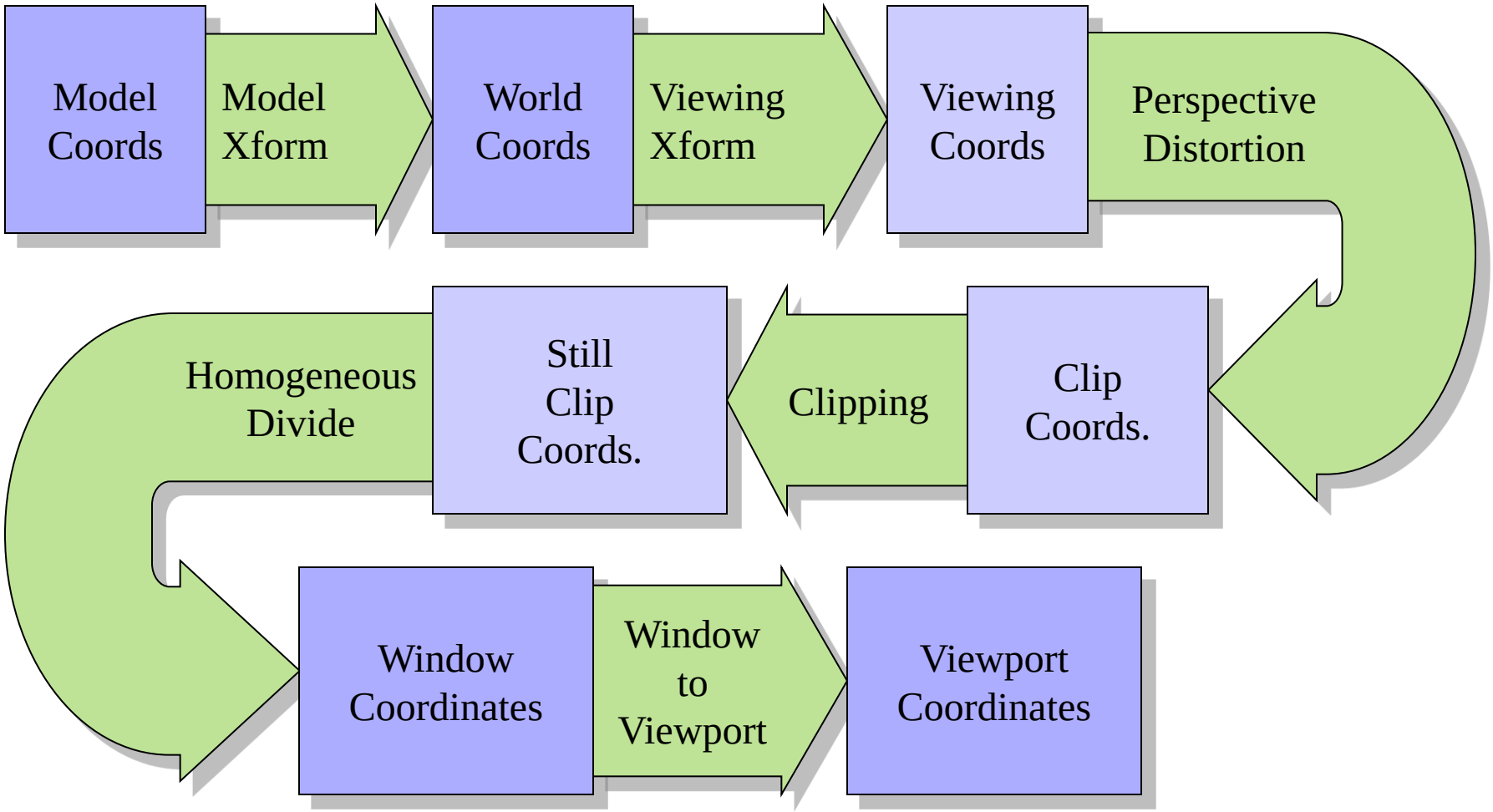
The Vertex Shader

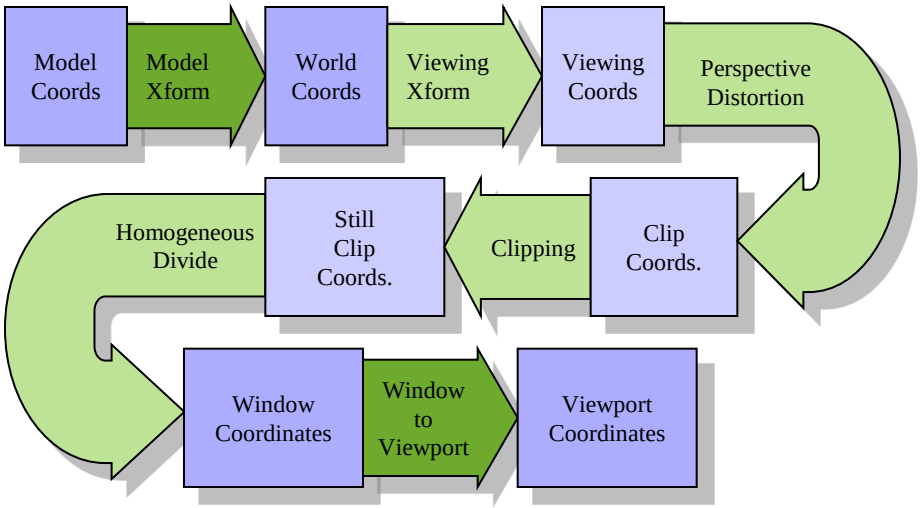
CS418 Computer Graphics

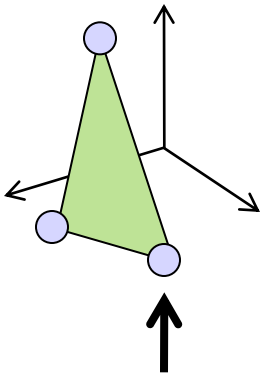
John C. Hart

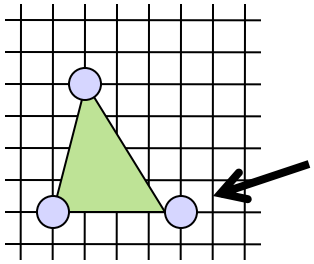
Graphics Pipeline



Graphics Pipeline

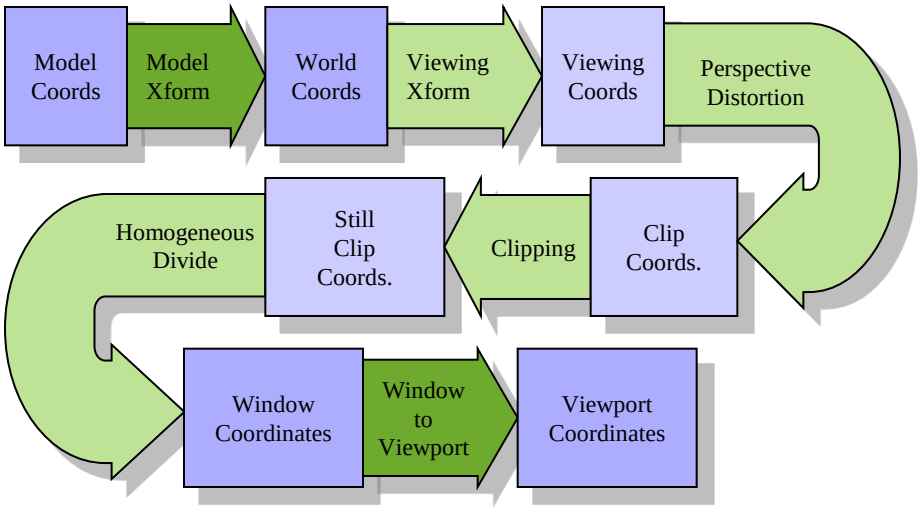


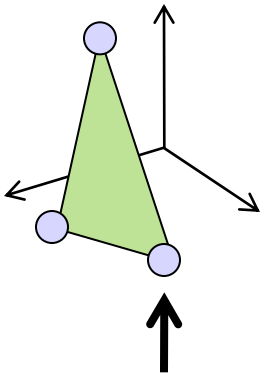


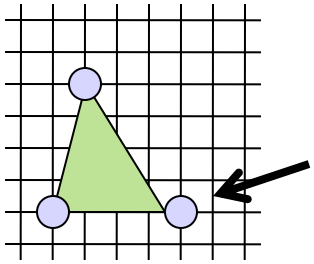


$$\begin{bmatrix} x_s \\ y_s \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \text{W2V} \end{bmatrix} \begin{bmatrix} \text{Persp} \end{bmatrix} \begin{bmatrix} \text{View} \end{bmatrix} \begin{bmatrix} \text{Model} \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ z_m \\ 1 \end{bmatrix}$$

