Autodesk MotionBuilder 2013

Programming in MotionBuilder || Focusing on Python

Naiqi Weng, Developer Consultant
Autodesk Developer Network

Module 8

Constraints

Module 8

Today's Agenda

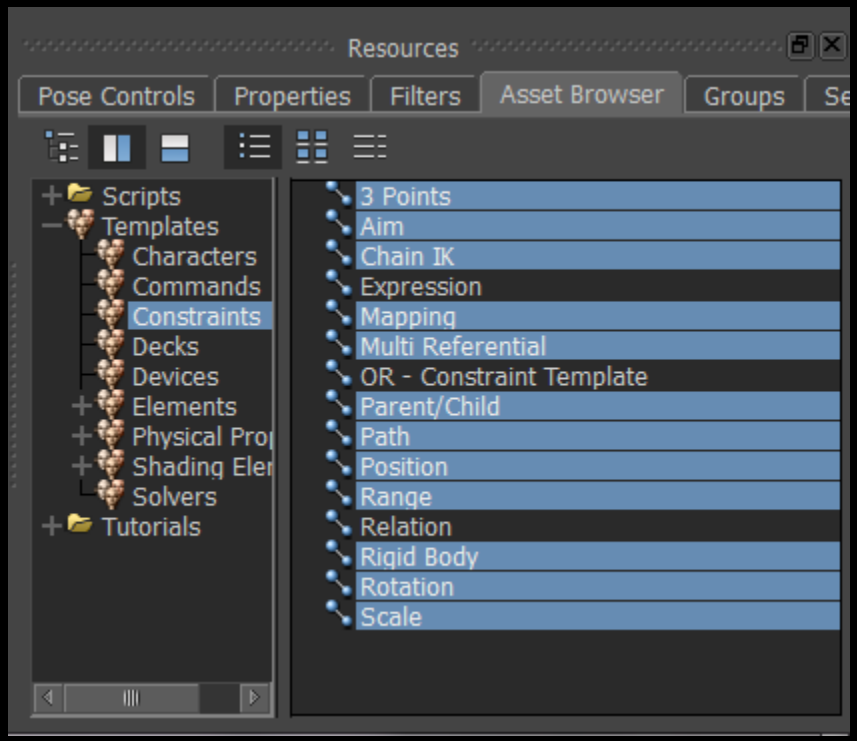
- The 'What' and 'Why' of Constraints?
- Different Constraints Types
- Working with Constraints
- Triggering Scripts in Constraints
- Assignment

The ‘What’ and ‘Why’ of Constraints?

- Real World Constraints
  - gravity constrains us to the ground
  - Dog is constrained by the length of his leash.
  - An eyeball following and object
- You can simulate these limits in the 3D animation world
- Constraints are tools used to create relationships between objects
- Within the connections, complex relationships are formed

Simple Constraints

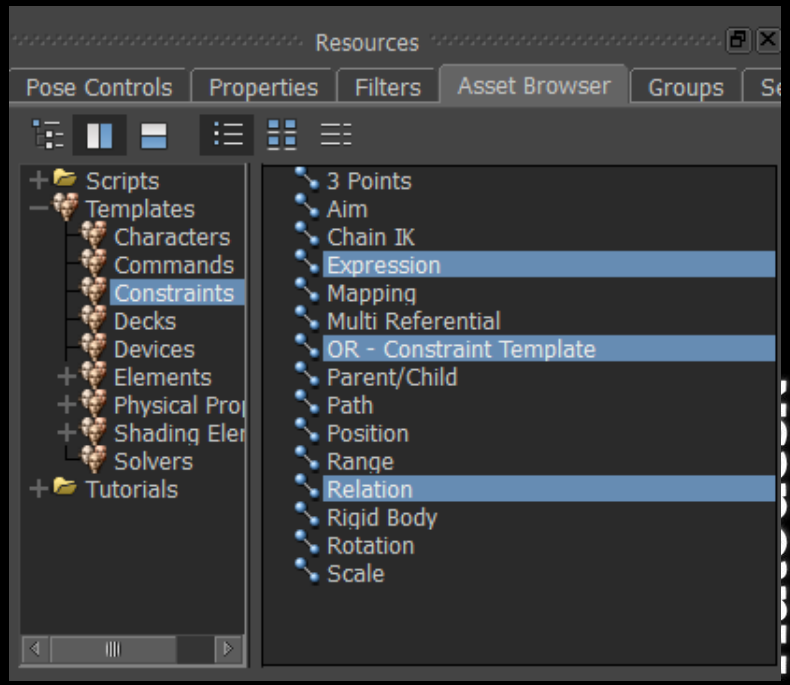
- 3 Points
- Aim
- Chain IK
- Mapping
- Multi-Referential
- Parent/Child
- Path
- Position
- Range
- Rigid Body
- Rotation
- Scale



