WEB GRAPHICS WITH X3D (PART 2)

By Raymond Pang

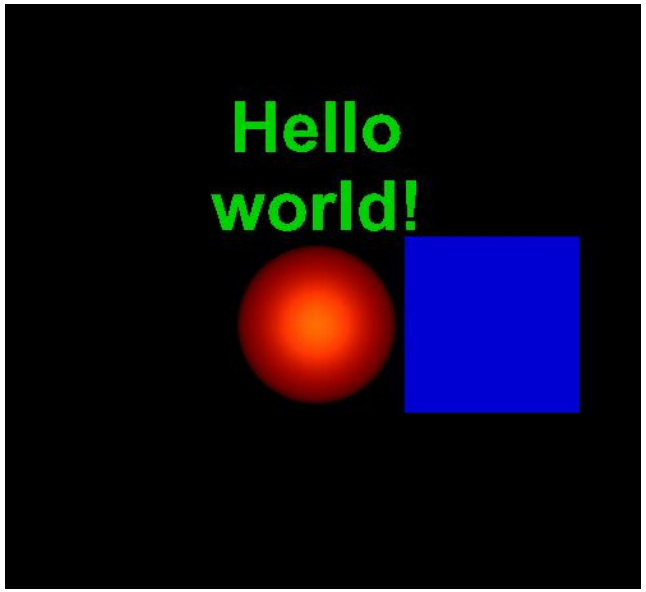
Lighting in x3D

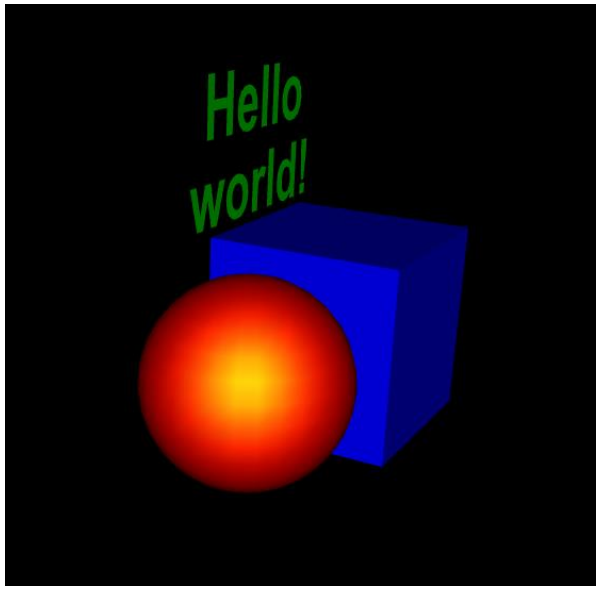
- Similar to many other rendering engine, following lights are supported:
  - Directional light
  - Point light
  - Spot light

- We can choose any of them from UI of X3DEdit
- A special kind of light is the “headlight”
  - Move together with the viewing camera
  - Define within the NavigationInfo

Lighting Node

- A head light is by default added
- Therefore, our previous example “HelloSceneGraph.x3d” is lit without adding any light
- No matter how I rotate the scene, it shines from the camera’s direction



