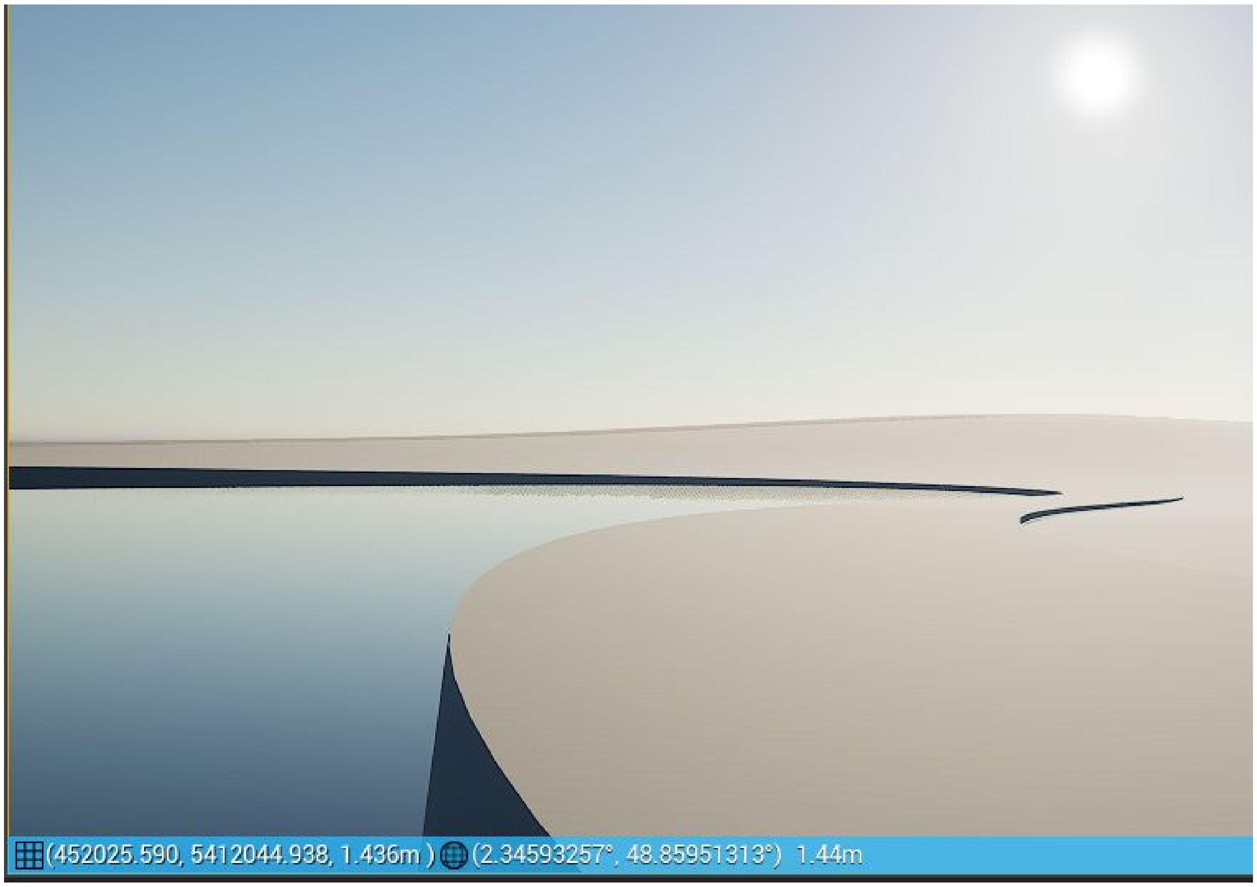
GeoReferencing Tools

Contained in Simulation plugin

Goal

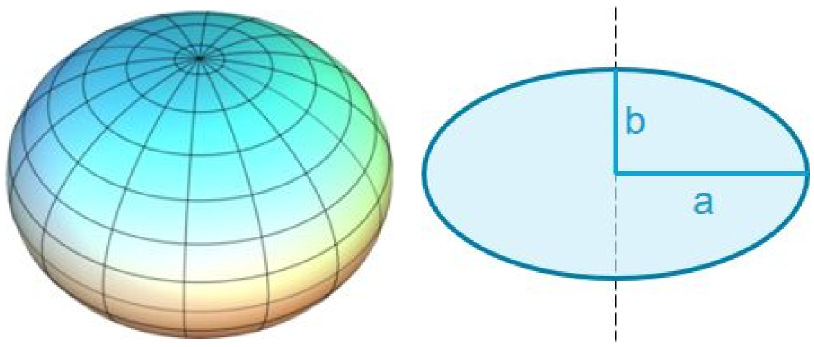

(452025.590, 5412044.938, 1.436m) (2.34593257°, 48.85951313°) 1.44m

Provide tools for users to express Coordinates in different Geographic Coordinates Reference Systems

Background

When it comes to locating objects on Earth, one must choose a reference coordinate system (CRS) to express the location.
https://en.wikipedia.org/wiki/Geographic_coordinate_system

What makes it complicated is that Earth is not flat, neither a sphere : It's an ellipsoid. There are different models of ellipsoids (aka datum), the most famous one being the WGS84.



$$f = \frac{a - b}{a}$$

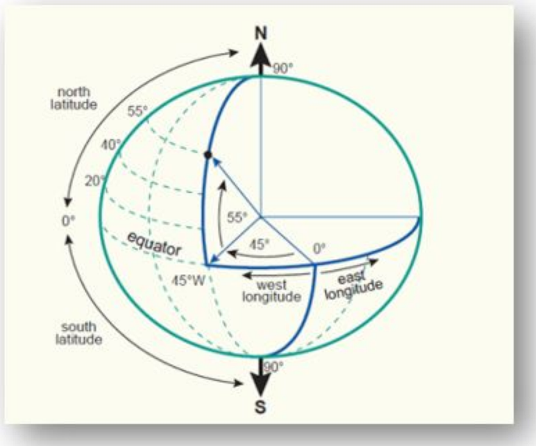
For WGS84 :

- a = 6 378 137.0 m
- b = 6 356 752.314 245 m
- 1/f = 298.257 223 563

Basically, there are 3 main kinds of CRS.

Geographical Coordinate system

Sometimes misnomer WGS84...



