Amazon Lumberyard Editor

Getting Started Guide

Version 1.1

Amazon Lumberyard Editor

Table of Contents

Introduction

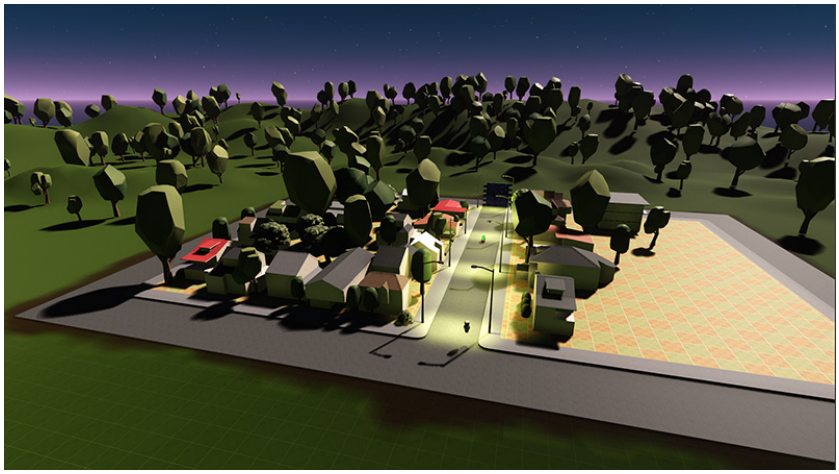
Welcome to the *Getting Started Guide for Amazon Lumberyard Editor*. Lumberyard is a free, cross-platform, 3D game engine for creating high quality games, connecting your games to the vast AWS cloud computing and storage, and engaging fans on Twitch.

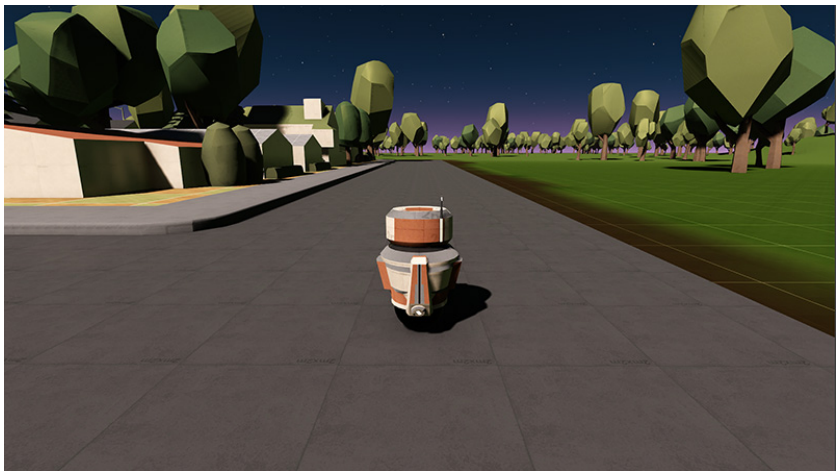
This Getting Started Guide familiarizes you with the basics of Lumberyard Editor. You'll be guided through nine tutorials that describe the most commonly used tools and features of this editor. This tutorial assumes you have some previous experience with game engines or 3D modeling tools. During this 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.

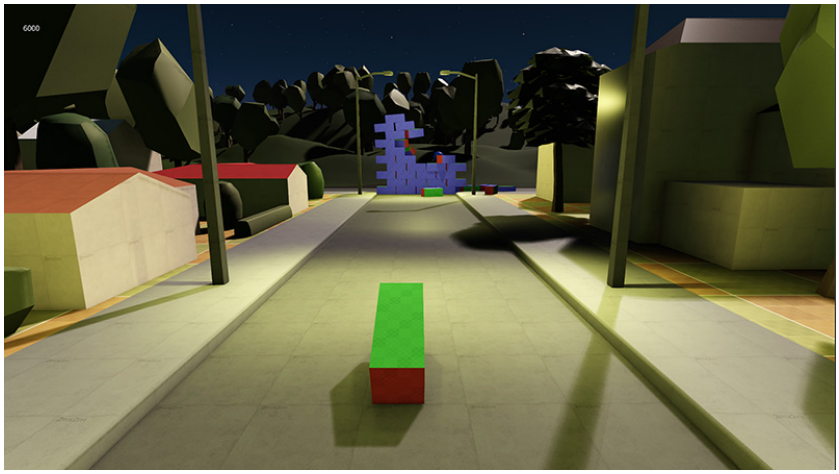
After completing this tutorial, you'll be knowledgeable enough to explore Lumberyard's wide range of tools and features. You can complete additional tutorials to help you learn more about specific tools and features and put you well on your way to building your next game.

To view the Getting Started Guide online, see

http://docs.aws.amazon.com/lumberyard/latest/gettingstartedguide







The Getting Started Guide is divided into the following sections:

Editor Overview

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.

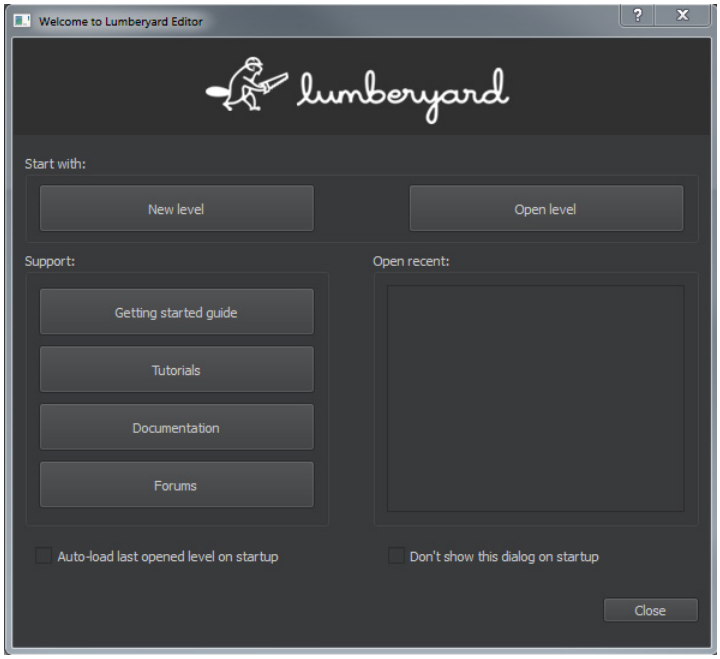
Topics

New Level Creation

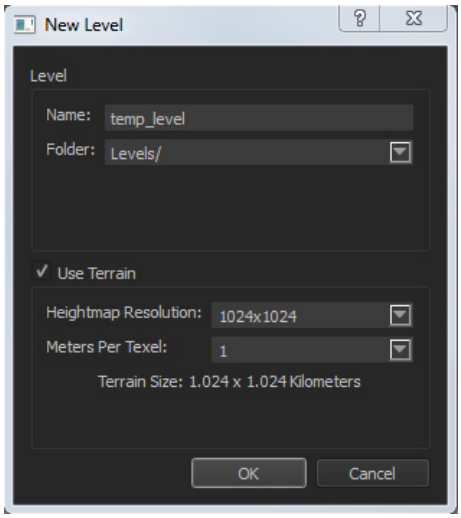
A level is a world or map that represents the space or area available to the player during the course of completing a discrete game objective. Most games consist of multiple levels. To create a level, you use **Lumberyard Editor**.

To create a level in Lumberyard Editor

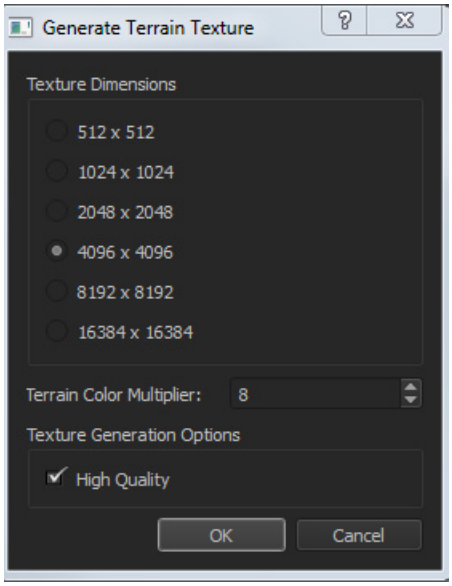
1. Start Lumberyard Editor as explained in the Quick Start Guide.
2. In the **Welcome to Lumberyard Editor** window, you can create a new level, open a recent level (if one exists), or open a level from within the level directory. You can also choose to stop showing this dialog on startup.



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 game play 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

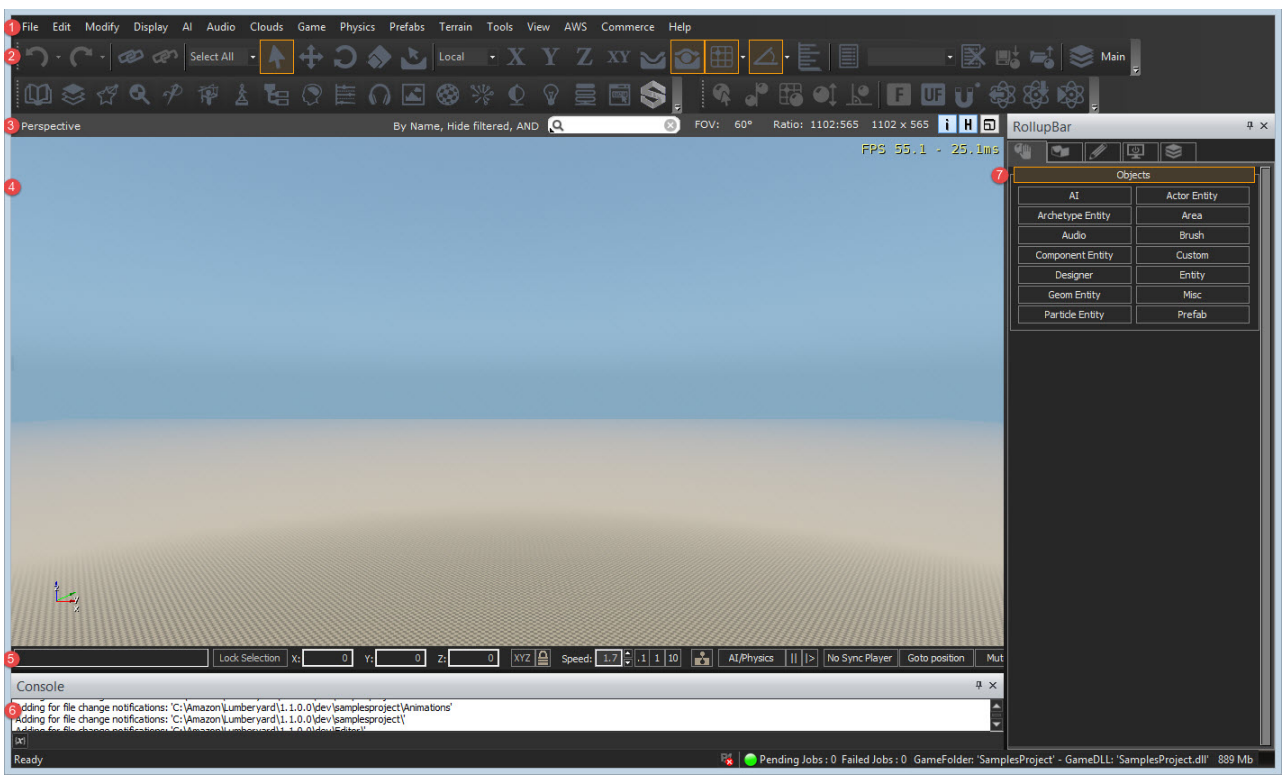
The Lumberyard Editor window comprises the following areas:

1. **Main menu** – All functions and settings
2. **Editor toolbar** – Most commonly used tools and editors
3. **Viewport header** – 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. **Console** – Input and output of editor and game data
7. **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

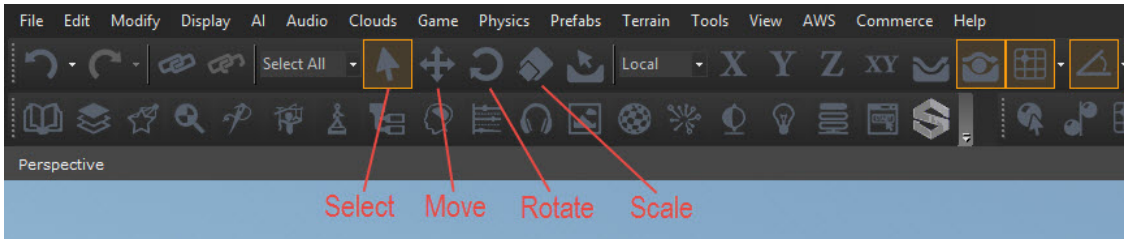
To get the most out of Lumberyard, familiarize yourself with these terms and areas.



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.



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

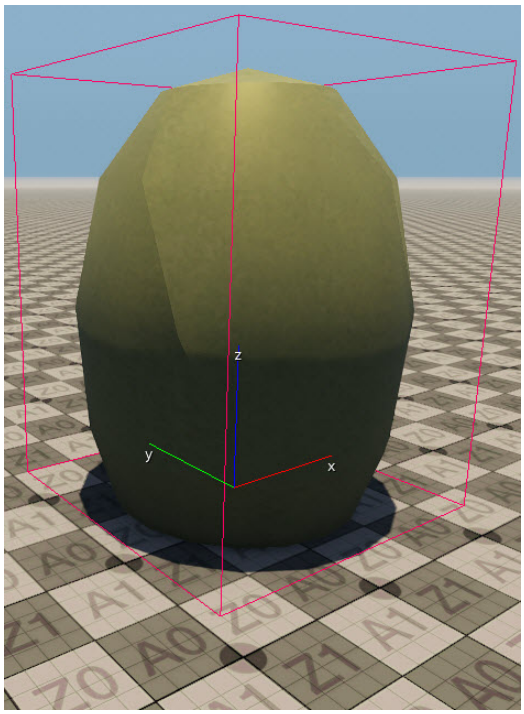
Place an object in your level so that you may follow along and test out the essential tools described in the following sections.

1. In the **Rollup Bar**, click **Brush** to display the assets you can add to the level.
2. Under the **Browser** heading, in the folder tree, expand `objects\styletown\architecture\props` and select `balloon`.
3. Drag the **balloon** object (the word *balloon*) into the **Perspective** viewport.
4. You are now ready to test the select, move, rotate, and scale tools.

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 cursor over the object you wish to select. When the object is highlighted yellow and the cursor changes to a +, left-click to select the object.

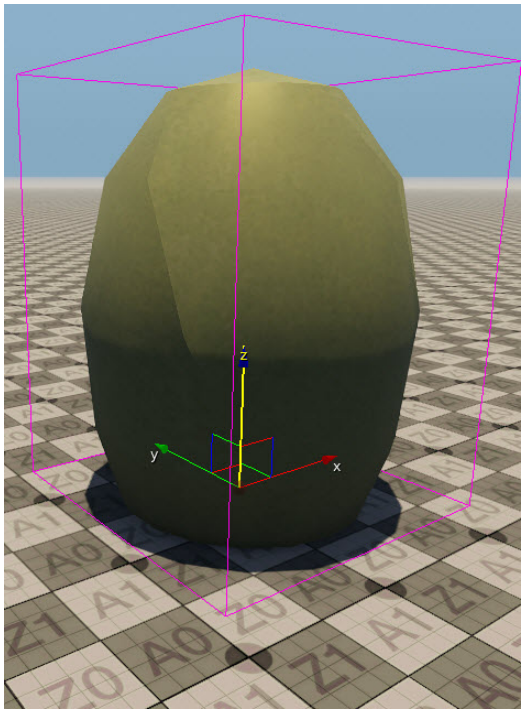


Move

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.

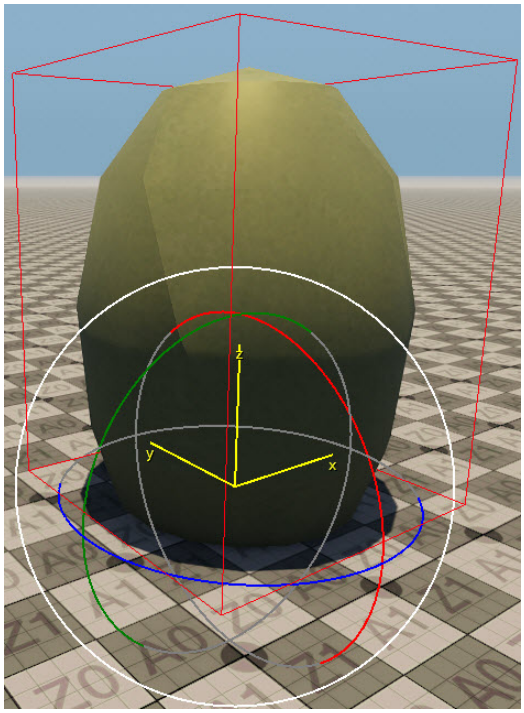


Rotate

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.

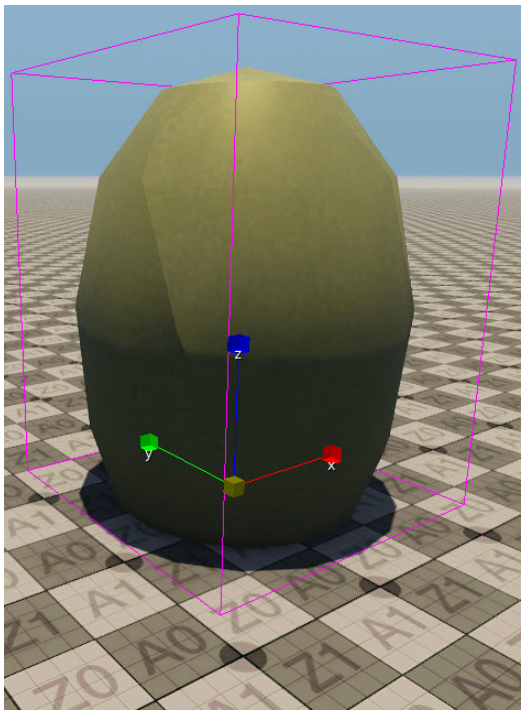
A larger outer circle also surrounds the entire gizmo. Select and drag this circle to rotate the object in relation to the screen display.



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.



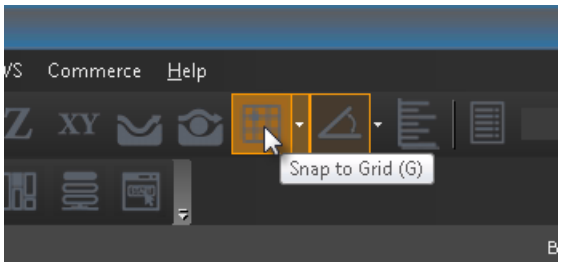
Snap Features

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.

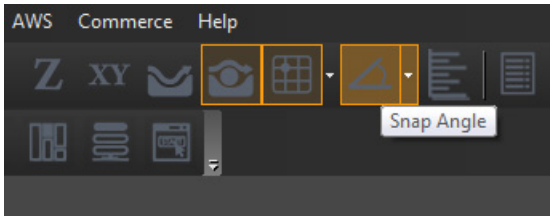


To use snap grid

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

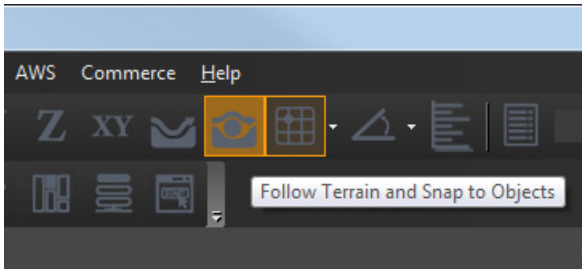
1. To turn snap angle on or off, click the **Snap Angle** icon on the toolbar.

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.

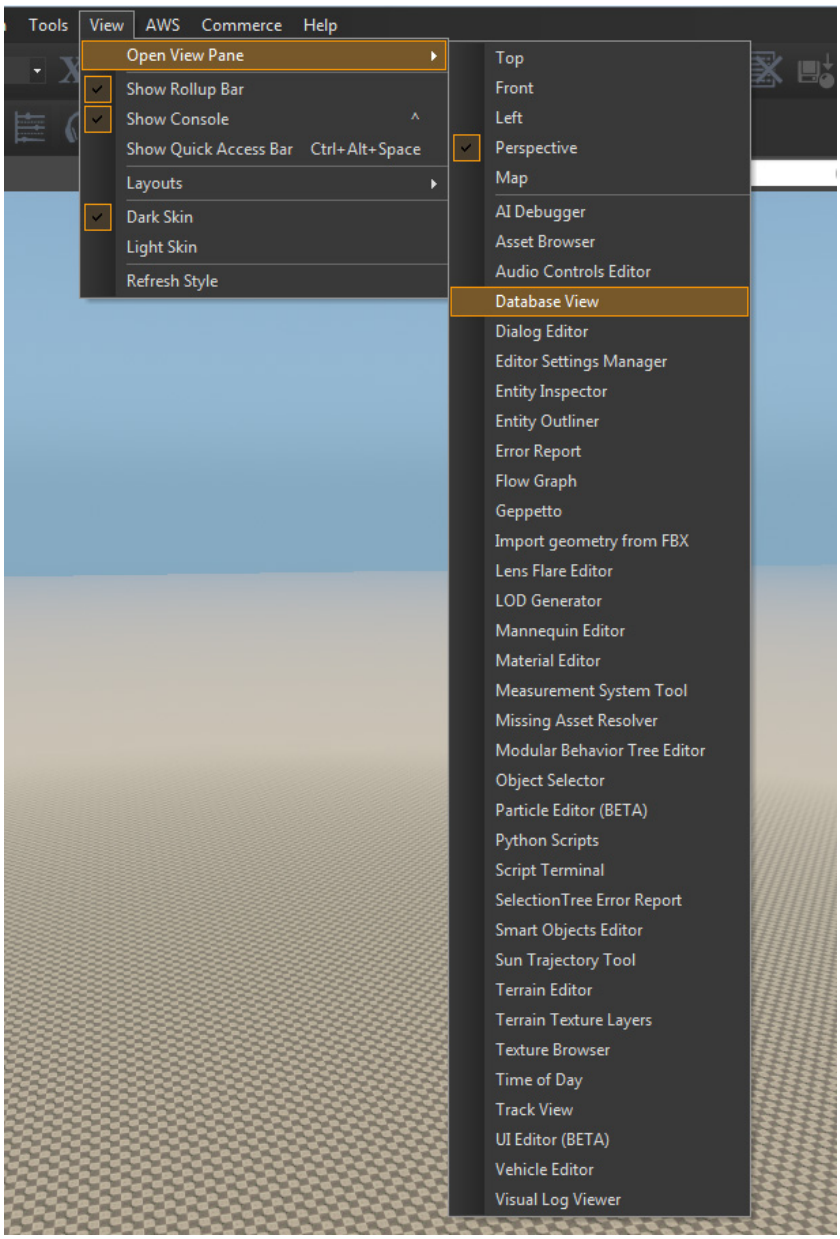
To navigate within your level in the **Perspective** Viewport, use the following click and drag actions:

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 **View, Open View Pane** menu.



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

Display Options and Settings

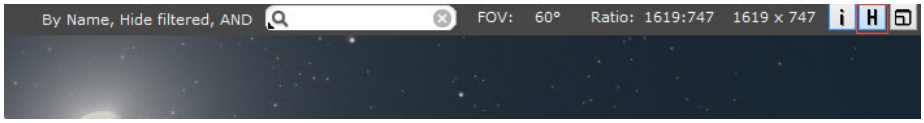
You can use Lumberyard Editor's display options and settings to customize your view to see the most useful tools and options.

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.

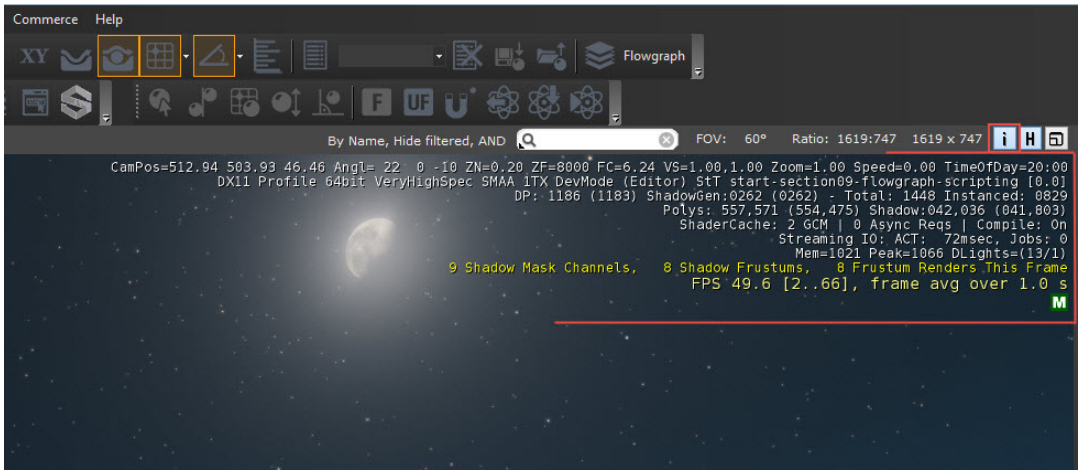
Show/Hide Helpers

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**  . Click this icon multiple times to choose the level of information you'd like to see.



Navigation Speed Settings

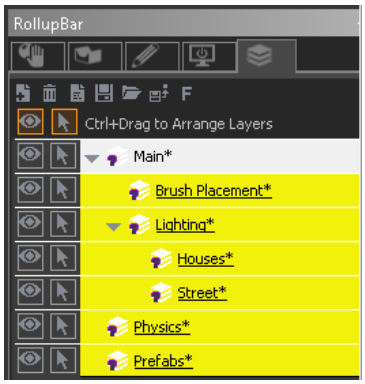
You can adjust your **Perspective** viewport navigation speed. The **Speed** setting displays the current movement speed setting. Type a number into the **Speed** field, or click **.1** (slow), **1** (normal), or **10** (fast).

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. 69).

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

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.

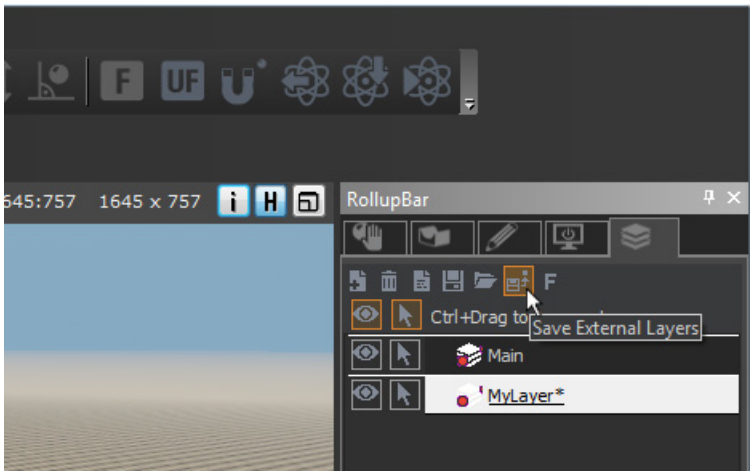
Furthermore, you can organize your layers into nested groups by holding **Ctrl** and dragging each layer to your preferred location.

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

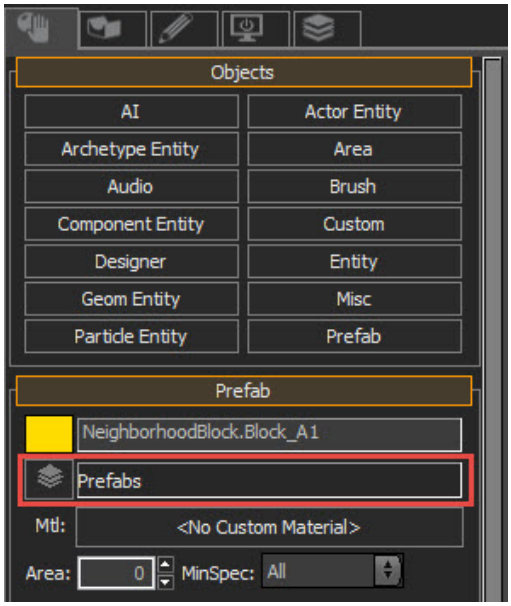
With the **Layers** tab multiple users can work within the same level. To do that, each user can create his or her own layer and building all their content within that layer. Although not required, a source control tool such as Perforce provides a useful way to manage these different layers; users just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

Moving Assets Between Layers

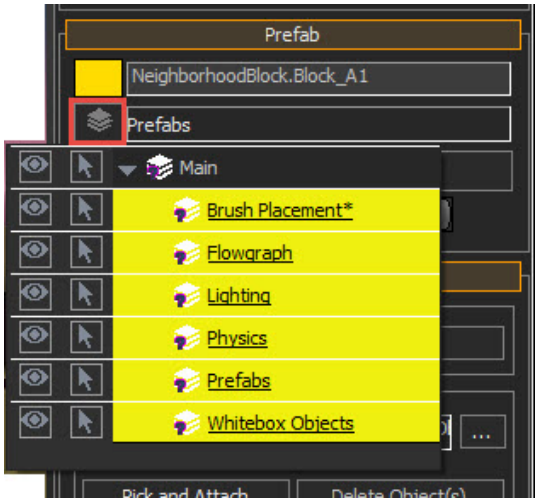
Each entity, brush, or designer object you place in the level is assigned to the currently selected layer. If you have not created any additional layers, objects are placed in the default main layer.

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.



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

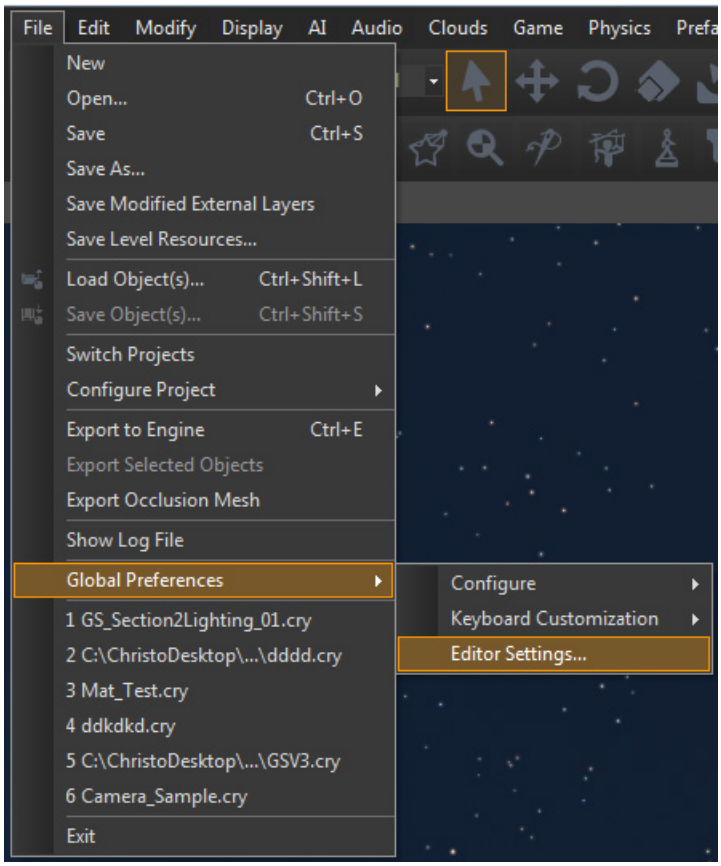


Auto Backup

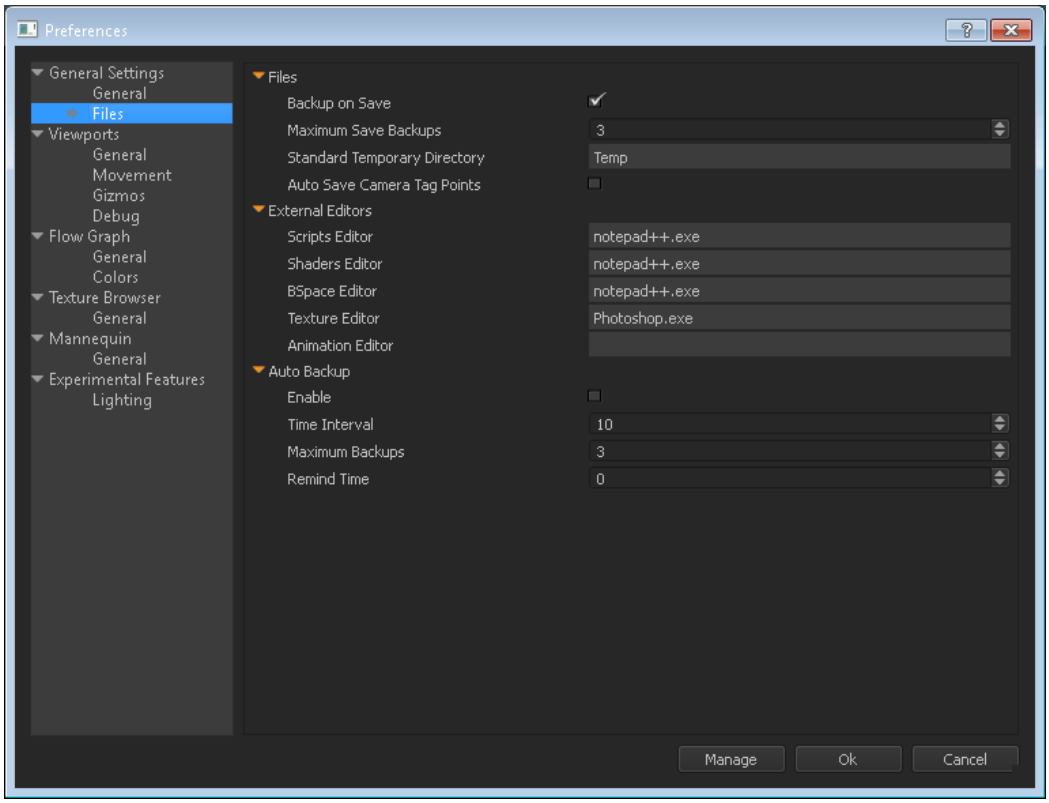
Lumberyard's **Auto Backup** feature is on by default. **Auto Backup** saves your level file incrementally. This helps prevent loss of your work in case of unexpected problems.

To customize your Auto Backup settings

1. From the main menu, open **File**, **Global Preferences**, **Editor Settings**.



2. Under **General Settings**, click **Files**. From here, you can customize your **Auto Backup** settings.

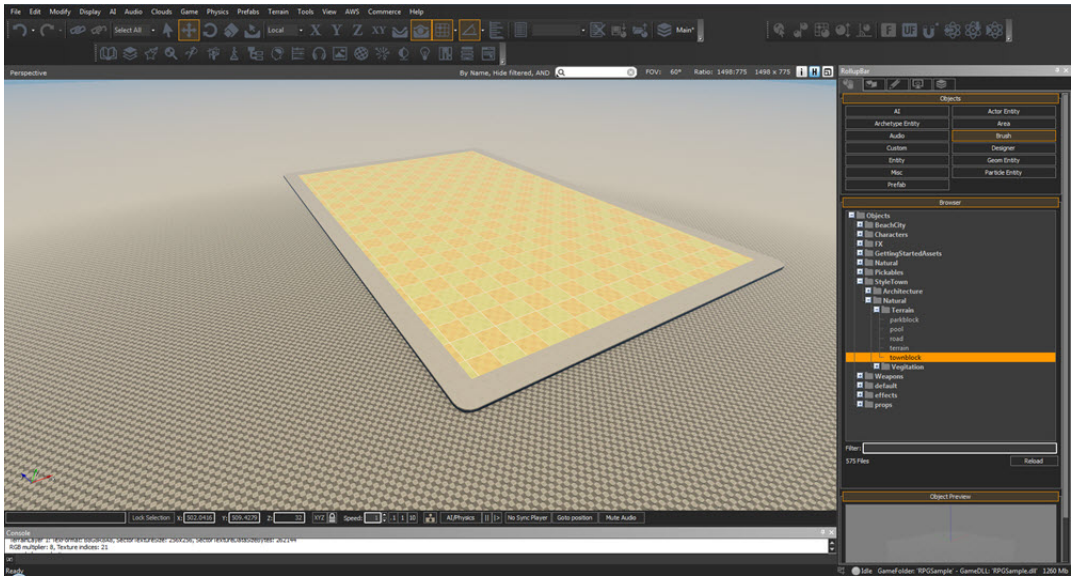


Brushes

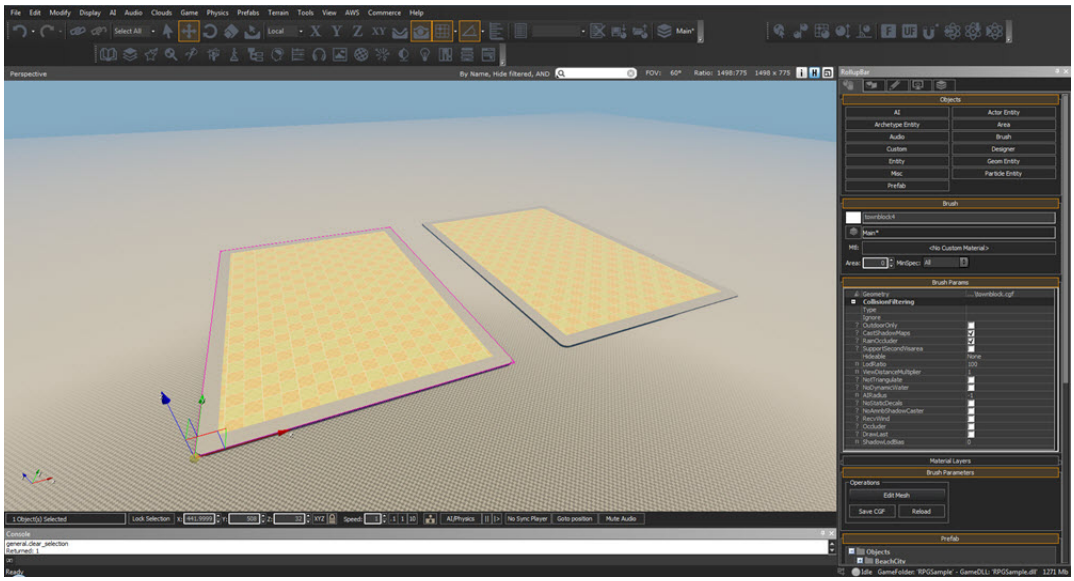
Placing objects in your level adds realism and interest to your environment. This tutorial teaches you how to start building your scene by adding a particular type of static 3D shapes known as brush objects .