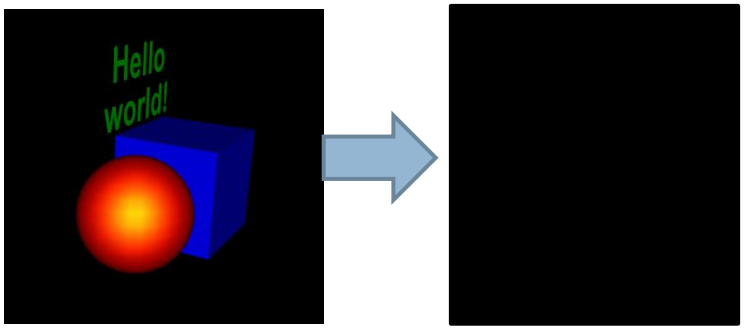


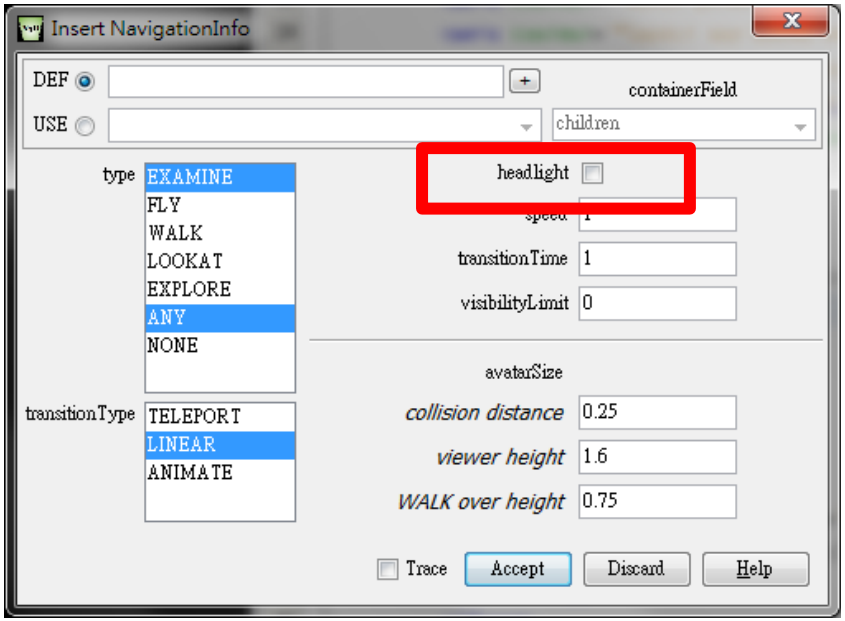
Navigation Node

- To turn off it, add a NavigationInfo Node with headlight off

  <NavigationInfo headlight='false'/>

- A completely dark scene is received

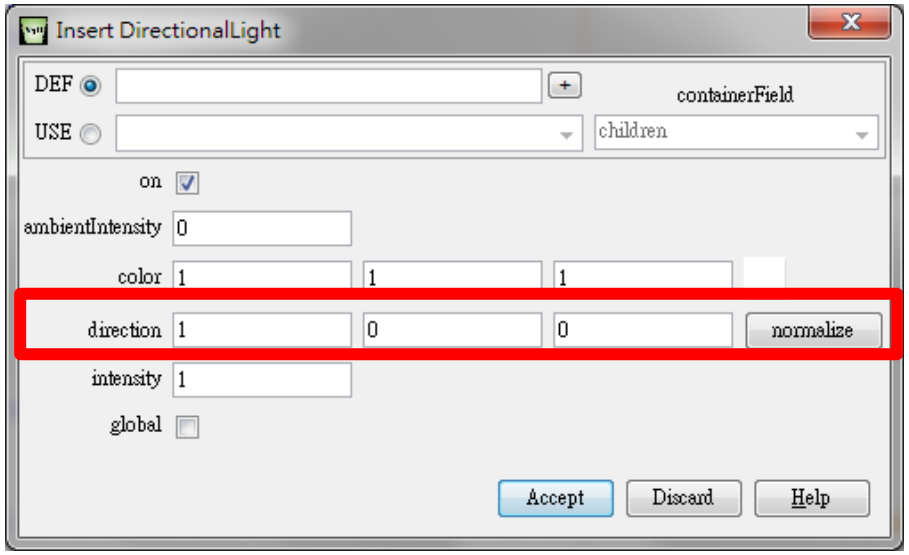


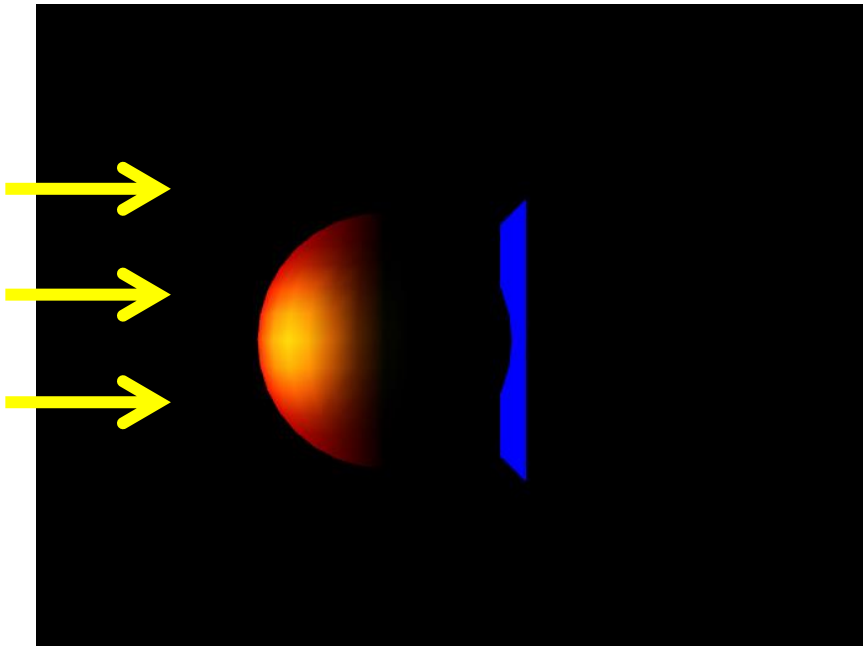


Directional Lighting

- Adding a directional light to our scene
  - Shining in +ve X direction

<DirectionalLight direction='1 0 0' />

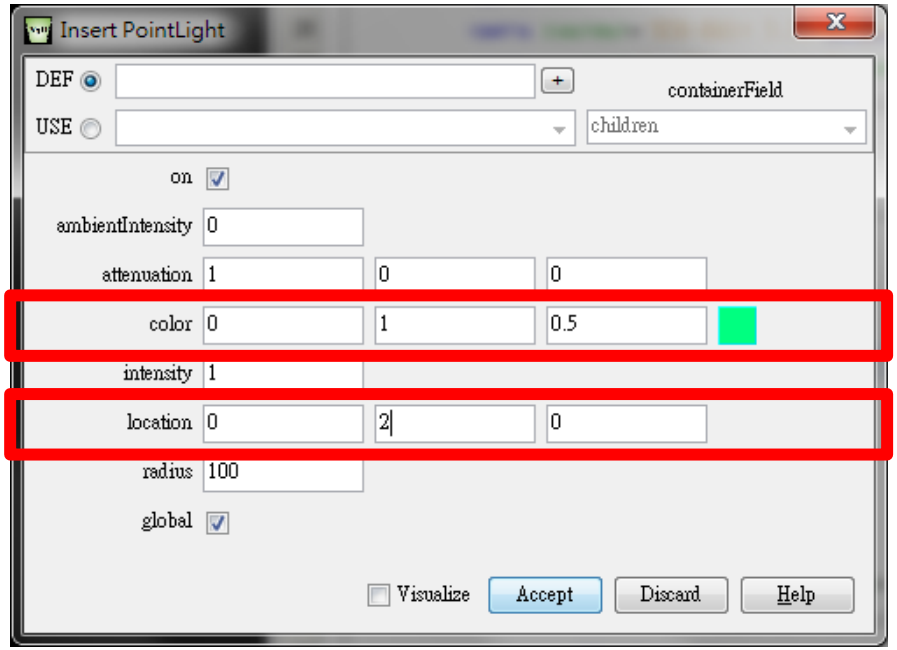


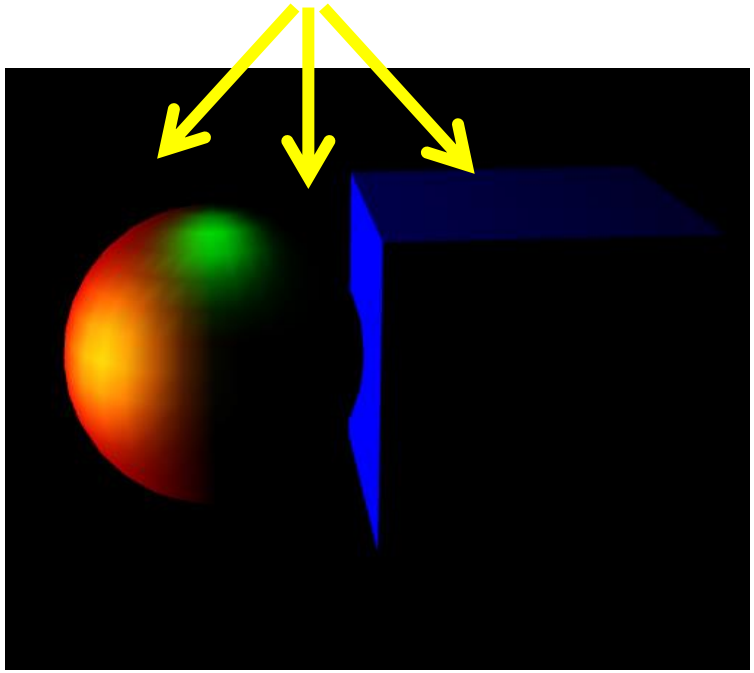


Point Light

- Add another point light on top of the scene at (0,2,0) with green color (0,1,0.5)

