

## Using the ArcGIS Maps SDK to Create Real-World Experiences in Unity

### Overview

This tutorial will demonstrate how to use ArcGIS Pro to prepare road and building layers for loading into the ArcGIS Maps SDK for Unity. Once the building and ground layers have been added to the Unity scene, participants will learn how to adjust the lighting to simulate a certain time of day. The tutorial will conclude with participants using their scene as the foundation of a racing game by adding a vehicle controller to drive to checkpoints placed on the map. A bonus section has been added at the end describing how the buildings and track can be aligned to terrain and exported from ArcGIS CityEngine.

### Tutorial Length

1.5 hours

### Requirements

- Windows PC with the following:
  - Software installed and licensed:
    - ArcGIS Pro 3.0 (trial version can be downloaded from <https://www.esri.com/en-us/arcgis/products/arcgis-pro/trial>)
    - ArcGIS Maps SDK for Unity v1.1.0: <https://developers.arcgis.com/downloads/#unity>
    - Unity Hub: <https://unity3d.com/get-unity/download>
    - Unity 2021.3.x (Downloaded through Unity Hub)
  - Hardware:
    - Unity development minimum requirements: <https://unity3d.com/unity/system-requirements>
    - ArcGIS Pro system requirements: <https://pro.arcgis.com/en/pro-app/latest/get-started/arcgis-pro-system-requirements.htm>
- Mouse recommended for manipulating the 3D scenes

### Data Sources

- Toronto Massing Buildings: <https://open.toronto.ca/dataset/3d-massing/>
- Tutorial Data: <https://arcg.is/4envC>

### Production Date

The Education and Research Group at Esri Canada makes every effort to present accurate and reliable information. The Web sites and URLs used in this tutorial are from sources that were current at the time of production, but are subject to change without notice to Esri Canada.

- Production Date: February 2023

### Learning Outcomes

- Creation of 3D scene layers using ArcGIS Pro
- Basic manipulation of lighting and game objects in a Unity scene
- Using the ArcGIS Maps SDK for Unity to load layers into the scene

### References and Reading

- 3D in ArcGIS Pro: <https://learn.arcgis.com/en/paths/3d-in-arcgis-pro/>
- Unity Karting Microgame: <https://learn.unity.com/project/karting-template>
- CityEngine Fence Rule: <https://www.youtube.com/watch?v=qrelcLSeBzc>

# Introduction

This tutorial will introduce the ArcGIS Maps SDK for Unity by outlining the steps to create a racing track in ArcGIS Pro to load into Unity. Unity's Karting Microgame provides the racing functionality and building models from Toronto's Open Data Portal are used to create the environment around the track. The track is created by applying a rule package in ArcGIS Pro to a buffered line feature.





The tutorial is separated into three sections, beginning with the data preparation in ArcGIS Pro and followed by loading the data into the ArcGIS Maps SDK for Unity. The last section is an optional extension to the tutorial that outlines the steps needed to modify the racetrack's height to the terrain using ArcGIS CityEngine.

Before starting the tutorial, download and unzip the workshop material file. The download link for the zip file can be found on the following webpage: <https://arcg.is/4envC>