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+C** to copy your townblock, and move your second townblock next to your first townblock. Click to place it.


File Edit Modify Display AI Audio Clouds Game Physics Prefabs Terrain Tools View AWS Commerce Help
Perspective
Objects
AI
Actor Entity
Archetype Entity
Area
Audio
Brush
Custom
Designer
Entity
Geom Entity
Misc
Particle Entity
Prefab
Brush
Brush Params
Material Layers
Brush Parameters
Operations
Edit Mesh
Save CGF
Reload
Prefab
Objects
BeachCity
Ready

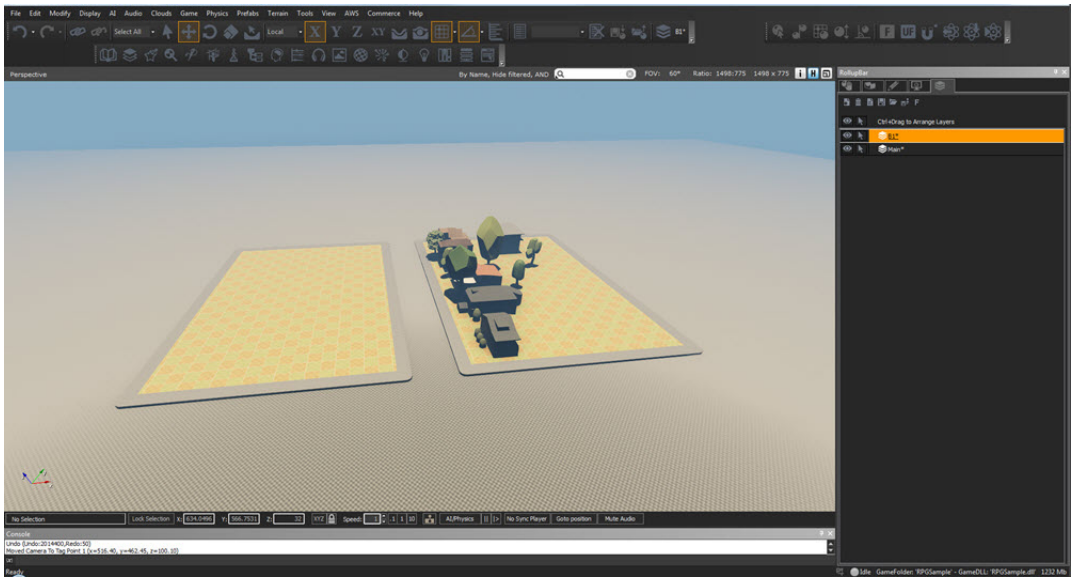
6. Click **Brush** again in the **Rollup Bar**.

   Expand `StyleTown\Architecture\Buildings`. Drag four buildings onto one of your townblocks.

   Tip
   When dragging, you see only the cursor. After you release the mouse button, your brush appears.

7. Click **Brush** again. 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:



Prefabs

Prefabs are combinations of pre-defined assets that enable you to create content more quickly.

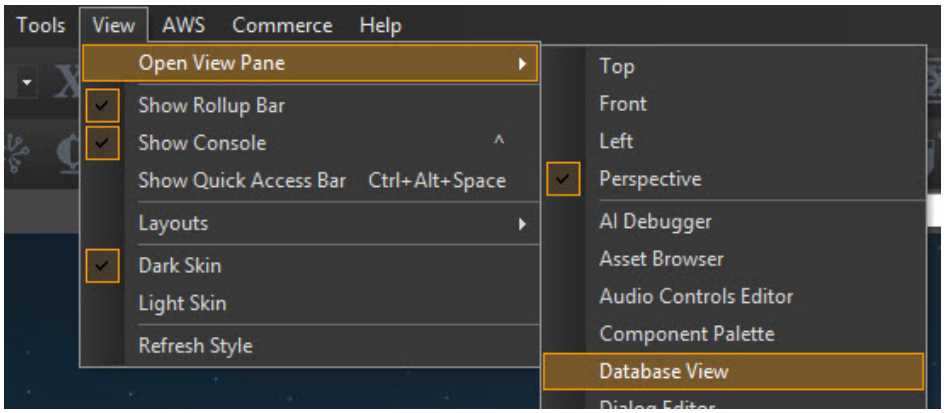
You'll learn the simplest way to use a prefab—by grouping sets of brushes together. This prefab simplifies and speeds up the building of your environment.

To place prefabs

1. To access previously created prefabs, open the **Database View** editor.

   #### To do this, do one of the following

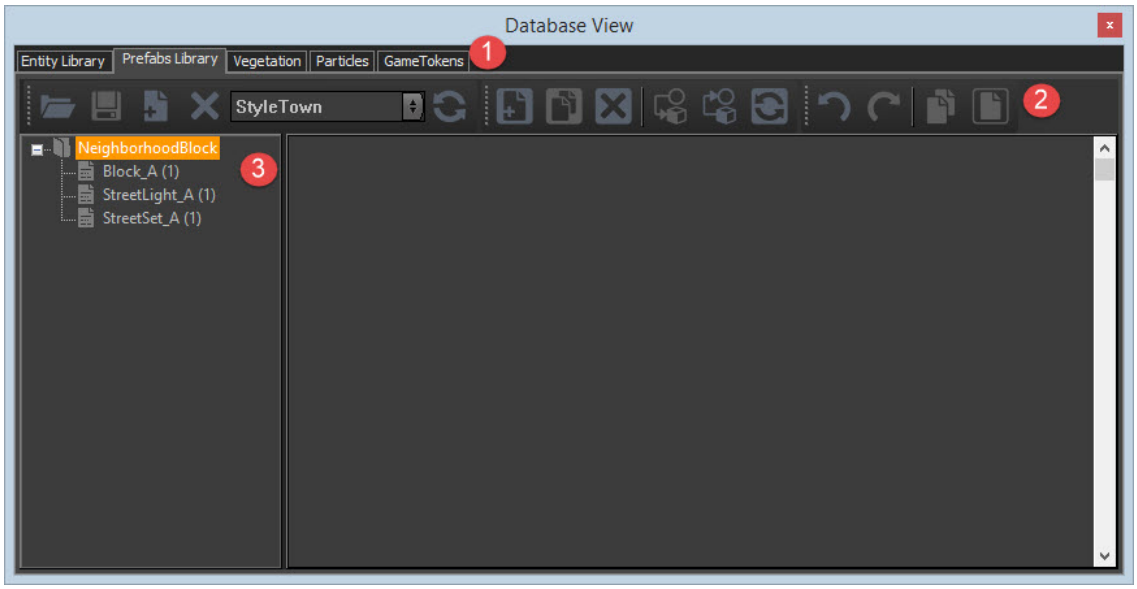
   - On the editor toolbar, click **Database View**
   - From the main menu, open **View**, **Open View Pane**, **Database View**



   #### The Database View editor has the following features

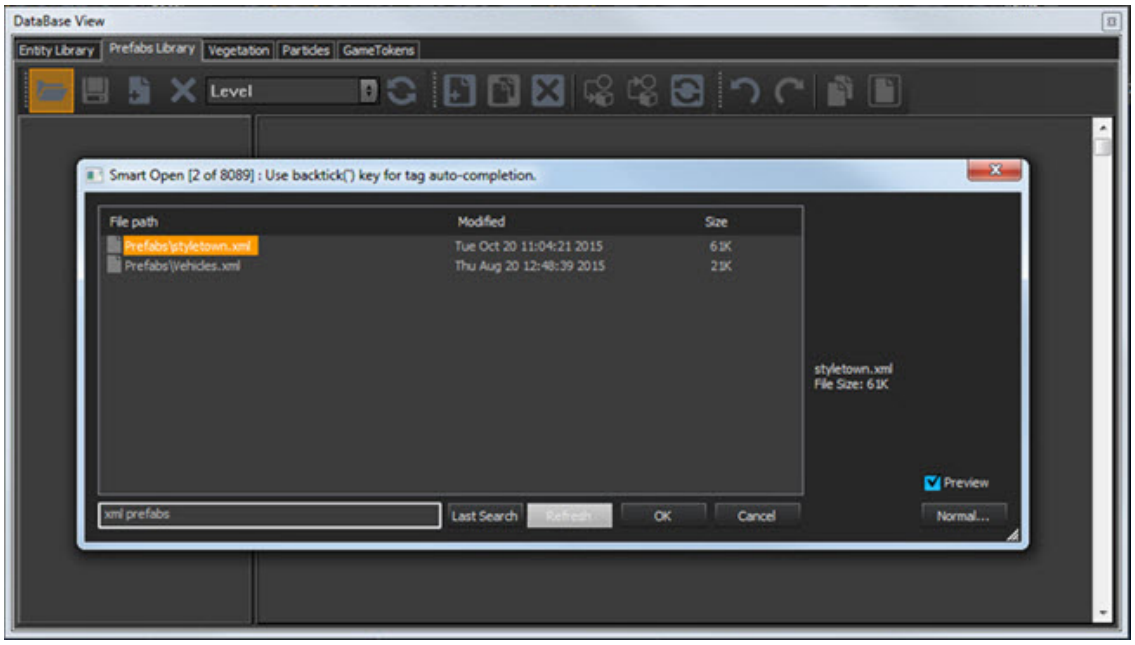
   - **1: Database tabs** – Database types you can view and manage:
     - **Entity Library**
     - **Prefabs Library**

  - **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 field is empty at first; the next steps describe how to open and access prefab libraries so you can see the NeighborhoodBlock prefabs as shown in the following image.



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

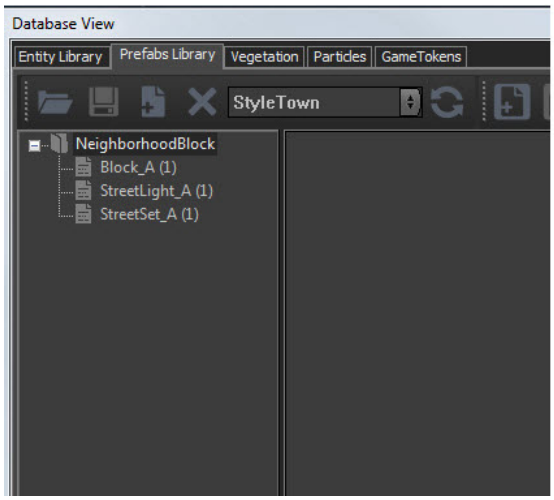
3. Click **Load Library** and select `Prefabs\styletown.xml`. Click **OK**.



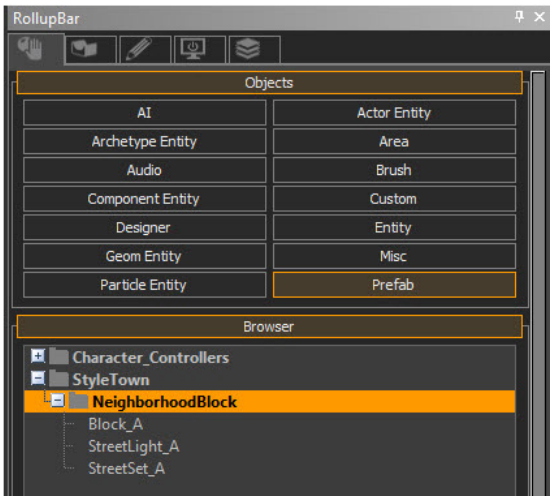
4. In the file tree view, the prefab library **NeighborhoodBlock** appears.

   After you add a prefab library to a level using this method, these prefabs appear when you click **Prefab** in the **Rollup Bar**, which enables you to more quickly place prefabs in a level.

Database View



Rollup Bar View



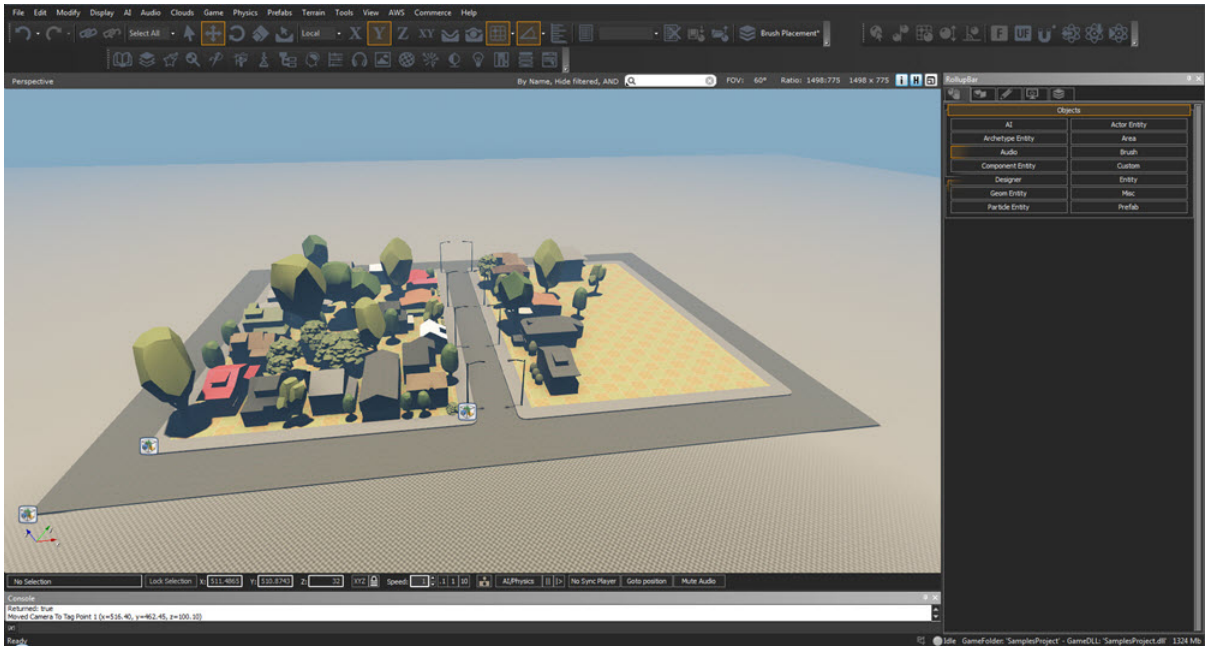
5. Drag **StreetSet_A** into the **Perspective** viewport.

   Position the street so that it fits around the two town blocks.

   This street prefab comes into the level sitting above the ground plane. Verify that **Grid Snap** is on, and then move the street down so that it aligns with the ground.
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:



Terrain

This tutorial teaches you how to apply materials to the terrain, modify the terrain height, and use the vegetation tool to paint trees.

Terrain Painting

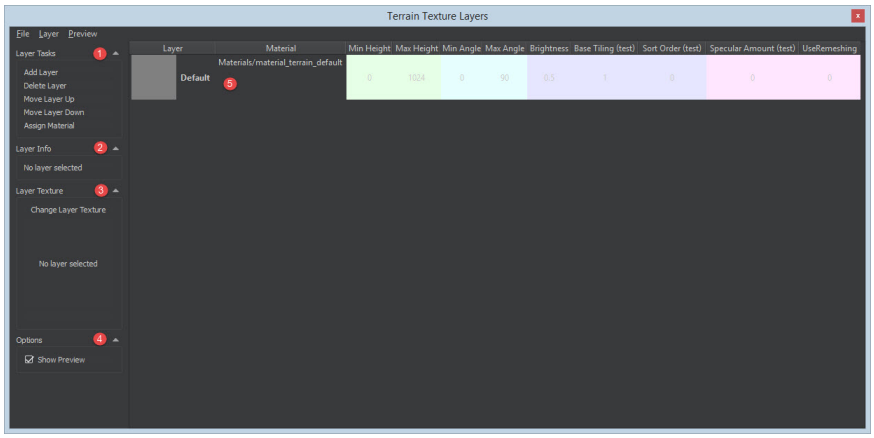
Now that you have the basics of the scene built with brushes, you'll start building the surrounding terrain environment.

To do this, you'll use two new editors: The **Terrain Texture Layers** editor and the **Material Editor**.

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

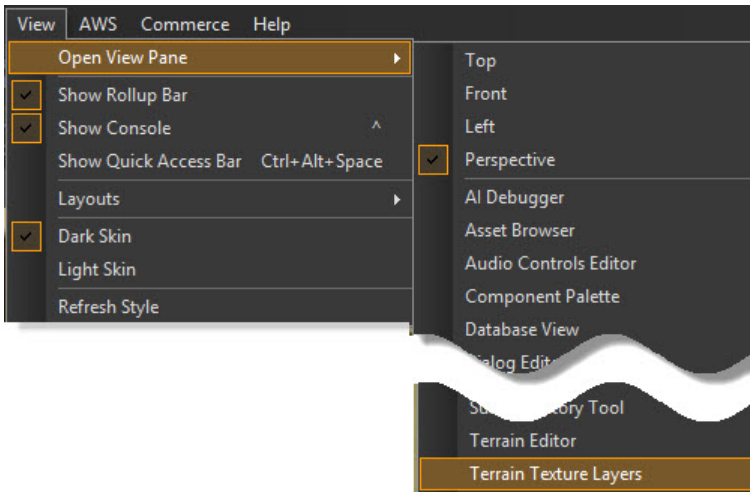
- **1: Layer Tasks** – Controls for adding, deleting, and reordering layers in the **Layer** List
- **2: Layer Info** – Information about the selected layer, including the layer size and surface type count
- **3: Layer Texture** – Low-detail texture swatch; displays color information for the surface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


Terrain Texture Layers
File
Layer
Preview
Layer Tasks
1
Add Layer
Delete Layer
Move Layer Up
Move Layer Down
Assign Material
Layer Info
2
No layer selected
Layer Texture
3
Change Layer Texture
No layer selected
Options
4
Show Preview
Layer
Material
Min Height
Max Height
Min Angle
Max Angle
Brightness
Base Tiling (test)
Sort Order (test)
Specular Amount (test)
UseRemeshing
Materials/material_terrain_default
Default
5
0
1024
0
90
0.5
1
0
0
0

To open the Terrain Texture Layers editor

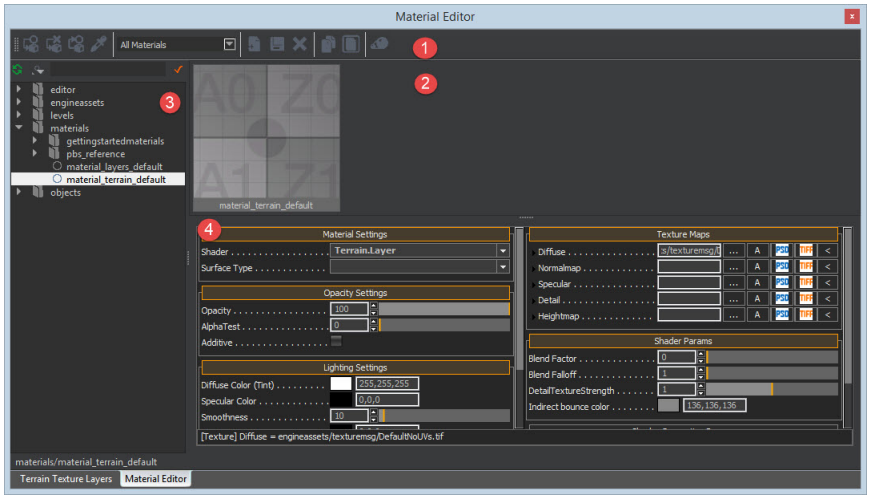
Do one of the following:

- On the editor toolbar, click the **Terrain Textures Layers Editor** icon
- From the main menu, open **View**, **Open View Pane**, **Terrain Texture Layers**



The Material Editor has the following features

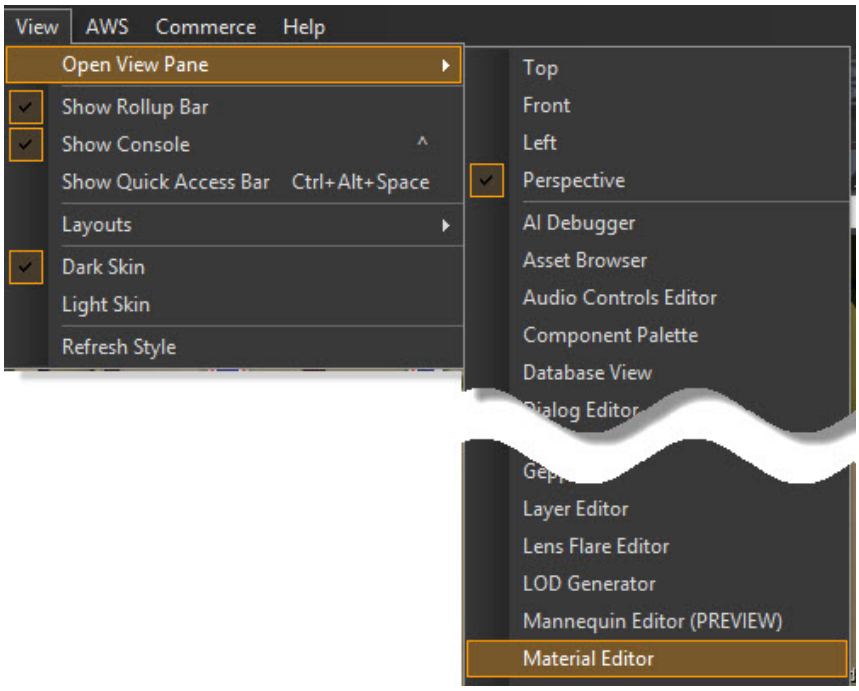
- **1: Editor toolbar** – Tool list for applying, deleting, saving, and creating materials
- **2: Material preview** – Display for the selected material's appearance
- **3: Material folder directory** – Folder tree to navigate through the materials available for use in the level
- **4: Material properties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



To open the Material Editor

Do one of the following:

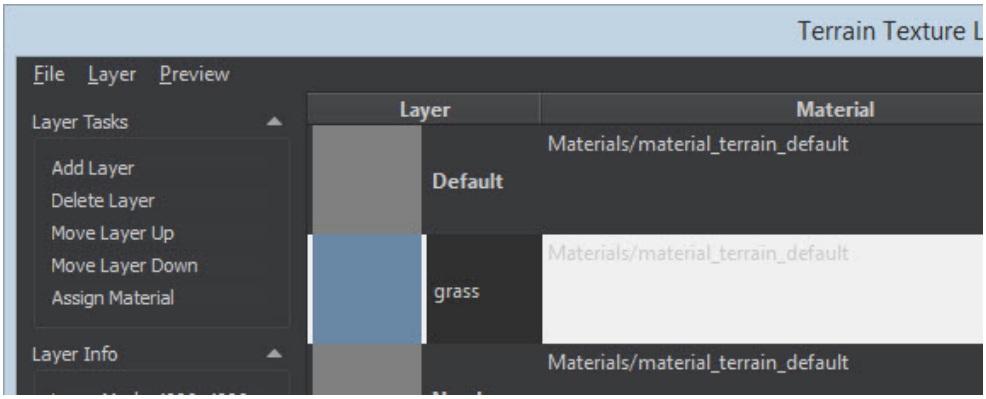
- On the editor toolbar, click the **Material Editor Dialog** icon
- From the main menu, open **View**, **Open View Pane**, **Material Editor**


View
AWS
Commerce
Help
Open View Pane
Show Rollup Bar
Show Console
Show Quick Access Bar
Ctrl+Alt+Space
Layouts
Dark Skin
Light Skin
Refresh Style
Top
Front
Left
Perspective
AI Debugger
Asset Browser
Audio Controls Editor
Component Palette
Database View
Layer Editor
Lens Flare Editor
LOD Generator
Mannequin Editor (PREVIEW)
Material Editor

You are ready to assign materials to your terrain mesh.

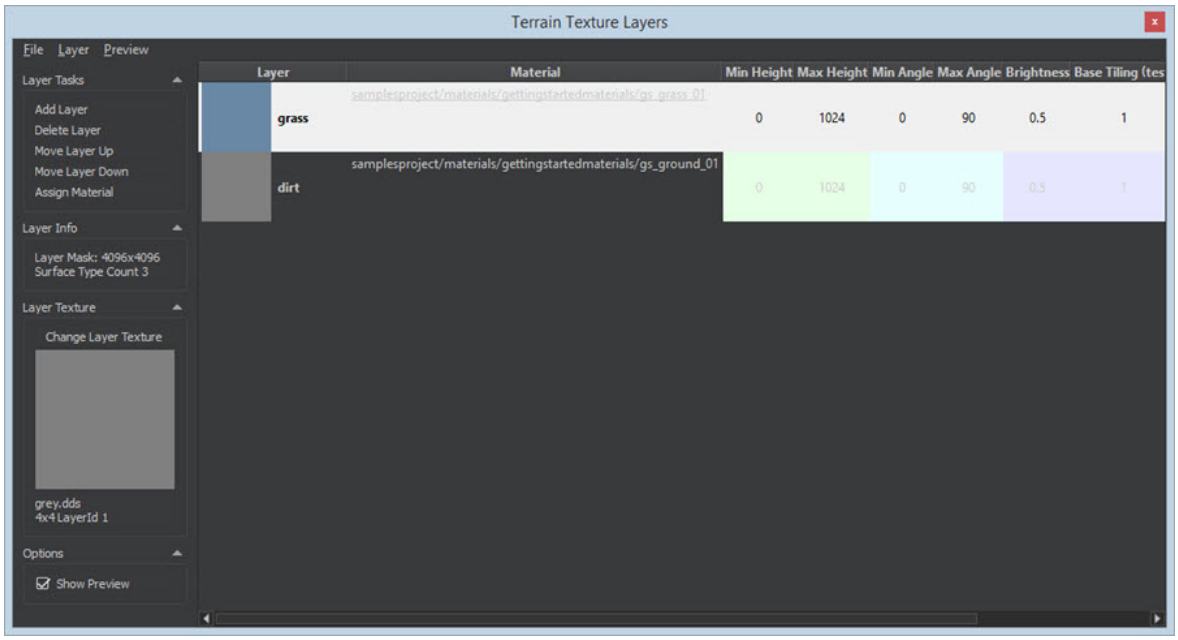
To build the surrounding terrain environment

1. In the **Terrain Texture Layers** editor, 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`**.

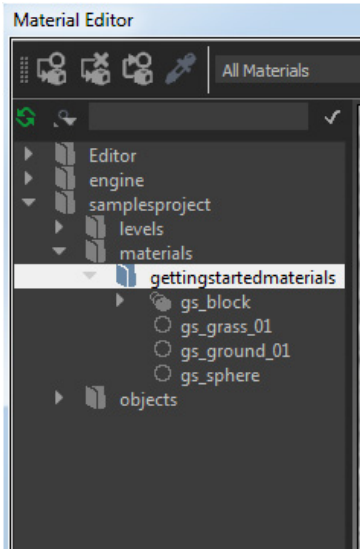


3. In the second new layer, double-click **NewLayer** and rename it **`dirt`**.

4. Select the **grass** layer.



5. Go back to the **Material Editor** and select the material **gs_grass_01** located in the following path: `materials\gettingstartedmaterials\gs_grass_01`.

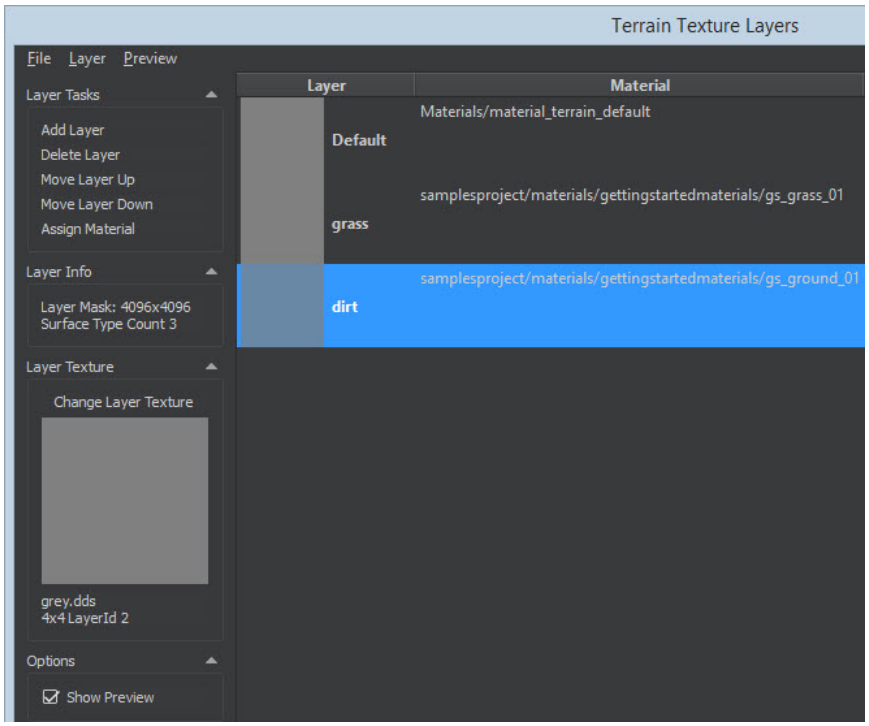


6. In the **Terrain Texture Layers** editor, the grass layer should still be selected. In the **Layer Tasks** area, click **Assign Material**.
7. Next, select the **dirt** layer.

   Switch back to the **Material Editor** and select **gs_ground_01** in the `gettingstartedmaterials` folder.

8. Switch to the **Terrain Texture Layers** editor. With the dirt layer still selected, click **Assign Material** in the **Layer Tasks** area.

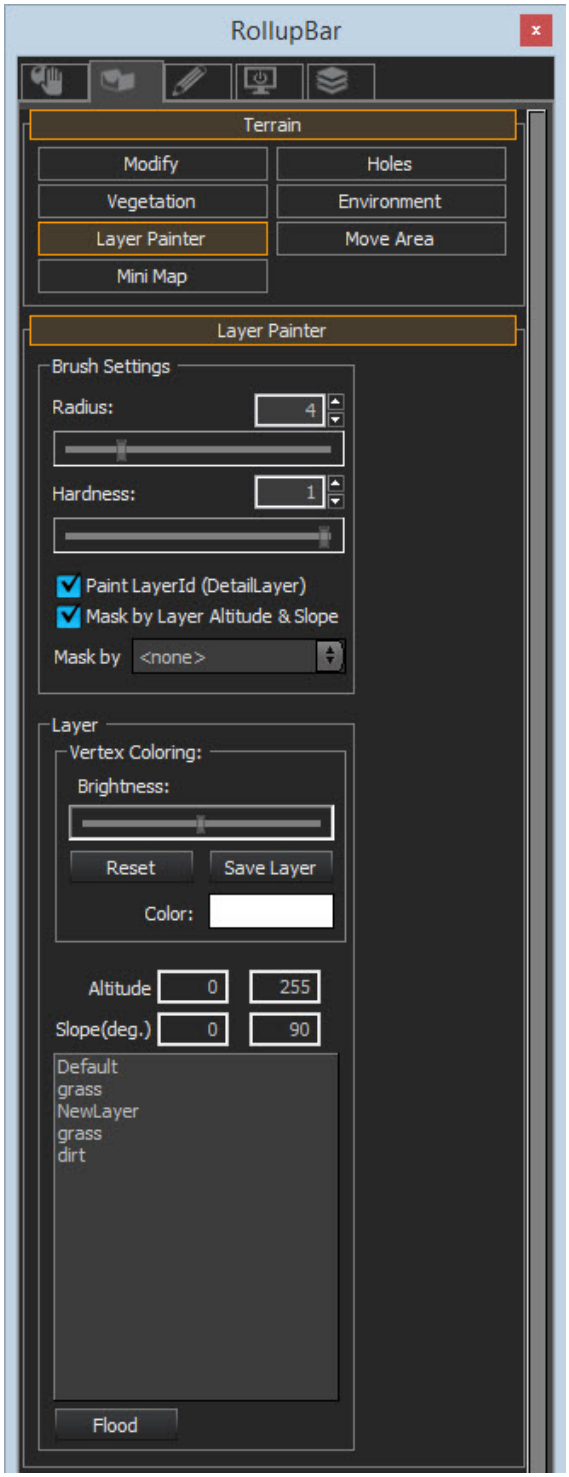
   Note
   In each of the terrain material layers is a small material preview box. This material preview box displays the assigned layer texture, not the material assigned from the material editor. For this tutorial, we are using the default `grey.dds` file, so both the grass and dirt layers appear with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



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

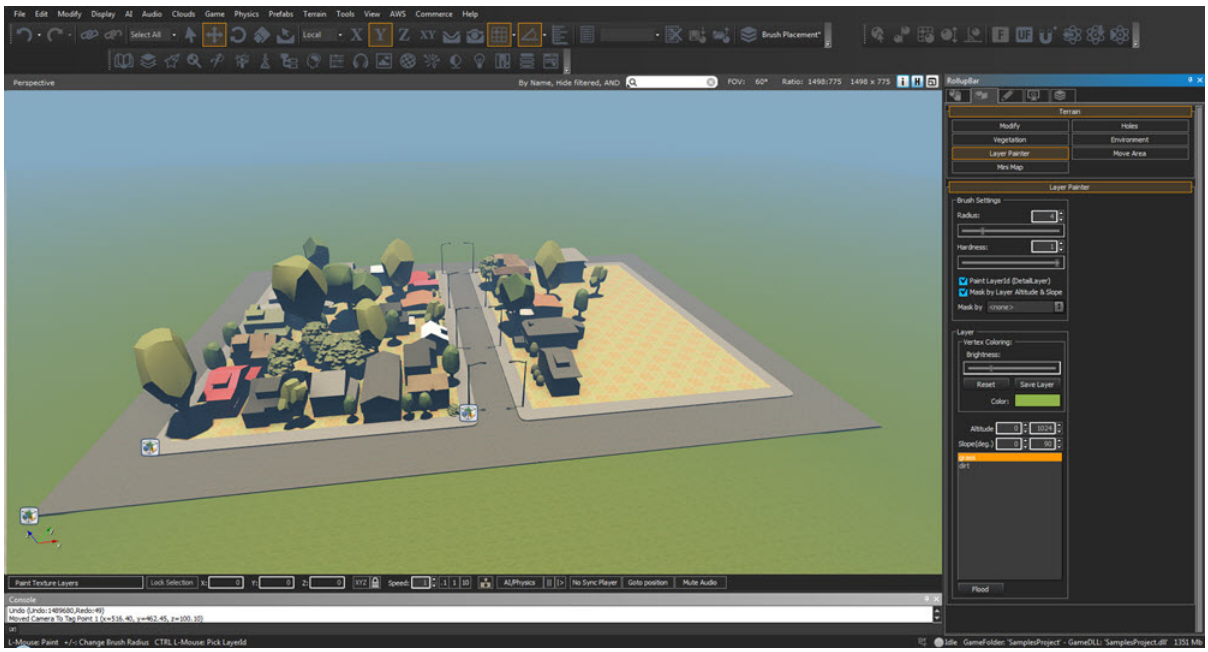
   You are now ready to paint grass and dirt textures onto the terrain.

