CSCI441 Library

v1.7

Generated by Doxygen 1.8.14

Contents

Chapter 1

Main Page

This library is intended to be used with OpenGL for CSCI441 at the Colorado School of Mines.

When building, the library must be compiled and linked against OpenGL and glm. Headers thatend in ∗3.hpp are built for OpenGL 3.0+ and additionally depend upon GLEW.

TODO Items

- Handle vertex and face colors in OFF files

Revision History

v1.8.1 - 04 Dec 2017

- Marked modelLoader3.hpp function implementations as inline to prevent redefinition errors

v1.8 - 16 Nov 2017

- Added support for MTL files for Phong Shading and diffuse maps

v1.7 - 16 Nov 2017

Added loadBMP() support to TextureUtils.hpp
Added support for ASCII STL files to modelLoader3.hpp
Added support for OFF files to modelLoader3.hpp
Fixed reallocation error if model did not load properly
Added support for ASCII PLY files to modelLoader3.hpp (as long as first three vertex properties are x/y/z location)
If PLY file does not contain normal information (we're currently not checking for it), can autogenerate vertex normals
If OFF file does not contain normal information, can autogenerate vertex normals
If OBJ file does not contain normal information, can autogenerate vertex normals

v1.6 - 15 Nov 2017

Added FramebufferUtils3.hpp to print Framebuffer info
Fixed off by 1 error for normals/texcoords in modelLoader3.hpp
Fixed overflow error for modelLoader3.hpp when reading in models with more than 65535 vertices

v1.5.1 - 10 Nov 2017

Fixed redefinition errors in teapot3.hpp and objects3.hpp
Fixed bug in ShaderUtils3.hpp to check if OpenGL is version 4.0+ before querying subroutine uniforms

v1.5 - 06 Nov 2017

Added loadTGA method to TextureUtils.hpp
Commenting added to TextureUtils.hpp
Converted OpenGLUtils from static non-implementable class to namespace
Added commenting to ShaderProgram3.hpp

v1.4.1 - 05 Nov 2017

Fixed bug in objects3.hpp of internally passing torus parameters in incorrect order

v1.4 - 03 Nov 2017

Created ShaderUtils3.hpp and ShaderProgram3.hpp to make working with Shaders easier

v.1.3.1 - 28 Oct 2017

Matched internal data types to prevent c++11 narrowing warnings on lab machines

v1.3 - 26 Oct 2017

Modified texture coordinates for cylinder to linear step from 0 to 1 in s instead of following cosine
Modified texture coordinates for sphere to linear step form 0 to 1 in s & t instead of following sine and cosine
Fixed bug when disk was not being displayed if consisting of 1 ring
Fixed bug with Partial Disk not starting at current angle
Fixed bug with normals on Sphere stacks
Added modelLoader3.hpp to handle loading and drawing OBJ files
Added objects3.hpp that allow for solid primitives to be drawn with OpenGL 3.0
Notes for teapot - the teapot cannot be textured and it is a pure teapot with no bottom
For a textured teapot, look into using an object model

v1.2 - 25 Sep 2017

Fixed error in draw∗Disk not completing final slice step
Added TextureUtils to load in a PPM
Added MaterialStruct structure to group together Phong properties
Fixed error in drawSolidDisk() not allowing inner radius to be zero

v1.1.1 - 22 Sep 2017

Removed GL_MAX_COLOR_ATTACHMENTS to comply with lab machines

v1.1 - 21 Sep 2017

Added OpenGLUtils class to store commonly used helper functions

v1.0.1 - 19 Sep 2017

Added documentation
Added inline definition to functions to prevent duplicate linking errors

v1.0 - 01 Sep 2017

Initial release of all OpenGL 3D objects

Chapter 2

Deprecated List

Member CSCI441::DEPRECATED (void pushMatrix(glm::mat4 mtx))

Multiplies current matrix by given matrix

Multiplies current matrix by inverse of given matrix

Multiplies current matrix by inverse of given matrix

Member CSCI441::DEPRECATED (void setMaterial(MaterialStruct material))

Multiplies current matrix by given matrix Sets the diffuse, ambient, specular, and shininess properties at once

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 CSCI441 Namespace Reference

CSCI441 Helper Functions for OpenGL.