  <PointLight color='0 1 0.5' location='0 2 0'/>

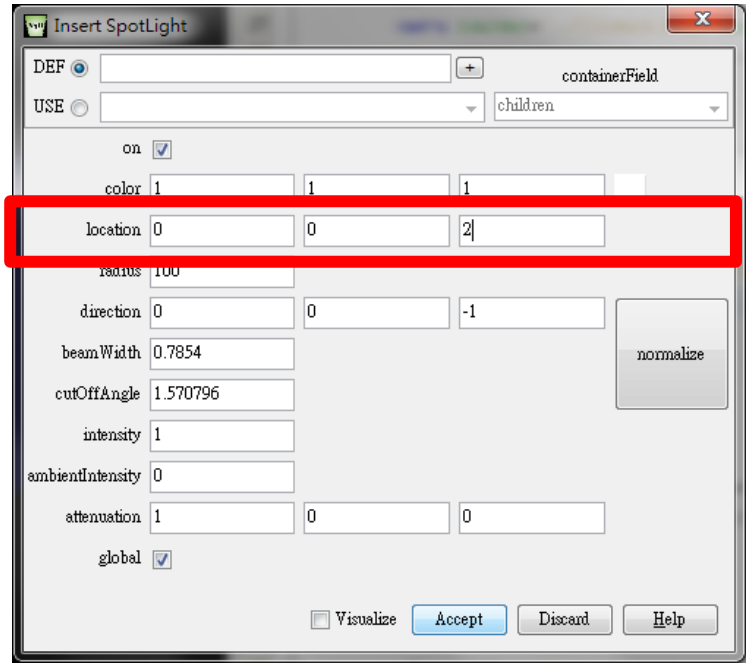


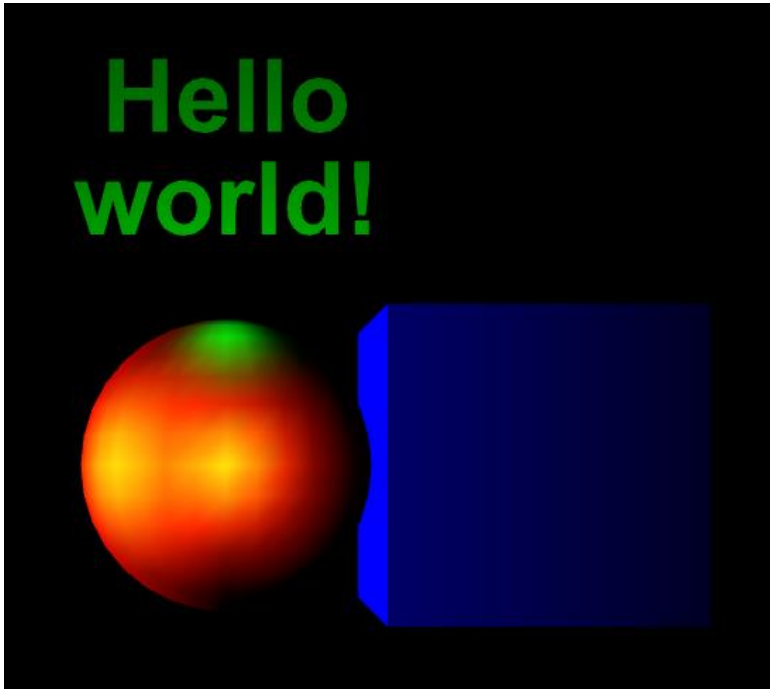


Spot Light

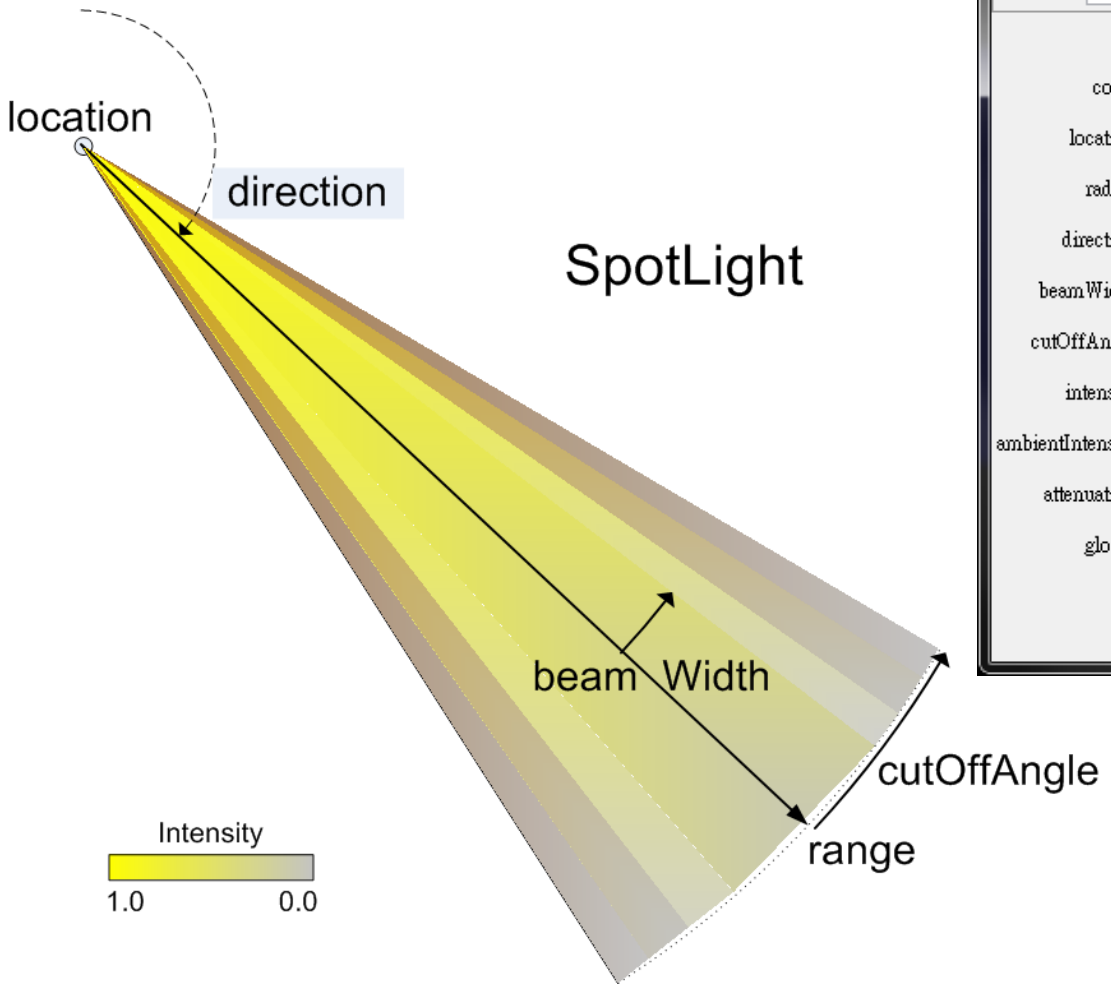
- Finally, add a spotlight at position from the camera's direction and shine in –ve Z axis

