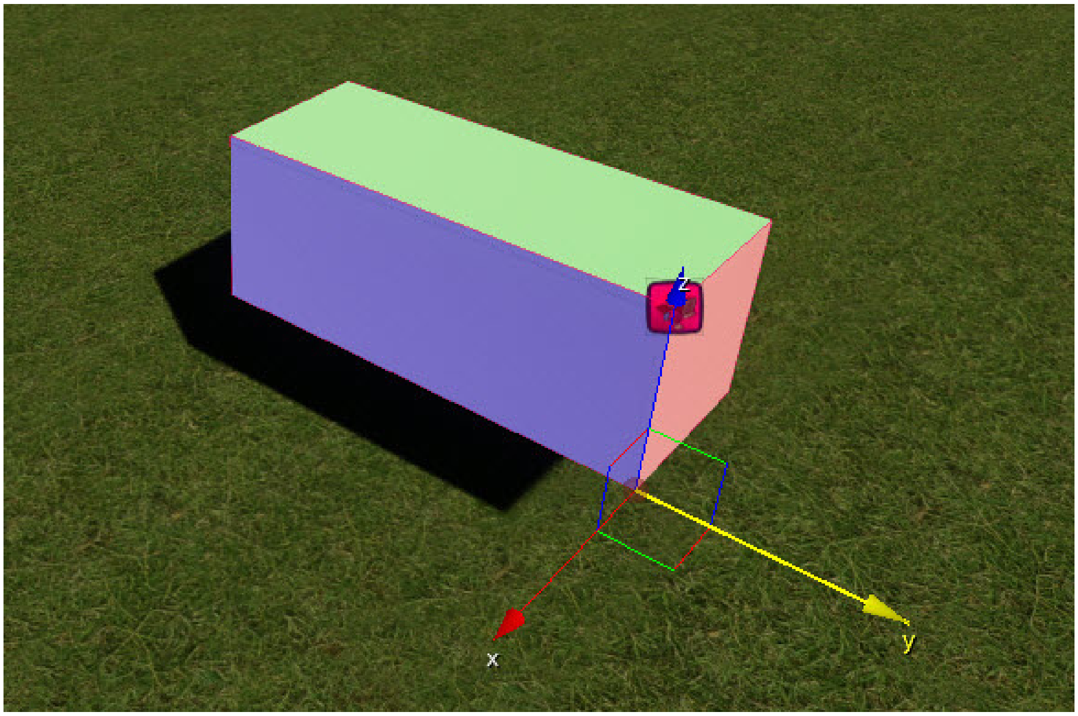
Files of type: Geometry Files (*.cgf,*.chr,*.skin,*.cdf,*.cga)

☑ Preview

□ Show LODs

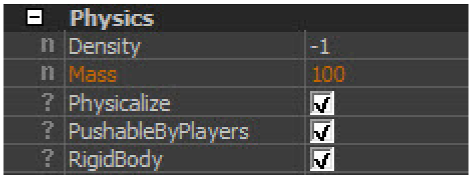
block.cgf
File Size: 2K
12 Faces
24 Verts
0 MaxLod
3 Materials

The sphere changes to the block that you created in the previous tutorials.



5. In **Entity Properties**, set **Mass** to **`100`**.

   This turns on the physics object so that the block can respond to events. Any value above 0 turns on the physics object. Higher values indicate greater mass, which affects the object's behavior.



⊟ Physics	
Density	-1
Mass	100
Physicalize	☑
PushableByPlayers	☑
RigidBody	☑

   You will do more with this physics block in the scripting tutorial.

Creating an Archetype Entity

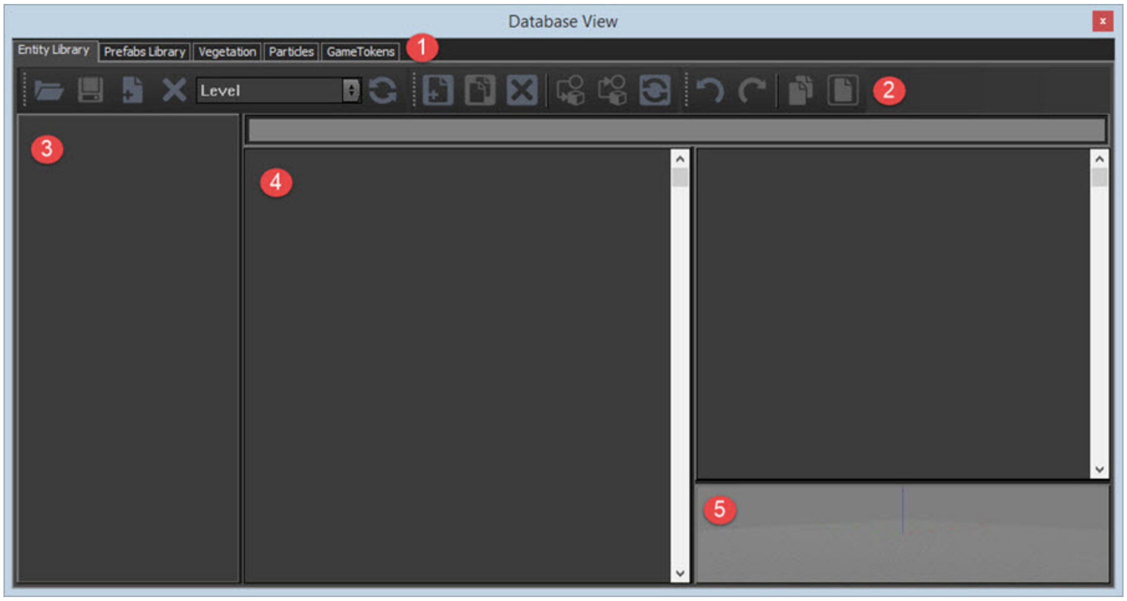
Now you'll create a physics object that appears in the level when activated through a flow graph–scripted event. To do this, you define an archetype entity in the **Database View** editor.

An archetype entity is based on a regular entity and specifies individual parameter values for that entity. If the value of an archetype parameter is changed, all instances of that archetype are updated automatically. As such, you can predefine variations of entity classes as archetype entities that can be used throughout the game. For global changes affecting all instances, the archetype entity needs to be changed only once.

This allows you to create specific entity types that can be placed across multiple level files or spawned into a level based on a scripted game event.

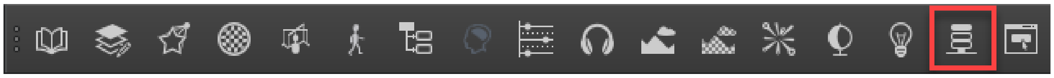
The **Database View** editor for archetype entities has the following features:

1. **Database tabs** – Database types to view and manage (such as **Prefabs Library** and the **Vegetation** tool set)
2. **Editor toolbar** – Tools to load, save, add, and remove archetype entities
3. **File tree view** – View of the archetypes available for use
4. **Settings and properites** – Settings that can be edited for any archetype entity selected
5. **Preview pane** – Preview pane for the selected archetype entity

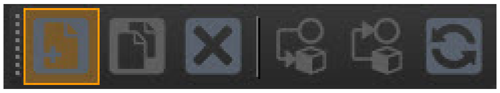


To define an archetype entity

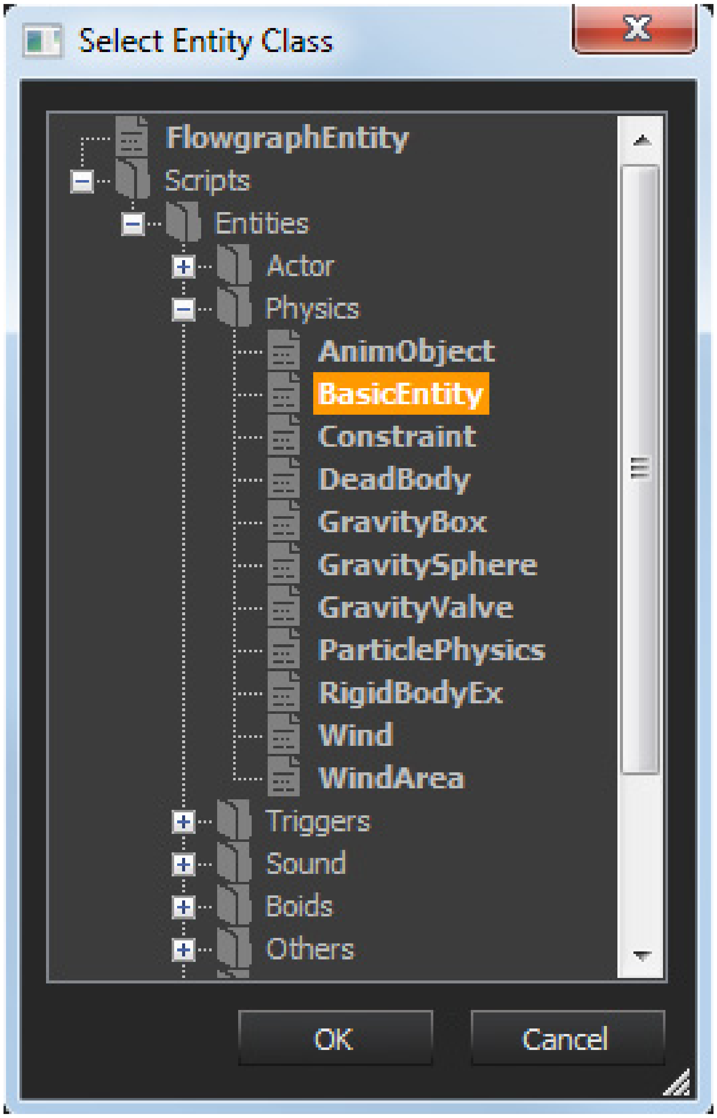
1. Do one of the following to open the **Database View** editor:
   - On the editor toolbar, click **Database View**.



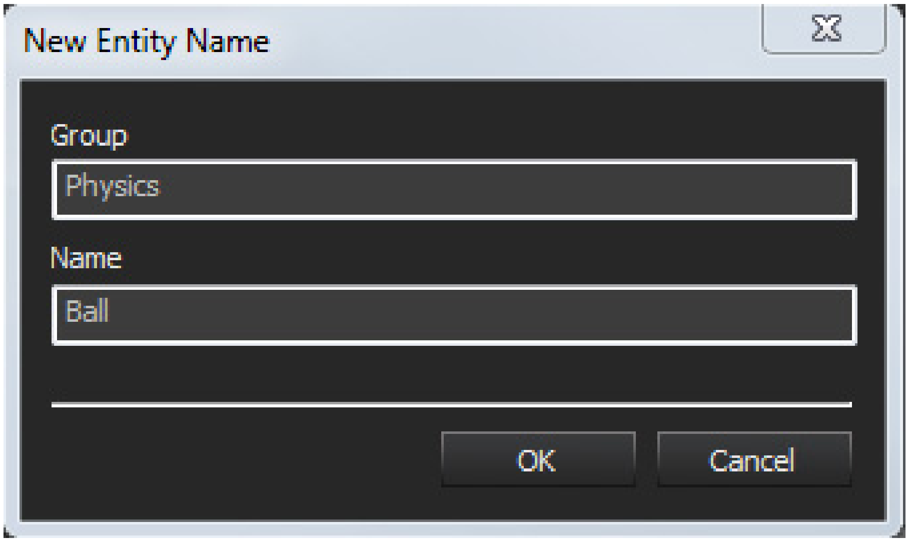
   - From the main menu, click **Tools**, **Other**, **Database View**.
2. In the toolbar, click **Add New Item**.



3. Expand `Scripts\Entities\Physics`. Select `Basic Entity`. Click **OK**.



4. For **Group**, type **`Physics`**. For **Name**, type **`Ball`**. Click **OK**.

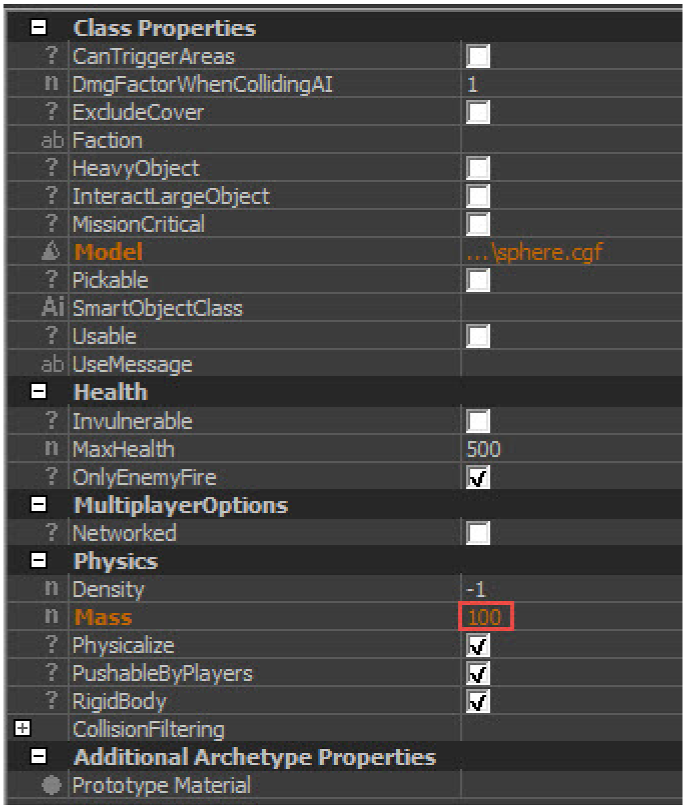


New Entity Name

Group: Physics

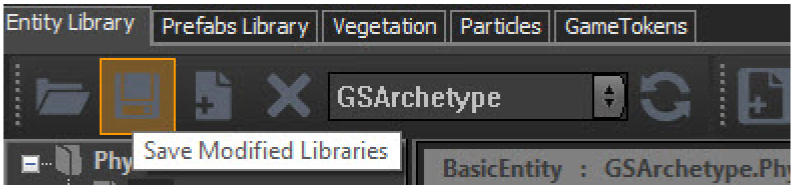
Name: Ball

5. Under **Class Properties**, click **Model**. Click the folder icon.
6. Navigate to `SamplesProject\Objects\GettingStartedAssets` and open `sphere.cgf`.
7. Under **Physics**, click **Mass** and set it to **`100`**.



⊟	**Class Properties**	
?	CanTriggerAreas	☐
n	DmgFactorWhenCollidingAI	1
?	ExcludeCover	☐
ab	Faction	
?	HeavyObject	☐
?	InteractLargeObject	☐
?	MissionCritical	☐
	Model	...\sphere.cgf
?	Pickable	☐
Ai	SmartObjectClass	
?	Usable	☐
ab	UseMessage	
⊟	**Health**	
?	Invulnerable	☐
n	MaxHealth	500
?	OnlyEnemyFire	☑
⊟	**MultiplayerOptions**	
?	Networked	☐
⊟	**Physics**	
n	Density	-1
n	**Mass**	100
?	Physicalize	☑
?	PushableByPlayers	☑
?	RigidBody	☑
⊞	CollisionFiltering	
⊟	**Additional Archetype Properties**	
●	Prototype Material	

8. Click the **Save** icon in the **Database View** editor.



9. Make note of the entity name: **Level.Physics.Ball**. You will use this name in the next tutorial to reference this object to spawn.

Scripting with Flow Graph

This tutorial introduces the concept of game play scripting using the **Flow Graph** editor. You'll create a scene that allows the player to use a makeshift cannon to shoot objects at a wall you that you build at the end of the street. To do that, you need to set up flow graph scripts that do five things:

1. Add a trigger that toggles the point of view between two cameras.
2. Use the keyboard to move a block (your cannon) left and right.
3. Set up same block to spawn (create) and launch a physics sphere (your cannonball).
4. Use the keyboard to control the power input of the block launcher (cannon).
5. Enable the cannon and its controls only when triggered by the camera state.

Topics

Preparing the Scene

Before you build a script, you must prepare the scene with several entities that the flow graph scripts will reference.

