CompSci 372 – Tutorial

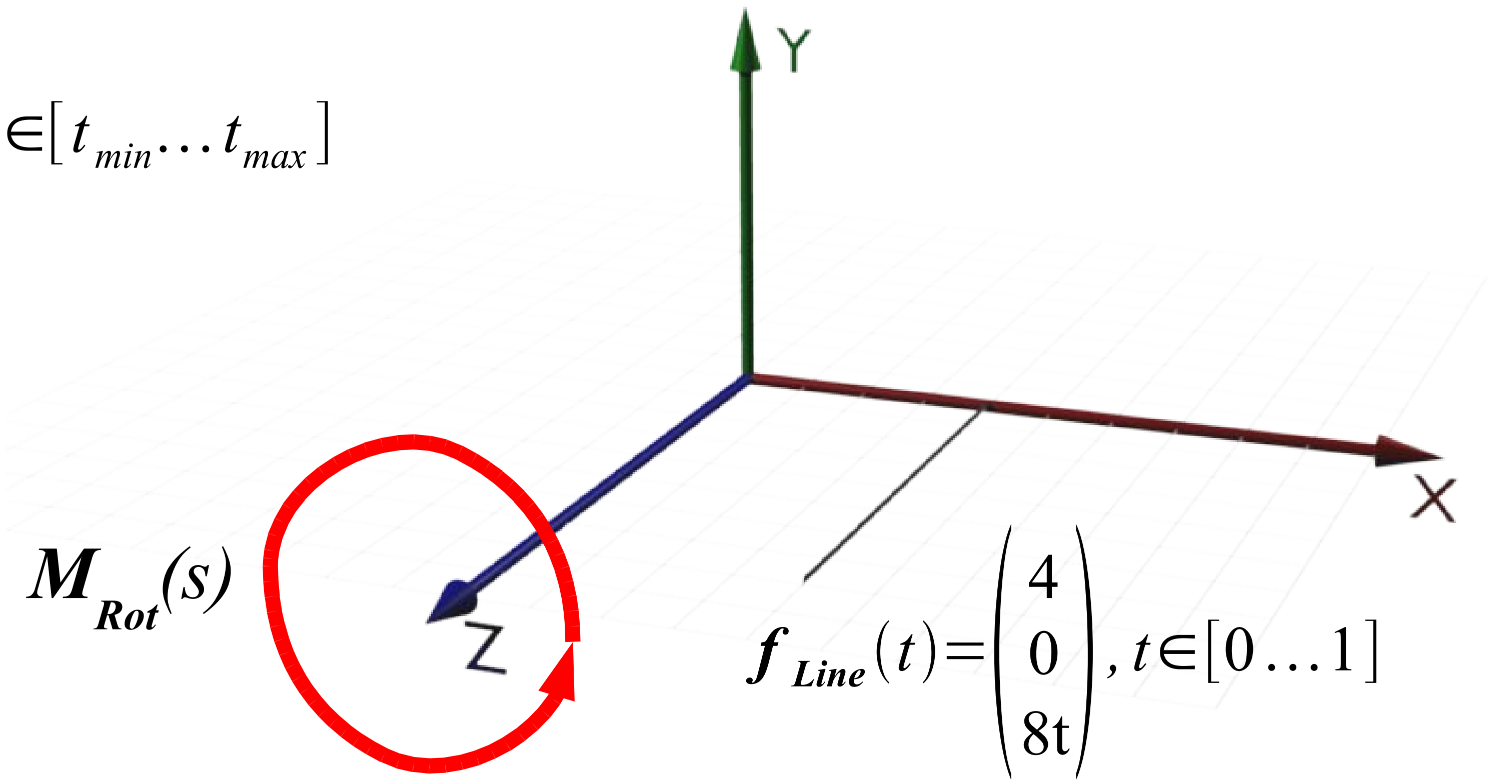
Part 6

Modelling and Texturing

Surfaces of Revolution

- General pseudo-2D Formula:

$$\boldsymbol{f}(t)=\begin{pmatrix} x(t) \\ 0 \\ z(t) \end{pmatrix}, t\in[t_{min}\dots t_{max}]$$



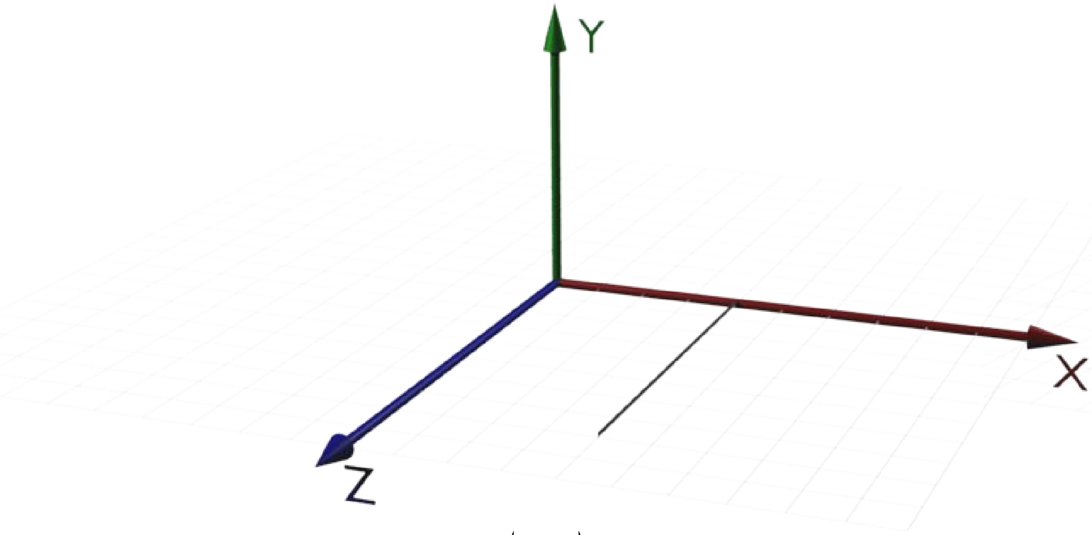
$$\boldsymbol{M}_{Rot}(s)$$

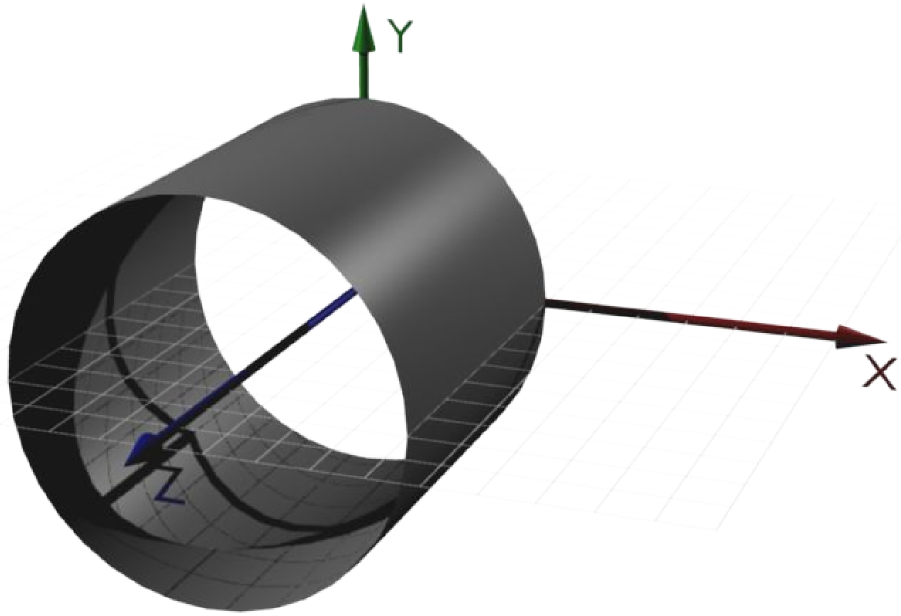
$$\boldsymbol{f}_{Line}(t)=\begin{pmatrix} 4 \\ 0 \\ 8t \end{pmatrix}, t\in[0\dots1]$$

