3D SYSTEMS

Haptics Direct

Unity Plugin Version 1.0

User Guide

Table of Contents

Installation the Plugin 7
Prerequisite 7
Configuring the Devices (Creating Profiles) 9
Plugin Installation 11
Understanding the components of the Plugin 14
FeaturesScenes 14
HapticCollisions Scene 15
HapticMaterialBasic Scene 15
HapticMaterialCombination Scene 16
HapticPainting Scene 16
HapticPlugin 16
HapticScripts 17
Plugin Scripts 17
Demo Scripts 17
Prefabs 18
Resources 18
The Haptic Scene 19
The Haptic Actor 20
What is the Haptic Actor? 20
How do you create a Haptic Actor? 20
Add Haptic Actor using Prefabs 20
Add Haptic Actor using Menu 21
Add Haptic Actor using Script 22
Overview Properties Sections 23
Device Setup 23
On Screen Stylus 23
Device Information 24
Simple Buttons Setup 25
Events 25
Camera and Navigation 26
Global Vibration Settings 28

Global Spring Settings 28
Global Constant Force Settings 29
Haptic Collider 30
How do you create a Haptic Collider? 31
Create a Haptic Collider by using the Menu 31
Create a Haptic Collider by adding the Script 32
Haptic Collider (component) settings 33
Haptic Material 34
How do you create a haptic material? 34
Add a Haptic Material component by Menu 34
Add a Haptic Material component by Script 35
Haptic Material settings 35
Gizmos 38
Workspace Gizmo 38
Virtual Stylus Gizmo 39
Virtual Haptic Gizmo 40
API Haptics Direct – Scripts 41
Haptic Plugin 41
getVersionString(dest,len) 41
initDevice(deviceName) 41
getDeviceSN(configName, dest,len) 41
getDeviceModel(configName, dest, len) 41
getDeviceMaxValues(configName, max_stiffness, max_damping, max_force) 41
startSchedulers() 41
getWorkspaceArea(configName, usable6, max6) 42
getPosition(configName, position3) 42
getVelocity(configName, velocity3) 42
getTransform(configName, matrix16) 42
getButtons(configName, buttons4, last_buttons4, inkwell) 42
getCurrentForce(configName, currentforce3) 42
getJointAngles(configName, jointAngles, gimbalAngles) 43
setForce(configName, lateral3, torque3) 43
setAnchorPosition(configName, position3) 43

addContactPointInfo(configName, Location, Normal, MatStiffness, MatDamping,..... 43
updateContactPointInfo(configName) 44
resetContactPointInfo(configName)..... 44
setVibrationValues(configName, direction3, magnitude, frequency, time) 44
setSpringValues(configName, anchor, magnitude) 44
setConstantForceValues(configName, direction, magnitude)..... 44
setGravityForce(configName, gForce3) 44
disconnectAllDevices()..... 45
int getHDError(Info, len); 45
CameraUpdate() 45
AngleToNeg(angle) 45
AngleToPos(angle) 45
isInRotRange() 45
isInTransRange() 45
isInZone(value, zone)..... 46
UpdateWorkspaceTransform() 46
UpdateDZZero()..... 46
InitializeHapticDevice()..... 46
DisconnectHapticDevice() 46
GetDeviceTransformationRaw()..... 46
UpdateDeviceInformation()..... 46
UpdateTransfrom() 46
UpdateButtonStatus()..... 47
EnableVibration()..... 47
DisableVibration()..... 47
EnableSpring() 47
DisableSpring() 47
EnableConstantForce() 47
DisableContantForce()..... 47
EnableNavTranslation()..... 47
DisableNavTranslation()..... 48
SwitchNavTranslation() 48
EnableNavRotation ()..... 48

DisableNavRotation () 48
SwitchNavRotation () 48
isNavRotation() 48
isNavTranslation() 48
Grab_Object() 48
Release_Object() 49
UpdateCollision(collision, isCollisionStart, isCollision, isCollisionExit) 49
UpdateForceOnCollision(collision) 49
SendContactpoints() 49
GrabObj(collision) 49
ReleaseObj() 49

Introduction

The purpose of the Haptics Direct Unity plugin is to enable users to seamlessly add Haptics (the ability of touch) to their virtual world. This plugin integrates the Unity 3D components with the haptic world there by creating a wholesome experience for the user. This plugin works in harmony with the Touch and Touch X devices and is meant to fill a void in the 3D and VR space by allowing for the perception and manipulation of objects using the senses of touch and proprioception.
It is imperative that users and/or programmers have at least an intermediate knowledge of Unity and Haptics Direct to derive the maximum benefit from this plugin.

Copyright

©1993-2021. 3D Systems, Inc. All rights reserved. The content of this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by 3D Systems, Inc. Any names, places, and/or events in this publication are not intended to correspond or relate in any way to individuals, groups or associations. Any similarity or likeness of the names, places, and/or events in this publication to those of any individual, living or dead, place, event, or that of any group or association is purely coincidental and unintentional.

Installation the Plugin

Prerequisite

In order to install the new Haptics Direct Unity Plugin, the following steps must be taken:

1. Download the latest Touch device drivers and configure the Haptic Devices to be used with the plugin (Use the Touch Smart setup or Touch Setup to configure device profiles).

    a. Access the download link for latest drivers: 2020.7.9: https://support.3dsystems.com/s/article/Haptic-Device-Drivers?language=en_US

    b. Click on “Downloads”, then click on “Haptic Device Drivers”

    c. Scroll down to Product : OpenHaptics

    d. Ensure that the OS is listed as: WIN 10, 7 or 8.1 and Platform : 64 bit (only)

    e. Click on “Download” version : 2020.7.9

OpenHaptics	v3.5	**Touch Device Driver v2020.7.9:** *for Ethernet Touch or Touch X, and USB Touch or Touch X devices* Interface: OS: 8.1/10 (64-bit)	Download
		Phantom Device Driver v5.1.7: *for Firewire or Parallel devices* Interface: OS: 7/8/10 (64 bit)	Download

Figure 1 – Download Driver

2. The user must have at the minimum the Unity software (free version).

    a. Download the Unity plugin from the following location: https://unity3d.com/get-unity/download

b. Unity2019 64 bit required. (Tested with Unity2019.4.28f1 LTS – 64 bit)

c. Operating System: Windows (PC Only) – 64 bit

Once the above pre-requisites are met, the user is ready to install and access the Haptics Direct Unity Plugin.

Configuring the Devices (Creating Profiles)

After downloading and installing the latest Touch Device Drivers, the next step is to set up the Haptic devices. Connect devices as needed by performing the following steps:

1. Open the Touch Smart Setup application from the desktop, or from the following location: C:\Program Files\3DSystems\Touch Device Drivers

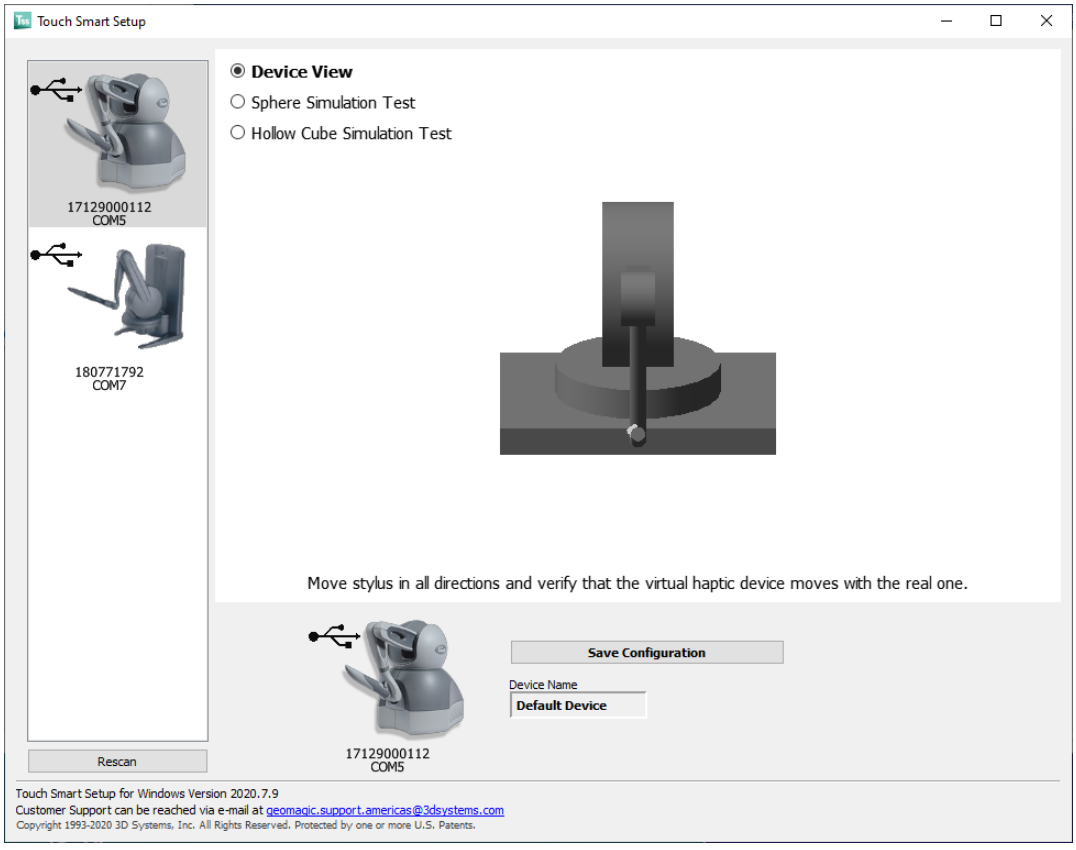
2. Select either Single or Dual device mode as needed

3. Follow the instructions in the application as prompted and set up the device(s) accordingly. For example: if using a LAN device the device will need to be paired prior to operation.

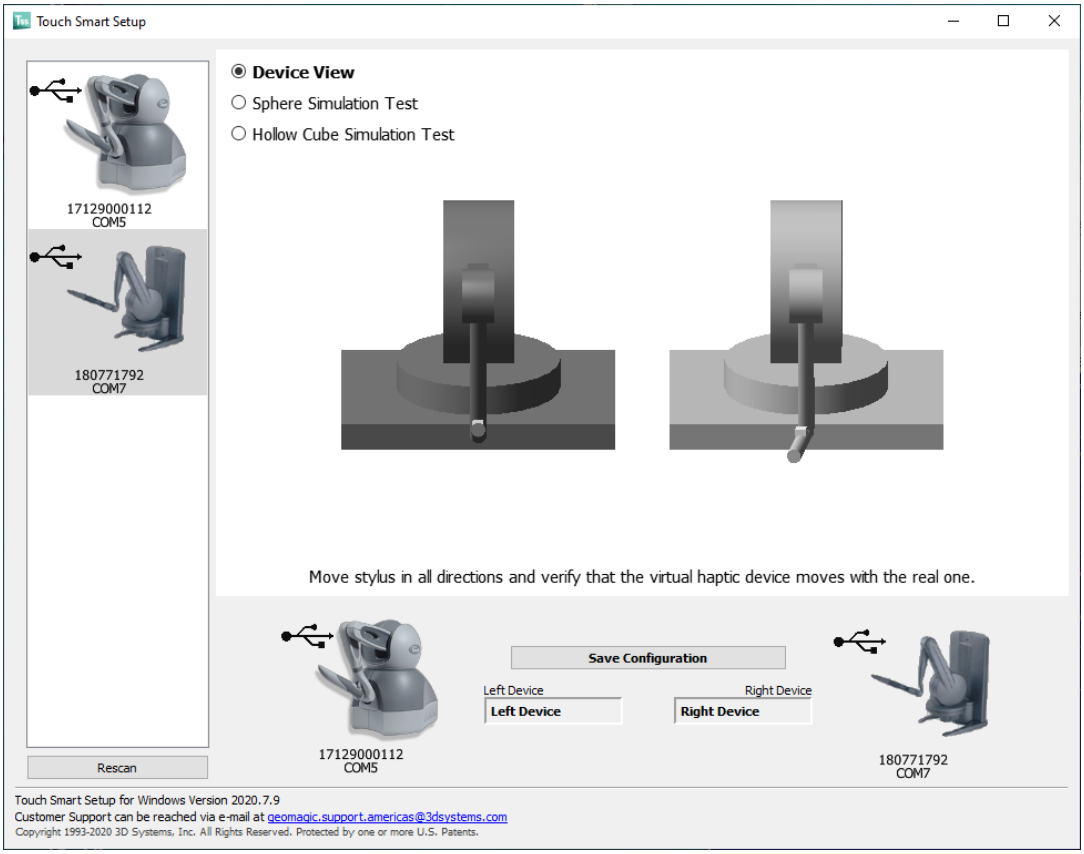
4. Create Configuration (Config) names by setting up and calibrating the devices.

The following profile or configuration names are created as a result:

- Single device: Default Device
- Two devices: Left Device and Right Device



Single Device



Two Devices

Note: Use the Touch Setup application and create custom profile names. The Touch Setup application can be accessed from the following location: C:\Program Files\3D Systems\Touch Device Drivers Select Touch Setup.

Tip: After creating a custom profile using the Touch Setup application, Users must calibrate the devices using the Touch Diagnostic application found in the same location as the Touch Setup application.