Polar coordinates with height.
Origin = Earth center
Latitude* = Elevation / Equator
Longitude = Azimuth / Greenwich
Altitude = height / Datum.

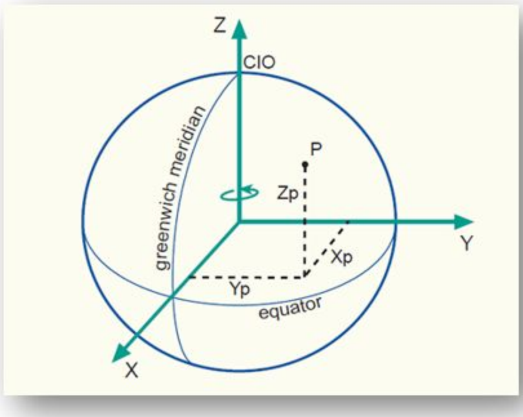
* There are in fact two kinds of latitude, when the datum is not a sphere, local normal to the ellipsoid model is involved

Angles are expressed in decimal degrees or DMS (degrees, Minutes Seconds)
Altitude in meters.

See https://en.wikipedia.org/wiki/World_Geodetic_System#WGS84

Geocentric Coordinates System

Example : ECEF (earth-centered, earth-fixed)



Cartesian coordinate system
Origin = Earth center
X pointing to Equator/Greenwich intersection
Z being the earth rotation axis
Y orthogonal to the two formers

Coordinates are expressed in meters

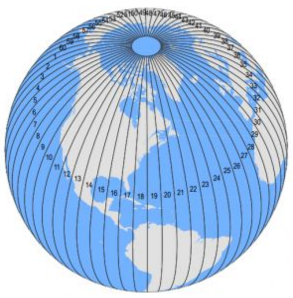
This CRS is the reference one in distributed simulation protocols such as DIS/HLA.
See https://en.wikipedia.org/wiki/ECEF

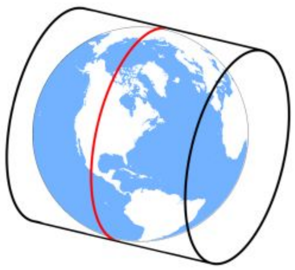
Projected Coordinates System

A projected coordinate system is a Geographic Coordinate system that has been flattened using a map projection. There are many map projections which come with more or less deformation.

Most frequently used is the Mercator projection which as several variants :
https://en.wikipedia.org/wiki/Mercator_projection
In simulation industry, the Universal Transverse Mercator (UTM) is widely used

https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system
It divides earth into 60 zones and projects each to the plane as a basis for its coordinates





Cartesian coordinate system, but zone and hemisphere has to be identified to define origin.
Each 6° wide UTM zone has a central meridian of 500,000 meters. This central meridian is an arbitrary value convenient for avoiding any negative easting coordinates. All easting values east and west of the central meridian will be positive.

If you're in the northern hemisphere, the equator has a northing value of 0 meters.
In the southern hemisphere, the equator starts at 10,000,000 meters. This is because all values south of the equator will be positive. This is called a false northing because y-coordinates in the southern geographic region will avoid negative values.

There are derivatives of this system such as the Military Grid Reference System (MGRS)
https://en.wikipedia.org/wiki/Military_Grid_Reference_System

But what with game engines ?

Game engines, as other rendering engines work in cartesian coordinate systems. It will not be possible to express geometry coordinates in a geographic CRS, at least to communicate with a GPU.

Another issue is coordinates precision. GPUs are working with 32bits floats, and precision is limited when more than seven significant numbers are required.
If you want to know why, this page is self explanatory :
https://www.h-schmidt.net/FloatConverter/IEEE754.html

This issue is solved by solutions out of this document scope (double precision, rebasing, and other…). But from an user perspective, it's important to use "real-life" coordinates rather than the ones of the game engine. This is the goal of this plugin.

Most of the time, projected systems can be used, as long as earth curvature and distance deformation is not a problem. For bigger scales, ECEF has to be used.

Currently, this early version of the tool is suitable for projected terrain only. It will evolve over time to support whole earth datasets using ECEF CRS.

Features

Provide functions to the user

- In editor : Used in an utility widget to display Viewport and Point under mouse cursor coordinates, in each defined CRS, to query for coordinates or check if actors are located at the right position,
- At runtime : Can be used to compute transformations, or display coordinates in an UMG widgets

System can take any kind of CRS. See below.

Users can define global settings for the whole project, or local settings related to a particular Level, using a dedicated actor. (eg : When a single project contains several environments in different locations)

Current limitations (for now..)

- Currently, only “Flat” projected terrains are supported. The Unreal world is considered in a projected CRS.
- UE4 World rebasing is not supported.

How to use this plugin ?

Installation

- Make sure your project is a C++ project (by adding a C++ class for instance)
- Copy this repository content into a Plugin subfolder of your project.
- Build it, and make sure it’s enabled for your project using the Plugins panel.