THE UNIVERSITY OF AUCKLAND FACULTY OF SCIENCE

Surfaces of Revolution

- Straight Line → Cylinder

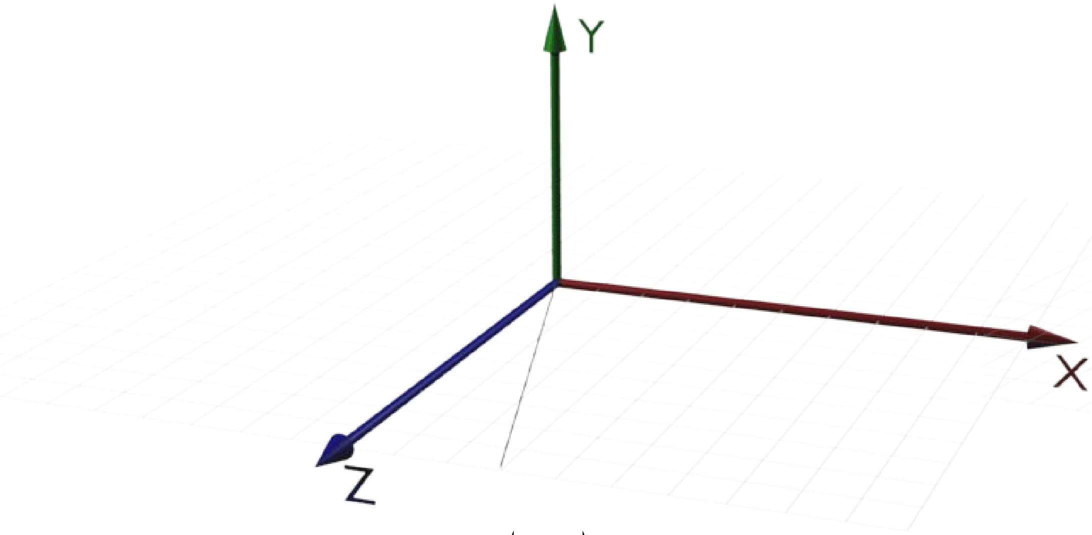


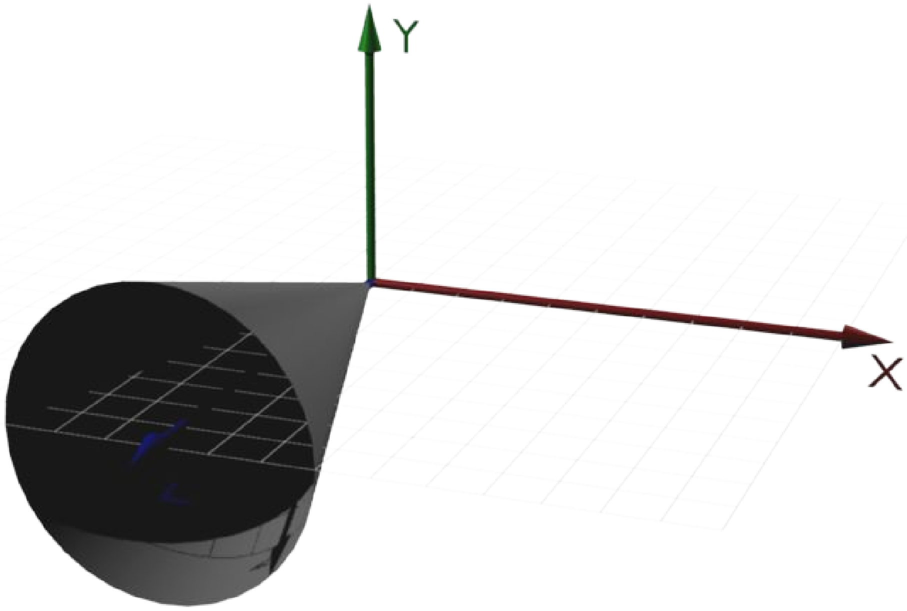


$$\boldsymbol{f}_{Line}(t)=\begin{pmatrix}4\\0\\8t\end{pmatrix}, t\in[0\ldots1]$$

Surfaces of Revolution

- Line from Origin → Cone

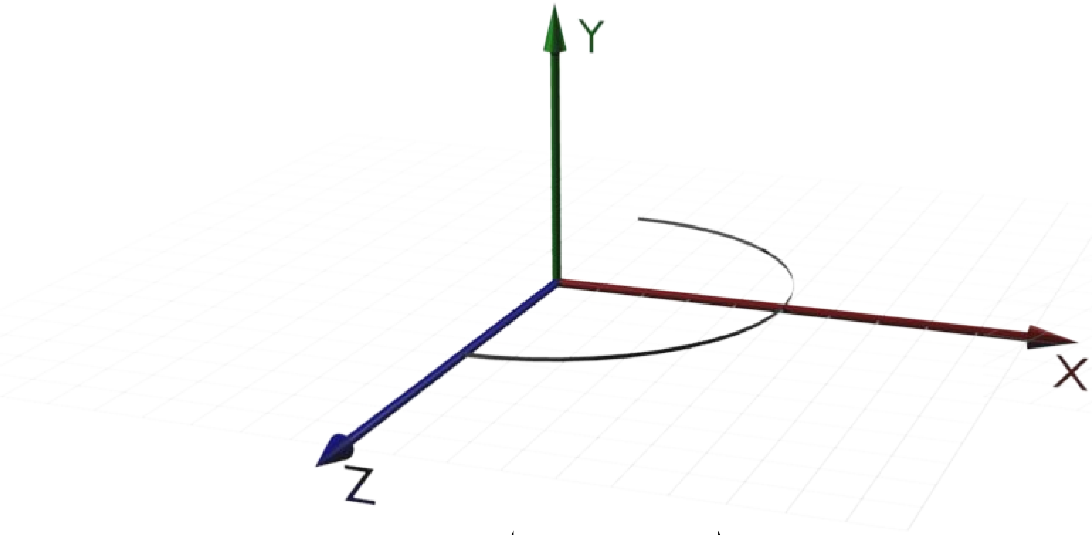


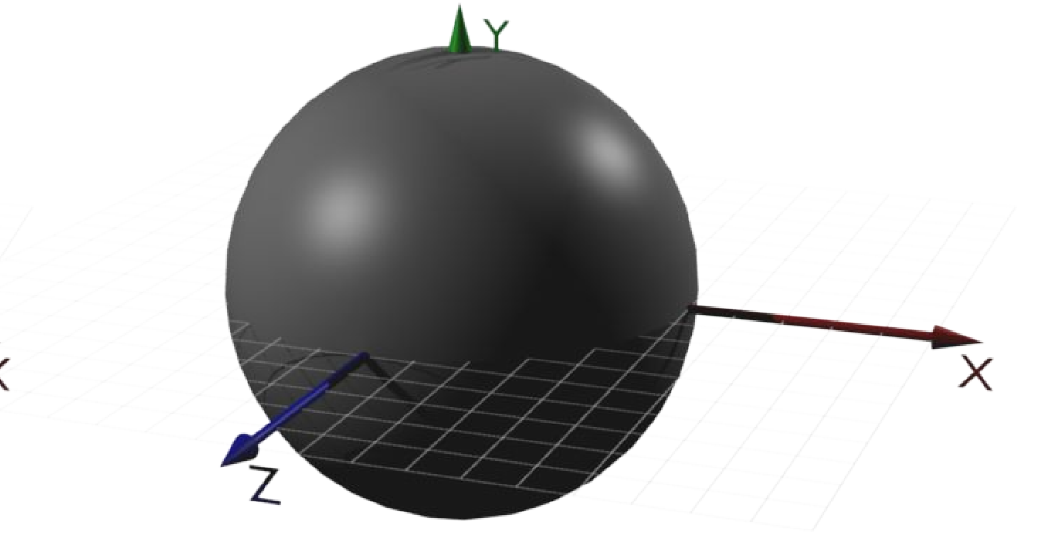


$$\boldsymbol{f}_{Line}(t)=\begin{pmatrix}4t\\0\\8t\end{pmatrix}, t\in[0\ldots1]$$