Plugin Installation

Perform the following steps to install the Unity plugin:

1. Download the Haptics Direct for Unity packages from the Unity Asset Store
2. Open Unity and open or create a new project
3. Open the **Project Settings** and change in **Player|Other Settings** the **Api Compatibility Level*** from .NET Standard 2.0 to **.NET 4.x**

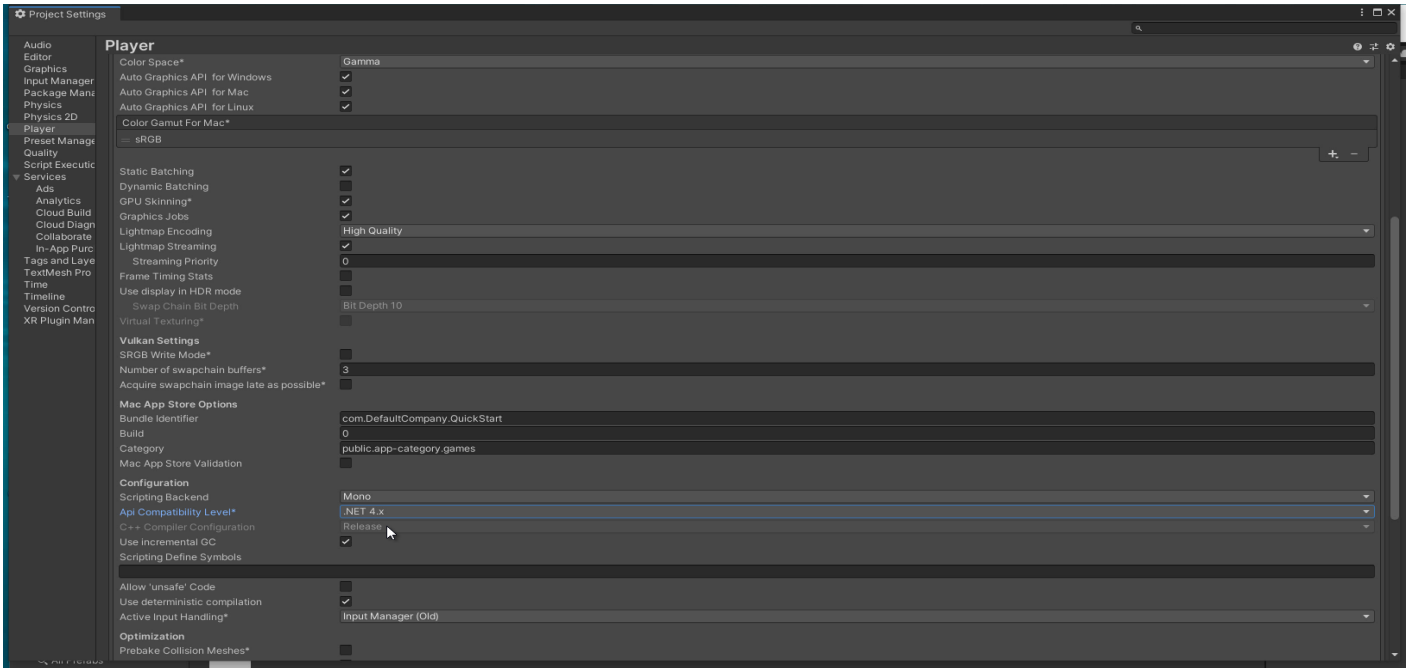


Figure 2 – Setup Api Comaptibility

4. Select **Physics** in the settings tree and check **Enable Adaptive Force**
5. Set **Default Contact Offset** to **0.001**

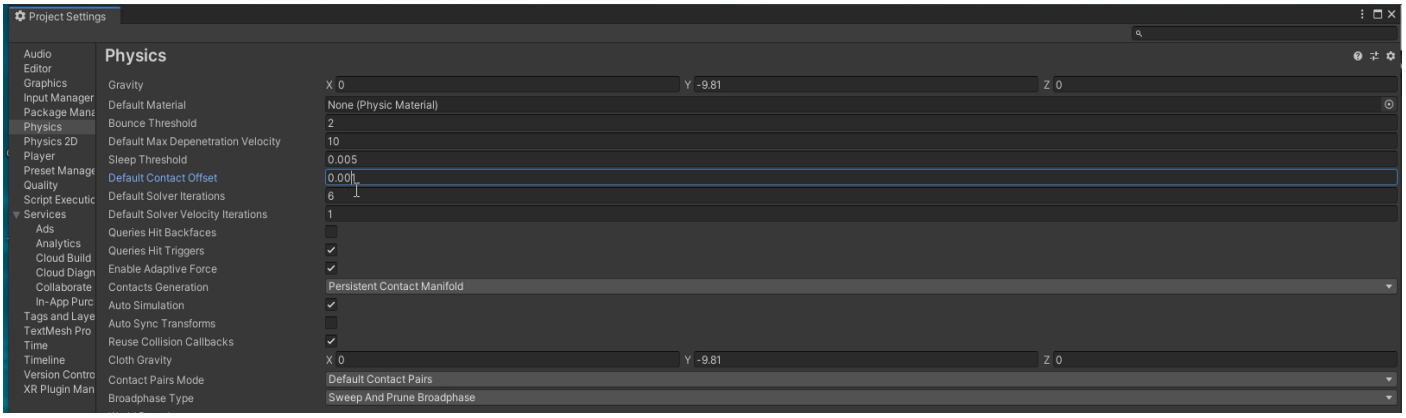


Figure 3 – Change Contact Offset

6. Select **Assets tab->Import Package-> Custom Package** (refer to the image below).

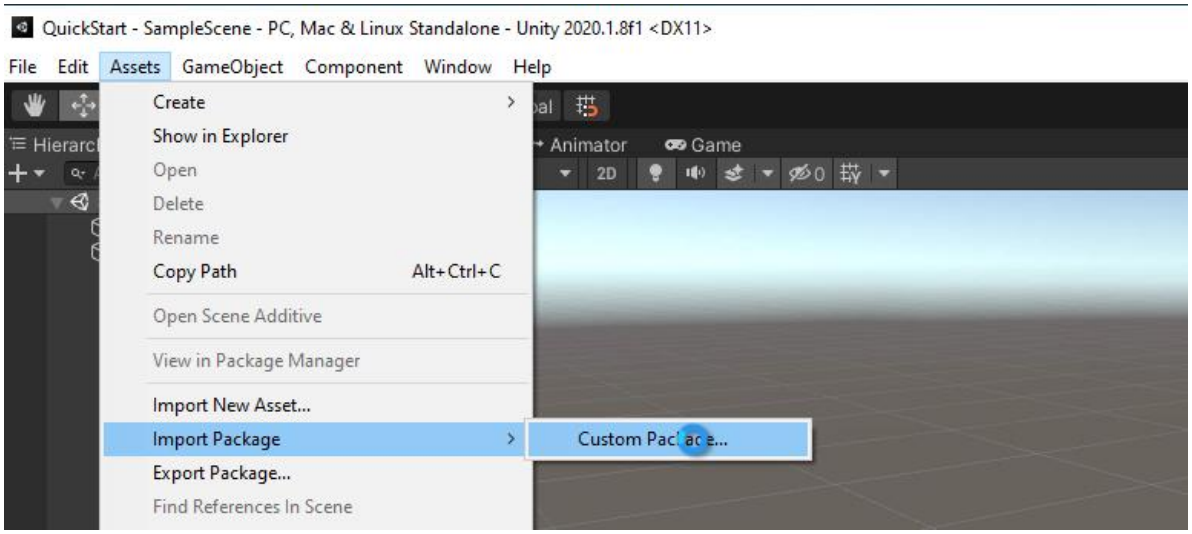


Figure 4 – Import Package

7. Select the asset package **HapticsDirectForUnityV1** from the downloaded location.

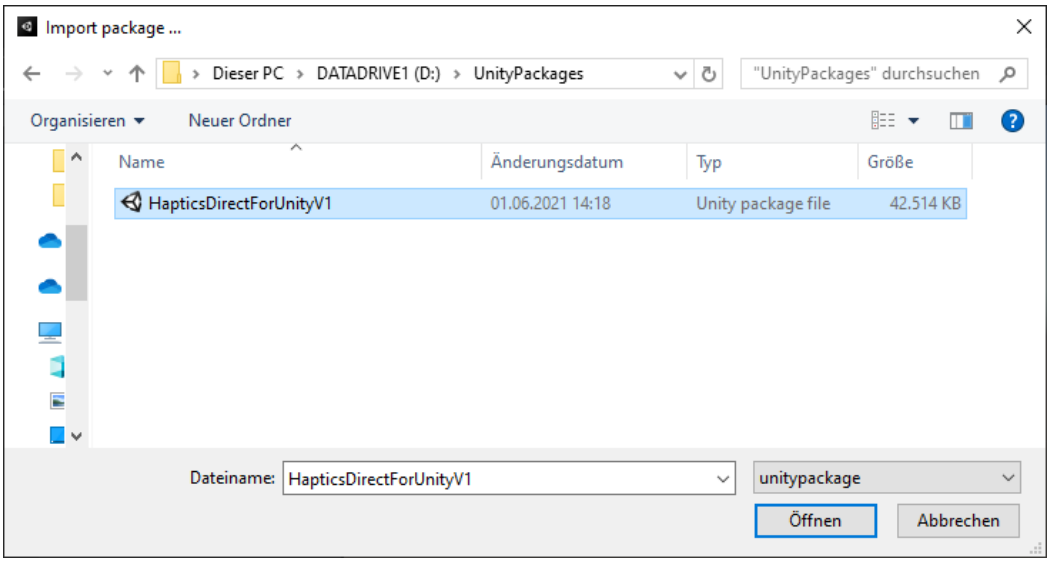


Figure 5 – Select HapticDirectForUnityV1 Package

8. Click the option for **Open**. All components of the package are listed for Import.

Figure 6 – Import Package

9. Click **All**, and then click **Import**. All the components of the plugin are imported and the components are displayed.

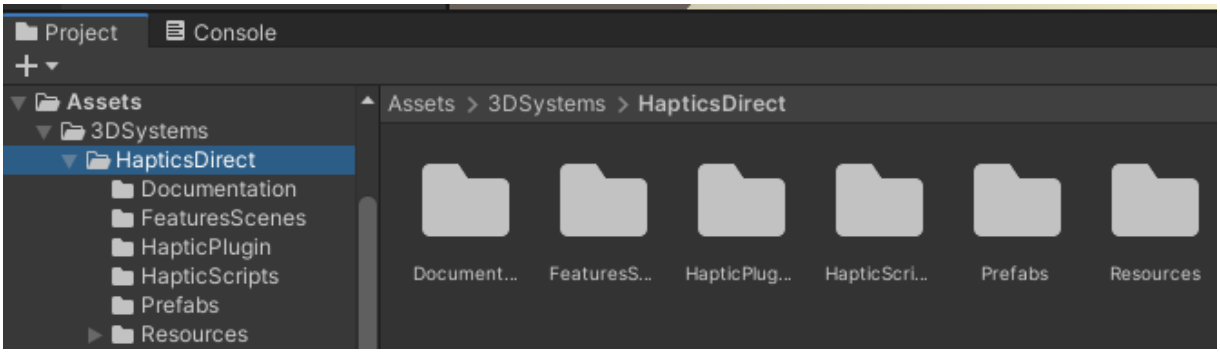


Figure 7 – Final Import

Understanding the components of the Plugin

There are five major components of the HapticsDirect Unity plugin:

- FeaturesScenes
- Documentation
- HapticPlugin
- HapticScripts
- Prefabs
- Resources

These are outlined below.

FeaturesScenes

The FeaturesScenes folder contains four feature demos that were developed using the plugin.

When loading one of the demo scenes for the first time, the import window for the TextMeshPro package will be displayed. Confirm the import by clicking **Import TMP Essentials**. After the import, reload the scene so that the text in the scene is displayed correctly.

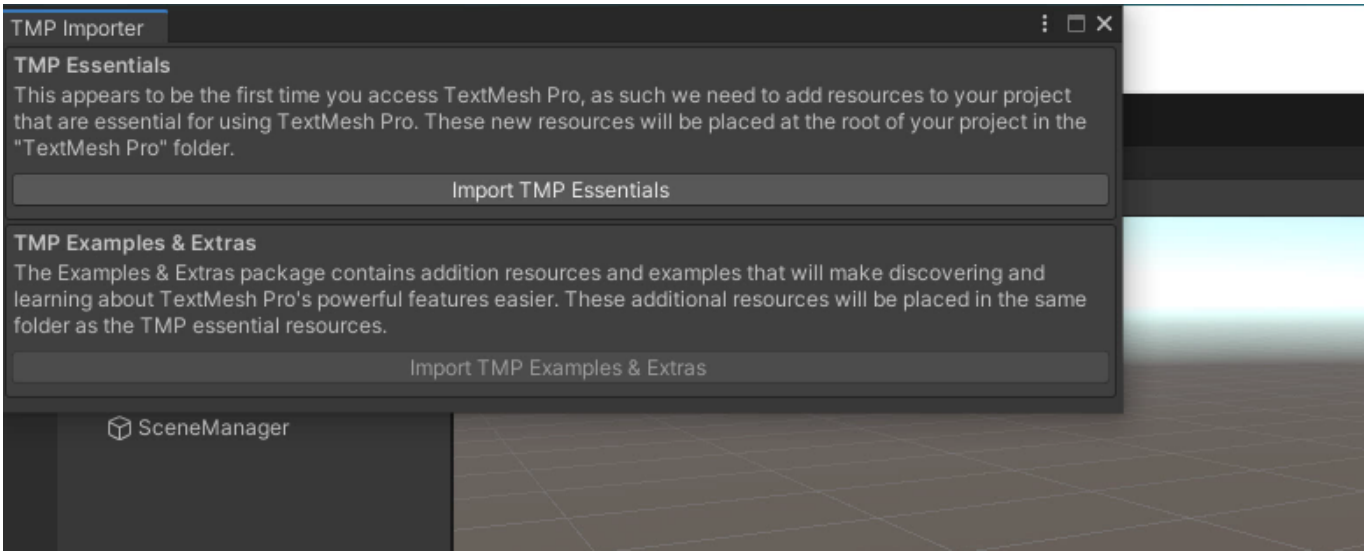


Figure 8 – TMP Import

Navigation in the demo scenes:

Press the "**T**" Key to enable/disable the navigation. By moving the stylus to the left or right side you can move to the next or previous demo section. Press the "**C**" Key to center the demo section.

