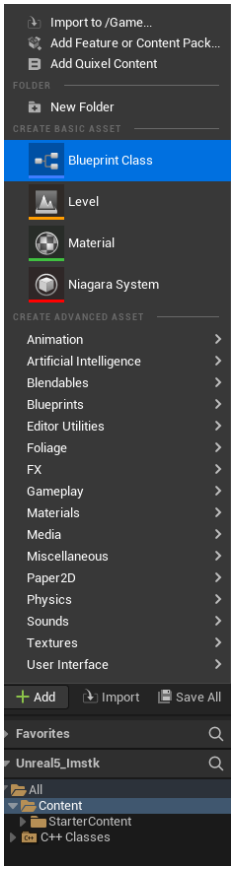
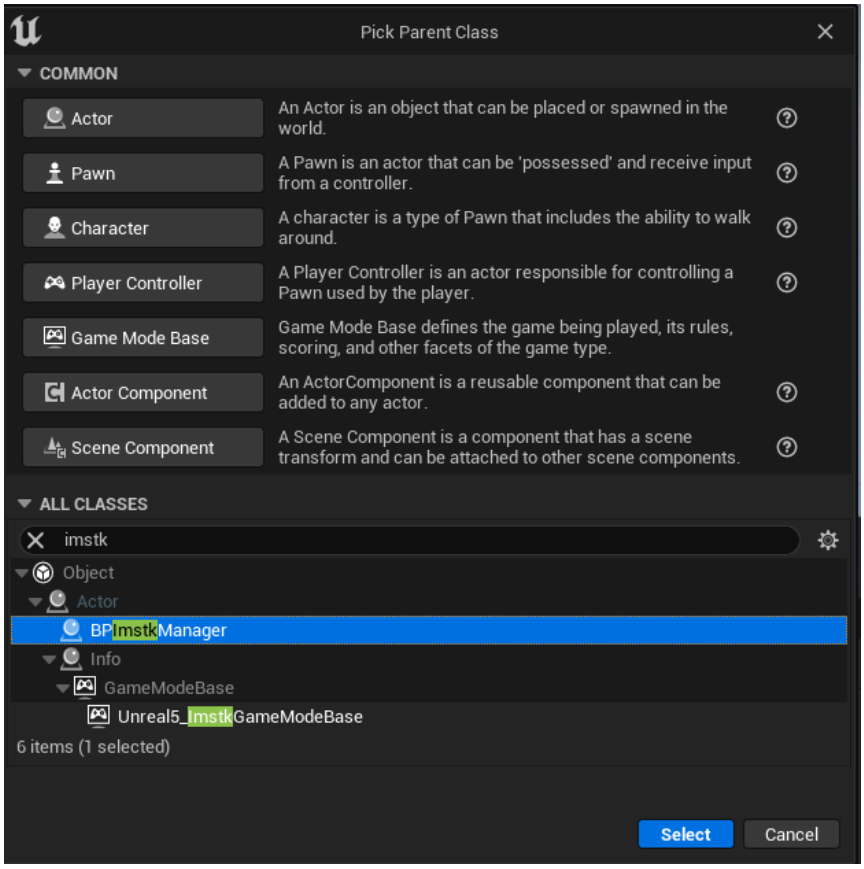
iMSTK Unreal Engine Plugin

Getting Started

1. Create a folder called Plugins if it doesn't already exist in the directory of your .uproject file
2. Extract Imstk.zip into the Plugins folder and launch unreal
3. For each map where iMSTK will be used, create a blueprint that is a child of "BPImstkManager"





4. Drag the blueprint into the map

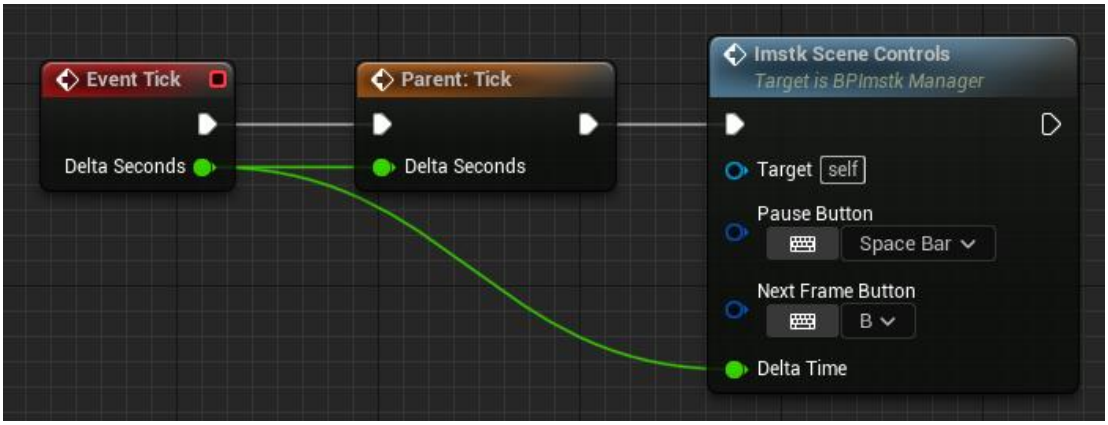
5. Open the blueprint and add the node FinalizeImstk after Parent:BeginPlay