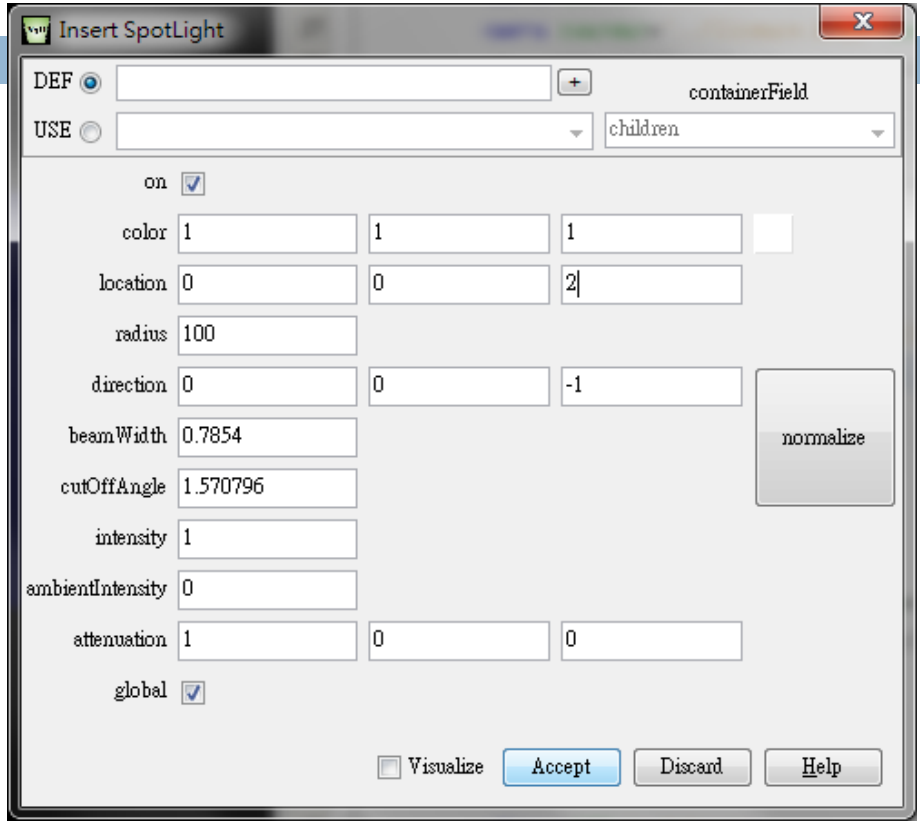
<SpotLight location='0 0 2'/>





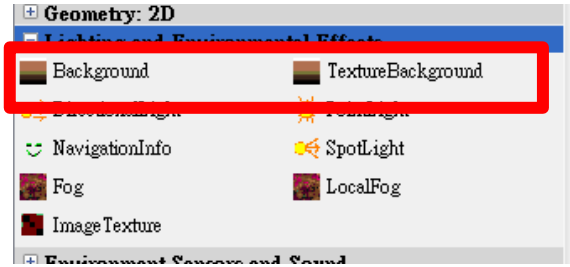
Parameters in Spot Light

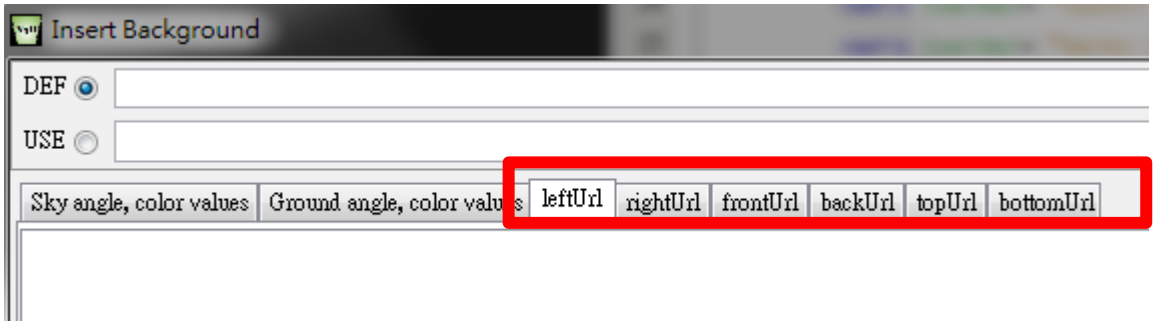




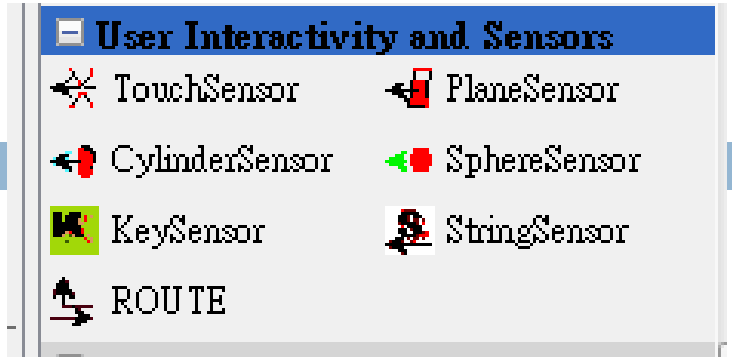
Lighting in X3D

- Maximum number of active lights: 8
  - Can use more if turned off/on appropriately
  - Matches limits of OpenGL, DirectX, GPU hardware
  - Actually this is a high number for most applications
- Background and TextureBackground can define the the skybox
  - E.g. the six images for the box