In this topic, you will:

- Place tag points, which mark the launching point of the ball and the points between which the cannon moves
- Link a tag point to the block to create the cannon
- Stack physcis blocks to create a wall at the end of the street
- Create a proximity trigger, which is an area that, when entered, triggers an action
- Add a camera, to which the view changes when a player enters the trigger volume

Topics

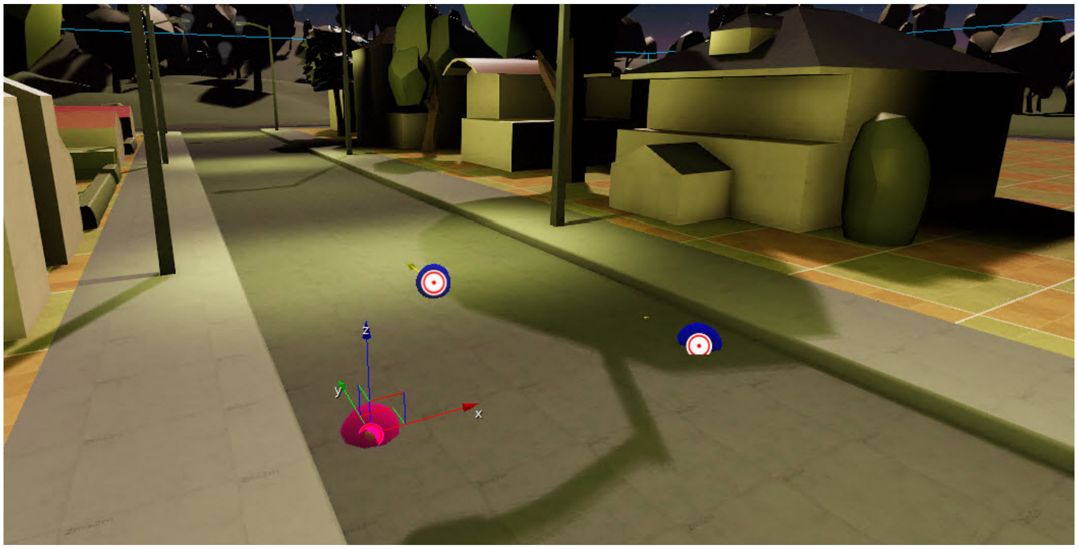
Placing Tag Points

First, you add three tag points, which are markers that use XYZ axes to define locations in a level. In Creating a Flow Graph Script (p. 110), you use two of these tag points to define the points between which the cannon moves when pressing input keys on your keyboard. The third tag point is used to define the point from which the ball is launched from the cannon.

To add tag points

1. In your **Perspective** viewport, navigate approximately to the halfway point on the length of the street. This is where you will place your tag points.
2. In the **Rollup Bar**, on the **Objects** tab, click **AI**.
3. Under **Object Type**, click **Tagpoint**. Move your pointer into the **Perspective** viewport and then click to place the tag point.
4. Repeat the previous two steps twice so that you have a total of three tag points in the scene. Place them as shown in the following picture.

   Tip
   If your tag points do not appear as spheres (as shown in the following image), click the **H** icon on the **Perspective** viewport's title bar.



5. Select one of the tag points. Notice that the XYZ gizmo—depending on how you've built the scene—runs along either the Y or X axis of the street. This means that either the X axis or the Y axis points in the same direction as your street. Note which axis matches the street's direction.

   In this example scene, the street is built along the Y axis, which means you'll use the Y direction to launch your ball.

   If your road runs along the X axis, you'll use the X forward direction instead.

   You will use this information in the next section of the tutorial.

Adding Block and Linking the Tag Point

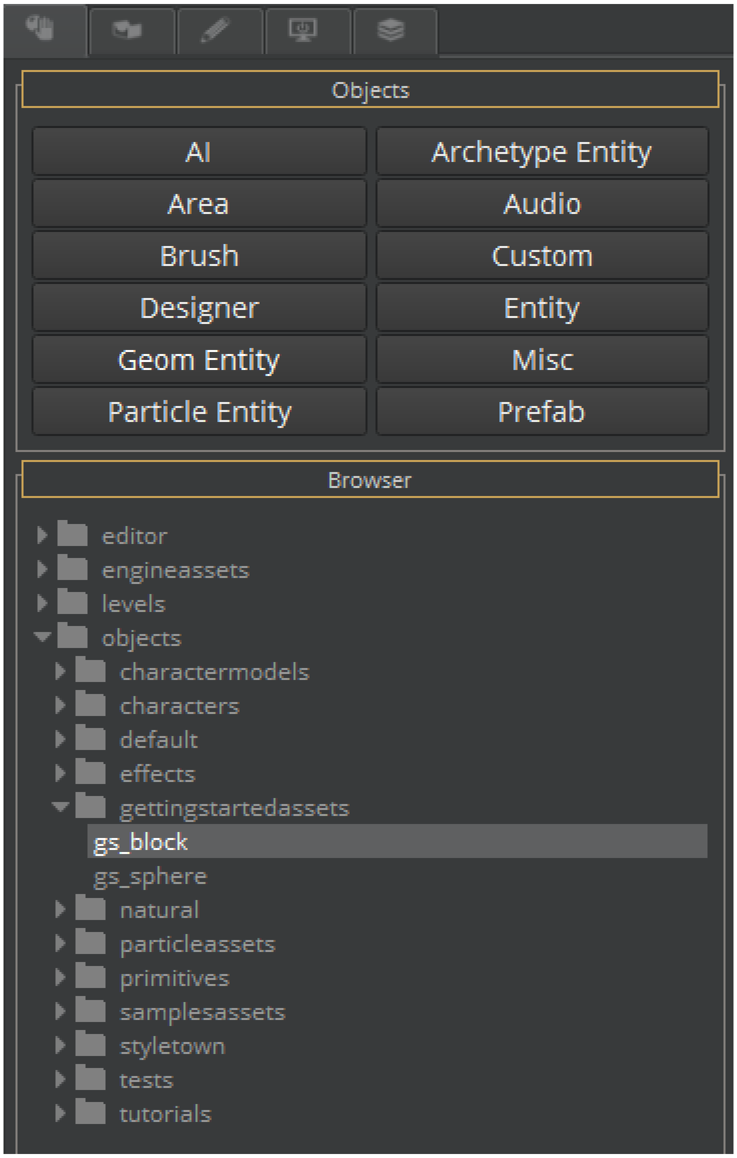
Next, you'll add the block you created earlier and then link a tag point to it. When you link a tag point to an object, you can move the object, and the tag point remains with its linked object. In this tutorial, you rotate your canon, and the tag point moves with and remains linked to the end of the block (your cannon).

In Creating a Flow Graph Script (p. 110), you'll use the linked tag point to determine the exact location from which the spawned ball is launched.

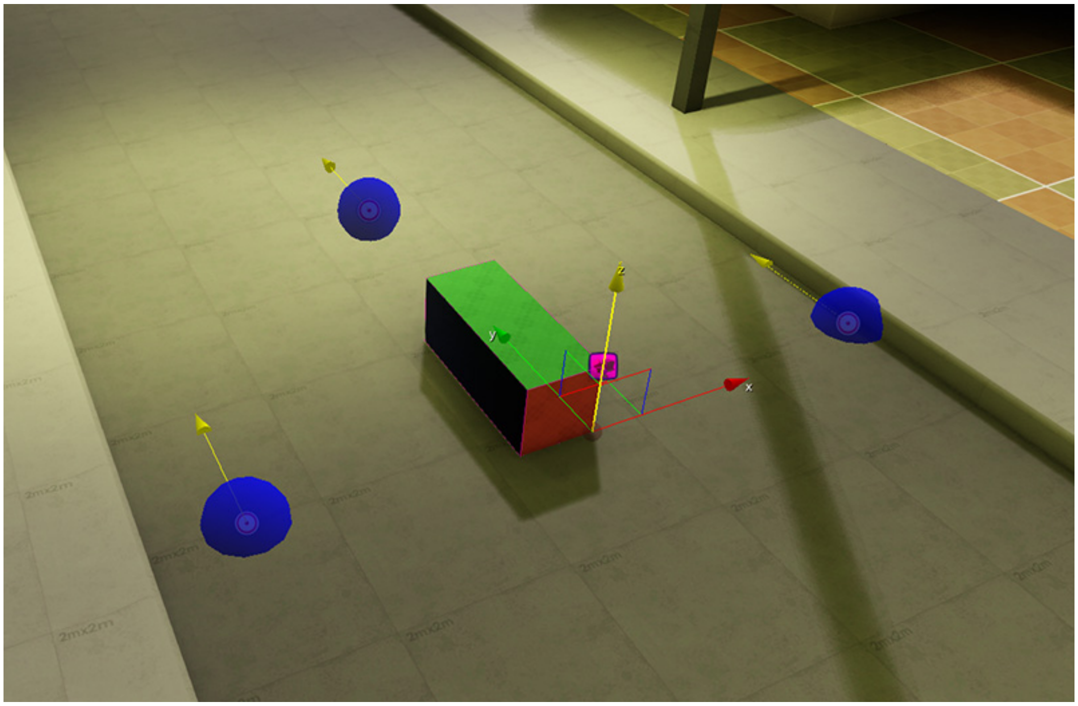
To add the block and link a tag point

1. In the **Rollup Bar**, on the **Objects** tab, click **Geom Entity**.
2. Under **Browser**, expand `objects\gettingstartedassets` and select the entity block that you previously created.
3. Drag the block from the **Rollup Bar** into the **Perspective** viewport near the tag points.

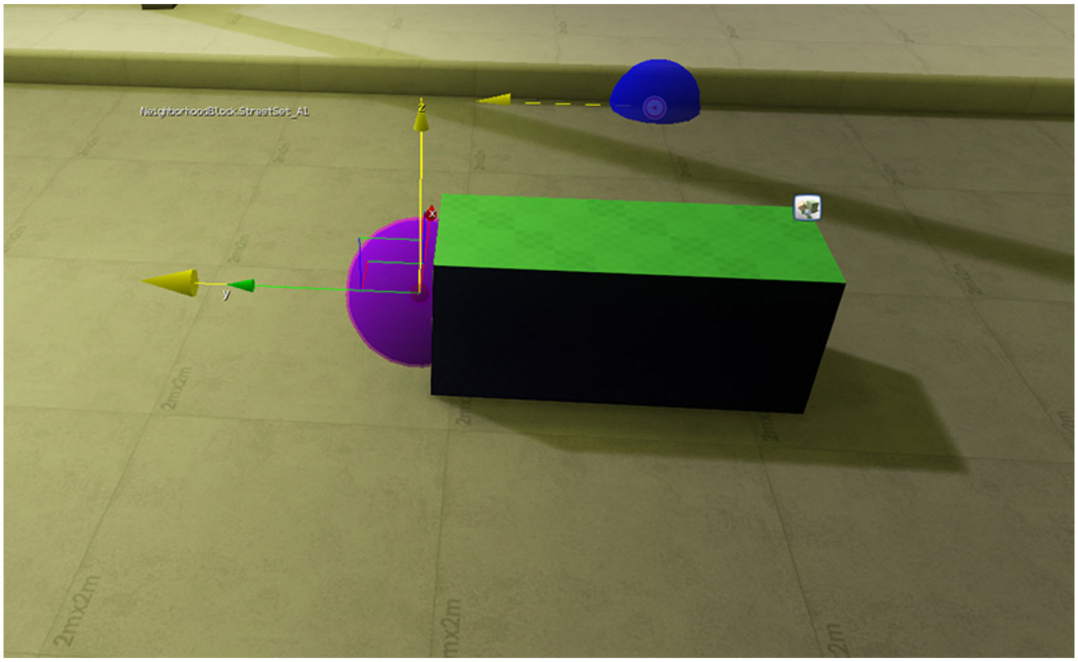
   Tip
   If you do not see your block and sphere in the list, click **Reload** at the bottom of the **Browser** area.



4. Position the block between the two tag points in the street.



5. Position the third tag point so that it's centered in front of the block.

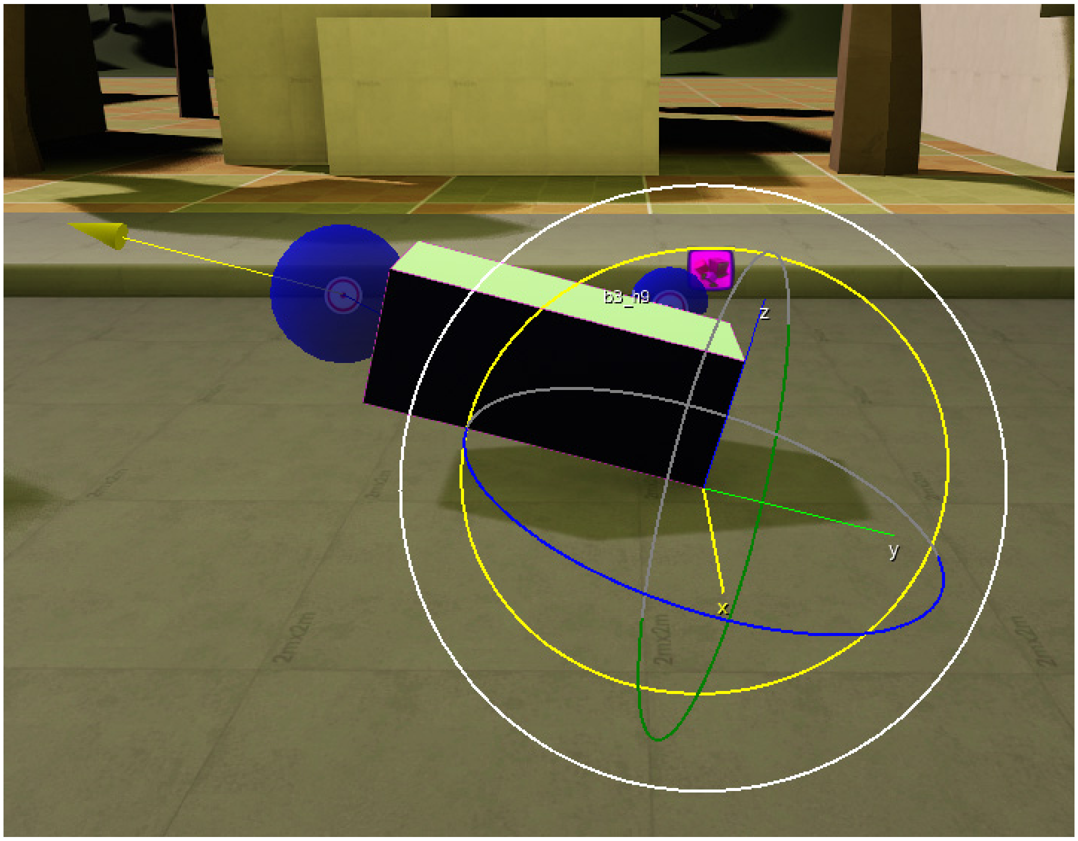


6. In the editor toolbar, click **Link Object** .

   Select the tag point in front of the block and then drag toward the block. The pointer displays a link icon as well as the object name it can link to (if present). When the block name appears, release the mouse button.

   The tag point and block are now linked. Test this by moving the block around. If the tag point moves with the block, the objects are successfully linked.

7. Rotate the block up about 20 degrees. The linked tag point moves with the block as it rotates.

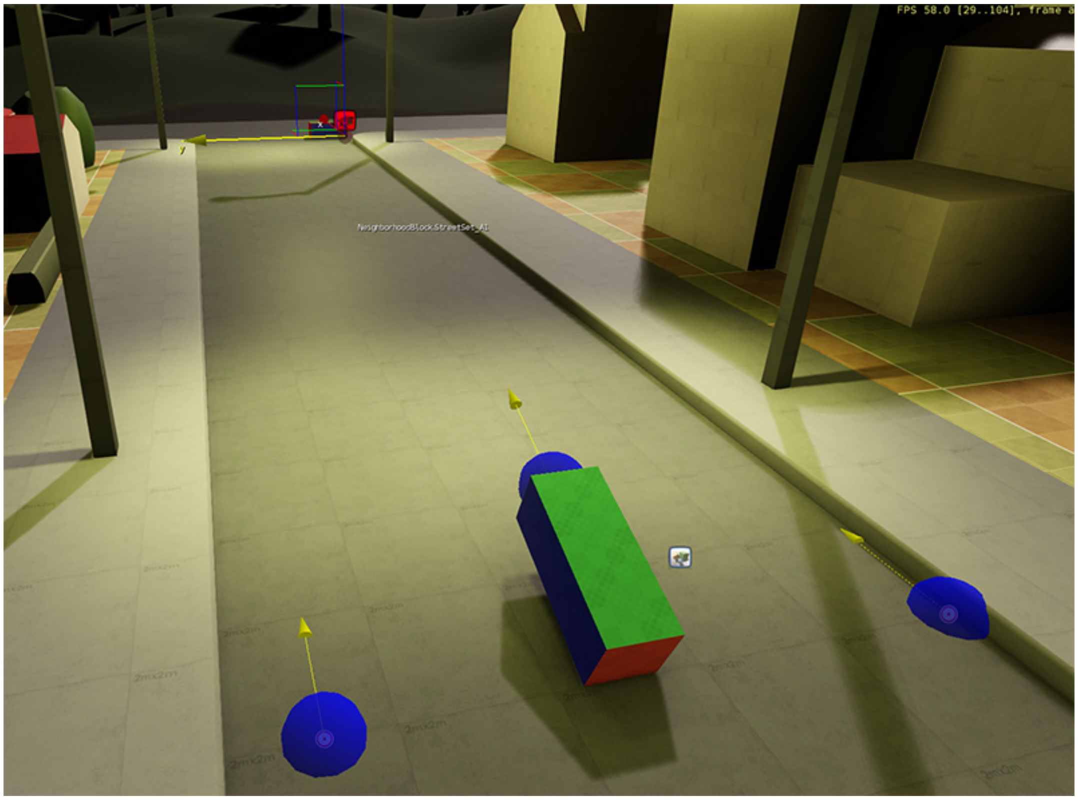


Stacking Physics Blocks

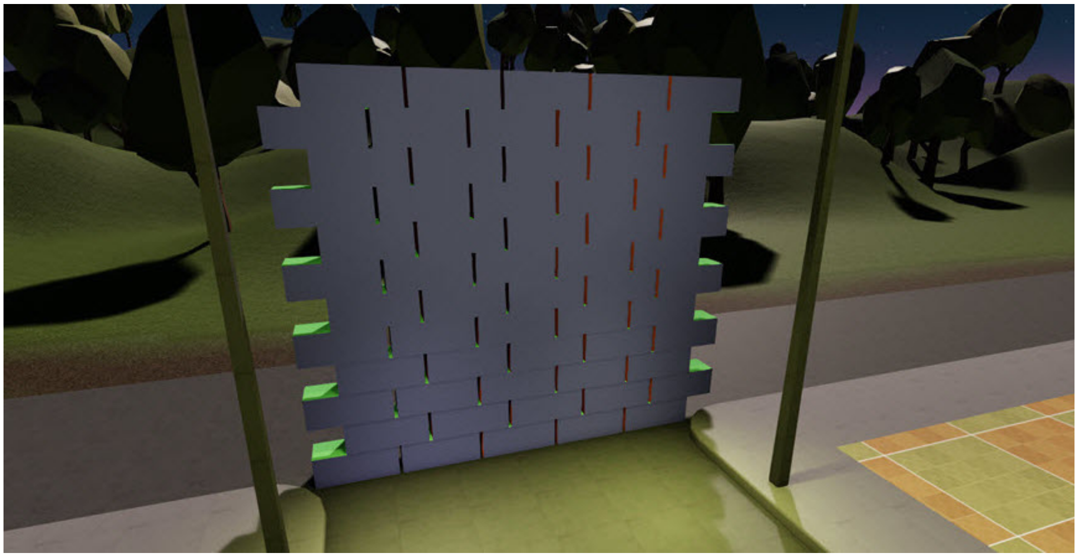
Next, you'll stack the physics blocks at the end of the street to create a wall to be knocked over by the balls you will shoot from your cannon.