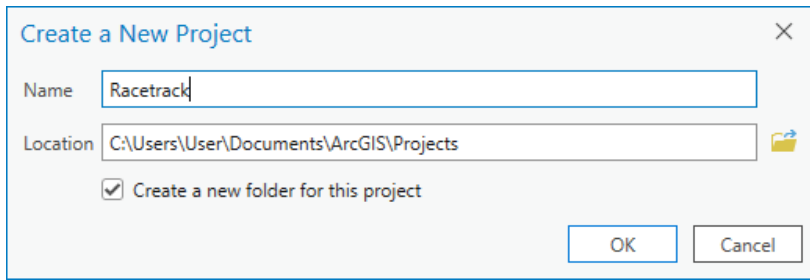
## Part 1: ArcGIS Pro Data Preparation

### Loading the 3D building models

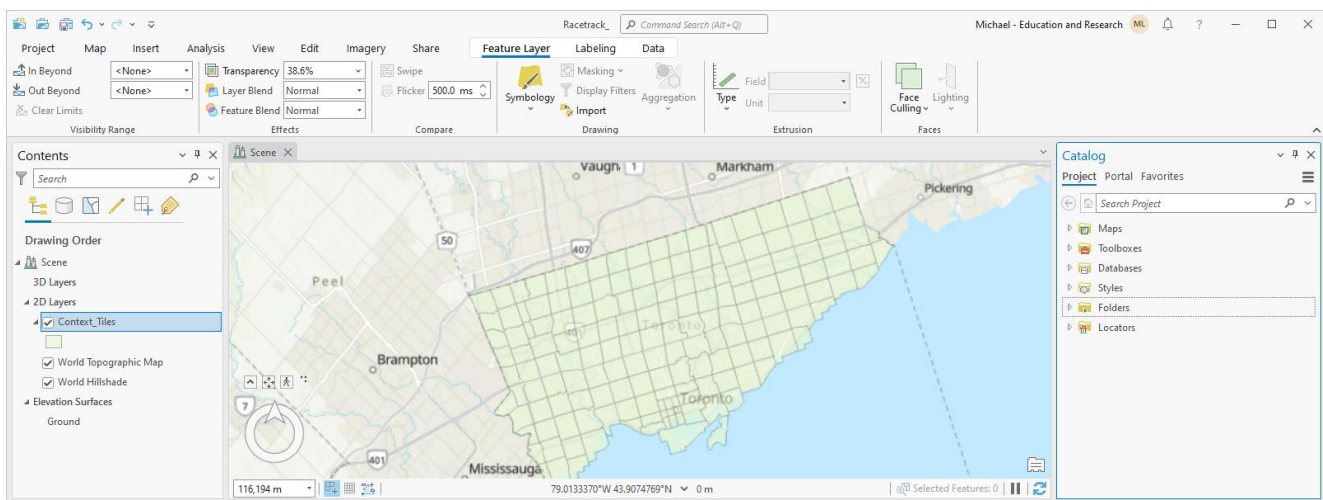
1. Visit <https://open.toronto.ca/dataset/3d-massing/> and expand the **DOWNLOAD DATA** section at the bottom of the page.
2. Click the button beside **3D Massing Multipatch (WGS84)** to download the Toronto buildings.

DATA PREVIEW			
Not available for this dataset			
DATA FEATURES			
DOWNLOAD DATA			
File	Format	Projection	Data
Interactive Map Tile Locator (AutoCAD, SKP)	web	Not Applicable	VISIT PAGE 
Tile Locator Reference File (PDF)	pdf	Not Applicable	DOWNLOAD 
3D Massing (WGS84)	shp	WGS84	DOWNLOAD 
3D Massing Multipatch (WGS84)	shp	WGS84	DOWNLOAD 
EXPLORE DATA			
FOR DEVELOPERS			

3. Unzip the file.
4. Start ArcGIS Pro. If you have not yet used ArcGIS Pro, you may be asked to sign in.
5. You will now create a new scene by clicking the **Local Scene** button under the **New Project** heading.
6. Name the project "Racetrack" and click OK.



7. Once the project has loaded, right-click on the **WorldElevation3D/Terrain3D** entry in the **Contents** panel and click **Remove**. The terrain is not needed because you will be creating a flat track surface to improve the driving experience in Unity.
8. Right click on the **Folders** entry on the Catalog panel and click **Add Folder Connection**.
9. Browse to the folder containing the data you previously unzipped. Click **OK** to create a connection to this folder.
10. Expand the geodatabase in the folder and drag the *Context\_Tiles* polygon layer to the scene to visualize the zones that split the 3D multipatches.
11. Click the **Feature Layer** tab on the ribbon and use the transparency slider to reveal the road network under the polygons.