iMSTK Scene Settings

1. Gravity, time step and other values for the iMSTK scene can be set from the manager blueprint
2. Scene Scale will reduce the size of the scene in iMSTK by that degree (default is 100 since one Unreal unit is 1cm and one iMSTK unit is 1m)

Adding Controls to iMSTK

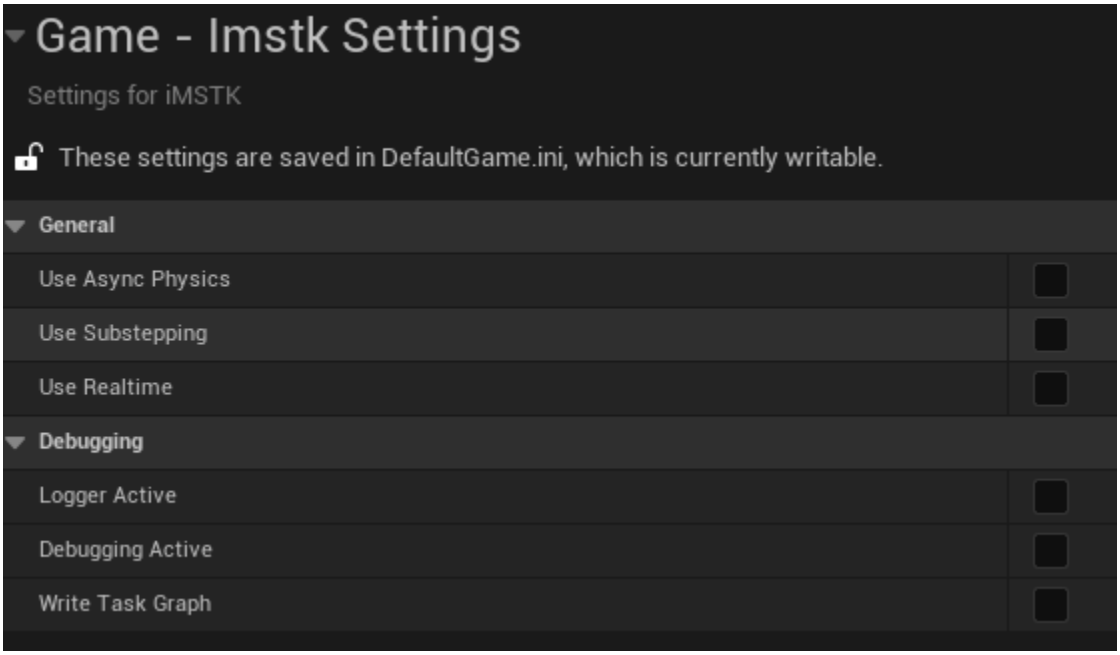
1. Open the IMSTK manager blueprint
2. Add the node “ImstkSceneControls” after Parent:Tick



3. Assign the buttons and delta time values

iMSTK Settings

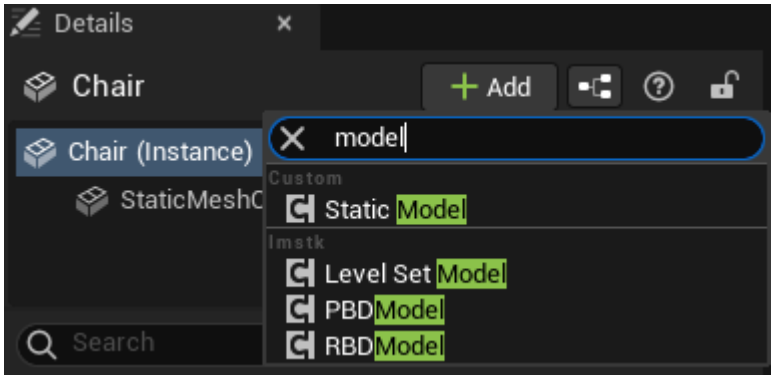
1. Settings for IMSTK can be found under Project Settings -> Game-> iMSTK Settings