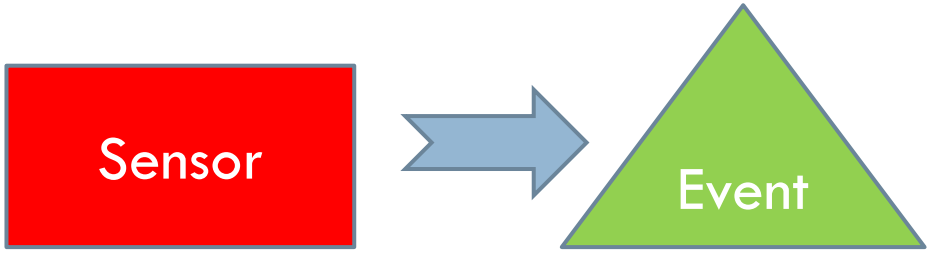
User Interaction



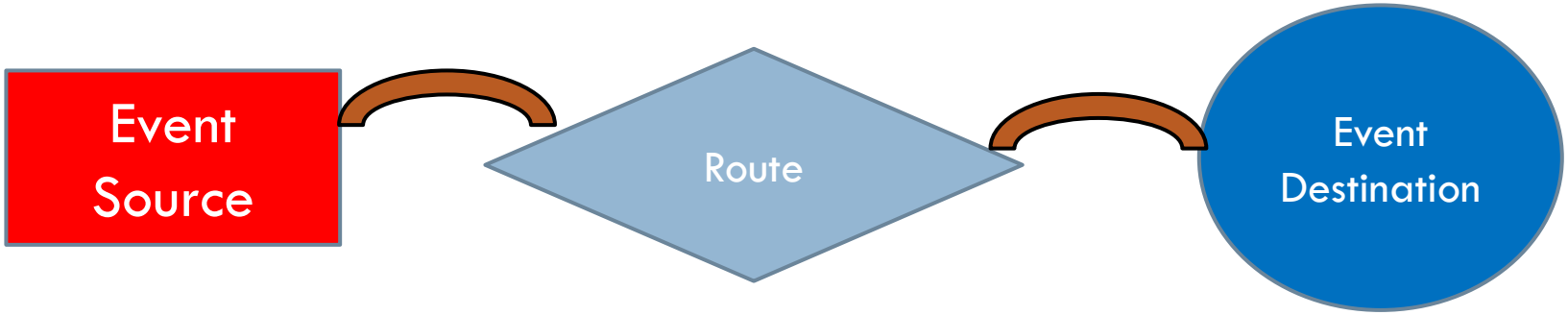
- User interactivity is initiated via sensor nodes
  - capture user inputs and are hooked up to provide appropriate responses
- Sensors detect various kinds of user interaction and produce events to ROUTE within a scene
  - Each sensor detects a certain kind of interaction, then produces one or more events
  - Authors decide how the events describing user interaction are interpreted and handled
  - This approach allows great flexibility for authors

Sensor Node, Route and Event

- Sensor is responsible to create events



- Route is to bridge between event source and event destination



TouchSensor node

- TouchSensor affects adjacent geometry, provides basic pointing-device contact interaction
  - Sends isOver true event when first pointed at
  - Sends isActive true event when selected
  - Sends isActive false event when deselected
  - Sends isOver false event when no longer pointed at
- Selection is deliberate action by user, for example
  - Mouse, touchpad, touchscreen: left-click button
  - Keyboard: <Enter> key
  - 3D wand: selection button

TouchSensor node

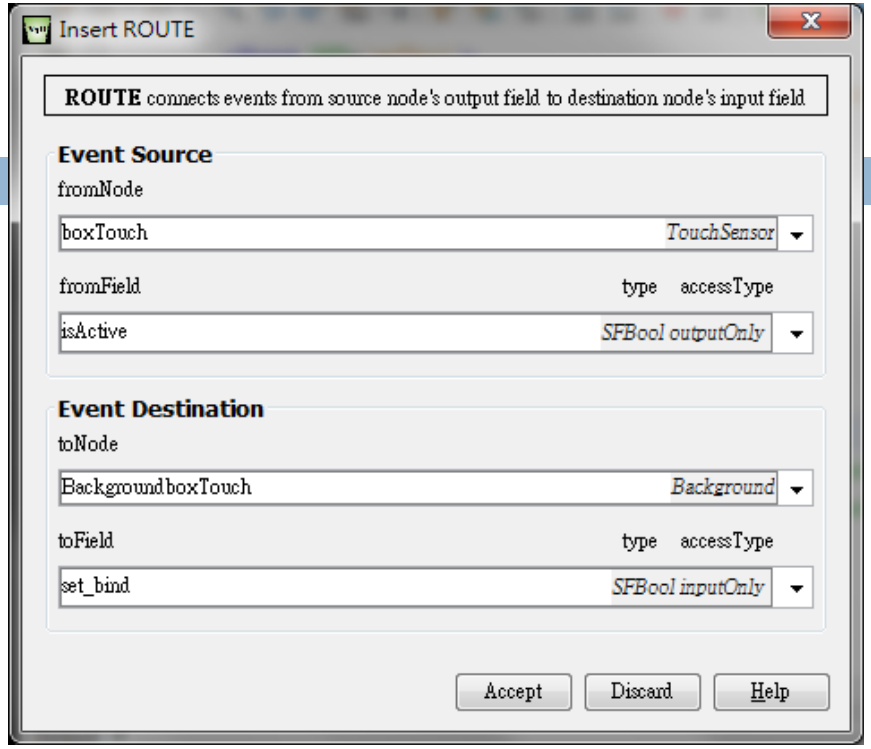
- All geometry that is a peer (or children of peers) of the TouchSensor nodes can be sensed
- Use a grouping node (Group, Transform, etc.) to isolate sensed geometry of interest

Sensor is effective within this block

```
<Transform translation='2 0 0'>
    <TouchSensor DEF='boxTouch'/>
        <Shape DEF='myBox'><Box size='2 2 2'/>
        <Appearance>  <Material diffuseColor='0 0 1'/>
.....
</Transform>
```

ROUTE node

- Event Source
  - fromNode
  - fromField
- Event Destination
  - toNode
  - toField




```
<ROUTE fromField='isActive' fromNode='boxTouch'
toField='set_bind' toNode='BackgroundboxTouch'/>
```

* We must therefore name our nodes with the DEF attributes

Field Data Type

- You may notice there is data type defined for each field in the node
- X3D is strongly typed language
  - E.g. boolean, integer, floating point or even vector
- Single value type
  - E.g. SFBool, SFFloat, SFVec2f, SFVec3f
- Multiple value type (i.e. Array)
  - E.g. MFBool, MFFloat, MFVec2f, MFVec3f

