Haptics 3D – API
Basic Force Models

Felix G. Hamza-Lup, Ph.D
Associate Professor / Director NEWS Lab
Computer Science and Information Technology
Armstrong State University
Savannah, Georgia, USA
Outline

• What is H3D?

• Python Programming Language

• Deformable Shapes in H3D

• Elastic vs Plastic Deformation
  • H3D Clay Model

• Force models:
  • H3D Spring Model
H3D

• Haptics3D API – **is OPEN source – developed by SenseGraphics**
  • is an implementation of X3D that also has an extension for haptics capabilities
  • haptics properties can be defined in the scene by adding nodes and fields to the scene graph
  • it takes care of interface to the haptics renderer and device controls
  • implements Python scripting capabilities to manipulate objects in the scene graph
  • implements the combination of graphics, audio and haptics in one unified scene graph, written in C++
  • uses OpenGL for graphics rendering and HAPI for haptics rendering
Python Programing Language (a “crash” course)

• Python:
  • an easy to learn, powerful programming language
  • efficient high-level data structures
  • simple but effective approach to object-oriented programming

• Python interpreter is easily extended with new functions and data types implemented in C or C++

• Current version V3.5
Python (1)

• An example: multiple assignments and a while loop:

```python
# Fibonacci series: the sum of two elements defines the next
a, b = 0, 1
while b < 10:
    print(b)
    a, b = b, a+b
```

• An example: For loops

```python
# Measure some strings:
words = ['cat', 'window', 'defenestrate']
for w in words:
    print(w, len(w))
...
cat 3
window 6
defenestrate 12
```
• An example: Defining functions

```python
def fib2(n):  # return Fibonacci series up to n
    """Return a list containing the Fibonacci series up to n."""
    result = []
    a, b = 0, 1
    while a < n:
        result.append(a)  # see below
        a, b = b, a+b
    return result

f100 = fib2(100)  # call it
```

Dr. Felix Hamza-Lup - Fulbright Lecture, Bangkok University
2015
Python (3)

Example:
• using lists as Stacks (LIFO)

```python
stack = [3, 4, 5]
stack.append(6)
stack.append(7)
stack
[3, 4, 5, 6, 7]
stack.pop()
7
stack
[3, 4, 5, 6]
stack.pop()
6
stack.pop()
5
stack
[3, 4]
```
Python (4)

• Modules:
  • A module is a file containing Python definitions and statements
  • The file name is the module name with the suffix .py appended
  • Within a module, the module’s name (as a string) is available as the value of the global variable __name__

# Fibonacci numbers module (fibo.py)

def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()

def fib2(n): # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result

>>> import fibo

>>> fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
• Classes:
  • Python’s class mechanism adds classes with a minimum of new syntax and semantics
  • provide all the standard features of Object Oriented Programming:
    • the class inheritance mechanism allows multiple base classes,
    • a derived class can override any methods of its base class or classes,
    • a method can call the method of a base class with the same name

```python
class Dog:
    kind = 'canine'          # class variable shared by all

    def __init__(self, name):
        self.name = name  # instance variable

>>> d = Dog('Fido')
>>> e = Dog('Buddy')

>>> d.kind                  # shared by all dogs
'canine'

>>> e.kind                  # shared by all dogs
'canine'

>>> d.name                  # unique to d
'Fido'

>>> e.name                  # unique to e
'Buddy'
```
Deformable Shapes

• 3D Deformable shapes:
  • Consist of deformable polygonal meshes
  • Vertices in the mesh can change position
  • The rendering complexity of such model is significantly increased
  • Vertices motion is given by appropriate function => goal is to mimic real deformable object visual behavior

• 3D Haptic component:
  • Interaction between the haptic pointer (we assume 1 point of interaction) and the object must be monitored
  • Collision with the object’s surface
  • Surface penetration distance
H3D: Deformable Shape (1)

Dr. Felix Hamza-Lup - Fulbright Lecture, Bangkok University
2015
H3D: Deformable Shape (2)

<Group>
  <Collision enabled="false">
    <Transform rotation="1 0 0 -0.5">
      <DeformableShape DEF="DEFORM_SHAPE">
        <CoordinateDeformer>
          <GaussianFunction width="0.05" containerField="distanceToDepth"/>
        </CoordinateDeformer>
        <Appearance DEF="LOCAL_APP">
          <ImageTexture url="plastic_2.jpg" />
          <Material/>
          <SmoothSurface stiffness="0.1"/>
        </Appearance>
        <Rectangle2D size="0.4 0.4" DEF="HAPTIC_GEOM" solid="FALSE" containerField="hapticGeometry"/>
      </DeformableShape>
    </Transform>
  </Collision>
  <PythonScript DEF="PS" url="deform.py">
    <DeformableShape USE="DEFORM_SHAPE" containerField="references"/>
  </PythonScript>
</Group>