Classes

- class ModelLoader

  Loads object models from file and renders using VBOs/VAOs.
- class ShaderProgram

  Handles registration and compilation of Shaders.

Functions

- void drawSolidCone (GLdouble base, GLdouble height, GLint stacks, GLint slices)

  Draws a solid cone.
- void drawWireCone (GLdouble base, GLdouble height, GLint stacks, GLint slices)

  Draws a wireframe cone.
- void drawSolidCube (GLdouble sideLength)

  Draws a solid cube.
- void drawWireCube (GLdouble sideLength)

  Draws a wireframe cube.
- void drawSolidCylinder (GLdouble base, GLdouble top, GLdouble height, GLint stacks, GLint slices)

  Draws a solid open ended cylinder.
- void drawWireCylinder (GLdouble base, GLdouble top, GLdouble height, GLint stacks, GLint slices)

  Draws a wireframe open ended cylinder.
- void drawSolidDisk (GLdouble inner, GLdouble outer, GLint slices, GLint rings)

  Draws a solid disk.
- void drawWireDisk (GLdouble inner, GLdouble outer, GLint slices, GLint rings)

  Draws a wireframe disk.
- void drawSolidPartialDisk (GLdouble inner, GLdouble outer, GLint slices, GLint rings, GLdouble start, G↩ Ldouble sweep)

  Draws part of a solid disk.

- void drawWirePartialDisk (GLdouble inner, GLdouble outer, GLint slices, GLint rings, GLdouble start, GLdouble sweep)

  Draws part of a wireframe disk.
- void drawSolidSphere (GLdouble radius, GLint stacks, GLint slices)

  Draws a solid sphere.
- void drawWireSphere (GLdouble radius, GLint stacks, GLint slices)

  Draws a wireframe sphere.
- void drawSolidTeapot (GLdouble size)

  Draws a solid teapot.
- void drawWireTeapot (GLdouble size)

  Draws a wireframe teapot.
- void drawSolidTorus (GLdouble innerRadius, GLdouble outerRadius, GLint sides, GLint rings)

  Draws a solid torus.
- void drawWireTorus (GLdouble innerRadius, GLdouble outerRadius, GLint sides, GLint rings)

  Draws a wireframe torus.
- void pushMatrix (glm::mat4 mtx)

  Multiplies current matrix by given matrix.
- void popMatrix (glm::mat4 mtx)

  Multiplies current matrix by inverse of given matrix.
- void setMaterial (MaterialStruct material)

  sets all material properties at once
- void setVertexAttributeLocations (GLint positionLocation, GLint normalLocation=-1, GLint texCoordLocation=-1)

  Sets the attribute locations for vertex positions, normals, and texture coordinates.
- DEPRECATED (void pushMatrix(glm::mat4 mtx))

  Multiplies current matrix by given matrix.
- DEPRECATED (void setMaterial(MaterialStruct material))

  sets all material properties at once

6.1.1 Detailed Description

CSCI441 Helper Functions for OpenGL.

6.1.2 Function Documentation

6.1.2.1 DEPRECATED() [1/2]

```
CSCI441::DEPRECATED (
            void  pushMatrixglm::mat4 mtx )
```

Multiplies current matrix by given matrix.

Multiplies current matrix by inverse of given matrix.