10. In the **Rollup Bar**, select 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 to **`145`**, **`180`**, **`75`** for a grass-green color. Click **OK**.

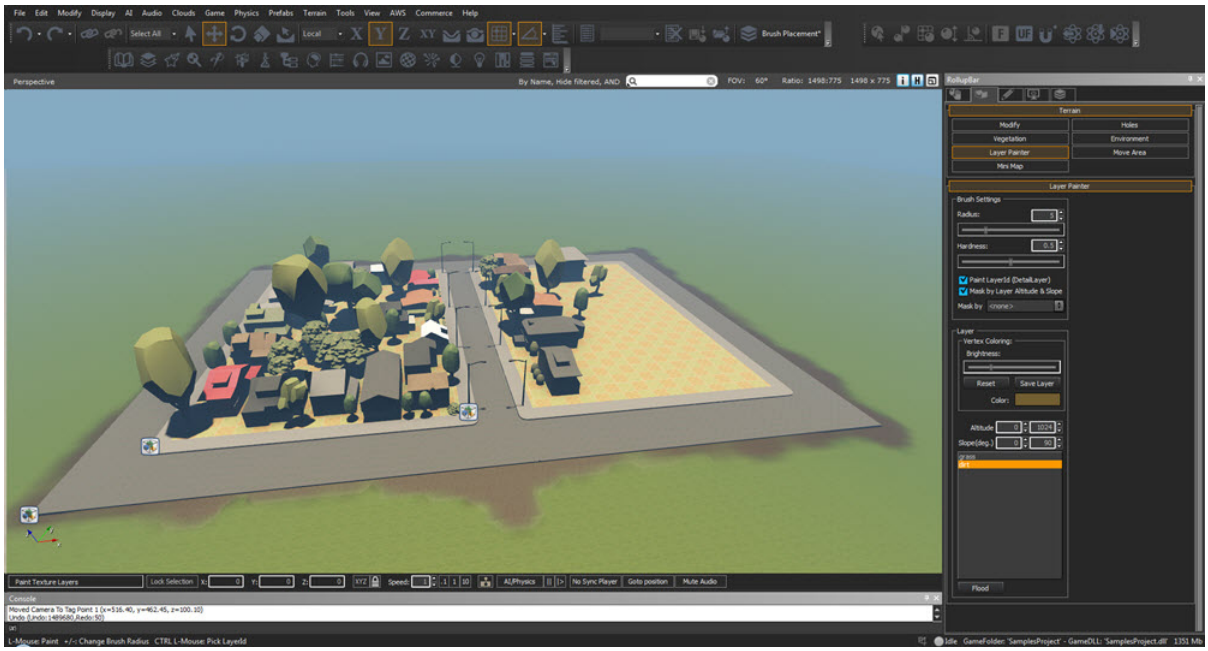
13. Click **Flood** at the bottom of the **Layer Painter** section. The terrain is now covered in the grass texture and looks similar to this:



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.

Terrain Height

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

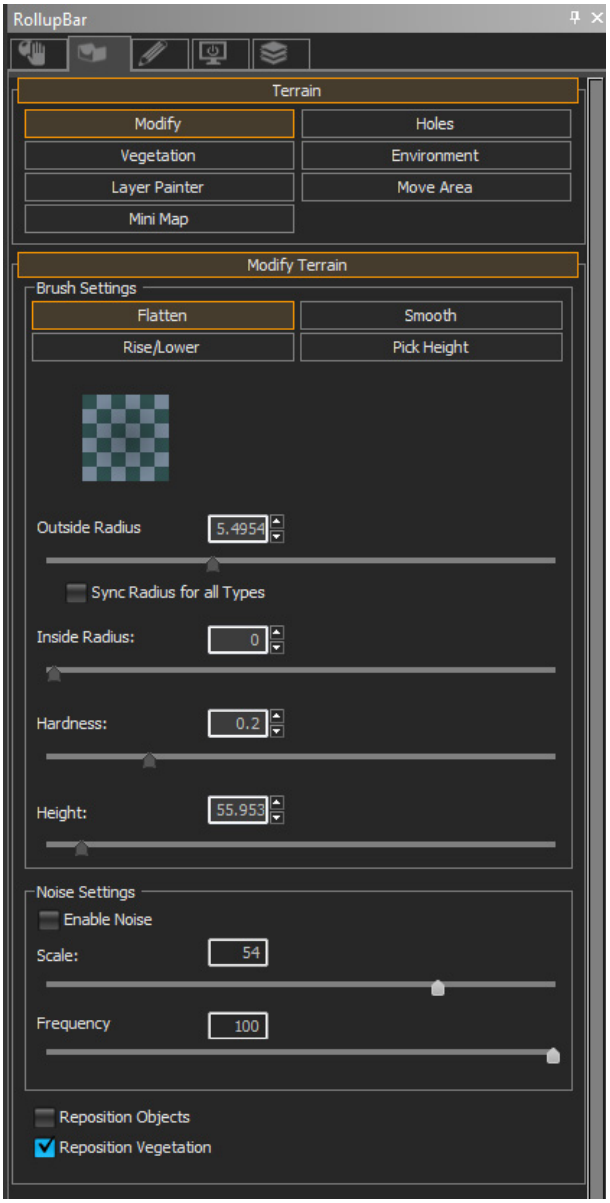
The **Modify Terrain** section features the following:

Brush Settings

- **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
- **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

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



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 click and 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`**

   Paint again on the terrain. Notice how the terrain rises up straight and rigid.
5. Click the **Smooth** tool, and use the following settings:

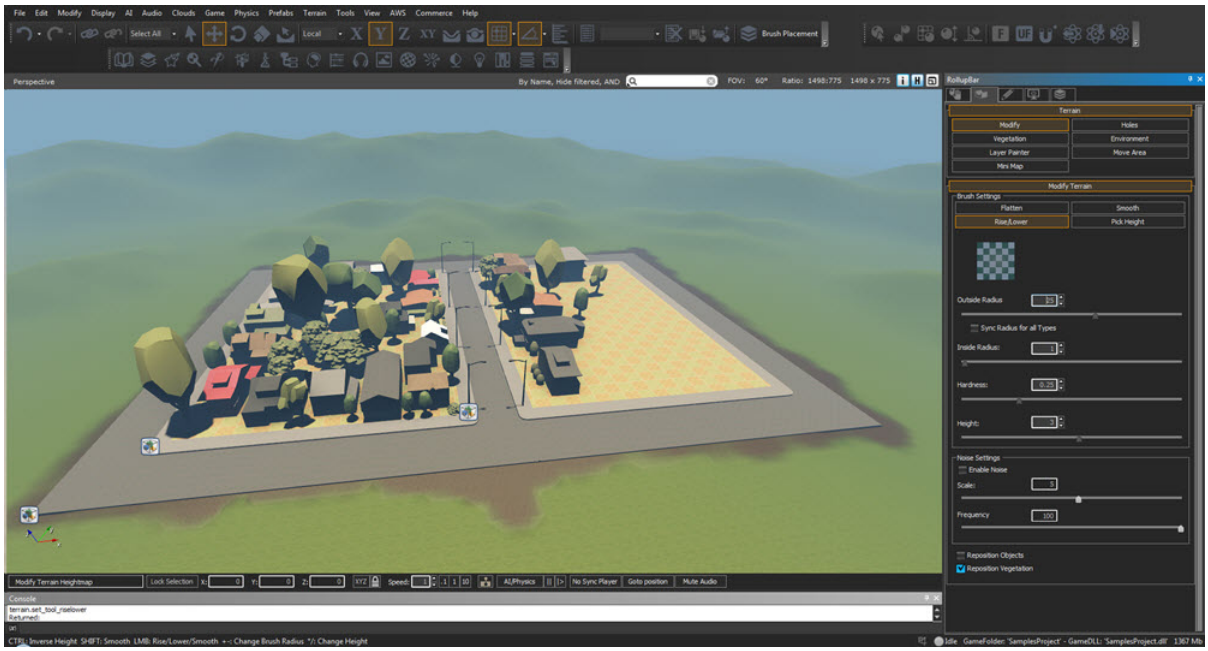
   - **Outside Radius** = **`25`**
   - **Hardness** = **`.2`**

   Paint with the smooth brush over the last area of terrain you created. Notice the smoothing of the terrain.
6. 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.

   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.
7. Click **Flatten**, and use the following settings:

   - **Outside Radius** = **`25`**
   - **Inside Radius** = 0
   - **Hardness** = **`1`**

   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.
8. Using the terrain height tools, create a range of high and long hills in the distant background from the neighborhood block area you created.
9. 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:



10. Save your level file.

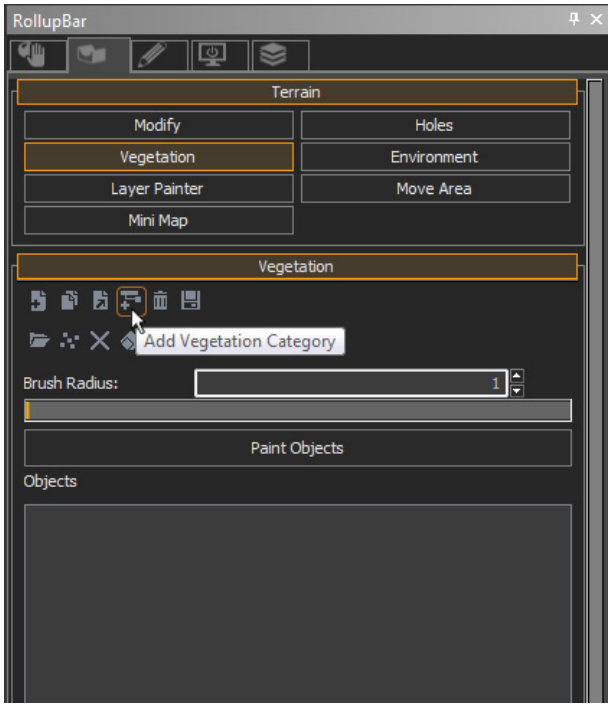
Terrain Vegetation

Using the **Vegetation** tool, you can 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.

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

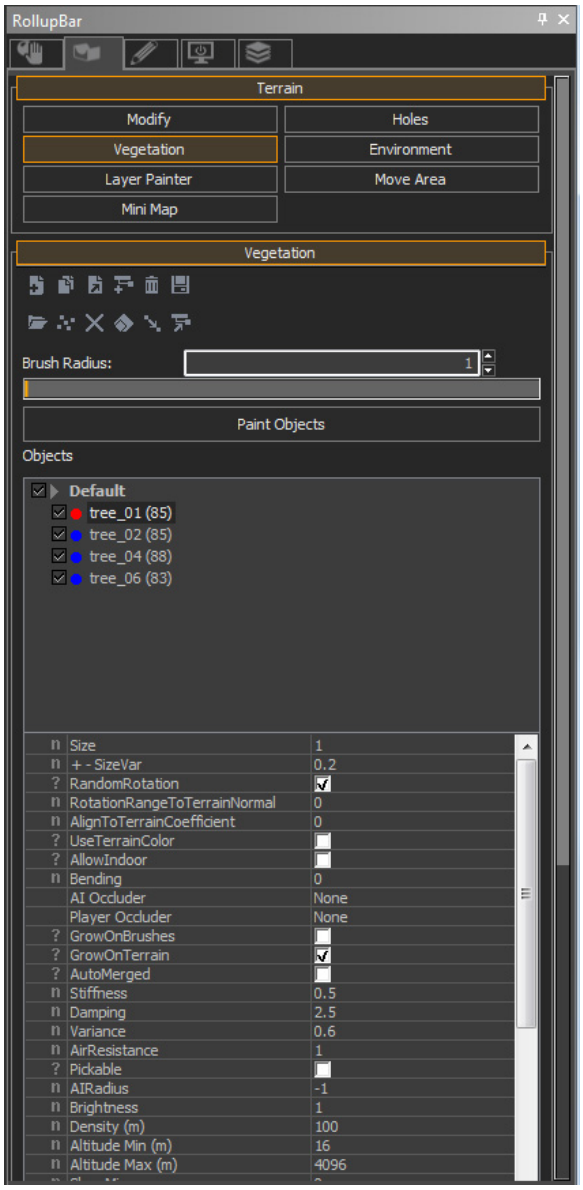
To add trees

1. In the **Rollup Bar**, on the **Terrain** tab, click **Vegetation** .

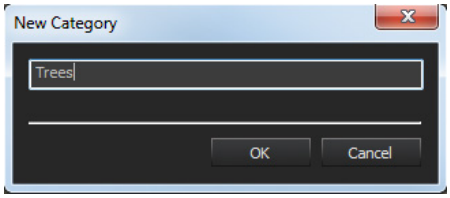


In the **Vegetation** tool, you can modify the following settings:

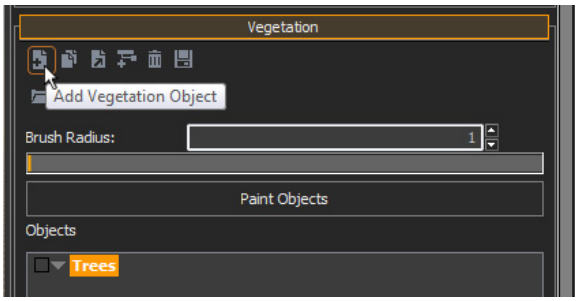
- **Toolbar** – Tools to create, modify, and organize vegetation types
- **Brush Radius** – Size of the brush used to paint vegetation into the level
- **Paint Objects** – Enables you to paint in the level
- **Objects** – List of vegetation objects
- **Table of attributes** – List of attributes that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



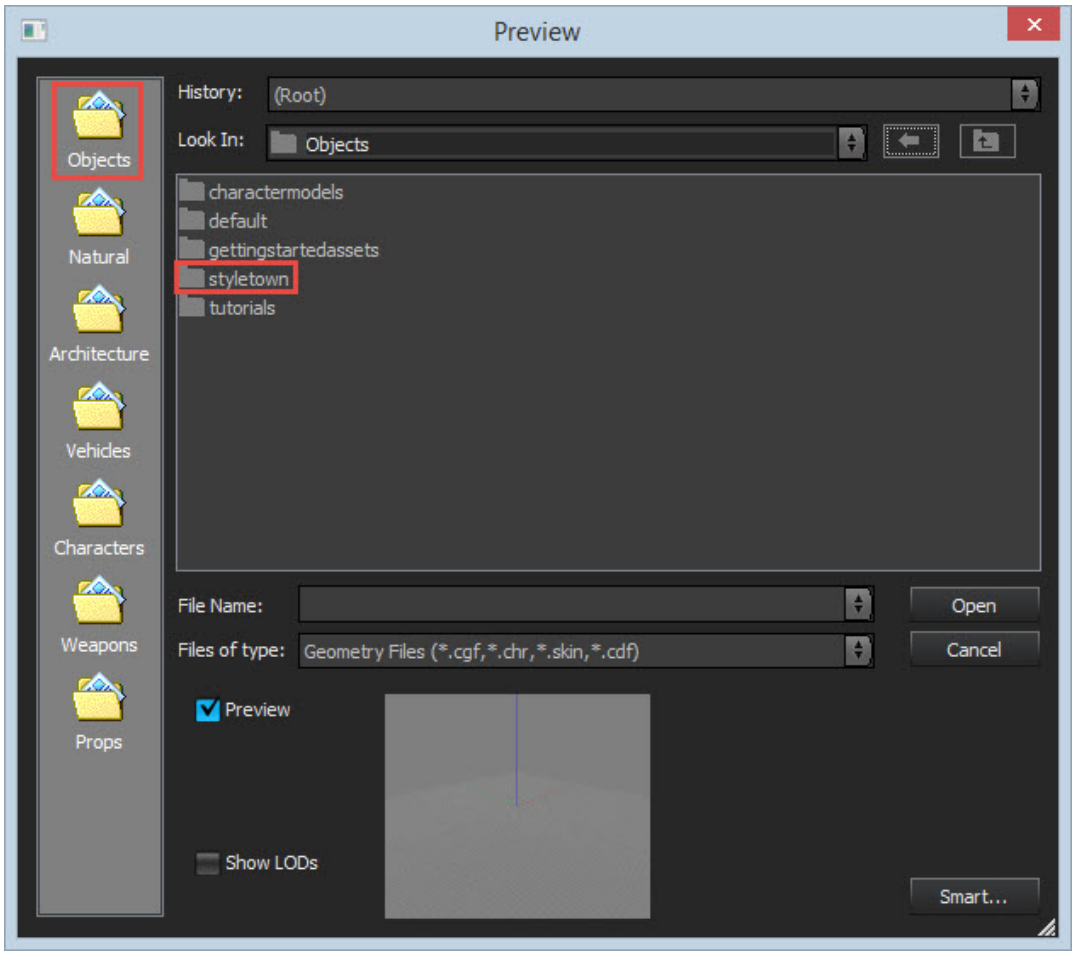
2. In the **New Category** dialog, type `Trees`. Click **OK**.



3. In the **Objects** list, select the **Trees** category you just created.
4. In the tool list, click **Add Vegetation Object** .



5. In the **Preview** dialog, click the `Objects` folder.
6. Open the folder `StyleTown`.

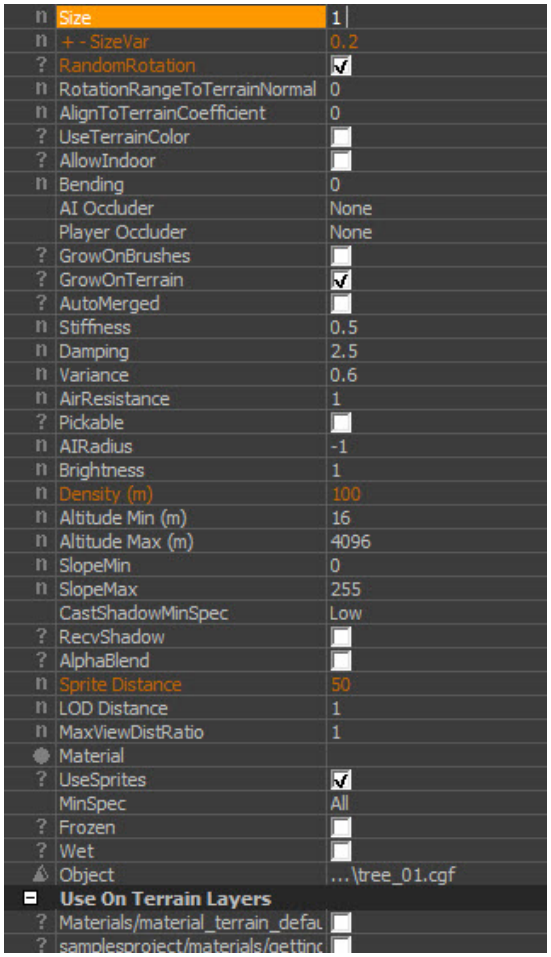

Preview
History:
(Root)
Look In:
Objects
Objects
Natural
Architecture
Vehicles
Characters
Weapons
Props
charactermodels
default
gettingstartedassets
styletown
tutorials
File Name:
Open
Files of type:
Geometry Files (*.cgf,*.chr,*.skin,*.cdf)
Cancel
Preview
Show LODs
Smart...

7. Open `Natural`, then `Vegetation`.

   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**.

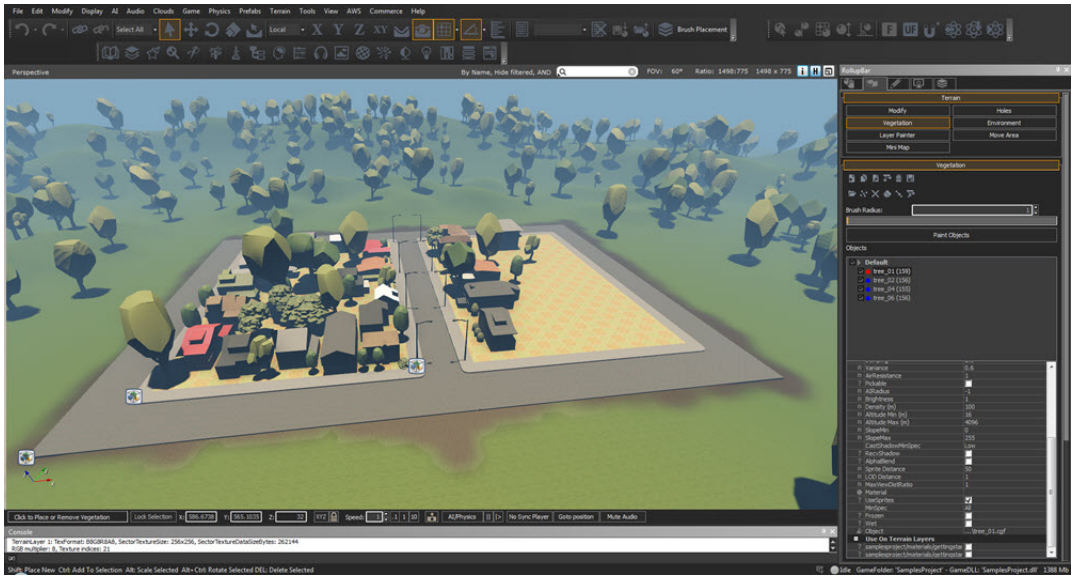
   If you painted trees into the environment at this point, every tree would appear with the default brush settings, providing no variation on size, rotation, or spread of the trees.
8. To create variation in tree size, rotation, and spread, select the first tree on the list and change the following settings in the attributes list. Do this for each of the trees in your list.

   - **+- Size Var** = 0.2 – Amount of variation in tree size
   - **Random Rotation** = On (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



9. Click the **Trees** group name. Adjust the **Brush Radius** to **`50`** (this size is appropriate for filling the terrain space quickly).
10. Click **Paint Objects**, then place your cursor 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:



11. Save your level file.

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

Lighting

This tutorial teaches you about the tools and features used to light a scene, including Environment Probes, Time of Day, and Lights.

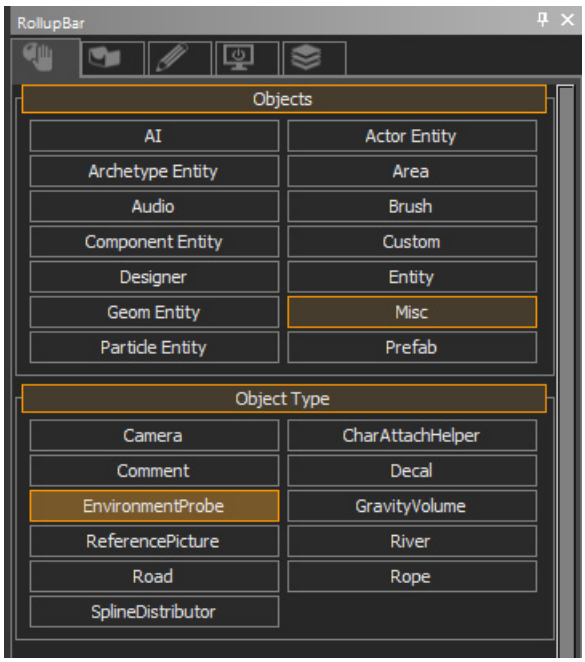
Environment Probes

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, place multiple environment probes to help adjust the space to the right visual quality. Always start with a default global environment probe. This tutorial helps you set up the global probe.

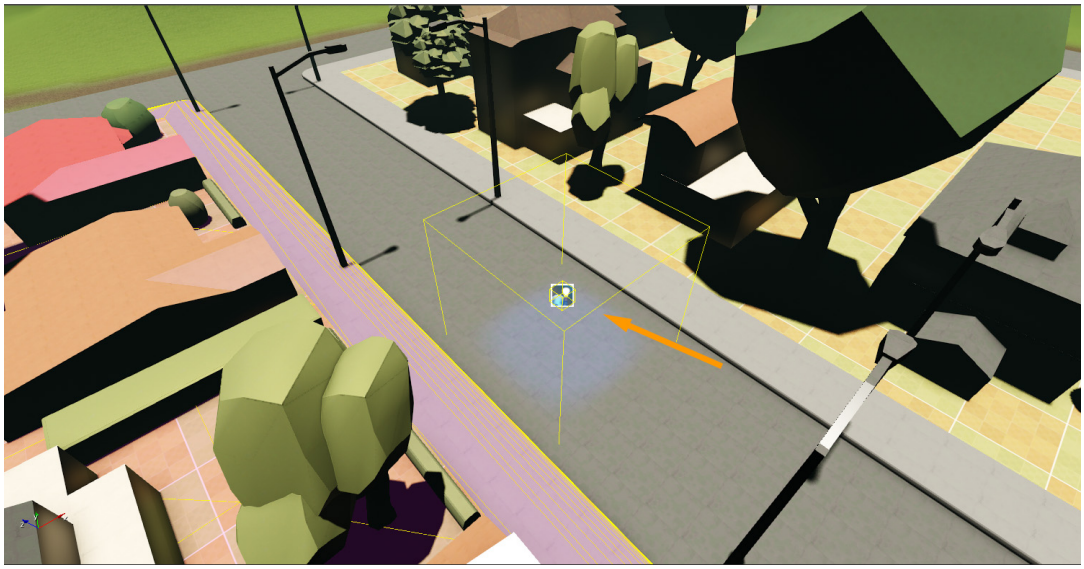
To add a global probe

1. In the **Rollup Bar**, click the **Objects** tab and then click **Misc**.
2. Under **Object Type**, click **EnvironmentProbe**. Move your mouse cursor into the **Perspective** viewport and then click to place the probe.



The environment probe appears as a square volume entity.

3. Position the probe approximately in the center of the level.



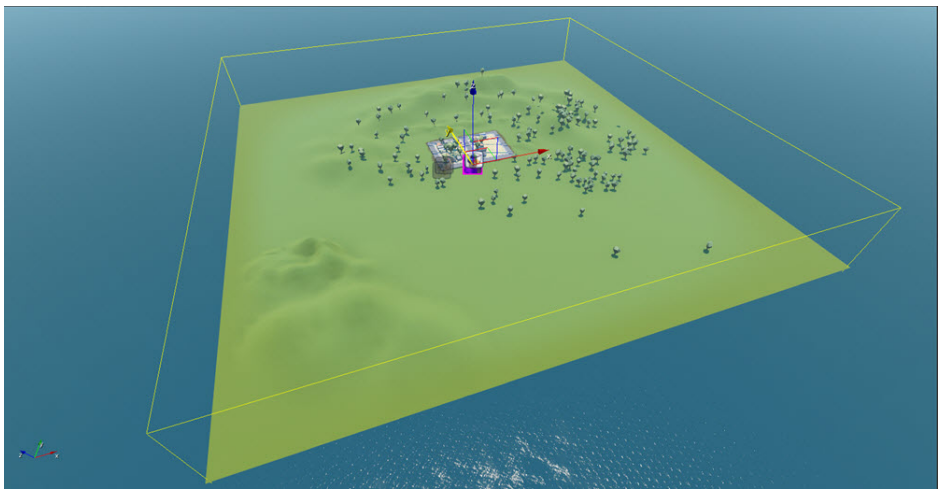
4. Under **EnvironmentProbe Properties**, apply the following settings:

   - **BoxSizeX** = `1024`
   - **BoxSizeY** = `1024`

- **BoxSizeZ** = **`250`**

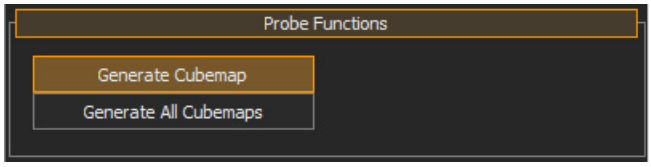
These settings sets size of the probe to the size of the entire map.

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 and generate the cubemap, select **Active** (in the EnvironmentProbe properties).
7. Under **Probe Functions** (above EnvironmentProbe Params), 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.



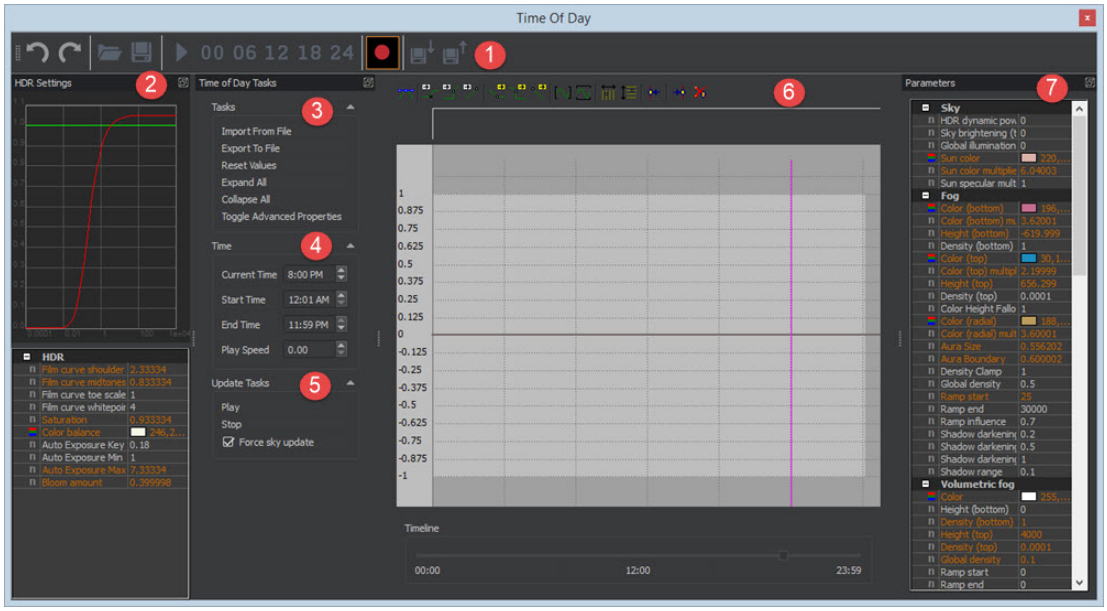
   Tip: Adjust the values for **Color: Diffuse** (RGB values) and **Color: DiffuseMultiplier** to see how it affects your scene.

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:

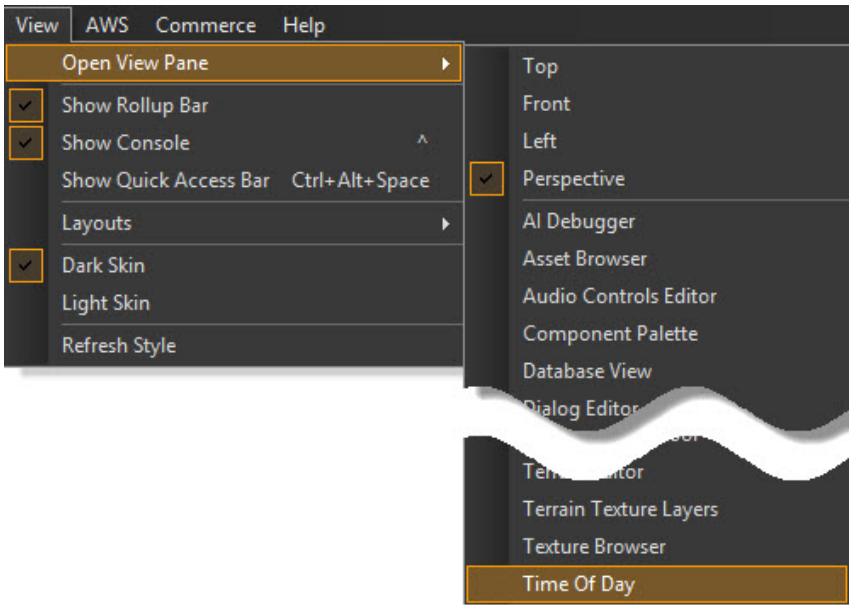
- **1: Editor toolbar** – Icon toolbar for most common functions: Undo, Redo, Import, Export
- **2: HDR Settings** – Settings to manage HDR lighting
- **3: Time of Day Tasks** – Management of basic tasks within the time of day editor
- **4: Current Time** – Display of times, start/end, and play speed
- **5: Update Tasks** – Controls to update the play or stop of time of day, based on play speed setting
- **6: Timeline editor** – Management of light settings along a 24-hour time cycle
- **7: Parameters** – Lighting settings to adjust time of day conditions



To adjust time of day

1. Click the **Time of Day** editor icon on the editor toolbar to open the **Time of Day** editor.

   You can also open it from the main menu: **View**, **Open View Pane**, **Time of Day**.


View
AWS
Commerce
Help
Open View Pane
Show Rollup Bar
Show Console
Show Quick Access Bar Ctrl+Alt+Space
Layouts
Dark Skin
Light Skin
Refresh Style
Top
Front
Left
Perspective
AI Debugger
Asset Browser
Audio Controls Editor
Component Palette
Database View
Terrain Texture Layers
Texture Browser
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 bar to the number **21** (indicating 21:00 or 9:00 p.m.). As you drag this bar, you can watch your scene in the **Perspective** viewport change from day to night.

   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.

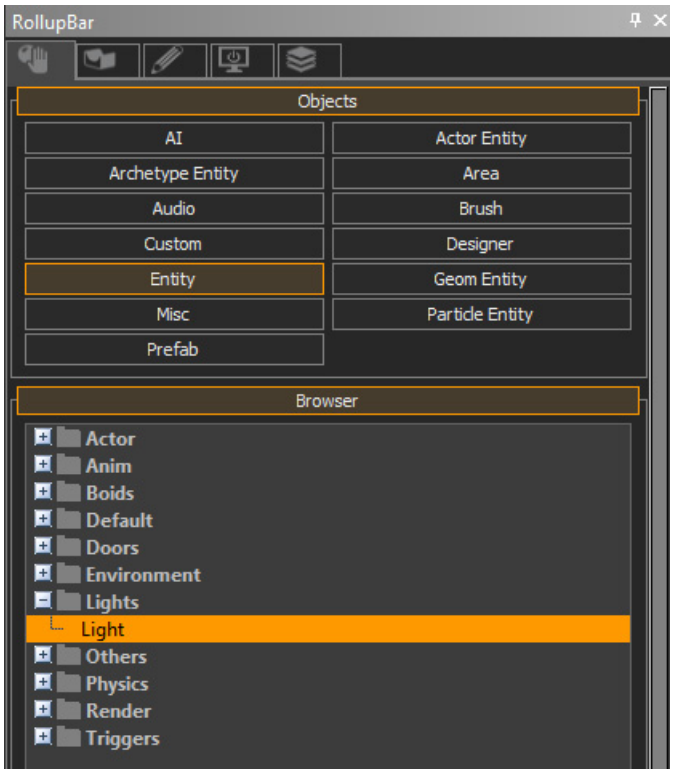
   Before moving to the next part of this tutorial, close the **Time of Day** editor.

Lights

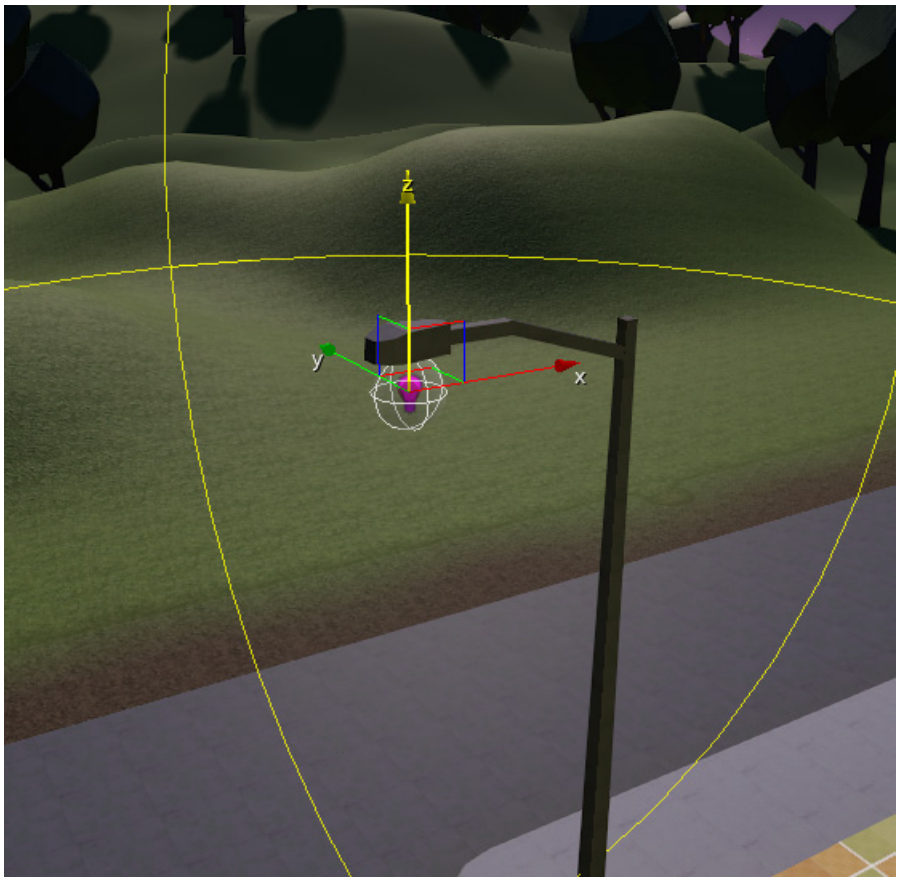
Now that your scene is set to nighttime, you can more clearly see the lights you are about to place.