To stack the physics blocks

1. Place into your scene the physics block that you created in the previous tutorial (p. 88).
2. Move the physics block to the end of the street, in front of the block and tag points, as shown in the following image.



3. Position the first block, then press **Ctrl+D** to duplicate the block and position the next block. Click to place it. Repeat this process to build a wall of blocks similar to the following:

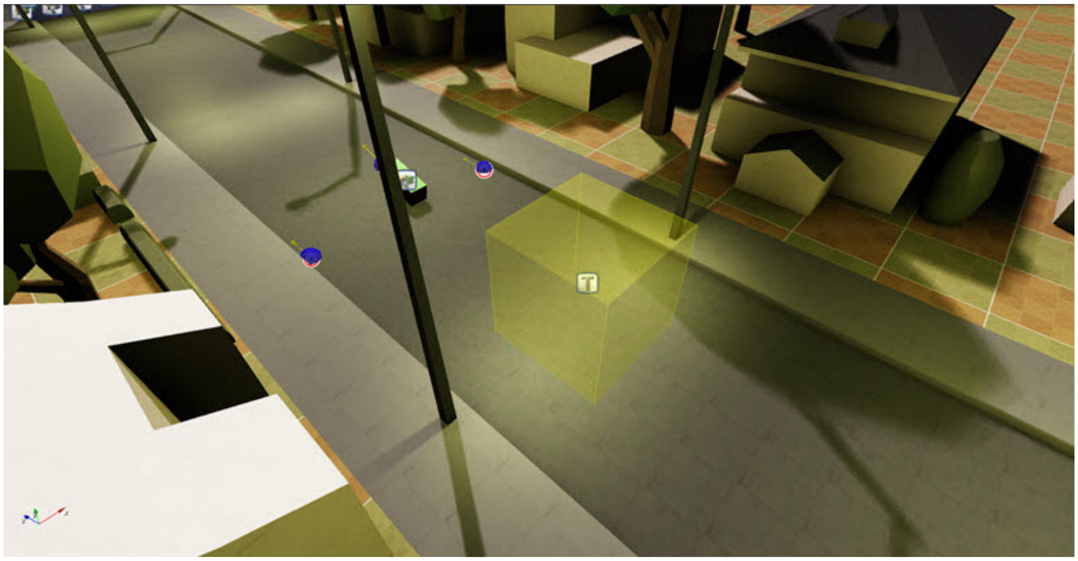


Adding a Proximity Trigger

A proximity trigger is a volume or region that, when entered, triggers a specified change in your level. In Creating a Flow Graph Script (p. 110), you use this proximity trigger to define the area that, when entered, switches to a first person camera view that allows the player to move and aim the cannon and shoot the ball at the wall. When the player exits the proximity trigger, the view returns to the default camera view.

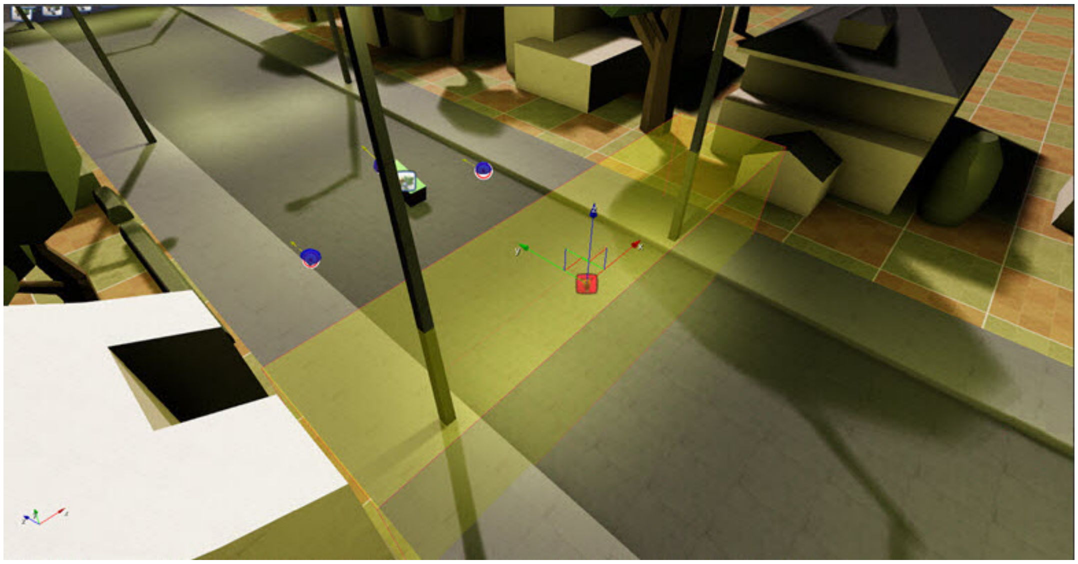
To add a proximity trigger

1. In the **Rollup Bar**, click **Entity**, and then expand **Triggers**. Drag **ProximityTrigger** into the **Perspective** viewport.
2. Position the proximity trigger behind the block and tagpoints, as shown in the following image.



3. With the proximity trigger selected, under **Entity Properties**, change the **DimX** (or **DimY**) dimensions so that the area, or volume, crosses the entire street. Change the **DimZ** dimension so that it is a little deeper on the street as well.

   For example, if Y is your forward direction, the street settings should be approximately **DimX**=`20`, **DimY**=`10`.



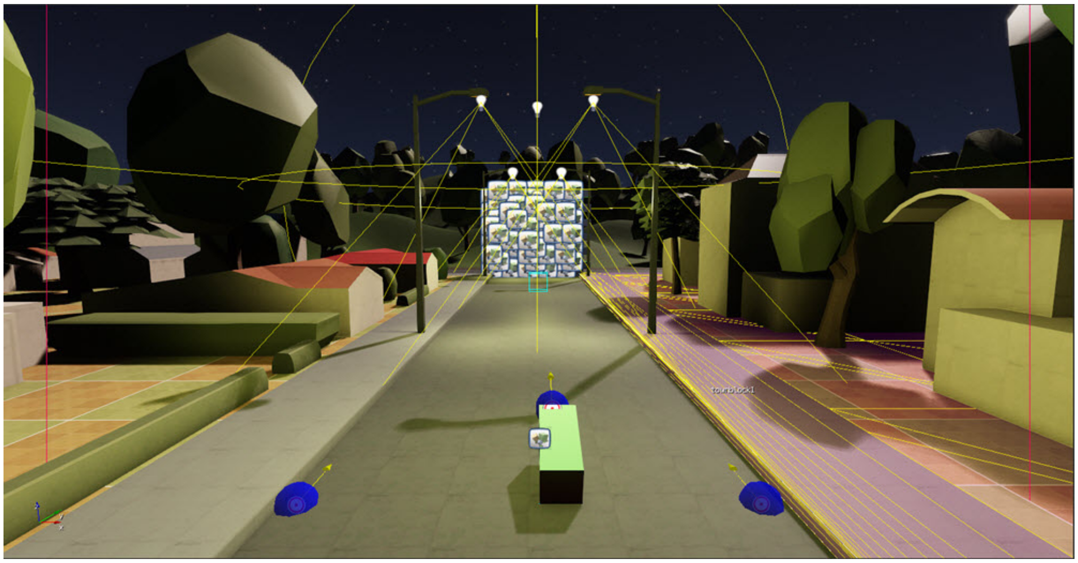
4. Under **Entity Properties**, deselect the **OnlyPlayer** check box.

Adding a Camera

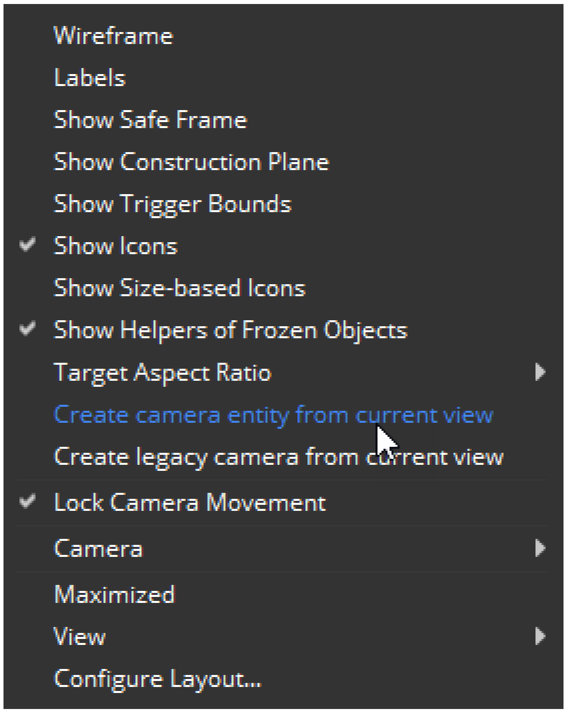
Now you'll add a camera that switches to a first person point of view when the player enters the proximity trigger area. This view allows the player to control the block cannon and shoot at the wall.

To add a camera

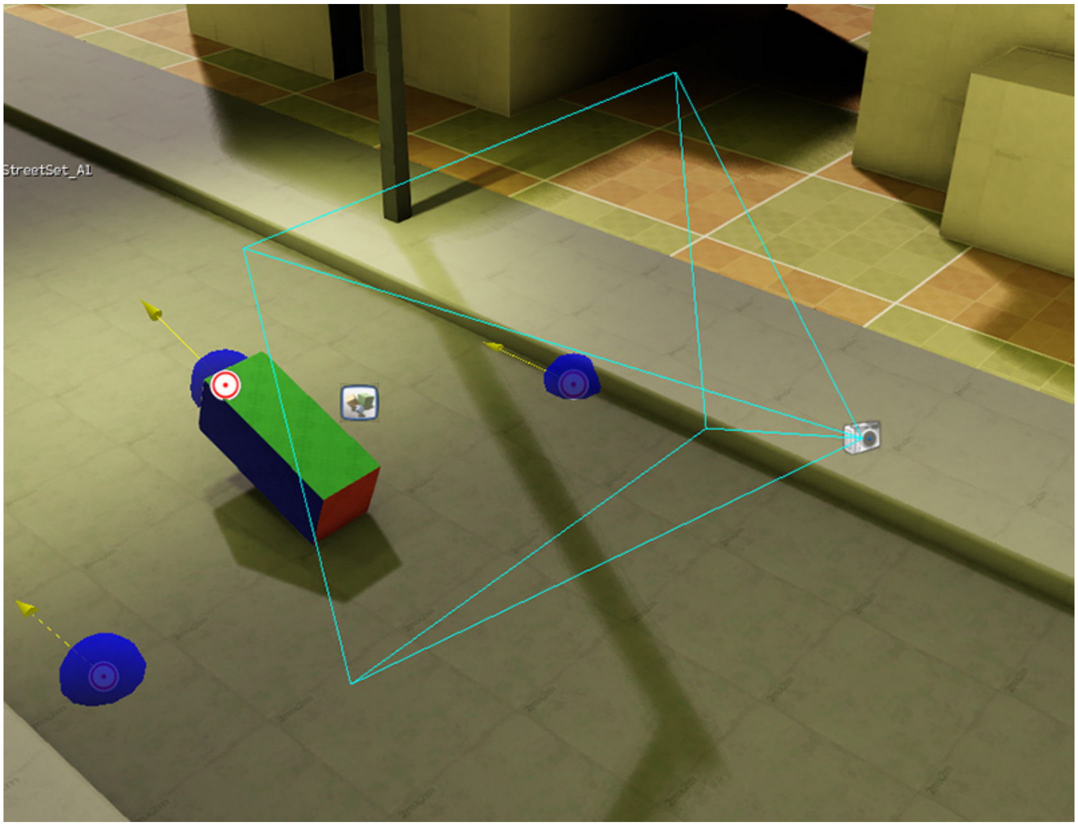
1. Position the view so that you can see the block at the bottom of the screen and the wall of physics blocks down the street. It should look something like the following:



2. In upper left corner of the **Perspective** viewport, right-click **Perspective** and click **Create Camera from Current View**. The current view from your **Perspective** viewport is now a fixed camera from that position.



3. Navigate away and notice the new camera entity.


StreetSet_A1

Creating a Flow Graph Script

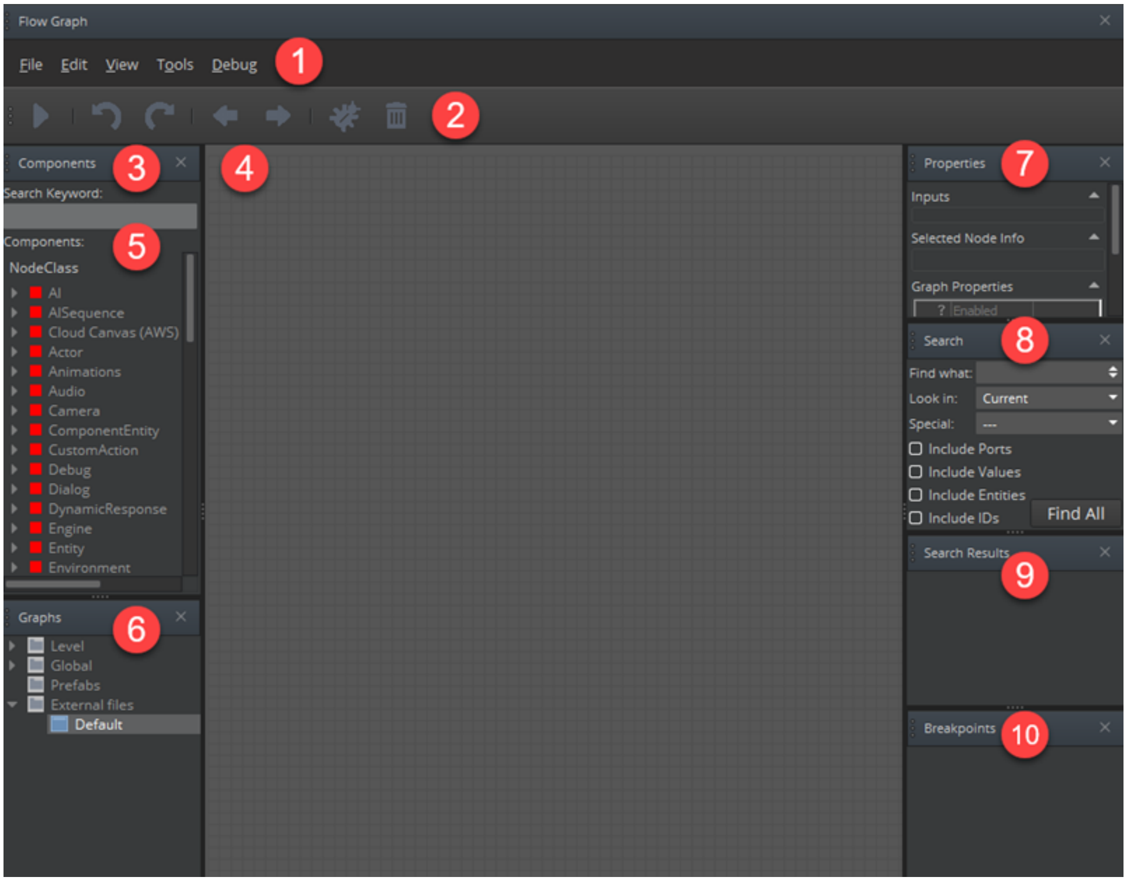
The **Flow Graph** editor is the Lumberyard visual game scripting tool. With the **Flow Graph** editor, you can script game play events using a visual node-based system that links game play actions to in-game events.