Deprecated Multiplies current matrix by given matrix

Parameters

glm::mat4	mtx - matrix to multiply the current matrix by

Deprecated Multiplies current matrix by inverse of given matrix

Parameters

glm::mat4	mtx - matrix to multiply the current matrix by the inverse of

6.1.2.2 DEPRECATED() [2/2]

```
CSCI441::DEPRECATED (
            void  setMaterialMaterialStruct material )
```

sets all material properties at once

Deprecated Multiplies current matrix by given matrix Sets the diffuse, ambient, specular, and shininess properties at once

Parameters

MaterialStruct	material - the material to set the properties for

6.1.2.3 drawSolidCone()

```
void CSCI441::drawSolidCone (
             GLdouble base,
             GLdouble height,
             GLint stacks,
             GLint slices )  [inline]
```

Draws a solid cone.

Cone is oriented along the y-axis with the origin along the base of the cone

Parameters

GLdouble	base - radius of the base of the cone
GLdouble	height - height of the cone from the base to the tip
GLint	stacks - resolution of the number of steps rotated around the central axis of the cone
GLint	slices - resolution of the number of steps to take along the height

Precondition

base must be greater than zero
height must be greater than zero
stacks must be greater than zero
slices must be greater than two

6.1.2.4 drawSolidCube()

```
void CSCI441::drawSolidCube (
             GLdouble sideLength )  [inline]
```

Draws a solid cube.

The origin is at the cube's center of mass. Cube is oriented with our XYZ axes

Parameters

GLdouble	sideLength - length of the edge of the cube

Precondition

sideLength must be greater than zero

6.1.2.5 drawSolidCylinder()

```
void CSCI441::drawSolidCylinder (
             GLdouble base,
             GLdouble top,
             GLdouble height,
             GLint stacks,
             GLint slices )  [inline]
```

Draws a solid open ended cylinder.

Cylinder is oriented along the y-axis with the origin along the base

Parameters

GLdouble	base - radius of the base of the cylinder
GLdouble	top - radius of the top of the cylinder
GLdouble	height - height of the cylinder from the base to the top
GLint	stacks - resolution of the number of steps rotated around the central axis of the cylinder
GLint	slices - resolution of the number of steps to take along the height

Precondition

either: (1) base is greater than zero and top is greater than or equal to zero or (2) base is greater than or equal to zero and top is greater than zero
height must be greater than zero
stacks must be greater than zero
slices must be greater than two

6.1.2.6 drawSolidDisk()

```
void CSCI441::drawSolidDisk (
            GLdouble inner,
            GLdouble outer,
            GLint slices,
            GLint rings )  [inline]
```

Draws a solid disk.

Disk is drawn in the XY plane with the origin at its center

Parameters

GLdouble	inner - equivalent to the width of the disk
GLdouble	outer - radius from the center of the disk to the center of the ring
GLint	slices - resolution of the number of steps rotated along the disk
GLint	rings - resolution of the number of steps to take along the disk width

Precondition

inner is greater than or equal to zero
outer is greater than zero
outer is greater than inner
slices is greater than two
rings is greater than zero

6.1.2.7 drawSolidPartialDisk()

```
void CSCI441::drawSolidPartialDisk (
            GLdouble inner,
            GLdouble outer,
            GLint slices,
            GLint rings,
            GLdouble start,
            GLdouble sweep )  [inline]
```

Draws part of a solid disk.

Disk is drawn in the XY plane with the origin at its center

Parameters

GLdouble	inner - equivalent to the width of the disk
GLdouble	outer - radius from the center of the disk to the center of the ring
GLint	stacks - resolution of the number of steps rotated along the disk
GLint	rings - resolution of the number of steps to take along the disk width
GLdouble	start - angle in degrees to start the disk at
GLdouble	sweep - distance in degrees to rotate through

Precondition

inner is greater than or equal to zero
outer is greater than zero
outer is greater than inner
slices is greater than two
rings is greater than zero
start is between [0, 360]
sweep is between [0, 360]

6.1.2.8 drawSolidSphere()

```
void CSCI441::drawSolidSphere (
             GLdouble radius,
             GLint stacks,
             GLint slices )  [inline]
```

Draws a solid sphere.