Field Data Type

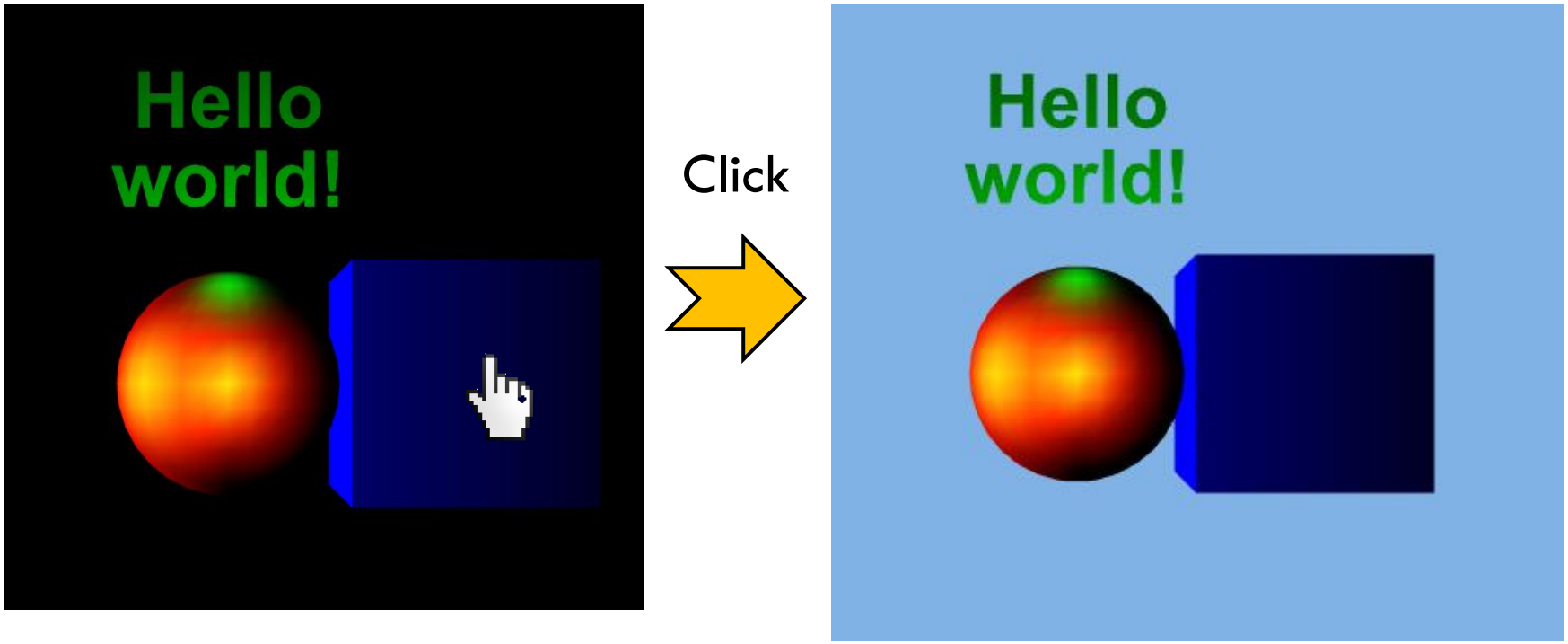
Field-type names	Description	Example values
SFBool	Single-field boolean value	true or false (X3D syntax), TRUE or FALSE (ClassicVRML syntax)
MFBool	Multiple-field boolean array	true false false true (X3D syntax), [TRUE FALSE FALSE TRUE] (ClassicVRML syntax)
SFColor	Single-field color value, red-green-blue	0 0.5 1.0
MFColor	Multiple-field color array, red-green-blue	1 0 0, 0 1 0, 0 0 1
SFColorRGBA	Single-field color value, red-green-blue alpha (opacity)	0 0.5 1.0 0.75
MFColorRGBA	Multiple-field color array, red-green-blue alpha (opacity)	1 0 0 0.25, 0 1 0 0.5, 0 0 1 0.75 (red green blue, varying opacity)
SFInt32	Single-field 32-bit integer value	0
MFInt32	Multiple-field 32-bit integer array	1 2 3 4 5
SFFloat	Single-field single-precision floating-point value	1.0
MFFloat	Multiple-field single-precision floating-point array	–1 2.0 3.14159

Field-type names	Description	Example values
SFDouble	Single-field double-precision floating-point value	2.7128
MFDouble	Multiple-field double-precision array	–1 2.0 3.14159
SFImage	Single-field image value	Contains special pixel-encoding values, see Chapter 5 for details
MFImage	Multiple-field image value	Contains special pixel-encoding values, see Chapter 5 for details
SFNode	Single-field node	<Shape/> or Shape {space}
MFNode	Multiple-field node array of peers	<Shape/><Group/><Transform/>
SFRotation	Single-field rotation value using 3-tuple axis, radian angle form	0 1 0 1.57
MFRotation	Multiple-field rotation array	0 1 0 0, 0 1 0 1.57, 0 1 0 3.14
SFString	Single-field string value	"Hello world!"
MFString	Multiple-field string array	"EXAMINE" "FLY" "WALK" "ANY"
SFTime	Single-field time value	0
MFTime	Multiple-field time array	–1 0 1 567890

Field-type names	Description	Example values
SFVec2f/SFVec2d	Single-field 2-float/2-double vector value	0 1.5
MFVec2f/MFVec2d	Multiple-field 2-float/2-double vector array	1 0, 2 2, 3 4, 5 5
SFVec3f/SFVec3d	Single-field vector value of 3-float/ 3-double values	0 1.5 2
MFVec3f/MFVec3d	Multiple-field vector array of 3-float/ 3-double values	10 20 30, 4.4 –5.5 6.6

Example

- Clicking a shape to change background color



Structure of the X3D

Four different parts are involved

1. Defining two different background setting: DefaultBackground and BackgroundboxTouch
2. Selectable shape/geometry (i.e. the blue box), with TouchSensor
3. Display shape/geometry (i.e. there sphere and text), no sensor
4. ROUTE connections

```
<Background DEF='DefaultBackground' skyColor='0 0 0' transparency='0'/>
<!-- BackgroundboxTouch ROUTE:  [from BackgroundboxTouch to set_bind ] -->
<Background DEF='BackgroundboxTouch' skyColor='0.5 0.7 0.9'
transparency='0'/>
<Transform translation='2 0 0'>
  <TouchSensor DEF='boxTouch'/>
      <Shape DEF='myBox'><Box size='2 2 2'/> <Appearance>
<Material diffuseColor='0 0 1'/></Appearance></Shape>
</Transform>
<Transform DEF='TransformSphere' translation='0 0 0'>
  <Shape DEF='mySphere'>
      <Sphere radius='1'/>
      ....
<ROUTE fromField='isActive' fromNode='boxTouch' toField='set_bind'
toNode='BackgroundboxTouch'/>
```

1

2

3

4

Trigger

HelloSceneWithTouch.x3d

PlaneSensor

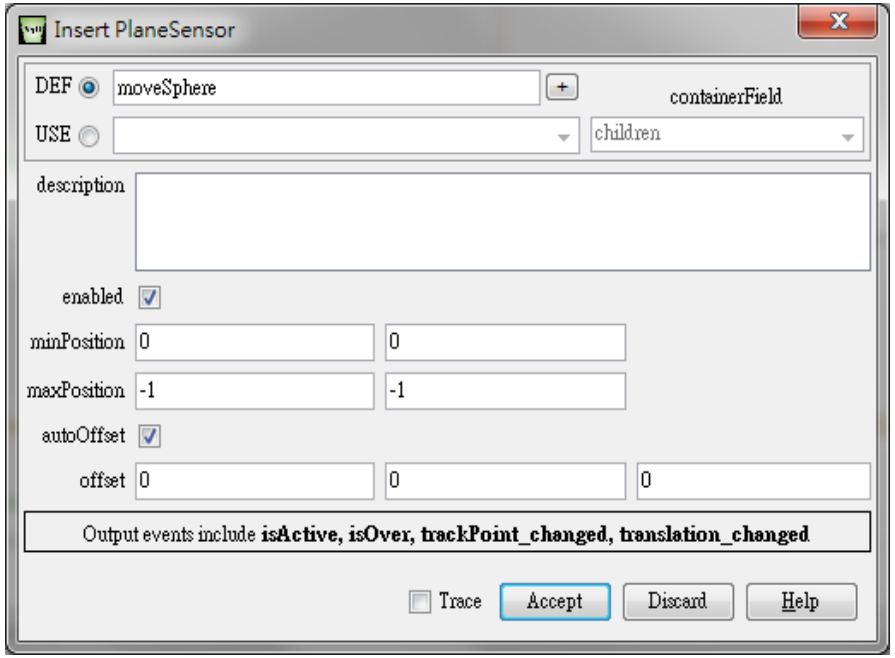
- Converts x-y dragging motion into lateral translation in plane
- Create dragging like effect to geometry
  - Motion is parallel to local z=0 plane (screen plane)
- Activated by peer geometry in scene graph
- Translation output values can follow a ROUTE connection to parent Transform translation Or connect to another SFVec3f field elsewhere