The nodes in flow graph scripts contain information about objects in the level or functions to call as they become activated through the script. You can obtain flow graph nodes from two places: From an entity object currently in the **Perspective** viewport, or from the list of components within the **Flow Graph** editor. In this tutorial, you'll use nodes from both places.

You can add multiple flow graphs to your level files. That way, you can separate script events based on location, function, or specific events. This gives you the flexibility to use whatever works best for the project you are creating.

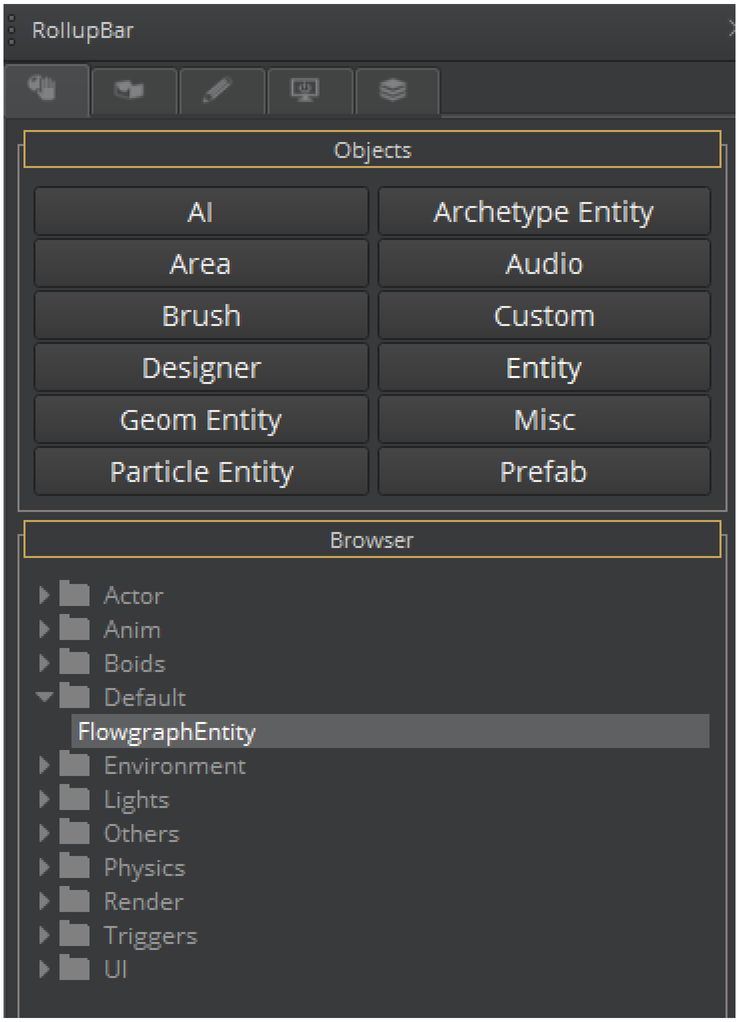
The **Flow Graph** editor has the following elements:

1. **Menu** – Tools to manage your flow graph files
2. **Toolbar** – Tools to undo, redo, step through, toggle debugging, and clear debugging
3. **Components Search Keyword** – Tool for finding available components
4. **Canvas** – Work area for building scripts
5. **Components** – List of nodes available for creating script events
6. **Graphs** – File tree access to flow graph files available in the level
7. **Properties** – Settings for selected component nodes
8. **Search** – Tool for finding specific nodes in canvas view
9. **Search Results** – Results of node search in canvas view
10. **Breakpoints** – Breakpoint management for debugging

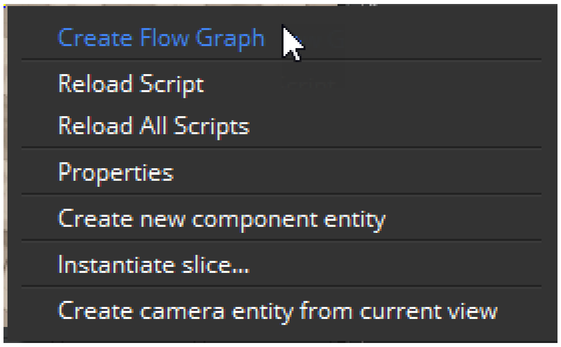


To script game events with Flow Graph

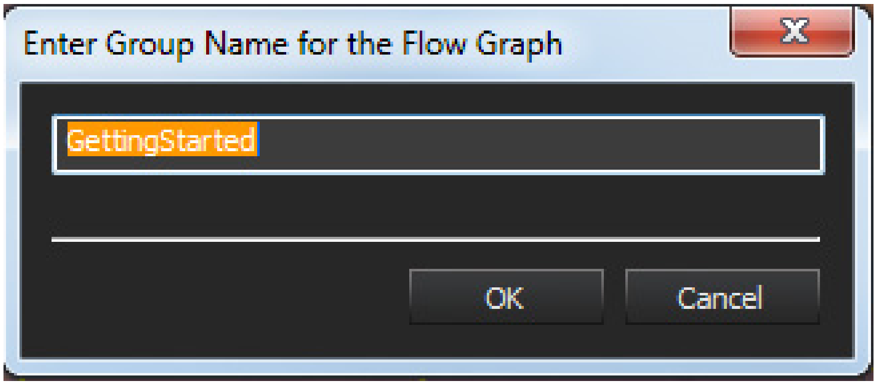
1. In the **Rollup Bar**, click the **Objects** tab and then click **Entity**. Under **Browser**, expand **Default**.
2. Drag **FlowgraphEntity** into the **Perspective** viewport.



3. Right-click the **FlowgraphEntity** and click **Create Flow Graph**.



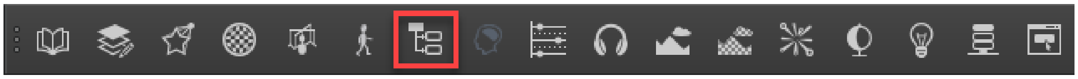
4. Name the flow graph group **`GettingStarted`** and click **OK**.



Enter Group Name for the Flow Graph

GettingStarted

5. From the editor toolbar, open **Flow Graph** editor. You can also open it from the main menu: **Tools**, **Flow Graph**.



6. Under **Graphs** in the file tree view, expand `Level Flowgraphs\Entities\GettingStarted` and select **FlowgraphEntity1**. This is the flow graph that you'll use to create the scripts.

   Level files can have multiple flow graphs. That way, you can separate script events based on location, function, or specific events. This gives you the flexibility to use whatever works best for the project you are creating.

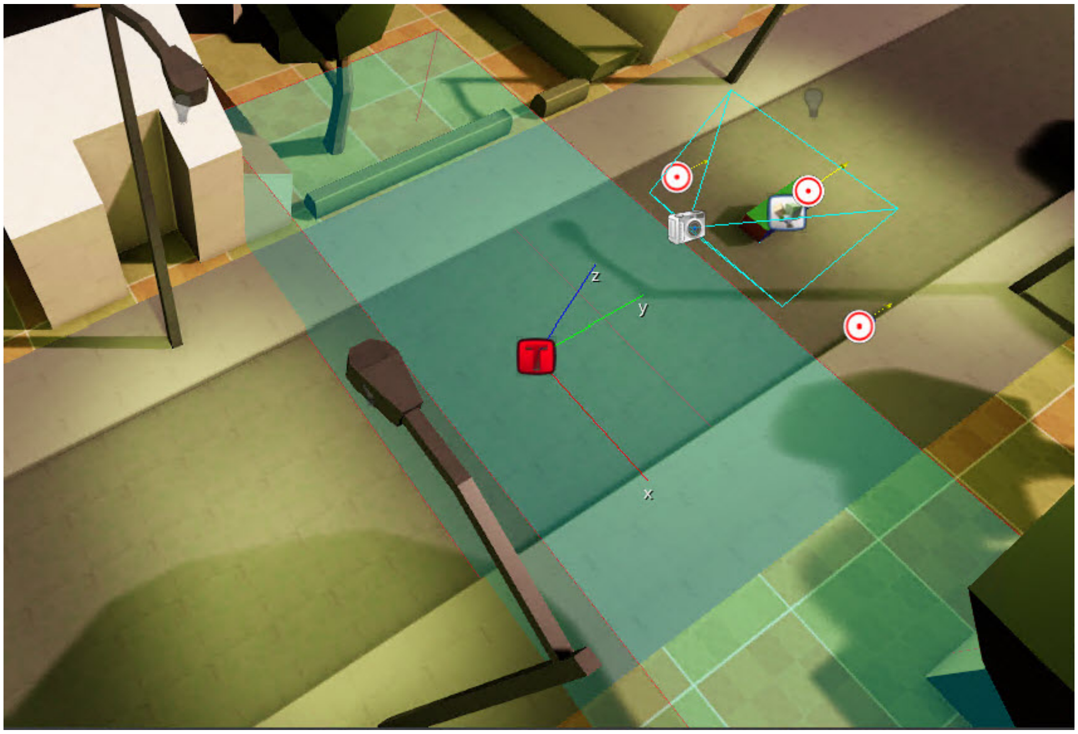
   Flow graph scripts are composed of nodes that contain information about objects in the level or functions to call when activated through the script. Flow graph nodes can be pulled from two places: From an entity object currently in the **Perspective** viewport, or from the list of components within the **Flow Graph** editor. In this tutorial, you use nodes from both places.

Scripting a Camera Event

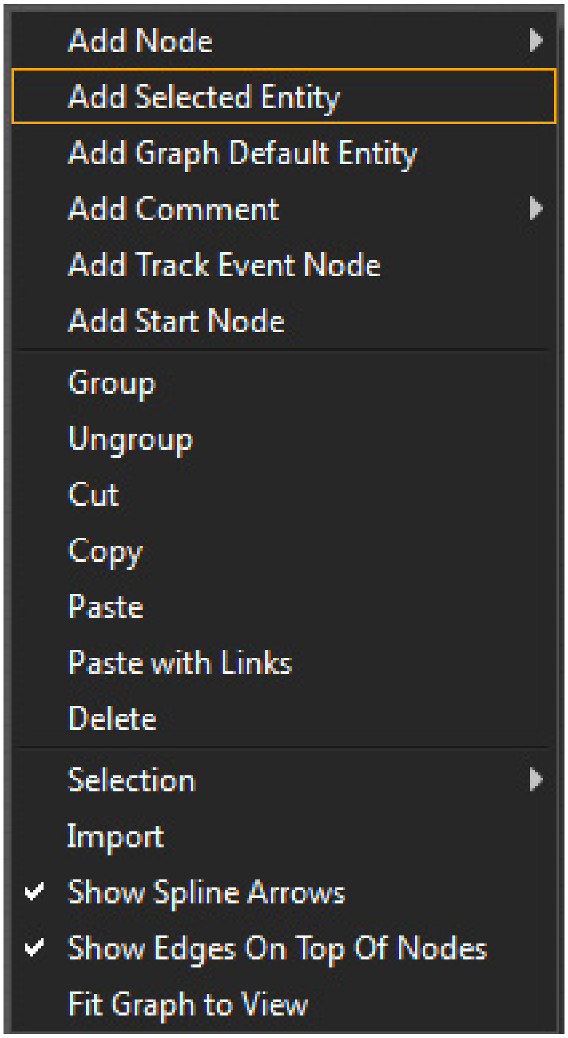
In this first script, you'll set up a trigger that activates when the player enters the trigger volume that you placed in Adding a Proximity Trigger (p. 106). This trigger switches from the default game play camera to the camera you placed in Adding a Camera (p. 108). You'll also add a trigger that switches back to the game play camera when the player leaves the trigger volume.

To set up a trigger

1. Select the proximity trigger (p. 106) that you placed earlier. (Look for and click an icon containing the letter T. If you don't see the icon, click the **H** icon on the **Perspective** viewport's title bar.)

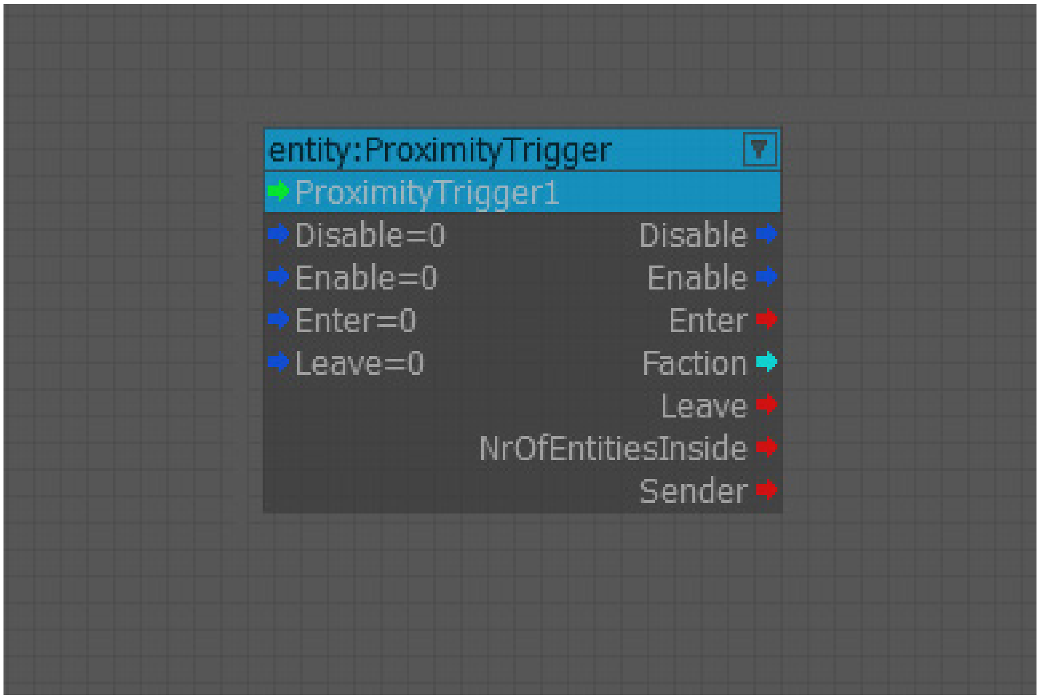


2. In the **Flow Graph** editor, right-click in the canvas view. Click **Add Selected Entity**.

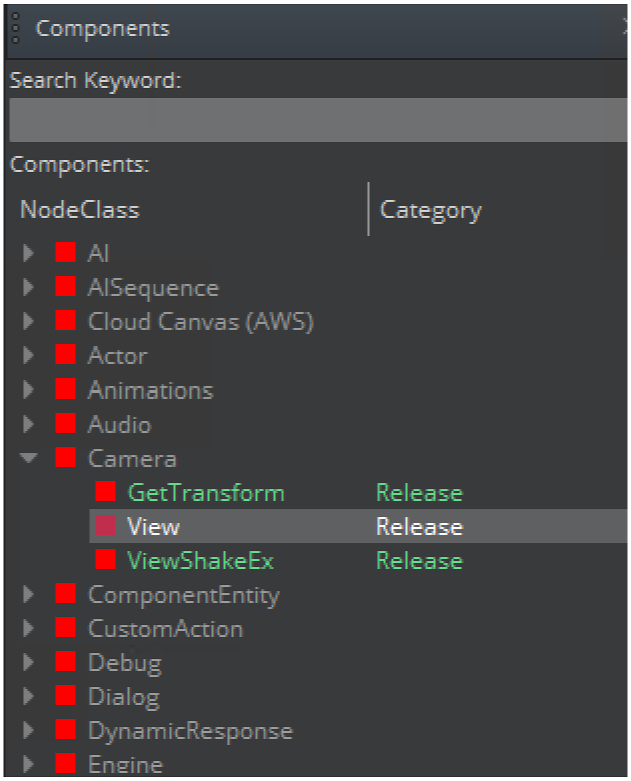


The node **entity:ProximityTrigger** appears in the canvas view.

This node represents the proximity trigger. You'll use the **Enter** and **Leave** outputs to determine what happens when a player enters or leaves the proximity trigger.