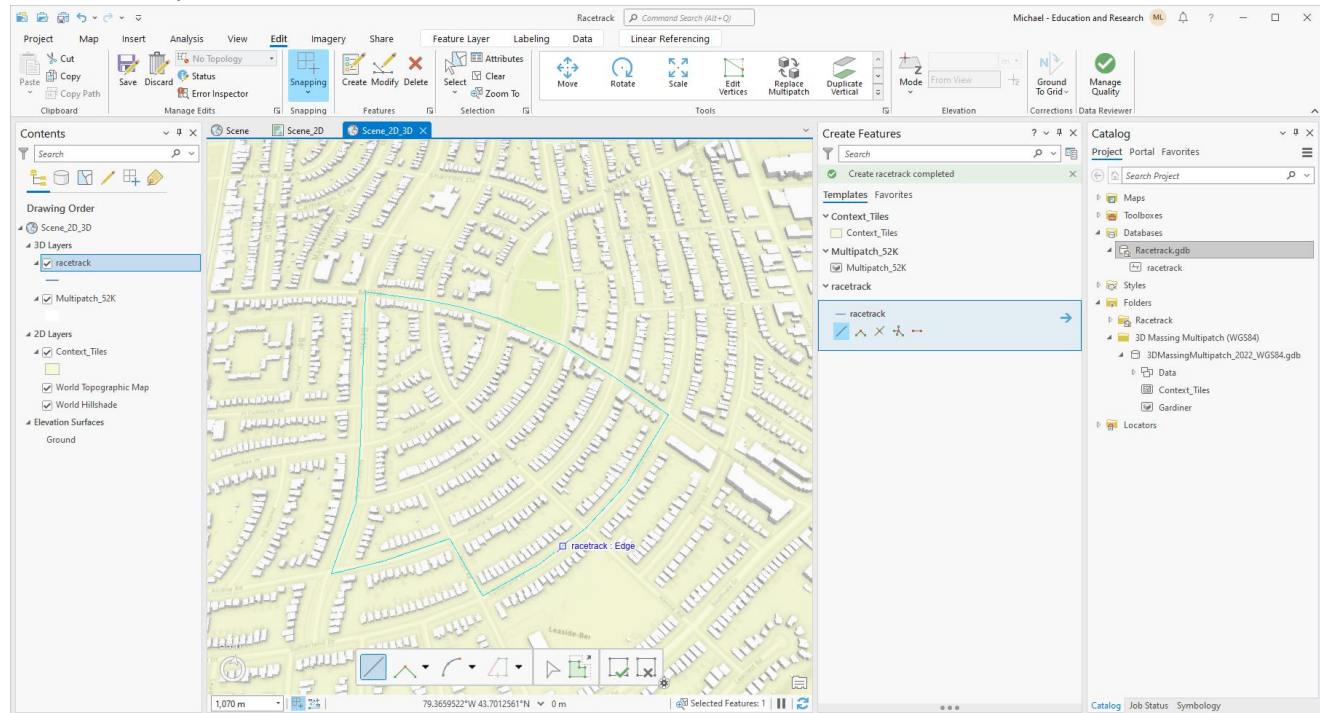
12. Click on an area in the city where you would like to create the racetrack. Make note of the *Tile\_Name* attribute on the Pop-up window that appears. If a Pop-up window doesn't appear, open the **Map** tab on the ribbon, click the down arrow on the **Explore** icon and choose **Topmost Layer**.
13. Expand the *Data* feature dataset in the 3D massing geodatabase and drag the multipatch with your tile name into the scene viewport.
14. Right-click on the new layer in the **Contents** panel and choose **Zoom To Layer** to view the buildings. You will export these buildings in this section's last step.

## Creating a racetrack on the map

You will now trace the path for your racetrack.

1. Right click on the *Racetrack.gdb* entry under the **Database** section of the **Catalog** panel and choose **New > Feature Class**.
2. Name the Feature Class "racetrack" and change the type to Line. Click the **Finish** button.
3. Click the **Edit** tab and then click on the **Create** button.
4. Choose the racetrack entry in the **Create Features** panel and trace a loop which will be the route for the racetrack. Ensure vertex snapping is enabled by clicking the snapping button on the bottom left of the viewport. To add the last

vertex, right-click on the starting vertex of your line and choose **Add Vertex**. Double click once you’ve added the last vertex to complete the track.