To add lights

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



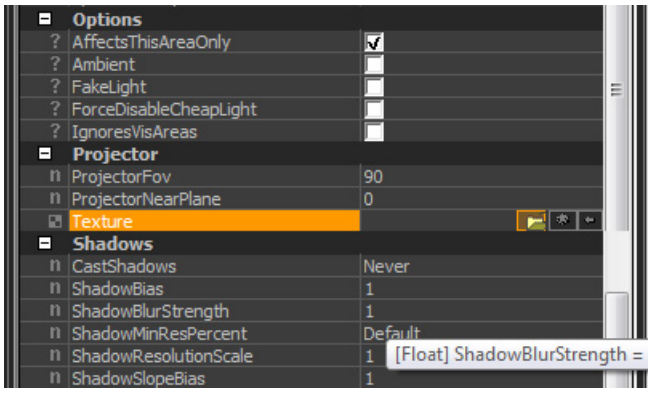
3. Position the light beneath one of the streetlamps.



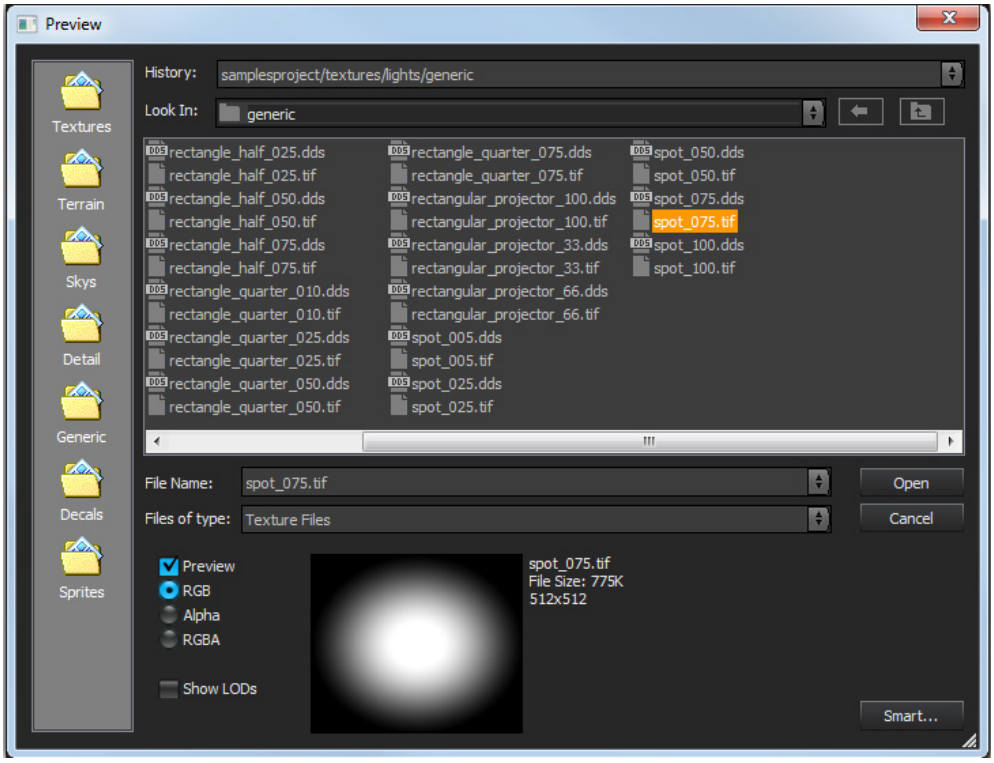
Tip
If you have trouble with precisely aligning the light, it is likely snapping to the grid. Turn off **Snap to Grid** in the editor toolbar, as shown in the following image, to move your light more precisely.

4. Because the light source is a streetlamp, it should be a spot light (rather than a fill light).

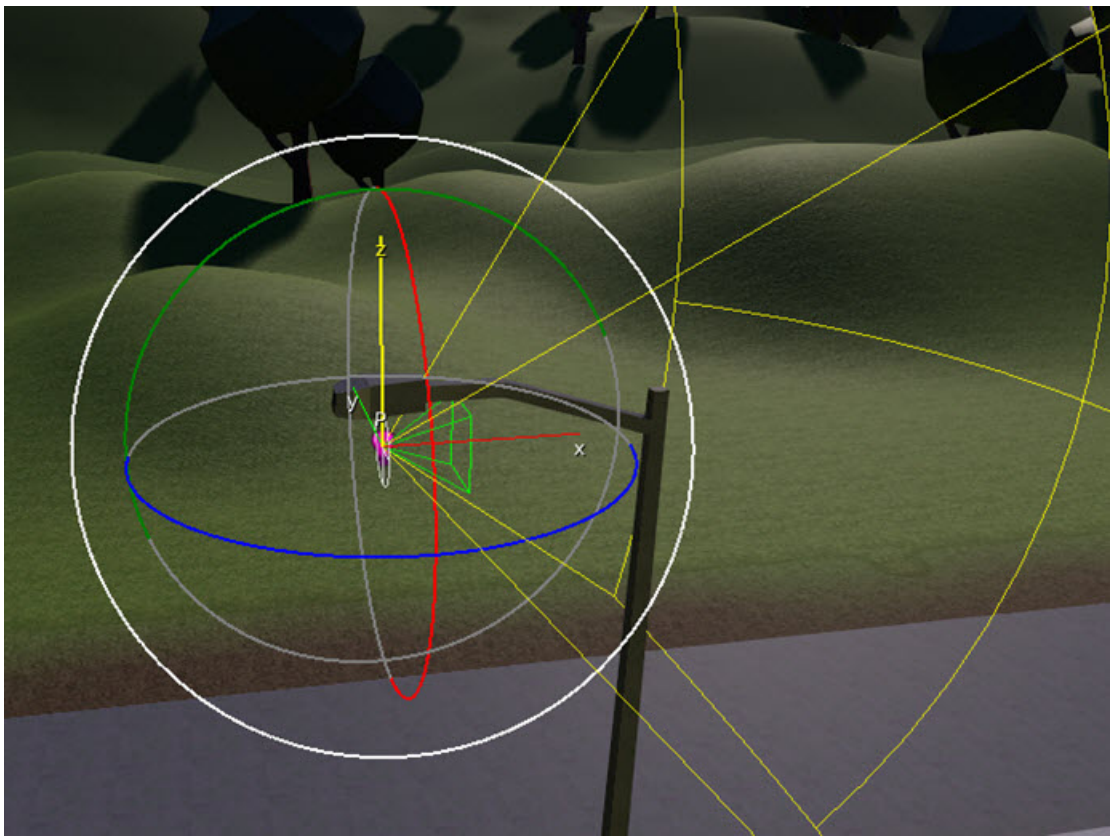
   To change the light from fill to spot: Beneath the **Entity Properties** header, find the **Projector** section. Under **Projector**, select **Texture** and then click the folder icon.


Options
AffectsThisAreaOnly
Ambient
FakeLight
ForceDisableCheapLight
IgnoresVisAreas
Projector
ProjectorFov
90
ProjectorNearPlane
0
Texture
Shadows
CastShadows
Never
ShadowBias
1
ShadowBlurStrength
1
ShadowMinResPercent
Default
ShadowResolutionScale
1
[Float] ShadowBlurStrength =
ShadowSlopeBias
1

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



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.



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:

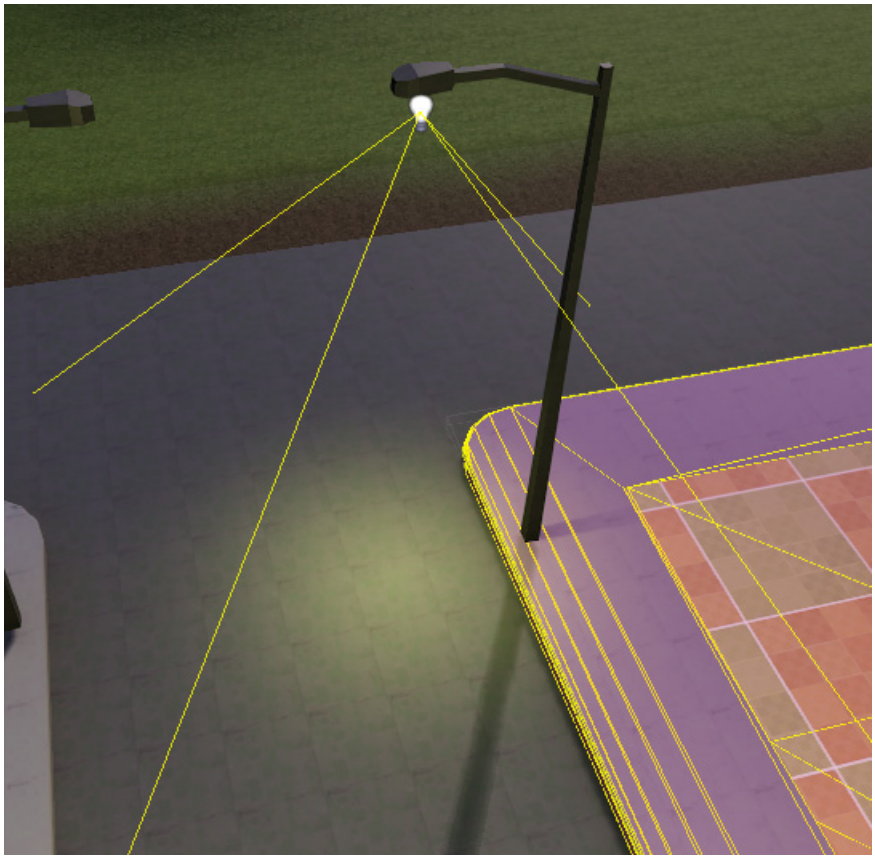
- **AttenuationBulbSize** – The size of the light bulb. This is the starting point where light begins to fall off exponentially. A value of 1 sets the light at full intensity for one meter before it begins to fall off. Adjust this size in relation to the diffuse multiplier to manage the brightness of the light source without entering unmanageable numbers.
- **Radius** – Surrounding area that the light affects as measured by the distance from the source.
- **Diffuse** – Color (RGB value) of the light.
- **DiffuseMultiplier** – Intensity of the diffuse color. Balance this value with the **AttenuationBulbSize** to define the balance of natural light levels.
- **ProjectorFov** – Field of view for the projection light.
- **CastShadows** – Makes the light cast a shadow based on the minimum selected config spec. A setting of **High**, for example, won't work on a config spec of **Low/Medium**. To ensure shadows are always cast, set this to **Lowspec**. Note that the **CastShadows** setting does not affect visual quality; use it to adjust performance based on the type of machine running the level.

7. For this tutorial, use the following settings:

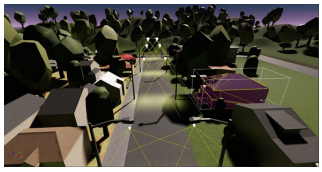
   - **AttenuationBulbSize** = **`6`**
   - **Radius** = **`20`**
   - **Diffuse Color** = **`250,250,150`**
   - **Diffuse Multiplier** = **`25`**
   - **ProjectorFov** = **`70`**
   - **CastShadows** = **Lowspec**

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

Your light should look something like this:



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

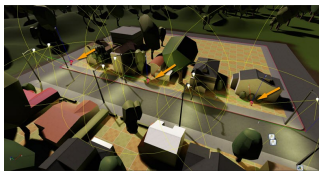


9. You can place additional fill lights to help light the scene, which is still somewhat dark.

   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 Viewport Header) to hide the entity icons. This allows you to view the scene without the visual clutter of entity icons.

    Save your level file.

Placing a Game Camera

To enable game play in a level, you must create and place a game play camera with character control and specify a start point. To save you time and get you to prototyping levels sooner, Lumberyard has created a game play camera (with supporting input controls) to use as a quick start to building a game play level.

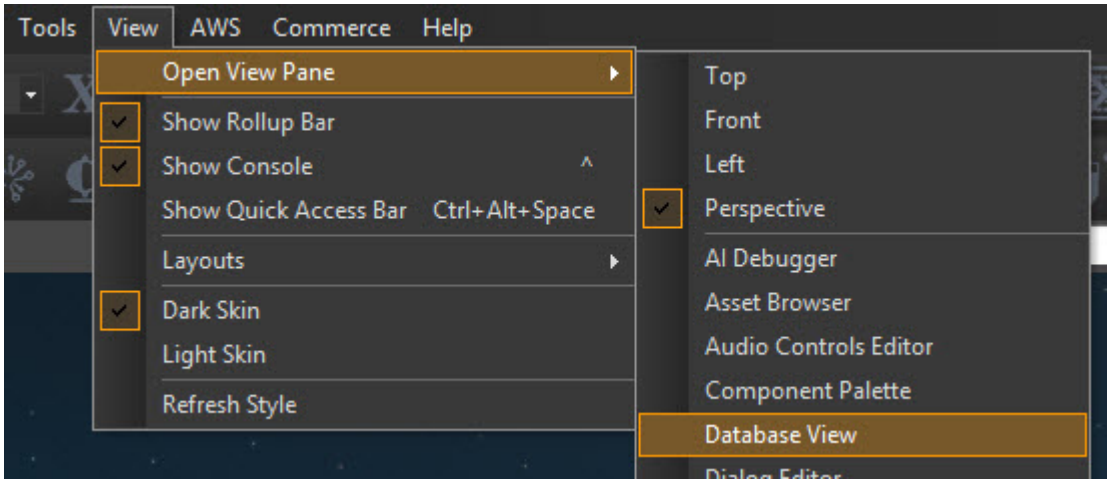
To place a game camera in a level

1. On the editor toolbar, click the **Database View** editor icon to open the **Database View** editor. You can also open it from the main menu by choosing **View**, **Open View Pane**, **Database View**.

   Editor Toolbar

   Main Menu

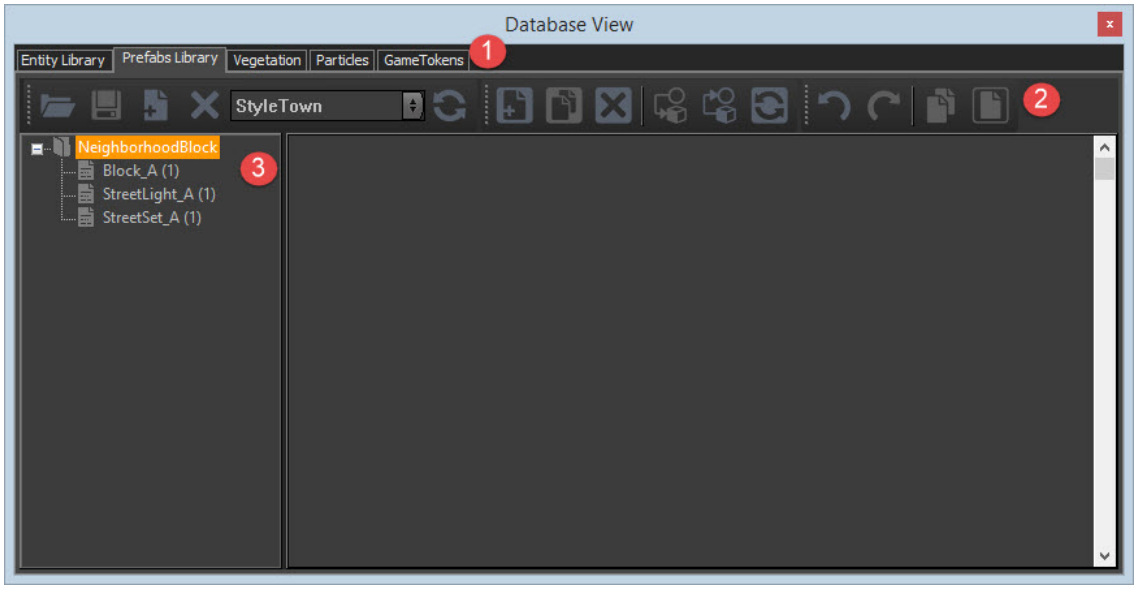
   

2. In the **Database View** editor, select the **Prefabs Library** tab.

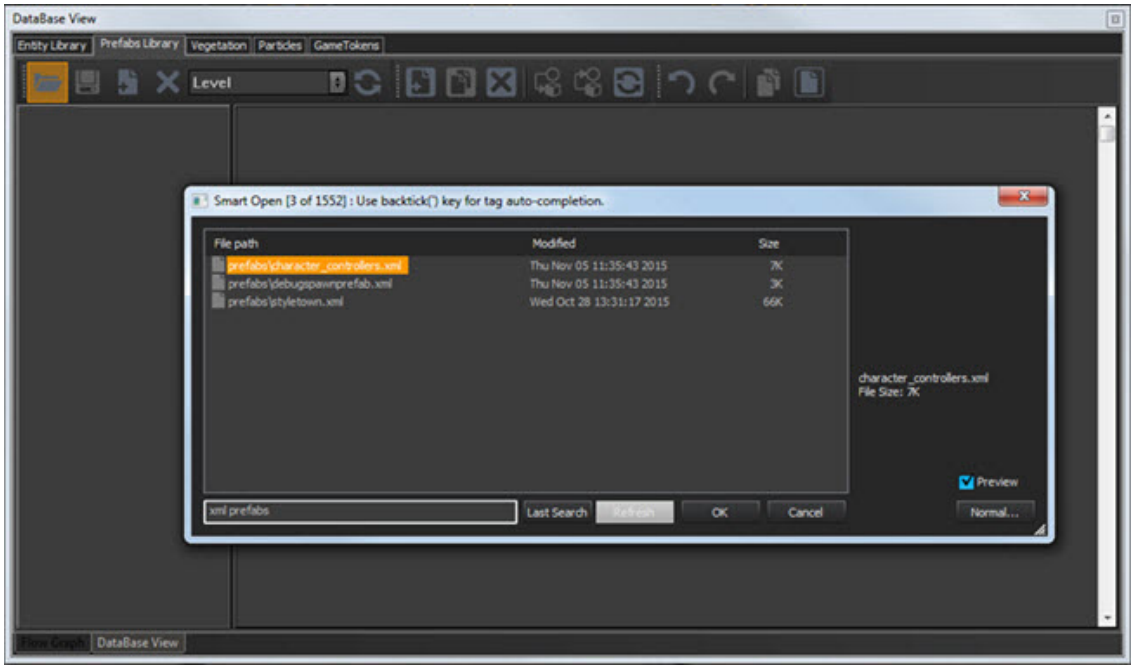
   ### The Database Editor for Prefabs has the following areas:

   - **1: Database tabs** – Database types to view and manage

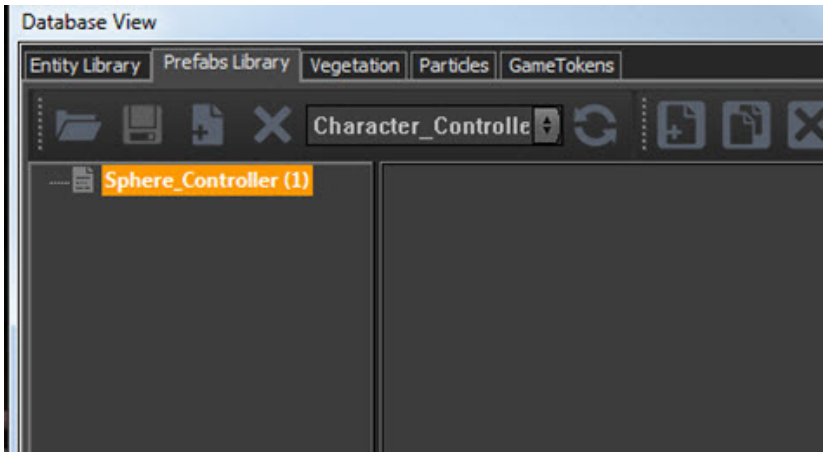
- **2: Editor toolbar** – Tools to open, save, add, remove Prefabs
- **3: File tree view** – View of the Prefabs available for use



3. Click the **Open Prefab** folder icon on the left side of the toolbar.
4. From the prefabs directory listing, select `prefabs\character_controllers.xml`.



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 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**, **Switch to Game**.
- On your keyboard, press **Ctrl+G**.

You are now running your first Lumberyard level.

To navigate, use the following controls:

- **Move forward/back** – W and S keys
- **Strafe left/right** – A and D keys
- **Turn left/right** – Mouse slide left/right
- **Jump** – Space bar
- **Camera Look** – Mouse movement

To exit game mode and return to edit mode

Press **Esc** on your keyboard.

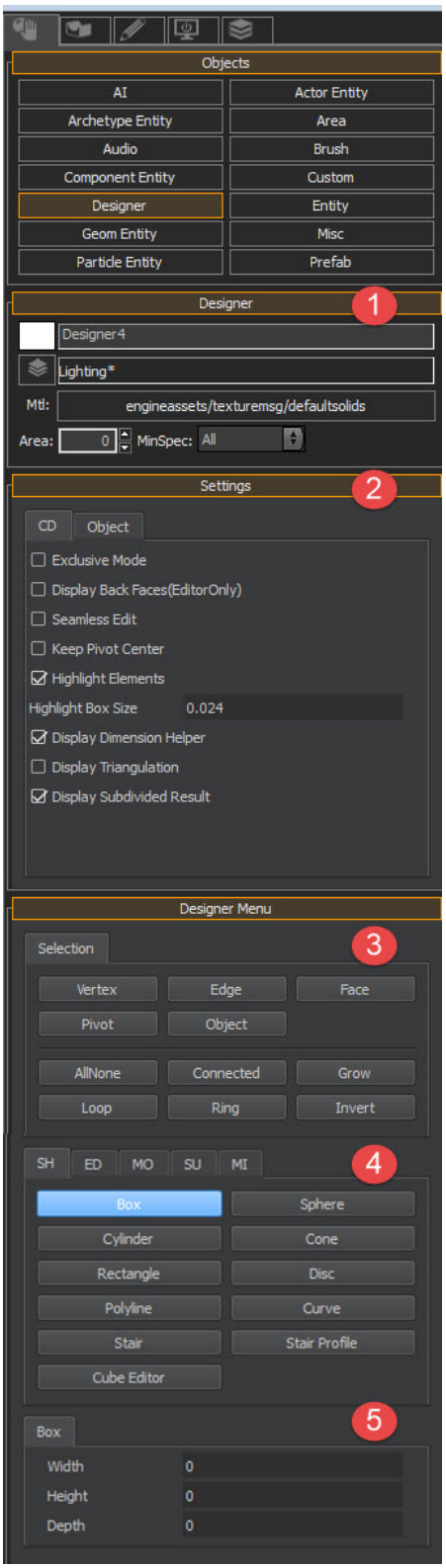
Creating Designer Objects

To create designer objects:

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

With the **Designer** tool, you can create and edit meshes, and perform UV Mapping and Exporting.

The Designer tool has the following areas:

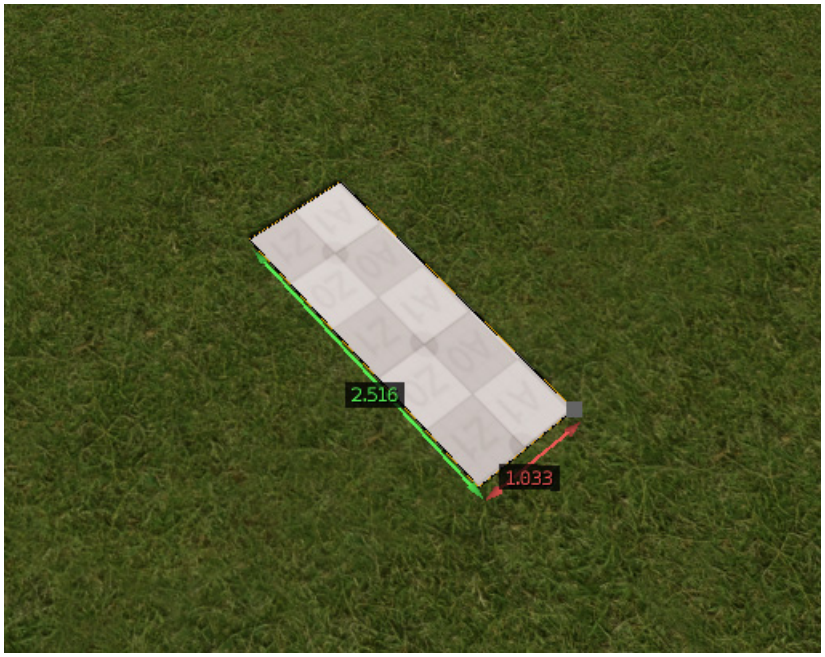


- **1: Standard Parameters** – Standard parameters such as name and layer location.
- **2: Settings** – Settings for toggling various object creation conditions. You can typically use the default settings.
- **3: Selection** – Selection (vertex, edge, or face). Also features secondary selection methods.
- **4: Create & Edit** – Select and edit shapes, modify objects, manage surface material, and export objects.
- **5: Mesh Dimensions** – Dimensions for subdivision of faces and numerical input of objects size and shape.

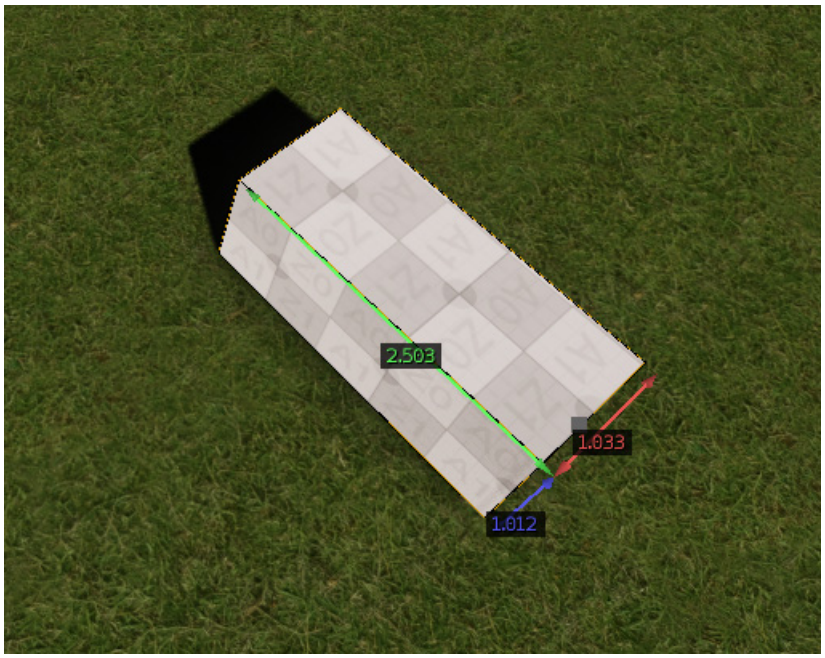
Objects you create with the **Designer** tool become a third type of mesh object within the editor. Though they are similar to brushes and entities, they have their own unique category. Designer objects do not have the same capabilities as entities, but they work just as well as brush objects when building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

2. For this tutorial, you'll create two Designer objects: A box and a sphere. Before you create an object, turn off **Snap to Grid** in the editor toolbar, or press **G** on your keyboard.
3. Under **Designer Menu**, click **Box**. 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.
4. 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.

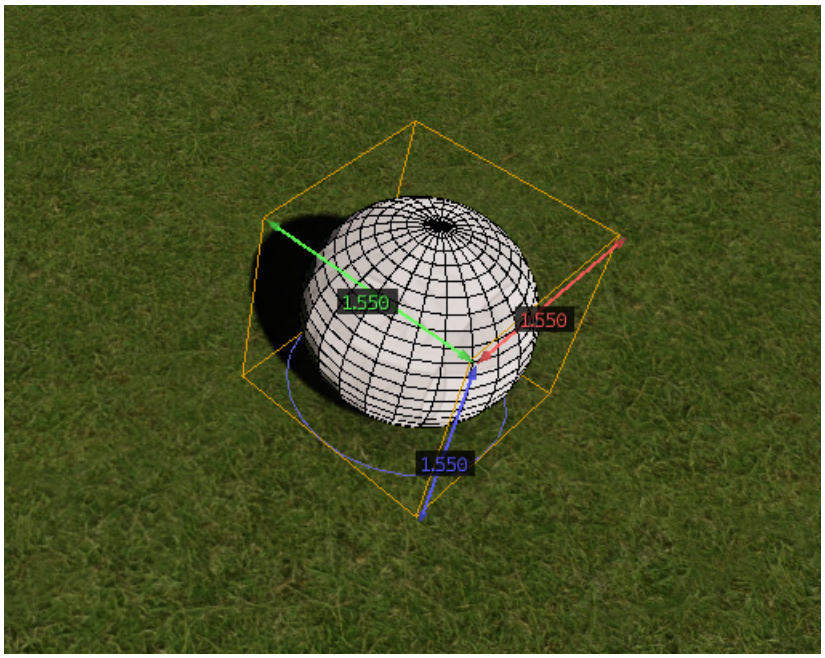


5. Move your cursor up to define the height at about 1 unit high. Click the left mouse button to lock the height. You have created your objects.



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**, in the create & edit section (with tabs **SH**, **ED**, **MO**, **SU**, **MI**), click **Sphere**.
8. In the **Perspective** viewport, drag the mouse to create a sphere that is about 1.5 units around.



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

9. To 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.

Materials

This tutorial teaches you how to create materials and apply those materials to a designer object, and then export that object.

Create a New Material

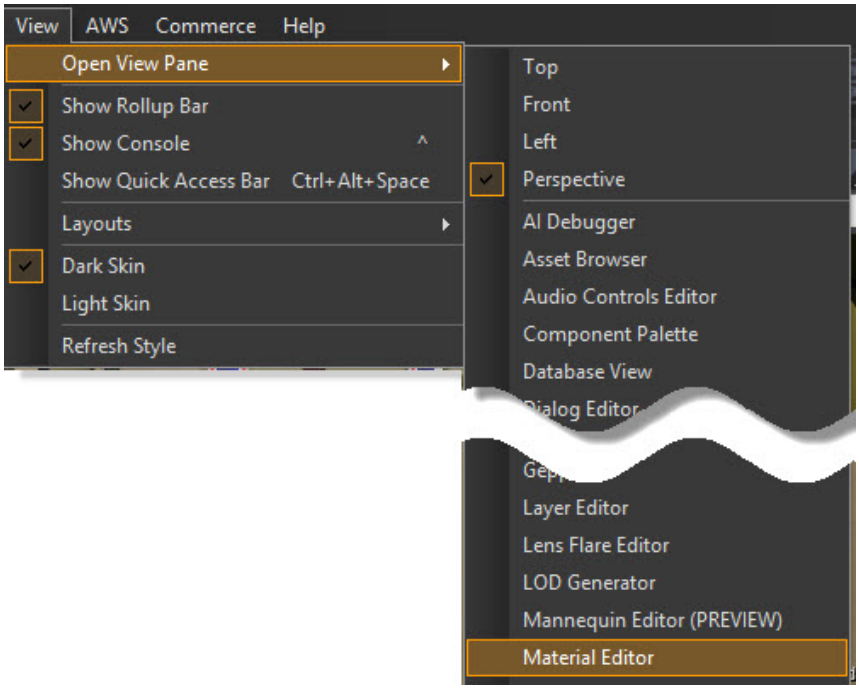
To create a new material, you'll use the **Material Editor**. The **Material Editor**'s interface is described in Terrain Painting (p. 31).

To create a new material

1. **To open the Material Editor**

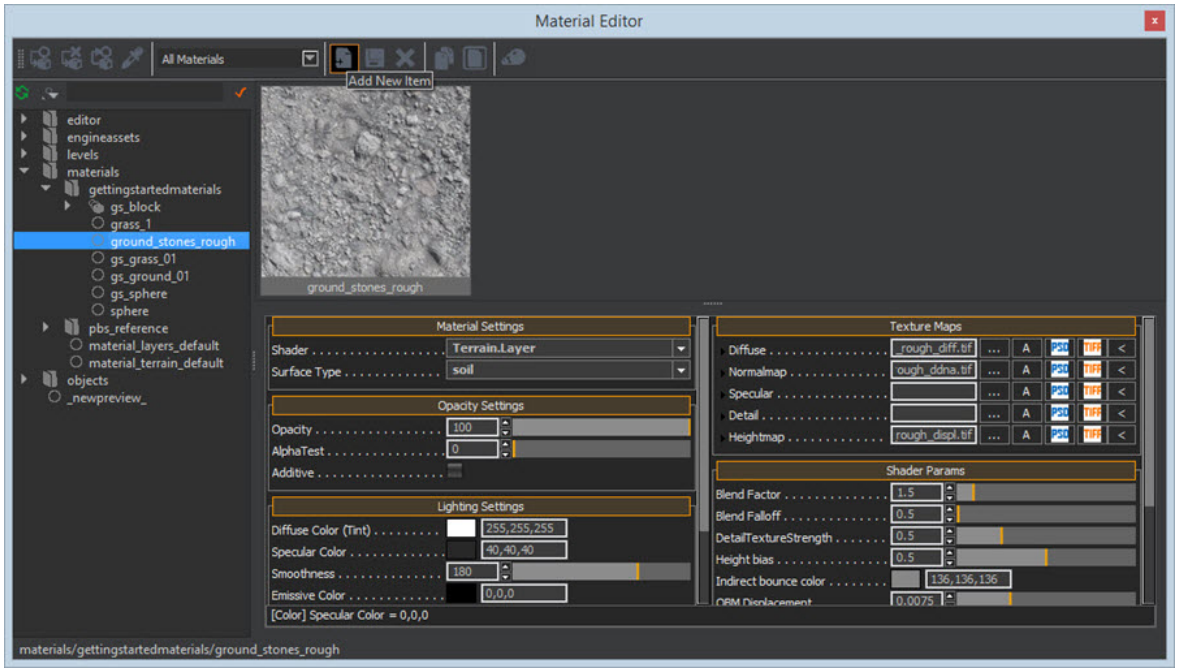
   Do one of the following:

   - On the editor toolbar, click the **Material Editor Dialog** icon
   - From the main menu, open **View**, **Open View Pane**, **Material Editor**



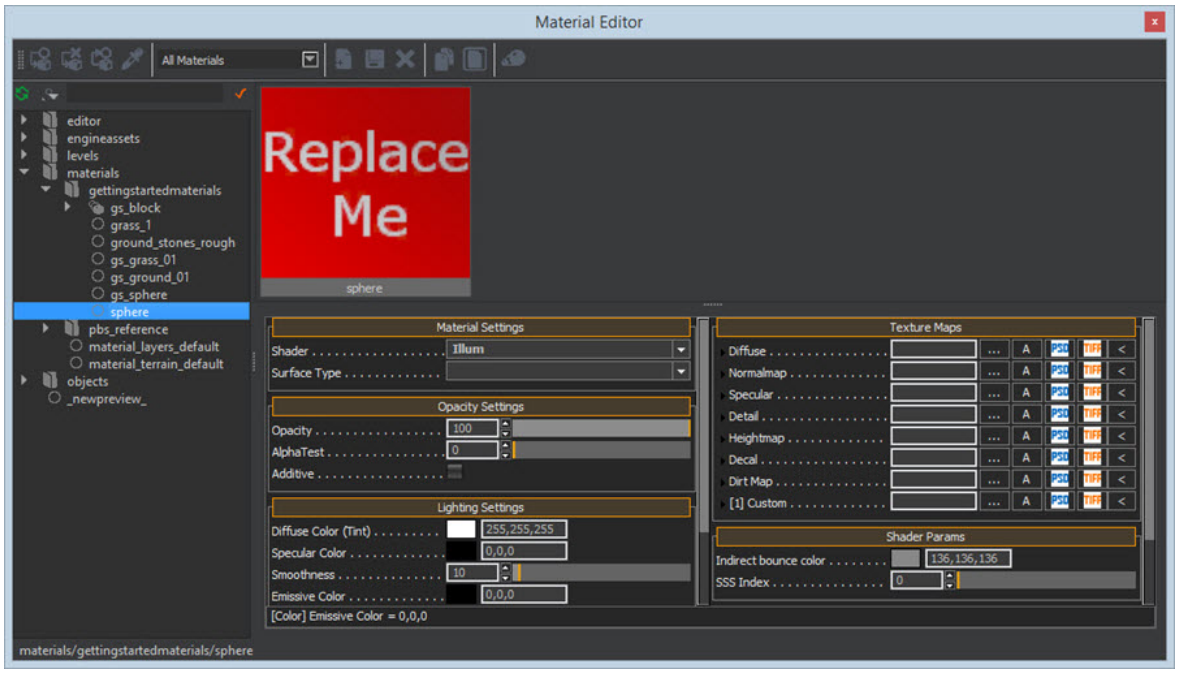
2. Expand `materials\GettingStartedMaterials`.