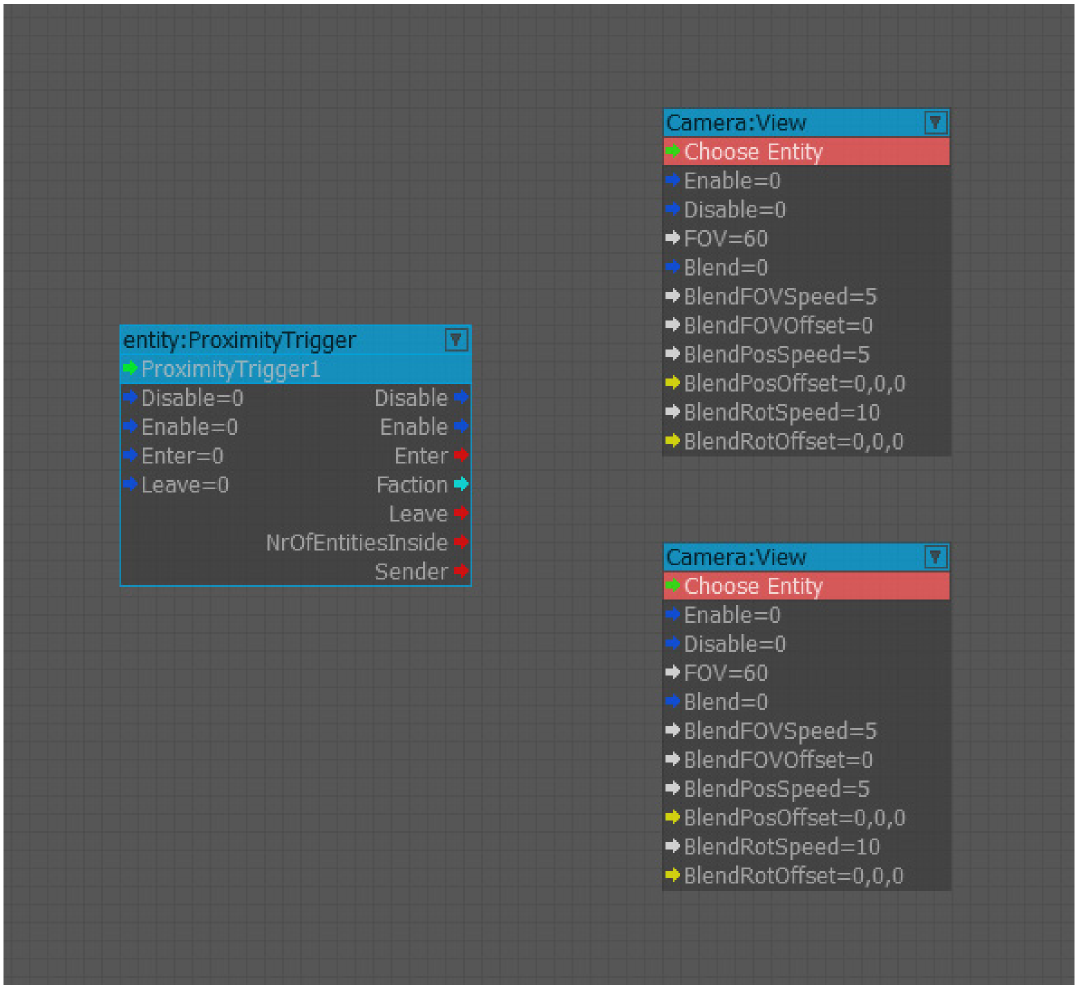


3. In the **Components** list view, expand **Camera**. Drag the node **View** into the canvas view.

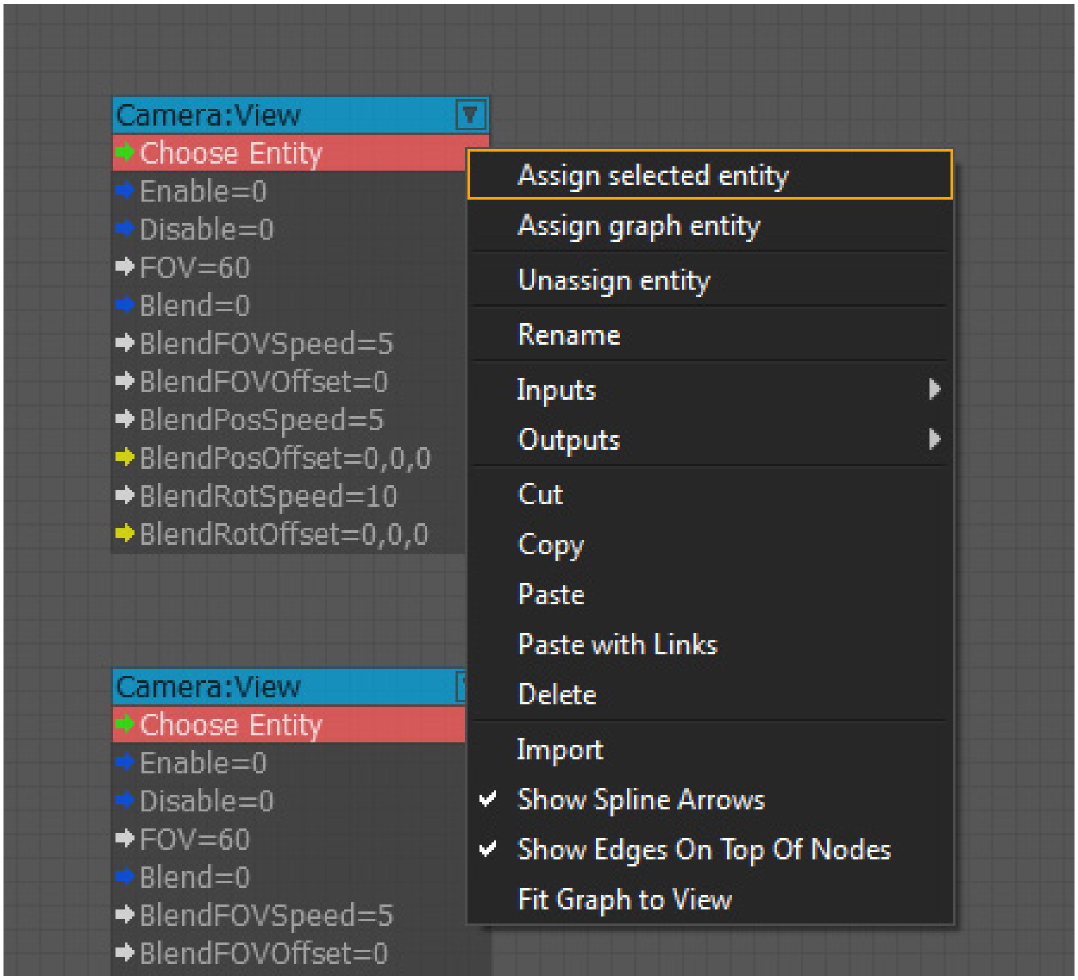


4. Repeat the previous step to drag another **View** node into the canvas view. The canvas should now look like this.

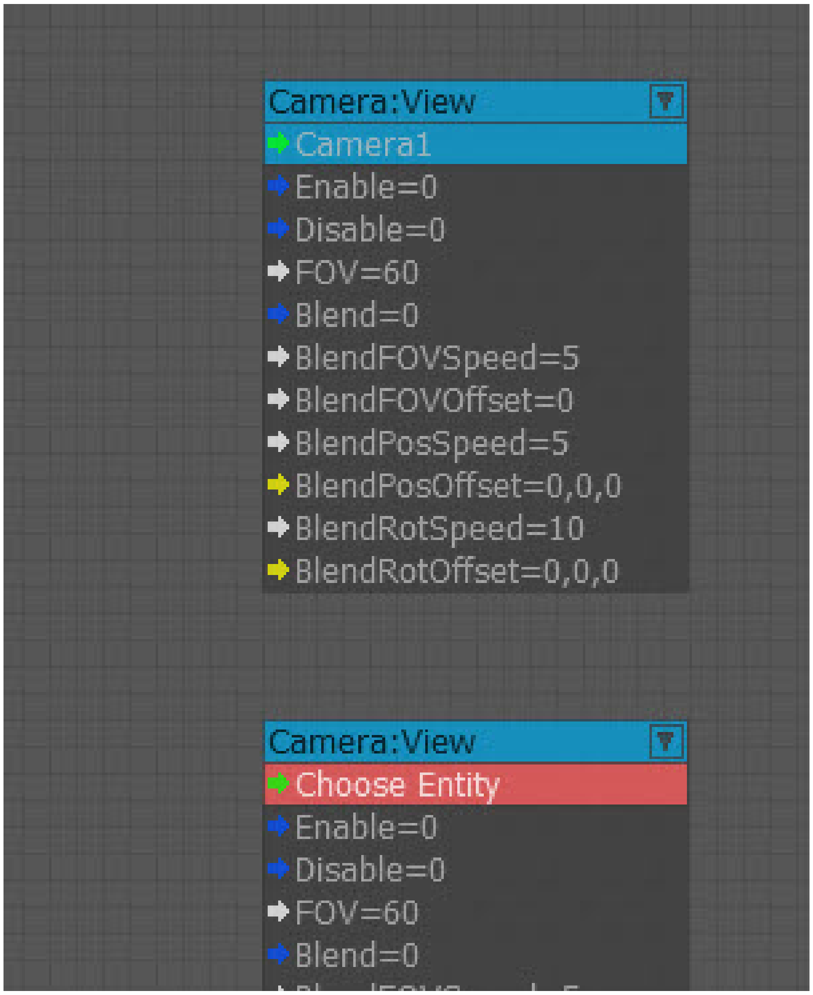
   These **Camera:View** nodes represent two different camera views.



5. Next, you'll assign a camera to each of the **Camera:View** nodes. To assign the first camera, go to the **Perspective** viewport and select the camera that you positioned in the middle of the street. This is most likely called **Camera1**.
6. In **Flow Graph** canvas, in the first **Camera:View** node, right-click **Choose Entity** and select **Assign selected entity**.



The red **Choose Entity** is replaced with a blue box containing the name of the assigned camera.

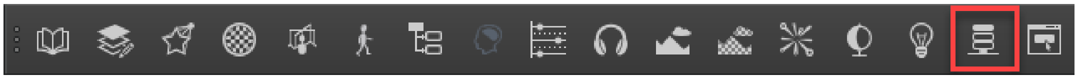

Camera:View
Camera1
Enable=0
Disable=0
FOV=60
Blend=0
BlendFOVSpeed=5
BlendFOVOffset=0
BlendPosSpeed=5
BlendPosOffset=0,0,0
BlendRotSpeed=10
BlendRotOffset=0,0,0

Camera:View
Choose Entity
Enable=0
Disable=0
FOV=60
Blend=0

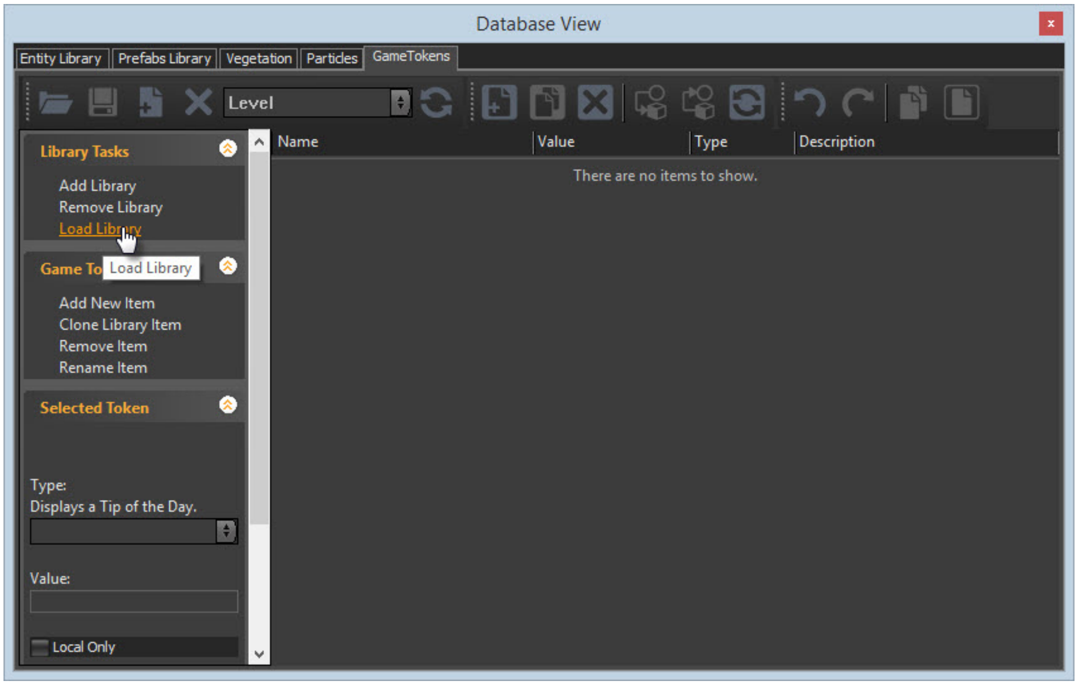
Now you assign the game play camera to the second camera node. To do this, you get the camera ID from the camera prefab using a game token.

Game tokens are values that exist outside of the existing flow graph script, or outside of the level currently in work.

7. From the editor toolbar, open the **Database View** editor.



8. At the top of the **Database View** editor, click the **Game Tokens** tab. In the **Library Tasks** area, click **Load Library**.



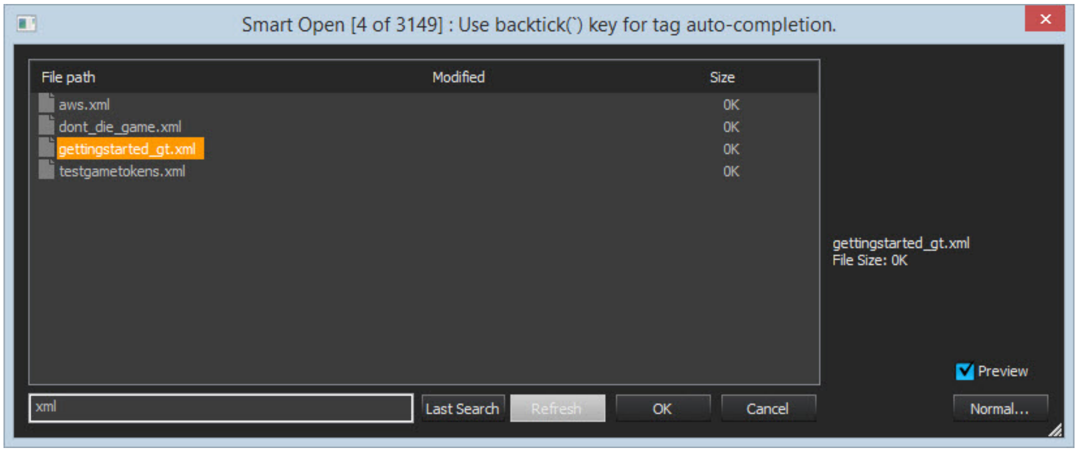
There are no items to show.

Type:
Displays a Tip of the Day.

Value:

9. Select `gettingstarted_gt.xml` and click **OK**.

   Note
   If you started with one of the Getting Started levels included with Lumberyard, you may get a "Loading Duplicate Library" error. If this occurs, close the error and continue to the next step, as the library is already loaded.


Smart Open [4 of 3149] : Use backtick(`) key for tag auto-completion.
Last Search
Refresh
OK
Cancel
Preview
Normal...

File path	Modified	Size
aws.xml		0K
dont_die_game.xml		0K
gettingstarted_gt.xml		0K
testgametokens.xml		0K

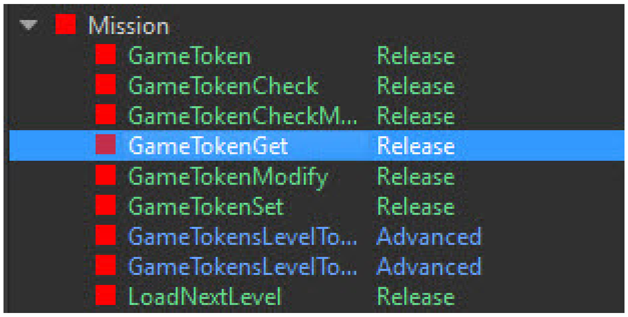
gettingstarted_gt.xml
File Size: 0K

☑ Preview

xml

   Close the **Database View** editor.

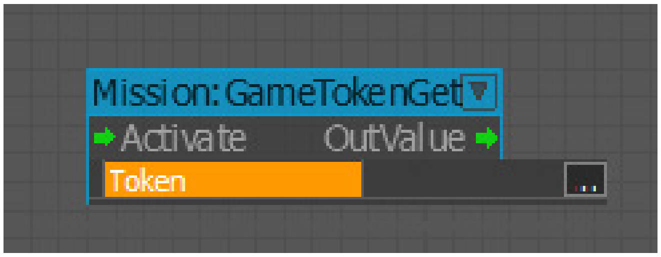
10. Return to the **Flow Graph** editor. In the **Components** list, expand **Mission**. Drag the node **GameTokenGet** onto the canvas.