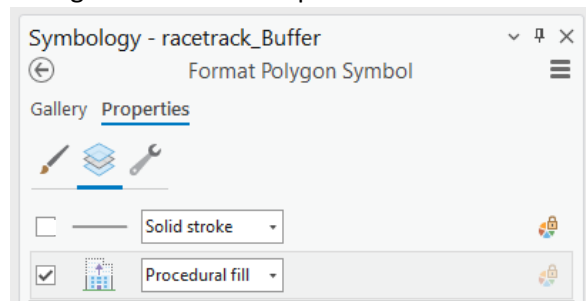


5. Click the **Save** button under the **Edit** tab and choose **Yes** on the confirmation box.
6. Click the **Analysis** tab and then the **Tools** button.
7. Search for and open the **Buffer** tool.
8. Choose the racetrack as the **Input Feature**, set the **Distance** value to 6 Meters, and click the **Run** button.

## Creating a fence around the racetrack

You will now apply a procedural rule to the racetrack layer to add a fence on the perimeter of the track.

1. Click the square representing the symbology under the *racetrack\_Buffer* layer in the **Contents** panel.
2. Click the **Properties** heading and then the **Layers** button.
3. Uncheck the **Solid stroke** layer. This layer will not be needed because you will be creating a fence for the track perimeter.
4. Change the **Solid fill** dropdown to **Procedural fill**.



5. Click the **Rule...** button and navigate to the downloaded “Fence\_On\_Polygon.rpk” rule package in the workshop material folder.
6. Click the Apply button at the bottom of the panel to see the fence rule applied to your track.
7. Expand the rule’s attributes and try changing the *myStyle* attribute to preview the different types of fences in the 3D window. You can also change the colour of the track by modifying the *groundColour* attribute. Click **Apply** in the **Symbology** panel once you have decided on the style you would like to use in your game.

## Exporting the racetrack

To create a scene layer package, you need to first convert the track layer to a multipatch.

1. Open the **Tools** panel and type “layer 3d” to open the **Layer 3D To Feature Class** tool.
2. Choose the *racetrack\_Buffer* as the **Input Feature Layer** and click the **Run** button.
3. Once the tool has executed, open the **Create 3D Object Scene Layer Content** tool.
4. Select the racetrack multipatch layer as the **Input Dataset** and define the **Output Scene Layer Package** as “track.slpk” in the output directory,
5. Click the **Run** button.

## Exporting the buildings to a scene layer package

The buildings also need to be saved to a scene layer package to load them into Unity.

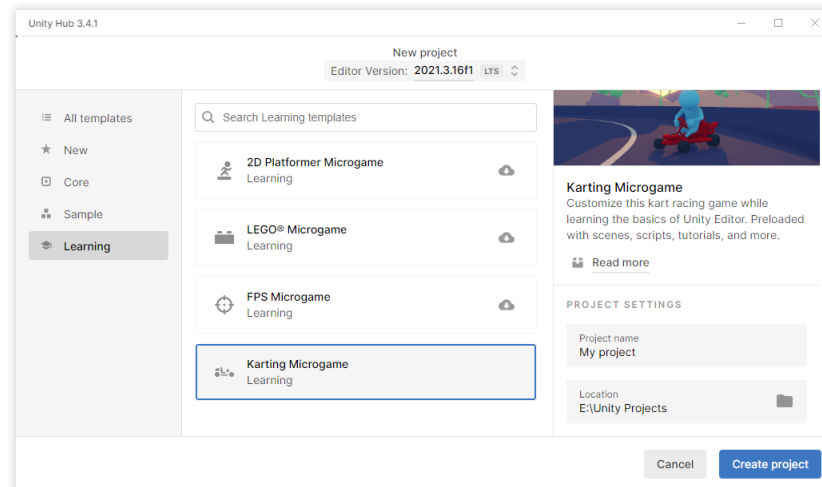
1. With the **Create 3D Object Scene Layer Content** tool open, select the buildings multipatch layer from the Input Dataset dropdown.
2. Set the Output Scene Layer Package to “buildings.slpk” in the output directory and click **Run**. This tool may take several minutes to run, so leave ArcGIS Pro open as you complete the next section.

## Part 2: Unity Karting Microgame and the ArcGIS Maps SDK for Unity

### Creating the Karting Microgame

You will be using a template that Unity has developed for learning the game engine.

1. Open Unity Hub and click the **New Project** button.
2. Choose the **Karting Microgame** under the **Learning** templates and click the **Download template** button.
3. Once downloaded, name the project “Racetrack” and click the **Create project** button.