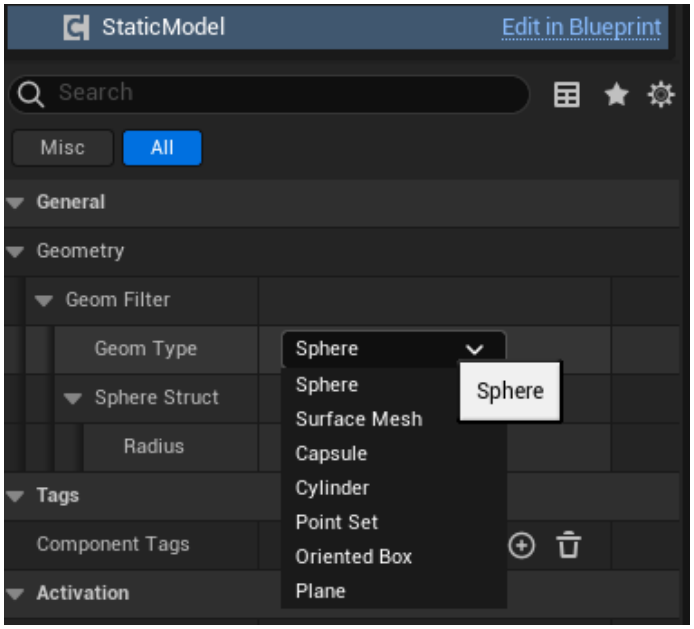
Adding Models to iMSTK

Static/RBD Models

1. Add the relevant model component to the actor



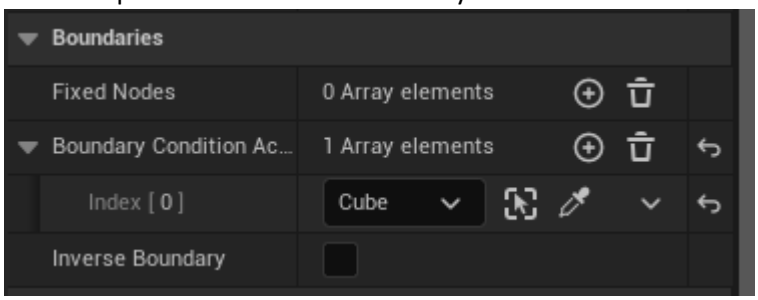
2. Under Geom Filter choose the type of geometry the object will have in iMSTK



Note: Surface Mesh and Point Set require a static mesh component to be attached to the actor. The other geometry types are generated in iMSTK without visuals in Unreal

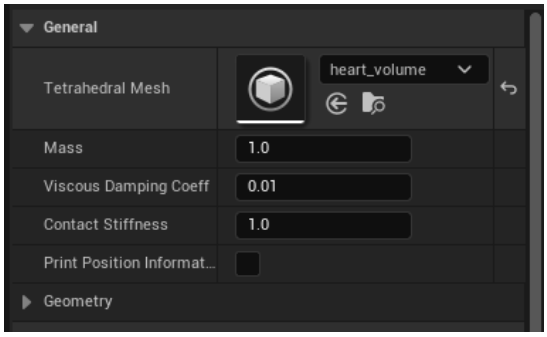
PBD Models

1. See steps for static and RBD models
2. Vertices can be explicitly stated for fixing on the PBDModel or an actor with a static mesh component can act as a boundary condition to fix all vertices within it



3. For tetrahedral meshes, import the tetrahedral model into Unreal. This will generate a Tetrahedral mesh asset and a static mesh of the model.

4. Assign the tetrahedral mesh on the PBDModel component and the static mesh on the static mesh component (The static mesh does **not** need to be the generated mesh from importing the tetrahedral model)



5. The relevant constraints can be assigned on the PBDModel

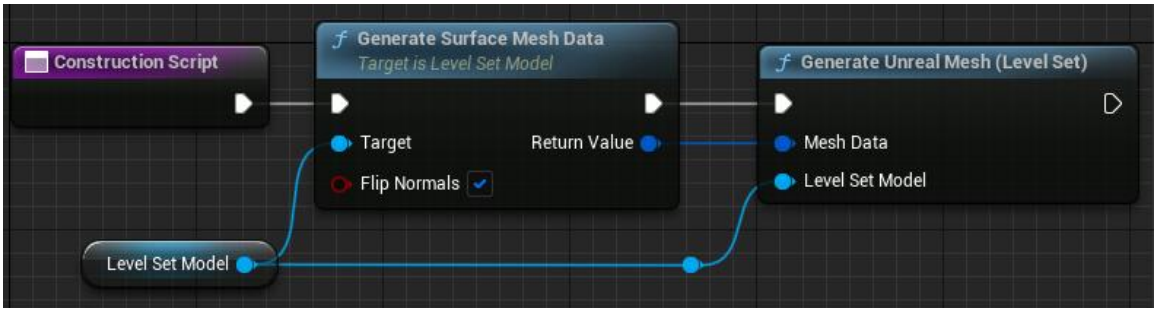
Note: The geometry on PBDModels must be set to surface meshes or point sets