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

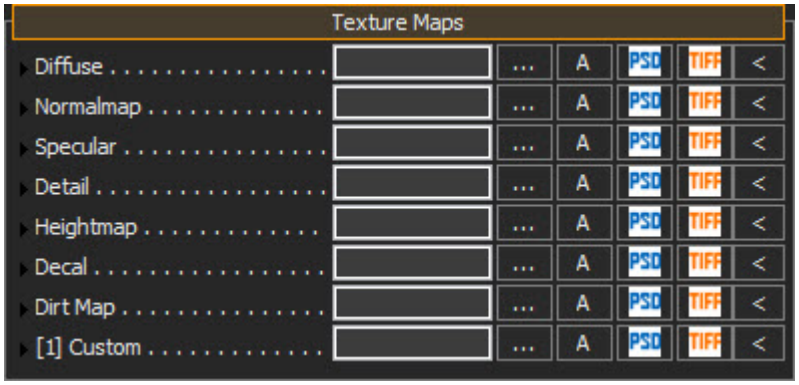


4. Navigate to `Materials\GettingStartedMaterials`. In the **File name** field, type **`sphere`**. Click **Save**.
5. Click **Add New Item** again. In the **File name** field, type **`block`**. Click **Save**.

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



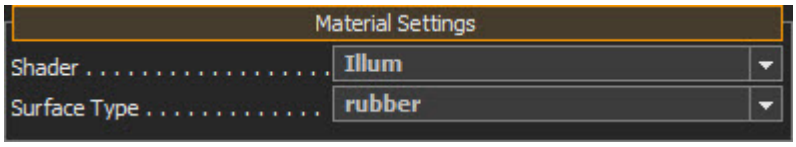
7. In the **Materials Properties and Settings** (lower right area), scroll down and find **Texture Maps**. On the **Diffuse** line, click the ellipsis '...'.



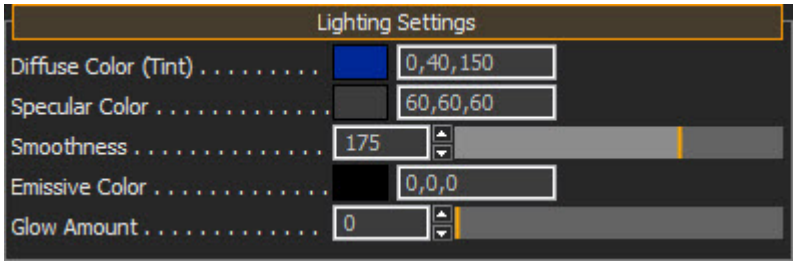
8. Navigate to `SamplesProject/textures/GettingStartedTextures`. Open `White.tif`.

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

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



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`**.



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

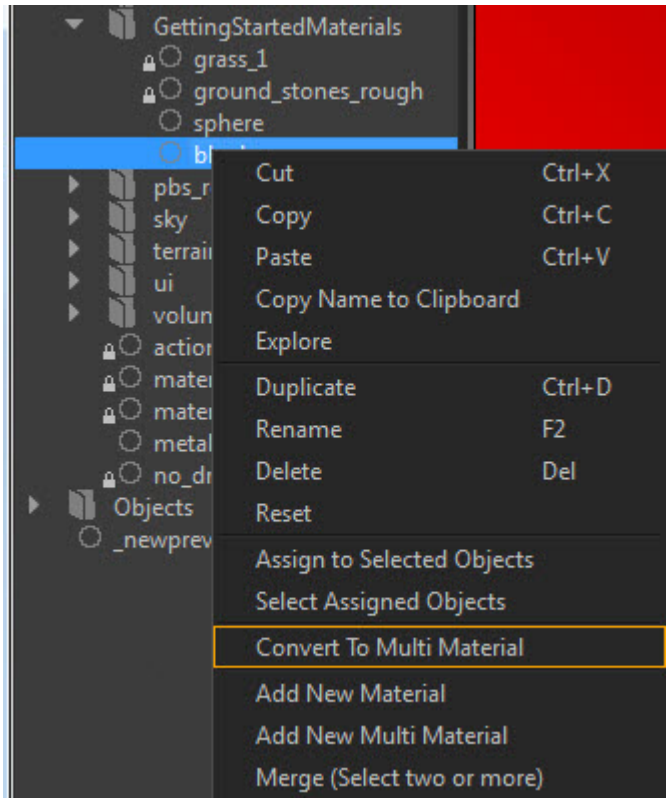
Multi-Material

Now you'll turn the previously-created 'block' material into a multi-material. A multi-material is a single material file with multiple materials.

A multi-material enables you to map multiple materials to a single object (for example, a block). Multi-materials are useful for whitebox prototyping 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 multi-material, you can create the siding, roof, window, and door textures and then assign the correct material id to that single mesh to represent a house.

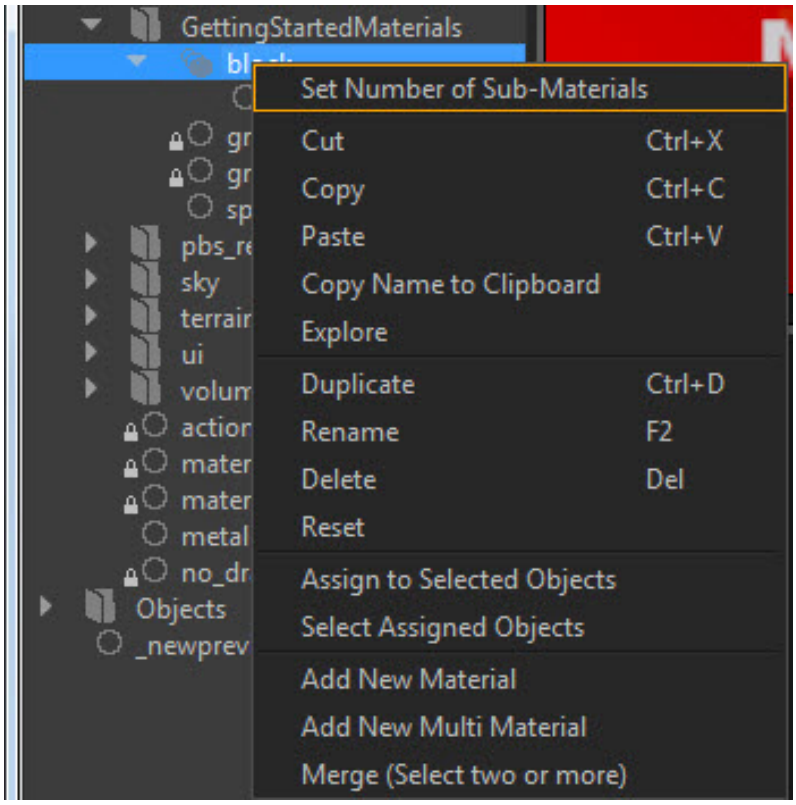
To create a multi-material

1. Right-click on the **block** material you previously created.
2. Select **Convert to Multi Material**.

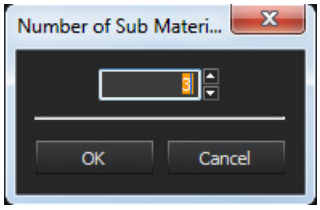


Beneath your initial **block**, **[1] block** appears.

3. Select **block** (not **[1] block**). Right-click and select **Set Number of Sub-Materials**.

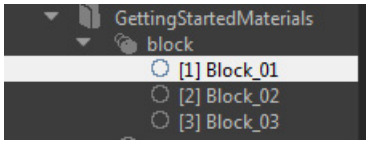


4. In the **Number of Sub Materials** dialog, enter **3**. Click **OK**.



Three new sub-materials appear beneath **block**.

5. Right-click on each sub-material and rename them **`block_01`**, **`block_02`**, and **`block_03`**.

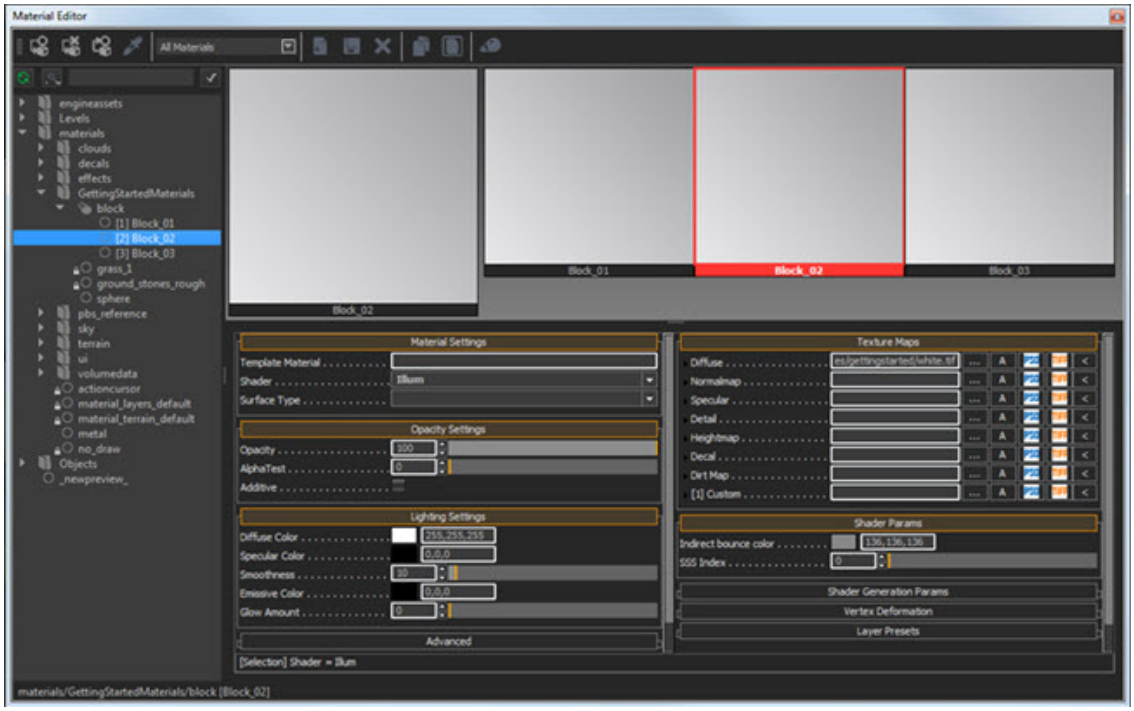


6. Select **block_01**. On the lower right, under **Texture Maps**, find **Diffuse** and click the ellipsis '...'.
7. Navigate to `SamplesProject\textures\GettingStartedTextures`. Open `White.tif`.

8. Repeat the previous step for **block_02** and **block_03**.

   All sub-materials now have a diffuse texture map assigned to them.

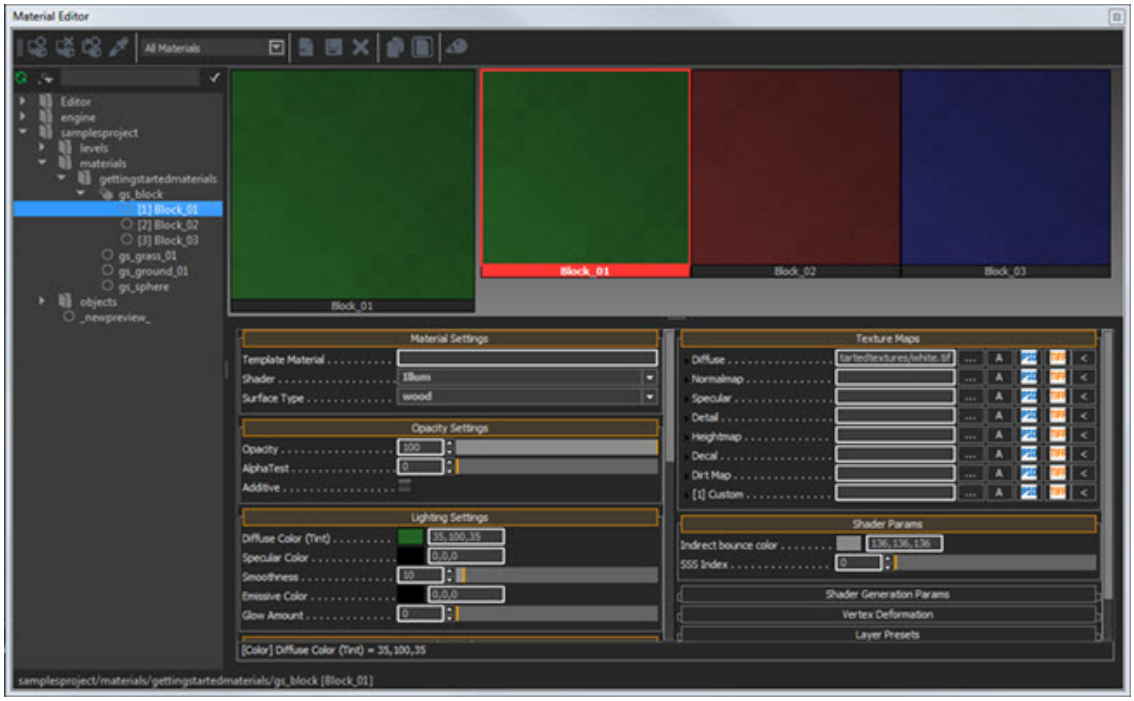
   The material preview window shows the three sub-materials on the right; the selected sub-material is also displayed (larger) on the left-hand side.



9. Select sub-material **block_01** and set the **Diffuse** color to **`35,100,35`**.
10. Select sub-material **block_02** and set **Diffuse** color to **`100,35,35`**.
11. Select sub-material **block_03** and set **Diffuse** color to **`35,35,100`**.

12. Under **Material Settings**, set **Surface Type** to wood for all three sub-materials.

    The block sub-material should look like the following:

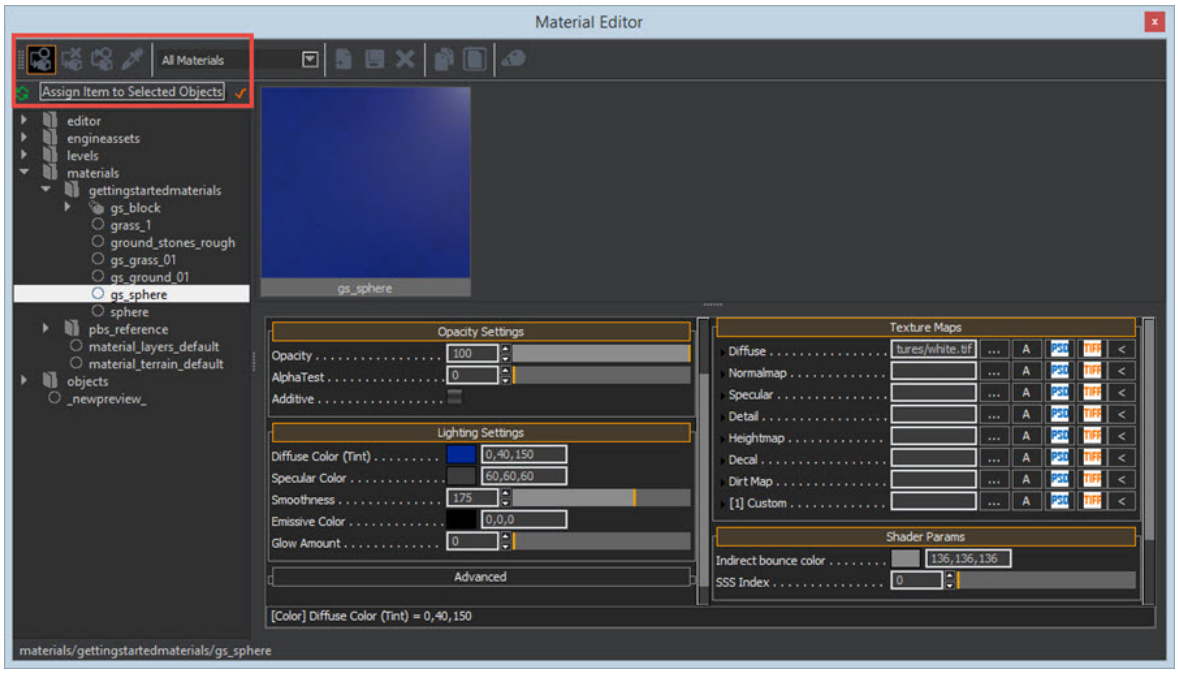


Assigning Material to Objects

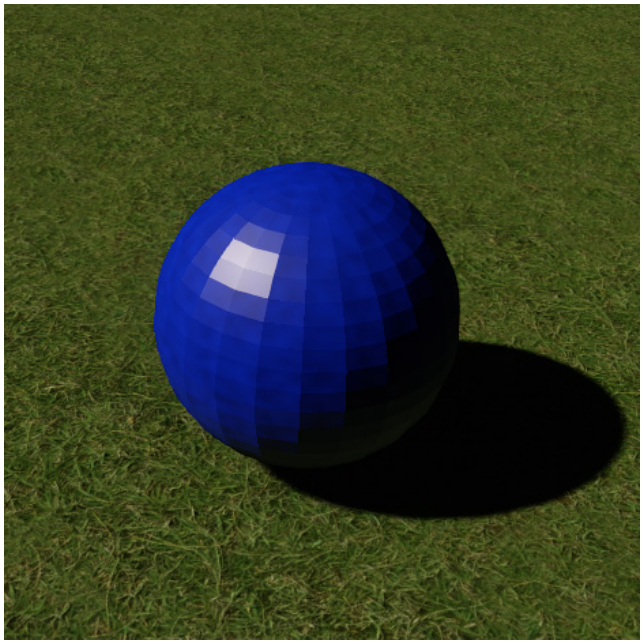
Now that you have made your materials, you can assign them to the designer objects you created in the previous tutorial.

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 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**.

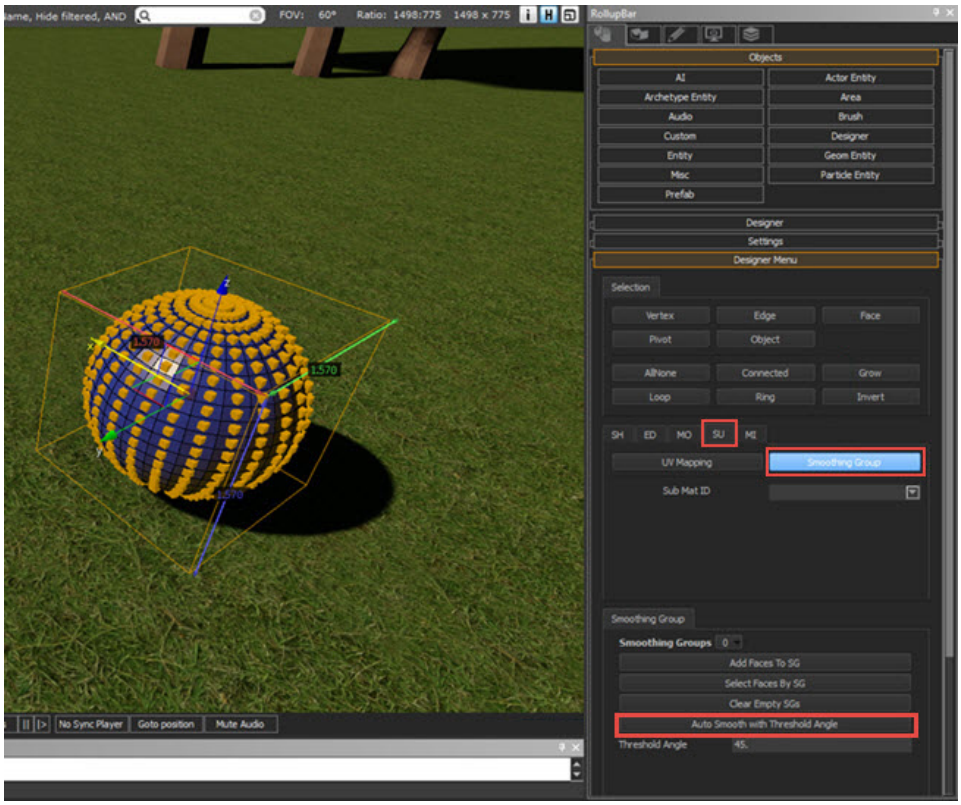


The sphere you created should now look something like this.



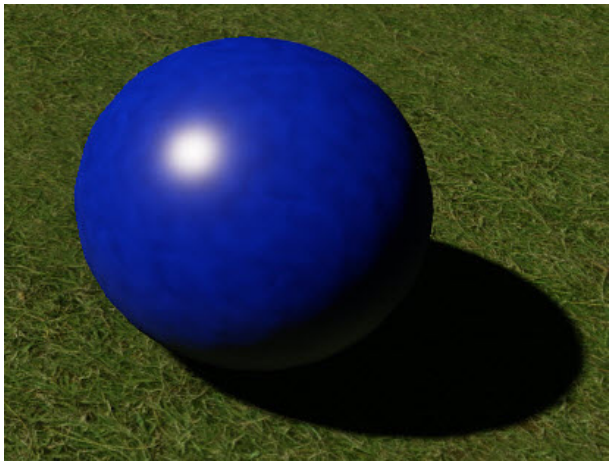
4. The designer sphere's faces are faceted. You can smooth the faces of the sphere. Select the sphere and open the **Designer** tool.
5. Under **Designer Menu**, select the **SU** tab.

6. Click **Smoothing Group**, and drag over the faces of the sphere to select them:



7. Below the **SU** tab, a **Smoothing Group** tab appears. Click **Auto Smooth with Threshold Angle**.

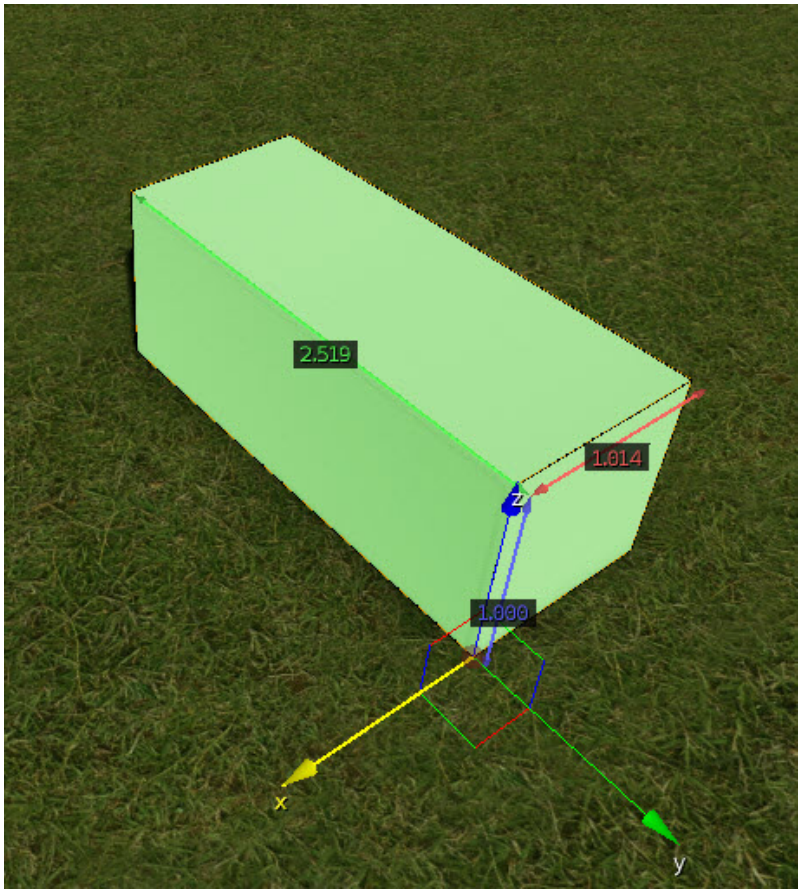
   This smooths out the sphere so that it appears smooth and round.



Apply a Multi-Material to an Object

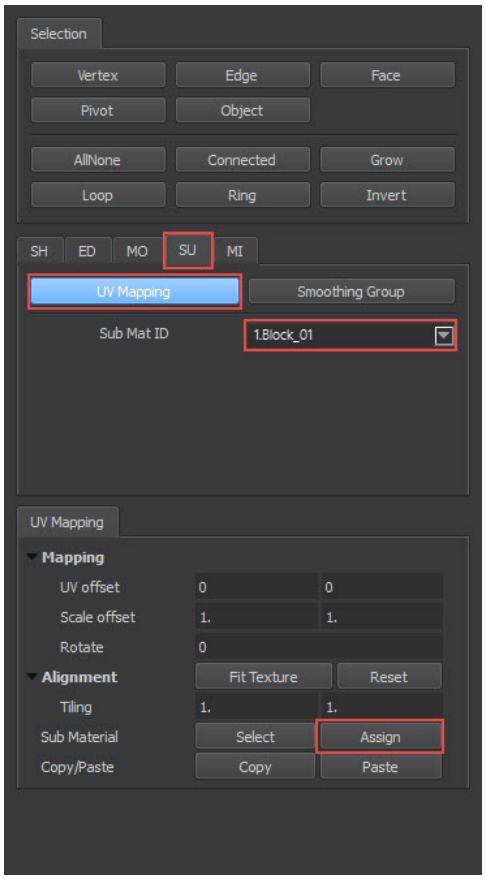
To apply to the block the multi-material you created:

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. Select the **Designer** tool.

4. Under **Designer Menu**, select the **SU** tab. Click **UV Mapping Button**.



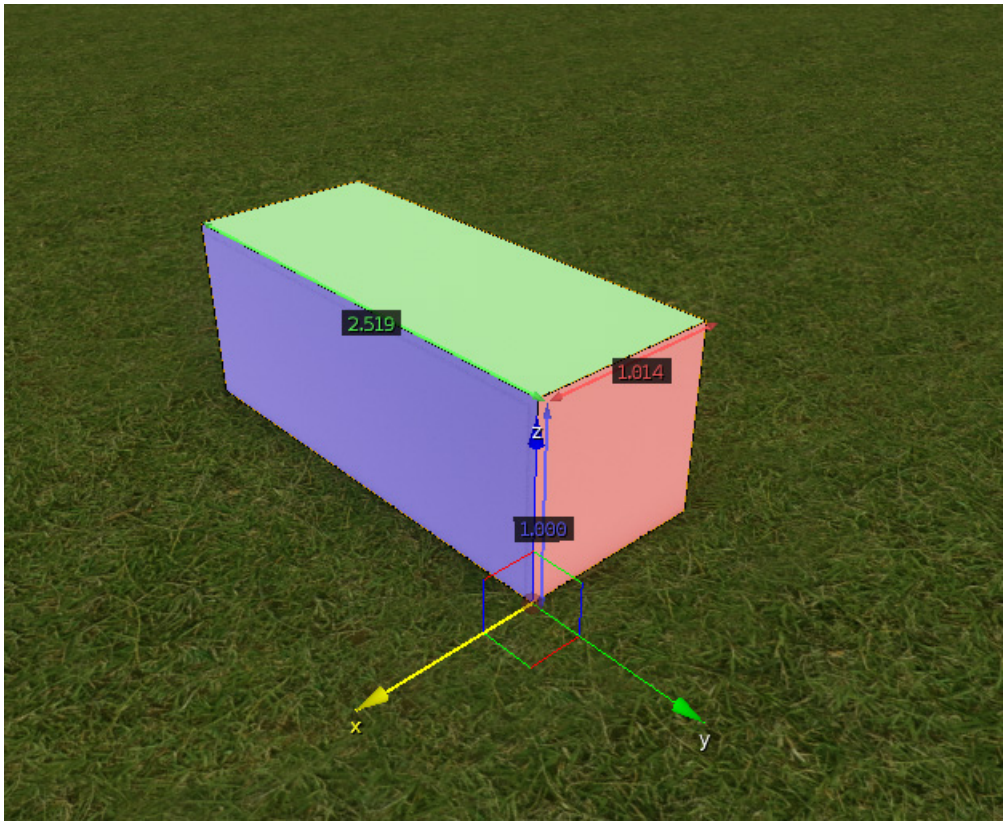
5. The faces are now selectable. Select the faces on each end of the block.

   To select more than one face, click on the first face, then hold down the **Ctrl** key to select additional faces.
6. With selected faces, open the **Sub Mat ID** menu and select **2.Block_02**.
7. In the **UV mapping** tab at the bottom, click **Assign**.

   The sub-material #2 is now assigned to the ends of the block.

8. Select the two sides of the box.
9. Open the **Sub Mat ID** menu and select **3.Block_03**. Click **Assign**.
10. Click **Object** in the **Selection** tab to close the tool.

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

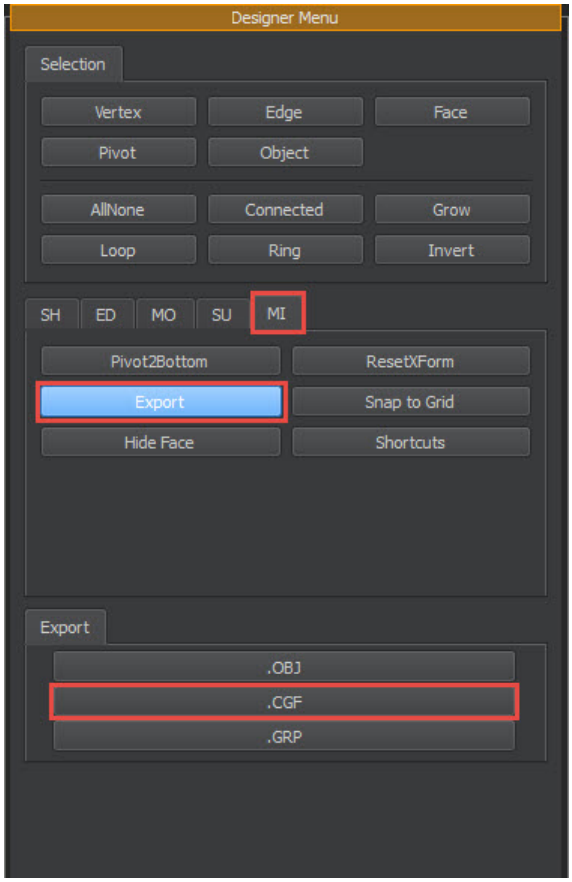


Exporting Objects

After creating the objects and applying material, you can now export these objects and use them for our physics entities that need to be set up.

To export the objects

1. Select the block in the **Perspective** viewport.
2. Select the **Designer** tool. Under **Designer Menu**, select the **MI** tab.
3. Click **Export**.
4. In the **Export** tab, click **.CGF**.



5. In the **Save As** dialog box, in the **File name** field, type **`block`**. Save it in the directory `SamplesProject\Objects\GettingStartedAssets`. Click **Save**.
6. Do the same steps for the sphere, but name it **`sphere`**.

Physics

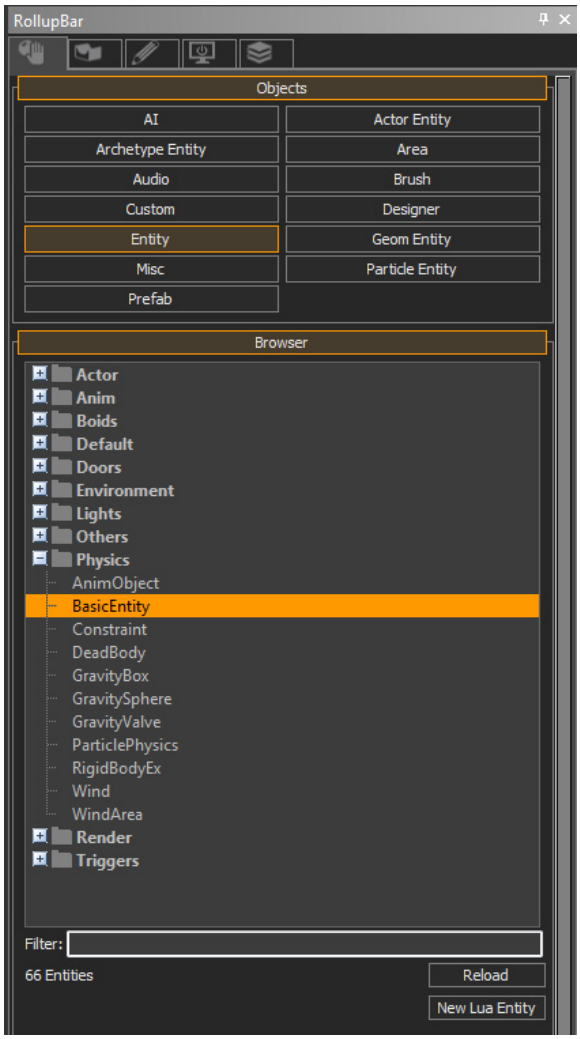
This tutorial steps you through how to to set up a physics object to work in the level and then activate during a game scripted event.

Create a physics block

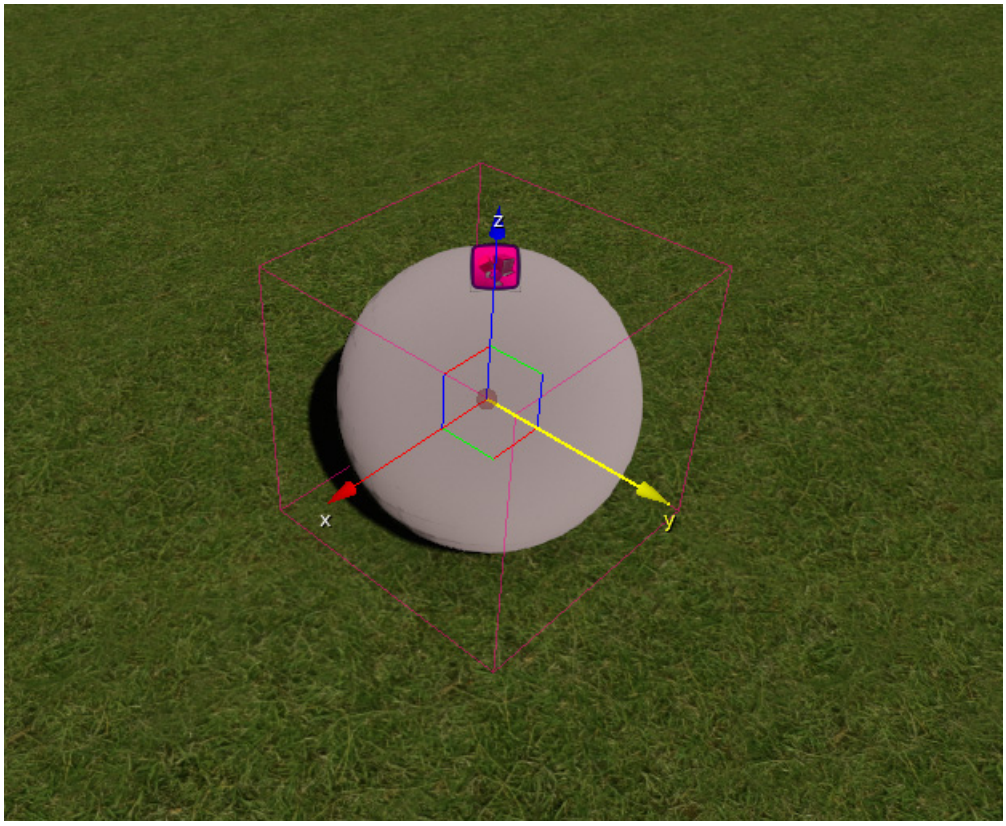
In this section, you'll learn how to add a physics entity to the level and assign it the block mesh created in previous tutorials.