Origin is at the center of the sphere

Parameters

GLdouble	radius - radius of the sphere
GLint	stacks - resolution of the number of steps to take along theta (rotate around Y-axis)
GLint	slices - resolution of the number of steps to take along phi (rotate around X- or Z-axis)

Precondition

radius must be greater than 0
stacks must be greater than 2
slices must be greater than 2

6.1.2.9 drawSolidTeapot()

```
void CSCI441::drawSolidTeapot (
             GLdouble size )  [inline]
```

Draws a solid teapot.

Oriented with spout and handle running along X-axis, cap and bottom along Y-axis. Origin is at the center of the teapot

Parameters

GLdouble	size - scale of the teapot

Precondition

size must be greater than zero

6.1.2.10 drawSolidTorus()

```
void CSCI441::drawSolidTorus (
            GLdouble innerRadius,
            GLdouble outerRadius,
            GLint sides,
            GLint rings )  [inline]
```

Draws a solid torus.

Torus is oriented in the XY-plane with the origin at its center

Parameters

innerRadius	- equivalent to the width of the torus ring
outerRadius	- radius from the center of the torus to the center of the ring
sides	- resolution of steps to take around the band of the ring
rings	- resolution of steps to take around the torus

Precondition

innerRadius must be greater than zero
outerRadius must be greater than zero
sides must be greater than two
rings must be greater than two

6.1.2.11 drawWireCone()

```
void CSCI441::drawWireCone (
            GLdouble base,
            GLdouble height,
            GLint stacks,
            GLint slices )  [inline]
```

Draws a wireframe cone.

Cone is oriented along the y-axis with the origin along the base of the cone

Parameters

GLdouble	base - radius of the base of the cone
GLdouble	height - height of the cone from the base to the tip
GLint	stacks - resolution of the number of steps rotated around the central axis of the cone
GLint	slices - resolution of the number of steps to take along the height

Precondition

base must be greater than zero
height must be greater than zero
stacks must be greater than zero
slices must be greater than two

6.1.2.12 drawWireCube()

```
void CSCI441::drawWireCube (
            GLdouble sideLength )  [inline]
```

Draws a wireframe cube.

The origin is at the cube's center of mass. Cube is oriented with our XYZ axes

Parameters

GLdouble	sideLength - length of the edge of the cube

Precondition

sideLength must be greater than zero

6.1.2.13 drawWireCylinder()

```
void CSCI441::drawWireCylinder (
            GLdouble base,
            GLdouble top,
            GLdouble height,
            GLint stacks,
            GLint slices )  [inline]
```

Draws a wireframe open ended cylinder.

Cylinder is oriented along the y-axis with the origin along the base

Parameters

GLdouble	base - radius of the base of the cylinder
GLdouble	top - radius of the top of the cylinder
GLdouble	height - height of the cylinder from the base to the top
GLint	stacks - resolution of the number of steps rotated around the central axis of the cylinder
GLint	slices - resolution of the number of steps to take along the height

Precondition

either: (1) base is greater than zero and top is greater than or equal to zero or (2) base is greater than or equal to zero and top is greater than zero
height must be greater than zero
stacks must be greater than zero
slices must be greater than two

6.1.2.14 drawWireDisk()

```
void CSCI441::drawWireDisk (
             GLdouble inner,
             GLdouble outer,
             GLint slices,
             GLint rings )  [inline]
```

Draws a wireframe disk.

Disk is drawn in the XY plane with the origin at its center

Parameters

GLdouble	inner - equivalent to the width of the disk
GLdouble	outer - radius from the center of the disk to the center of the ring
GLint	slices - resolution of the number of steps rotated along the disk
GLint	rings - resolution of the number of steps to take along the disk width

Precondition

inner is greater than or equal to zero
outer is greater than zero
outer is greater than inner
slices is greater than two
rings is greater than zero

6.1.2.15 drawWirePartialDisk()

```
void CSCI441::drawWirePartialDisk (
             GLdouble inner,
```

```
            GLdouble outer,
            GLint slices,
            GLint rings,
            GLdouble start,
            GLdouble sweep )  [inline]
```

Draws part of a wireframe disk.