Surfaces of Revolution

- Half Circle → Sphere

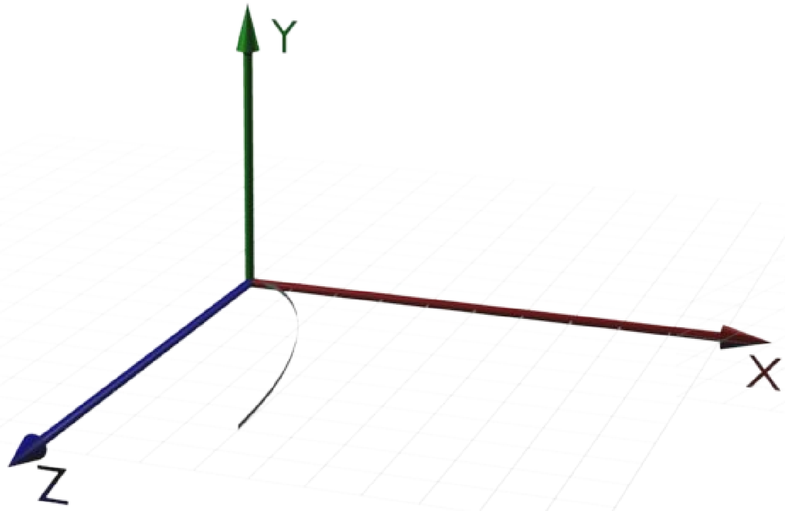


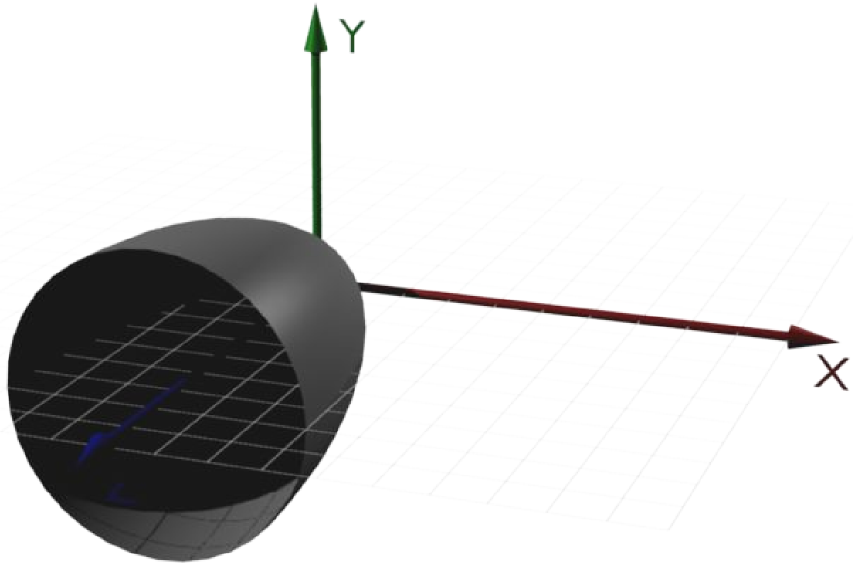


$$\boldsymbol{f}_{Line}(t)=\begin{pmatrix}5\sin(t)\\0\\5\cos(t)\end{pmatrix}, t\in[0\ldots\pi]$$

Surfaces of Revolution

- Quarter Ellipsis → Basket





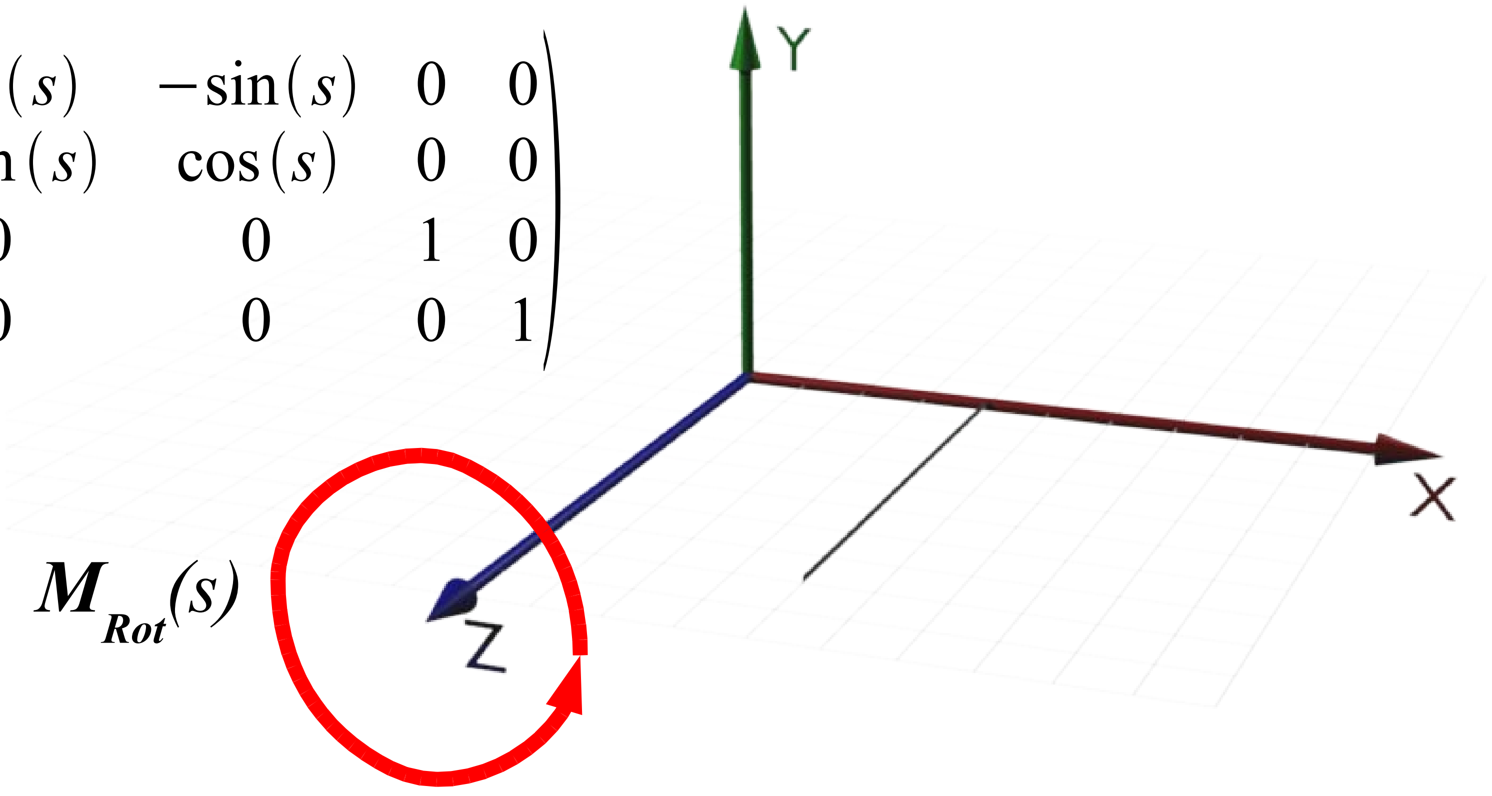
$$\boldsymbol{f}_{Line}(t)=\begin{pmatrix}3\sin(t)\\0\\8-8\cos(t)\end{pmatrix},\ t\in[0\ldots\frac{1}{2}\pi]$$