Autodesk®

Complex Constraints

- Relation
  - Macro Boxes (not covered here)
- Expression (not covered here)
- Custom OR SDK (not covered here)



Creating Simple Constraints

- Create instance of FBConstraintManger
- Adding a Constraint to the scene:
  - TypeCreateConstraint() is used to create constraint

Finding Constraints

- Use FBScene from FBSystem; this returns a list of FBConstraint

```
from pyfbsdk import*

lConstraints = FBSystem().Scene.Constraints

for lCon in lConstraints:
    print lCon.Name
```

Deleting Constraints

- Use FBScene from FBSystem; this returns a list of FBConstraint

```
from pyfbsdk import*

lConstraints = FBSystem().Scene.Constraints

for lCon in lConstraints:
    if lCon.Name == 'Aim':
        lCon.FBDelete()
```

The ‘FBConstraint’ Class

- This is the base class for constraints
- This class handles properties common to all constraints.

Constraint Properties

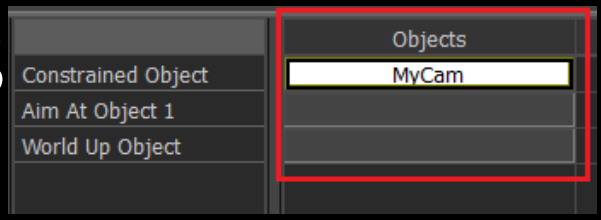
- Some of the properties are directly exposed in the class FBConstraint
- Some of the properties you need to look for using PropertyList.Find

- You cannot access the ‘Zero’ button/functionality in Python it is not exposed at this point.

Reference Groups

- Reference Groups are the categories where objects involved in a constraint can be found
- There are several functions for managing reference groups in the class FBConstraint
- A reference group is identified by its index or name, it can have more than one object.

Reference Groups



- *ReferenceGroupGetCount()* returns the total number of reference groups, in the above screen shot the reference group count is 3.
- *ReferenceGroupGetName()* returns the name of a group given an index, in the above screen shot the reference group names are Constrained Object, Aim At Object and World Up Object.
- *ReferenceAdd()* adds an object to the group represented by the given index. ReferenceRemove() removes the object from the specified group, in the above screen shot, Constrained Object is set to MyCam.
- *ReferenceGet()* obtains an object at the given index in the specified group index, in the above screen shot this would return the object MyCam.
- You can also create new groups with *ReferenceGroupAdd()*, where the group name and its maximum object count are specified.

Autodesk®

What is a Relation?

- A constraint you create using a graphical interface like connect-the-dots.
- Relations constraints come with mathematical operators that you can use as building blocks to create very specific actions for your models. These building blocks are called Operators.
- FBConstraintRelation class

Working with Relation

- Finding existing ‘Relation’ constraint in Python
- Creating ‘Relation’ constraint

Components that make up a Relation Constraint

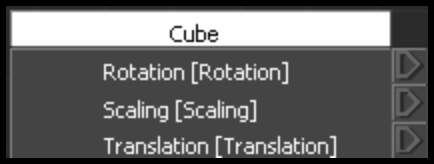
- The objects used to create a Relations constraint can be broken down into four types:

1. Senders
2. Operators
3. Receivers
4. Connections

Senders

- A Sender can be an input device or a model.
- Senders are used to transmit data to operations and Receivers.
- Senders only send data.

