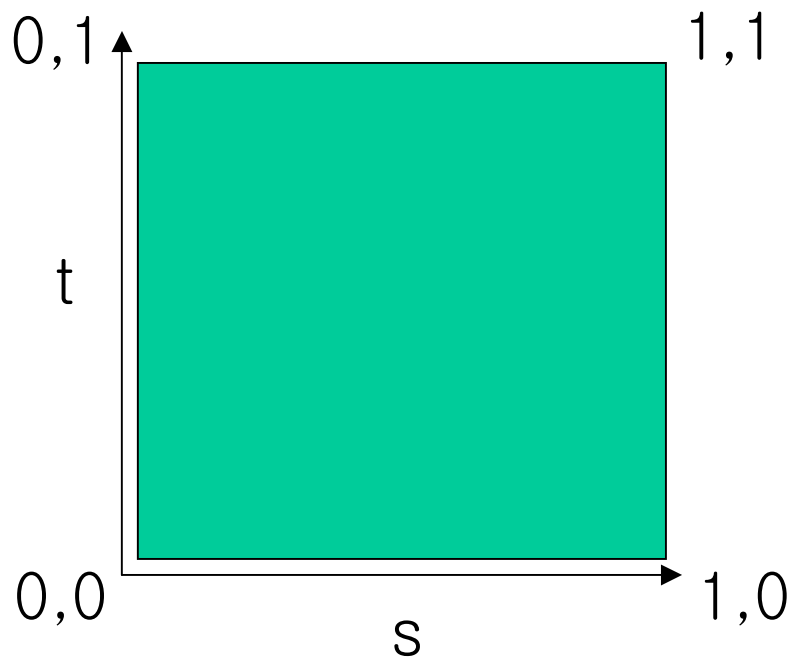
OpenInventor

Chapter 7, Textures

김진홍

What is a texture map?

- A 2D array of pixel information for a particular pattern, or texture.



Nodes Used for Texture Mapping

- SoTexture2
  - specifies a 2D texture map to be used and associated parameters for texture mapping.
- SoTextureCoordinate2
  - explicitly defines the set of 2D texture coordinates to be used by subsequent vertex shape.
- SoTextureCoordinateBinding
  - specifies how the current texture coordinates are to bound to subsequent shape nodes.

Nodes Used for Texture Mapping (cont'd)

- SoTextureCoordinatePlane
- SoTextureCoordinateEnvironment
  - allow you to use a function to map from spatial coordinates to texture coordinates.
- SoTextureCoordinateDefault
  - turns off any previous texture-coordinate function so that all following shapes use their default texture coordinates.
- SoTexture2Transform
  - defines a 2D transformation for the texture map.

An Example

```
#include <Inventor/Xt/SoXt.h>
#include <Inventor/Xt/viewers/SoXtExaminerViewer.h>
#include <Inventor/nodes/SoCube.h>
#include <Inventor/nodes/SoSeparator.h>
#include <Inventor/nodes/SoTexture2.h>

main(int, char **argv)
{
        Widget myWindow = SoXt::init(argv[0]);
        if (myWindow == NULL) exit(1);

        SoSeparator *root = new SoSeparator;
        root->ref();

        // Choose a texture
        SoTexture2 *rock = new SoTexture2;
        root->addChild(rock);
        rock->filename.setValue("brick.1.rgb");

        // Make a cube
        root->addChild(new SoCube);

        SoXtExaminerViewer *myViewer =
                new SoXtExaminerViewer(myWindow);
        myViewer->setSceneGraph(root);
        myViewer->setTitle("Default Texture Coords");
        myViewer->show();

        SoXt::show(myWindow);
        SoXt::mainLoop();
}
```

Wrapping a Texture around an Object

- The texture can either be repeated as many times as necessary to cover the face (or stretched to cover the face)
- Or the last row of pixels can be repeated to cover the rest of the face (called clamping)

How a texture affects the underlying colors

- MODULATE
  - the model can be used with any texture file. (works best on bright materials)
- DECAL
- BLEND
  - use the texture intensity to blend between the shaded color and a specified constant blend color.

Storing an Image

- you can store a texture map as an SoSFImage and then specify the image in the image field of the SoTexture2 node.
- one-component texture
- two-component texture
- three-component texture
- four-component texture

Fields of a SoTexture2 Node

- filename(SoSFName)
- image(SoSFImage)
- wrapS(SoSFEnum)
- wrapT(SoSFEnum)
  - REPEAT
  - CLAMP
- model(SoSFEnum)
  - MODULATE
  - DECAL
  - BLEND
- blendColor(SoSFColor)
  - specifies the color to blend when using the BLEND texture model.

Using the default texture mapping

- SoSphere
- SoCube
- SoCylinder
- SoCone
- SoNurbsSurface
- SoText3

Specifying Texture Coordinates Explicitly

```
...
SoTexture2 *brick = new SoTexture2;
root->addChild(brick);
brick->filename.setValue("brick.1.rgb");

SoCoordinate3 *coord = new SoCoordinate3;
root->point.set1Value(0, SbVec3f(-3, -3, 0));
root->point.set1Value(1, SbVec3f(3, -3, 0));
root->point.set1Value(2, SbVec3f(3, 3, 0));
root->point.set1Value(3, SbVec3f(-3, 3, 0));

SoNormal *normal = new SoNormal;
root->addChild(normal);
normal->vector.set1Value(0, SbVec3f(0,0,1));

SoTextureCoordinate2 *texCoord = new SoTextureCoordinate2;
root->addChild(texCoord);
texCoord->point.set1Value(0, SbVec2f(0, 0));
texCoord->point.set1Value(1, SbVec2f(1, 0));
texCoord->point.set1Value(2, SbVec2f(1, 1));
texCoord->point.set1Value(3, SbVec2f(0, 1));
...
```

SoTextureCoordinateEnvironment

- subsequent object should reflect their environment.
- environment maps are accurate only if the camera does not move relative to the environment being reflected.