THE UNIVERSITY OF AUCKLAND
FACULTY OF SCIENCE

Surfaces of Revolution

- Rotation Matrix:
  - Rotation around Z-Axis

$$\boldsymbol{M}_{Rot}(s)=\begin{pmatrix} \cos(s) & -\sin(s) & 0 & 0 \\ +\sin(s) & \cos(s) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Surfaces of Revolution

- Result:

$$\boldsymbol{M}_{Rot}(s)\cdot\boldsymbol{f}(t)=\begin{pmatrix}\cos(s) & -\sin(s) & 0 & 0\\ +\sin(s) & \cos(s) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\cdot\begin{pmatrix}x(t)\\ 0\\ z(t)\\ 1\end{pmatrix}$$

$$SOR(s,t)=\begin{pmatrix}x(t)\cos(s)\\ x(t)\sin(s)\\ z(t)\\ 1\end{pmatrix},\quad s\in[0\ldots 2\pi],t\in[t_{min}\ldots t_{max}],$$

Compare to the formula of a parametric surface!

$$p(s,t)=\begin{pmatrix}x(s,t)\\ y(s,t)\\ z(s,t)\end{pmatrix}$$

Surfaces of Revolution

- Continuous Implementation

```
for ( float fT = 0 ... 1 )
{

  float fX = functionX(fT);
  float fZ = functionZ(fT);

  for ( float fS = 0 ... 1 )
  {

    float fA = 2 * Pi * fS;

    vertices[fT][fS][0] = fX * cos(fA);
    vertices[fT][fS][1] = fX * sin(fA);
    vertices[fT][fS][2] = fZ;
  }
}
```

THE UNIVERSITY OF AUCKLAND
FACULTY OF SCIENCE

Surfaces of Revolution

- Discrete Implementation

```
for ( int iT = 0 ; iT < MAX_T ; iT++ )
{
  float fT = (float) iT / (MAX_T - 1);
  float fX = functionX(fT);
  float fZ = functionZ(fT);

  for ( int iS = 0 ; iS < MAX_S ; iS++ )
  {
    float fS = (float) iS / (MAX_S - 1);
    float fA = 2 * Pi * fS;

    vertices[iT][iS][0] = fX * cos(fA);
    vertices[iT][iS][1] = fX * sin(fA);
    vertices[iT][iS][2] = fZ;
  }
}
```

$fT \in [0..1]$

$fS \in [0..1]$

$fA \in [0..2\pi]$

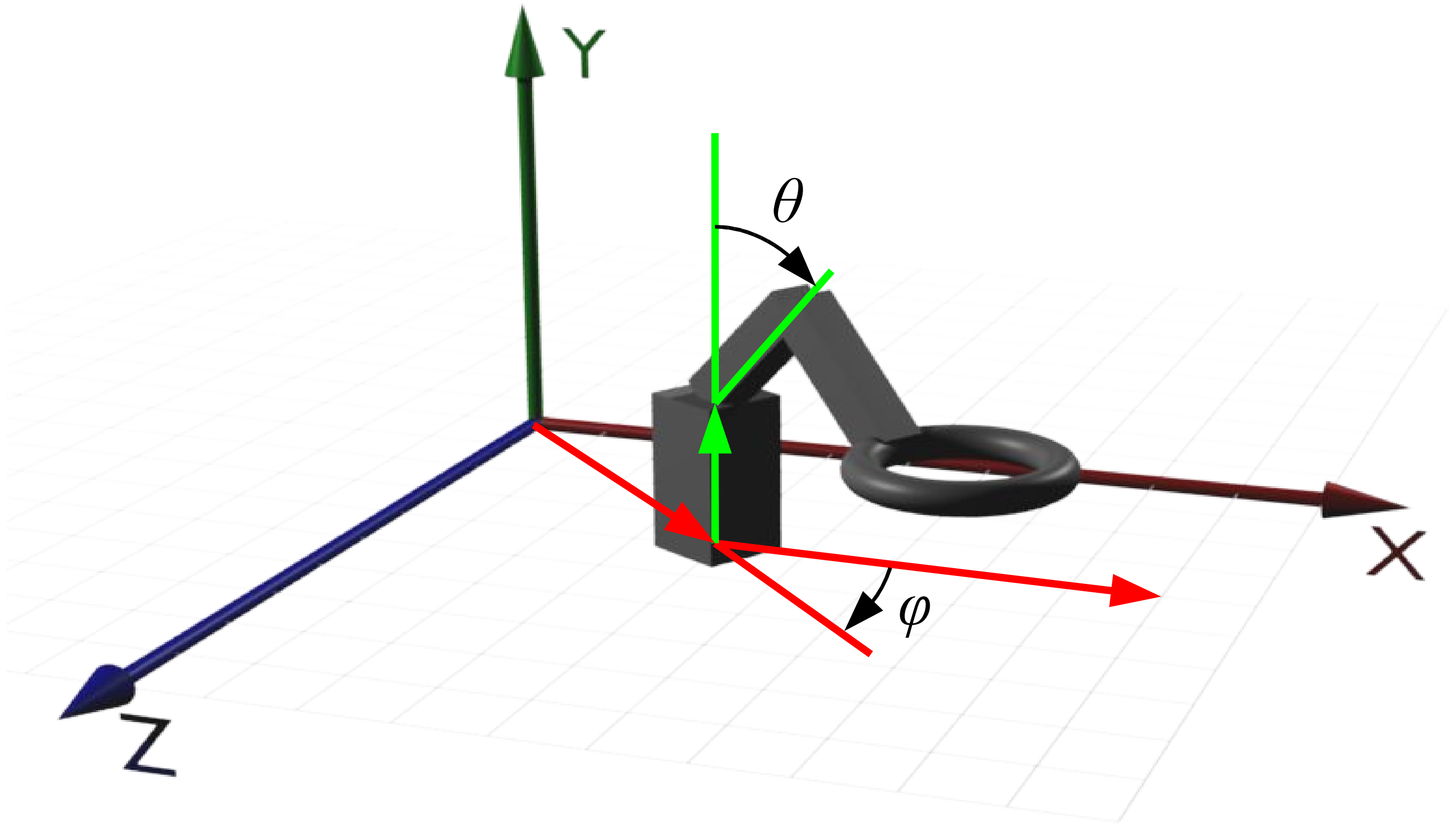
Hierarchical Modelling

- Robot Arm:
  - Transformations for structure
  - Transformations for drawing
  - Keep these separated!
  - **`glPush/PopMatrix`** is your friend

THE UNIVERSITY OF AUCKLAND
FACULTY OF SCIENCE

Hierarchical Modelling

- Robot Arm:
  - Transformations for structure



Hierarchical Modelling

```
// Prepare robot base coordinate system
glTranslated(...); // world position
glRotated(...);    // robot rotation

// Draw robot base

// How? Don't know yet!


// Prepare first arm coordinate system
glTranslated(...); // relative position second arm
glRotated(...);    // relative rotation second arm

// Draw first arm

// How? Don't know yet!


...
```

