Drawing Primitives

- `glBegin(Glenum mode)`
  - Beginning of a group of primitives
  - `mode`: GL_TRIANGLES, GL_QUADS, etc

- `glVertex3f(GLfloat x, GLfloat y, GLfloat z)`
  - Draw one vertex; must be after `glBegin()`!

- `glEnd()`
  - End of a group of primitives
Drawing Primitives

- `glColor3f(GLbyte red, GLbyte blue, GLbyte green)`
  - Set the color for vertices drawn after the call

- `glNormal3f(GLfloat nx, GLfloat ny, GLfloat nz)`
  - Set the normal vector

- `glTexCoord2f(GLfloat s, GLfloat v)`
  - Set the texture coordinates
Transformations

- `glMatrixMode(GLenum mode)`
  - Sets the current matrix
  - `mode`: `GL_MODELVIEW`, `GL_PROJECTION`

- `glLoadIdentity()`
  - Loads the identity matrix into the current matrix

- `glLoadMatrixf(GLfloat *matrix)`
  - Loads an arbitrary matrix
  - `matrix`: An array of values in column-major form; elements from same column are contiguous

- `glGetFloatv(GL_MODELVIEW_MATRIX, matrix)`
- `glGetFloatv(GL_PROJECTION_MATRIX, matrix)`
  - Gets a matrix – use sparingly!
Transformations

- `glMultMatrixf(GLfloat *matrix)`
  - Multiplies an arbitrary matrix with the current matrix

- `glTranslatef(GLfloat x, GLfloat y, GLfloat z)`
  - Multiplies the current matrix by a translation matrix

- `glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z)`
  - Multiplies the current matrix by a rotation matrix
  - `x, y, z`: Vector to rotate around
  - `angle`: How much to rotate, in degrees

- `glScalef(GLfloat x, GLfloat y, GLfloat z)`
  - Scales along the `x, y,` and `z` axis
Transformations

- **glFrustum(GLfloat left, GLfloat right, GLfloat bottom, GLfloat top, GLfloat near, GLfloat far)**
  - Multiplies the current matrix by a perspective matrix (more detail in next session)
  - Parameters set up a viewing frustum (clipped pyramid) representing the visible space

- **glOrtho(GLfloat left, GLfloat right, GLfloat bottom, GLfloat top, GLfloat near, GLfloat far)**
  - Similar to glFrustum(), but sets up an orthographic projection instead
Matrix Stack

- `glPushMatrix()`
  - Pushes a copy of the current matrix down on the matrix stack
  - There is one matrix stack per mode; `GL_PROJECTION` and `GL_MODELVIEW` each have their own stack

- `glPopMatrix()`
  - Replaces the current matrix with the top of the stack, and then pops the stack
Lighting

- glEnable(GL_LIGHTING)
  - Enable lighting

- glEnable(GL_LIGHTX)
  - Enable light X

- glLightfv(GL_LIGHT0, GL_AMBIENT, GLfloat *params)
- glLightfv(GL_LIGHT0, GL_SPECULAR, GLfloat *params)
- glLightfv(GL_LIGHT0, GL_DIFFUSE, GLfloat *params)
  - Sets the ambient, diffuse, specular light for light #0
  - params: 4-vector: [red, green, blue, alpha]
Lighting

- `glLightfv(GL_LIGHT0, GL_POSITION, GLfloat *params)`
  - Sets the light position or direction
  - `params`: 4-vector: [x, y, z, point?]; if last argument is non-zero then the light is a point light

- `glMaterialfv(GL_FRONT, GL_AMBIENT, GLfloat *params)`
- `glMaterialfv(GL_FRONT, GL_SPECULAR, GLfloat *param)`
- `glMaterialfv(GL_FRONT, GL_DIFFUSE, GLfloat *params)`
  - Sets the ambient, diffuse, and specular material colors
  - `params`: 4-vector: [red, blue, green, alpha]

- `glMaterialf(GL_FRONT, GL_SHININESS, GLfloat param)`
  - Sets the hardness/shininess (size of specular highlights)
  - `param`: shininess in the range [0, 128]
Other

- `glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)`
  - Must call at the beginning of each frame

- `glEnable(GL_DEPTH_TEST)`
  - Enable depth-testing using depth buffer
These functions let you draw from arrays, rather than using `glColor`, `glVertex`, etc.

- `glEnableClientState(GL_VERTEX_ARRAY)`
- `glEnableClientState(GL_NORMAL_ARRAY)`
- `glEnableClientState(GL_COLOR_ARRAY)`
- `glVertexPointer()`
- `glNormalPointer()`
- `glColorPointer()`
- `glDrawArrays()`