Graphics Pipeline

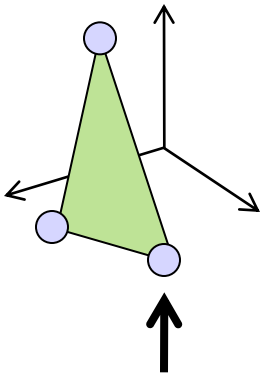


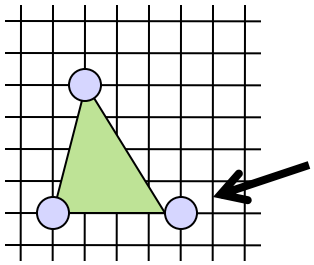




$$\begin{bmatrix} x_s \\ y_s \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \text{W2V} \end{bmatrix} \begin{bmatrix} \text{Persp} \end{bmatrix} \begin{bmatrix} \text{ModelView} \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ z_m \\ 1 \end{bmatrix}$$

Graphics Pipeline



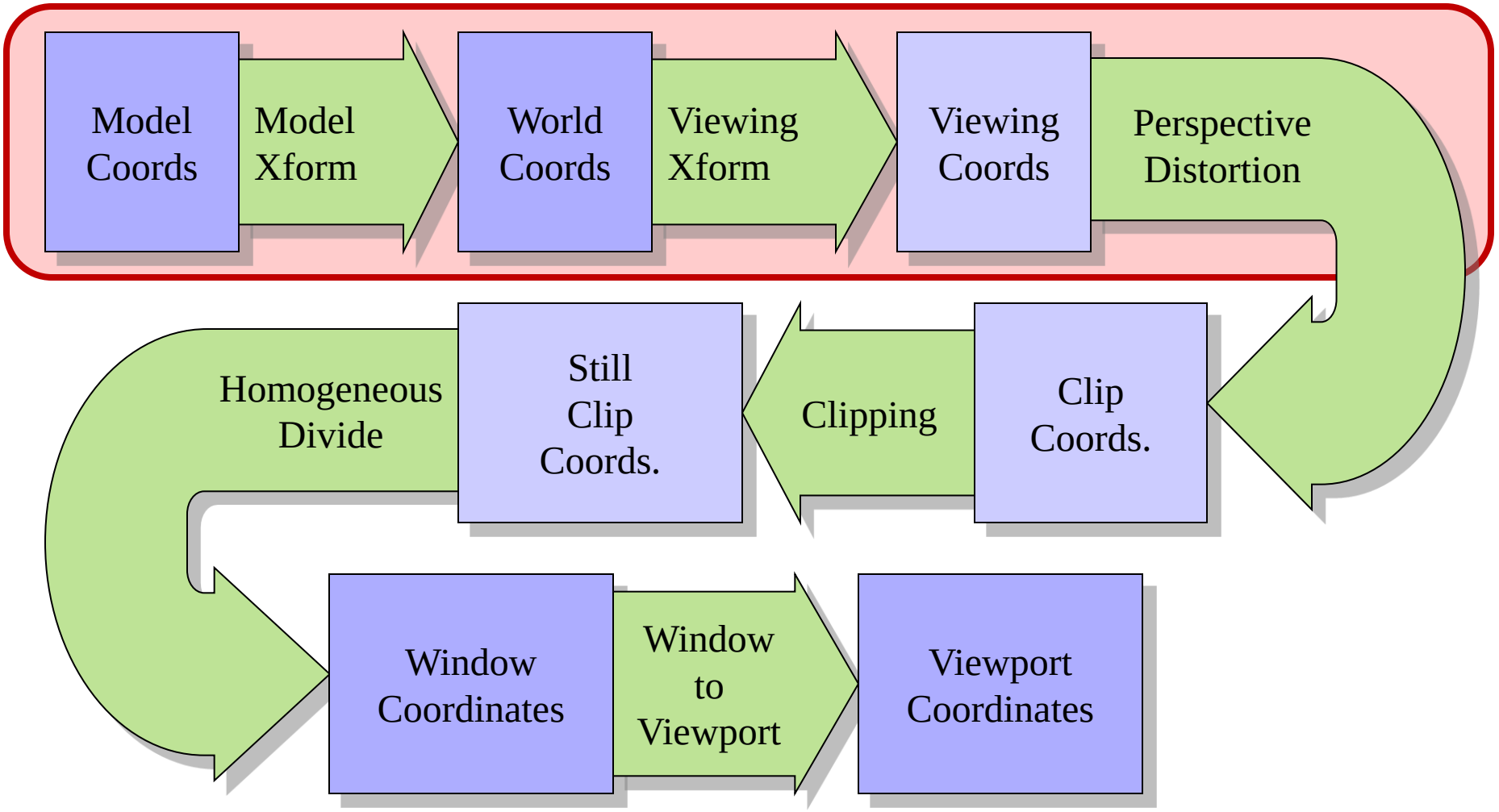


$$\begin{bmatrix} x_s \\ y_s \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \text{W2V} \end{bmatrix} \begin{bmatrix} \text{Persp} \end{bmatrix} \begin{bmatrix} \text{ModelView} \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ z_m \\ 1 \end{bmatrix}$$

```
glMatrixMode(GL_PROJECTION);
glFrustum(left,right,bottom,top,near,far);
glMatrixMode(GL_MODELVIEW);
gluLookAt(…);
```

…modeling transformations in reverse order…

Vertex Shader



Vertex Programming

Languages

- NVIDIA: Cg
- Microsoft: HLSL
- OpenGL: GLSL
- ATI: RenderMonkey
- Assembly Language
- Register Combiners
- Multipass Processing

OpenGL GLSL

- C-like language with some convenient C++ constructs
- Little language devoted to shaders (inspired by Pixar's Renderman)
- Device/OS independent, as opposed to Cg or HLSL
- Direct access into OpenGL state including matrices

GLSL Vertex Shader

```
GLchar vertexShaderCode = "
 void main() {