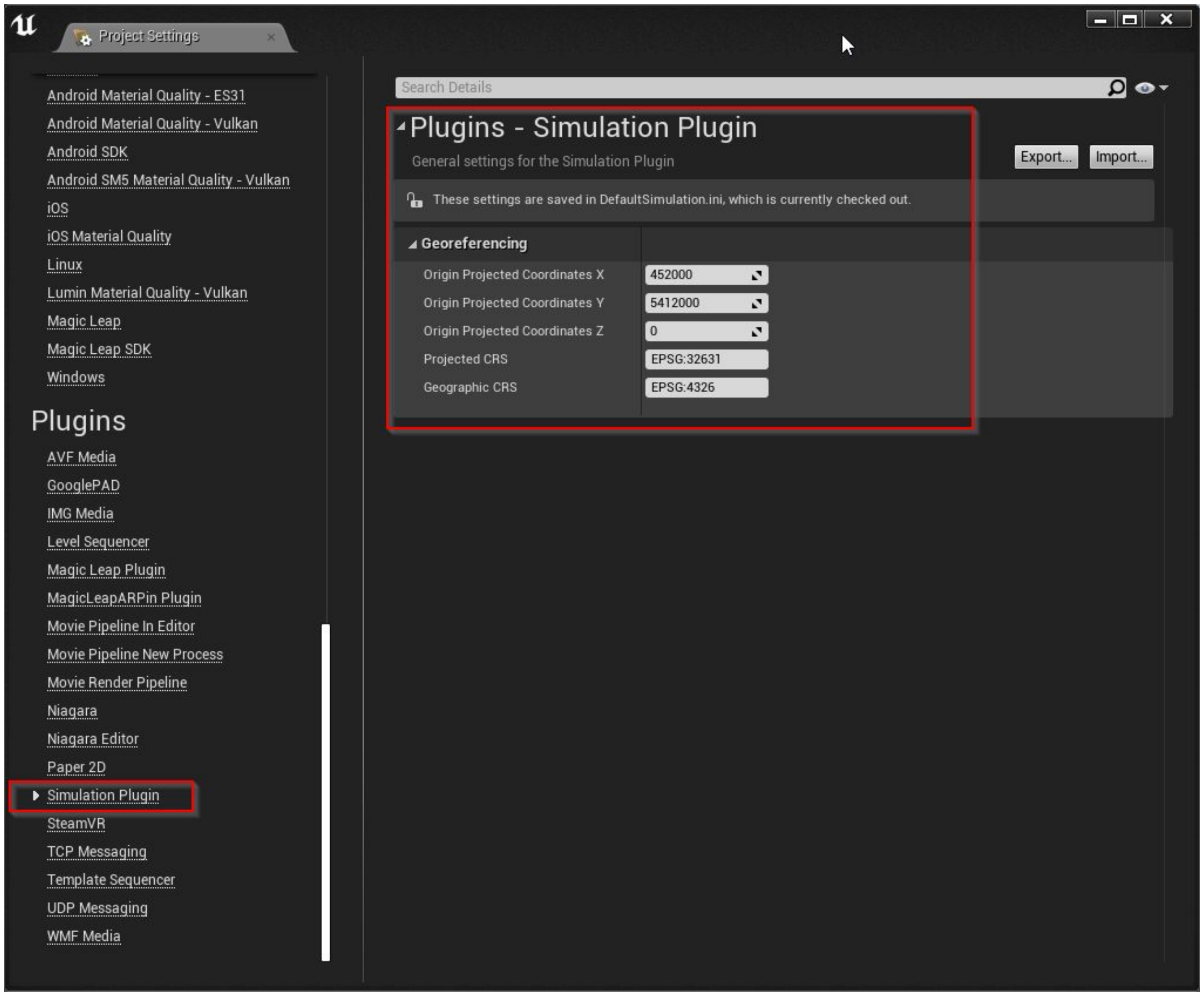
This will add several components to your project :

- A new world subsystem **UGeoReferencingSubsystem**
- A new settings section
- A new kind of Actor, **AGeoreferencingActor**.
- An Editor utility widget to display coordinates in Editor.
- A sample UMG widget to display coordinates at runtime

Global settings

Create at least one level in your project. For now, we'll make the assumption that we are working in a projected CRS of your choice.

Go to project settings, in the Simulation Plugin page, and fill the relevant information there :



Plugins - Simulation Plugin

General settings for the Simulation Plugin

These settings are saved in DefaultSimulation.ini, which is currently checked out.

Georeferencing

Origin Projected Coordinates X	452000
Origin Projected Coordinates Y	5412000
Origin Projected Coordinates Z	0
Projected CRS	EPSG:32631
Geographic CRS	EPSG:4326

- Origin Coordinates are the projected coordinates of the reference point that you will choose to be at the level origin.
  - In other words, the offset vector between UE CRS and Projected CRS.
  - It has to be integral, to keep precision.
  - It's recommended to round it at km steps.
- Projected/Geographic CRS = Strings to define the CRS you want to use in your application
  Can be of different kind :

- EPSG:code - see https://epsg.io/
- WKT (Well known Text) Wikipedia
- Any simple CRS name supported by PROJ (eg : WGS84, UTM 31N)

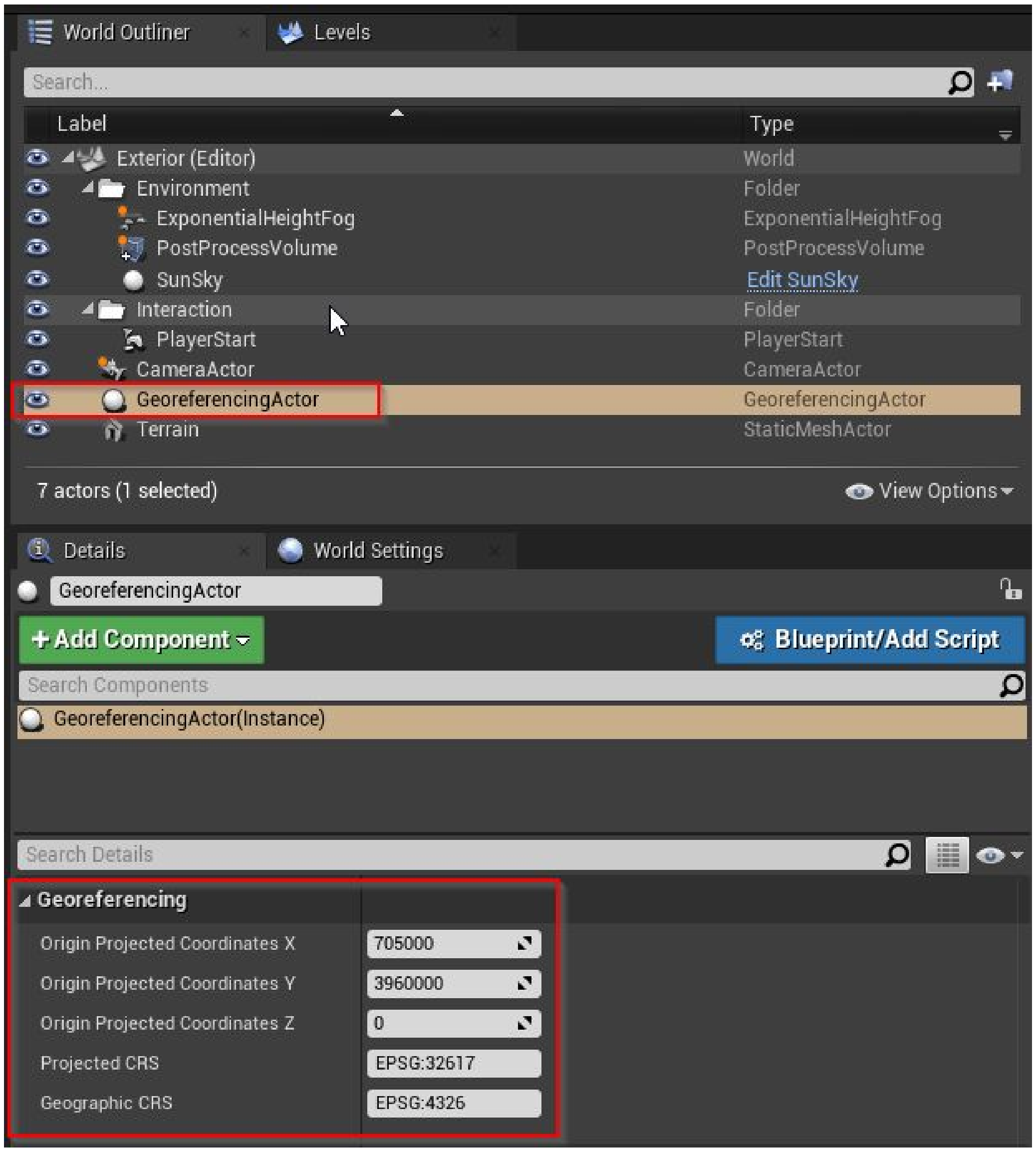
If a wrong string is provided, il will be logged as an error in the message log.