### Learning the basics of the Unity editor

Once the project has loaded in the editor, click the **Load Tutorials** button, and follow through the first four sections as follows:

1. Get Started
2. Editor basics
3. Change colours
4. Add a jump

### Importing the ArcGIS Maps SDK for Unity

Now that you have completed the tutorials to familiarize yourself with Unity, you can download and install the ArcGIS Maps SDK for Unity.

1. Visit <https://developers.arcgis.com/downloads/#unity> to download version 1.1.0 of the Maps SDK for Unity. You will need to sign in with your ArcGIS Online or developer account.
2. In Unity, click **Windows > Package Manager**.
3. Click the + button in the Package Manager and select **Add package from tarball...**
4. Navigate to the downloaded Maps SDK and click **Open**.
5. Close the package manager once the import is complete.

### Create an API key on ArcGIS Online

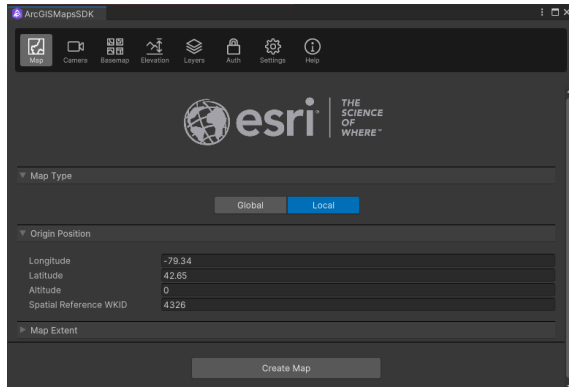
To access the ArcGIS location services such as basemaps used in this tutorial, you need to create an API key on ArcGIS Online.

1. Open <https://developers.arcgis.com> in a browser and sign into your ArcGIS Online account. If you are already signed in, click the **Dashboard** link in the top menu.
2. Click **API Keys** in the top panel.
3. Click the **New API Key** button on the top left of the page.
4. Name the API Key “Unity Maps SDK” and click **Create API Key**.
5. Click the copy button.



## Adding the map to the scene

1. Click on **ArcGIS Maps SDK > Map Creator** to load the ArcGIS Maps SDK dialog window.



2. Choose a local scene and enter -79.34 as longitude and 43.65 for latitude to place the origin in Toronto.
3. Click the **Create Map** button.
4. Open the **Camera** section and click the **Create Camera** button.
5. Open the **Elevation** section and deselect **Default Elevation** to turn off the terrain.
6. Open the **Layers** section and add the two scene layer packages you created with ArcGIS Pro. Make sure you change the **Type** dropdown to **ArcGIS 3DObject Scene Layer** for the building and fence scene layer.
7. Click the **Auth** button and paste your API key into the text box.
8. Open the **Settings** section and check the **Enable Mesh Colliders** checkbox.
9. Open the **Basemap** section and choose a basemap you would like to use for your game.
10. Close the ArcGISMapsSDK dialog. If you zoom out in the editor with the scroll wheel, you should now see the basemap tiles for Toronto.

## Modifying the scene's game objects to work with ArcGIS Maps SDK for Unity

The project has several game objects that are not needed since you are using the Maps SDK to create the track.