To quickly navigate to the next section simply swipe the stylus to the left or right with a flick of the wrist.

HapticCollisions Scene

In the HapticCollision scene are different geometry combinations to demonstrate the new feature multiple contact points.

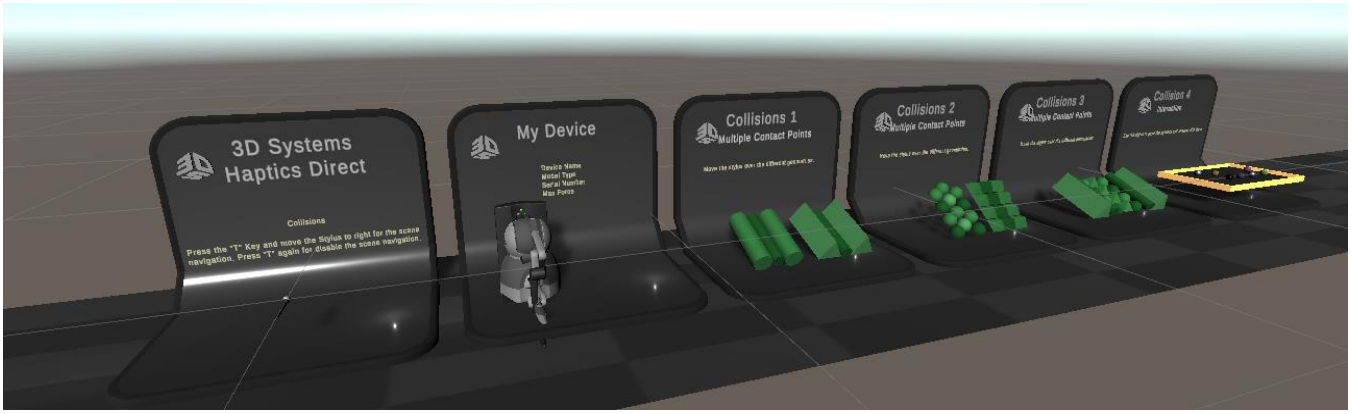


Figure 9 – Haptic Collisions Scene

HapticMaterialBasic Scene

In the HapticMaterialBasic scene the different Haptic Material parameters are explained and demonstrated.

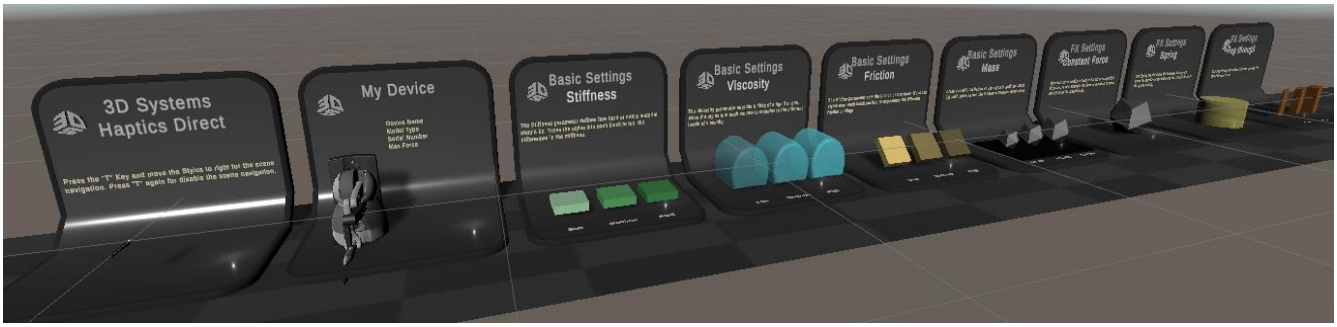


Figure 10 – Haptic Material Basic Scene

HapticMaterialCombination Scene

In the HapticMaterialCombination Scene the combination of different Haptic Material settings are demonstrated.

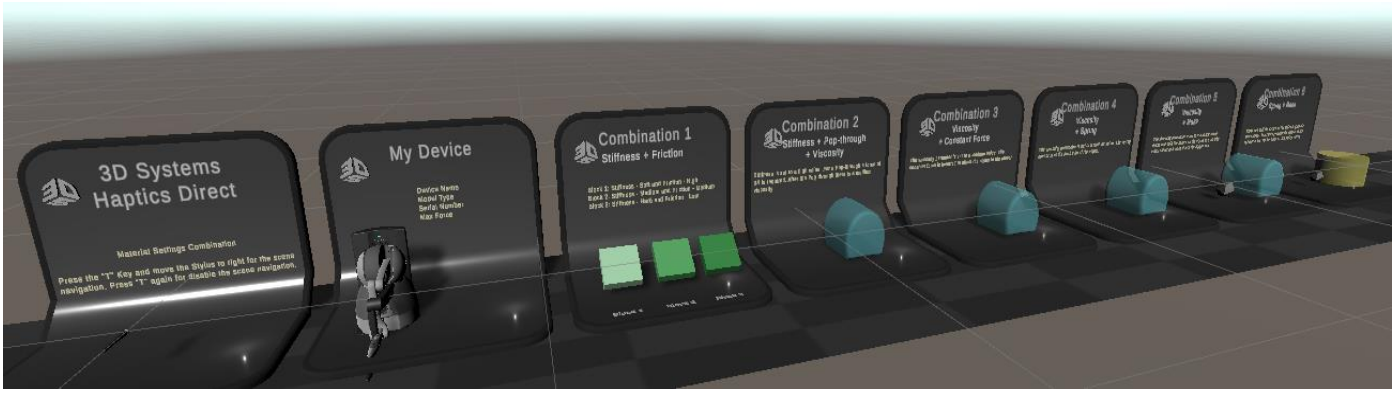


Figure 11 – Haptic Material Combination Scene

HapticPainting Scene

The HapticPainting Scene demonstrates how the Haptic Device can be used as painting tool.

Figure 12 – Haptic Painting Scene

HapticPlugin

This folder encompasses the .dll file which acts as the bridge between Unity and HapticsDirect exposing the various functions of HapticsDirect.

HapticScripts

This folder contains the main plugin scripts and various haptic scripts used in the demos, as well as others that can be added by the user to create their own custom haptic experience.

Plugin Scripts

HapticEditor.cs creates the menu entries for adding the HapticActor, HapticMaterial and HapticCollider to the scene, which are needed for the interaction with the Haptic Device and the virtual environment.

HapticPlugin.cs contains the main functions for the integration of the HapticsDirect API in Unity, event control as well as workspace navigation and other elementary functions for the use of a haptic device in Unity.

HapticCollider.cs has the main task to forward all collision information to the HapticPlugin. There are some additional settings to control the stiffness and friction of the virtual stylus.

HapticMaterial.cs contains all properties and functions for defining the haptic behavior of an object with the virtual stylus.

VirtualHaptic.cs provides all functions to control the virtual haptic model.

Demo Scripts

SceneControl.cs is used in the feature scenes to handle all interactions.

HelpWindow.cs is used in the feature scenes to display the instructions.

HapticPainter.cs simply provides sample implementation of how the Haptic Devices can be used as painting tool.

HapticSwipe.cs provides the swipe functionality.

Prefabs

The Prefabs folder contains two fully preconfigured HapticActors. One prefab each for the Touch and Touch X device.

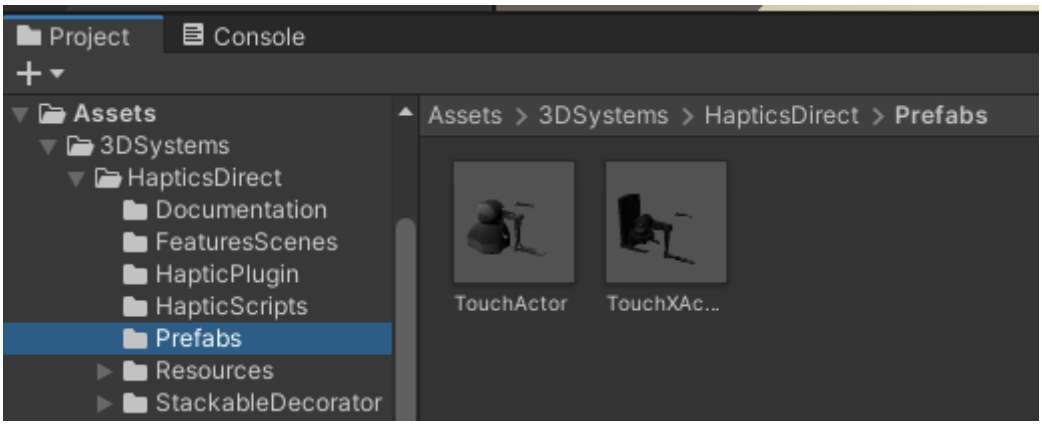


Figure 13 - Prefabs

Resources

The Resources directory contains all 3D models, textures and prefabs used in the demo scenes.

The Haptic Scene

The basic elements of a Haptic Scene are the Haptic Actor, the Virtual Stylus consisting of the visual component and the Collider, and the Haptic Material, which is assigned to the corresponding objects in the scene. The next sections describe the individual components in more detail and explain how they are configured.

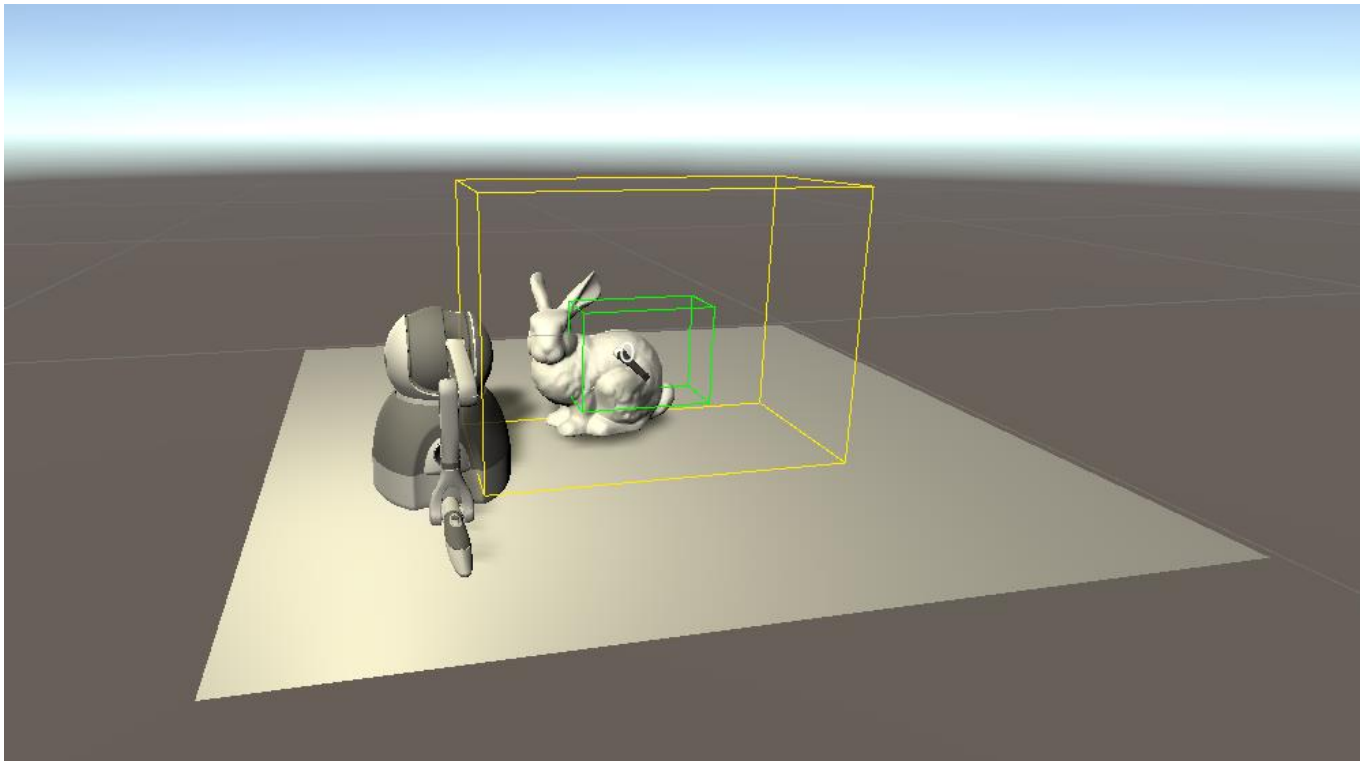


Figure 14 - The Haptic Scene

The Haptic Actor

What is the Haptic Actor?

The Haptic Actor provides all the functionality necessary to interact with the Haptic Device in the virtual environment. The Haptic Actor includes the Haptic Plugin (HapticPlugin.cs), the Visual Mesh and the Collision Mesh (HapticCollider.cs).

How do you create a Haptic Actor?

There are several ways to add a Haptic Actor to the scene. The following sections show the possibilities.

Add Haptic Actor using Prefabs

One of the easiest ways to add a Haptic Actor to the scene is to use the two prefabs provided. The advantage of this method is that in addition to the Haptic Actor, the On Screen Stylus is also fully configured and the Virtual Haptic is added to the scene.

In the Project window, navigate to the **HapticsDirect->Prefabs** directory (see Figure) and add one of the prefabs to the scene. The difference between the two prefabs is only in the 3D model of the Virtual Haptics (Touch / TouchX).