These settings will apply to all levels in your project.

Per-level settings

If you have several different levels that correspond to different locations on earth, you must have per-level settings. This is done by **GeoreferencingActor**

When present in the Level, this actor overrides the global settings.



Label	Type
Exterior (Editor)	World
Environment	Folder
ExponentialHeightFog	ExponentialHeightFog
PostProcessVolume	PostProcessVolume
SunSky	Edit SunSky
Interaction	Folder
PlayerStart	PlayerStart
CameraActor	CameraActor
GeoreferencingActor	GeoreferencingActor
Terrain	StaticMeshActor

7 actors (1 selected)

GeoreferencingActor

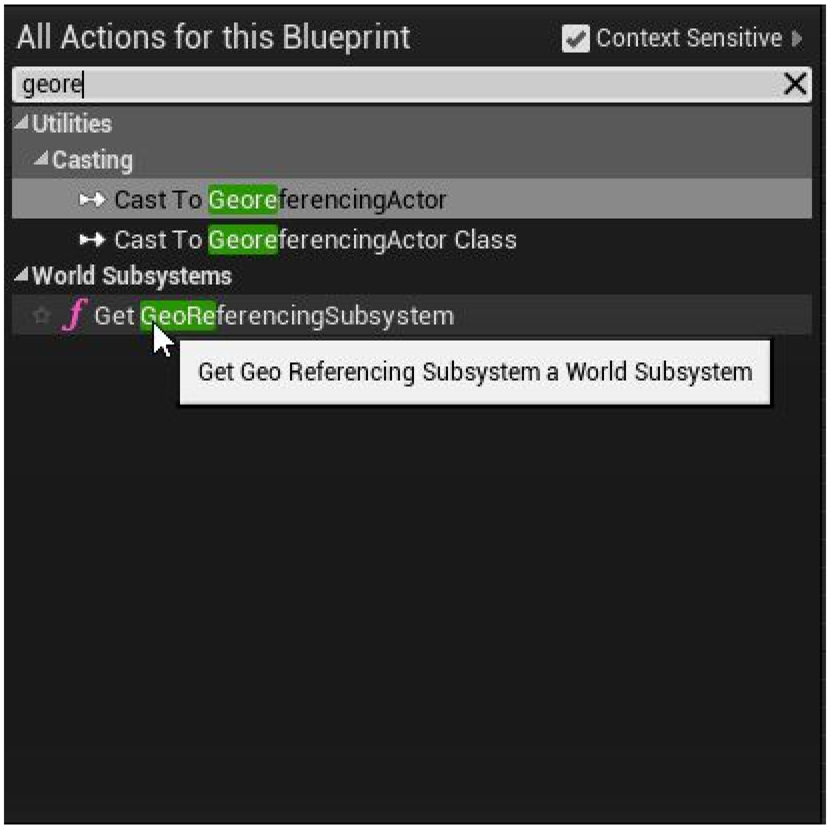
GeoreferencingActor(Instance)

Georeferencing

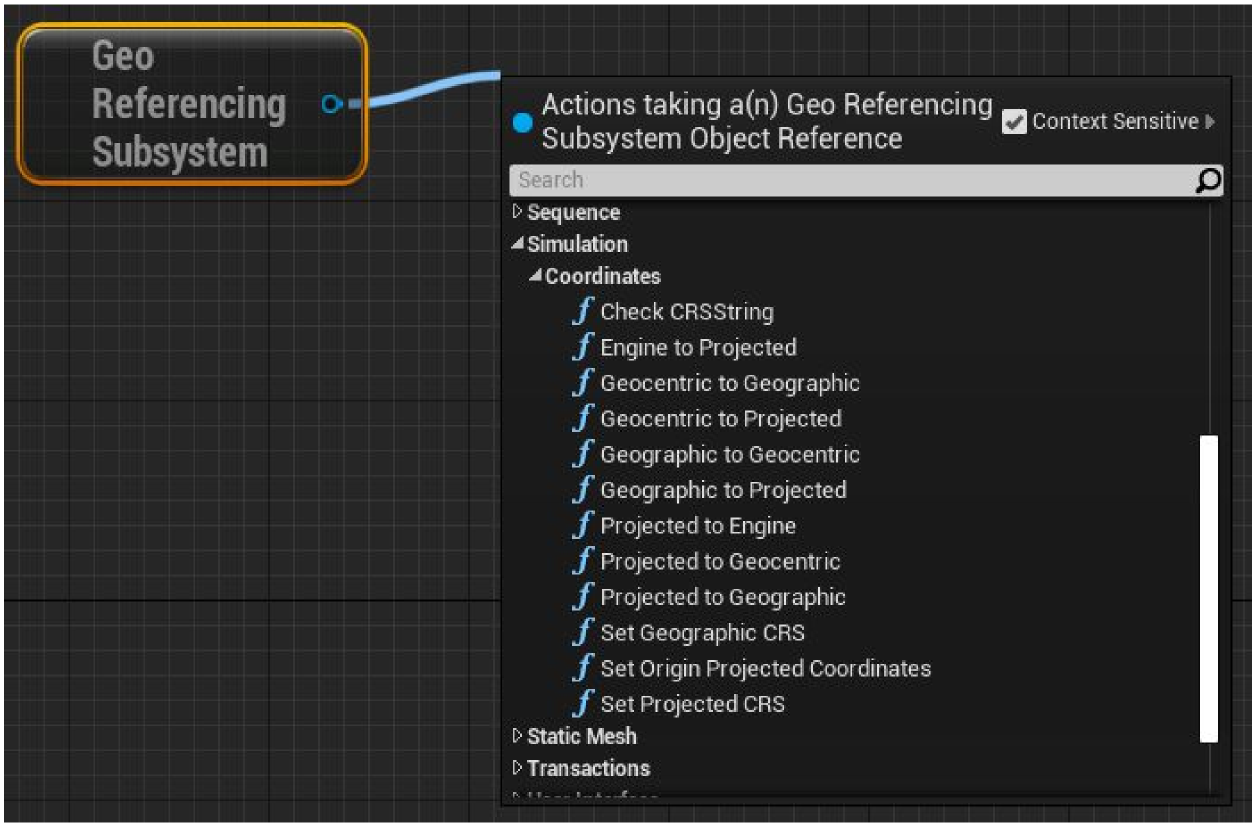
Origin Projected Coordinates X	705000
Origin Projected Coordinates Y	3960000
Origin Projected Coordinates Z	0
Projected CRS	EPSG:32617
Geographic CRS	EPSG:4326

