Graph Data Structure v1.0

Last updated 2023-01-22

Contents

1. Classes provided

The project currently consists of 1 class and 2 structures:

- The main graph class.
- The node structure - a wrapper around a UObject.
- The edge structure - a wrapper around 2 nodes.

2. Creating a graph object

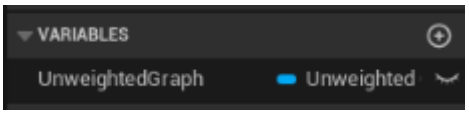
To create a graph you can construct an object in Blueprints you can use the Construct Object from Class node with the graph class and save it to a variable.

To create a graph object in C++ you can initialize it like so:

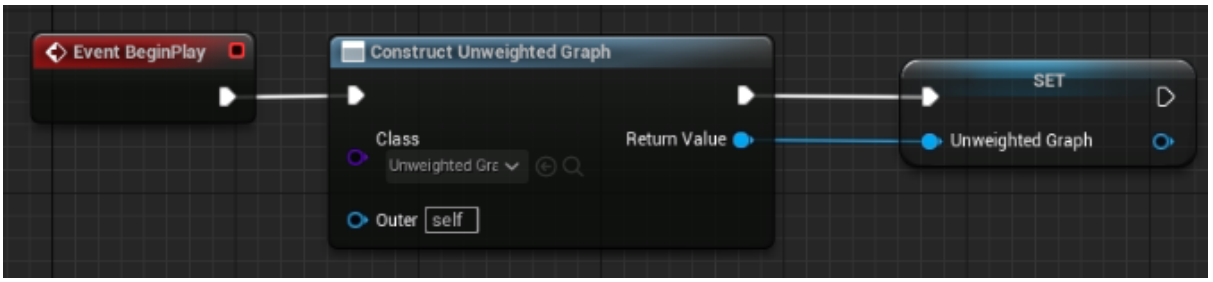
```
UUnweightedGraph* Graph = NewObject<UUnweightedGraph>();
```

Later on you can call the needed functions for it.

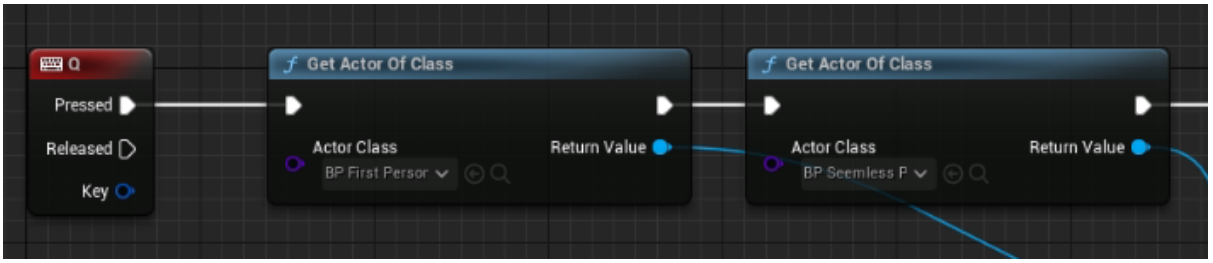
Example of setting up a graph and using it in Blueprints:
First create a graph variable