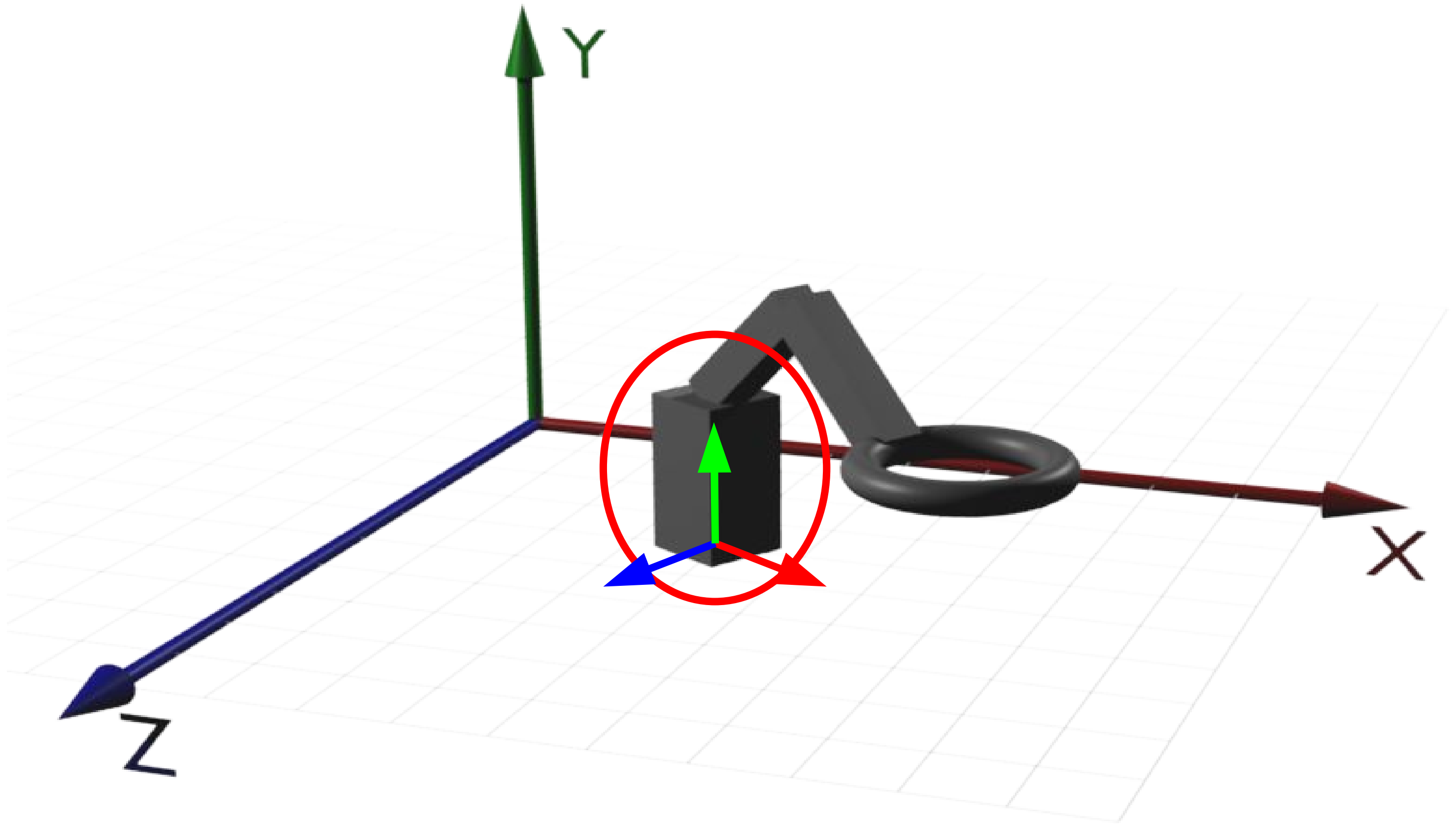
THE UNIVERSITY OF AUCKLAND
FACULTY OF SCIENCE

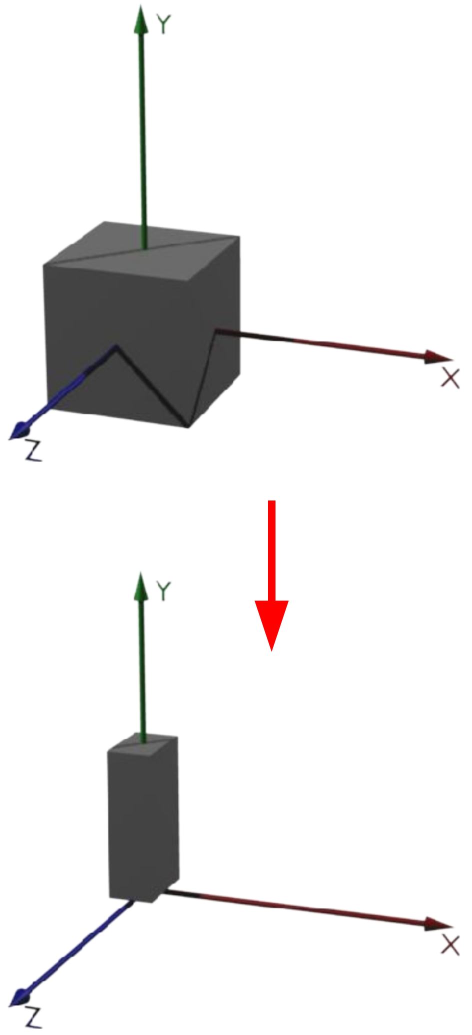
Hierarchical Modelling

```
// Prepare robot base coordinate system
glTranslated(...); // world position
glRotated(...);    // robot rotation

// Draw robot base
glPushMatrix();
  // How? Don't know yet!
glPopMatrix();

// Prepare first arm coordinate system
glTranslated(...); // relative position second arm
glRotated(...);    // relative rotation second arm

// Draw first arm
glPushMatrix();
  // How? Don't know yet!
glPopMatrix();

...
```

Hierarchical Modelling

- Robot Arm:
  - Transformations for drawing





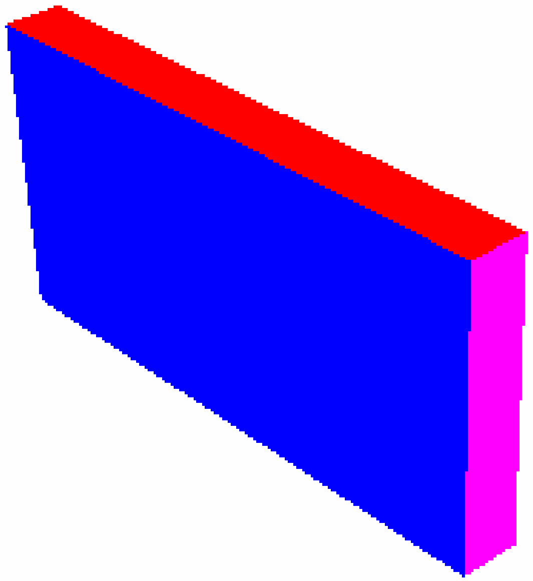
Hierarchical Modelling

```
// Prepare robot base coordinate system
glTranslated(...); // world position
glRotated(...);    // robot rotation

// Draw robot base
glPushMatrix();
  glScaled/Translated/Rotated(...);  glutSolidCube(...);
glPopMatrix();

// Prepare first arm coordinate system
glTranslated(...); // relative position second arm
glRotated(...);    // relative rotation second arm

// Draw first arm
glPushMatrix();
  glScaled/Translated/Rotated(...);  glutSolidCube(...);
glPopMatrix();


...
```