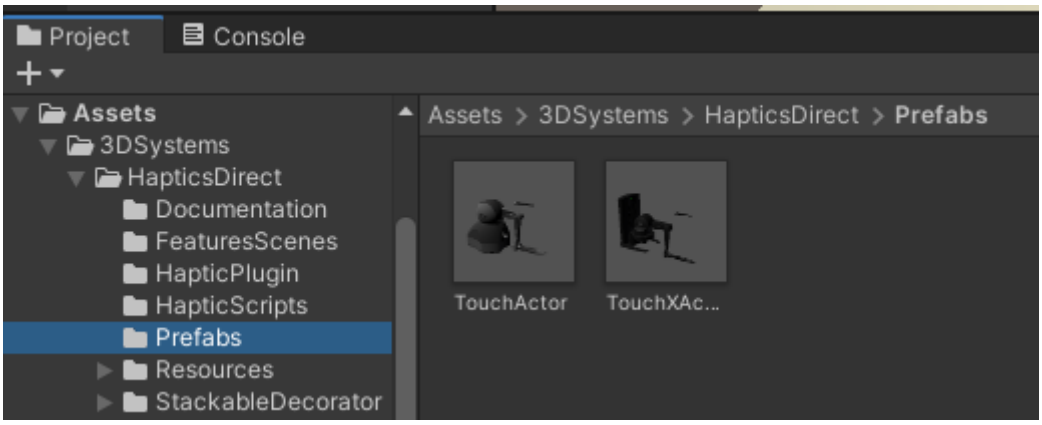


Figure 15 - Add Haptic Actor using Prefabs

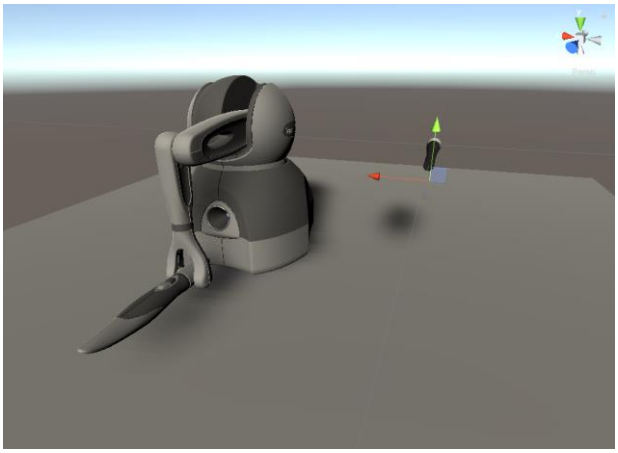


Figure 16 - Prefab Touch Device

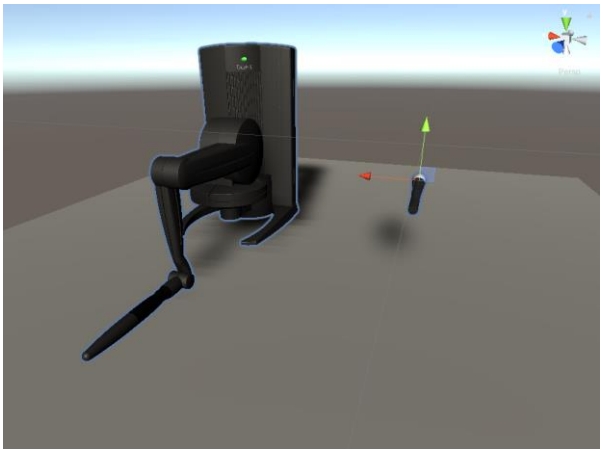


Figure 17 - Prefab TouchX Device

Add Haptic Actor using Menu

1. Navigated to the menu **GameObject->Haptics Direct->Haptic Actor** and add a device according to the device configuration.

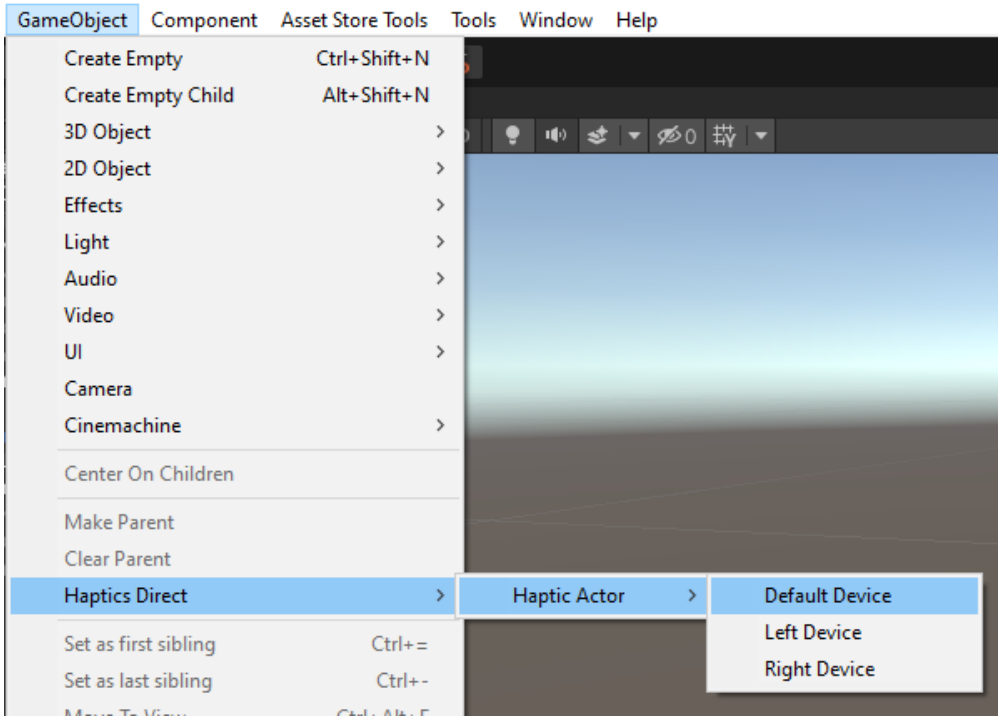


Figure 18 - Add Haptic Actor by menu

2. In the Haptic Actor, assign a mesh GameObject to the Visual Mesh and a HapticCollider GameObject to the Collision Mesh. (see section How to setup a Collision Mesh)

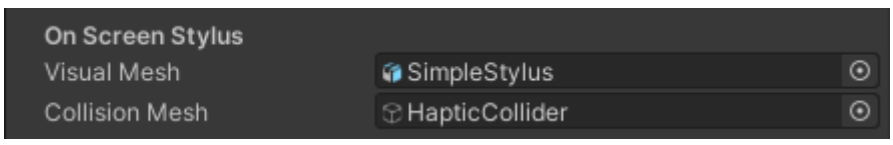


Figure 19 - On Screen Stylus Setup

Add Haptic Actor using Script

1. Create an empty GameObject.
2. Add the HapticPlugin.cs script to the empty GameObject.

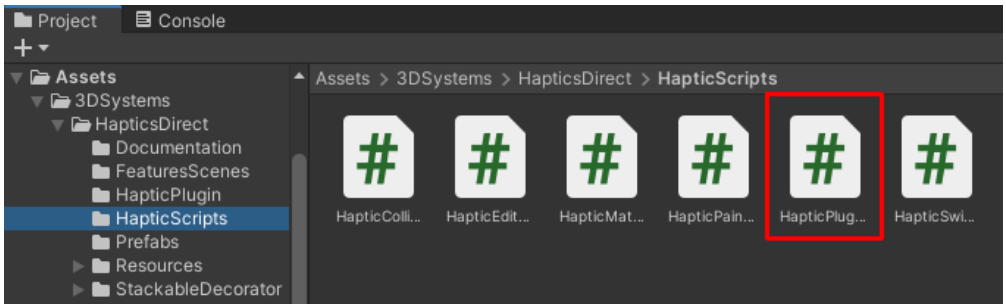


Figure 20 - Haptic Actor by Script

3. In the Haptic Actor, assign a Mesh GameObject to the Visual Mesh and a HapticCollider GameObject to the Collision Mesh (see section How to setup a Collision Mesh).

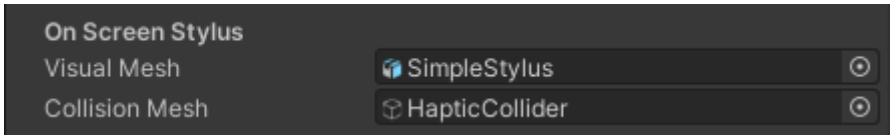


Figure 21 - On Screen Stylus Setup

Overview Properties Sections

Device Setup

Device Identifier: This field by default is set to Default Device. The user can change this after setting up an appropriate profile using the Touch Smart Setup or the Touch Setup application. Note that this field is case sensitive. The Device Identifier must match the ones created using the driver apps. Refer to the Prerequisite section for information on setting up config names.

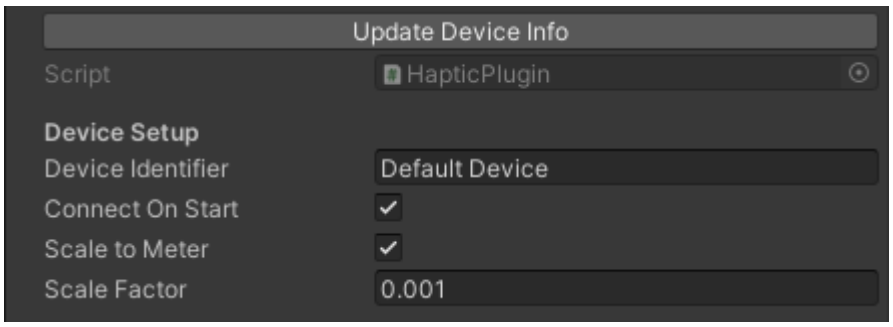


Figure 22 - Device Setup

Update Device Info (Button): Click this button to update the device information and workspace dimensions.

Connect On Start: The default status is checked and causes the haptic device to be initialized in game mode and the servo loop to be started.

Scale to Meter: Activates the scaling of the workspace.

Scale Factor: Defines by which factor the workspace is scaled. Default value 0.001. (mm->m)

On Screen Stylus

The **Visual Mesh** represents the on screen stylus. It can be any mesh.

The **Collision Mesh** has the function of a collider of the on screen stylus. It consists of a (convex) mesh, one or more convex colliders, a rigid body and the HapticCollider.cs script.

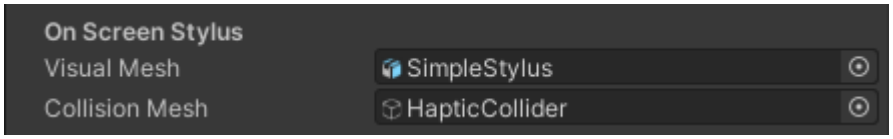


Figure 23 - On Screen Stylus - Setup

Device Information

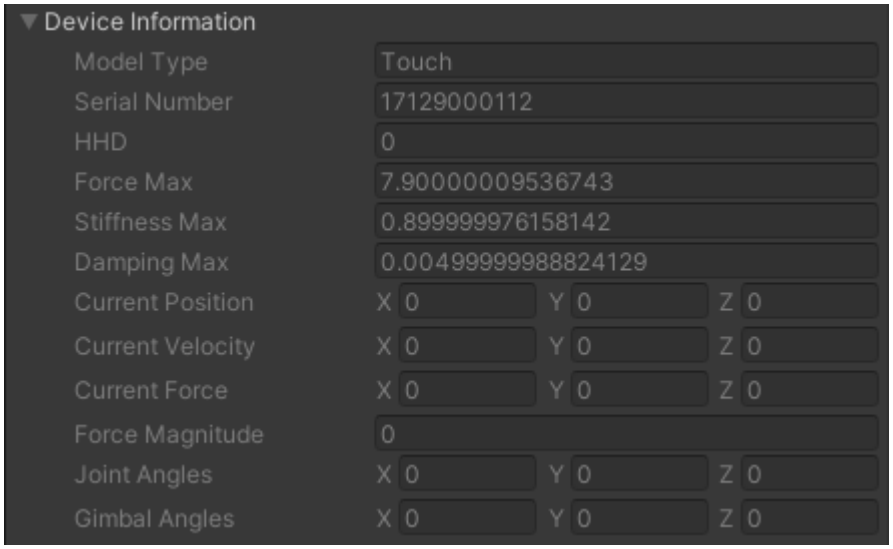


Figure 24 - Device Information

Model Type displays the device model type.

Serial Number displays the device serial number.

HHD displays the device handle of an initialized device

Force Max displays the nominal maximum force, i.e. the amount of force that the device can sustain when the motors are at room temperature (optimal).

Stiffness Max displays the maximum closed loop stiffness that is recommended for the device.

Damping Max displays the maximum level of damping that is recommended for the device.

Current Position displays the current position of the device facing the device base.

Current Velocity displays the current velocity of the device. Note: This value is smoothed to reduce high frequency jitter.

Current Force shows the current force as Cartesian coordinated vector.

Force Magnitude displays the magnitude of the current force vector.

Joint Angles displays the joint angles of the device. These are joint angles used for computing the kinematics of the armature relative to the base frame of the device. For Touch devices: Turret Left +, Thigh Up +, Shin Up + .

Gimbal Angles Get the angles of the device gimbal. For Touch devices: From Neutral position Right is +, Up is -, CW is + .

Simple Buttons Setup

The Simple Button Setup provides a simple way to define the buttons of the haptic device for grabbing and releasing objects. There are different configuration options.