  glPosition = gl_ProjectionMatrix*gl_ModelViewMatrix*glVertex;
 }
";

GLuint vertexShaderObj = glCreateShader(GL_VERTEX_SHADER);
glShaderSource(vertexShaderObj, 1, vertexShaderCode, NULL);
glCompileShader(vertexShaderObj); /* Converts to GPU code */

GLuint programObj = glCreateProgram();
glAttachObject(programObj,vertexShaderObj);
glLinkProgram(programObj); /* Connects shaders & variables */
glUseProgram(programObj); /* OpenGL now uses the shader */
```

GLSL Variables

Variable Types

- Vectors
  - vec4 eye = vec4(1.0,2.0,3.0,1.0);
  - eye.x, eye.y, eye.z, eye.w
  - also eye.r, eye.g, eye.b, eye.a
- Matrices
  - mat4 mv = glModelViewMatrix;
  - elements: mv[1][2]
  - mv[1] is a vec4
  - mv * eye gives matrix vector product
- Swizzling
  - eye.xz = eye.zx

Variable Qualifiers

- Const
  - Unchanged by the shader
- Uniform
  - Set once per triangle
  - Set outside begin, end
- Attribute
  - Set once per vertex
- Varying
  - Interpolated across triangle

Passing Variables

```
GLchar vertexShaderCode = “
  const amp = 0.1;
  uniform float phase;
  attribute float pos;
  void main() {
    glVertex.y += amp*sin(pos + phase);
    glPosition = gl_ModelViewProjectionMatrix*glVertex;
  }
“;

GLint phaseParam = glGetUniformLocation(programObj,”phase”);
GLint posAttrib = glGetAttribLocation(programObj,”pos”);

glUniform1f(programObj,phaseParam,time);
glBegin(…)
glVertexAttrib1f(posAttrib,x*z);
```