Transform coordinates

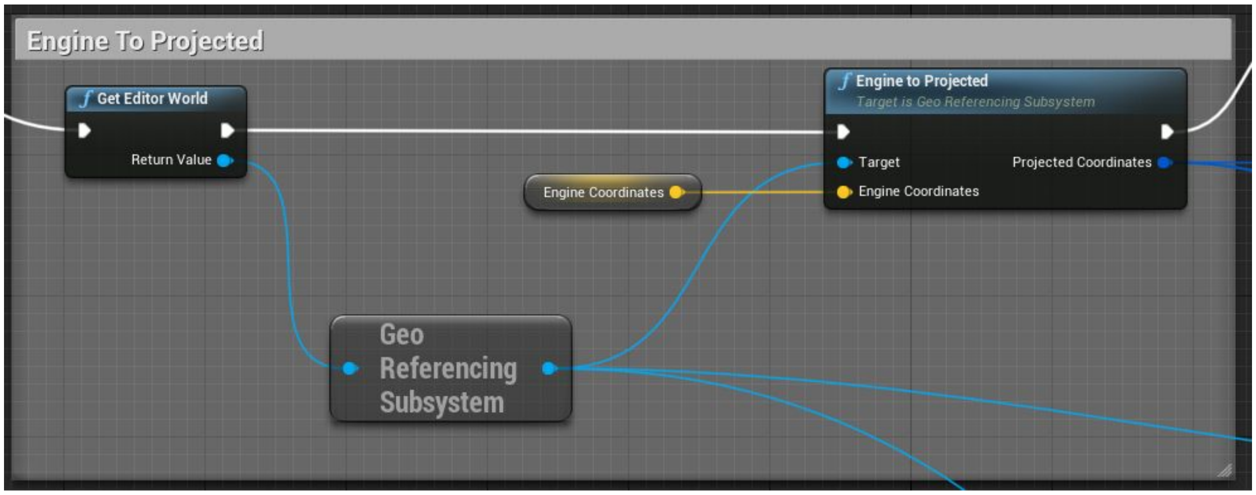
From inside a compatible blueprint editor, right click and find the GeoReferencing Subsystem. Depending on context, you might need to get a reference to the World first. (eg : from inside an Editor Utility widget)


All Actions for this Blueprint
Context Sensitive
geore
Utilities
Casting
Cast To GeoreferencingActor
Cast To GeoreferencingActor Class
World Subsystems
Get GeoReferencingSubsystem
Get Geo Referencing Subsystem a World Subsystem

Then drag a link and select the Coordinates operations you want to do.

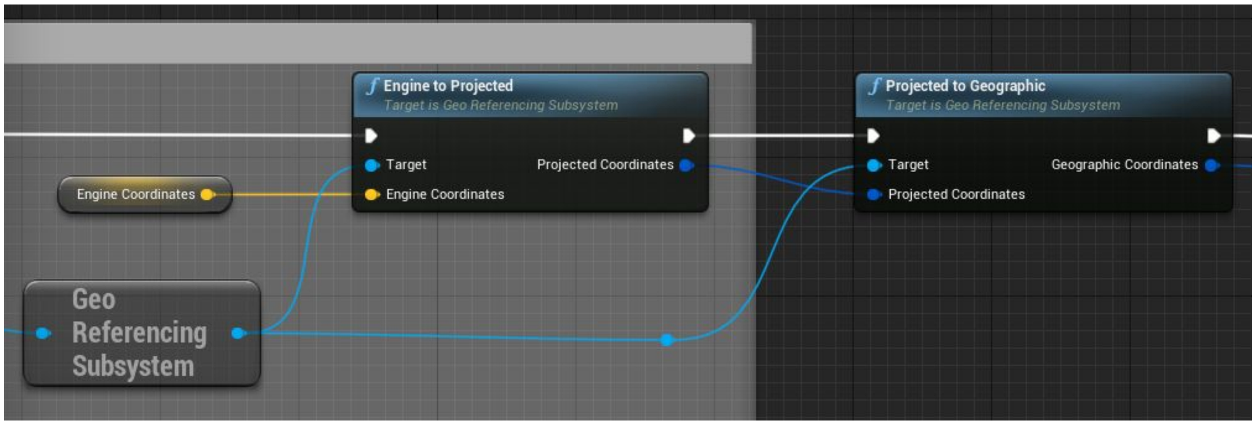


First step will always be to transform the UE coordinates in the projected system