LevelSet Models

From Image Data File

1. Attach a LevelSetModel component to the actor
2. Import the ImageData file into Unreal (currently only .nii supported)
3. Assign the generated ImageData onto the LevelSetModel

4. Attach a procedural mesh component to the actor
5. For editor visuals, convert the actor into a blueprint and create and attach *Generate Surface Mesh Data* and *Generate Unreal Mesh* to the construction script in the blueprint



From Static Mesh

1. Attach a LevelSetModel component to the actor
2. Set the dimensions for the model
3. Create and set a bounding actor that surrounds the model
4. Set the number of chunks
5. Set the material of the LevelSetModel (Does not inherit from the static mesh currently and only one material can be set)
6. Attach a procedural mesh component to the actor

Note: The geometry must be set to surface mesh for LevelSet Models

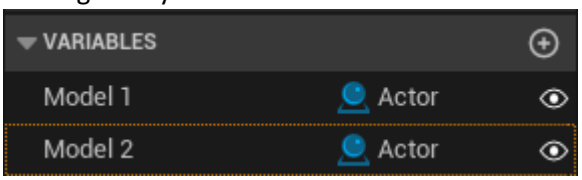
PBD Threads

1. Create an empty actor
2. Attach a spline component to the actor
3. Attach the PBDThread component to the actor

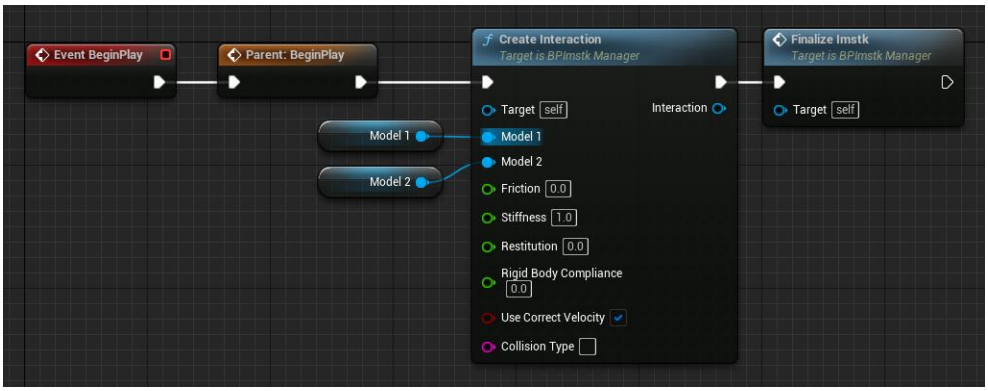
Spline meshes can be used to visualize the thread in game. A blueprint (PBDThreadBP) is included with the plugin that can be used as a template.

Adding iMSTK Interactions

1. Open the iMSTK manager blueprint
2. Create variables of type Actor for each model interacting and set the variable to public by clicking the eye

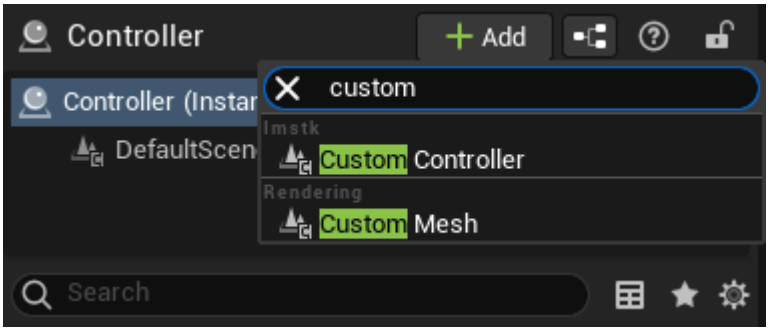


3. Assign the models on the Manager to the relevant Actors
4. Add “Create Interaction” nodes between Parent:BeginPlay and Finalize iMSTK nodes and assign parameters. Note: If “Collision Type” is not set, the most appropriate type will be chosen automatically