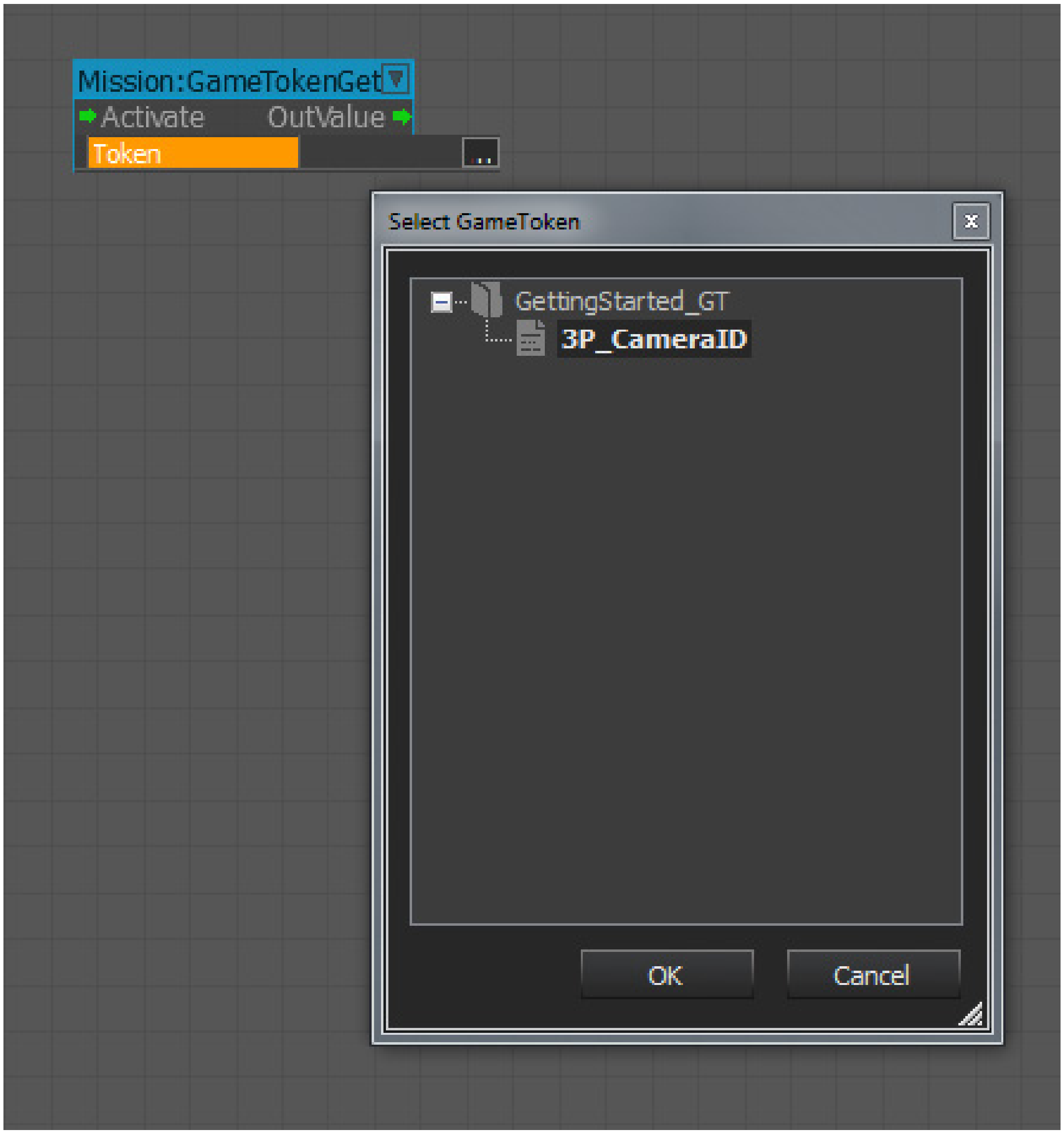
Mission

GameToken	Release
GameTokenCheck	Release
GameTokenCheckM...	Release
GameTokenGet	Release
GameTokenModify	Release
GameTokenSet	Release
GameTokensLevelTo...	Advanced
GameTokensLevelTo...	Advanced
LoadNextLevel	Release

11. Double-click **Token=**, and then click the box with the two dots. This opens the `GameToken` folder.

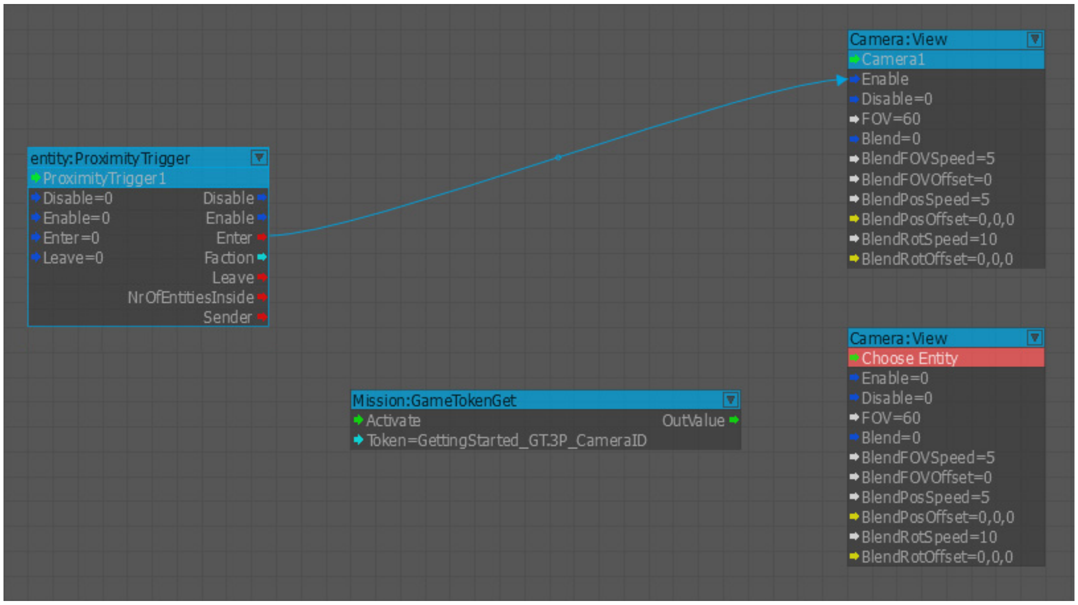


12. Expand `GettingStarted_GT` and select `3P_CameraID`. Click **OK**.



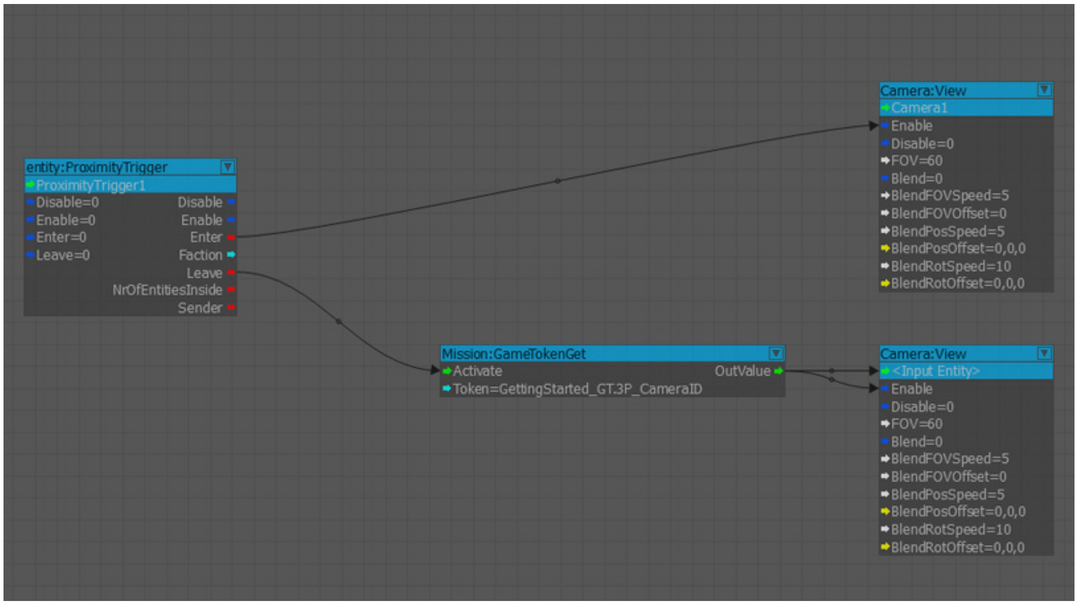
13. In the canvas view, from the **Entity:ProximityTrigger** node, drag the **Enter** output port to the first **Camera:View** node and attach it to the **Enable** input port.

    This creates a connection that links one node to the next. This particular connection tells the game to enable the camera you placed when the player enters the proximity trigger.



14. From the **entity:ProximityTrigger** node, drag the **Leave** output port to the **Mission:GameTokenGet** node's **Activate** input port. This tells the game to activate the game token, which is the third person camera view, when a player leaves the proximity trigger.
15. Drag the **Mission:GameTokenGet** node's **Outvalue** output port to the second **Camera:View** node's **Choose Entity** input port, which changes the port name to**<Input Entity>**. This tells the game that the second **Camera:View** node's input entity comes from the game token (third person camera view).
16. Drag another connection from the **Mission:GameTokenGet** node's **Outvalue** output port to the **Camera:View** node's **Enable** input port. This enables the third person game play camera when the player leaves the proximity trigger.

    The flow graph should look something like this:



17. With the trigger and cameras set up, switch to game mode (**Ctrl+G**) to test the functionality. Make sure the trigger switches to the first person camera when you enter the proximity trigger area and then, when you leave the proximity trigger, returns to the game play camera.

Creating a Mover

A flow graph can have multiple script events within the same canvas. If you like, you can always create a new script file, which may work best when organizing your level. For this tutorial, keep all the scripts in the same flow graph file.

You'll create a script that uses the keyboard to move the block between the two tag points.

In this procedure, you'll first place all the nodes on your flow graph canvas.

Next, you'll make the connections between the nodes.

Last, you'll enter values for the relevant fields in the nodes.

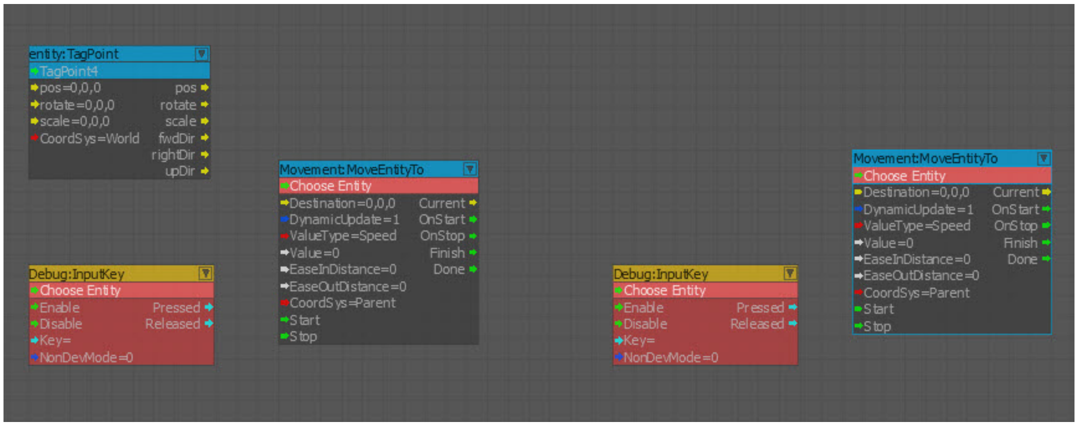
To create a script to move the block between tag points

1. In the same flow graph script file, expose a blank area of the canvas to build your next script node set. Use your mouse wheel to zoom out, or drag with the right mouse button to pan around the canvas.
2. In the **Components** list, expand **Movement** and drag two of the **MoveEntityTo** nodes into the canvas view.

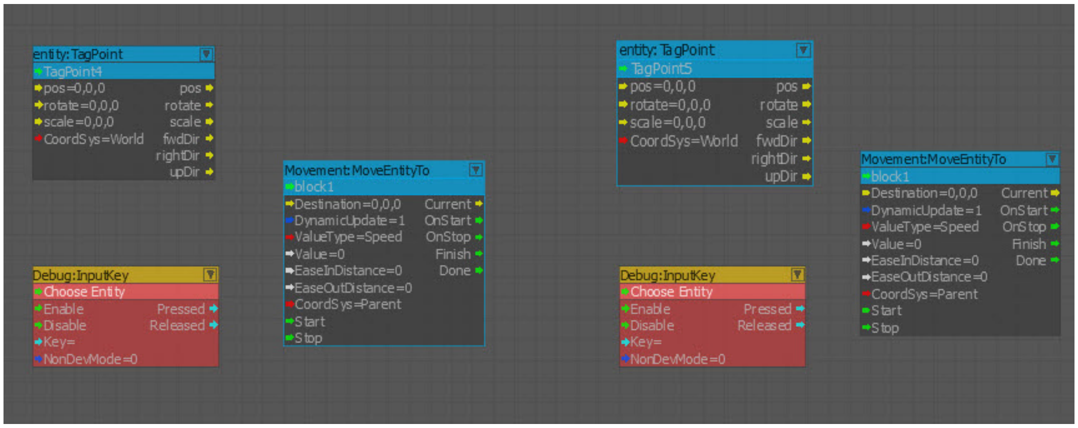
   The **MoveEntityTo** node enables you to specify movement of the selected entity (you'll enter the **block** entity in a later step).
3. In the **Components** list, expand **Debug** and drag two of the **InputKey** nodes into the canvas view.

   The **InputKey** node enables you to specify the key on the keyboard that triggers outputs when that key is pressed and released (you'll enter these values in a later step).
4. In the **Perspective** viewport, select the tag point entity on the left side of the street (assuming you are facing the block wall). Return to the **Flow Graph** editor and right-click in a blank area and click **Add Selected Entity**. Position the nodes so they look like the picture in the next step.
5. Select the second tag point entity on the right side of the street and add it to the canvas view; place it next to the **Movement:MoveEntityTo** node on the right.

   The **entity:TagPoint** nodes you just placed will be used as destinations for the **Movement:MoveEntityTo** nodes (you'll make the connections in a later step).



6. Select the block between the two tagpoints. On each of the **Movement:MoveEntityTo** nodes, on the **Choose Entity** line, assign the selected block.



7. From the **entity:Tagpoint** node on the left, drag the **pos** output to **Movement:MoveEntityTo** node's **Destination** input.

   This tells the game to set the specified tag point (in **entity:Tagpoint**) as the destination for the block entity when the **Movement:MoveEntityTo** node's **Start** input is activated.

8. Make the same connection between the other **entity:Tagpoint** and **Movement:MoveEntityTo** nodes on the right.

9. Drag the left **Debug:InputKey** node's **Pressed** output to **Movement:MoveEntityTo** node's **Start** input.

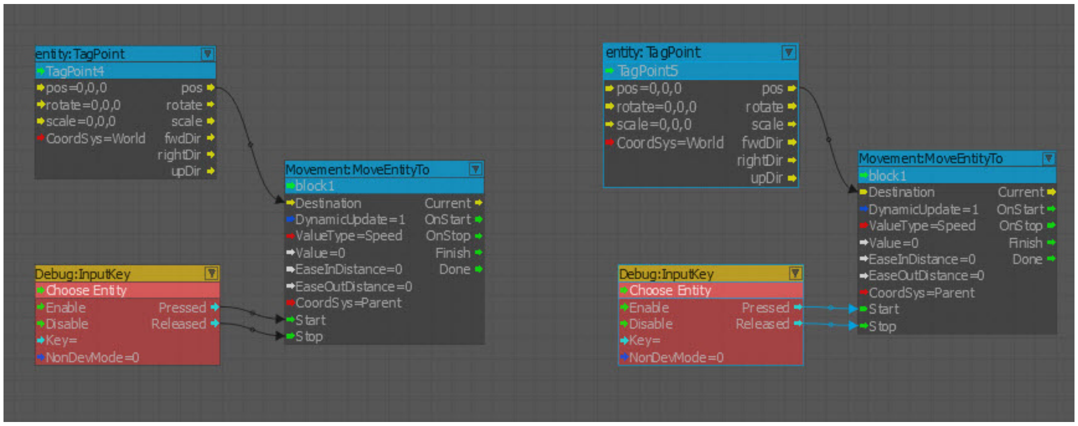
   This tells the game to start moving the block towards the tag point destination when the key is pressed.

10. On the same set of nodes, connect the **Released** output to the **Stop** input.

    This tells the game to stop moving the block towards the tag point destination when the key is released.