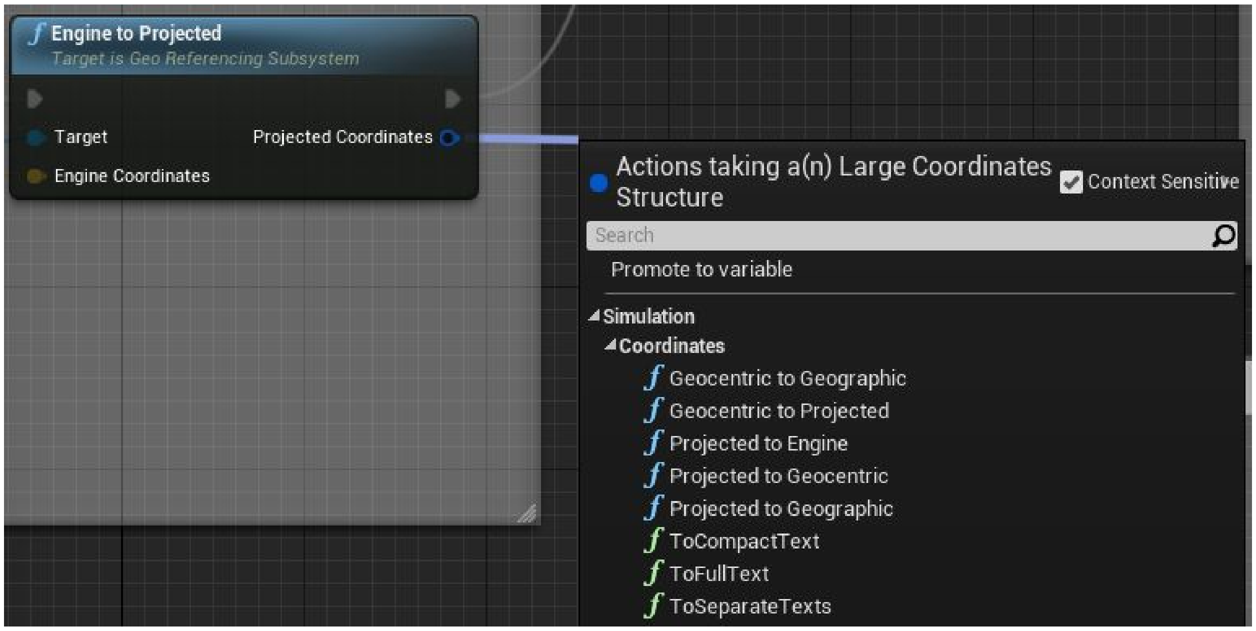


It will apply the origin rebasing and the scaling of UE Units.

Then you can convert these coordinates to Geographic or Geocentric coordinates.



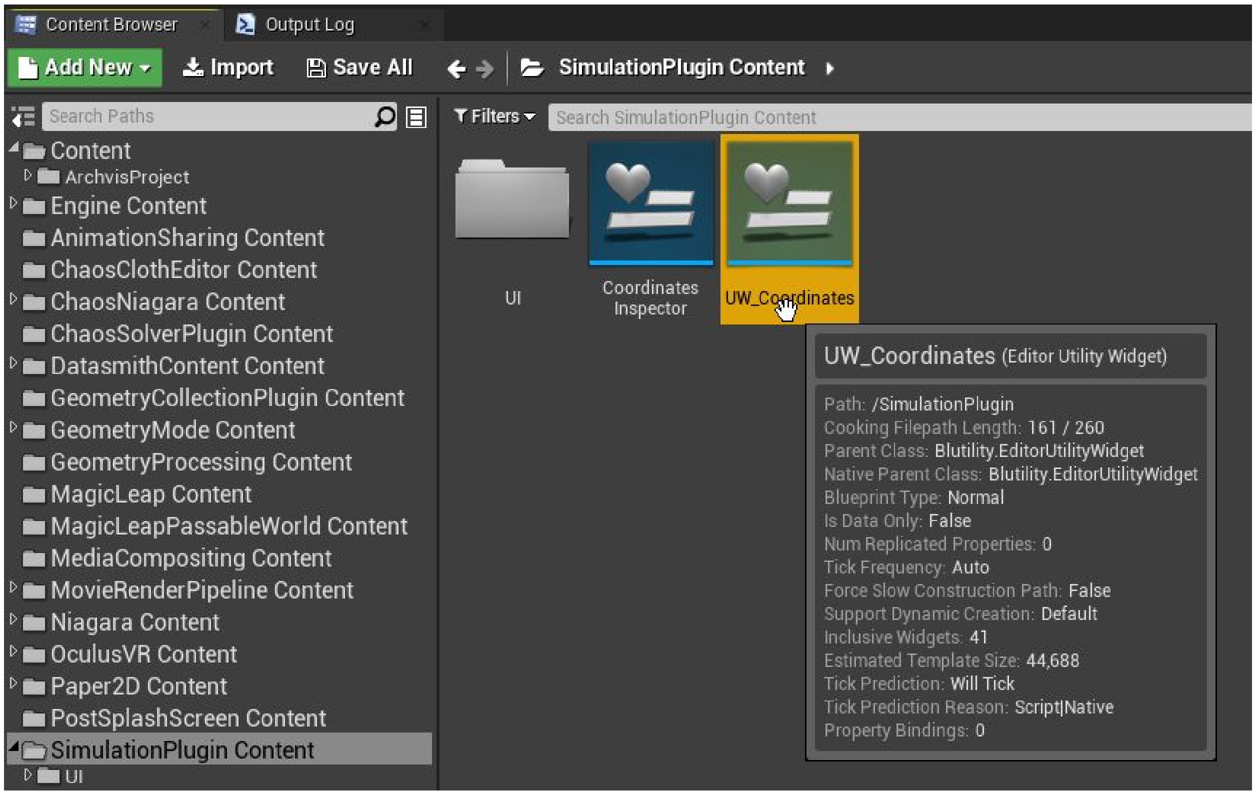
All computations are done in double precision to keep accuracy. The results are custom structs **FLargeCoordinates** and **FGeographicCoordinates.**
As blueprints don't support double, you can't get these properties, but ToString conversion methods are provided in a BPFunctionLibrary



Editor Utility Widget

The plugin comes with an utility widget widget that (Once ran) displays Viewport and Point under mouse cursor coordinates.