Disk is drawn in the XY plane with the origin at its center

Parameters

GLdouble	inner - equivalent to the width of the disk
GLdouble	outer - radius from the center of the disk to the center of the ring
GLint	stacks - resolution of the number of steps rotated along the disk
GLint	rings - resolution of the number of steps to take along the disk width
GLdouble	start - angle in degrees to start the disk at
GLdouble	sweep - distance in degrees to rotate through

Precondition

inner is greater than or equal to zero
outer is greater than zero
outer is greater than inner
slices is greater than two
rings is greater than zero
start is between [0, 360]
sweep is between [0, 360]

6.1.2.16 drawWireSphere()

```
void CSCI441::drawWireSphere (
            GLdouble radius,
            GLint stacks,
            GLint slices )  [inline]
```

Draws a wireframe sphere.

Origin is at the center of the sphere

Parameters

GLdouble	radius - radius of the sphere
GLint	stacks - resolution of the number of steps to take along theta (rotate around Y-axis)
GLint	slices - resolution of the number of steps to take along phi (rotate around X- or Z-axis)

Precondition

radius must be greater than 0
stacks must be greater than 2
slices must be greater than 2

6.1.2.17 drawWireTeapot()

```
void CSCI441::drawWireTeapot (
            GLdouble size )  [inline]
```

Draws a wireframe teapot.

Oriented with spout and handle running along X-axis, cap and bottom along Y-axis. Origin is at the center of the teapot

Parameters

GLdouble	size - scale of the teapot

Precondition

size must be greater than zero

6.1.2.18 drawWireTorus()

```
void CSCI441::drawWireTorus (
            GLdouble innerRadius,
            GLdouble outerRadius,
            GLint sides,
            GLint rings )  [inline]
```

Draws a wireframe torus.

Torus is oriented in the XY-plane with the origin at its center

Parameters

innerRadius	- equivalent to the width of the torus ring
outerRadius	- radius from the center of the torus to the center of the ring
sides	- resolution of steps to take around the band of the ring
rings	- resolution of steps to take around the torus

Precondition

innerRadius must be greater than zero
outerRadius must be greater than zero
sides must be greater than two
rings must be greater than two

6.1.2.19 popMatrix()

```
void CSCI441::popMatrix (
            glm::mat4 mtx )  [inline]
```

Multiplies current matrix by inverse of given matrix.

Parameters

glm::mat4	mtx - matrix to multiply the current matrix by the inverse of

6.1.2.20 pushMatrix()

```
void CSCI441::pushMatrix (
            glm::mat4 mtx )  [inline]
```

Multiplies current matrix by given matrix.

Parameters

glm::mat4	mtx - matrix to multiply the current matrix by

6.1.2.21 setMaterial()

```
void CSCI441::setMaterial (
            MaterialStruct material )  [inline]
```

sets all material properties at once

Sets the diffuse, ambient, specular, and shininess properties at once

Parameters

MaterialStruct	material - the material to set the properties for

6.1.2.22 setVertexAttributeLocations()

```
void CSCI441::setVertexAttributeLocations (
            GLint positionLocation,
            GLint normalLocation = -1,
            GLint texCoordLocation = -1 )  [inline]
```

Sets the attribute locations for vertex positions, normals, and texture coordinates.

Needs to be called after a shader program is being used and before drawing geometry

Parameters

GLint	positionLocation - location of the vertex position attribute
GLint	normalLocation - location of the vertex normal attribute
GLint	texCoordLocation - location of the vertex texture coordinate attribute

6.2 FramebufferUtils Namespace Reference

OpenGL Texture Utility functions.

6.2.1 Detailed Description

OpenGL Texture Utility functions.

6.3 OpenGLUtils Namespace Reference

contains OpenGL Utility functions

6.3.1 Detailed Description

contains OpenGL Utility functions

6.4 TextureUtils Namespace Reference

OpenGL Texture Utility functions.

