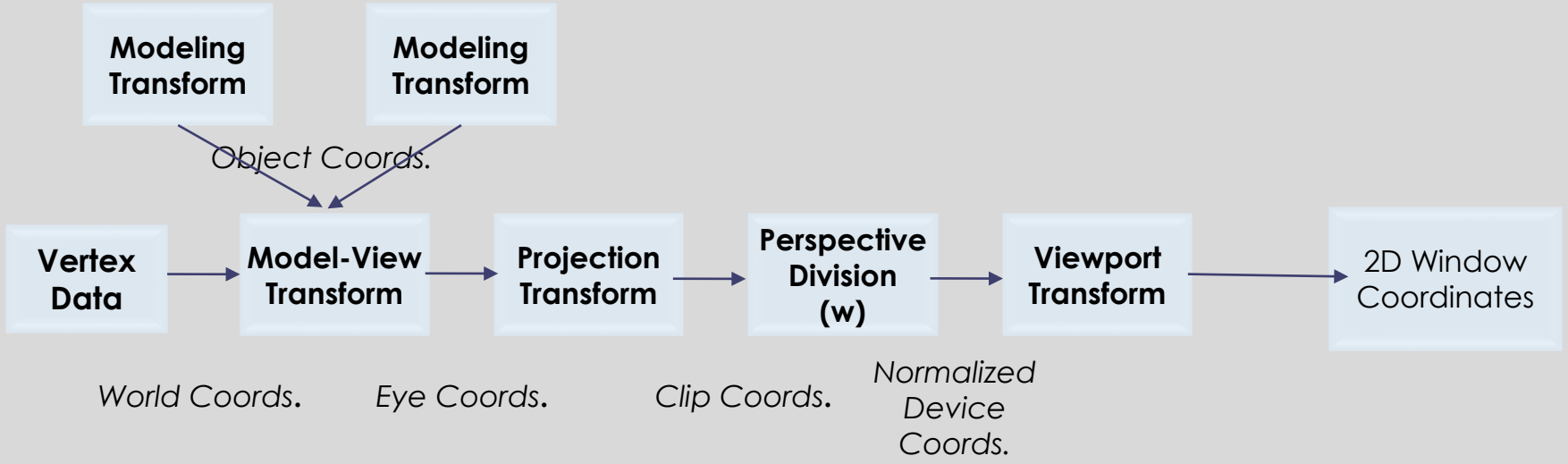
Geometric Transformations (OpenGL)

Carol O'Sullivan

Credits: Some slides from Rachel McDonnell and from Robb T. Koether, Hampden-Sydney College

Transformation Pipeline

- Transformations take us from one "space" to another
  - All of our transforms are 4 x 4 matrices

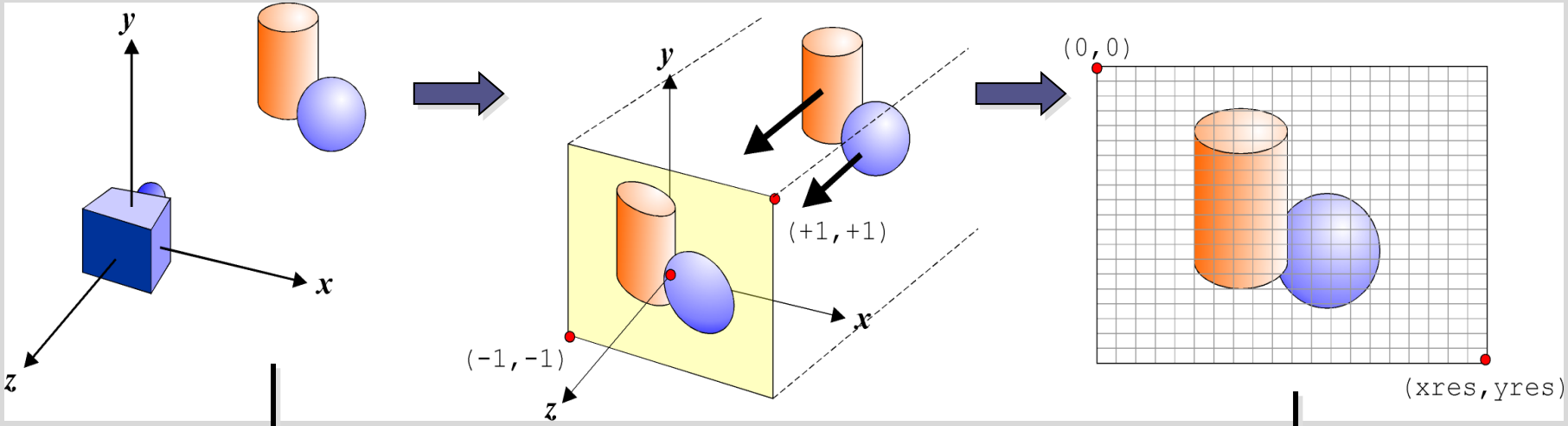


Camera Analogy

- Modelling transformations
  - Moving the model
- Viewing transformations
  - Tripod- define position and orientation of the viewing volume in the world
- Projection transformations
  - Adjust the lens of the camera
- Viewport transformations
  - Enlarge or reduce the physical photograph

Camera Modeling in OpenGL ®

camera coordinate system → viewport coordinate system → device/screen coordinate system



MODELVIEW

GL_PROJECTION

glViewport

Model Matrix

- When you create a triangle or
- Load a mesh from a file
- Has some (0,0,0) origin, local to that particular mesh
- Translate, rotate, scale to position in a virtual world
  - Multiply points with a model matrix (“world matrix”)
  - `mat4 M = T * R * S;`
- `vec4 pos_wor = M * vec4 (pos_loc, 1.0);`

Using Uniforms to Transform Geometry

- Now it is time to put all our knowledge together and build a program that does a little more than pass vertices through un-transformed

Vertex Shader for Rotation

```
// Remember: these matrices are column-major
(unlike typical c-programming array filling)

mat4 rx = mat4( 1.0,  0.0,  0.0, 0.0,
                0.0,  c.x,  s.x, 0.0,
                0.0, -s.x,  c.x, 0.0,
                0.0,  0.0,  0.0, 1.0 );

mat4 ry = mat4( c.y, 0.0, -s.y, 0.0,
                0.0, 1.0,  0.0, 0.0,
                s.y, 0.0,  c.y, 0.0,
                0.0, 0.0,  0.0, 1.0 );

//note – theta will be in radians in C
//Right-hand rule for rotation directions
//glUniformMatrix4v – set flag to “false”
```

OpenGL - Uniforms

- Pass data into a shader that stays the same – is uniform
  - e.g., transformation matrix
- Get data directly from application to shaders
- Two approaches
  - Declare in default block
  - Store in buffer object
- Simply place the keyword **uniform** at beginning of variable definition
  - uniform float fTime
  - uniform mat4 modelMatrix

The Old Vertex Shader

```
in vec4 vPosition;

void main () {
  // The value of vPosition should be between -1.0 and +1.0
  gl_Position = vPosition;
}
```

```
out vec4 fColor ;

void main () {
  // No matter what, color the pixel red!
  fColor = vec4 (1.0, 0.0, 0.0, 1.0);
}
```

A Better Vertex Shader

```
in vec4 vPosition; // the vertex in local coordinate system
uniform mat4 mM; // the matrix for the pose of the model
uniform mat4 mV; // The matrix for the pose of the camera
uniform mat4 mP; // The projection matrix (perspective)


void main () {
   // The value of vPosition should be between -1.0 and +1.0
   gl_Position = mP * mV * mM * vPosition;
}
```