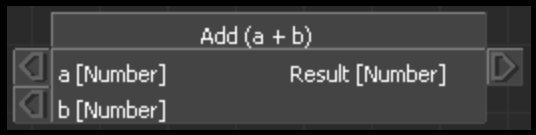
Sender =relConst.SetAsSource(cube)



Operators

- An Operator is an object that performs mathematical operations, comparisons, or conversions.
- It is placed between a Sender and a Receiver.
- Operators receive and send data.

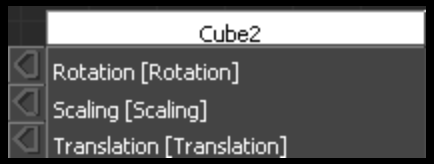
```
lBox=lCon.CreateFunctionBox('Number','
    Add (a + b)')
```



Receivers

- A Receiver can be a model or an output device.
- Receivers receive data transmitted from Operators and Senders.
- Receivers only receive data.

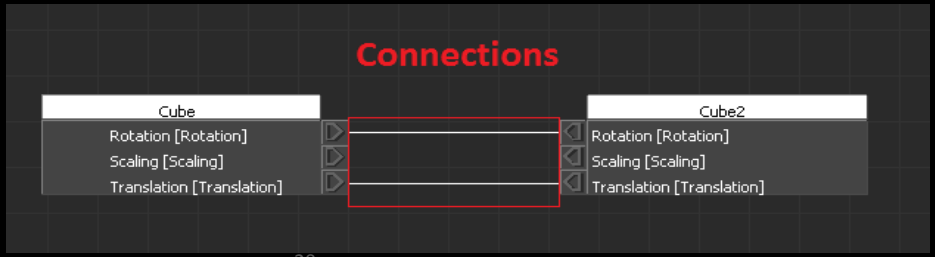
  Receiver = relConst.ConstrainObject(cube)



Connections

- This is the data that is being passed around from senders, operators and receivers.

```
lCubeOut = FindAnimationNode(
    lBoxSenderCube.AnimationNodeOutGet(), 'Rotation' )
lCamIn = FindAnimationNode(
    lBoxReceiverCam.AnimationNodeInGet(),'Translation' )
if lCubeOut and lCamIn:
    FBConnect( lCubeOut, lCamIn )
```



Triggering Scripts in Constraints

1. Drag and drop script into Viewer, choose ‘Add to scene’ (not programmable)
2. Add a script device to the scene:

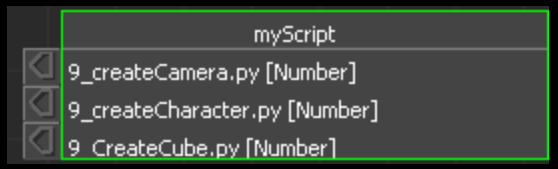
```
from pyfbsdk import *

lScript= FBCreateObject ("Boxes/Devices", "Script", "myScript")
lSystem = FBSystem()
lSystem.Scene.Devices.append(lScript)
```

3. Create a relation constraint:

```
relConst = FBConstraintRelation('MyConst')
```

4. Add the device to the Relation Constraint as a Receiver.

```
Receiver = relConst.ConstrainObject(lScript)
```



Assignment

A. Accessing the Relation Constraint
   - Find all the constraints in the scene, find out relation constraint.

B. Determining the types of objects
   - Finding the objects in the Relation Constraint, and determining if they are a sender, operator, or a receiver.

C. Determining the connections
   - Finding out what is Animation Node is connected to what Animation Node between the objects and operators.

La Fin!

Autodesk®