To add a physics entity

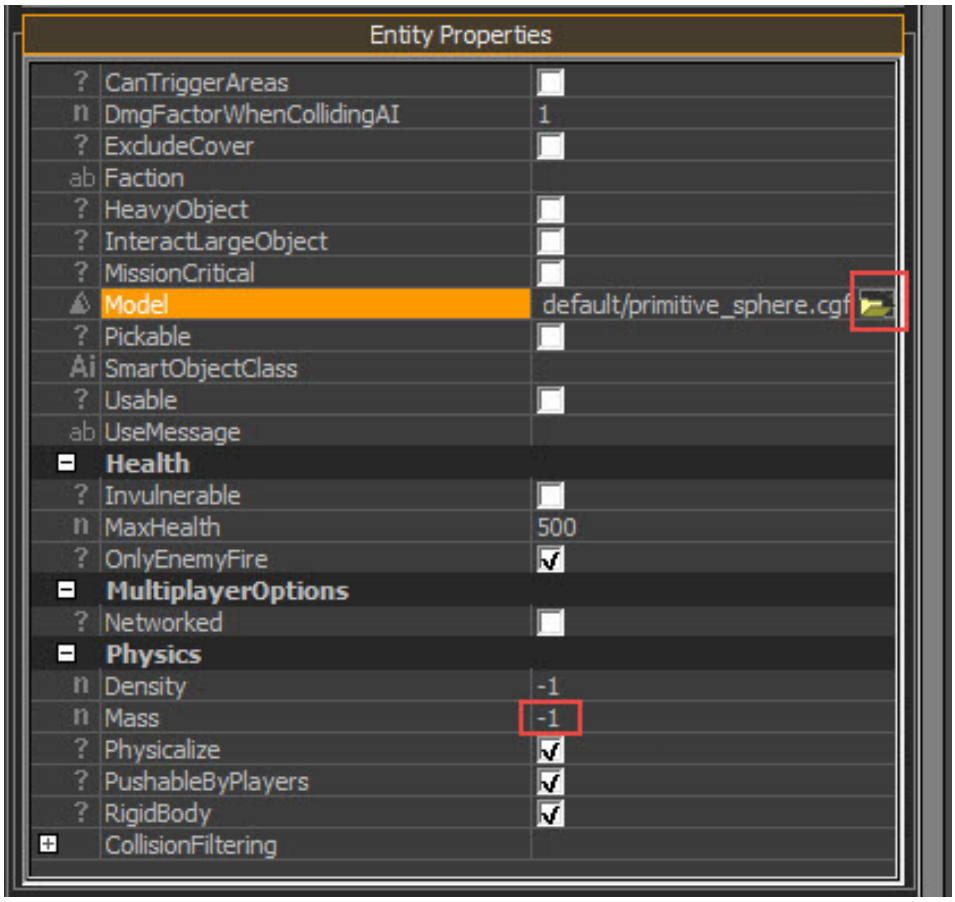
1. From the **Rollup Bar**, on the **Objects** tab, click **Entity**.
2. Under **Browser**, expand the **Physics** folder. Drag **BasicEntity** into the **Perspective** viewport.



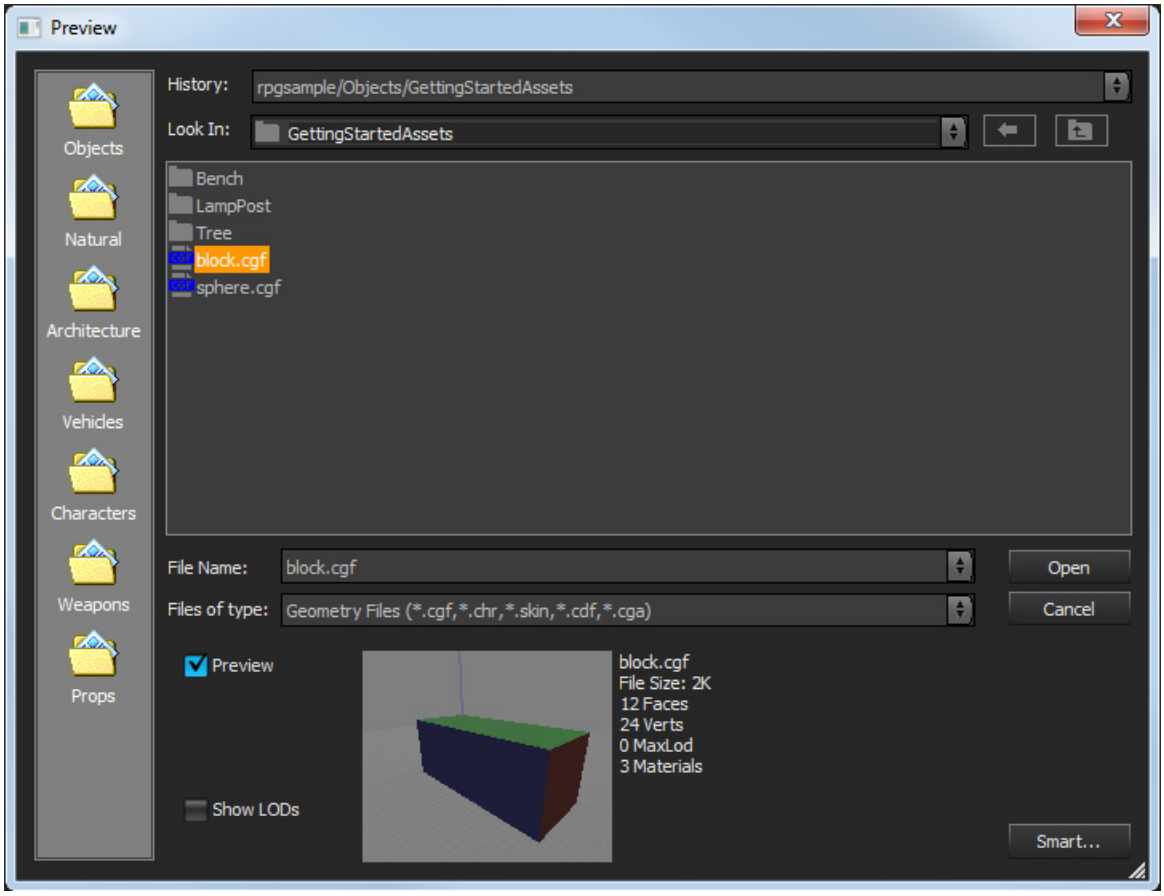
The **BasicEntity** object appears as a sphere.



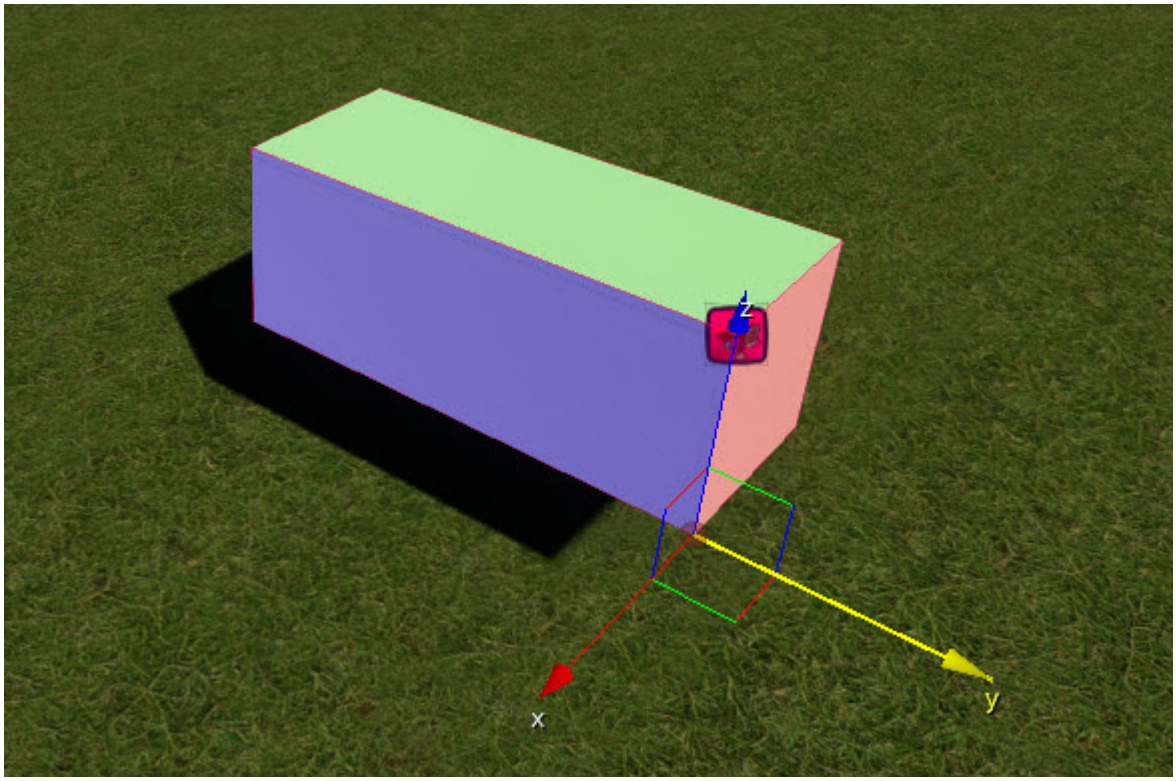
3. Under **Entity Properties**, select the attribute **Model** and click on the folder icon.


Entity Properties
CanTriggerAreas
DmgFactorWhenCollidingAI
1
ExcludeCover
Faction
HeavyObject
InteractLargeObject
MissionCritical
Model
default/primitive_sphere.cgf
Pickable
SmartObjectClass
Usable
UseMessage
Health
Invulnerable
MaxHealth
500
OnlyEnemyFire
MultiplayerOptions
Networked
Physics
Density
-1
Mass
-1
Physicalize
PushableByPlayers
RigidBody
CollisionFiltering

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



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.

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

Create an Archetype Entity

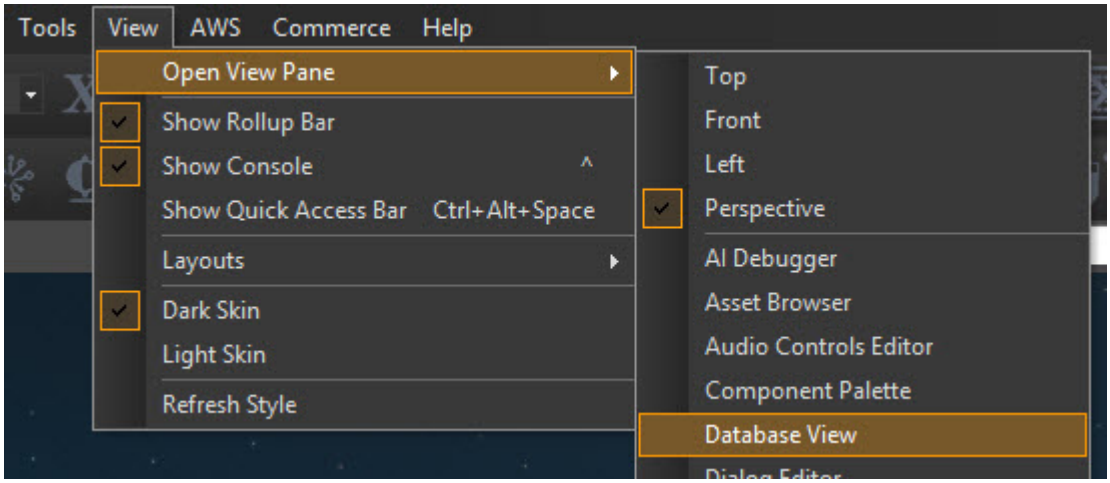
Now you'll create a physics object that spawns into the level when activated through a flow graph scripted event. To do this, you define an Archetype Entity in the **Database View** editor. Archetype entities are like regular entities, but they have their attributes defined before they are placed in a level. 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.

To define an archetype entity:

1. Open the **Database View** editor.

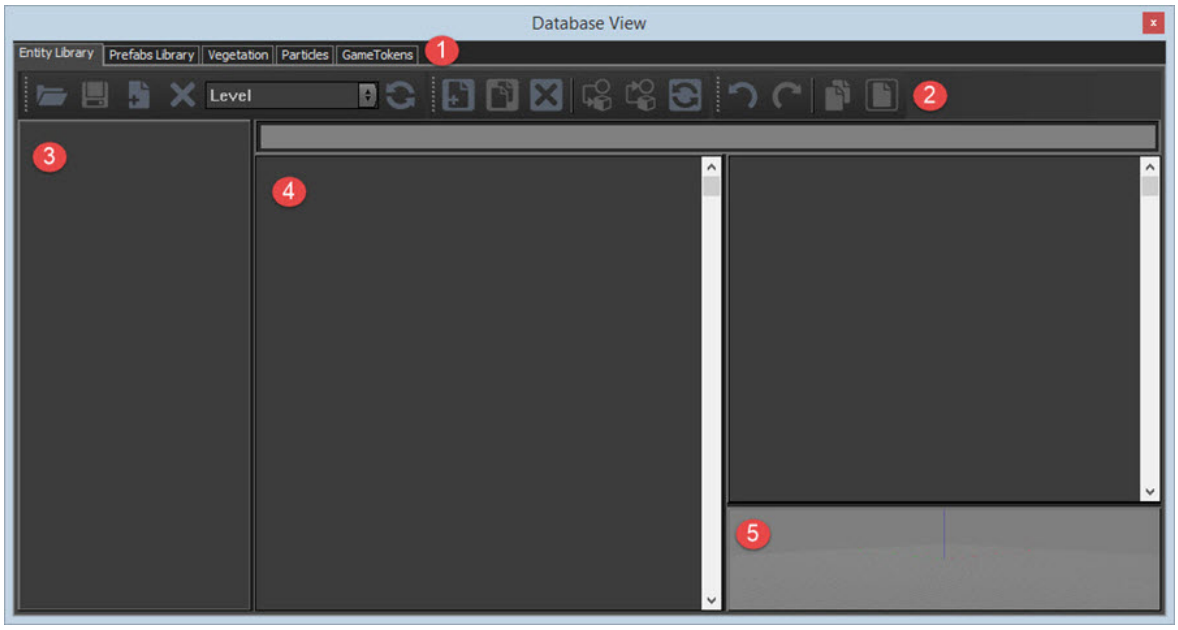
   To do this, do one of the following:

   - On the editor toolbar, click **Database View**
   - From the main menu, open **View**, **Open View Pane**, **Database View**.



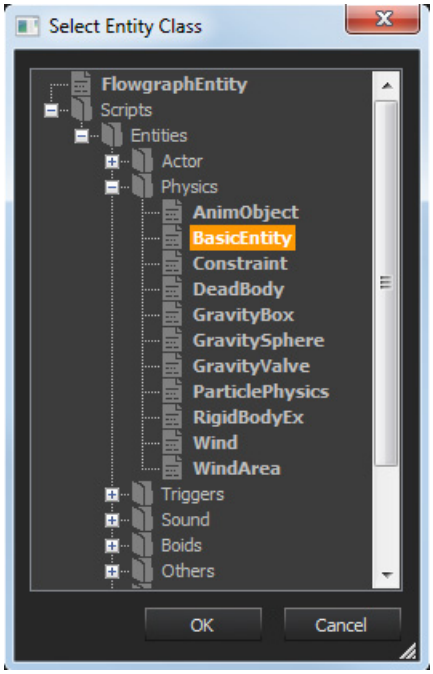
The Database Editor for Archetype Entities has the following features:

- **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 window** – Preview window for the selected archetype entity

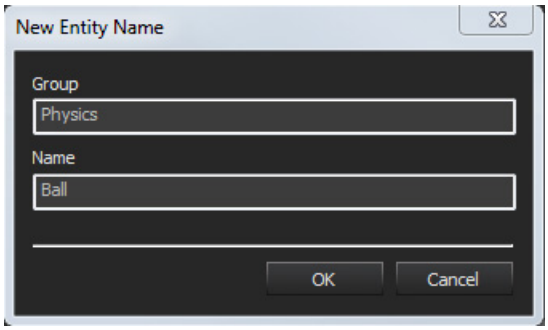


2. In the toolbar, click **Add New Item**  .

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



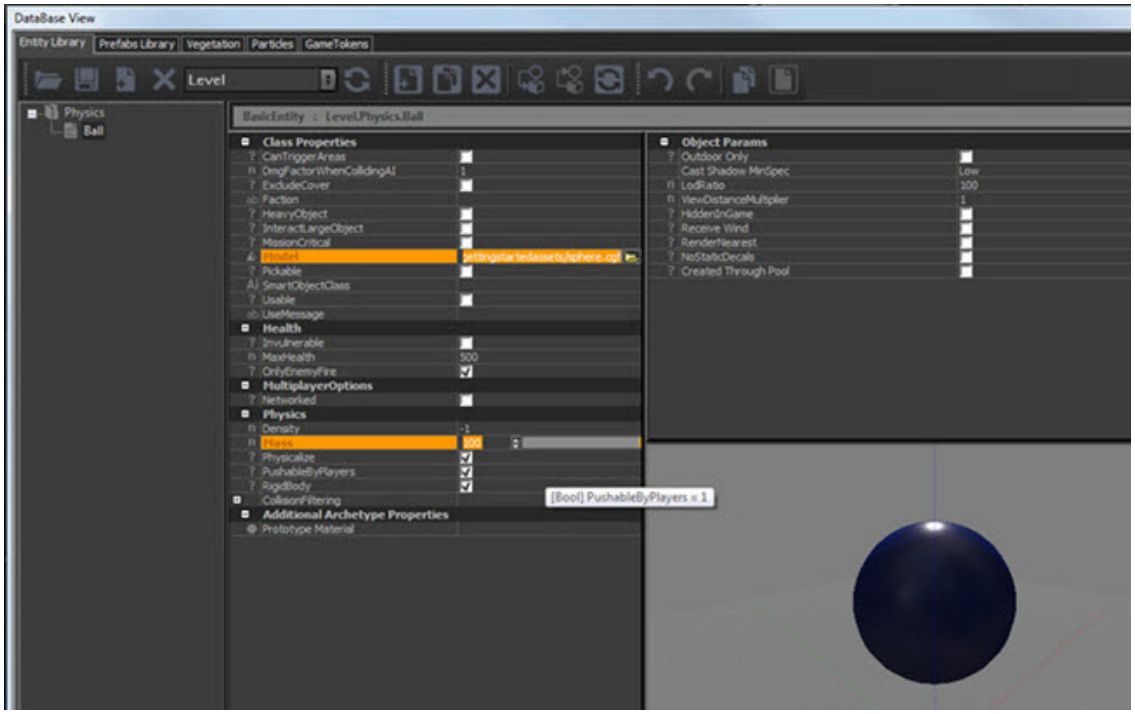
4. In the **Group** field, type **`Physics`**. In the **Name** field, type **`Ball`**. Click **OK**.



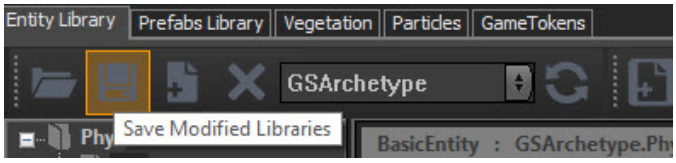
The new archetype entity is displayed.

5. Under **Class Properties**, click **Model**. Click the folder icon.
6. Expand `SamplesProject\Objects\GettingStartedAssets` and open `sphere.cgf`.

7. Under **Physics**, select **Mass** and set it to **`100`**.



8. Click **Save**.



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.

Flow Graph Scripting

This tutorial introduces the concept of game play scripting using the Flow graph editor.

You will set up flow graph scripts that do five things:

1. Trigger a change between the game play camera and a fixed camera
2. Move a block left and right with keyboard input
3. Set up same block to launch (spawn) a physics sphere
4. Control the power input of the block launcher through keyboard input
5. Toggle input controls to block based on camera state

Topics in the section include:

Prepare the scene

Before you build a script, you'll prepare the scene with several entities that you'll reference when building the flow graph scripts.

Place Tag Points

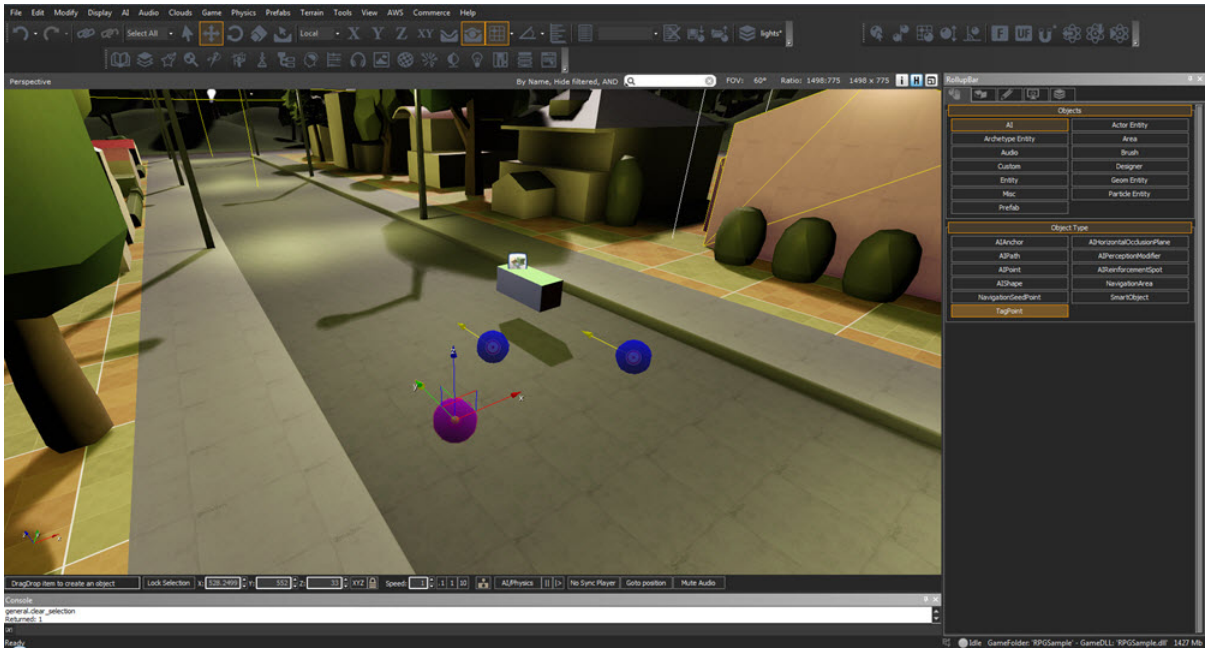
First, you'll add some tag points, which are markers that define an XYZ point in a level. They are used to define a destination, move-to position, or start point.

To add tag points

1. In the **Rollup Bar**, on the **Objects** tab, click **AI**.
2. Under **Object Type**, click **Tagpoint**. Move your mouse cursor into the **Perspective** viewport and then click to place the tag point.

   Repeat steps 1 and 2 an additional two times so you have a total of three tag points in the scene.

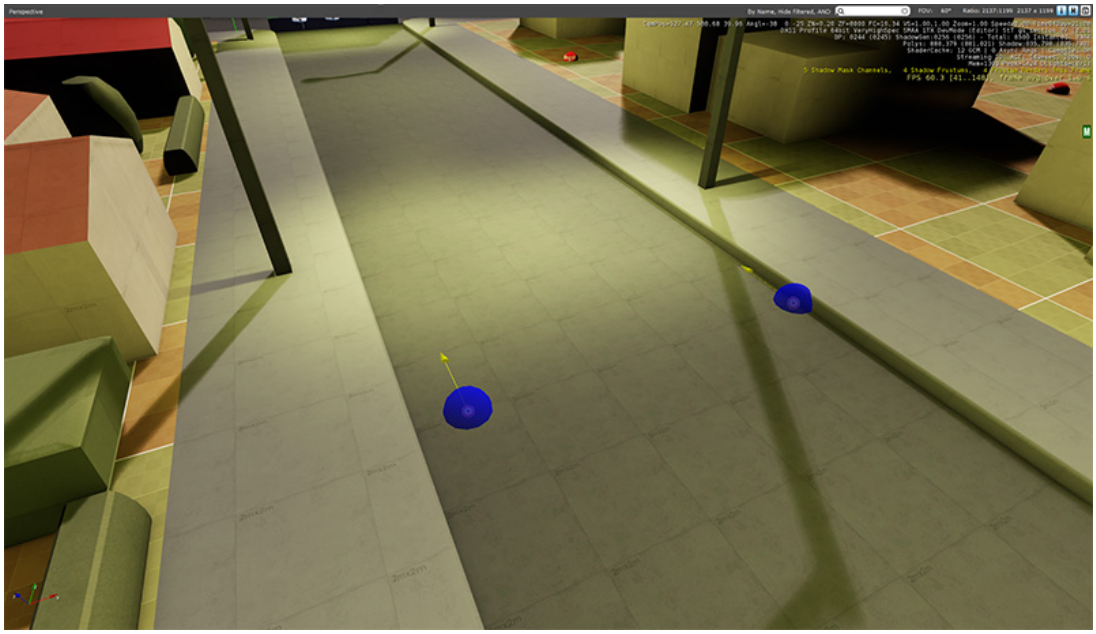
   Tip
   If your tag points do not appear as blue spheres (as shown in the following image), click the **H** icon on the **Perspective** viewport's header.



3. 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.

   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.
4. Position two of the tag points midway on the length of the street on each side of the street. It should look something like this:



Add Block and Link

Next, you'll add the block you created earlier and then link a tag point to it. You'll later spawn (shoot) balls from this linked object.

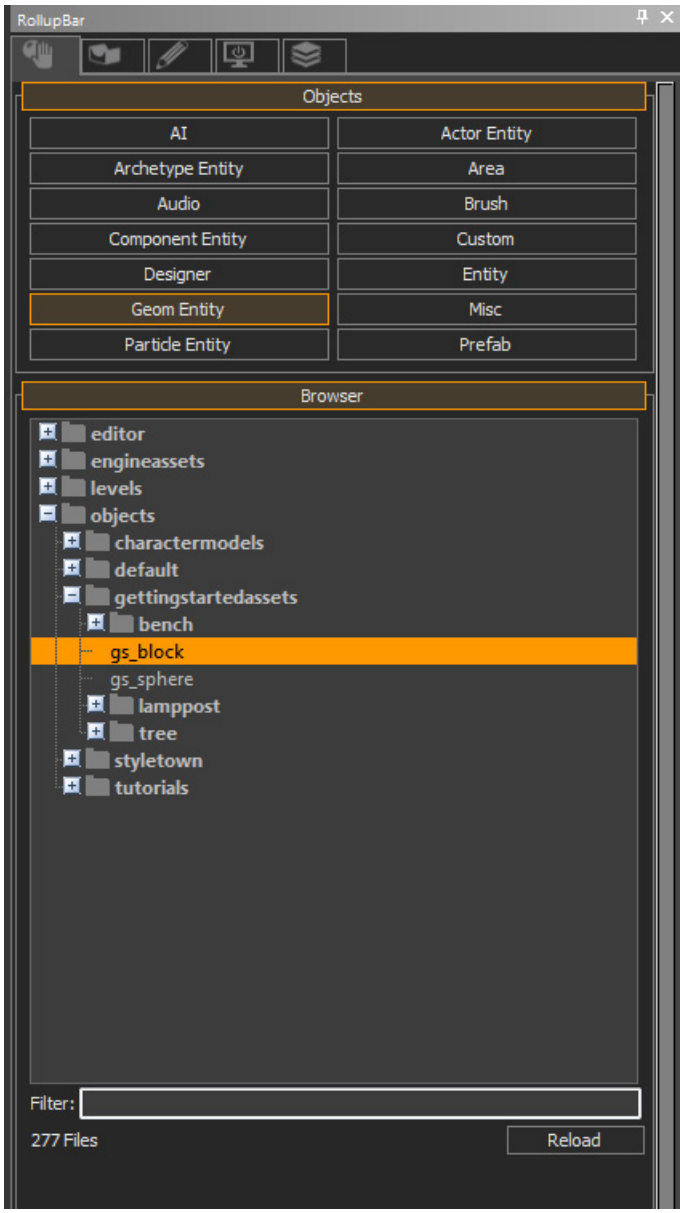
To add the block and link a tag point

1. In the **Rollup Bar**, on the **Objects** tab, select **Geom Entity**.

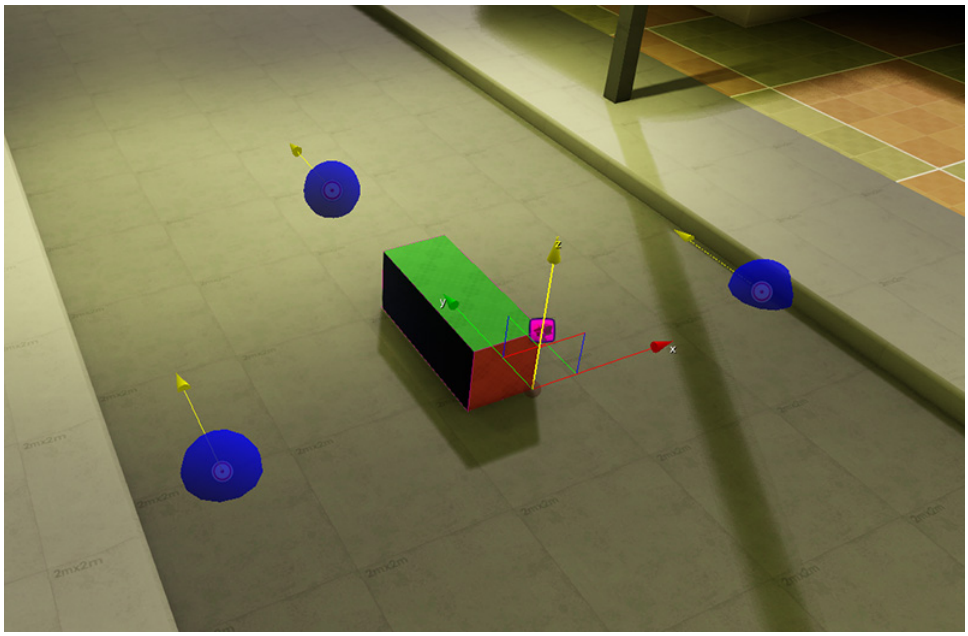
   Under **Browser**, expand `objects\gettingstartedassets` and select the entity block that you previously created.

   Drag the block into the **Perspective** viewport near the tag points.

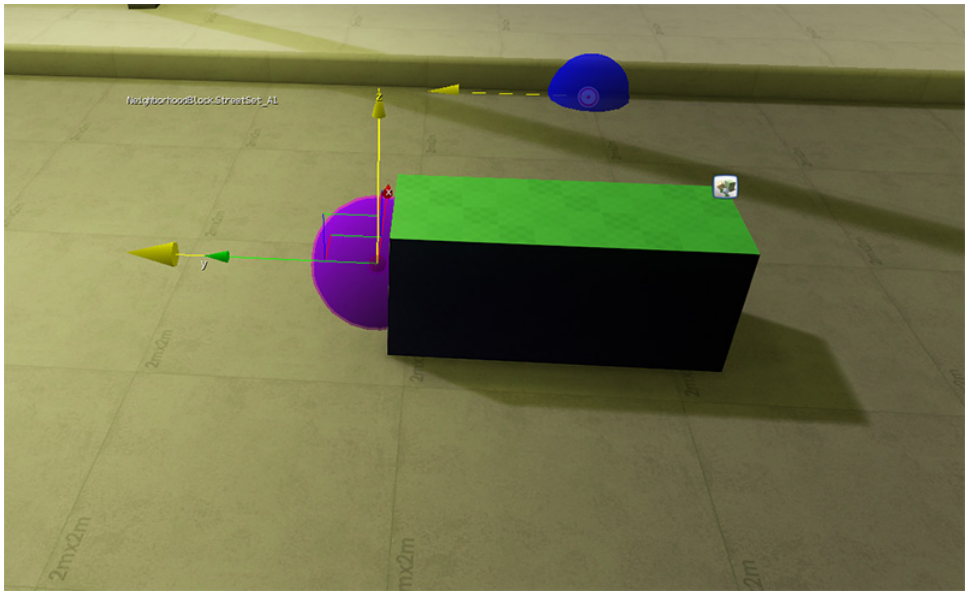
   Tip
   If you do not see your block and sphere in the list, click **Reload** at the bottom of the **Browser** area.


RollupBar
Objects
AI
Actor Entity
Archetype Entity
Area
Audio
Brush
Component Entity
Custom
Designer
Entity
Geom Entity
Misc
Particle Entity
Prefab
Browser
editor
engineassets
levels
objects
charactermodels
default
gettingstartedassets
bench
gs_block
gs_sphere
lamppost
tree
styletown
tutorials
Filter:
277 Files
Reload

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



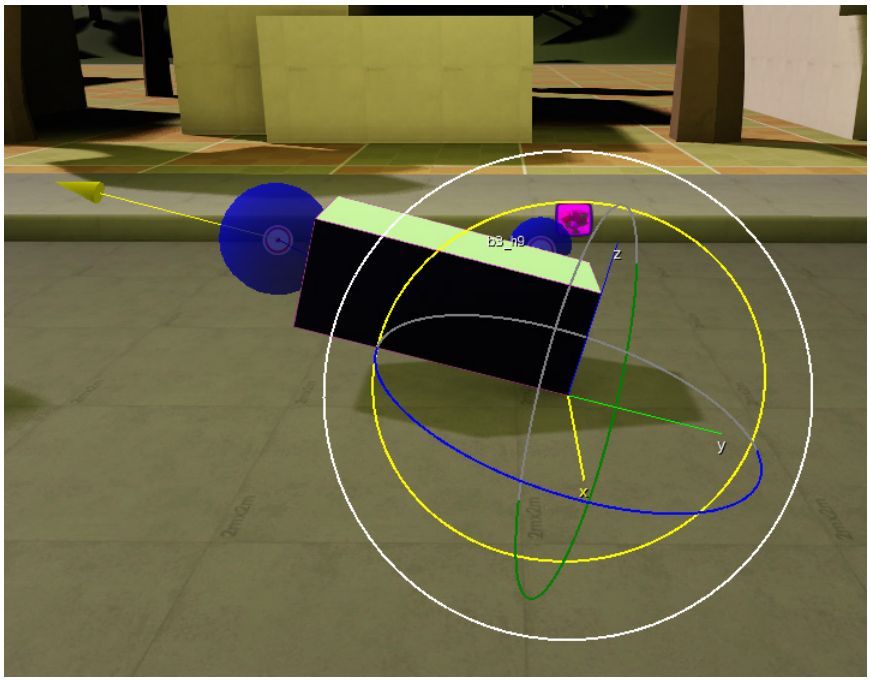
3. Position the third tag point so that it's centered and in front of the block.



4. In the editor toolbar, click **Link Object**  . Select the tag point in front of the block, then left-click and drag toward the block. The cursor displays a link icon as well as the object name it can link to (if present).

   When the block name is displayed, release the mouse. 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.
5. Rotate the block up about 20 degrees. The linked tag point moves with the block as it rotates.