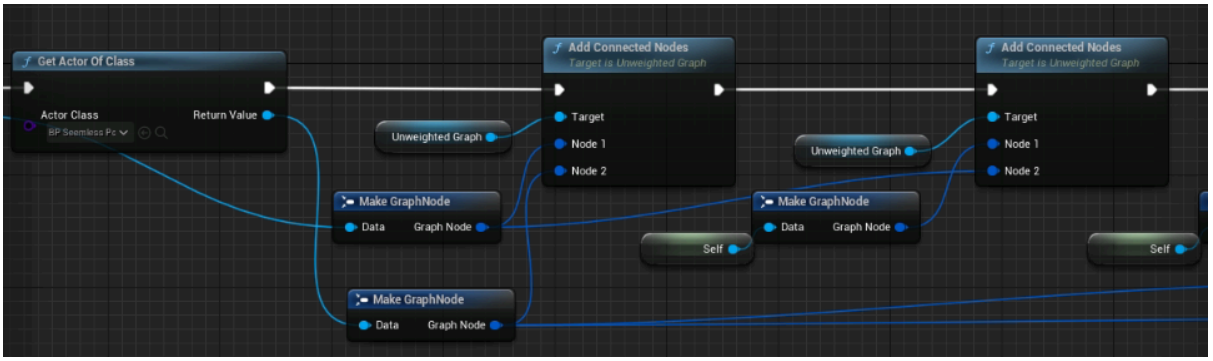


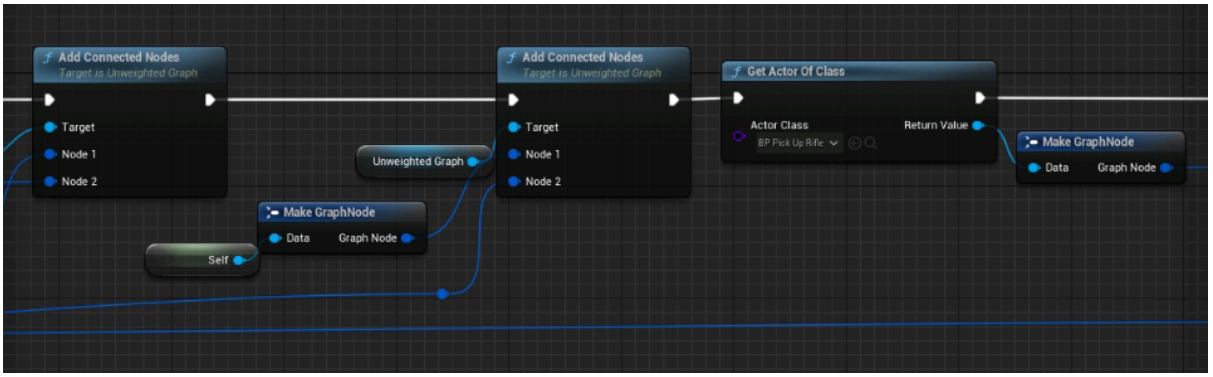
Next you need to initialize it

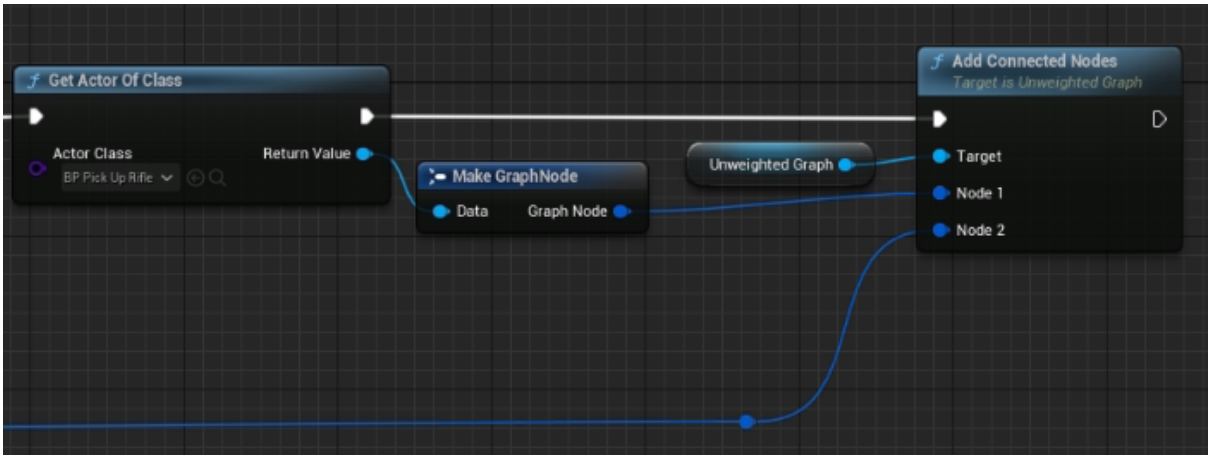


Next let's bind data addition to our Q button and code it out like so:

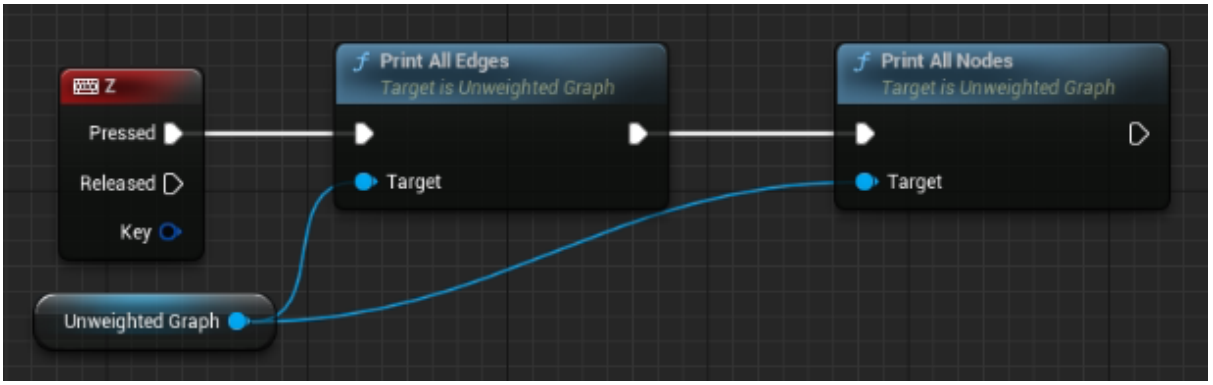








Finally, let's call some functionality on the graph



The result:

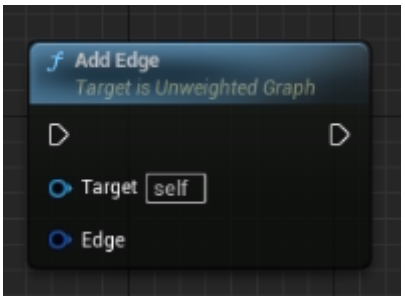
```
LogZablasGraph: --- Print all edges ---
LogZablasGraph: Node1: BP_FirstPersonCharacter_C_0 <-> Node2: BP_SeemlessPortalsGameMode_C_0
LogZablasGraph: Node1: BP_PortalPlayerController_C_0 <-> Node2: BP_FirstPersonCharacter_C_0
LogZablasGraph: Node1: BP_PortalPlayerController_C_0 <-> Node2: BP_SeemlessPortalsGameMode_C_0
LogZablasGraph: Node1: BP_Rifle_C_UAID_B42E9936F5422D3B01_1556789383 <-> Node2: BP_SeemlessPortalsGameMode_C_0
LogZablasGraph: --- Edge printing finished ---
LogZablasGraph: --- Print all nodes ---
LogZablasGraph: Node: BP_FirstPersonCharacter_C_0
LogZablasGraph: Node: BP_SeemlessPortalsGameMode_C_0
LogZablasGraph: Node: BP_PortalPlayerController_C_0
LogZablasGraph: Node: BP_Rifle_C_UAID_B42E9936F5422D3B01_1556789383
LogZablasGraph: --- Node printing finished ---
```

3. Provided functions

3.1. Add Edge

Adds a new edge to the graph if it does not already exist.

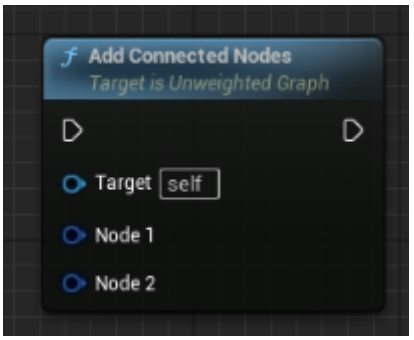
Parameters:
Edge - The new edge to be added.