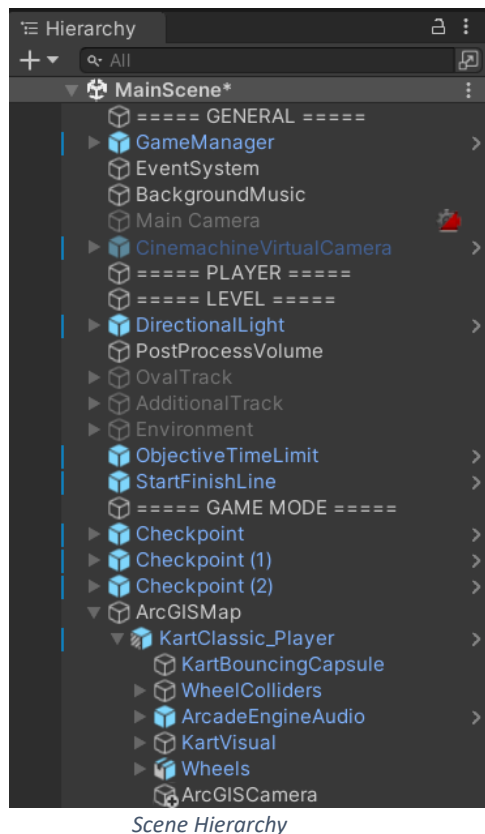
1. Hide the following game objects by selecting them in the **Hierarchy** panel and unchecking the checkbox in the **Inspector** panel:
  - *Main Camera*
  - *CinemachineVirtualCamera*
  - *OvalTrack*
  - *AdditionalTrack*
  - *Environment*
2. Some of the game objects need to be moved in the scene to align with the racetrack.
  - Drag the *KartClassic\_player* to the *ArcGISMap* game object in the **Hierarchy** panel. The *KartClassic\_player* should now be a child element of the *ArcGISMap* game object.
  - Drag the *ArcGISCamera* to make it a child element of the *KartClassic\_player* game object and set the Position transform of the camera object to (0,0,0) in the **Inspector** panel.



- Space the *Checkpoint* game objects along the track and move the *StartFinishLine* and *KartClassic\_player* game objects to the starting position of the track. You can use the translate tool by selecting the game object in the **Hierarchy** panel or **Scene** viewport and pressing W on the keyboard to bring up the translation arrows. These arrows can be dragged to position the objects in the scene.
- Select the *ArcGISCamera* game object and use the translate and rotate tools (W and E keys) to position it centred behind and facing the go-kart. When the camera component is selected, you can preview the camera in the viewport as shown below:



Your game hierarchy should now look like the following screen capture:



3. From the menu bar, select **GameObject > 3D Object > Plane** to create a new plane in the scene.
4. This plane will be used as a collider on the ground so the kart does not fall through the basemap. Use the translate tool to move the plane to the center of your track.



5. Use the scale tool (R key) to expand the size of the plane to cover your track. You can also use the rotate tool to orient the plane in the direction of the track.
6. Uncheck the **Mesh Renderer** on the plane so it doesn't interfere with the visibility of your track layer.
7. To ensure the game sound works, click on the **ArcGISCamera** object and in the **Inspector** panel, click the **Add Component** button to add an **Audio Listener**.

## Testing the racetrack

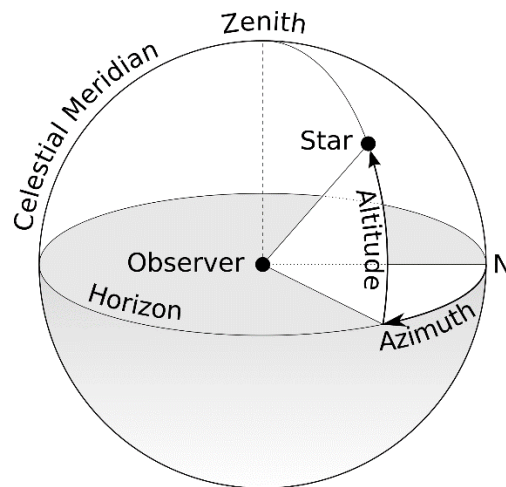
Click the play button at the top of the Unity editor window to test the game. You should be able to make it all the way around the track before the time expires. If the time does expire, exit the Game mode by clicking the play button and select the *ObjectiveTimeLimit* game object to increase the **Total Time** in the **Inspector**.

## Modifying the skybox and setting the time of day

The Karting Microgame includes a cartoonish skybox. To change this to a more realistic environment you can create a procedural skybox.

1. In the **Project** panel, right-click in the **Assets** folder and select **Create > Material**.
2. Name the new material "skybox" and click on it to open it in the **Inspector** panel.
3. Change the **Shader** dropdown to **Skybox > Procedural**.
4. Open the **Lighting** window by selecting **Window > Rendering > Lighting**.
5. Click the **Environment** tab and set the **Skybox Material** to the procedural skybox by dragging the material from the **Project** panel to the skybox material slot.
6. Change the **Environment Lighting Source** to **Skybox** in the **Lighting** panel.
7. Click the *DirectionalLight* in the **Hierarchy** panel. Expand the **Emission** section of the **Light** in the **Inspector** panel and use the colour picker to change the **Light Appearance** from orange to white.