Dr. Felix Hamza-Lup - Fulbright Lecture, Bangkok University 2015
#deform.py
from H3DInterface import *
di = getActiveDeviceInfo()
if( di ):
    devices = di.device.getValue()
for d in devices:
    d.proxyWeighting.setValue( 0 )

columns = 31
rows = 31
size = Vec2f( 0.4, 0.4 )
coords = []
tex_coords = []
index = []

step_c = size.x / (columns-1)
step_r = size.y / (rows-1)
tc_step_c = 1.0/ (columns-1)
tc_step_r = 1.0/ (rows-1)
#deform.py
for c in range( columns ):  
    for r in range( rows ):  
        coords.append( Vec3f( step_c * c - size.x / 2, step_r * r - size.y/2, 0 ) ) 
        tex_coords.append( Vec2f( tc_step_c * c, tc_step_r * r ) )

for c in range( columns - 1 ):  
    for r in range( rows - 1 ):  
        v0 = r * columns + c 
        v1 = r * columns + c+1 
        v2 = (r+1) * columns + c+1 
        v3 = (r+1) * columns + c 
        index = index + [v0, v1, v2, v0, v2, v3 ]
H3D: Deformable Shape – Python Scripting (5)

```python
#deform.py

def deform_node, = references.getValue()

its = createX3DNodeFromString("<IndexedTriangleSet solid="FALSE" />")[0]
coord = createX3DNodeFromString("<Coordinate />")[0]
coord.point.setValue(coords)
tex_coord = createX3DNodeFromString("<TextureCoordinate />")[0]
tex_coord.point.setValue(tex_coords)

its.index.setValue(index)
its.coord.setValue(coord)
its.texCoord.setValue(tex_coord)
deform_node.geometry.setValue(its)
```

Dr. Felix Hamza-Lup - Fulbright Lecture, Bangkok University 2015
Outline

• What is H3D?

• Python Programming Language

• Deformable Shapes in H3D

• Elastic vs Plastic Deformation
  • H3D Clay Model

• Force models:
  • H3D Spring Model
Elastic vs Plastic Deformation

• Elastic Deformation:
  • Visual component returns to the resting position (rest state – not deformed)
  • Tactile component receives a force of opposite direction and proportional with the deformation until the object returns to the rest state.
    • Ex: Spring Model => $F = k \Delta x$

• Plastic Deformation:
  • Visual component keeps last position (of maximum deformation)
  • Tactile component received a resistance force until maximum deformation is reached then the force is 0
    • Ex: Clay Model => $F = h f(\Delta x)$, until max deformation is reached, then $F = 0$
H3D: Plastic Deformation - Clay Effect (1)

• DeformableShape has three fields which describe plasticity:
  • `origCoord` which contains the coordinates of the geometry before deformation, specified by the coordinates of DeformableShape's geometry,
  • `restingCoord` which contains the final coordinates of the geometry after deformation,
  • `deformedCoord` which are the coordinates of the geometry as the deformation occurs.
H3D: Plastic Deformation - Clay Effect (2)

- A non-plastic deformation would ensure that the values of restingCoord are always the same as that of origCoord, while in a full plastic deformation, restingCoord is always the same as deformedCoord.

- The extent to which the restingCoord falls back to the origCoord is adjusted with plasticity value between 0 and 1.
H3D: Plastic Deformation - Clay Effect (3)

<Scene>
  <GlobalSettings>
    <HapticsOptions maxDistance="0.1" useBoundTree="false" />
  </GlobalSettings>
  <Collision enabled="false">
    <DeformableShape DEF="D">
      <CoordinateDeformer plasticity="0.3">
        <GaussianFunction containerField="distanceToDepth" center="0" amplitude="1" width="0.012"/>
      </CoordinateDeformer>
      <Appearance>
        <Material diffuseColor="0.543 0.273 0.121"/>
        <FrictionalSurface stiffness="0.5" staticFriction="0.7" dynamicFriction="0.5"/>
      </Appearance>
    </DeformableShape>
  </Collision>
  <PythonScript moduleName="CUBE" url="ITSCube.py" />
  <PythonScript url="script.py"/>
    <DeformableShape USE="D" containerField="references" />
  </PythonScript>
</Scene>
# script.py
from H3DInterface import *
import CUBE

if getActiveDeviceInfo():
    for h in getActiveDeviceInfo().device.getValue():
        h.proxyWeighting.setValue( 0 )

d, = references.getValue()
size = Vec3f(0.15, 0.15, 0.15)

d.geometry.setValue( CUBE.createConstantITSCube(size, 25, 25) )
h = CUBE.createConstantITSCube(size, 25, 25)
d.hapticGeometry.setValue( h )
d.restingCoord.route( h.coord )
H3D: Plastic Deformation - Clay Effect (5)

See ITSCube.py
The spring effect models force by spring and is added to a scene with the SpringEffect node.
H3D: Elastic Deformation - Spring Effect (2)

The _Spring.x3d_

```xml
<Group>
  <Shape>
    <Appearance>
      <Material diffuseColor="1 0 0" transparency="0.5"/>
    </Appearance>
    <Sphere DEF="SPHERE" radius = "0.01" />
  </Shape>
  <SpringEffect DEF="SPRING" escapeDistance="0.03"/>

  <PythonScript DEF="PS" url="springs.py" />
  <ROUTE fromNode="SPRING" fromField="active" toNode="PS" toField="sphereRadius"/>
  <ROUTE fromNode="SPRING" fromField="startDistance" toNode="PS" toField="sphereRadius"/>
  <ROUTE fromNode="SPRING" fromField="escapeDistance" toNode="PS" toField="sphereRadius"/>
  <ROUTE fromNode="PS" fromField="sphereRadius" toNode="SPHERE" toField="radius"/>
</Group>
```

Dr. Felix Hamza-Lup - Fulbright Lecture, Bangkok University
2015
# springs.py
from H3DInterface import *

class SphereRadius( TypedField( SFFloat, ( SFBool, SFFloat, SFFloat ) ) ):
    def update( self, event ):  
        routes_in = self.getRoutesIn()
        active = routes_in[0].getValue()
        start_dist = routes_in[1].getValue()
        escape_dist = routes_in[2].getValue()
        if( active ):
            return escape_dist
        else:
            return start_dist

sphereRadius = SphereRadius()
Acknowledgments

• Tommy Forsell and Daniel Evestedt

http://sensegraphics.com