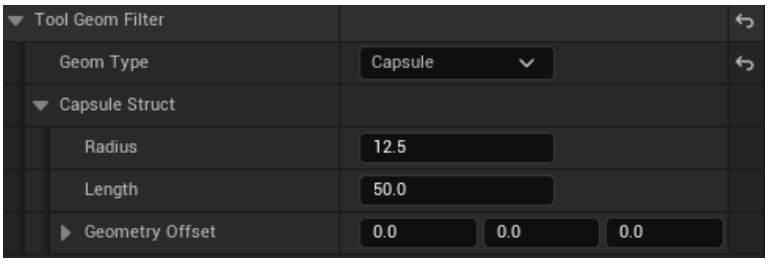
Adding iMSTK Controllers

1. Attach the CustomController component to the actor

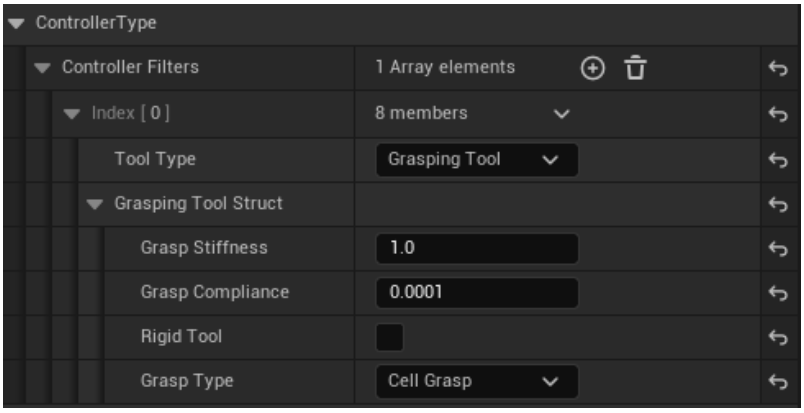


2. Set the chosen preset or choose controller type and geometry manually.

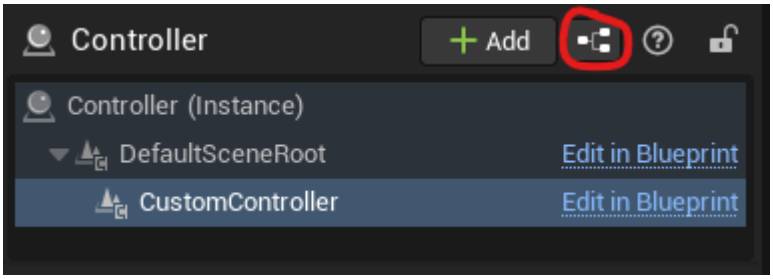
If choosing manually, select the wanted geometry type,



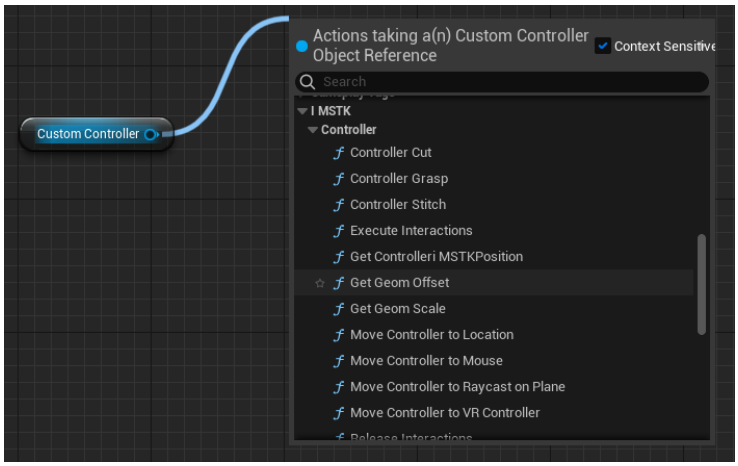
add a controller type to the controller filters and set the desired parameters