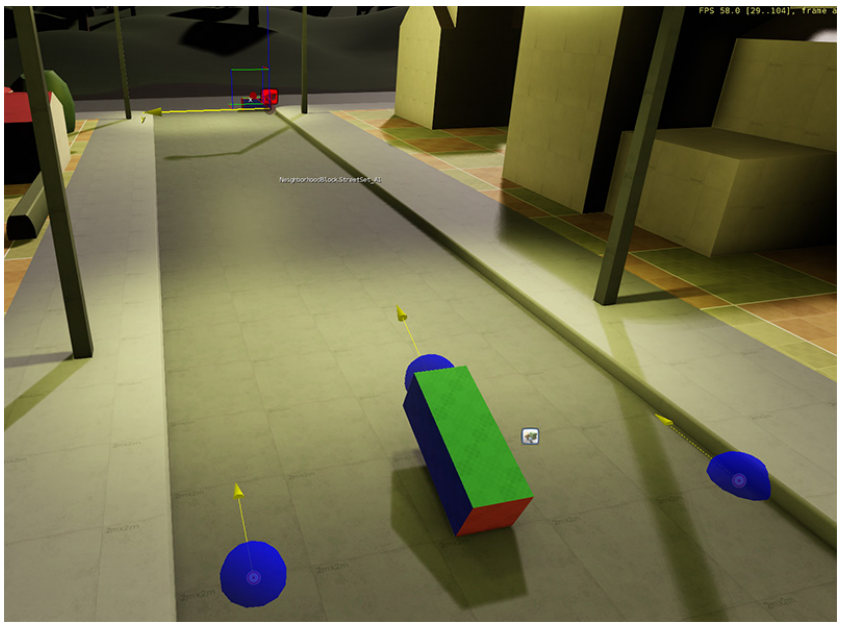


Stack Physics Blocks

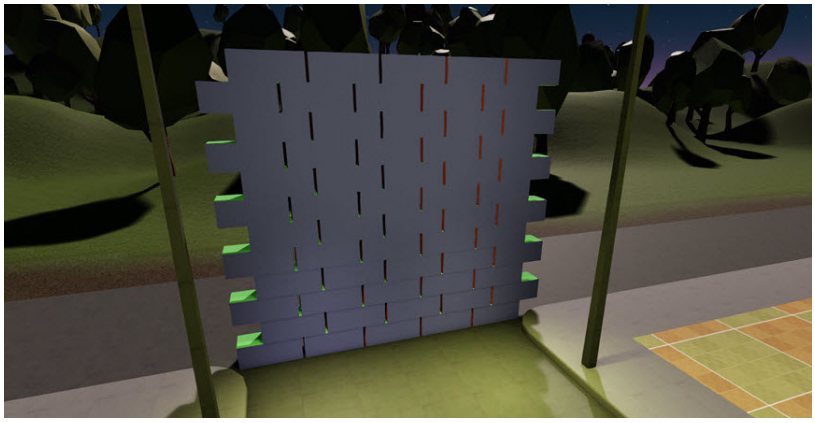
Next, you'll stack the physics blocks into a wall at the end of the street.

To stack the physics blocks

1. Place into your scene the physics block that you created in the previous tutorial.
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.



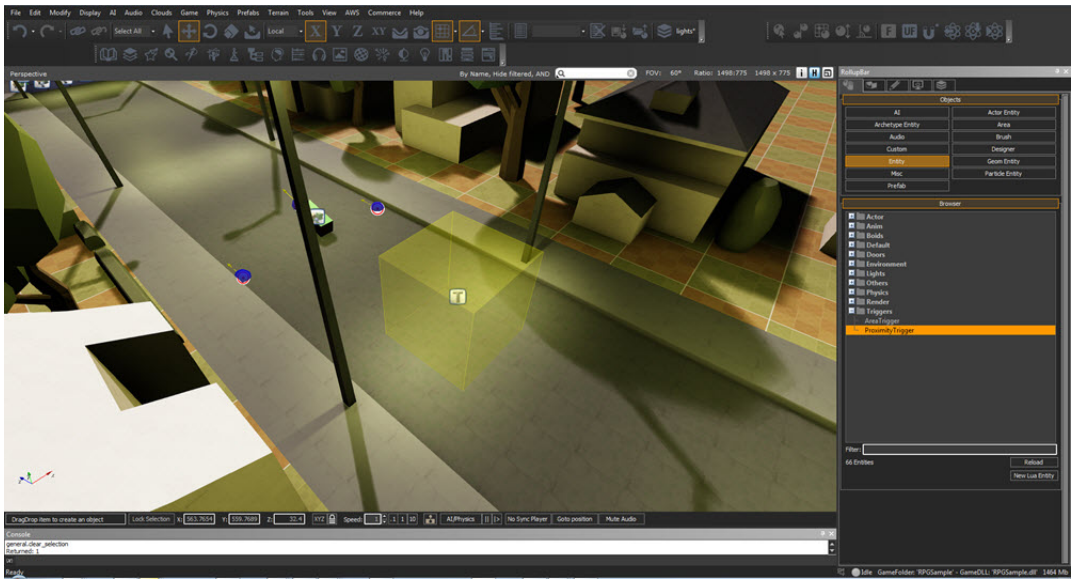
3. Position the first block, then press **Ctrl+C** to clone the block and position the next block. Click to place it. Repeat this process to build a wall of blocks similar to the following:



Add A Trigger Volume

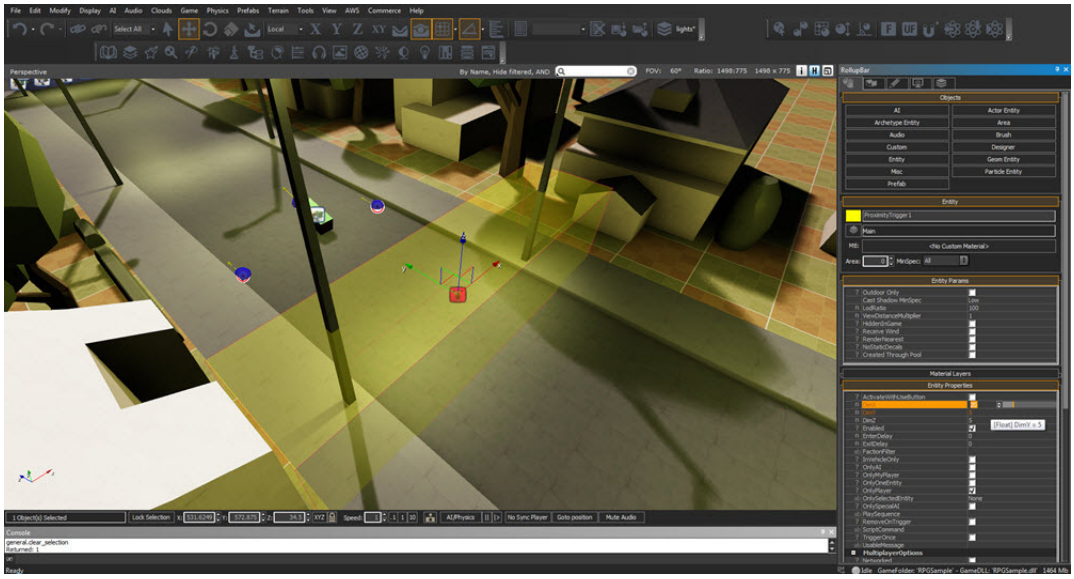
To add a trigger volume

1. From the **Rollup Bar**, click **Entity**, and then expand **Triggers**. Drag **ProximityTrigger** into the **Perspective** viewport.
2. Position the 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 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`**.

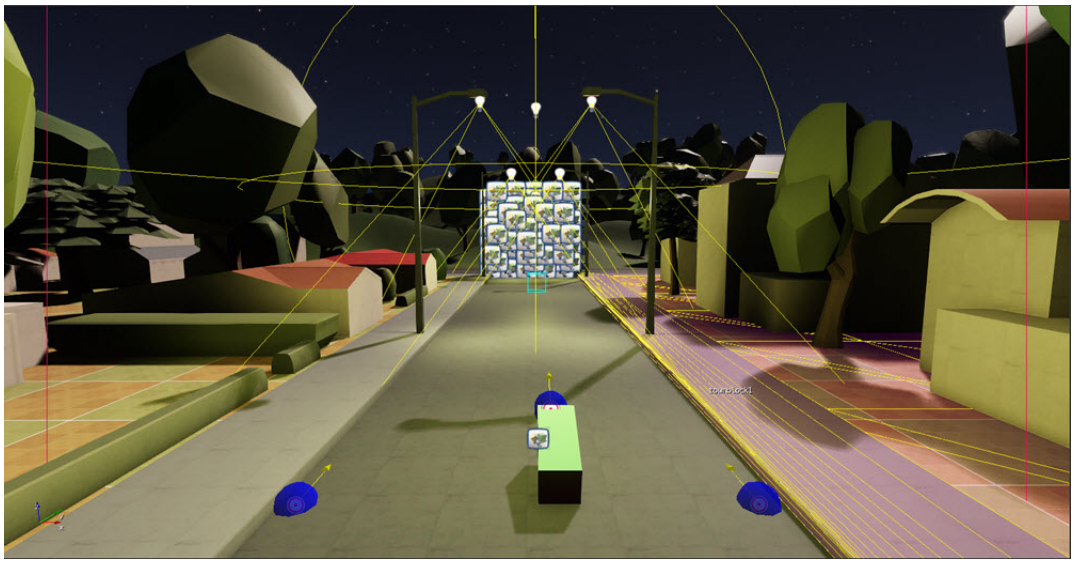


Add A Camera

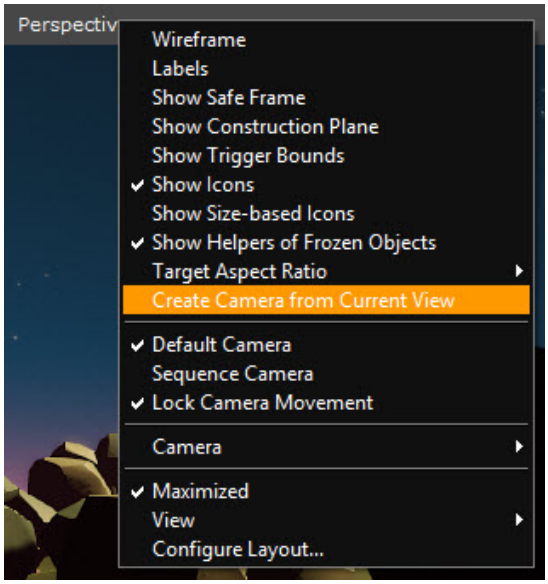
Now you'll add a camera to which the view switches when controlling the block and shooting at the wall.

To add a camera

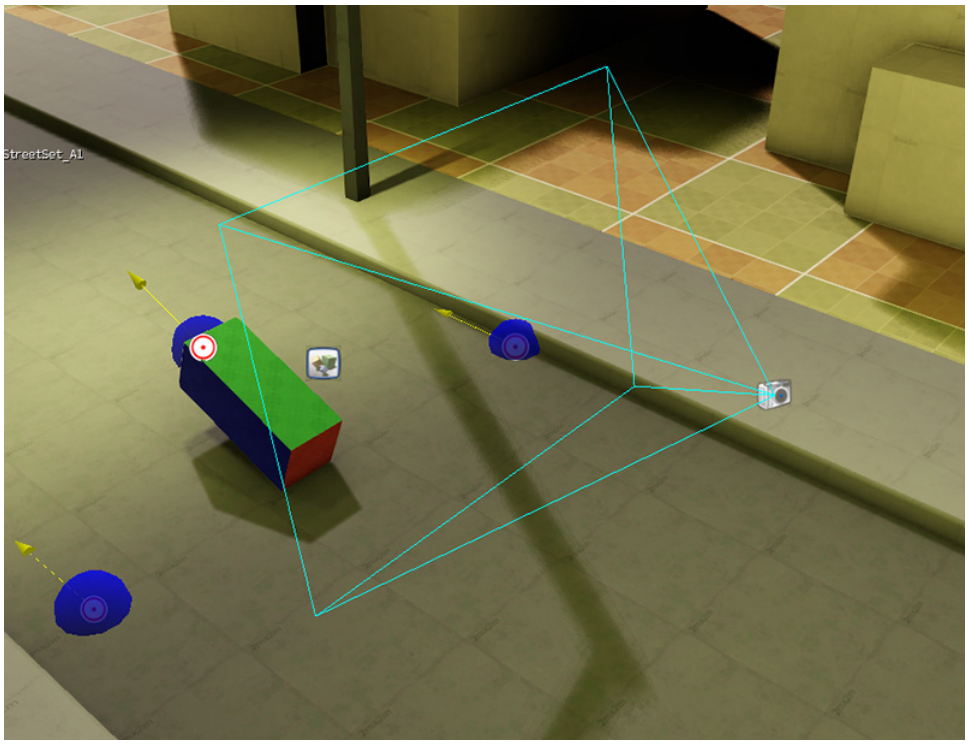
1. Position the view so that you can see the block at the bottom of the screen and the wall of physics blocks down the street. It should look something like the following:



2. In upper left corner of the **Perspective** viewport, right-click on '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.



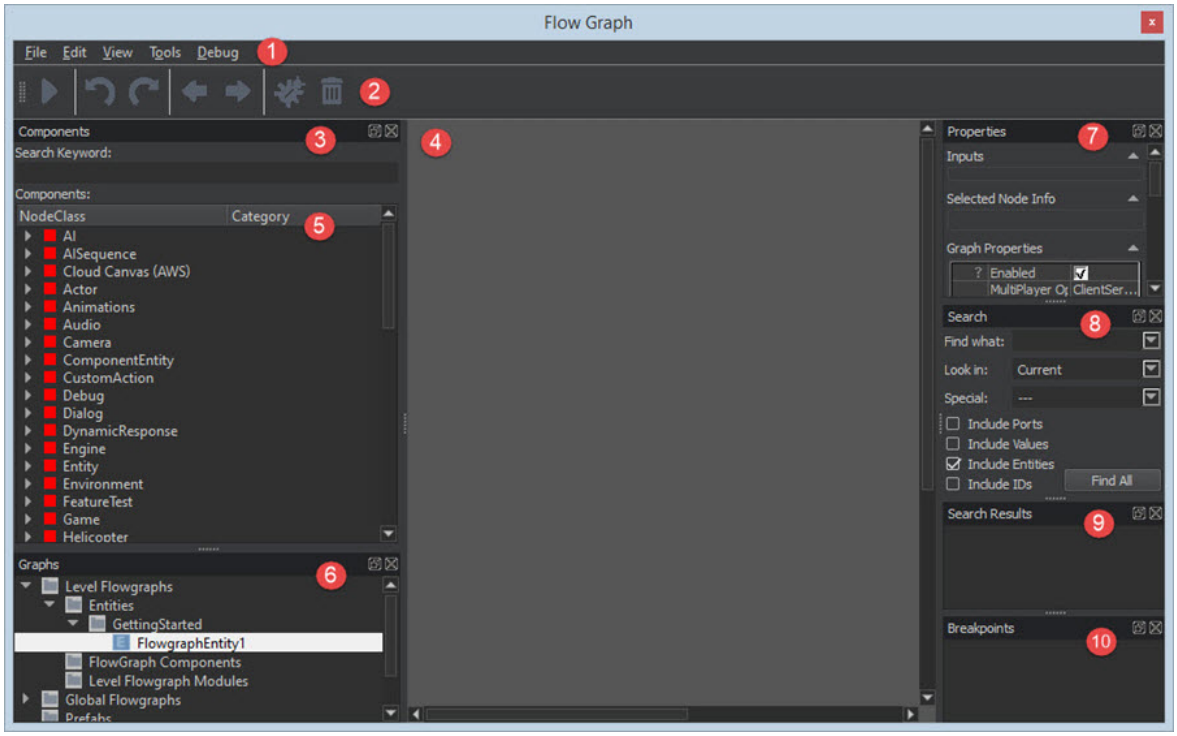
Create a Flow Graph Script

Now that all the objects needed are placed, the next step is to create a flow graph file and begin building the scripts.

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

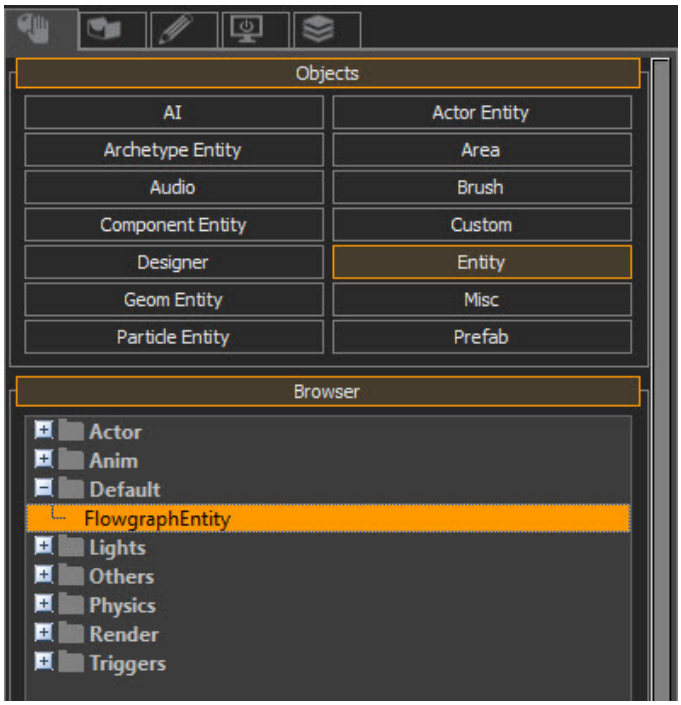
The Flow Graph Editor has the following areas:

- **1: Main Menu** – Tools to manage your Flow Graph files
- **2: Editor Toolbar** – Tools to undo, redo, step through, toggle debug and clear debug
- **3: Component Search Bar** – Search for available components
- **4: Canvas View** – Area where scripts are built
- **5: Component List** – Listing of nodes available to create script events
- **6: Flow Graph File Tree** – Access to Flow Graph files available in level and project
- **7: Node Properties** – Property settings of selected component nodes
- **8: Canvas Node Search** – Search for specific nodes in canvas view
- **9: Canvas Node Search Results** – Results of search for nodes in canvas view
- **10: Debug Breakpoints** – Breakpoint management for debug

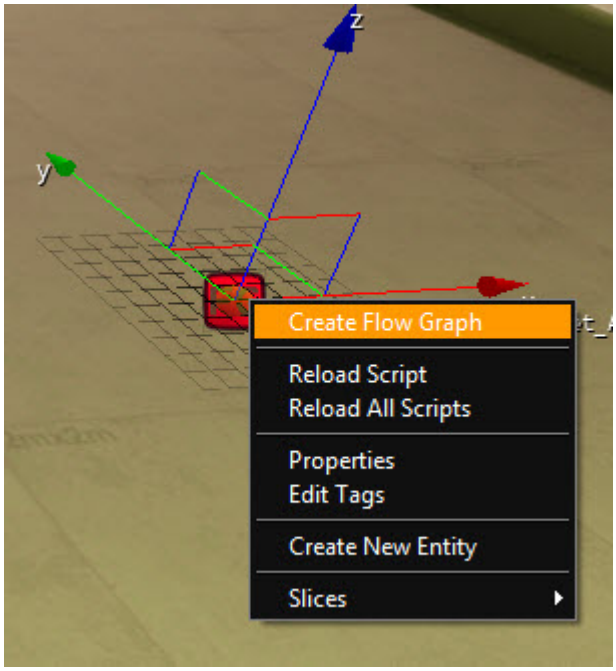


To script game events with Flow Graph

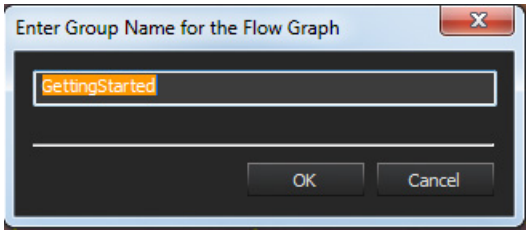
1. In the **Rollup Bar**, click the **Objects** tab and then click **Entity**. Under **Browser**, expand **Default**.
2. Drag **FlowgraphEntity** into the **Perspective** viewport.



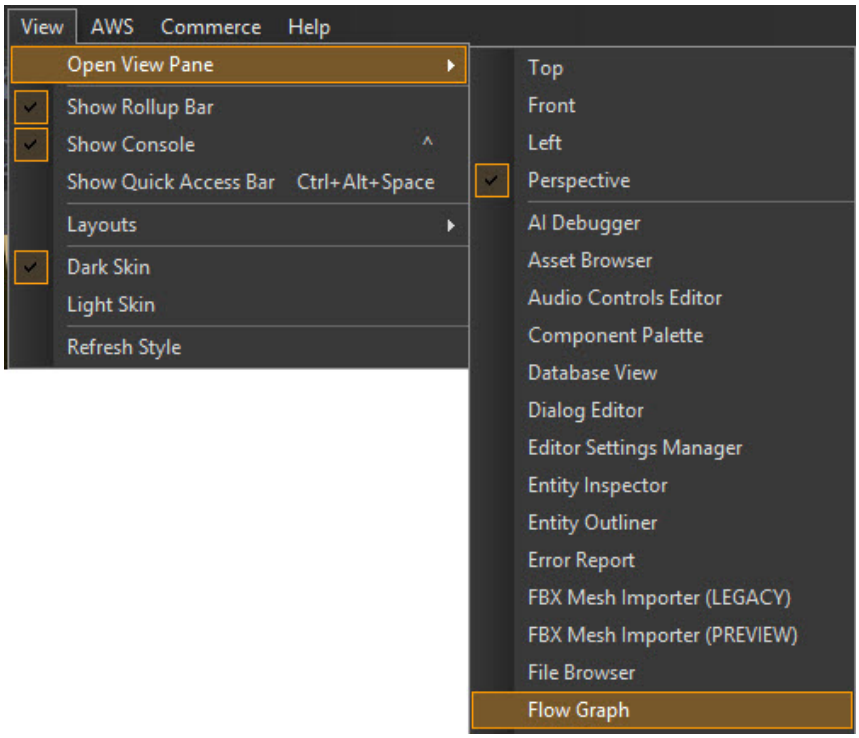
3. Right-click the **FlowgraphEntity** and click **Create Flow Graph**.



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



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

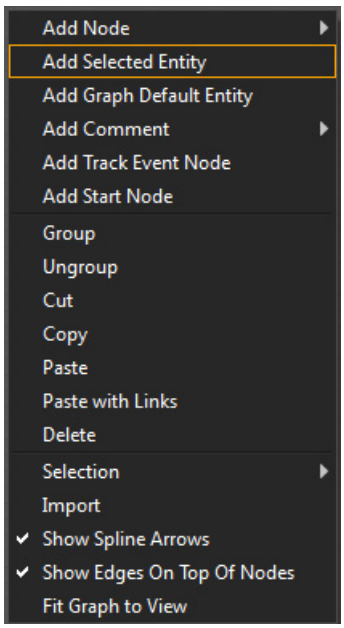
   Flow graph scripts are composed of nodes that contain information about objects in the level or functions to call when activated through the script. Flow graph nodes can be pulled from two places: From an entity object currently in the **Perspective** viewport, or from the list of components within the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

Script Camera Event

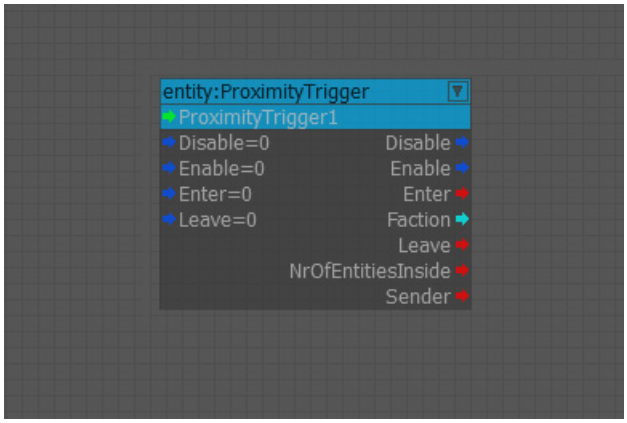
In this first script, you'll set up a trigger that switches between the game play camera and another camera placed in the level.

To set up a trigger

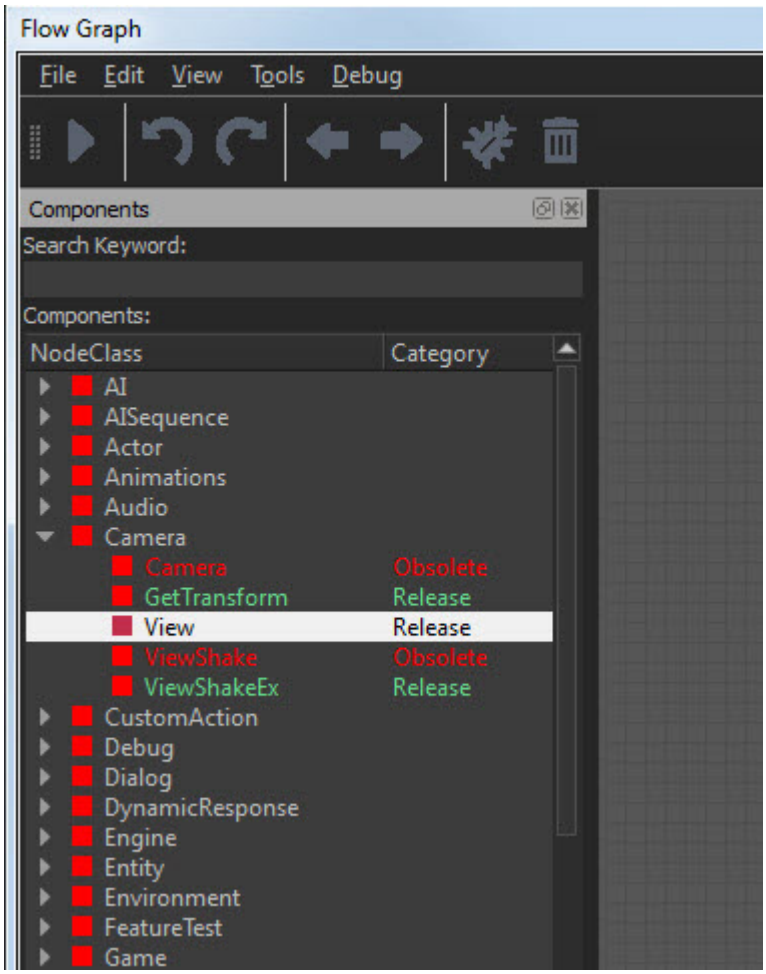
1. Select the proximity trigger that you placed earlier.
2. In the **Flow Graph** editor, right-click in the Canvas view. Click **Add Selected Entity**.



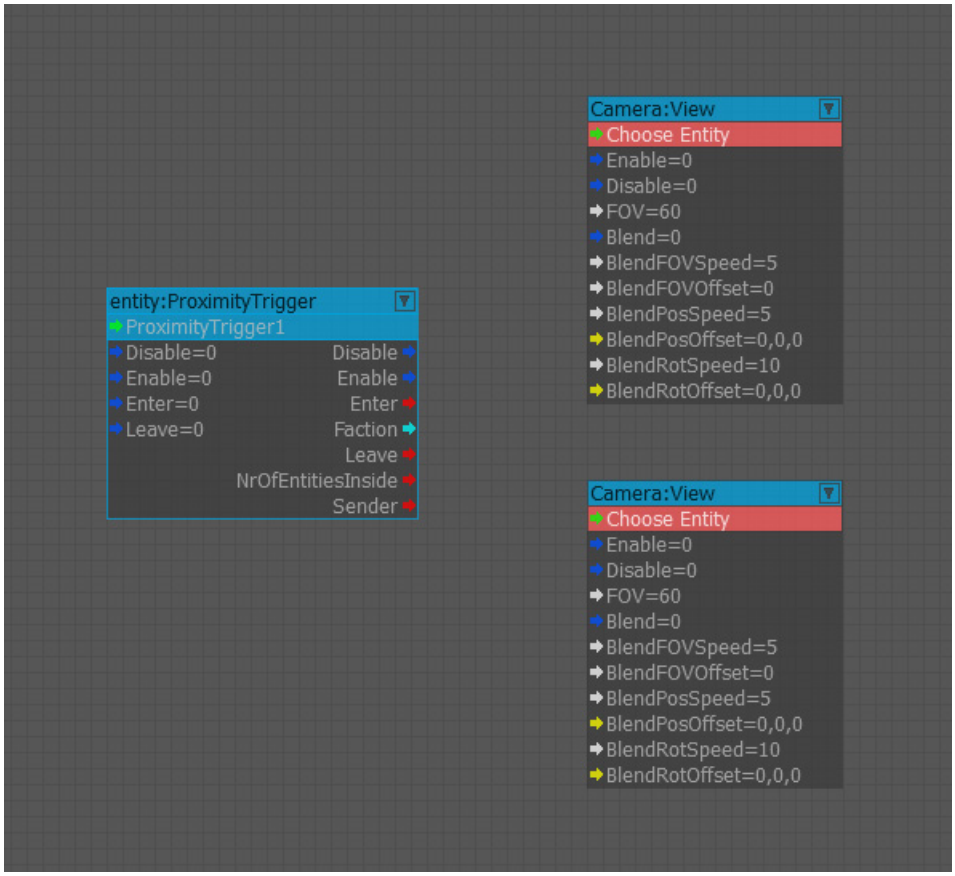
The node **entity:ProximityTrigger** is displayed in the canvas view.



3. In the Component list view, expand **Camera**. Drag the node **View** into the Canvas view.



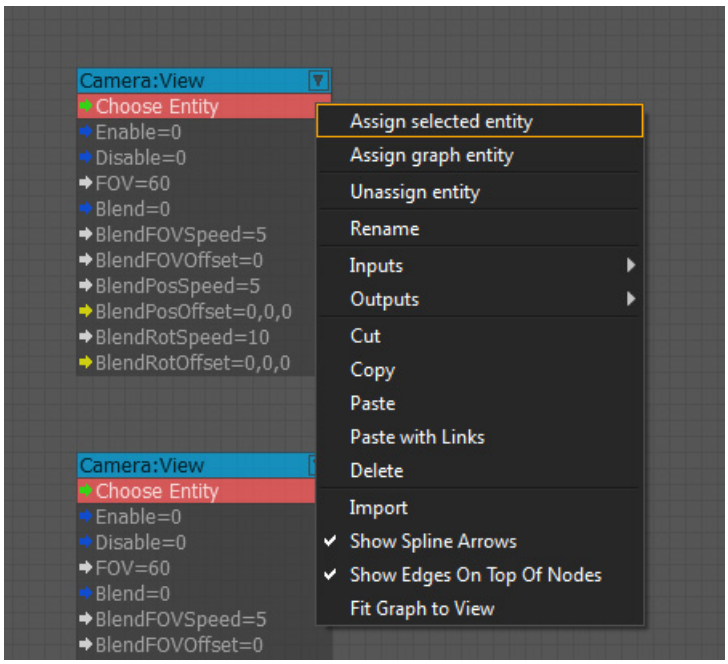
4. Repeat the previous step to drag another **View** node into the Canvas view. The canvas should now look like this.



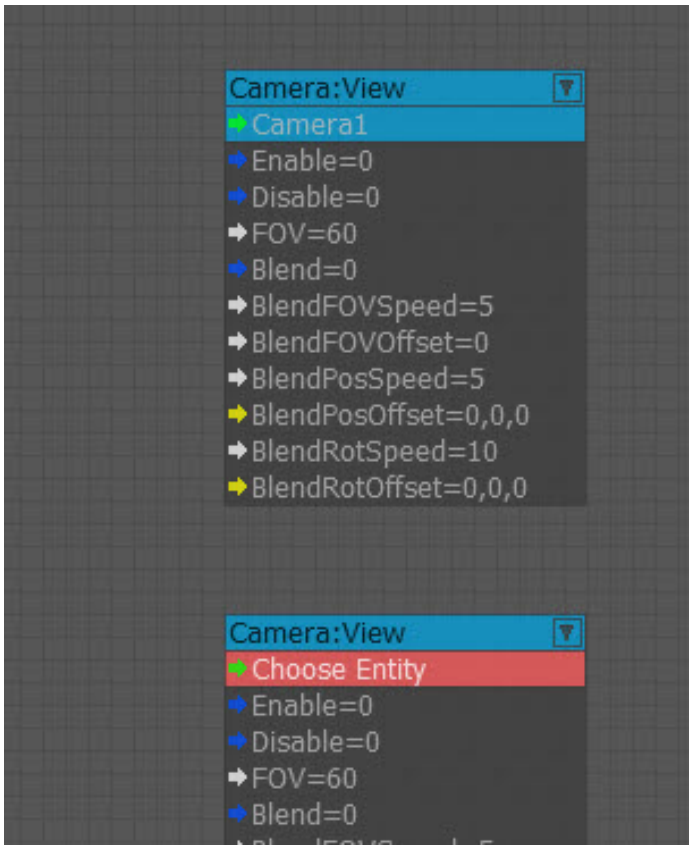
5. Now you'll assign a camera to each **Camera:View** node.

   To do this, 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.

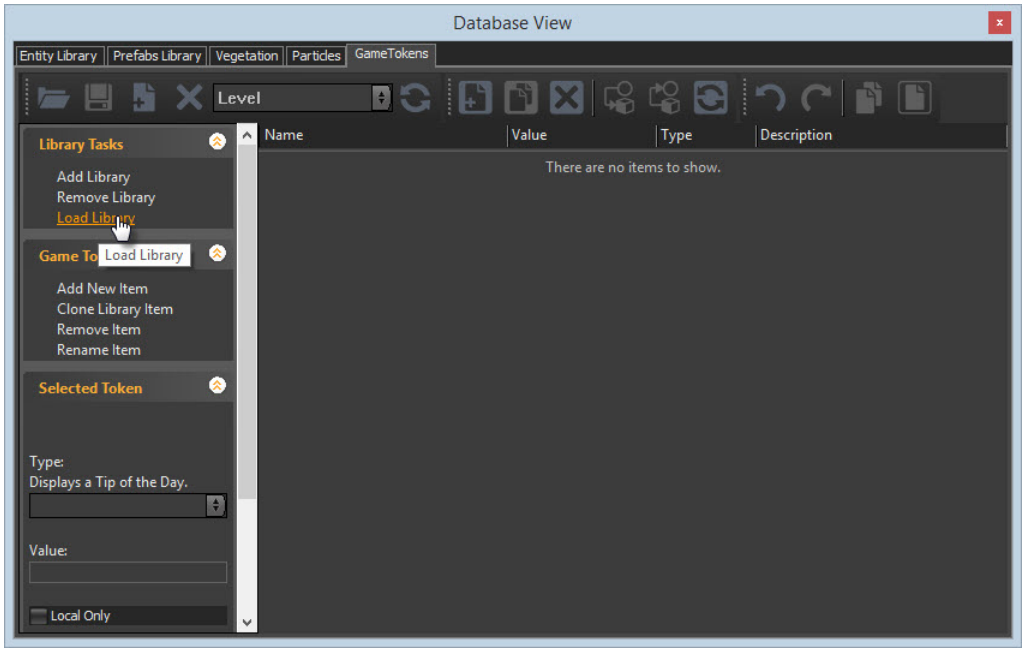


Now you'll assign the game play camera to the second camera node. To do this, you pull out the Camera ID from the camera prefab using a game token.

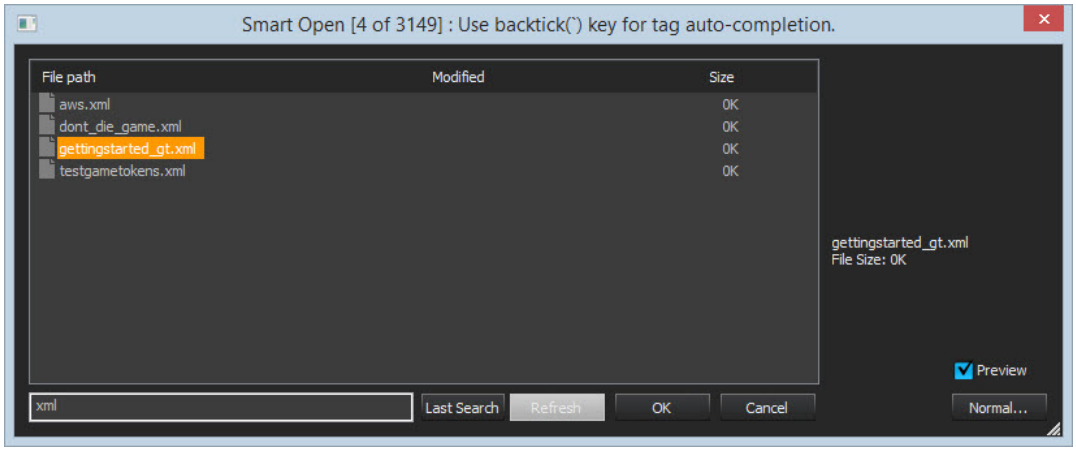
Game Tokens are values that exist outside of the existing flow graph script, or outside of the level currently in-work.