PlaneSensor Example

- We add a planesensor to the red sphere in the scene

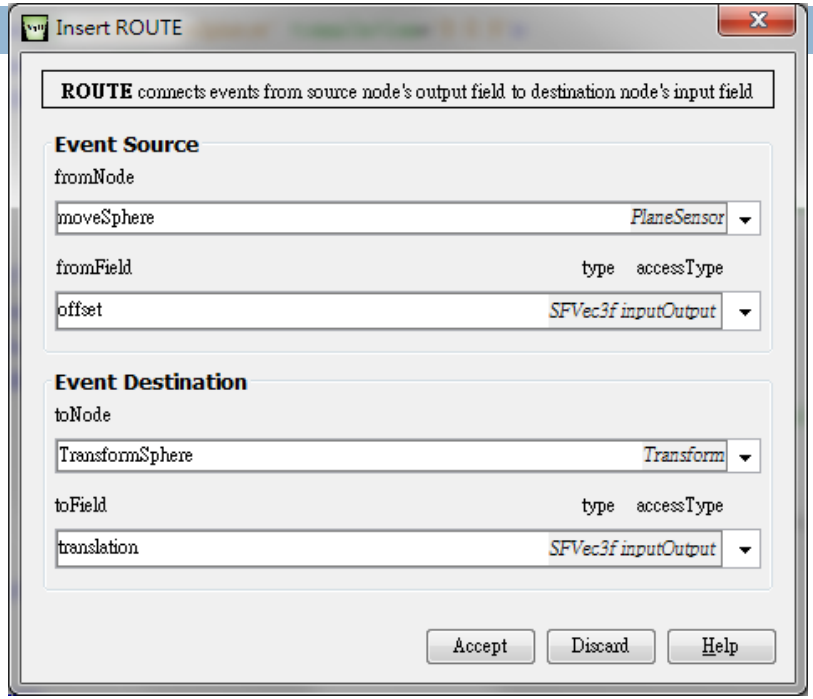
```
<Transform DEF='TransformSphere'
translation='0 0 0'>
    <PlaneSensor
DEF='moveSphere'
description="drag to move"/>
      <Shape DEF='mySphere'>
        <Sphere radius='1'/>
        <Appearance>


.....
</Transform>
```



PlaneSensor Example

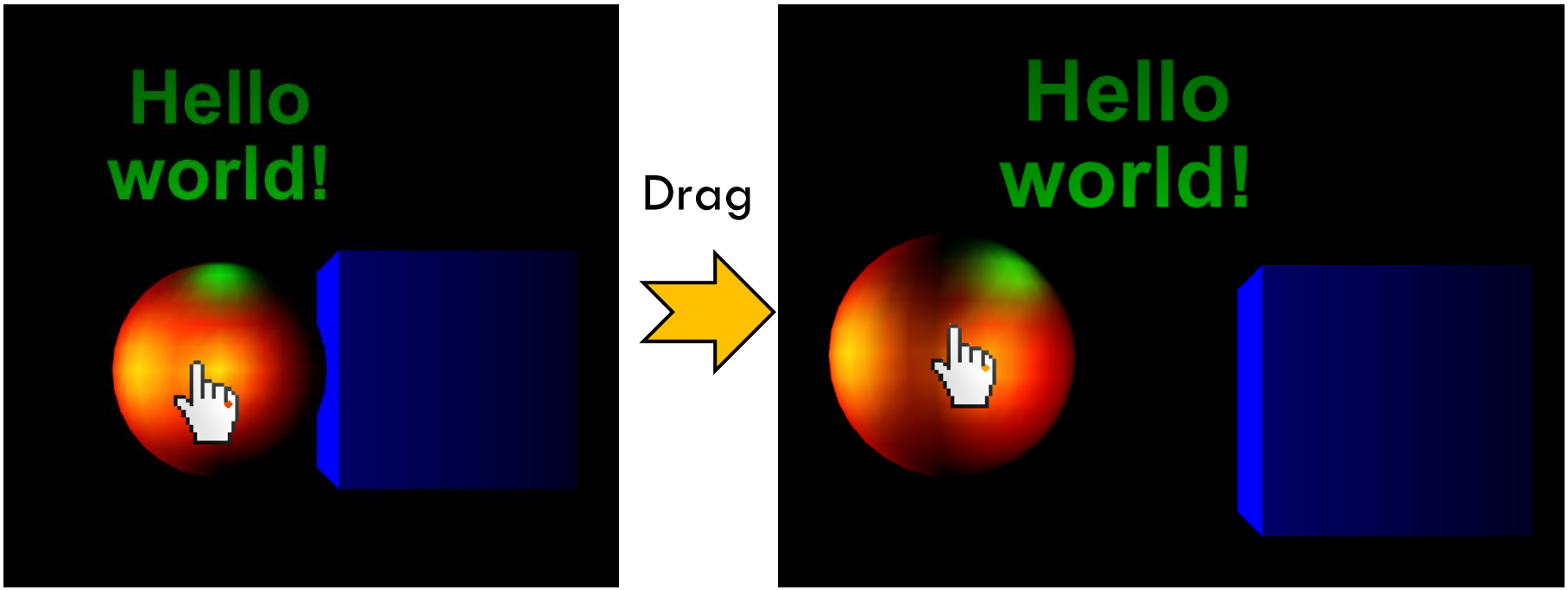
- Then, add a route from the "moveSphere" node to the "TransformSphere" node
- The field to change is "set_translation" with field values of "offset"




```
<ROUTE fromField='offset' fromNode='moveSphere'
toField='set_translation' toNode='TransformSphere'/>
```

PlaneSensor Example

- The effect is that the red ball can be dragged to where the cursor is when releasing the mouse button



HelloSceneWithPlane.x3d

KeySensor

- Receive keyboard events
- A one-character-at-a-time interface, capturing key presses from user's keyboard
  - Helpful for selecting from menu choices
  - Helpful for creating a special keyboard-driven navigation interface
- Control, alt, shift keys sent as separate events
  - As are certain special “action keys”

KeySensor

- Steps to receive key inputs

1. Add KeySensor with enable option

```
<KeySensor DEF='DefaultKeySensor' enabled='true'/>
```

2. Create a ROUTE to send event from KeySensor to a Script node (It is required in X3D)

```
<ROUTE fromNode='DefaultKeySensor' fromField='keyPress'
toNode='KeyboardProcessor' toField='keyInput'/>
```