The position of the sun can be modified by changing the rotation vector of the directional light. The X and Y axis of the vector change the altitude and azimuth values, as illustrated on the following diagram:



TWCarlson ([https://commons.wikimedia.org/wiki/File:Azimuth-Altitude\\_schematic.svg](https://commons.wikimedia.org/wiki/File:Azimuth-Altitude_schematic.svg)), Azimuth-Altitude schematic, <https://creativecommons.org/licenses/by-sa/3.0/legalcode>

You can calculate the altitude and zenith for a certain time of day using a calculator found here:

<https://www.suncalc.org/#/43.6614,-79.3769,13>

You can adjust the time of day and the calendar date on the top and left sections of the web page. Copy the altitude value as the X rotation value. Subtract 180 from the Azimuth value and set this as the Y rotation value.

## Building your Unity app

Before building the Unity app, you will need to copy your scene layer packages to a *StreamingAssets* folder in the project and add a script that will set the file path of the layers when the executable is launched.

1. Open the **Project** panel, right-click on the **Assets** folder and select **Create > Folder**. Name the folder "StreamingAssets".
2. Copy the two scene layer packages from Windows Explorer into the *StreamingAssets* folder.
3. Click on the *GameManager* object in the **Hierarchy** panel and click the **Add Component** button in the Inspector window.
4. Scroll down and click **New script** and type in "loader". Click the **Create and Add** button.
5. Right click on the **loader (Script)** entry and choose **Edit Script**.
6. Add the following declarations at the top of the file:

```
using System.IO;
using Esri.ArcGISMapsSDK.Components;
```

7. Copy the following code into the "loader.cs" file to replace the empty Start function:

```
void Start(){

    #if !UNITY_EDITOR
        ArcGISMapComponent mainmap = GameObject.Find("ArcGISMap").GetComponent<ArcGISMapComponent>();
        List<ArcGISLayerInstanceData> newLayers = new List<ArcGISLayerInstanceData>();
        List<ArcGISLayerInstanceData> layers = mainmap.Layers;
        int index = 0;
        foreach (ArcGISLayerInstanceData layer in layers)
        {
            string filename = Path.GetFileName(layer.Source);
            if (filename.IndexOf(".") != -1) {
                string newPath = Application.dataPath + "/StreamingAssets/" + filename;
                ArcGISLayerInstanceData layerData = mainmap.Layers[index];
                layerData.Source = newPath;
                newLayers.Add(layerData);
            }
            index += 1;
        }
        mainmap.Layers = newLayers;
    #endif
}
```

8. Click **File > Build Settings...** and click the **Build** button.
9. Create a new folder to place the executable and wait for the app to build. Click **Save** if a dialog appears asking to save the scene.

## Next Steps

Additional modifications to the game can be explored by selecting **Tutorials > Show Tutorials** to open the **Tutorials** panel. Click on the **Microgame Mods** item in the panel to view the additions you can add to the game.

# Bonus Exercise: Adding Elevation to the Track and Buildings with ArcGIS CityEngine

ArcGIS CityEngine is a 3D tool that uses rules for procedural modelling. This means that rules and parameters drive the creation of 3D forms. Whether you are creating a building, street network, landscape feature or infrastructure element, all 3D forms are generated through the same rule-based system and can be adjusted dynamically in real-time through user-entered or database-driven values.



*CityEngine creates 3D models by combining multiple data layers*

The track rule package used in this tutorial was created by combining CityEngine's built-in street and fence rules to create a rule package for use in ArcGIS Pro. The track rule package is meant to work with polygons, which for this tutorial were the buffered lines representing the track.

CityEngine allows lines to be used as start shapes in addition to polygons. In this section, you will import the line network representing the track into CityEngine, align it to Esri's World Elevation Service, and apply a version of the fence rule tailored for line features.