6.4.1 Detailed Description

OpenGL Texture Utility functions.

Chapter 7

Class Documentation

7.1 CSCI441::ModelLoader Class Reference

Loads object models from file and renders using VBOs/VAOs.

`#include <modelLoader3.hpp>`

Public Member Functions

- ModelLoader ()

  Creates an empty model.
- ModelLoader (const char *filename)

  Loads a model from the given file.
- ~ModelLoader ()

  Frees memory associated with model on both CPU and GPU.
- bool loadModelFile (const char *filename, bool INFO=true, bool ERRORS=true)

  Loads a model from the given file.
- bool draw (GLint positionLocation, GLint normalLocation=-1, GLint texCoordLocation=-1, GLint matDiffLocation=-1, GLint matSpecLocation=-1, GLint matShinLocation=-1, GLint matAmbLocation=-1, GLenum diffuseTexture=GL_TEXTURE0)

  Renders a model.

Static Public Member Functions

- static void enableAutoGenerateNormals ()

  Enable autogeneration of vertex normals.
- static void disableAutoGenerateNormals ()

  Disable autogeneration of vertex normals.

7.1.1 Detailed Description

Loads object models from file and renders using VBOs/VAOs.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 ModelLoader()

```
CSCI441::ModelLoader::ModelLoader (
            const char * filename )  [inline]
```

Loads a model from the given file.

Parameters

const	char* filename - file to load model from

7.1.3 Member Function Documentation

7.1.3.1 disableAutoGenerateNormals()

```
void CSCI441::ModelLoader::disableAutoGenerateNormals ( )  [inline], [static]
```

Disable autogeneration of vertex normals.

If an object model does not contain vertex normal data, then normals will be computed based on the cross product of vertex winding order.

Note

Must be called prior to loading in a model from file
No normals are generated by default

7.1.3.2 draw()

```
bool CSCI441::ModelLoader::draw (
            GLint positionLocation,
            GLint normalLocation = -1,
            GLint texCoordLocation = -1,
            GLint matDiffLocation = -1,
            GLint matSpecLocation = -1,
            GLint matShinLocation = -1,
            GLint matAmbLocation = -1,
            GLenum diffuseTexture = GL_TEXTURE0 )  [inline]
```

Renders a model.

Parameters

GLint	positionLocation - attribute location of vertex position
GLint	normalLocation - attribute location of vertex normal
GLint	texCoordLocation - attribute location of vertex texture coordinate
GLint	matDiffLocation - attribute location of material diffuse component
GLint	matSpecLocation - attribute location of material specular component
GLint	matShinLocation - attribute location of material shininess component
GLint	matAmbLocation - attribute location of material ambient component
GLenum	diffuseTexture - texture number to bind diffuse texture map to

Returns

true if draw succeeded, false otherwise

7.1.3.3 enableAutoGenerateNormals()

```
void CSCI441::ModelLoader::enableAutoGenerateNormals ( )  [inline], [static]
```

Enable autogeneration of vertex normals.

If an object model does not contain vertex normal data, then normals will be computed based on the cross product of vertex winding order.

Note

Must be called prior to loading in a model from file

7.1.3.4 loadModelFile()

```
bool CSCI441::ModelLoader::loadModelFile (
            const char * filename,
            bool INFO = true,
            bool ERRORS = true )  [inline]
```

Loads a model from the given file.

Parameters

const	char* filename - file to load model from
bool	INFO - flag to control if informational messages should be displayed
bool	ERRORS - flag to control if error messages should be displayed

Returns

true if load succeeded, false otherwise

The documentation for this class was generated from the following file:

- CSCI441/modelLoader3.hpp

7.2 CSCI441::ShaderProgram Class Reference

Handles registration and compilation of Shaders.