  - Apart from keyPress, keyRelease (type SFString) can also be used here

KeySensor

3. Add the Script Node with fields and Javascript functions

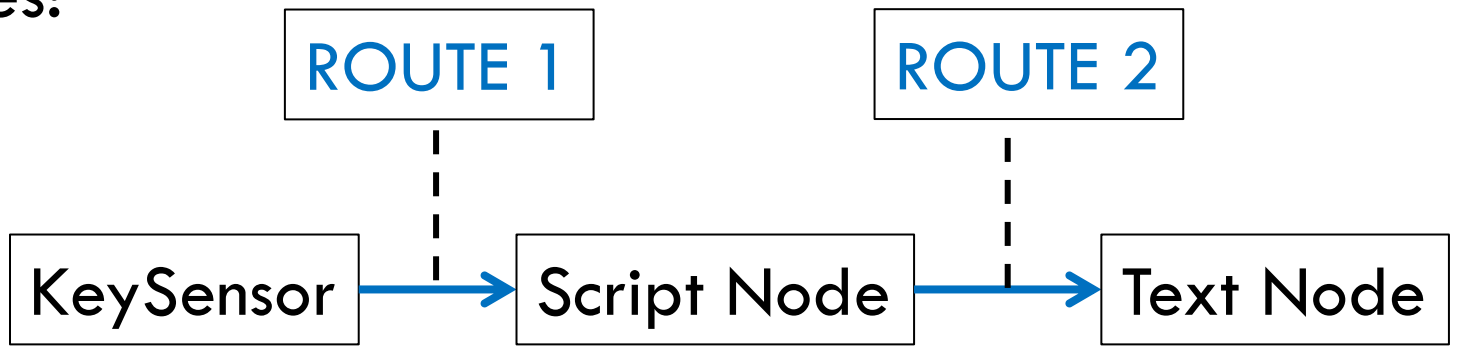
```
<Script DEF='KeyboardProcessor'>
        <field name='keyInput' type='SFString' accessType='inputOnly'/>
        <field name='keyOutput' type='MFString' accessType='outputOnly'/>
        <![CDATA[
        ecmascript:
        function keyInput (inputValue)  {
              keyOutput = new MFString (inputValue); // Only type conversion
        }
        ]]>
 </Script>
```

- Here, we have both input and output fields, but it is up to your application whether output is needed.
- And you can define how to handle the key input with javascript function.

4. Optional. Create another ROUTE to send output from the script node to the text node for display

```
<ROUTE fromNode='KeyboardProcessor' fromField='keyOutput'
toNode='KeyText' toField='string'/>
```

Therefore, in total the event is sent via the following nodes:



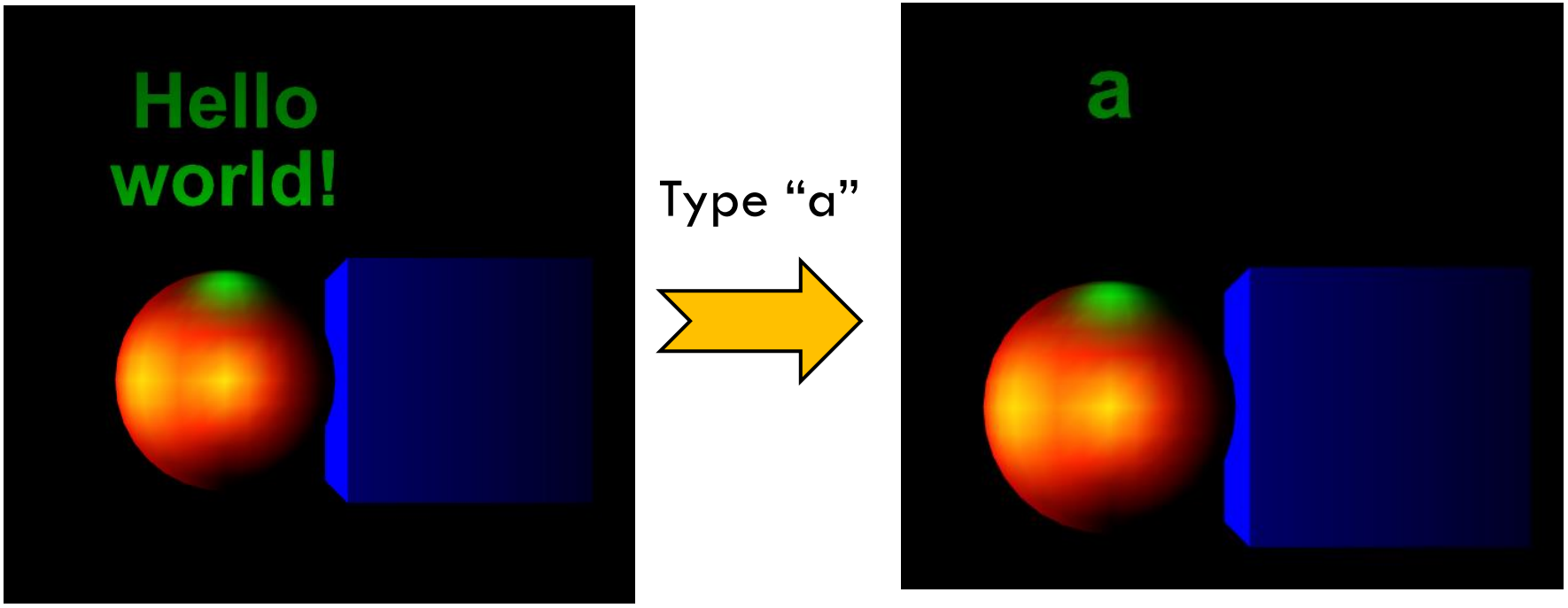
KeySensor Example

- Putting all these together in our example, to change the “Hello World” text when key input…

```
……
 <Transform translation='0 3 0'>
       <Shape DEF='myText'>
    <Text DEF='KeyText' string='"Hello" "world!"' solid='false'>
……
   <KeySensor DEF='DefaultKeySensor' enabled='true'/>
     <Script DEF='KeyboardProcessor'>
       <field name='keyInput' type='SFString' accessType='inputOnly'/>
       <field name='keyOutput' type='MFString' accessType='outputOnly'/>
       <![CDATA[  ecmascript:
       function keyInput (inputValue) {   keyOutput = new MFString (inputValue); // type conversion
       }          ]]>
      </Script>
       <ROUTE fromNode='DefaultKeySensor' fromField='keyPress' toNode='KeyboardProcessor'
toField='keyInput'/>
       <ROUTE fromNode='KeyboardProcessor' fromField='keyOutput' toNode='KeyText'
toField='string'/>
```

KeySensor Example

- The result should look like the following



HelloSceneWithKey.x3d

Other Sensors

- String Sensor
  - Works very similar to the key sensor
  - Record the keys/characters input
- Sphere Sensor
  - Similar to a plane sensor, it tracks the motion of mouse drag
  - But it turns mouse drag into rotational motions
  - Useful to apply rotation to objects (like arcball control)

Summary

- User interactivity is initiated via sensor nodes, which capture user inputs and are hooked up to provide appropriate responses
- Create routes to bridge between event source and event destination, so that certain action can be performed whenever an event occurs