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

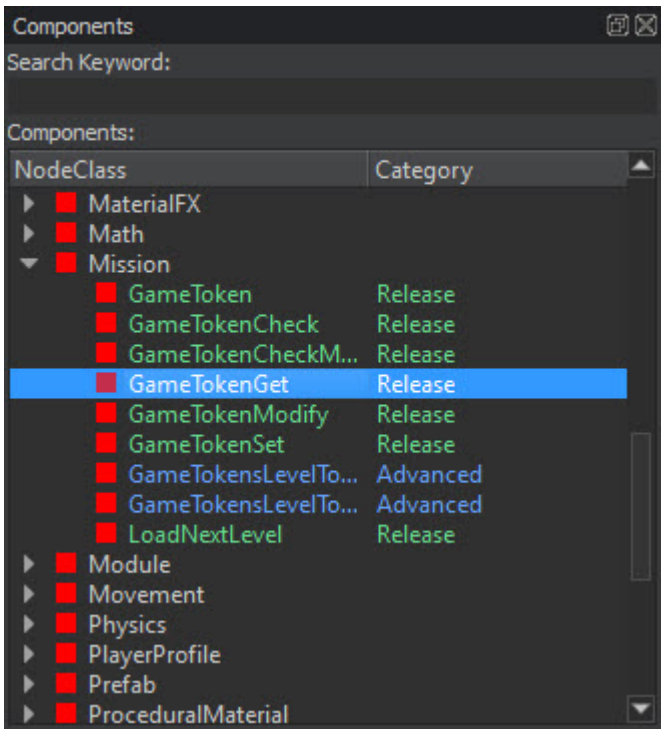
8. Click the **Game Tokens** tab. In the **Library Tasks** area, click **Load Library**.



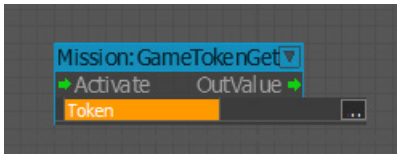
9. Select `libs\gametokens\gettingstarted_gt.xml` and click **OK**.



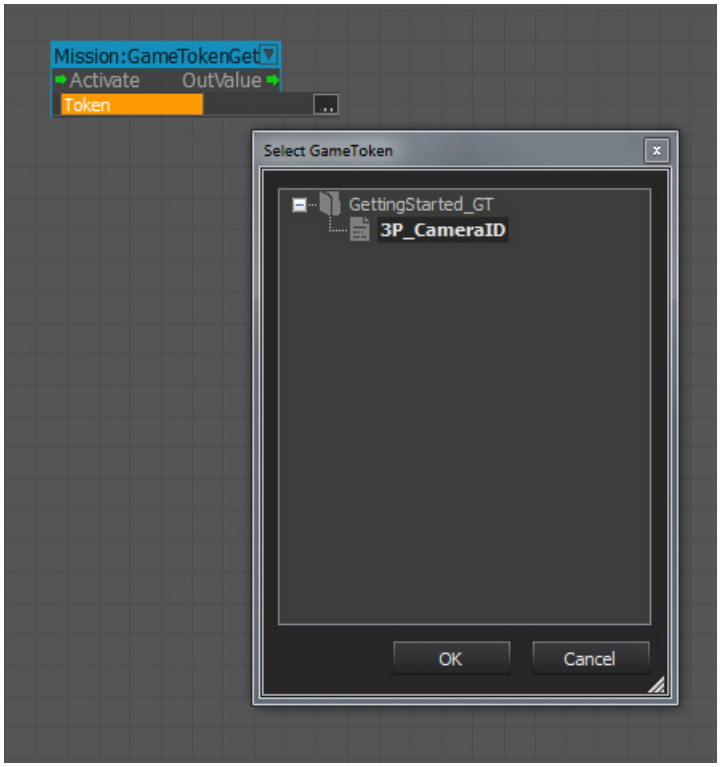
10. Return to the **Flow Graph** editor. In the **Components** list, expand **Mission**. Drag the node **GameTokenGet** onto the canvas.



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

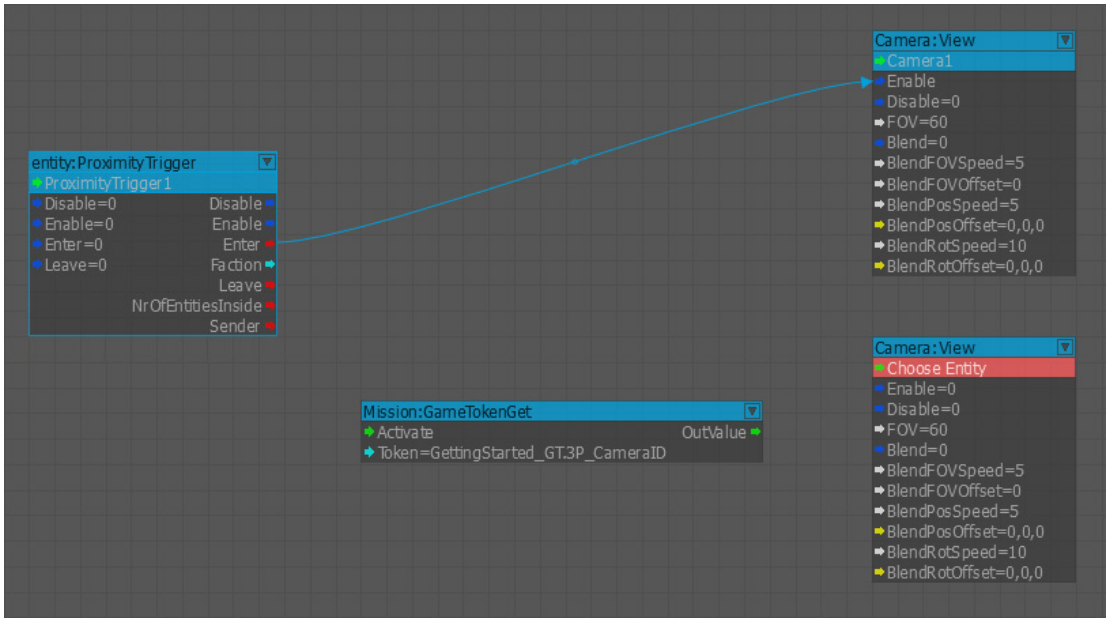


12. Expand `GettingStarted_GT` and select `3P_CameraID`. Click **OK**.



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

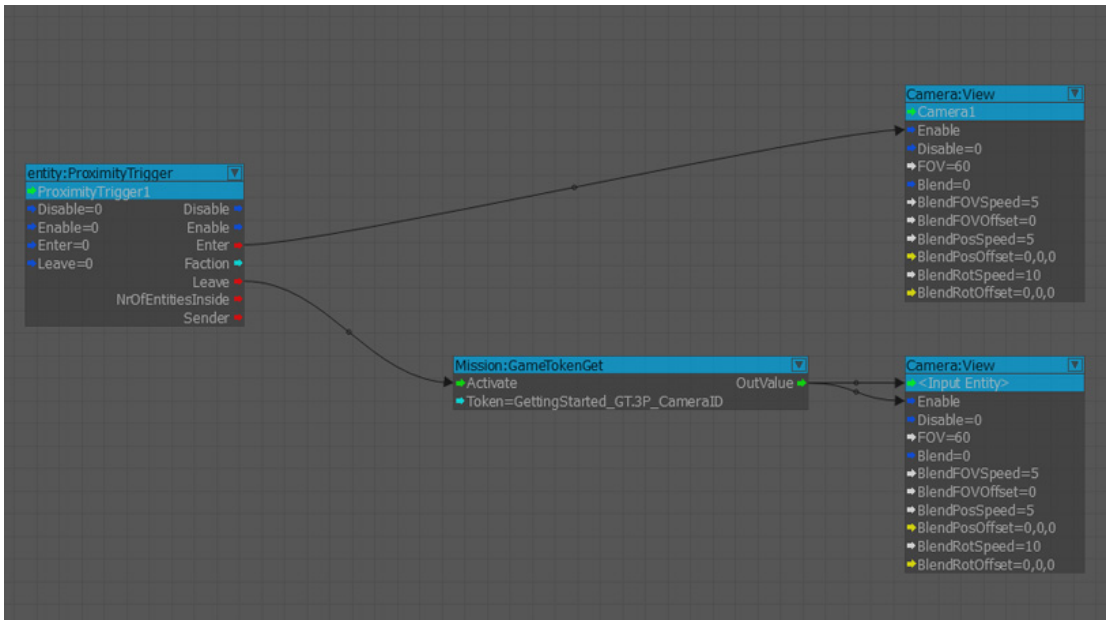
    This creates a connection that links one node to the next. These connections tell the game what to do and when to do it. A flow graph script comes together by connecting nodes to one another.



14. From the **entity:ProximityTrigger** node, drag the **Leave** output port to the **Mission:GameTokenGet** node's **Activate** input port.

    Drag the **Mission:GameTokenGet** node's **Outvalue** output port to the second **Camera:View** node's **<Input Entity>** input port as well as its **Enable** input port.

    The Flow graph should look something like this:



15. With the trigger and cameras set up, test the functionality to make sure the trigger switches to the first camera while in the trigger volume and then returns to the game play camera when leaving the trigger volume.

Create a Mover

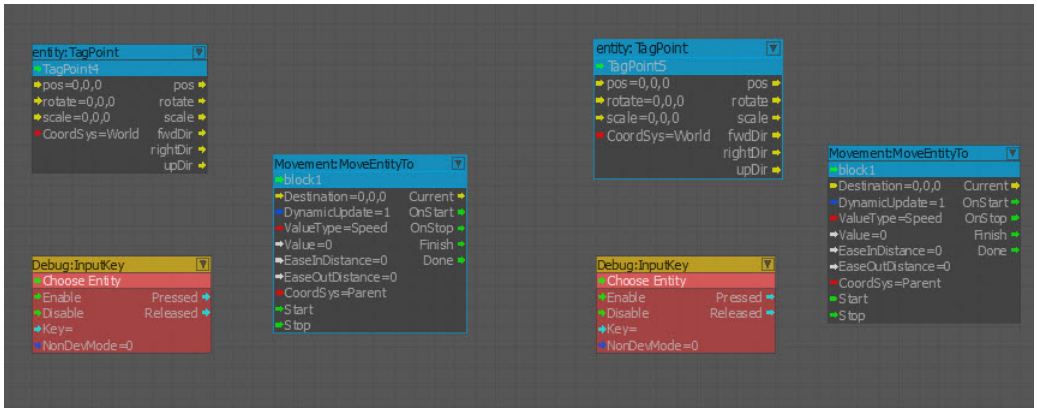
In this script, you'll make a block move between two points with keyboard input control.

To create a script to make the block move between two 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, and right-click drag to move the canvas.

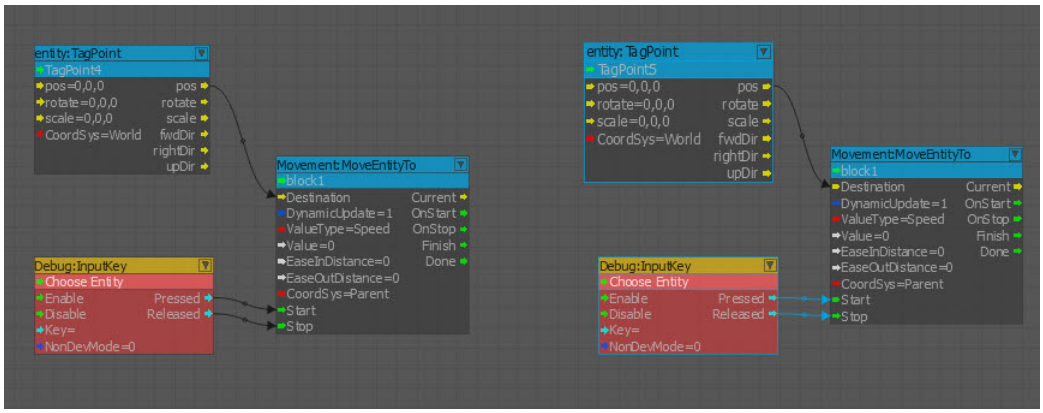
   A flow graph can have multiple script events within the same canvas. If desired, you can always create a new script file, which may work best when organizing your level. For this tutorial, keep all the scripts in the same flow graph file.
2. In the component list, expand the folder **Movement** and drag two of the **MoveEntityTo** nodes into the canvas view .
3. In the component list, expand the folder **Debug** and drag two of the **Debug:InputKey** nodes into the canvas view .
4. In the **Perspective** viewport, select the tag point entity on the left side of the street (assuming you are facing the block wall). Add this to the canvas view with the **Add Selected Entity** command. Position the nodes so they look like the image 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, placing it next to the **Movement:MoveEntityTo** node on the right.

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

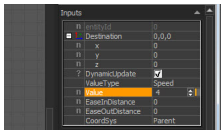


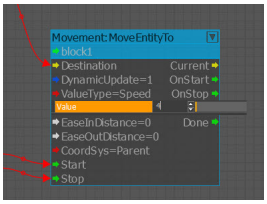
7. From the **entity:Tagpoint** node on the left, drag the **pos** output to **Movement:MoveEntityTo** node's **Destination** input.
8. Assign the same linkage sets with the other **Tagpoint** and **MoveEntityTo** nodes on the right.
9. Drag the left **Debug:InputKey** node's **Pressed** output to **Movement:MoveEntityTo** node's **Start** input.
10. On the same set of nodes, connect the **Released** output to the **Stop** input.
11. Apply the same linkage set with the other **InputKey** and **MoveEntityTo** nodes on the right.

    Your nodes should now look like this:



12. Select one of the **MoveEntityTo** nodes. In **Node Properties**, change **Value** = **4**. Or, you can double-click on a property directly in the node and enter the value there.





13. Select the other **MoveEntityTo** node and change **Value** = **4** as well. With the node's **ValueType** = **Speed**, setting the value to 4 sets the speed at which the block moves when activated.
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 and test that the block moves left and right with the keyboard inputs of `j` and `l`. If necessary, adjust the position of the tag points to which the block moves.

Spawn Object

In this script, you'll spawn a physics object to shoot out of a designated spawn point.

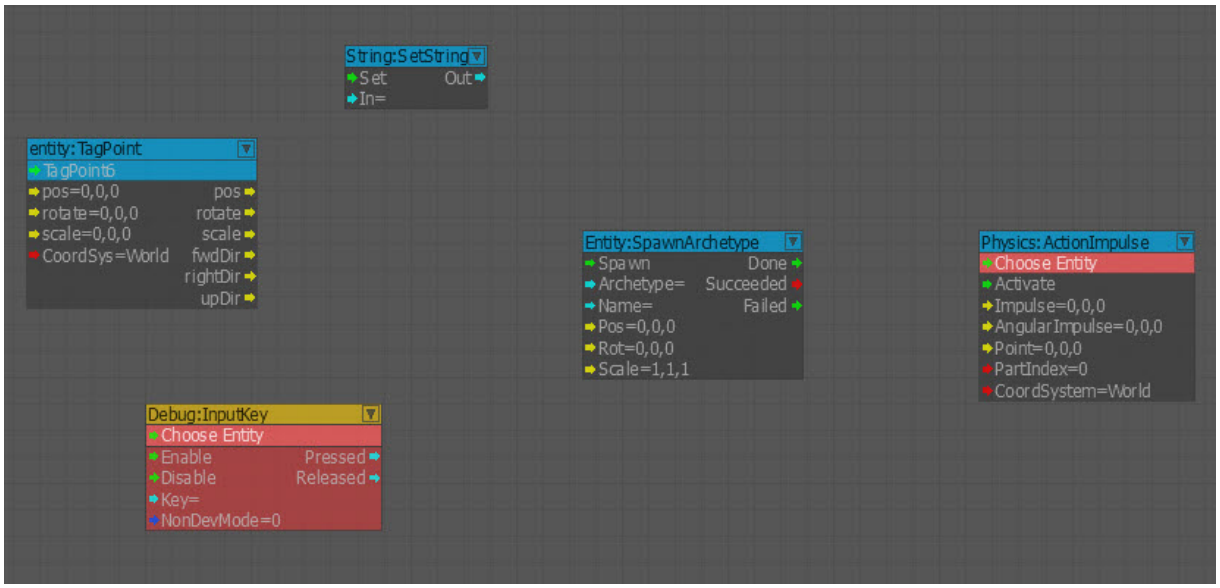
To spawn a physics objects

1. In the **Components** list:

    - Expand `Entity` and drag **SpawnArchetype** onto the canvas

- Expand `String` and drag **SetString** onto the canvas
- Expand `Physics` and drag **ActionImpulse** onto the canvas
- Expand `Debug` and drag **node:InputKey** onto the canvas

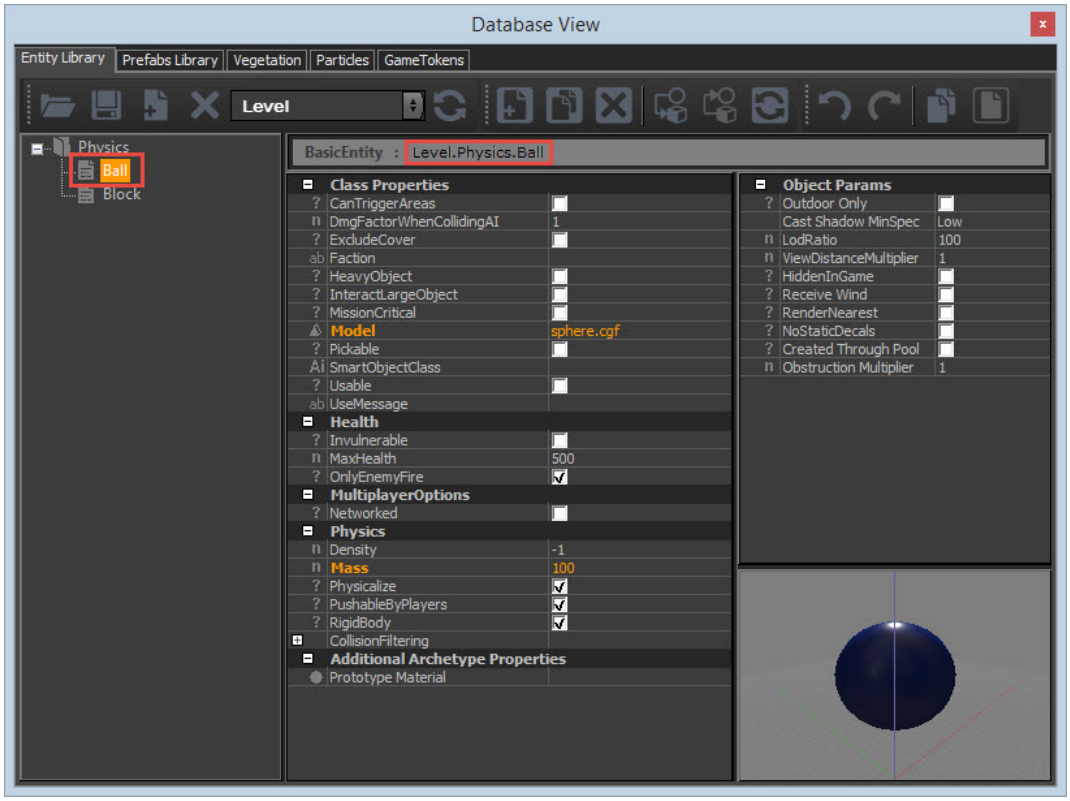
2. Select the tag point that was linked to the block in the scene. Add this entity to the canvas view.
3. Arrange the nodes to look like the following image.



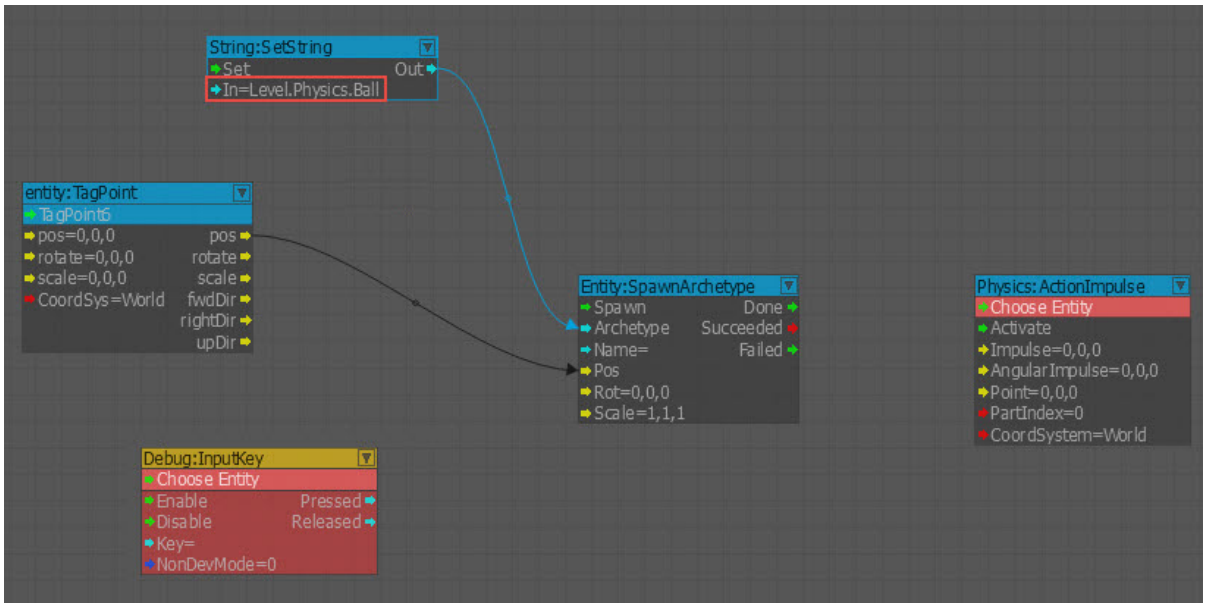
4. Drag the **TagPoint** node's **Pos** output to the **SpawnArchetype** node's **Pos** input.

   You will spawn the physics ball that you created in the Archetytpe Entity Database. The object name is entered into the SetString Node.

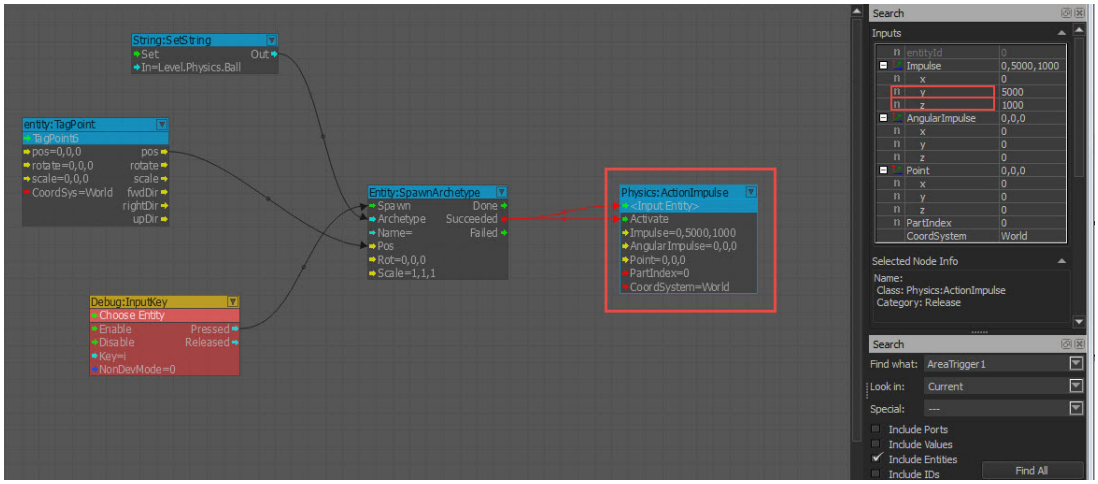
5. Open the **Database View** editor. Click the **Entity Library** tab, and then expand the **Physics** folder. Select **Ball**.



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



9. Set the **Inputkey** node's property **Key**= **`i`**.
10. Drag the **InputKey** node's **Pressed** output to the **SpawnArchetype** node's **Spawn** input.
11. Drag the **SpawnArchetype** node's **Succeeded** output to the **ActionImpulse** node's **Activate** input as well as its **Input Entity** input.
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.



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

Set Impulse Value

In this script, you'll modify the spawn object to shoot a physics object at different velocities based on how long you hold the keyboard input.

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

1. In the **Components** list, expand **Interpolate** and drag **Int** onto the canvas.
2. Expand **Vec3** and drag **ToVec3** onto the canvas.
3. In the **Debug:Inputkey i** that spawns the physics ball, disconnect the **pressed** output from the **SpawnArchetype** node's **Spawn** input.

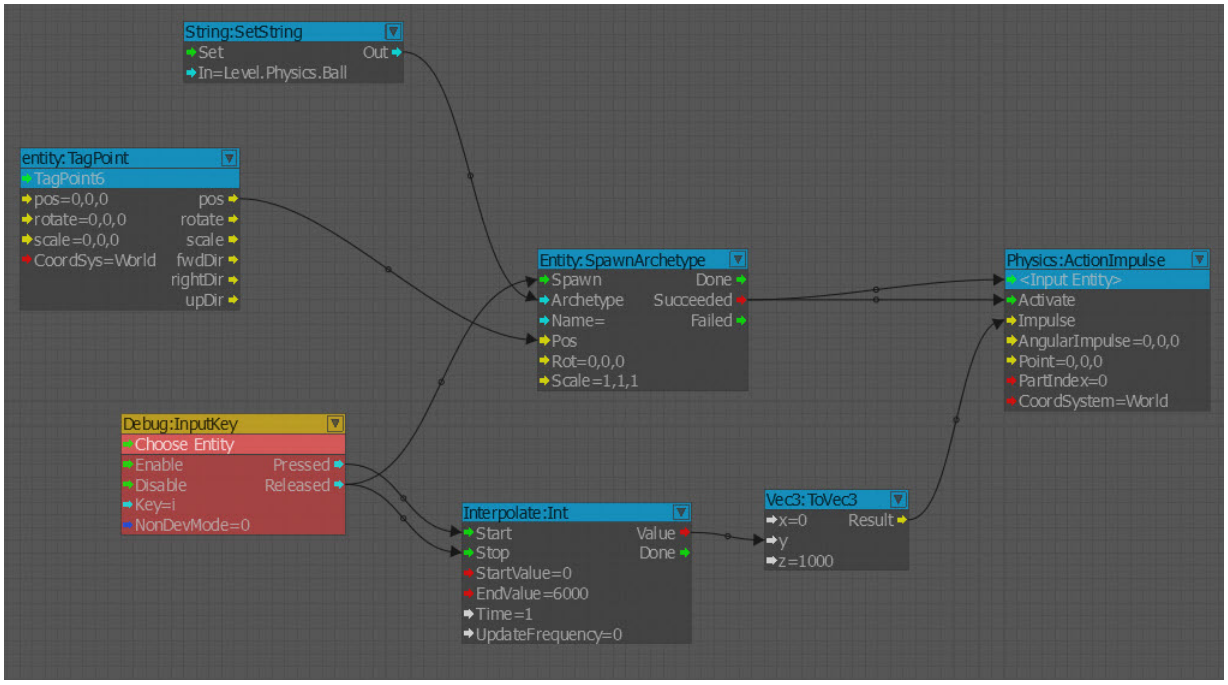
   In the same debug node, also:

   - Drag the **Released** output to the **Spawnarchetype** node's **Spawn** input. The ball should now spawn on release of the key.
   - Drag the **Pressed** output to the **Interpolate:Int** node's **Start** input.
   - Drag the **Released** output to the **Interpolate:Int** node's **Stop** input.

4. In the **Interpolate:Int** node, change the **EndValue** = **`6000`**. Also change the **Time** = **`2`**.
5. In the **Vec3:ToVec3** node, change **Z**=**`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)

7. Drag the **Vec3:ToVec3** node's **Result** output to the **ActionImpulse** node's **Impulse** input.

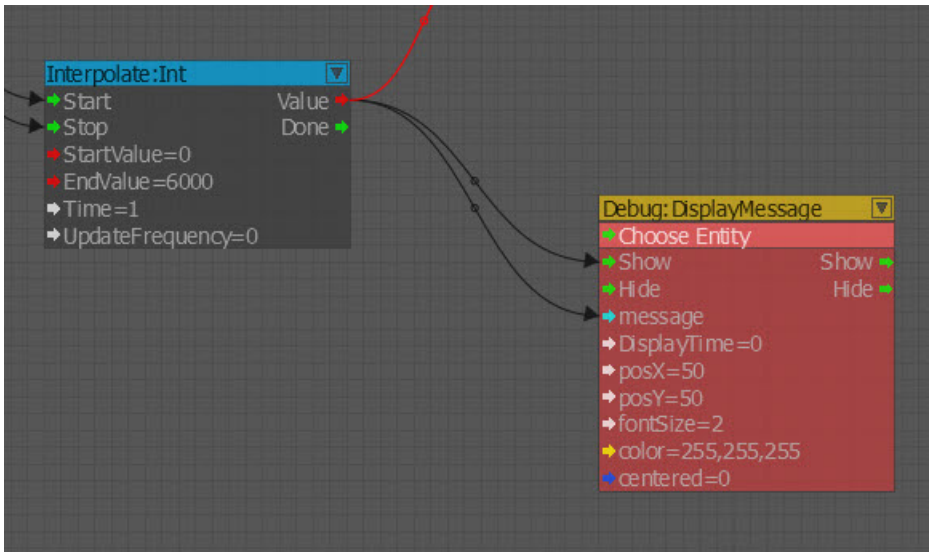
   The flow graph script should now look like this:



8. 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.

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



11. 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.

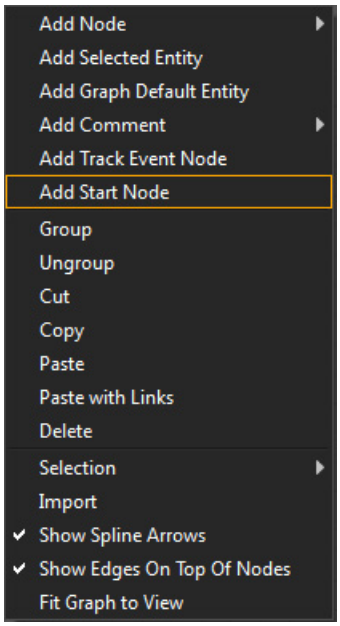
Any Nodes

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

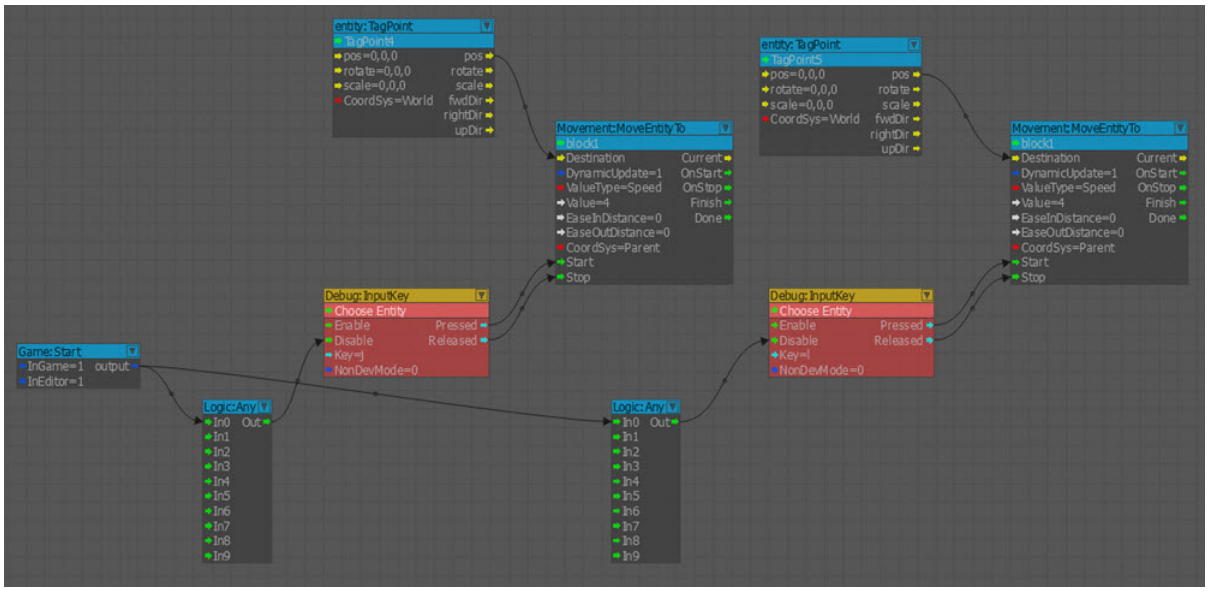
To disable the cannon until triggered

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

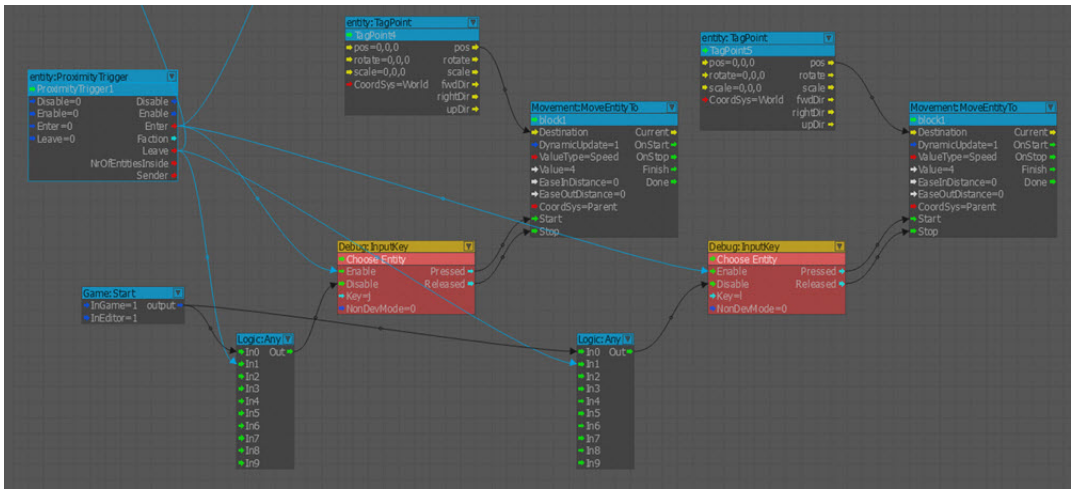
   Place an **Any** node by each of the three **Debug:InputKey** nodes.
2. Next to the **Debug:InputKeys** (these control the block moving left and right), add a **Game:Start** node. To do this, right-click and select **Add Start Node**.



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.



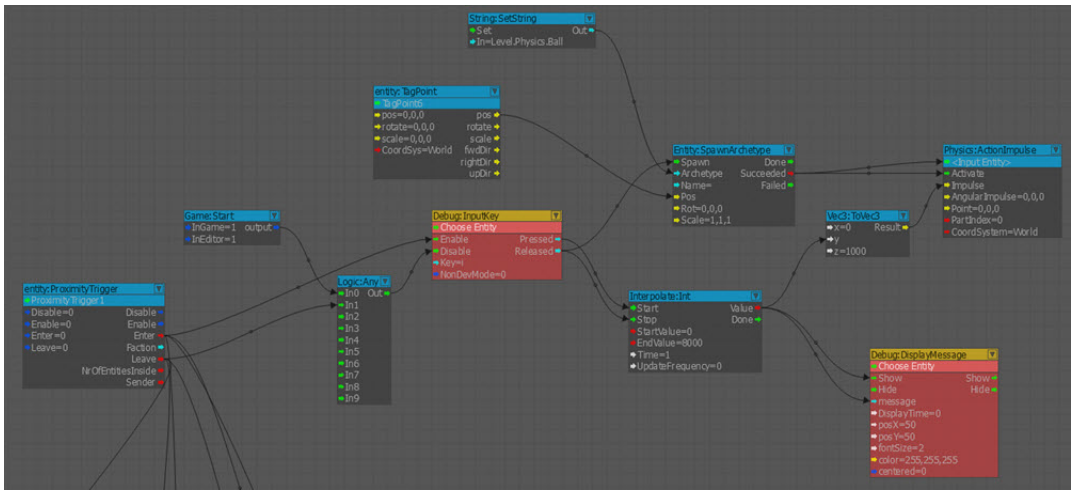
5. Drag the **Entity:ProximityTrigger** node's **Enter** output to each of the two **Debug:InputKey** nodes' **Enable** inputs.
6. Drag the **ProximityTrigger** node's **Leave** output each of the two **Logic:Any** node's **In1** inputs.



The ability to move the block left and right is now active only when the Proximity Trigger is activated. The inputs are disabled when leaving the proximity triggers.

The **Any** 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 multiple inputs to be collated into a single output.

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

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, select **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. Select 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 that launches a ball at a wall of blocks.

Conclusion

This concludes Lumberyard's Getting Started Guide Tutorial. Using this set of tutorials, you have begun to understand the power of Lumberyard Editor. You learned about and used features that you will use regularly while making your games. When you are comfortable with these basic features, browse our tutorial pages to learn even more about Lumberyard's advanced tools and features.