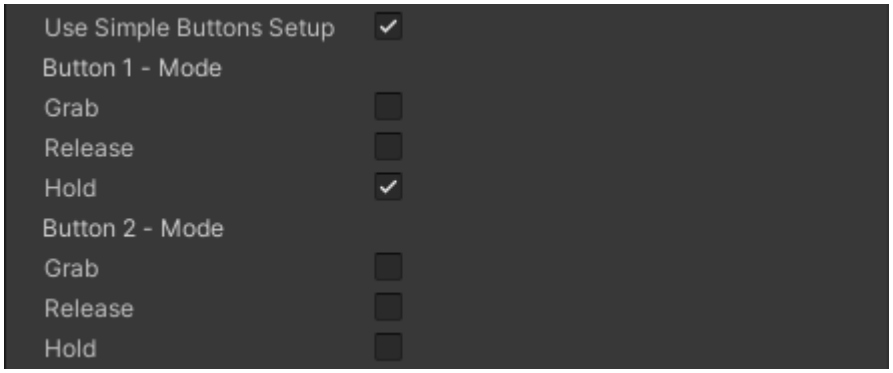


Figure 25 - Simple Buttons Setup

The checkboxes can be used to define when the grabbing mode is activated or deactivated. When Grab and Release are both checked, the button acts as a switch.

Events

The button events can be used to execute user-defined function at the respective event.

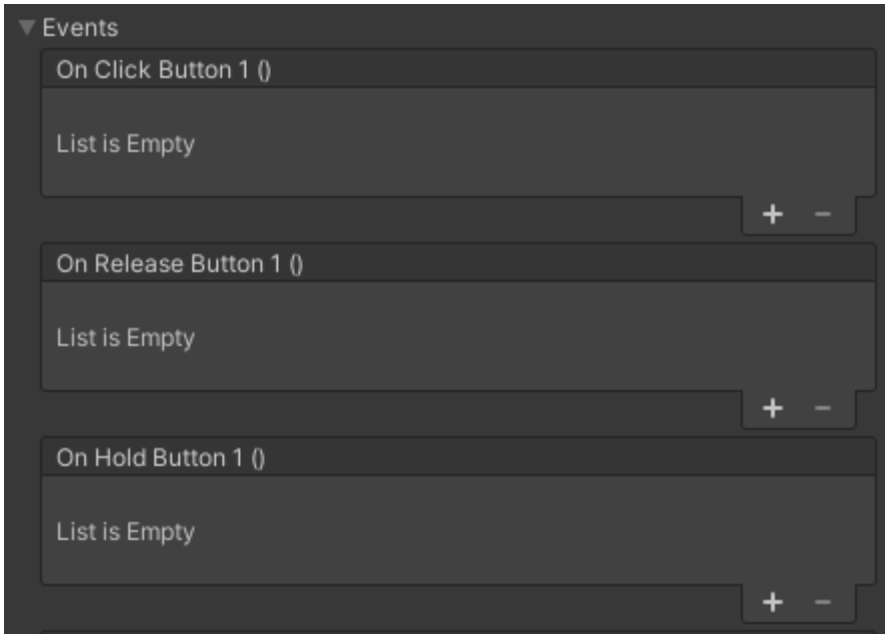


Figure 26 - Button Events

Camera and Navigation

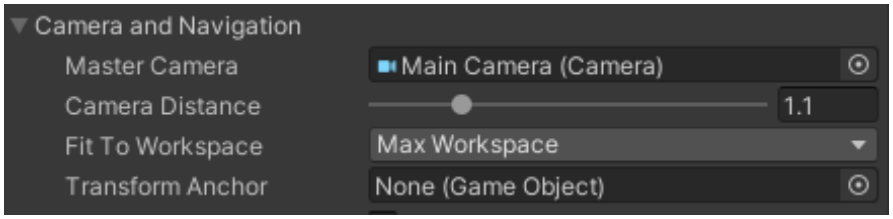


Figure 27 - Camera Setup

Master Camera defines the camera that is used to focus on the Haptic Device Workspace.

Camera Distance sets the distance to the workspace.

Fit To Workspace defines to which workspace (Usable / Max) the camera should focus.

Transform Anchor allows to transform the workspace around an anchor point. (optional)

Figure 28 - Limit Scene Navigation

Limit Scene Navigation - By enabling this option, the transformation range of the workspace can be limited via **Min/Max Position** and **Min/Max Rotation** (Angle). The valid transformation range is displayed as a red box (gizmo).

Freeze Translation - The **X,Y,Z,All** checkboxes can be used to block the translation to one or more axes.

Freeze Rotation - The **X,Y,Z,All** checkboxes can be used to block the rotation to one or more axes.

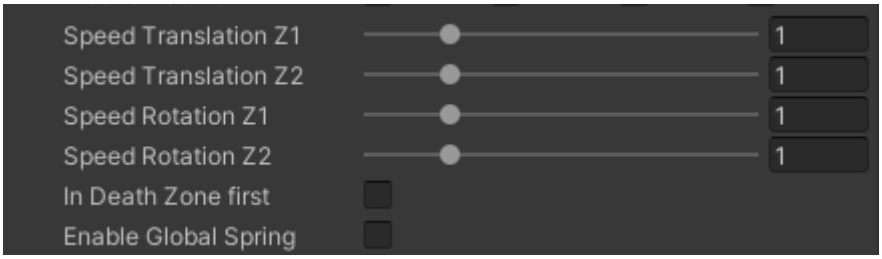


Figure 29 - Speed Control

Speed Translation Z1 – This value defines how fast the workspace is moved in zone 1.

Speed Translation Z2 – This value defines how fast the workspace is moved in zone 2.

Speed Rotation Z1 – This value defines how fast the workspace is rotated in zone 1.

Speed Rotation Z2 – This value defines how fast the workspace is rotated in zone 2.

In Death Zone first – If this option is activated, the angle of the joint must first be in the angle range of the death zone before translation or rotation become active.

Disable Collisions – By enabling this options all collisions will be ignored as long as navigation mode is enabled.

Edit Speed Zones – By enabling this options it is possible to specify the individual zones for navigation. (Joint X,Y,Z – Translation X,Y,Z, Gimbal Joint X,Y,Z - Rotation X,Y,Z)

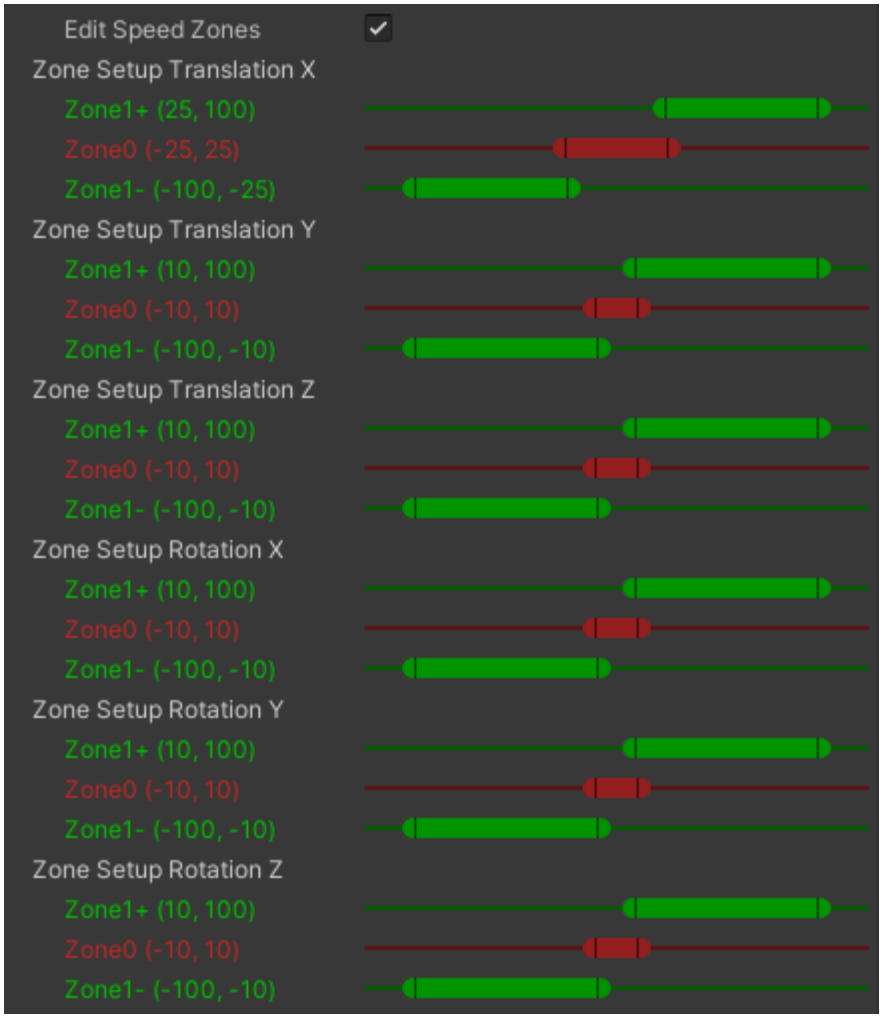


Figure 30 - Speed Zones Setup

Global Vibration Settings

It adds a vibration effect to the total force sent to the haptic device. User must specify: Frequency, Direction, Magnitude and Gain.

Frequency: Defines the frequency of the vibration.

Magnitude: Defines the magnitude of the vibration force.

Direction: Defines the direction of the vibration.

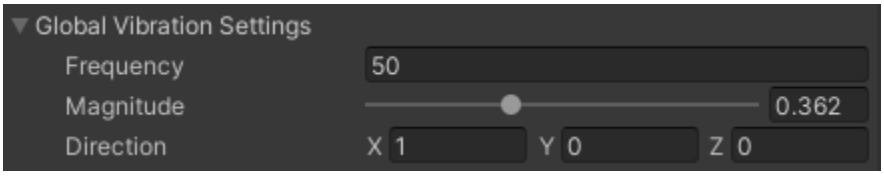


Figure 31 - Global Vibration Settings

Global Spring Settings

It adds a spring force to the local force sent to the haptic device. The spring force pulls the haptic device towards the effect position and is proportional to the product of the magnitude and the distance between the spring anchor and the device position.

Specifically, the spring force is calculated using the expression F = k(P-X) where F is the spring force, P is the spring anchor, X is the current haptic device position and k is the magnitude.

Magnitude: Defines the magnitude of the spring force.

Anchor: Defines the anchor position of the spring.

Anchor Object: A games object can be used as spring anchor.

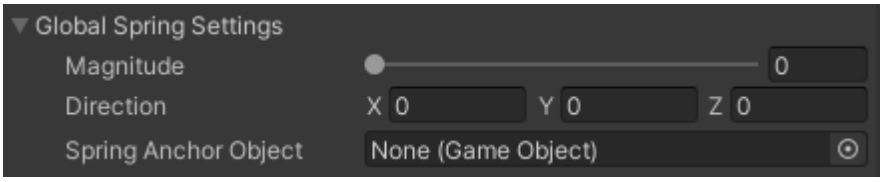


Figure 32 - Global Spring Settings

Global Constant Force Settings

It adds a constant force vector to the local force sent to the haptic device. User can specify the direction and magnitude of the force vector.

Direction: Defines the direction of the force vector.

Magnitude: Defines the magnitude of the force.

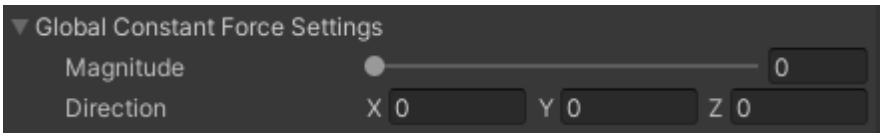


Figure 33 - Global Constant Force Settings

Haptic Collider

The Haptic Collider consists of the components: Mesh Filter, Collider (mesh, or basic collider), Rigidbody and the Haptic Collider (script). A mesh renderer is not mandatory, but can be added as a visualization during creation. Note that only convex colliders work and that "Use Gravity" and "Is Kinematic" are disabled for the Rigidbody.

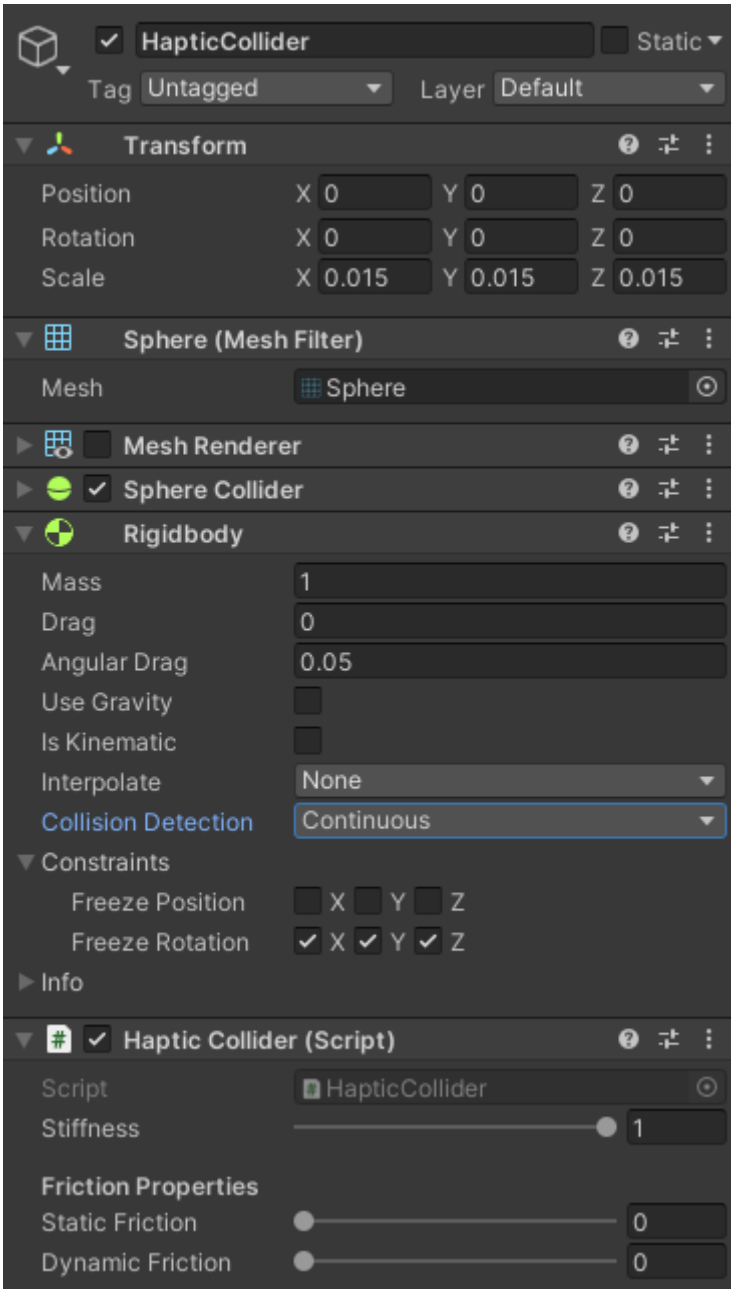


Figure 34 - Haptic Collider Components

How do you create a Haptic Collider?