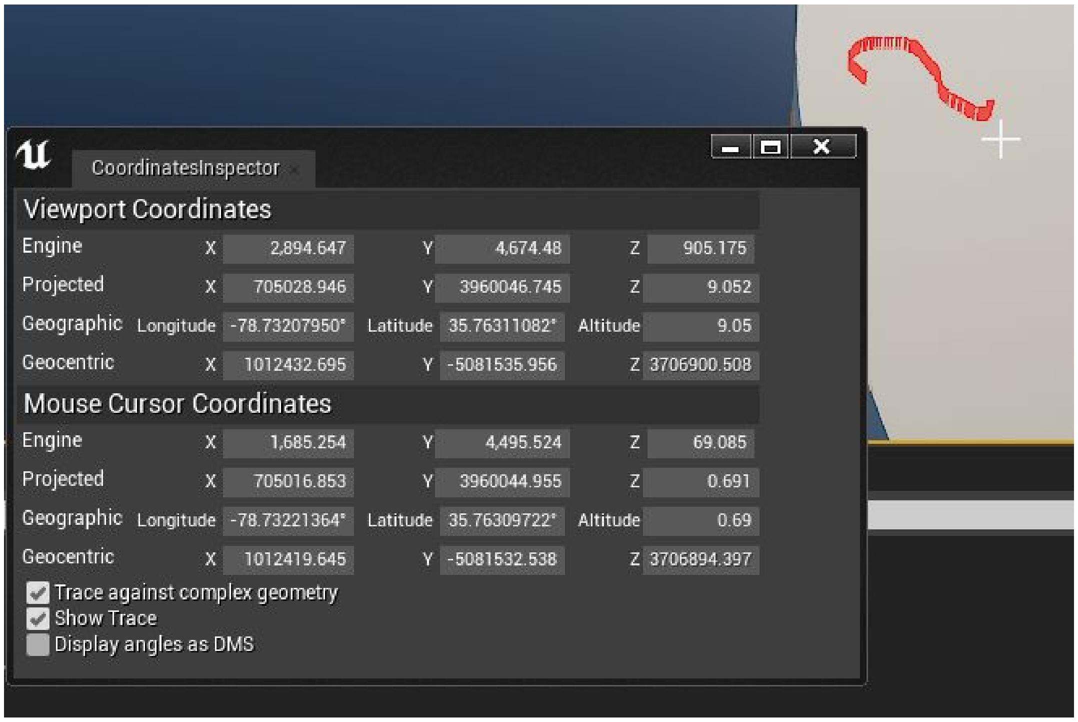
The utility is located under the Simulation plugin content. Be sure to have enabled “show plugin content first”.



UW_Coordinates (Editor Utility Widget)

Path: /SimulationPlugin
Cooking Filepath Length: 161 / 260
Parent Class: Blutility.EditorUtilityWidget
Native Parent Class: Blutility.EditorUtilityWidget
Blueprint Type: Normal
Is Data Only: False
Num Replicated Properties: 0
Tick Frequency: Auto
Force Slow Construction Path: False
Support Dynamic Creation: Default
Inclusive Widgets: 41
Estimated Template Size: 44,688
Tick Prediction: Will Tick
Tick Prediction Reason: Script|Native
Property Bindings: 0

Then right click and Run the utility widget.



Viewport Coordinates

Engine	X	2,894.647	Y	4,674.48	Z	905.175
Projected	X	705028.946	Y	3960046.745	Z	9.052
Geographic	Longitude	-78.73207950°	Latitude	35.76311082°	Altitude	9.05
Geocentric	X	1012432.695	Y	-5081535.956	Z	3706900.508

Mouse Cursor Coordinates

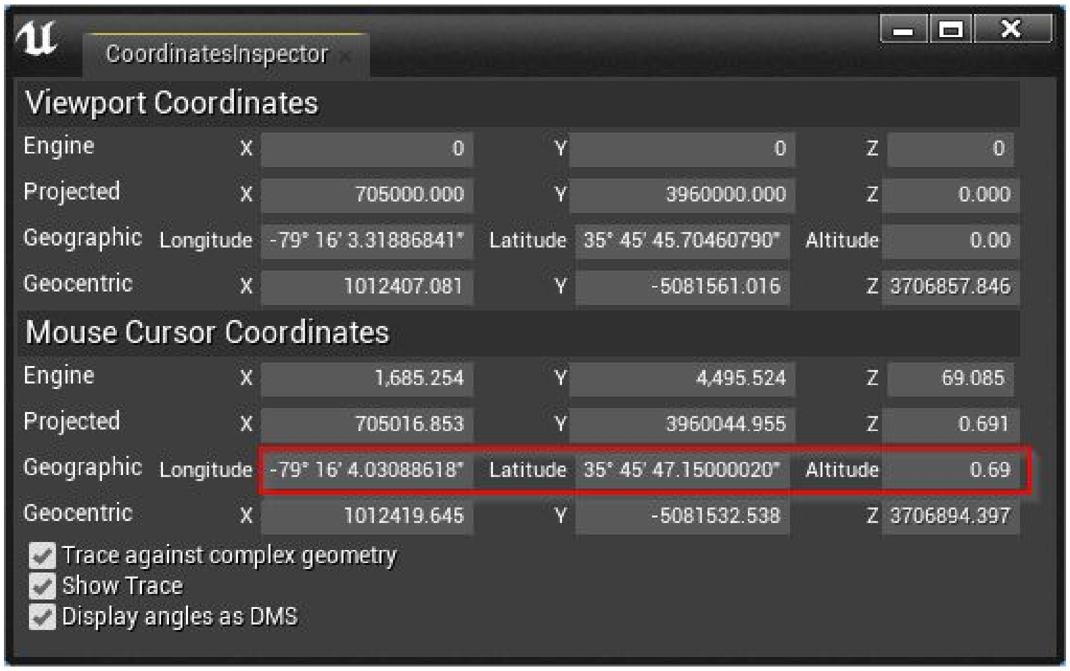
Engine	X	1,685.254	Y	4,495.524	Z	69.085
Projected	X	705016.853	Y	3960044.955	Z	0.691
Geographic	Longitude	-78.73221364°	Latitude	35.76309722°	Altitude	0.69
Geocentric	X	1012419.645	Y	-5081532.538	Z	3706894.397

- ✓ Trace against complex geometry
- ✓ Show Trace
- ☐ Display angles as DMS

For each of these 2 points, coordinates in each CRS are computed.