3.2. Add Connected Nodes

Adds 2 new nodes connected to each other and the graph.

Parameters:
Node1 - The first node to be added.
Node2 - The second node to be added.



3.3. Edge Count

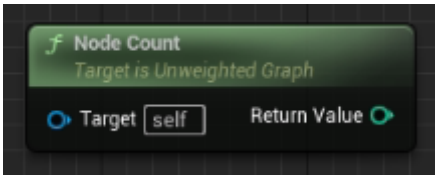
Gets the total amount of edges in the graph.

Returns: The amount of edges in the graph.

3.4. Node Count

Gets the total number of nodes in the graph.

Returns: The amount of nodes in the graph.



3.5. Get Edges

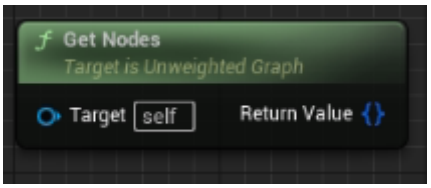
Gets all of the edges in the graph.

Returns: All of the edges.

3.6. Get Nodes

Gets all of the nodes in the graph.

Returns: All of the nodes.

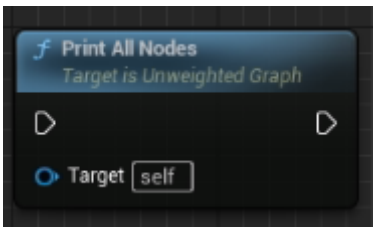


3.7. Print All Edges

Prints all of the edges to the output log.

3.8. Print All Nodes

Prints all of the nodes to the output log.

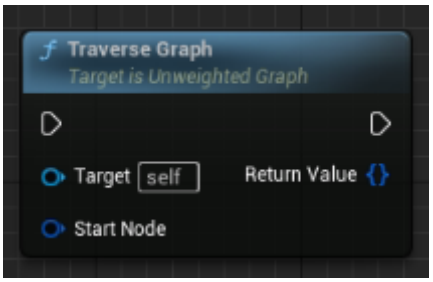


3.9. Traverse Graph

Traverses the graph in a Depth-First manner from the given node.

Parameters:
StartNode - The node from which to start the traversal.

Returns: The path that the traversal used.

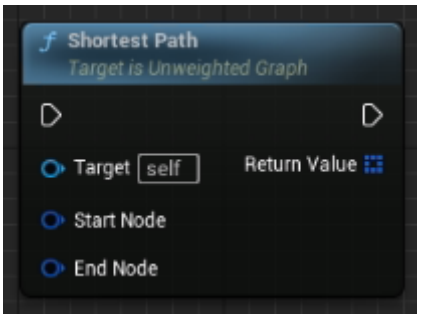


3.10. Shortest Path

Finds the shortest path between the given nodes.

Parameters:
StartNode - The node from which to start the search.
EndNode - The node at the end of the path.

Returns: The shortest path between the two nodes.



3.11. Cycle Count

Finds the amount of cycles of the given length.

Parameters:

DesiredLengthOfCycle - The desired length of a single cycle.

Returns: The amount of cycles found.

3.12. Calculate Diameter

Calculates the diameter of the graph.

Returns: The diameter.

3.13. Get Eccentricity

Calculates the eccentricity for the given node.

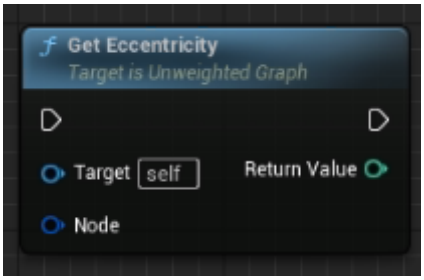
Parameters:
Node - The node for which to count the eccentricity.

Returns: The node's eccentricity.

3.14. Calculate Radius

Calculates the radius of the graph.

Returns: The radius.



3.15. Calculate Center

Calculates the center nodes of the graph.

Returns: A set of center nodes.

3.16. Calculate Girth

Calculates the girth of the graph.

Returns: The girth.