Create a Haptic Collider by using the Menu

1.) Import or create a mesh to be used as a haptic collider. (e.g. a sphere)

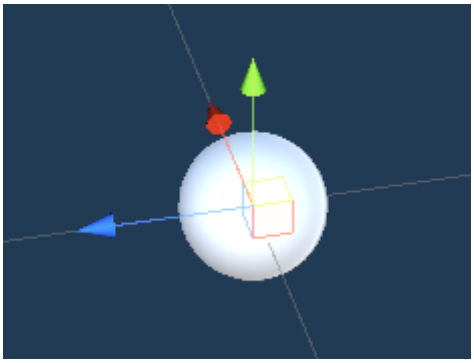


Figure 35 - Sphere Mesh

2.) Add a Collider component and a Rigidbody component.
3.) Navigate to **Component->Haptics Direct->Haptic Collider** and add this component to the sphere game object.

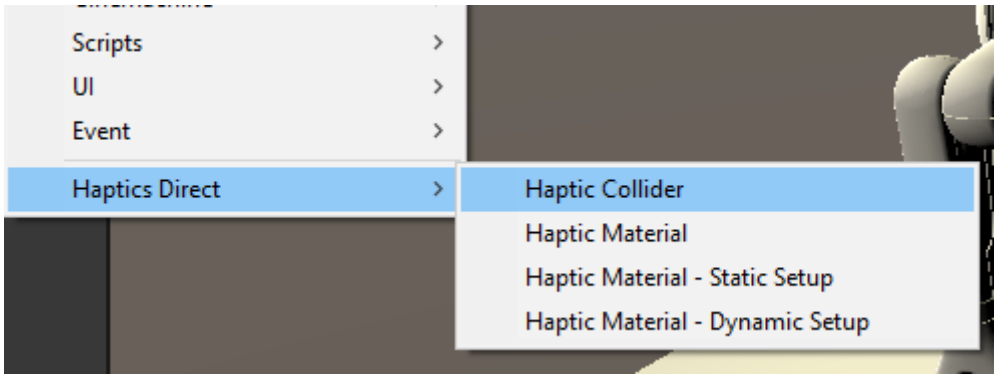


Figure 36 - Add Haptic Collider

If you forget to add a Collider Component, the following warning will be displayed.

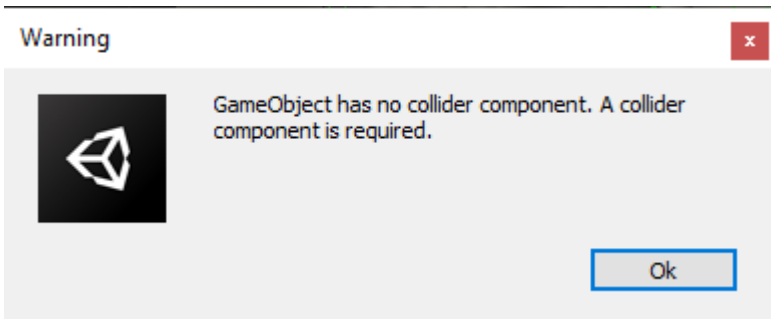


Figure 37 - Warning: Missing Collider Component

If there is no rigidbody component, the following dialog will be displayed. It is possible to add the rigidbody to the game object in this step.

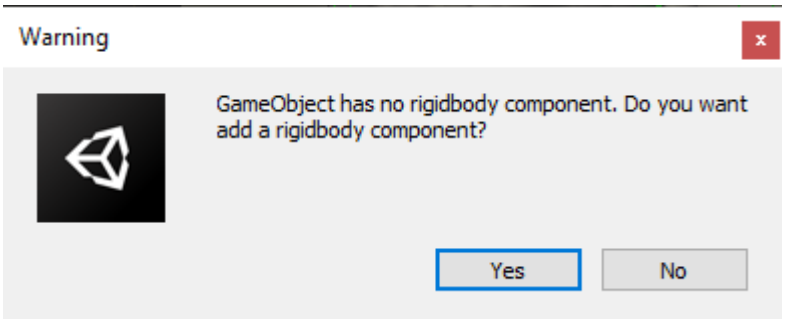


Figure 38 - Dialog Missing Rigidbody

Now the Haptic Collider gameobject can be assigned to the Haptic Actor.

Create a Haptic Collider by adding the Script

1.) Make sure that the game object that will act as a haptic collider has the following components: **Mesh Filter**, **Collider** (mesh, or basic collider), **Rigidbody**
2.) In the Project tab, navigate to **3DSystem->HapticsDirect->HapticScripts** and add the **HapticCollider.cs** script to the game object.

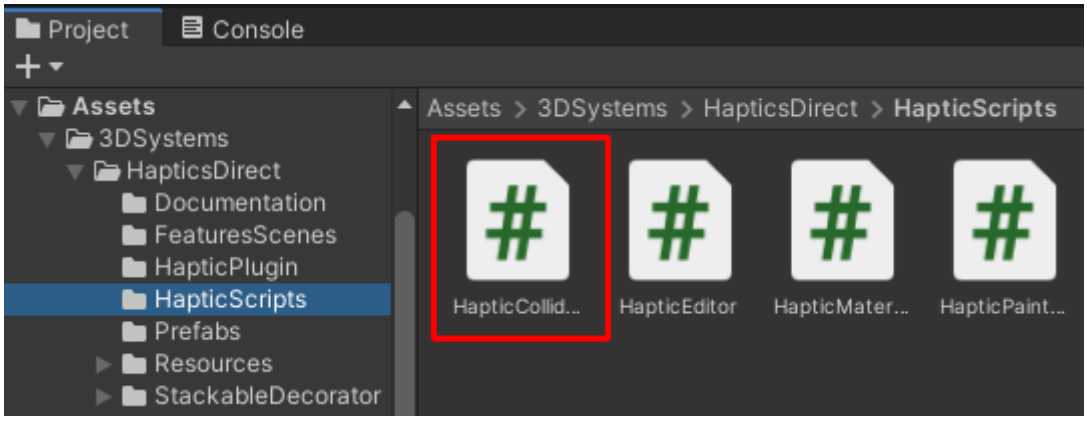


Figure 39 - Add HapticCollider script

Haptic Collider (component) settings

Stiffness: The Stiffness parameter can be used to map different hard tools. The resulting stiffness is calculated in combination with the stiffness from the haptic material. Default value is one.

Static Friction: The Static Friction parameter can be used to map different friction behavior of the tool. Default value is zero.

Dynamic Friction: The Dynamic Friction parameter can be used to map different friction behavior of the tool. Default value is zero.

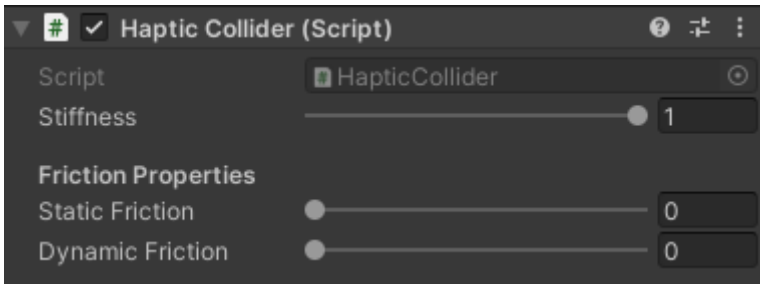


Figure 40 - Haptic Collider settings

Haptic Material

The Haptic Material provides various settings to simulate the haptic behavior with a surface or an object. These include Stiffness, Damping, Viscosity, Friction and some local **effects** like Constant Force, Spring and Pop-through.

How do you create a haptic material?

Add a Haptic Material component by Menu

1. Before a haptic material can be added to a game object, make sure that the corresponding game object contains a collider component and a rigidbody component.

2. Select the game object(s) to which you want to add a Haptic Material component.

3. Navigate to Component->HapticsDirect and click on Haptic Material.

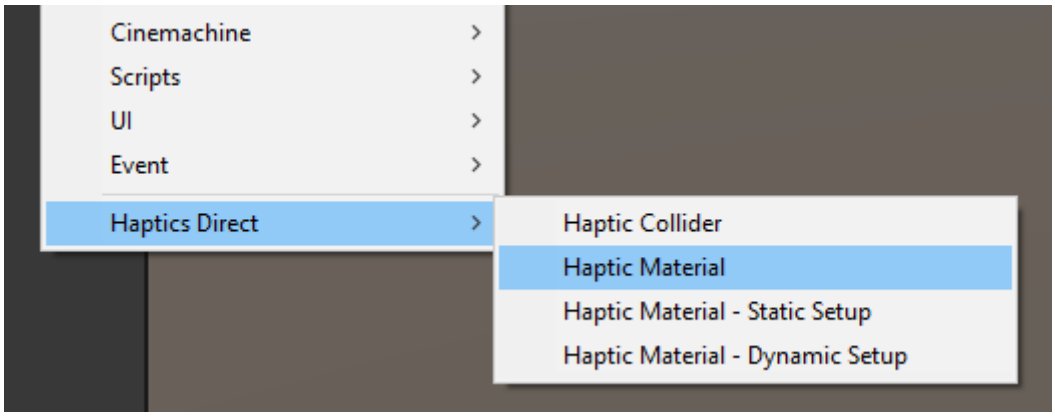


Figure 41 - Add Haptic Material by Menu

4. The corresponding settings for the haptic material can now be made. (see section Haptic Material settings)

Add a Haptic Material component by Script

1. Before a haptic material can be added to a game object, make sure that the corresponding game object contains a collider component and a rigidbody component.

2. In the Project tab, navigate to **3DSystem->HapticsDirect->HapticScripts** and add the **HapticMaterial.cs** script to the game object.

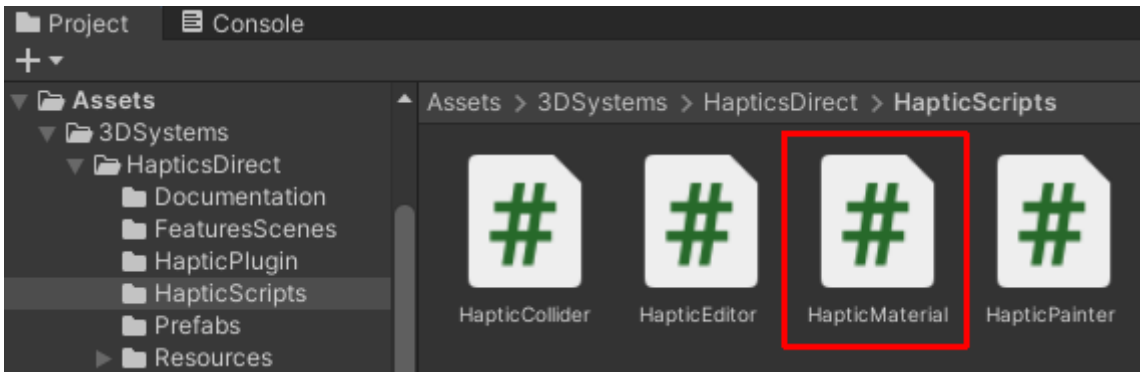


Figure 42 - Add Haptic Material by Script

3. The corresponding settings for the haptic material can now be made. (see section Haptic Material settings)

Haptic Material settings

Basic Properties

Grabbing: When this option is enabled, the game object can be grabbed with the virtual stylus.

Mass: Defines the object mass in kg. This parameter is used for calculating the gravity force.

Stiffness: The stiffness Haptic material surface directly translates to how hard a surface feels and haptic feedback can be expressed using the formula, F=Kx where F is the force applied is directly proportional to the distance (x) and K is the spring constant.

Note: If stiffness is set at “Zero” then no other haptic material effect can be felt.

Damping: Damping reduces the spring effect of the surface. Param must be between 0 and 1 where 0 represents no damping, i.e. a highly springy surface and 1 represents the maximum level of damping possible. Damping is added to reduce rebound effect.