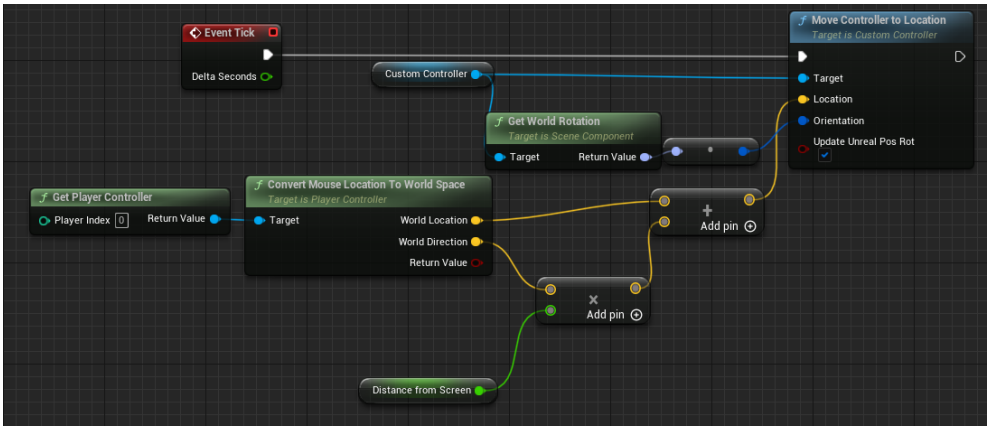
3. Create a blueprint from the actor



4. The controller functions can then be accessed from the blueprint



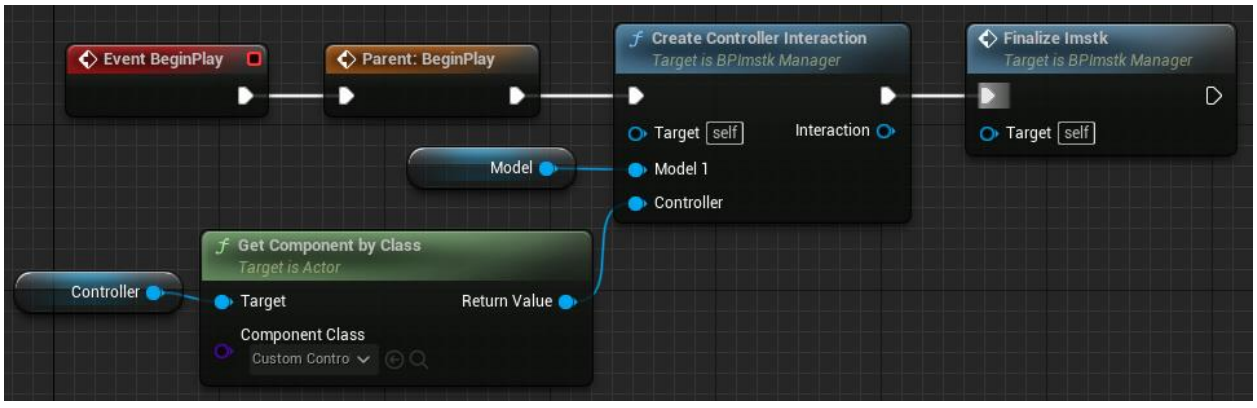
Example: Code to move and update the controller's position to the location of the mouse in iMSTK and Unreal



Note: If creating a surface mesh tool, a static mesh component must be attached as a child of the custom controller component

Adding Controller Interactions

1. Open the iMSTK manager blueprint
2. Add variables of type Actor for the controller and the object you would like the controller to interact with
3. Add "CreateControllerInteraction" and assign the model and controller

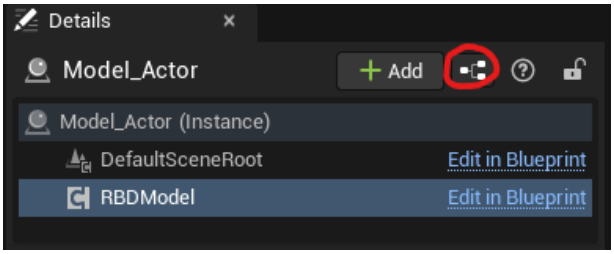


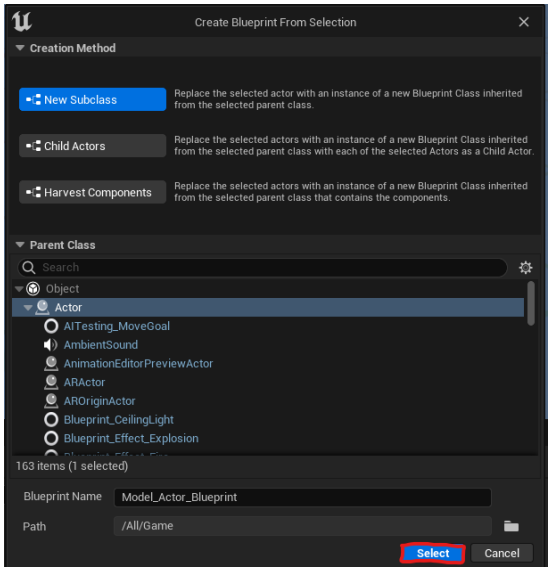
Note: For many controller interactions involving PBDModels "Shared Model" must be enabled in the PBDModel