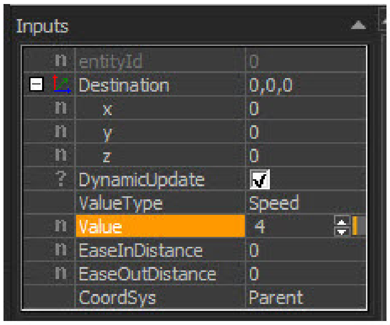
11. Make the same connection between the other **InputKey** and **MoveEntityTo** nodes on the right.

    Your nodes should now look like this:



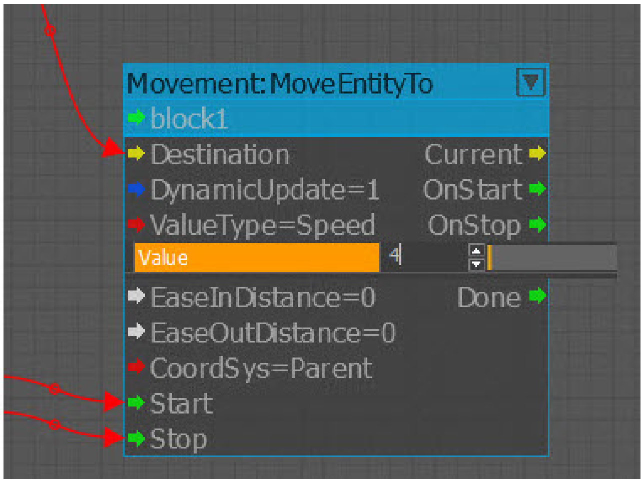
12. Select one of the **MoveEntityTo** nodes. In the **Inputs** section of **Properties**, change **Value** = **4**. You can can also double-click a property directly in the node and enter the value there.

    With the node's **ValueType** = **Speed**, setting the value to 4 sets the speed at which the block moves when activated.



Inputs

n	entityId	0
⊟	Destination	0,0,0
n	x	0
n	y	0
n	z	0
?	DynamicUpdate	☑
	ValueType	Speed
n	Value	4
n	EaseInDistance	0
n	EaseOutDistance	0
	CoordSys	Parent



13. Select the other **MoveEntityTo** node and change **Value** = **4** as well.
14. Select the **Debug:InputKey** node on the left. For its **Key** property, type the letter `j`.

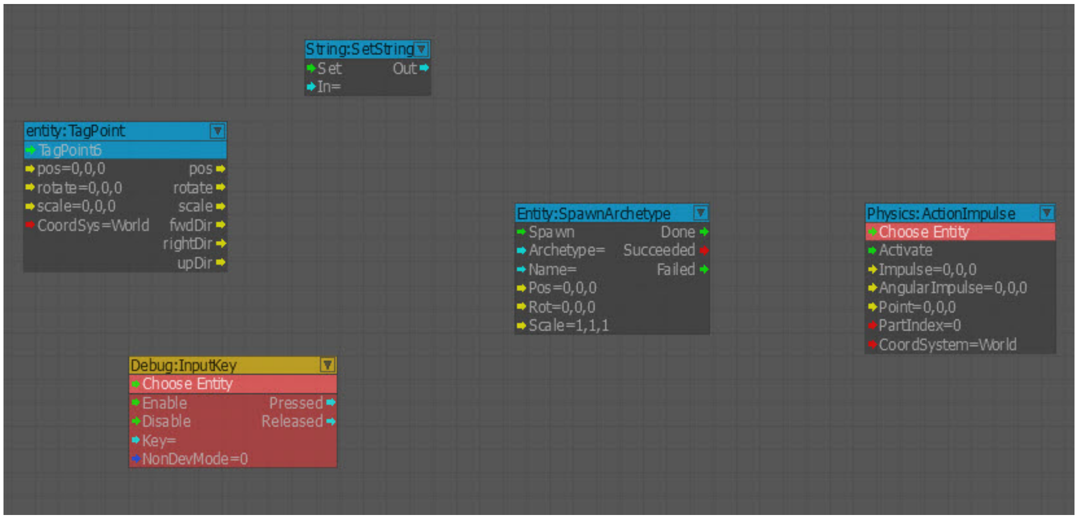
    This sets the **MoveEntityTo** to start and stop movement when `j` is pressed on the keyboard.
15. Select the **Debug:InputKey** node on the right, set its **Key** = `l` (lower case L).
16. Run the level by switching to game mode (**Ctrl+G**) and test that the block moves left and right with the keyboard inputs of `j` and `l`. If necessary, move (with the **Move** tool) the position of the tag points to which the block moves.

Spawning an Object

In this script, you'll *spawn* a physics ball—that is, cause it to appear in a level—so that the ball shoots out of the designated spawn point, which is the tag point linked to the block cannon.

To spawn a physics objects

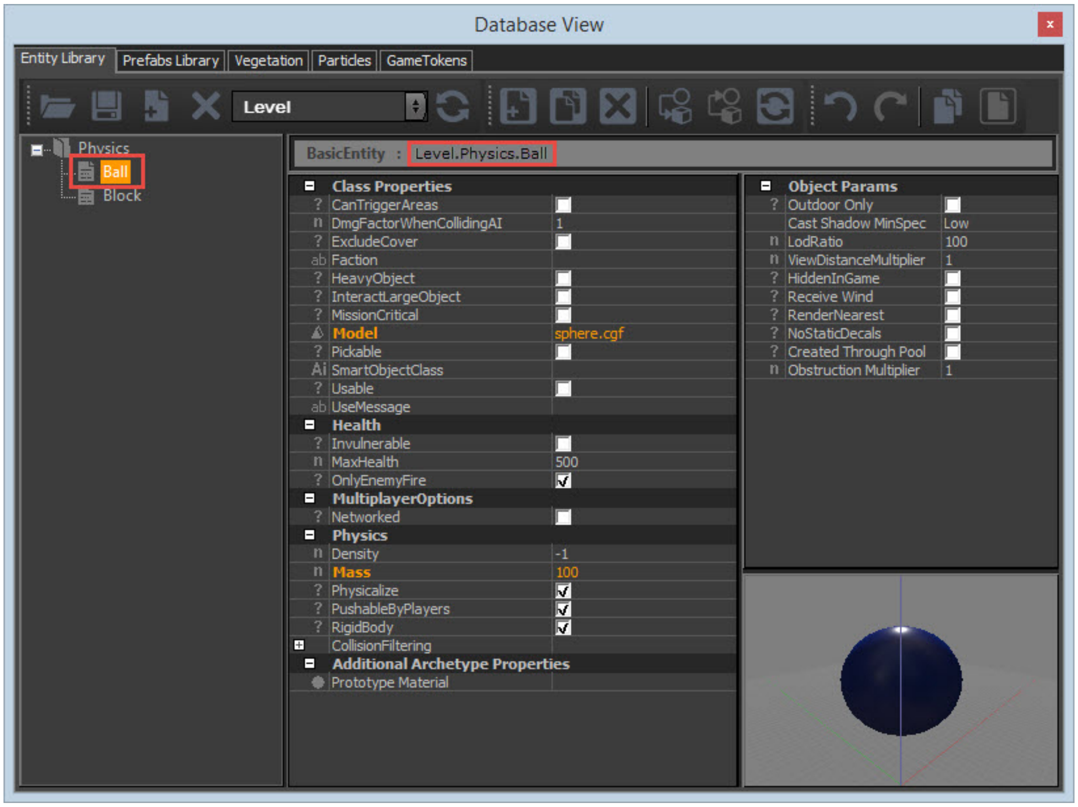
1. In the **Flow Graph** editor **Components** list, do the following:
   - Expand **Entity** and drag **SpawnArchetype** onto the canvas. This node will contain the name and position of the archetype object to spawn.
   - Expand **String** and drag **SetString** onto the canvas. This node enables you to set a text string for the name of the archetype object to spawn.
   - Expand **Physics** and drag **ActionImpulse** onto the canvas. This node sets a velocity for the physics ball after it is spawned.
   - Expand **Debug** and drag **InputKey** onto the canvas. In this node, you'll specify a keyboard input that triggers spawning of the ball.
2. Select the tag point linked to the block in the scene. Add this entity to the canvas view.
3. Arrange the nodes to look like the following image.



4. Drag the **TagPoint** node's **pos** output to the **SpawnArchetype** node's **Pos** input.

   This sets the position at which the physics ball is spawned to the position of the tag point linked to the block.

5. Open the **Database View** editor. Click the **Entity Library** tab, and then expand the **Physics** folder. Select **Ball**.



BasicEntity : Level.Physics.Ball

Class Properties	
CanTriggerAreas	☐
DmgFactorWhenCollidingAI	1
ExcludeCover	☐
Faction	
HeavyObject	☐
InteractLargeObject	☐
MissionCritical	☐
Model	sphere.cgf
Pickable	☐
SmartObjectClass	
Usable	☐
UseMessage	
Health	
Invulnerable	☐
MaxHealth	500
OnlyEnemyFire	☑
MultiplayerOptions	
Networked	☐
Physics	
Density	-1
Mass	100
Physicalize	☑
PushableByPlayers	☑
RigidBody	☑
CollisionFiltering	
Additional Archetype Properties	
Prototype Material	

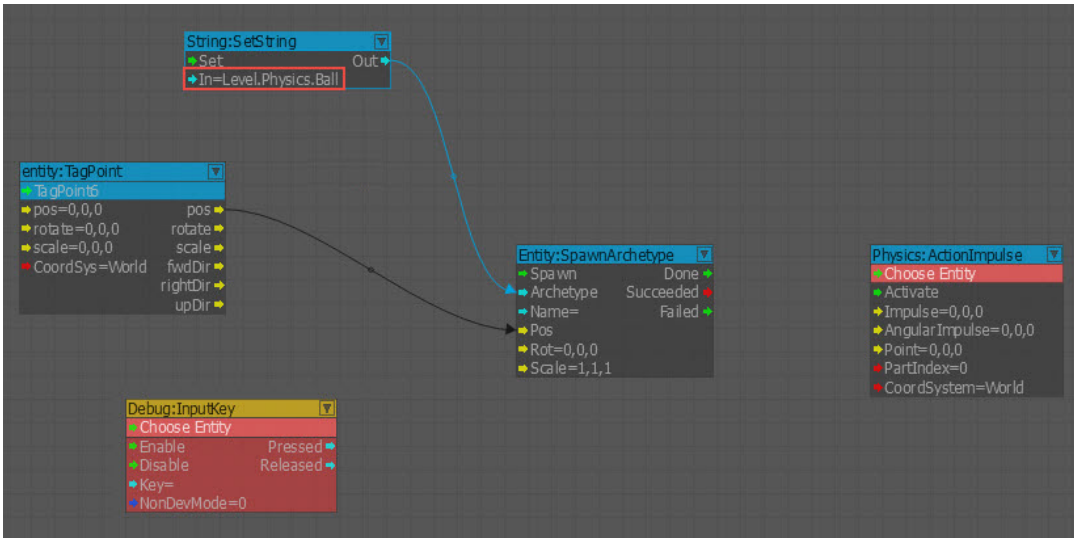
Object Params	
Outdoor Only	☐
Cast Shadow MinSpec	Low
LodRatio	100
ViewDistanceMultiplier	1
HiddenInGame	☐
Receive Wind	☐
RenderNearest	☐
NoStaticDecals	☐
Created Through Pool	☐
Obstruction Multiplier	1

6. Take note of the string: **Level.Physics.Ball**. This identifies the ball coming from the **Level** file, within the **Physics** folder, and having the name **Ball**.

7. In the **Flow Graph** canvas view, select the **SetString** node. Double-click the **In** property and type the string: `Level.Physics.Ball`.

8. Drag the **SetString** node's **Out** output to the **SpawnArchetype** node's **Archetype** input.

This sets the name of the archetype object to spawn.



9. Set the **Inputkey** node's property **Key**= `i`.

10. Drag the **InputKey** node's **Pressed** output to the **SpawnArchetype** node's **Spawn** input.

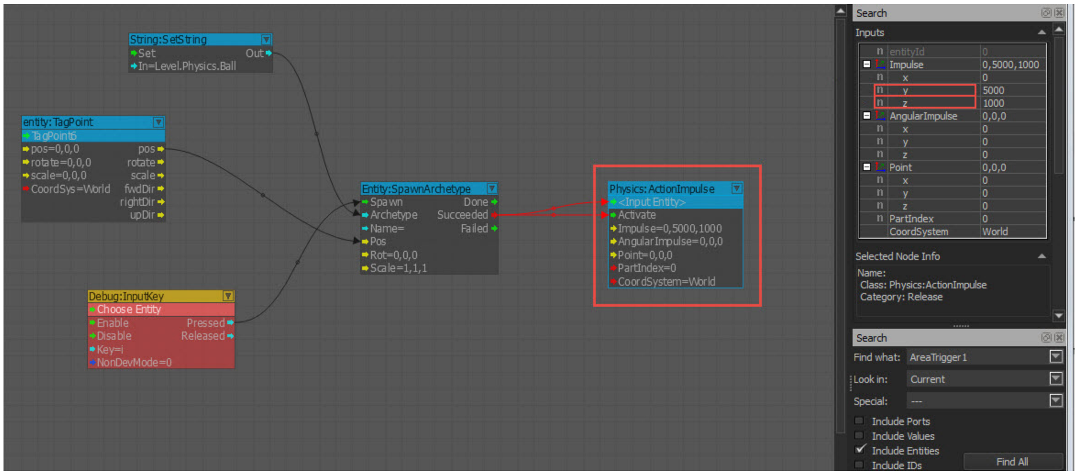
    This tells the game to spawn the ball when the `i` key is pressed.

11. Drag the **SpawnArchetype** node's **Succeeded** output to the **ActionImpulse** node's **Activate** input as well as its entity input, changing it to read **Input Entity**.

    This tells the game to apply an impulse force, or velocity, to the spawned ball entity.

12. In the **ActionImpulse** node, set the **Impulse** values to **X**=**`0`**, **Y**=**`5000`**, **Z**=**`1000`**. If X is forward in your level, swap the X and Y values.

    This sets an upward (**Z**) velocity of **`1000`**, forward (**Y**) velocity of **`5000`**, and sideways (**X**) velocity of **`0`**.



entityId	0
Impulse	0,5000,1000
x	0
y	5000
z	1000
AngularImpulse	0,0,0
x	0
y	0
z	0
Point	0,0,0
x	0
y	0
z	0
PartIndex	0
CoordSystem	World

Selected Node Info

Name:
Class: Physics:ActionImpulse
Category: Release

Search

Find what: AreaTrigger1
Look in: Current
Special: ---

□ Include Ports
□ Include Values
☑ Include Entities
□ Include IDs

13. Test the level. You should be able to move the block from side to side and launch a sphere at the block wall using the `i` key.

Setting Impulse Values

In this script, you'll modify the flow graph so that you can shoot the physics ball at different velocities based on how long you hold the keyboard input.

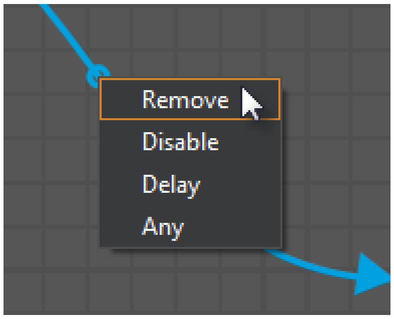
To create a script that determines velocity of the spawned physics ball based on user input

1. In the **Flow Graph** editor **Components** list, expand **Interpolate** and drag **Int** onto the canvas.

   This node interprets the length of the keyboard press.
2. Expand **Vec3** and drag **ToVec3** onto the canvas.

   This node receives the keyboard press length information and inserts it as the **Y** value in an **XYZ** value scheme for the ball's velocity. You'll manually specify the **X** and **Z** values.
3. In the **Debug:Inputkey** (with **Key**=**i**) node, disconnect the **Pressed** output from the **SpawnArchetype** node's **Spawn** input.

   To disconnect, click on one of the nodes. This selects the connecting arrow(s). Right-click the small circle on the connection and click **Remove**.

   In the same debug node, also do the following:

   - Drag the **Released** output to the **SpawnArchetype** node's **Spawn** input. This causes the ball to be spawned when the player releases the key.
   - Drag the **Pressed** output to the **Interpolate:Int** node's **Start** input. This tells the **Interpolate:Int** node to start gathering the length of the keyboard press.
   - Drag the **Released** output to the **Interpolate:Int** node's **Stop** input. This tells the **Interpolate:Int** node to stop gathering the length of the keyboard press.
4. In the **Interpolate:Int** node, change the **EndValue** = **6000**. Also change the **Time** = **2**.