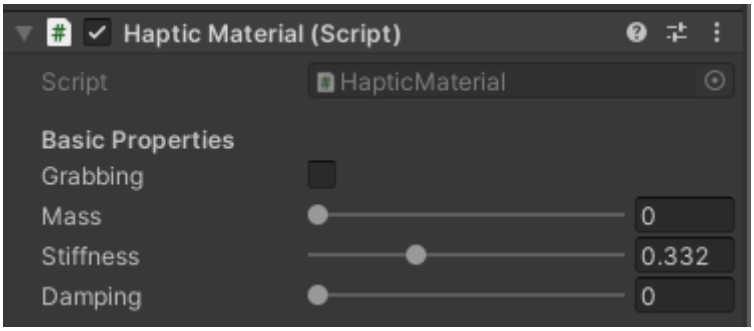


Figure 43 - Haptic Material - Basic Properties

Viscosity Properties

It adds a viscous force to the local force sent to the haptic device. The viscous force is based on the current velocity of the haptic device and is calculated to resist the motion of the haptic device. Specifically the force is calculated using the expression F = -kV where f is the spring force, V is the velocity and k is the magnitude.

Friction Properties

Static Friction: The surface of the object (eg: plane) how hard it is to slide along the surface starting from a complete stop. A param value of 0 is a completely frictionless surface and a value of 1 is the maximum amount of static friction the haptic device is capable of rendering.

Dynamic Friction: This property defines how hard it is to slide along the surface once already moving. A param value of 0 is a completely frictionless surface and a value of 1 is the maximum amount of dynamic friction the haptic device is capable of rendering.

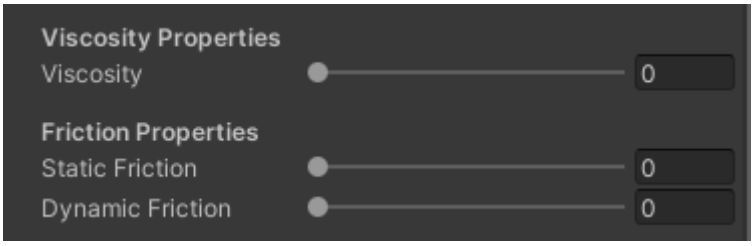


Figure 44 - Viscosity and Friction

Constant Force Properties

It adds a constant force vector to the local force sent to the haptic device. User can specify the direction and magnitude of the force vector.

Direction: Defines the direction of the force vector.

Magnitude: Defines the magnitude of the force.

Use Contact Normal: If this option is activated, the contact normal is used as direction for the force vector.

Contact Normal Inverse: If this option is activated, the inverse contact normal is used as direction for the force vector.

Spring Properties

It adds a spring force to the local force sent to the haptic device. The spring force pulls the haptic device towards the effect position and is proportional to the product of the magnitude and the distance between the spring anchor and the device position.

Specifically, the spring force is calculated using the expression F = k(P-X) where F is the spring force, P is the spring anchor, X is the current haptic device position and k is the magnitude.

Spring Anchor: Defines the anchor position of the spring.

Spring Anchor Object: A games object can be used as spring anchor.

Spring Magnitude: Defines the magnitude of the spring force.

Popthrough Properties

Popthrough controls the amount of force the user must apply to a geometry before the device pops through the surface to the other side. The larger the param value, the higher the force required. A param value of 0 disables pop through.

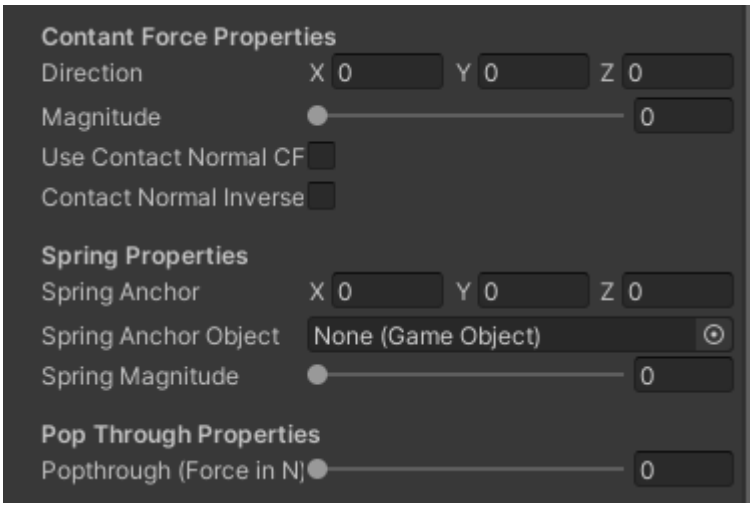
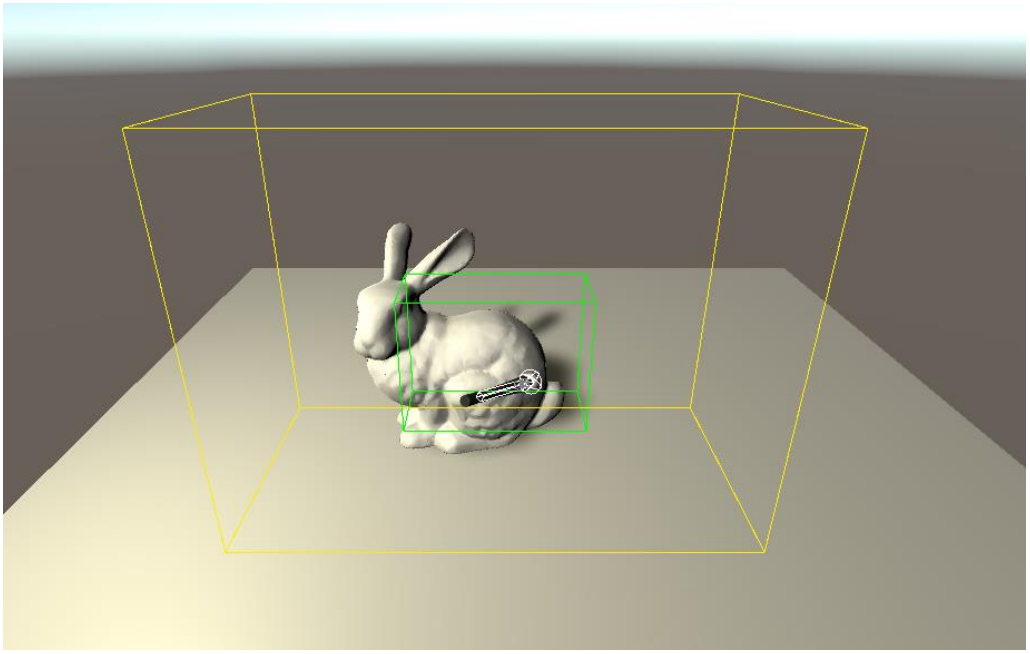


Figure 45 - Local Effects

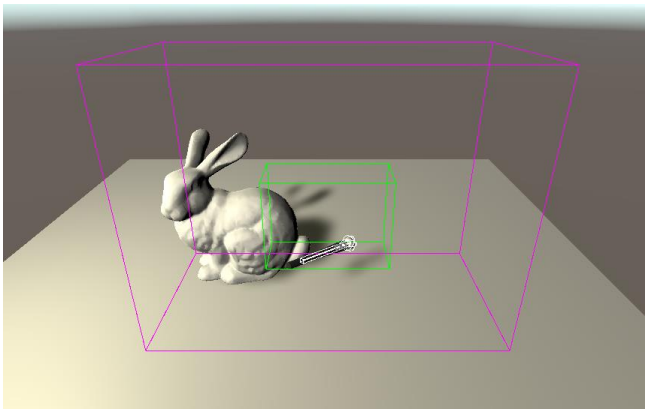
Gizmos

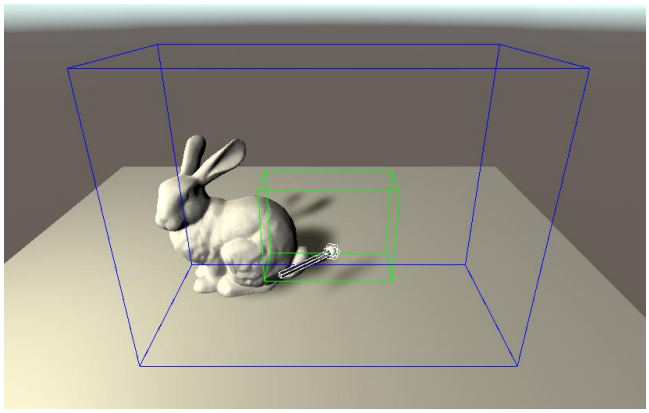
Workspace Gizmo

The workspace gizmo shows the useable workspace as a green box and the maximum workspace as a yellow box. If the workspace is not displayed, the device information may be missing. The device information can be updated in the HapticActor by clicking on "Update Device Information".



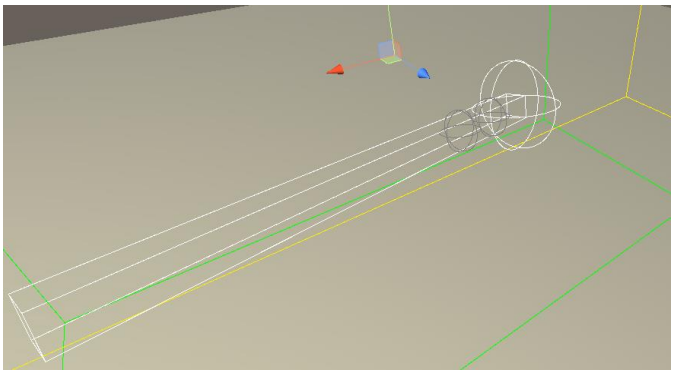
When activating the translation mode, the maximum workspace is displayed in a purple color and when activating the rotation mode, the color changes to blue.



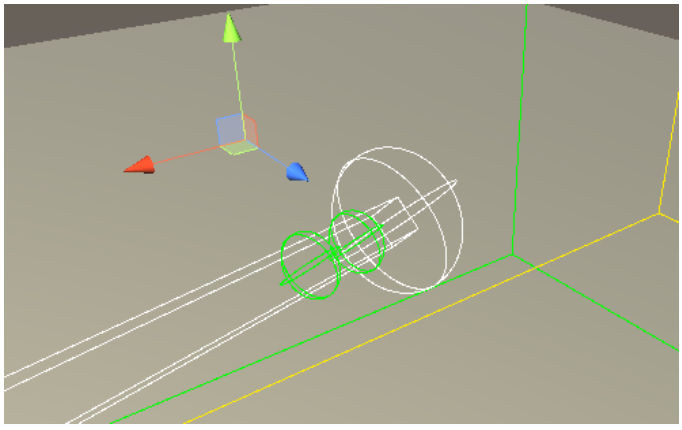


Virtual Stylus Gizmo

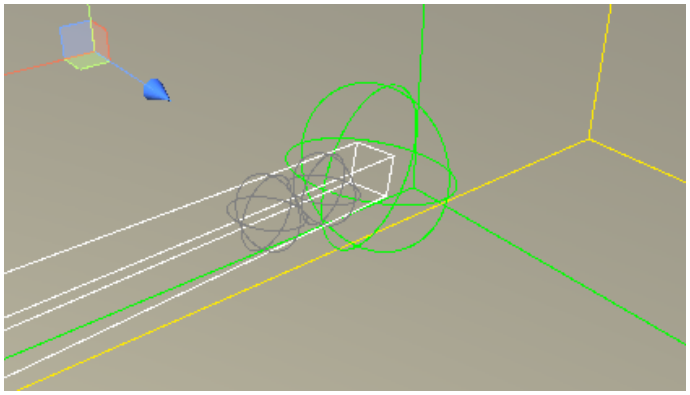
The Virtual Stylus Gizmo is shown as box and sphere in white and the buttons in gray.



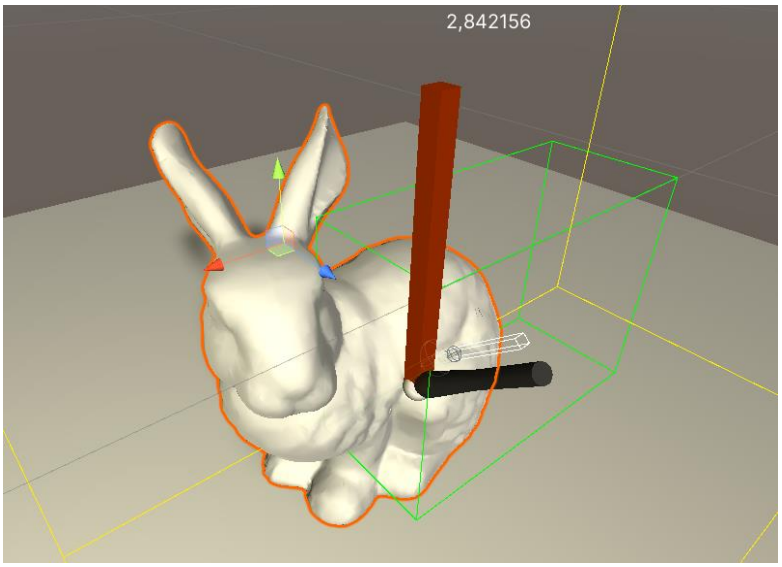
When a button is pressed, the color of the respective button changes from gray to green.



The white sphere changes color to green when grabbing mode is active.

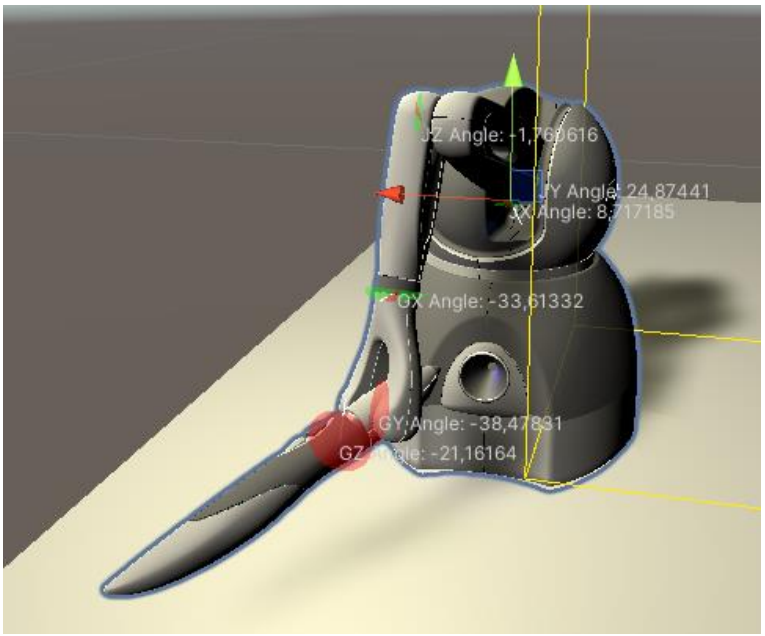


When interacting with other objects, a ForceMeter is displayed, which shows the force as a bar and as a value (in Newton).