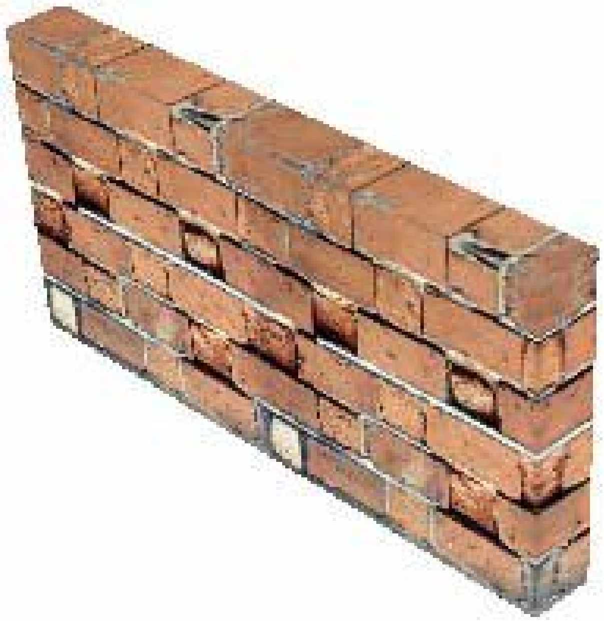
Texture Mapping

- Principle: Rubber Wallpaper & Pins

+



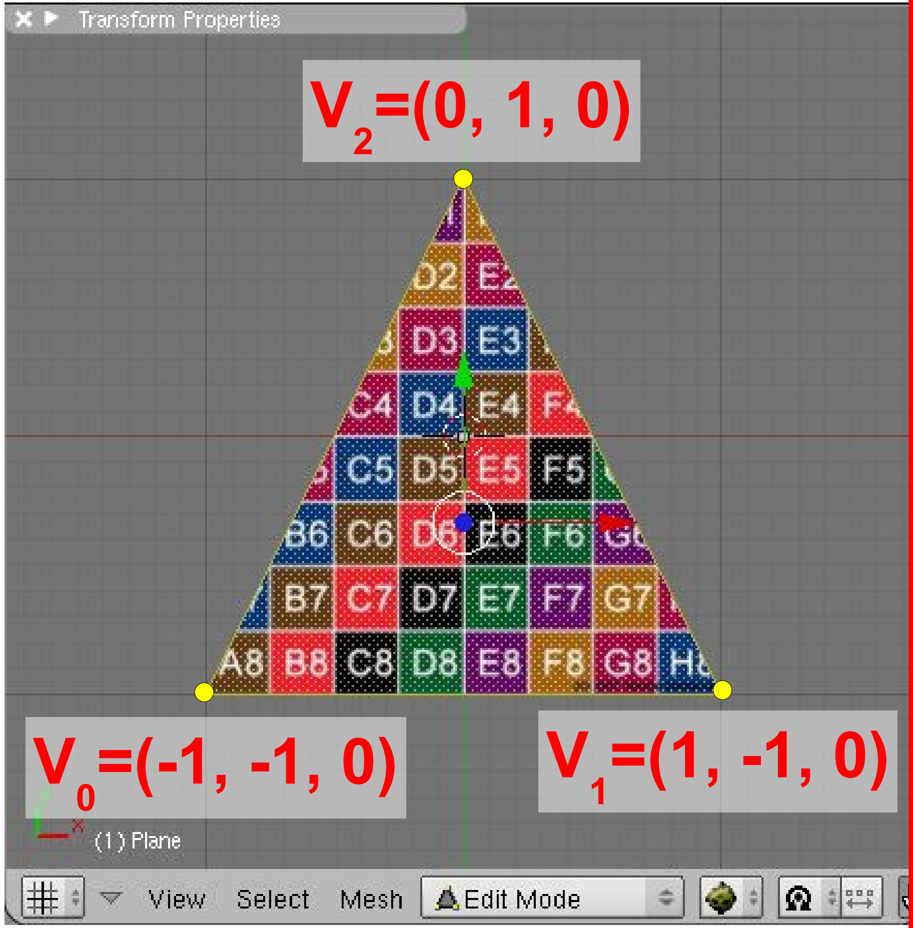
=



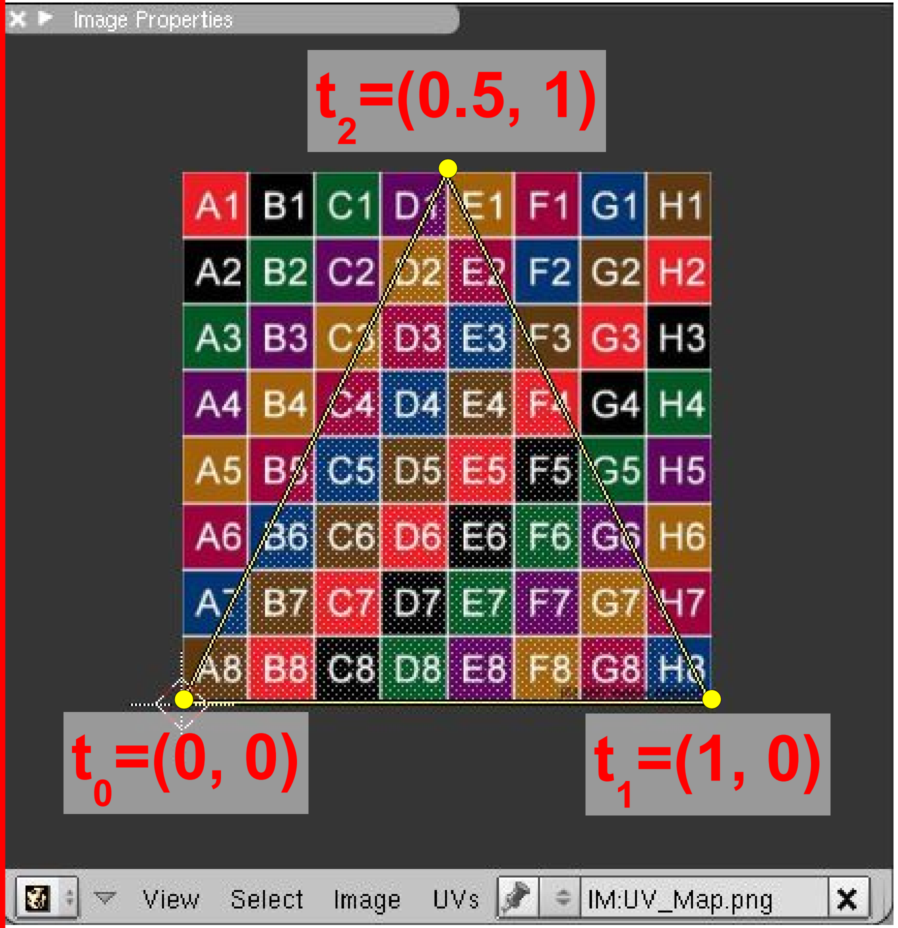
THE UNIVERSITY OF AUCKLAND
FACULTY OF SCIENCE

