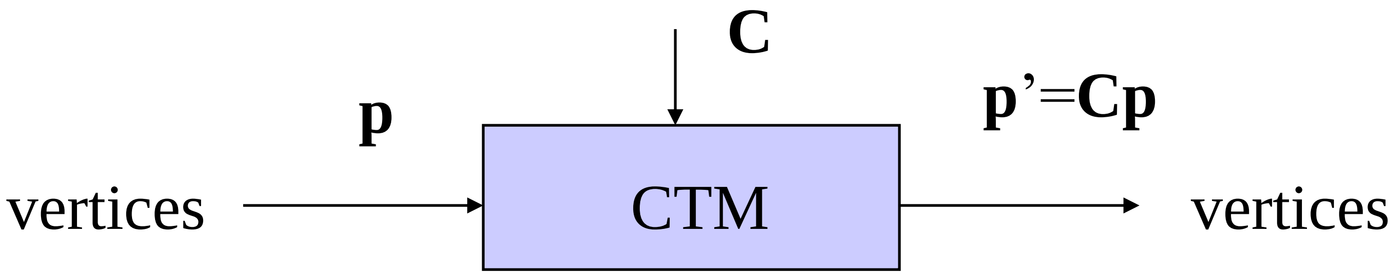
The University of New Mexico

# OpenGL Matrices

- In OpenGL matrices are part of the state
- Multiple types
  - Model-View (**`GL_MODELVIEW`**)
  - Projection (**`GL_PROJECTION`**)
  - Texture (**`GL_TEXTURE`**) (ignore for now)
  - Color(**`GL_COLOR`**) (ignore for now)
- Single set of functions for manipulation
- Select which to manipulated by
  - **`glMatrixMode(GL_MODELVIEW);`**
  - **`glMatrixMode(GL_PROJECTION);`**

# Current Transformation Matrix (CTM)

- Conceptually there is a 4 x 4 homogeneous coordinate matrix, the *current transformation matrix* (CTM) that is part of the state and is applied to all vertices that pass down the pipeline
- The CTM is defined in the user program and loaded into a transformation unit



## CTM operations

- The CTM can be altered either by loading a new CTM or by postmutiplication

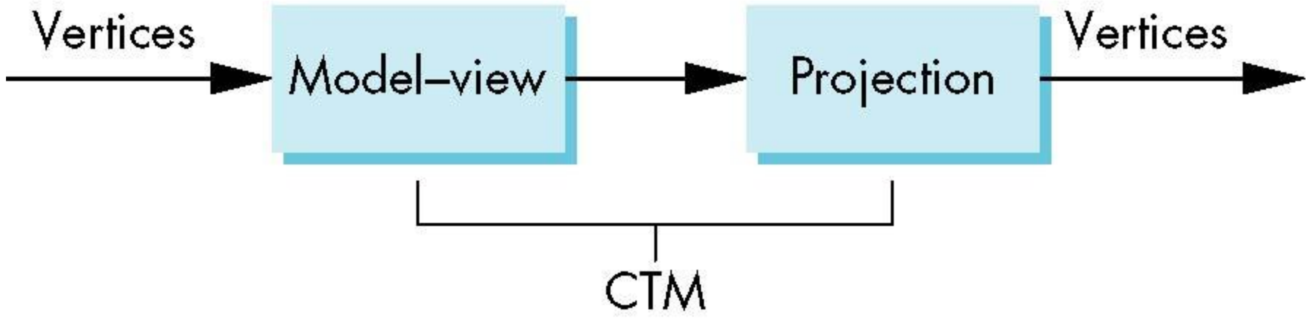
  Load an identity matrix: $\mathbf{C} \leftarrow \mathbf{I}$
  Load an arbitrary matrix: $\mathbf{C} \leftarrow \mathbf{M}$

  Load a translation matrix: $\mathbf{C} \leftarrow \mathbf{T}$
  Load a rotation matrix: $\mathbf{C} \leftarrow \mathbf{R}$
  Load a scaling matrix: $\mathbf{C} \leftarrow \mathbf{S}$

  Postmultiply by an arbitrary matrix: $\mathbf{C} \leftarrow \mathbf{CM}$
  Postmultiply by a translation matrix: $\mathbf{C} \leftarrow \mathbf{CT}$
  Postmultiply by a rotation matrix: $\mathbf{C} \leftarrow \mathbf{C\,R}$
  Postmultiply by a scaling matrix: $\mathbf{C} \leftarrow \mathbf{C\,S}$

# CTM in OpenGL

- OpenGL has a model-view and a projection matrix in the pipeline which are concatenated together to form the CTM
- Can manipulate each by first setting the correct matrix mode



# Rotation, Translation, Scaling

Load an identity matrix:

```
glLoadIdentity()
```

Multiply on right:

```
glRotatef(theta, vx, vy, vz)
```

**`theta`** in degrees, (**`vx, vy, vz`**) define axis of rotation

```
glTranslatef(dx, dy, dz)
glScalef( sx, sy, sz)
```

Each has a float (f) and double (d) format (**`glScaled`**)

# Example

- Rotation about z axis by 30 degrees with a fixed point of (1.0, 2.0, 3.0)

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(1.0, 2.0, 3.0);
glRotatef(30.0, 0.0, 0.0, 1.0);
glTranslatef(-1.0, -2.0, -3.0);
```

- Remember that last matrix specified in the program is the first applied

## Matrix Stacks

- In many situations we want to save transformation matrices for use later
  - Traversing hierarchical data structures (Chapter 10)
  - Avoiding state changes when executing display lists
- OpenGL maintains stacks for each type of matrix
  - Access present type (as set by **`glMatrixMode`**) by

```
glPushMatrix()
glPopMatrix()
```

The University of New Mexico

# Reading Back Matrices

- Can also access matrices (and other parts of the state) by *query* functions

```
glGetIntegerv
glGetFloatv
glGetBooleanv
glGetDoublev
glIsEnabled
```

- For matrices, we use as

```
double m[16];
glGetFloatv(GL_MODELVIEW, m);
```