Virtual Haptic Gizmo

In the Virtual Haptic, the angle ranges for navigation are displayed for each joint, as well as the current angle as a white line and as a numerical value. The red angle section shows the "Death Zone" in which no translation or rotation is active. The two green areas show the angle range of speed zone 1. Any angle outside of speed zone 1 is speed zone2. A complete red circle indicates that this joint is locked.



API Haptics Direct – Scripts

Haptic Plugin

getVersionString(dest,len)	
Description	Retreives the HapticsDirect version string.
Parameters	StringBuilder dest int len
Return Value	void

initDevice(deviceName)	
Description	Connects to and Initializes a haptic device
Parameters	string deviceName (e.g. *Default Device, Left Device, Right Device*)
Return Value	int - HHD > 0, if connection was successful,

getDeviceSN(configName, dest,len)	
Description	Retrieves device serial number of the Haptic Device
Parameters	string deviceName (e.g. *Default Device, Left Device, Right Device*) StringBuilder dest int len
Return Value	void

getDeviceModel(configName, dest, len)	
Description	Retrieves devices model name of the Haptic Device
Parameters	string deviceName (e.g. *Default Device, Left Device, Right Device*) StringBuilder dest int len
Return Value	void

getDeviceMaxValues(configName, max_stiffness, max_damping, max_force)	
Description	Retrieves devices model name of the Haptic Device
Parameters	string deviceName (e.g. *Default Device, Left Device, Right Device*) ref double max_stiffness (*maximal stiffness of the haptic device*) ref double max_damping (*maximal damping of the haptic device*) ref double max_force (*maximal force of the haptic device*)
Return Value	void

startSchedulers()	
Description	Starts the Haptics Direct schedulers and assigns the required internal callbacks
Parameters	
Return Value	void

getWorkspaceArea(configName, usable6, max6**)**	
Description	Retrieves the bounds created by the physical limitations of the device.
Parameters	string deviceName (e.g. *Default Device, Left Device, Right Device*) double[] usable6 (*useable workspace, [xmin,ymin,zmin,xmax,ymax,zmax]*) double[] max6 (*maximal workspace, [xmin,ymin,zmin,xmax,ymax,zmax]*)
Return Value	void

getPosition(configName, position3**)**	
Description	Get the current position in mm of the device facing the device base. Left is + x, up is +y, toward user is +z. (Unity CSys)
Parameters	string deviceName (e.g. *Default Device, Left Device, Right Device*) double[] position3 (*Position [x,y,z]*)
Return Value	void

getVelocity(configName, velocity3**)**	
Description	Get the current velocity in mm/s of the device. Note: This value is smoothed to reduce high frequency jitter. (Unity CSys)
Parameters	string deviceName (e.g. *Default Device, Left Device, Right Device*) double[] velocity3 (*Velocity [x,y,z]*)
Return Value	void

getTransform(configName, matrix16**)**	
Description	Get the column-major transform of the device endeffector. (Unity CSys)
Parameters	string deviceName (e.g. *Default Device, Left Device, Right Device*) double[] matrix16
Return Value	void

getButtons(configName, buttons4, last_buttons4, inkwell**)**	
Description	Get the button, last button states and get whether the inkwell switch, if one exists is active.
Parameters	string deviceName (e.g. *Default Device, Left Device, Right Device*) int[] buttons4 (*[button1_state,button2_state,...]*) int[] last_buttons4 (*[last_button1_state,last_button2_state,...]*) ref int inkwell (*0 = off, 1 = on*)
Return Value	void

getCurrentForce(configName, currentforce3**)**	
Description	Get the current force in N of the device. (Unity CSys)
Parameters	string deviceName (e.g. *Default Device, Left Device, Right Device*) double[] currentforce3
Return Value	void

getJointAngles(configName, jointAngles, gimbalAngles**)**		
Description	Get the joint angles in rad of the device. These are joint angles used for computing the kinematics of the armature relative to the base frame of the device. For Touch devices: Turret Left +, Thigh Up +, Shin Up + Get the angles in rad of the device gimbal. For Touch devices: From Neutral position Right is +, Up is -, CW is +	
Parameters	string	deviceName (e.g. *Default Device, Left Device, Right Device*)
	double[]	jointAngles (*[x_angle,y_angle,z_angle]*)
	double[]	gimbalAngles (*[gimbal_x_angle,gimbal_y_angle,gimbal_z_angle]*)
Return Value	void	

setForce(configName, lateral3, torque3**)**		
Description	Set the current force as Cartesian coordinated vector in N. to the haptic device. Can be used for scripted forces, but in most cases using an Effect is preferable.	
Parameters	string	deviceName (e.g. *Default Device, Left Device, Right Device*)
	double[]	lateral3 (in N)
	double[]	torque3 (in Nm) 6DOF only
Return Value	void	

setAnchorPosition(configName, position3**)**		
Description	Set the anchor position of the virtual stylus (Unity CSys)	
Parameters	string	deviceName (e.g. *Default Device, Left Device, Right Device*)
	double[]	position3 (*Position [x,y,z]*)
Return Value	void	

addContactPointInfo(configName, Location, Normal, MatStiffness, MatDamping**,** MatForce, MatViscosity, MatFrictionStatic, MatFrictionDynamic, MatConstForceDir, MatConstForceMag, MatSpringDir, MatSpringMag, MatPopThroughRel, MatPopThroughAbs, MatMass, RigBSpeed, RigBVelocity, RigBAngularVelocity, RigBMass, ColImpulse, PhxDeltaTime**)**		
Description	Add a collision contact point info to the contact points list	
Parameters	string	deviceName (e.g. *Default Device, Left Device, Right Device*)
	double[]	Location (*Location [x,y,z]*)
	double[]	Normal (*Normal [x,y,z])*
	float	MatStiffness (*stiffness*)
	float	MatDamping (*damping*)
	double[]	MatForce (*force*)
	float	MatViscosity (*viscosity*)
	float	MatFrictionStatic (*static friction*)
	float	MatFrictionDynamic (*dynamic friction*)
	double[]	MatConstForceDir (*constant force direction*)
	float	MatConstForceMag (*constant force magnitude*)
	double[]	MatSpringDir (*spring direction*)
	float	MatSpringMag (*spring magnitude*)
	float	MatPopThroughRel (*relative force pop through*)
	float	MatPopThroughAbs (*absolute force pop through*)
	double	MatMass (*mass in kg*)

	double	RigBSpeed (*rigid body speed*)
	double[]	RigBVelocity (*rigid body velocity*)
	double[]	RigBAngularVelocity (*rigid body angular velocity*)
	double	RigBMass (*rigid body mass*)
	double[]	ColImpulse (*collision impulse*)
	double	PhxDeltaTime (*physics delta time*)
Return Value	void	

updateContactPointInfo(configName**)**		
Description	Update the contact point info list	
Parameters	string	deviceName (e.g. *Default Device, Left Device, Right Device*)
Return Value	void	

resetContactPointInfo(configName**)**		
Description	Reset the contact point info list	
Parameters	string	deviceName (e.g. *Default Device, Left Device, Right Device*)
Return Value	void	

setVibrationValues(configName, direction3, magnitude, frequency, time**)**		
Description	Set the parameters of the Vibration FX	
Parameters	string	deviceName (e.g. *Default Device, Left Device, Right Device*)
	double[]	direction3 (vibration direction [x,y,z])
	double	magnitude (vibration magnitude)
	double	frequency (vibration frequency in Hz)
	double	time (duration in s)
Return Value	void	

setSpringValues(configName, anchor, magnitude**)**		
Description	Set the parameters of the Spring FX	
Parameters	string	deviceName (e.g. *Default Device, Left Device, Right Device*)
	double[]	anchor (*anchor point [x,y,z])*
	double	magnitude (*spring magnitude*)
Return Value	void	

setConstantForceValues(configName, direction, magnitude**)**		
Description	Set the parameters of the Constant Force FX	
Parameters	string	deviceName (e.g. *Default Device, Left Device, Right Device*)
	double[]	direction (*direction [x,y,z]*)
	double	magnitude (*force magnitude*)
Return Value	void	

setGravityForce(configName, gForce3**)**	
Description	Set the gravity force

Parameters	string double[]	deviceName (e.g. *Default Device, Left Device, Right Device*) gForce (*Force [x,y,z]*)
Return Value	void	

disconnectAllDevices()	
Description	Disconnect all devices and stop all schedulers
Parameters	void
Return Value	void

int getHDError(Info, len**);**		
Description	Retreives the HD Error	
Parameters	StringBuilder int	Info (*error string*) len
Return Value	Int	Error Code

CameraUpdate()	
Description	Update the assigned camera transformation
Parameters	void
Return Value	void

AngleToNeg(angle**)**		
Description	Converts an angle into negative value	
Parameters	float	angle (*0-360 deg*)
Return Value	float	angle (*-180 – 180*)

AngleToPos(angle**)**		
Description	Converts an angle into positive value	
Parameters	float	angle (*-180 - 180 deg*)
Return Value	float	angle (*0 – 360 deg*)

isInRotRange()		
Description	Checks if the transformation angles are in range of the rotation limits	
Parameters	void	
Return Value	bool[]	[*MIN_X,MAX_X,MIN_Y,MAX_Y,MIN_Z,MAX_Z*]

isInTransRange()		
Description	Checks if the translation values are in range of the translation limits	
Parameters	void	
Return Value	bool[]	[*MIN_X,MAX_X,MIN_Y,MAX_Y,MIN_Z,MAX_Z*]

isInZone(value, zone)	
Description	Checks if the value is in range of the defined zone (Used in workspace transformation)
Parameters	float value Vector2 zone
Return Value	bool [*true, if in Zone*]

UpdateWorkspaceTransform()	
Description	Update the workspace transformation
Parameters	void
Return Value	void
UpdateDZZero()	
Description	Update the status, if a joint is in the “death zone” (Used in workspace transformation)
Parameters	void
Return Value	void

InitializeHapticDevice()	
Description	Initialize the Haptic device
Parameters	void
Return Value	bool true = successful, false = failed

DisconnectHapticDevice()	
Description	Disconnect the Haptic device
Parameters	void
Return Value	void

GetDeviceTransformationRaw()	
Description	Get the raw transformation of the Haptic device
Parameters	void
Return Value	void

UpdateDeviceInformation()	
Description	Get and update the device information
Parameters	void
Return Value	void

UpdateTransfrom()	
Description	Update the transformation of the virtual stylus (Visual Mesh, Collision Mesh)
Parameters	void
Return Value	void

UpdateButtonStatus()	
Description	Update the button states and invoke button events
Parameters	void
Return Value	void

EnableVibration()	
Description	Enable the Vibration FX
Parameters	void
Return Value	void

DisableVibration()	
Description	Disable the Vibration FX
Parameters	void
Return Value	void

EnableSpring()	
Description	Enable the Spring FX
Parameters	void
Return Value	void

DisableSpring()	
Description	Disable the Spring FX
Parameters	void
Return Value	void

EnableConstantForce()	
Description	Enable the Constant Force FX
Parameters	void
Return Value	void

DisableContantForce()	
Description	Disable the Constant Force FX
Parameters	void
Return Value	void

EnableNavTranslation()	
Description	Enable the Workspace translation
Parameters	void
Return Value	void

DisableNavTranslation()	
Description	Disable the Workspace translation
Parameters	void
Return Value	void

SwitchNavTranslation()	
Description	Enable / Disable the Workspace translation based on the state
Parameters	void
Return Value	void

EnableNavRotation ()	
Description	Enable the Workspace rotation
Parameters	void
Return Value	void

DisableNavRotation ()	
Description	Disable the Workspace rotation
Parameters	void
Return Value	void

SwitchNavRotation ()	
Description	Enable / Disable the Workspace rotation based on the state
Parameters	void
Return Value	void

isNavRotation()	
Description	Returns the enable / disable state for the workspace rotation
Parameters	void
Return Value	bool false = disable, true = enable

isNavTranslation()	
Description	Returns the enable / disable state for the workspace translation
Parameters	void
Return Value	bool false = disable, true = enable

Grab_Object()	
Description	Enable the object grabbing mode on collision
Parameters	void
Return Value	void

Release_Object()	
Description	Disable the object grabbing mode and release the grabbed object
Parameters	void
Return Value	Void

UpdateCollision(collision, isCollisionStart, isCollision, isCollisionExit**)**	
Description	Update the control members on collisions
Parameters	Collision collision bool isCollisionStart bool isCollision bool isCollisionExit
Return Value	void

UpdateForceOnCollision(collision**)**	
Description	Update the contact point information for calculating the force
Parameters	Collision collision
Return Value	void

SendContactpoints()	
Description	Send the contact point information to the Haptic Direct routines
Parameters	Void
Return Value	void

GrabObj(collision**)**	
Description	Attach an object as fixed joint to the Collision mesh
Parameters	Collision collision
Return Value	void

ReleaseObj()	
Description	Deattach the grabbed object from the Collision mesh
Parameters	Collision collision
Return Value	void