Texture Mapping

Vertex coordinates:
-∞...+∞, 3D



Texture coordinates:
0..1, 2D

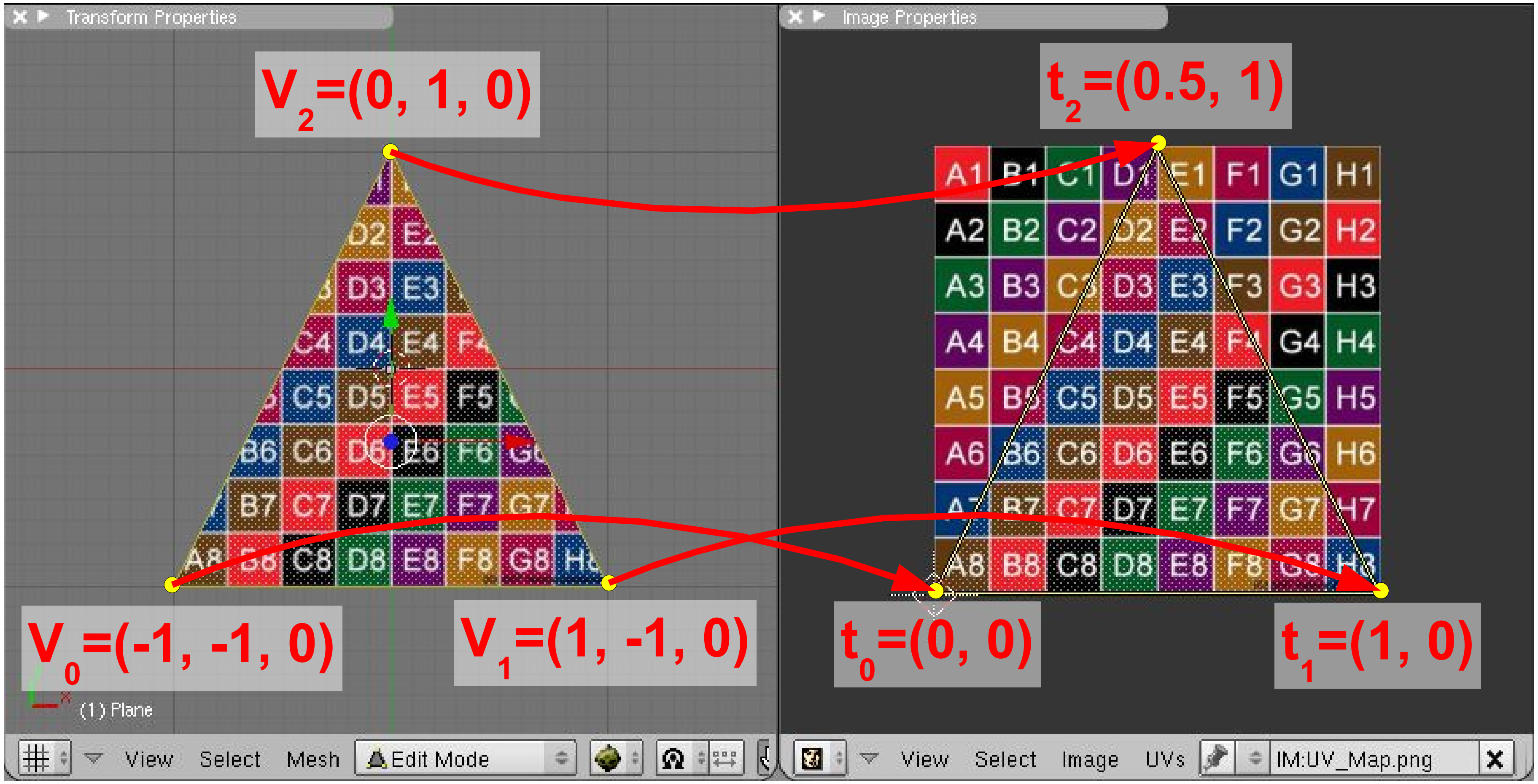


Texture Mapping

```
glTexCoord2f(0, 0);
glVertex3f(-1, -1, 0);
```

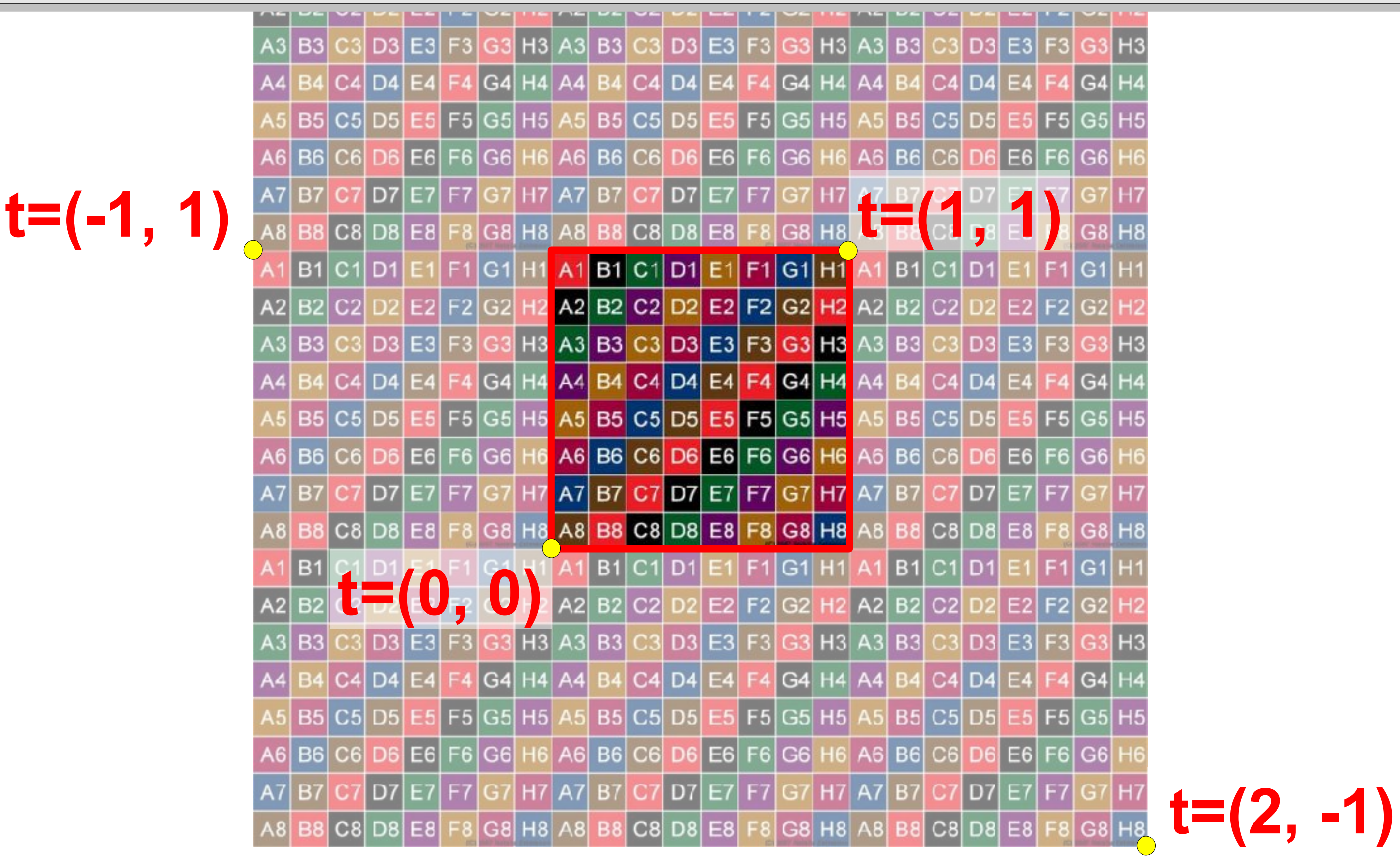
```
glTexCoord2f(1, 0);
glVertex3f(1, -1, 0);
```

```
glTexCoord2f(0.5, 1);
glVertex3f(0, 1, 0);
```