This section outlines the high-level steps needed to create the terrain aligned scene layer packages and does not walk through all the needed actions in the software. Thus, experience with CityEngine is recommended to complete this part of the tutorial. A beginner tutorial for CityEngine can be found here:

<https://doc.arcgis.com/en/cityengine/latest/tutorials/tutorial-1-essential-skills.htm>

## Creating Terrain-Aligned Scene Layers in CityEngine

### 1. Create a new project and scene in CityEngine

The tutorial linked above provides an overview for creating a new CityEngine project and scene. Once the project is created, merge the *assets* and *rules* folders from the *CityEngine Rules* directory of the workshop zip into your CityEngine project's directory structure.

### 2. Import racetrack line into CityEngine

Import the track line feature you created in ArcGIS Pro using the ArcGIS Pro project's file geodatabase.

### 3. Import the buildings into CityEngine

Import the buildings by importing from the downloaded file geodatabase containing the Toronto massing multipatches.

### 4. Use the Get Map Data tool to download the terrain

The **Get Map Data** tool allows you to download terrain from Esri's World Elevation Service. Select an area around your study area to download the elevation model. You can deselect downloading the Open Street Map networks and footprints.

### 5. Align the street network to the terrain

Select all the roads in the viewport and run the **Align Graph to Terrain** tool. Choose your heightmap layer to align the graph and add an offset of 0.3 to ensure the terrain does not protrude through the track. With both the

street network and ground layer visible, check to see if the road disappears below the ground. This can be resolved by creating vertices in the middle of the line segment and translating them to a higher elevation with the translate tool or by running the align graph tool again.

6. [Align the buildings to the terrain](#)

Select all the buildings and open the **Align Shapes to Terrain** tool. Choose to **Translate to Maximum** and select your terrain layer as the heightmap.

7. [Assign the fence rule to the street network](#)

The fence rule uses the sidewalk shape to generate the barriers on the sides of the road and is named *Fence\_On\_Graph.cga* in the *rules* folder. An added benefit of using the graph network instead of the polygon buffer is that the street's road markings can be aligned and textured along the road.

8. [Adjust the fence](#)

You may want to adjust the vertices representing the road to ensure that the road does not intersect buildings. Now that the rule is applied to individual parts of the road segment, you can also try changing the *myStyle* attribute for different part of the track to alternate the fence type. You can also change the width of the sidewalk segments to 0 in the line segment's shape parameters to remove the fence for portions of the road.

9. [Export the racetrack and buildings as scene layer packages](#)

The track model can be exported from CityEngine by selecting the track model, choosing **File > Export Models...** and selecting **Esri Scene Layer Package**. The same process can be followed for the buildings layer.

10. [Load the new scene layer packages and enable elevation in Unity](#)

The Maps Creator in Unity (**ArcGIS Maps SDK > Map Creator**) can be used to enable the default elevation and to replace the building and track layers with the CityEngine generated layers.

11. [Align the game objects to the correct height](#)

Use the translate tool in Unity to elevate the kart player, plane collider, checkpoints and finish line to the correct height on the terrain.

## Next Steps

If you would like to learn more about how the procedural rule for the fence and track works, you can try using the model hierarchy tool in CityEngine to inspect the fence rule: <https://doc.arcgis.com/en/cityengine/latest/help/help-model-hierarchy.htm>.

# Summary

---

You now know how to digitize line features in ArcGIS Pro and how to apply procedural symbology to create multipatch features and scene layer packages. In Unity, you've learned how to manipulate the position, orientation, and scale of game objects and how to setup the ArcGIS Maps SDK for your study area. You've also learned the basics of adjusting the lighting to realistic conditions.

If you tried the bonus exercise, you will also have knowledge of how ArcGIS CityEngine can be used to prepare scene layer packages by importing features from a geodatabase and applying rules to create textured roads.

More 3D GIS resources created by Esri Canada's Education and Research group can be found here:

<https://hed.esri.ca/resourcefinder/#/category=cityeng/lang=en>

---

[k12.esri.ca](https://k12.esri.ca) [hed.esri.ca](https://hed.esri.ca)

© 2023 Esri Canada. All rights reserved. Trademarks provided under license from Environmental Systems Research Institute Inc. Other product and company names mentioned herein may be trademarks or registered trademarks of their respective owners. Errors and omissions excepted.



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/). Some content in this tutorial is derived from Esri's GeoInquiries template, available at <http://www.esri.com/connected> under a Creative Commons Attribution-NonCommercial-ShareAlike license.

