* The main focus of this exam is to determine if you know what the standard OpenGL commands do.
* For each of the following commands be able to explain:
  + what “actually” occurs when the command is executed
  + when they are used.
* THIS IS A CLOSED BOOK EXAM! You need to know these by sight!!!!
  + Bring to class on exam day a document
    - with your name at the top
    - that YOU write
    - that has the answer to each command.
* Use more than one sentence for every answer
  + be complete and precise.
  + You will hand this in to me before the exam.
* This will be 20% of your exam.
  + You will get a zero on this part if it is a copy of another students document.  Capice!?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* glGenVertexArrays(GLsizei *n*, GLuint \**arrays*); r 17-18
  + **what occurs**: returns *n* unused names for use as vertex-array objects in the array *arrays.* The names returned are marked as used for the purposes of allocating additional buffer objects, and initialized with values representing the default state of the collection ofunitialized vertex arrays.
  + **when used:** in the init function

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* glBindVertexArray(Gluint *array*); r 17-18
  + **what occurs:** 
    - when using the value array that is other than zero and was returned from glGenVertexArrays(), a new vertex-array object is created and assigned that name.
    - when binding to a previously created vertex-array object, that vertex array object becomes active, which additionally affects the vertex array state stored in the object.
    - when binding to an array value of zero, OpenGL stops using application-allocated vertex-array objects and returns to the default state for vertex arrays.
  + **when used:**
    - in the init function, for the first reason listed above
    - subsequently, when making a different object active (as we used in program 4)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* GLuint Loc = glGetUniformLocation(GLuint *program*, const char\* *name*); r 47, 569, 744, 784
  + **what occurs:** returns the index of the unifiorm variable name associated with the shader program.
    - name is a null-terminated character striing with no spaces
    - a value of -1 is returned if name doesn’t correspond to a uniform variable in the active shader program, or if a reserved shader variable name (gl\_ prefix) is specified
    - **there’s more**
  + **when used:**
  + GLuint Loc

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* GLuint vPosition = glGetAttribLocation(GLuint *program*, GLchar \**name*); r 129, **739**, **784**
  + **what occurs:**
  + **when used:** in load

GLuint vPosition = glGetAttribLocation(program, "vPosition");

glEnableVertexAttribArray(vPosition);

glVertexAttribPointer(vPosition, 4, GL\_FLOAT, GL\_FALSE, 0,

BUFFER\_OFFSET(0)); // offset 0: vertex position data in first half of buffer

GLuint vColor = glGetAttribLocation(program, "vColor");

glEnableVertexAttribArray(vColor);

glVertexAttribPointer(vColor, 4, GL\_FLOAT, GL\_FALSE, 0,

BUFFER\_OFFSET(size)); // offset size: color data in second half of buffer

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* glUniformMatrix4fv(ModelViewLoc, 1, GL\_TRUE, model\_view); r 554, **openGL.org**
  + what are the params, etc.
    - ModelViewLoc: GLint *location*
    - 1: GLsizei *count*
    - GL\_TRUE: GLboolean *transpose*
    - model\_view: constGLfloat *\*value*
  + **what occurs:**
  + **when used:**
    - in reshape, with projection

projection = Perspective(fovy, aspectRatio, zNear, zFar); // SSZ

glUniformMatrix4fv(ProjectionLoc, 1, GL\_TRUE, projection); // SSZ

* + - in display, with model\_view:

glUniformMatrix4fv(modelViewLoc, 1, GL\_TRUE, model\_view);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* glEnableVertexAttribArray(vPosition); r **27-8**, 112-6, 129-30
  + **what occurs:** the array designated by the variable *index,* in this case vPosition, is enabled
  + **when used:** in load

GLuint vPosition = glGetAttribLocation(program, "vPosition");

glEnableVertexAttribArray(vPosition);

glVertexAttribPointer(vPosition, 4, GL\_FLOAT, GL\_FALSE, 0, BUFFER\_OFFSET(0));

// offset 0: vertex position data in first half of buffer

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* glVertexAttribPointer(vPosition, 4, GL\_FLOAT, GL\_FALSE, 0,BUFFER\_OFFSET(0));
  + **26-7**, 30, 93, 108-13, 129, 130, 149
  + Specifies where the data values for shader attribute location *index* can be found.
  + (GLuint *index*, GLint *size*, GLenum *type*, GLboolean *normalized*, GLsizei *stride*, const GLvoid *\*pointer*)
    - index: location of vshader input variable
    - size: number of values for each vertex in the array
    - type:
    - normalized: are values constrained to (-1,1), that is, normalized?
    - stride: byte offset between data members; 0 means tightly packed
    - \*pointer: the location within the buffer where the data starts; buffer offset (0) means the data begins at the first byte (address zero) of the buffer.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* glDrawArrays(GL\_TRIANGLES, start, length);
  + what is start and length?
  + Is length the number of bytes or vertices or triangles?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* LookAt(); // what are the params. what is returned

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* RotateX(theta); //what is the parameter and what is returned.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Ortho() ; // what are the params. what is returned. What does this return value actually do?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Perspective();//// what are the params, and what is returned. What does this return value actually do?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* glDrawArrays(GL\_TRIANGLES, start, length);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* glGenBuffers(1, &buffer);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* glBindBuffer(GL\_ARRAY\_BUFFER, buffer);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* glBufferData(GL\_ARRAY\_BUFFER, Size+Size,  
  NULL, GL\_STATIC\_DRAW);glBufferSubData(GL\_ARRAY\_BUFFER, 0, Size, &vertexList[0]);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* glBufferSubData(GL\_ARRAY\_BUFFER, Size, Size, &normalList[0]);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Know the meaning of the code in the following vertex shader.
  + How are the in variables fed.
  + How are the uniform variable set.
  + What do the commands do?

#version 420  
in vec4 vPosition;  
in vec4 vNormal;  
out vec4 color;uniform vec3 LAmbient, LDiffuse, LSpecular;  
uniform vec3 MAmbient, MDiffuse, MSpecular;  
uniform mat4 ModelView;  
uniform mat4 Projection;  
uniform vec4 LightPosition;  
uniform float Shininess;void main()  
{  
// Transform vertex position into eye coordinates  
vec3 pos = (ModelView \* vPosition).xyz;vec3 L = normalize( LightPosition.xyz – pos );  
vec3 E = normalize( -pos );  
vec3 H = normalize( L + E );// Transform vertex normal into eye coordinates  
vec3 N = normalize( ModelView\*vNormal).xyz;// Compute terms in the illumination equation  
vec3 ambient = LAmbient\*MAmbient;float Kd = max( dot(L, N), 0.0 );  
vec3 diffuse = Kd\*LDiffuse\*MDiffuse;float Ks = pow( max(dot(N, H), 0.0), Shininess );  
vec3 specular = Ks \* LSpecular\*MSpecular;

if( dot(L, N) < 0.0 ) {  
specular = vec3(0.0, 0.0, 0.0);  
}

gl\_Position = Projection \* ModelView \* vPosition;

color = vec4(ambient + diffuse + specular,1.0);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*