Texture Mapping

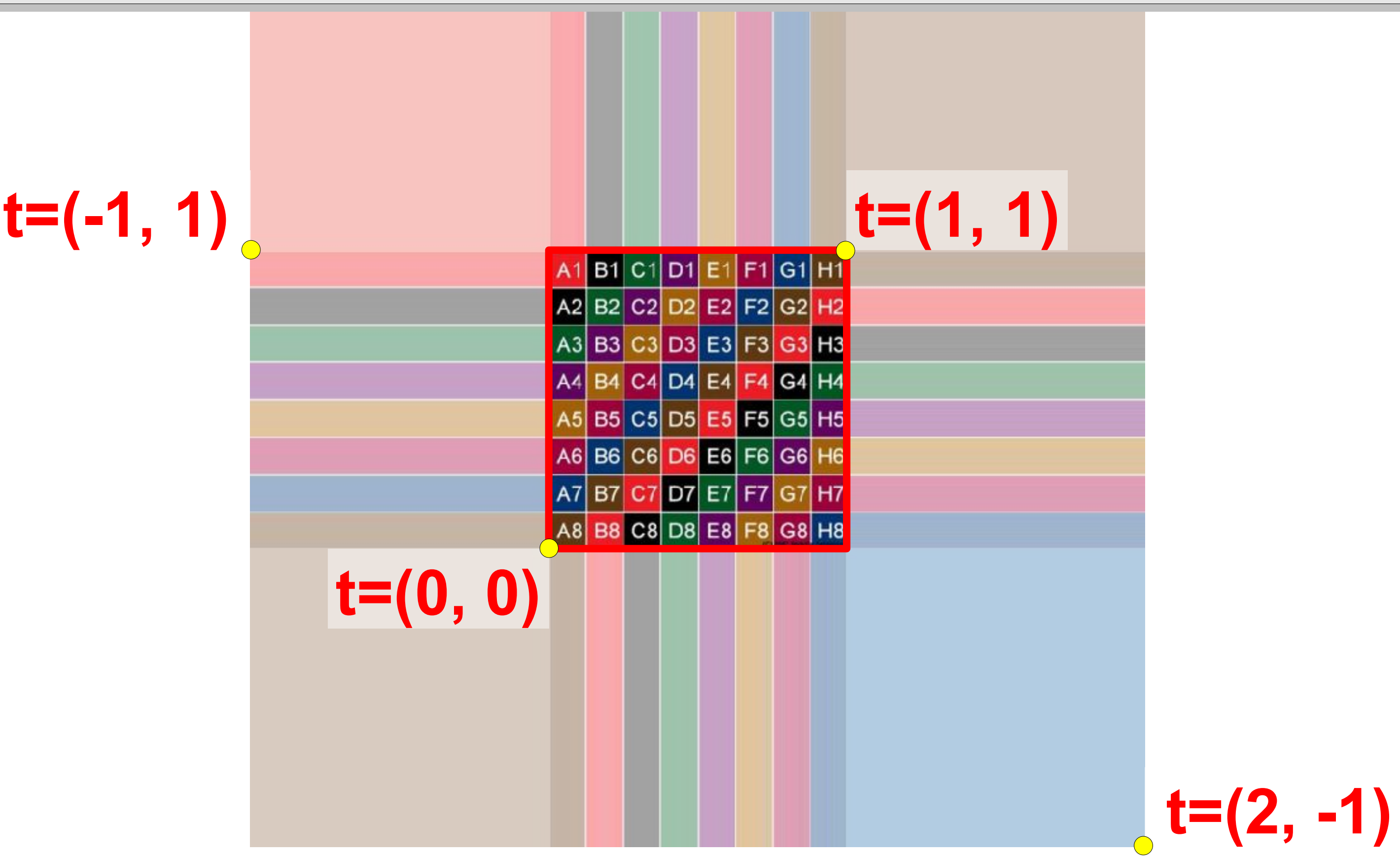
- What is beyond [0...1]?
  - Repeat (Tiling)
  - `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S/T, GL_REPEAT);`



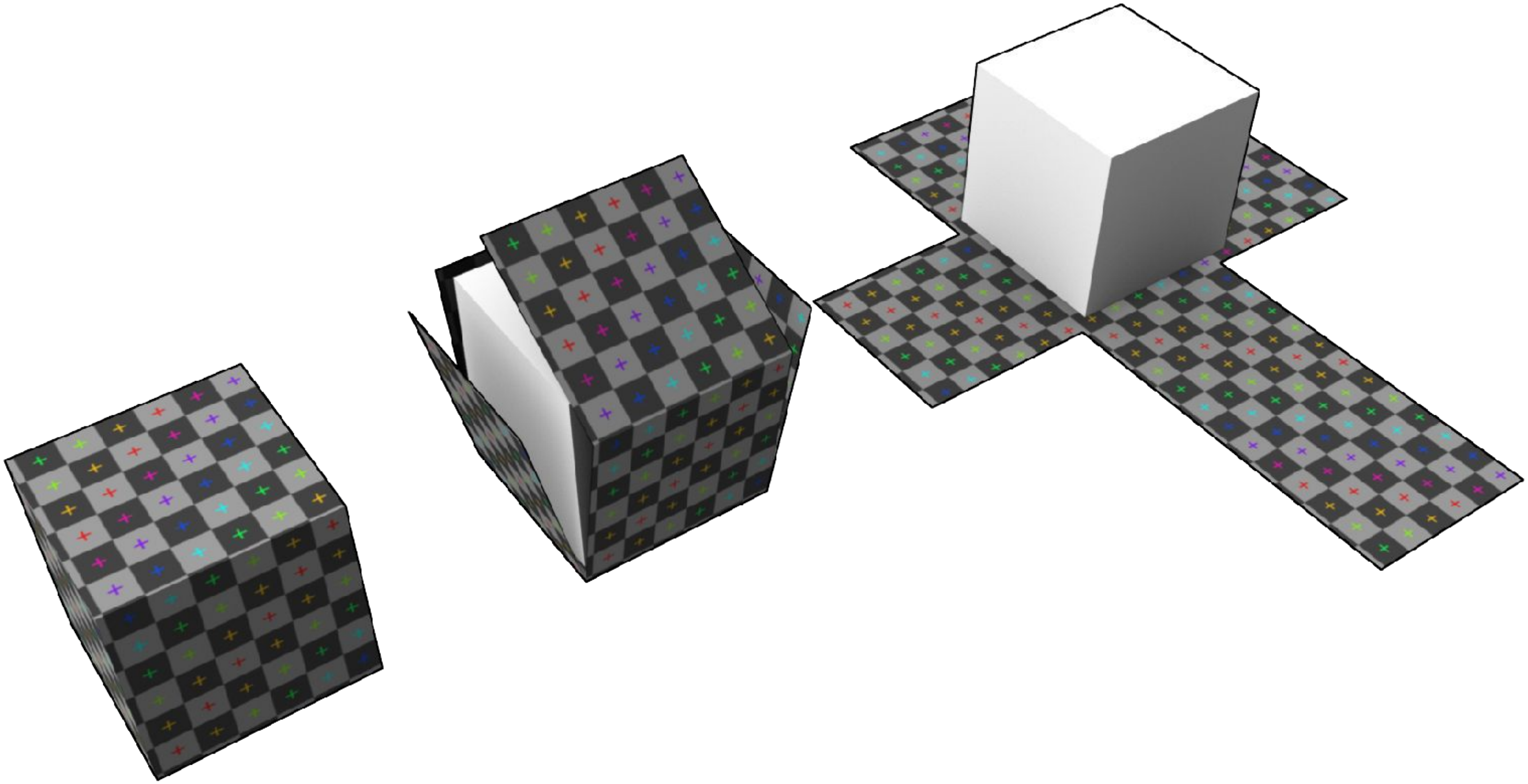
Texture Mapping

- What is beyond [0...1]?
  - Clamp

  -
    ```
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S/T, GL_CLAMP);
    ```



UV Mapping



UV Mapping