Adding Unreal Visuals for iMSTK Objects

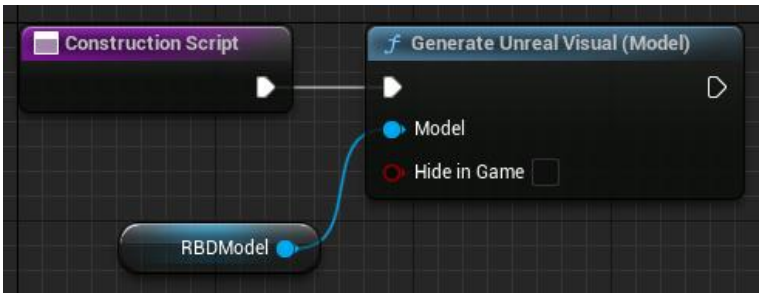
Note: For visualizing spheres, capsules, cylinders and line meshes

1. Create a blueprint from the actor with the model component attached





2. Add and attach *Generate Unreal Visual* (for a wire mesh) or *Generate Unreal Mesh* (for a solid mesh) to the construction script of the blueprint

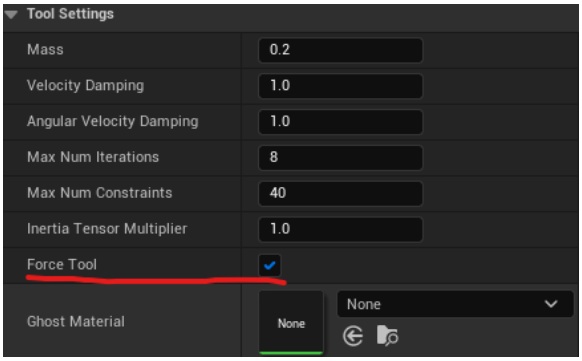


3. Click compile and the visuals for the model will update in the Viewport

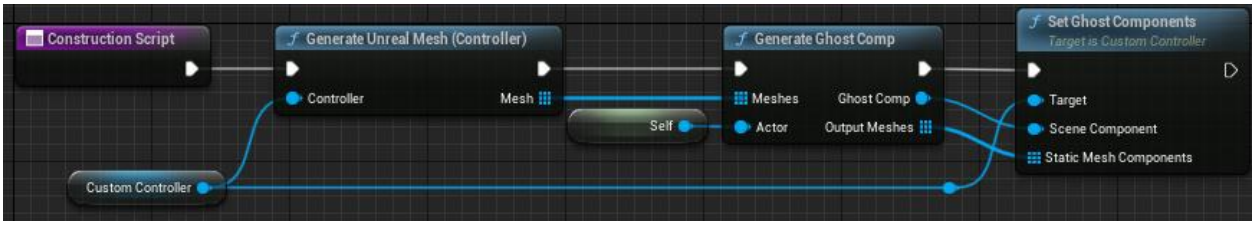
Note: Changes to the properties of the model must be done within the blueprint for the visuals to update

Ghost Visuals for Controllers

1. Set the controller to a force tool



2. The material for the ghost tool can be set after enabling force tool
3. Create and attach *Generate Ghost Comp* and *Set Ghost Components* after *Generate Unreal Mesh* on the construction script of the controller blueprint

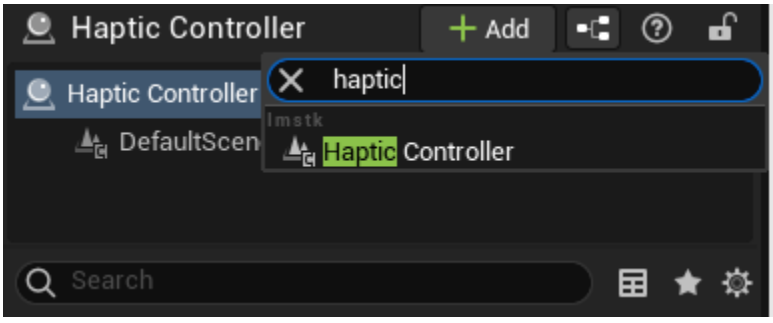


Note: The rotation of the ghost model will not be reflected in the editor

Haptic Device Controllers

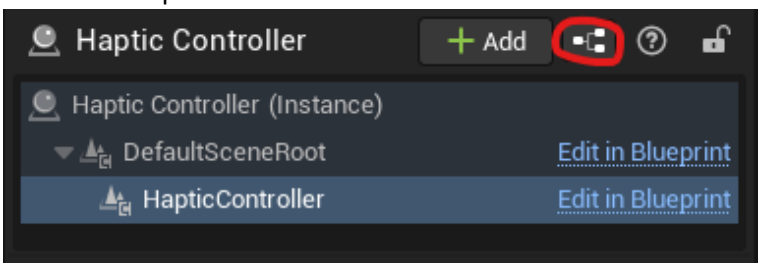
(Currently only Open Haptics supported)