The user can opt for

- Raycasting against Simple or Complex geometries
- Displaying Trace debug lines
- Displaying Lat/Long in Decimal or Degree Minutes Seconds format.



Viewport Coordinates

Engine	X	0	Y	0	Z	0
Projected	X	705000.000	Y	3960000.000	Z	0.000
Geographic	Longitude	-79° 16' 3.31886841"	Latitude	35° 45' 45.70460790"	Altitude	0.00
Geocentric	X	1012407.081	Y	-5081561.016	Z	3706857.846

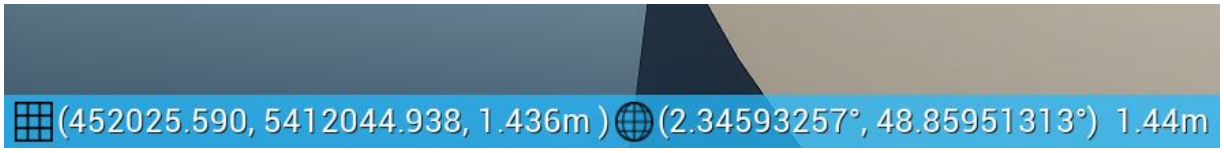
Mouse Cursor Coordinates

Engine	X	1,685.254	Y	4,495.524	Z	69.085
Projected	X	705016.853	Y	3960044.955	Z	0.691
Geographic	Longitude	-79° 16' 4.03088618"	Latitude	35° 45' 47.15000020"	Altitude	0.69
Geocentric	X	1012419.645	Y	-5081532.538	Z	3706894.397

- ✓ Trace against complex geometry
- ✓ Show Trace
- ✓ Display angles as DMS

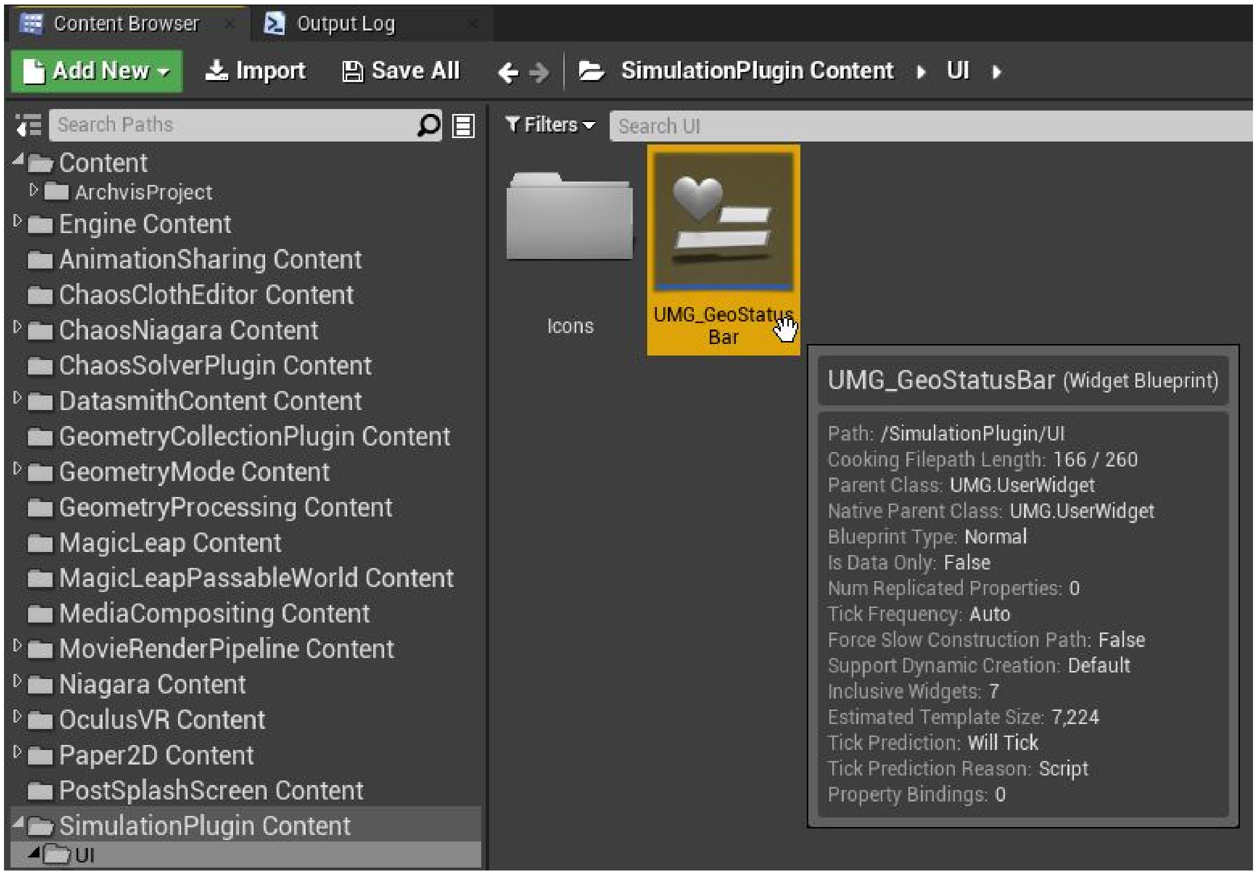
Runtime Usage

Example of an UMG Widget displaying player location in UTM or WGS84 in a status bar



(452025.590, 5412044.938, 1.436m) (2.34593257°, 48.85951313°) 1.44m

The utility is located under the UI folder of Simulation plugin content. Be sure to have enabled “show plugin content first”.



UMG_GeoStatusBar (Widget Blueprint)

Path: /SimulationPlugin/UI
Cooking Filepath Length: 166 / 260
Parent Class: UMG.UserWidget
Native Parent Class: UMG.UserWidget
Blueprint Type: Normal
Is Data Only: False
Num Replicated Properties: 0
Tick Frequency: Auto
Force Slow Construction Path: False
Support Dynamic Creation: Default
Inclusive Widgets: 7
Estimated Template Size: 7,224
Tick Prediction: Will Tick
Tick Prediction Reason: Script
Property Bindings: 0

Add it to your viewport like any UMG widget.