`#include <ShaderProgram3.hpp>`

Public Member Functions

- ShaderProgram (const char *vertexShaderFilename, const char *fragmentShaderFilename)

  Creates a Shader Program using a Vertex Shader and Fragment Shader.
- ShaderProgram (const char *vertexShaderFilename, const char *tesselationControlShaderFilename, const char *tesselationEvaluationShaderFilename, const char *geometryShaderFilename, const char *fragment↩ ShaderFilename)

  Creates a Shader Program using a Vertex Shader, Tesselation Shader, Geometry Shader, and Fragment Shader.
- ShaderProgram (const char *vertexShaderFilename, const char *tesselationControlShaderFilename, const char *tesselationEvaluationShaderFilename, const char *fragmentShaderFilename)

  Creates a Shader Program using a Vertex Shader, Tesselation Shader, and Fragment Shader.
- ShaderProgram (const char *vertexShaderFilename, const char *geometryShaderFilename, const char *fragmentShaderFilename)

  Creates a Shader Program using a Vertex Shader, Geometry Shader, and Fragment Shader.
- ~ShaderProgram ()

  Clean up memory associated with the Shader Program.
- GLint getUniformLocation (const char *uniformName)

  Returns the location of the given uniform in this shader program.
- GLint getUniformBlockIndex (const char *uniformBlockName)

  Returns the index of the given uniform block in this shader program.
- GLint getUniformBlockSize (const char *uniformBlockName)

  Returns the size of the given uniform block in this shader program.
- GLubyte * getUniformBlockBuffer (const char *uniformBlockName)

  Returns an allocated buffer for the given uniform block in this shader program.
- GLint * getUniformBlockOffsets (const char *uniformBlockName)

  Returns an array of offsets into the buffer for the given uniform block in this shader program.
- GLint * getUniformBlockOffsets (const char *uniformBlockName, const char *names[])

  Returns an array of offsets into the buffer for the given uniform block and names in this shader program.
- void setUniformBlockBinding (const char *uniformBlockName, GLuint binding)

  Set the binding point for the given uniform block in this shader program.
- GLint getAttributeLocation (const char *attributeName)

  Returns the location of the given attribute in this shader program.
- GLuint getSubroutineIndex (GLenum shaderStage, const char *subroutineName)

  Returns the index of the given subroutine for a shader stage in this shader program.
- GLuint getNumUniforms ()

Returns the number of active uniforms in this shader program.

- GLuint getNumUniformBlocks ()

  Returns the number of active uniform blocks in this shader program.

- GLuint getNumAttributes ()

  Returns the number of active attributes in this shader program.

- GLuint getShaderProgramHandle ()

  Returns the handle for this shader program.

- void useProgram ()

  Sets the Shader Program to be active.

Static Public Member Functions

- static void enableDebugMessages ()

  Enables debug messages from Shader Program functions.

- static void disableDebugMessages ()

  Disables debug messages from Shader Program functions.

7.2.1 Detailed Description

Handles registration and compilation of Shaders.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 ShaderProgram() [1/4]

```
CSCI441::ShaderProgram::ShaderProgram (
            const char * vertexShaderFilename,
            const char * fragmentShaderFilename )
```

Creates a Shader Program using a Vertex Shader and Fragment Shader.

Parameters

const	char* vertexShaderFilename - name of the file corresponding to the vertex shader
const	char* fragmentShaderFilename - name of the file corresponding to the fragment shader

7.2.2.2 ShaderProgram() [2/4]

```
CSCI441::ShaderProgram::ShaderProgram (
            const char * vertexShaderFilename,
            const char * tesselationControlShaderFilename,
```

```
            const char * tesselationEvaluationShaderFilename,
            const char * geometryShaderFilename,
            const char * fragmentShaderFilename )
```

Creates a Shader Program using a Vertex Shader, Tesselation Shader, Geometry Shader, and Fragment Shader.