1. Add a HapticController component to the actor

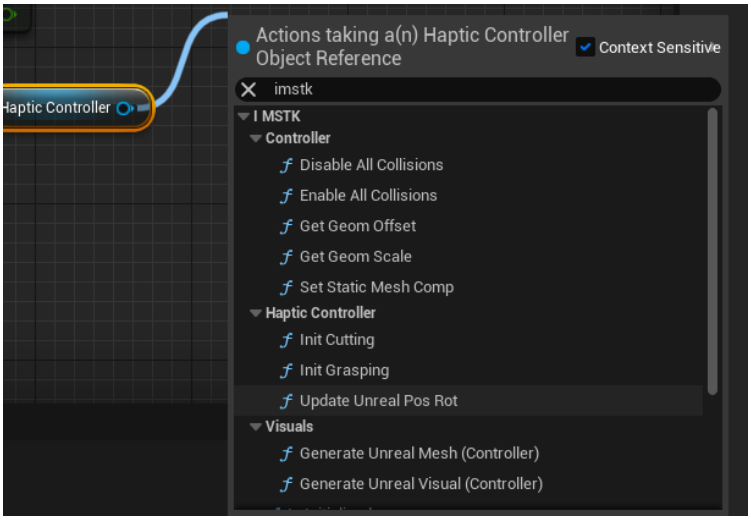


2. Set the relevant variables on the controller

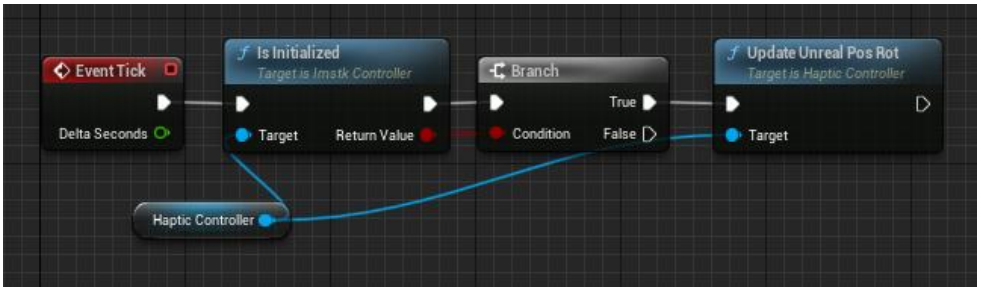
3. Create a blueprint from the actor



4. A list of relevant functions can be found in the blueprint

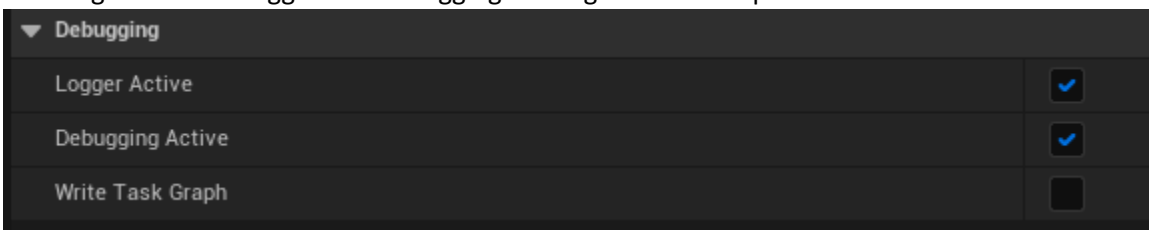


Example: To make the controller update it's position within unreal



Troubleshooting

- Enabling the iMSTK Logger and debugging messages can be helpful



- The log can be found in the …\Saved\Logs folder within your project

Known Issues

- Certain collision types (primarily deformable to capsule/sphere) can cause issues if “use correct velocity” is set to true on the PBDModel
- Ghost components are currently not functional with haptic controllers
- Two PBDModels with FEM constraints will cause one of the models to lose it’s constraints
- Rotation does not apply to LevelSetModels generated from image data
- Task graph visual generation is currently disabled