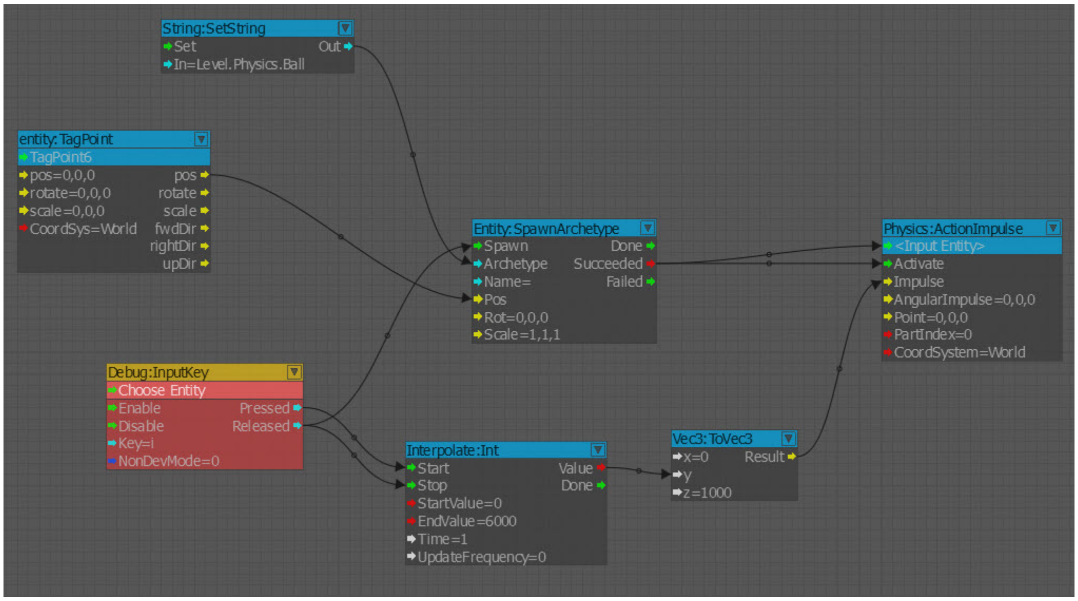
   This sets the maximum impulse value to **6000** within a **2** second time frame.
5. In the **Vec3:ToVec3** node, change **Z**=**1000**.

   This manually sets the **Z** value in the **XYZ** value scheme to **1000**.
6. Drag the **Interpolate:Int** node's **Value** output to the **Vec3:ToVec3** node's **Y** input. (Or the X input, if that is the direction your street is oriented)

   This sets the **Y** value in the **XYZ** scheme to the value of the length of the keyboard press.
7. Drag the **Vec3:ToVec3** node's **Result** output to the **ActionImpulse** node's **Impulse** input.

   This passes the **XYZ** value to the **ActionImpulse** node to use as the ball's velocity.

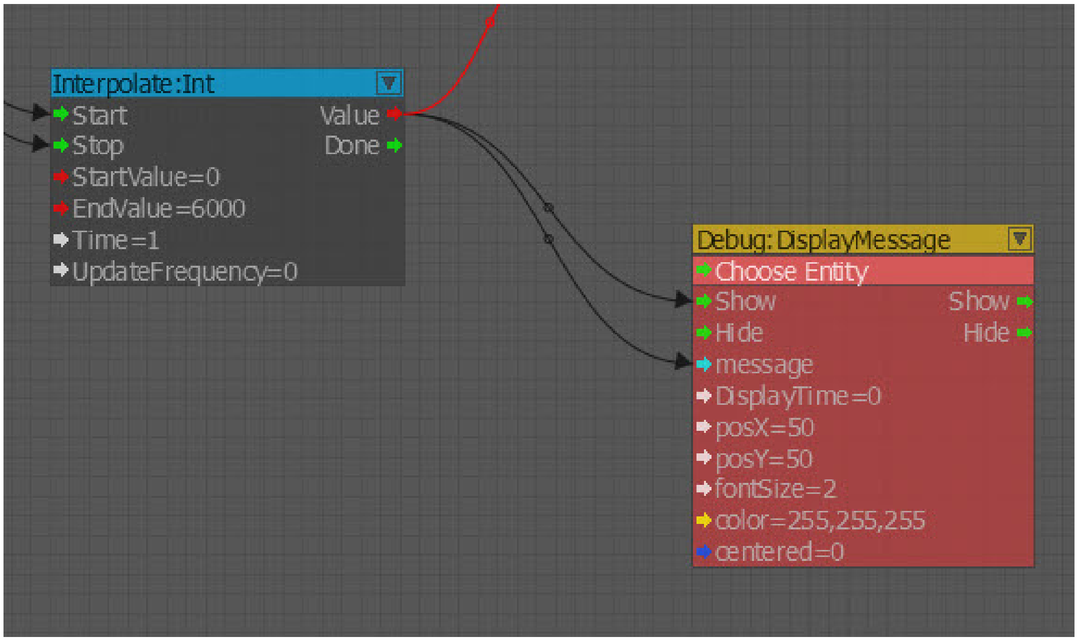
   The flow graph script should now look like this:



With these script modifications, you have changed how much impulse is applied to the spawn objects based on how long the `i` key is held down. It also sets the maximum impulse output to **6000**.

As the script is currently, you will not see an output value displayed when you release the `i` key. The next few steps enable the display of that output value.

8. In the **Components** list, expand **Debug** and drag **DisplayMessage** onto the canvas.
9. Drag the **Interpolate:Int** node's **Value** output to the **Debug:Displaymessage** node's **Show** and **message** inputs.



10. Run the level again and test the inputs. You see that the length of input affects the ball's spawn velocity. The impulse output value is also displayed in the upper left of the screen.

Using Any Nodes

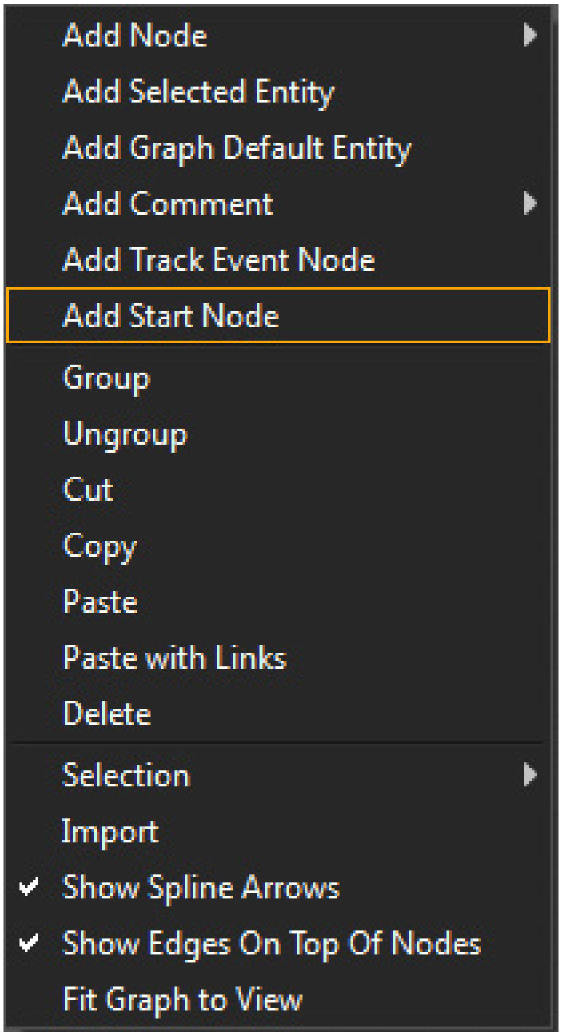
Next, you'll create a script that disables the cannon and its inputs until it has been triggered with the camera switch.

To do that, you'll use the **Any** node. This node enables multiple inputs to be routed into a single output. Because any given input port can accept only one input, using the **Any** node allows you to collate multiple inputs into a single output.

To disable the cannon until triggered

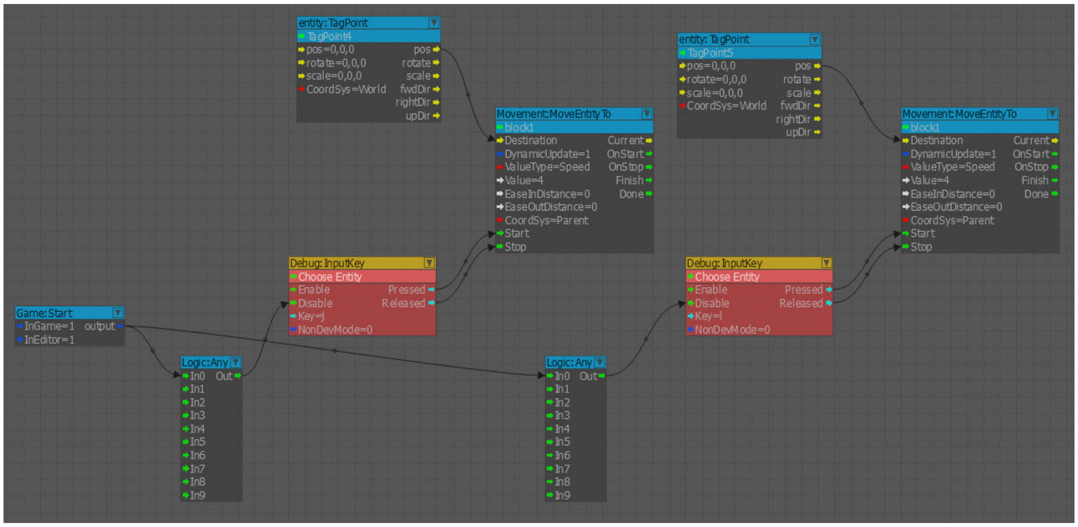
1. In the **Flow Graph** editor **Components** list, expand **Logic** and drag the **Any** node onto the canvas. Repeat two more times for a total of three **Any** nodes.

   Place an **Any** node by each of the three **Debug:InputKey** nodes.
2. Next to the **Debug:InputKeys** (the ones that control the block moving left and right), add a **Game:Start** node. To do this, right-click the canvas and choose **Add Start Node**.

3. Drag the **Game:Start** node's **output** output to both **Logic:Any** nodes' **In0** inputs.
4. Drag the first **Any** node's **Out** output to the **Debug:InputKey** node's **Disable** input. Do the same for the second **Logic:Any** node, as shown in the following image.

   This disables the input keys when the game is started.

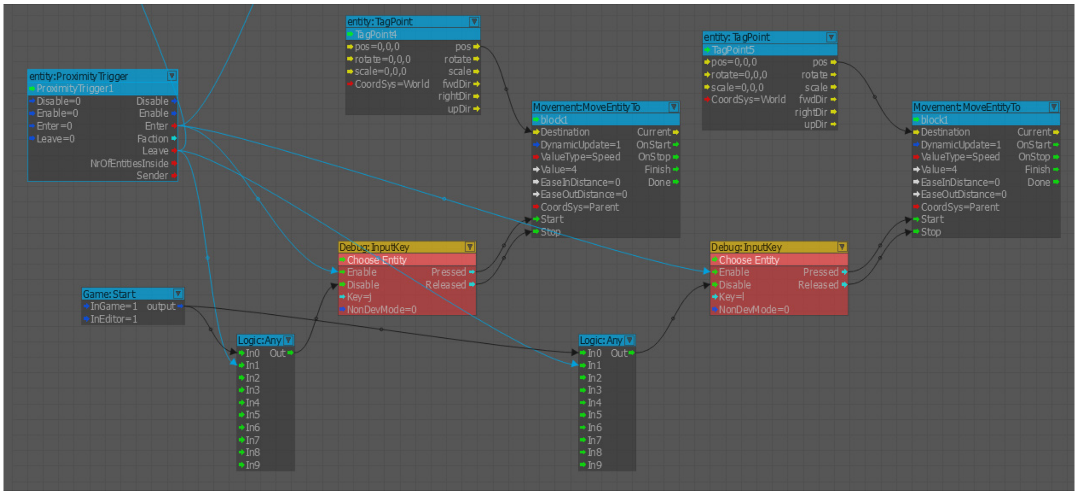


5. Drag the **entity:ProximityTrigger** node's **Enter** output to each of the two **Debug:InputKey** nodes' **Enable** inputs.

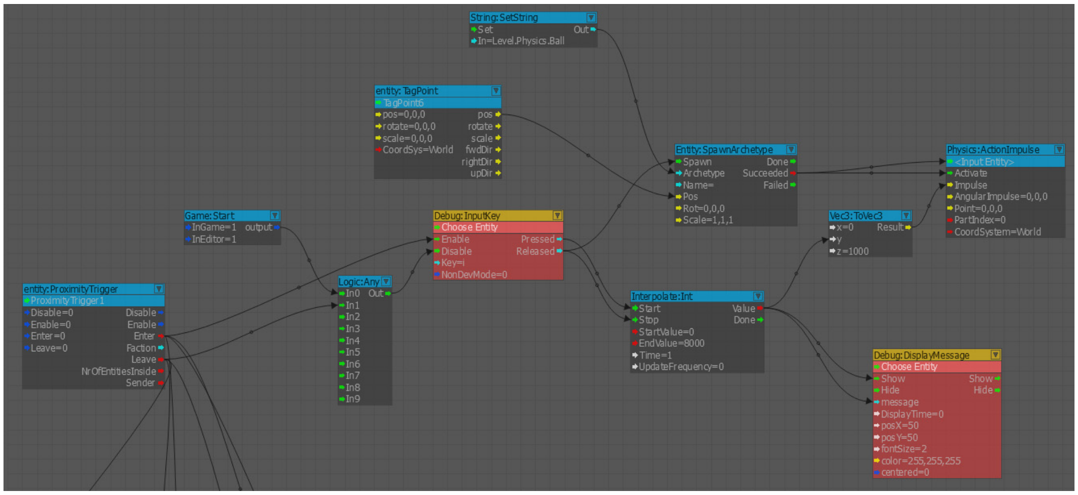
   This enables the input keys when the player enters the proximity trigger area.

6. Drag the **ProximityTrigger** node's **Leave** output each of the two **Logic:Any** node's **In1** inputs.

   This disables the input keys when the player leaves the proximity trigger area.



7. Apply the same connections for the **Debug:InputKey** node that controls the launching of the physics ball.

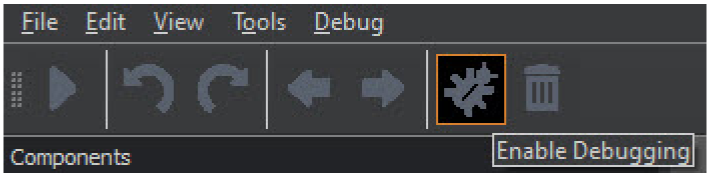

String:SetString
Set
Out
In=Level.Physics.Ball
entity: TagPoint
TagPoint6
pos=0,0,0
pos
rotate=0,0,0
rotate
scale=0,0,0
scale
CoordSys=World
fwdDir
rightDir
upDir
Entity:SpawnArchetype
Spawn
Done
Archetype
Succeeded
Name=
Failed
Pos
Rot=0,0,0
Scale=1,1,1
Physics:ActionImpulse
<Input Entity>
Activate
Impulse
AngularImpulse=0,0,0
Point=0,0,0
PartIndex=0
CoordSystem=World
Game:Start
InGame=1
output
InEditor=1
Debug:InputKey
Choose Entity
Enable
Pressed
Disable
Released
Key=i
NonDevMode=0
Vec3:ToVec3
x=0
Result
y
z=1000
entity:ProximityTrigger
ProximityTrigger1
Disable=0
Disable
Enable=0
Enable
Enter=0
Enter
Leave=0
Faction
Leave
NrOfEntitiesInside
Sender
Logic:Any
In0
Out
In1
In2
In3
In4
In5
In6
In7
In8
In9
Interpolate:Int
Start
Value
Stop
Done
StartValue=0
EndValue=8000
Time=1
UpdateFrequency=0
Debug:DisplayMessage
Choose Entity
Show
Show
Hide
Hide
message
DisplayTime=0
posX=50
posY=50
fontSize=2
color=255,255,255
centered=0

Using Debug View

If your script does not work as you expect, you can turn on **Debug** in the Flow Graph editor. Debug is a helpful tool to detect where your logic has failed.

To turn on Debug:

1. In the **Flow Graph** editor's toolbar, click **Debug**.



2. Move the flow graph window to the side (but still visible).
3. Run the level (**Ctrl+G**). Move around in the level and trigger the camera switch. Move and shoot from the block. Notice how each action in the level is displayed in the flow graph events.
4. Exit the level. Click the **Trash** icon (next to the **Debug** icon) to clear the debug events.
5. Save your file.

You have completed the flow graph scripting of your level.

You now have a fully working level that allows you to move a third-person character around an environment that you created, trigger a camera that switches to a new view, and turn on the control of a box cannon that shoots a ball at a wall of blocks.

Lumberyard Blog, Forums, and Feedback

As we continue to improve Lumberyard, we want to thank everyone in our developer community. Without your participation in the forums, your messages, and your bug reports, Lumberyard wouldn't be as strong as it is.

- Keep sending your feedback to `<lumberyard-feedback@amazon.com>`.
- If you haven't spoken up on the forums yet, we would love to have you.
- You can also keep up with new changes on our blog and leave comments to let us know what you think.