Parameters

const	char* vertexShaderFilename - name of the file corresponding to the vertex shader
const	char* tesselationControlShaderFilename - name of the file corresponding to the tesselation control shader
const	char* tesselationEvaluationShaderFilename - name of the file corresponding to the tesselation evaluation shader
const	char* geometryShaderFilename - name of the file corresponding to the geometry shader
const	char* fragmentShaderFilename - name of the file corresponding to the fragment shader

7.2.2.3 ShaderProgram() [3/4]

```
CSCI441::ShaderProgram::ShaderProgram (
            const char * vertexShaderFilename,
            const char * tesselationControlShaderFilename,
            const char * tesselationEvaluationShaderFilename,
            const char * fragmentShaderFilename )
```

Creates a Shader Program using a Vertex Shader, Tesselation Shader, and Fragment Shader.

Parameters

const	char* vertexShaderFilename - name of the file corresponding to the vertex shader
const	char* tesselationControlShaderFilename - name of the file corresponding to the tesselation control shader
const	char* tesselationEvaluationShaderFilename - name of the file corresponding to the tesselation evaluation shader
const	char* fragmentShaderFilename - name of the file corresponding to the fragment shader

7.2.2.4 ShaderProgram() [4/4]

```
CSCI441::ShaderProgram::ShaderProgram (
            const char * vertexShaderFilename,
            const char * geometryShaderFilename,
            const char * fragmentShaderFilename )
```

Creates a Shader Program using a Vertex Shader, Geometry Shader, and Fragment Shader.

Parameters

const	char* vertexShaderFilename - name of the file corresponding to the vertex shader
const	char* geometryShaderFilename - name of the file corresponding to the geometry shader
const	char* fragmentShaderFilename - name of the file corresponding to the fragment shader

7.2.3 Member Function Documentation

7.2.3.1 disableDebugMessages()

```
void CSCI441::ShaderProgram::disableDebugMessages ( )  [static]
```

Disables debug messages from Shader Program functions.

Disables debug messages from Shader Program functions. Debug messages are on by default.

7.2.3.2 enableDebugMessages()

```
void CSCI441::ShaderProgram::enableDebugMessages ( )  [static]
```

Enables debug messages from Shader Program functions.

Enables debug messages from Shader Program functions. Debug messages are on by default.

7.2.3.3 getAttributeLocation()

```
GLint CSCI441::ShaderProgram::getAttributeLocation (
            const char * attributeName )
```

Returns the location of the given attribute in this shader program.

Note

Prints an error message to standard error stream if the attribute is not found

Parameters

const	char* attributeName - name of the attribute to get the location for

Returns

GLint - location of the given attribute in this shader program

7.2.3.4 getNumAttributes()

```
GLuint CSCI441::ShaderProgram::getNumAttributes ( )
```

Returns the number of active attributes in this shader program.

Returns

GLuint - number of active attributes in this shader program

7.2.3.5 getNumUniformBlocks()

```
GLuint CSCI441::ShaderProgram::getNumUniformBlocks ( )
```

Returns the number of active uniform blocks in this shader program.

Returns

GLuint - number of active uniform blocks in this shader program

7.2.3.6 getNumUniforms()

```
GLuint CSCI441::ShaderProgram::getNumUniforms ( )
```

Returns the number of active uniforms in this shader program.

Returns

GLuint - number of active uniforms in this shader program

7.2.3.7 getShaderProgramHandle()

```
GLuint CSCI441::ShaderProgram::getShaderProgramHandle ( )
```

Returns the handle for this shader program.

Returns

GLuint - handle for this shader program

7.2.3.8 getSubroutineIndex()

```
GLuint CSCI441::ShaderProgram::getSubroutineIndex (
            GLenum shaderStage,
            const char * subroutineName )
```

Returns the index of the given subroutine for a shader stage in this shader program.

Note

Prints an error message to standard error stream if the subroutine is not found

Parameters

GLenum	shaderStage - stage of the shader program to get the subroutine for. Allowable values: GL_VERTEX_SHADER, GL_TESS_CONTROL_SHADER, GL_TESS_EVALUATION_SHADER, GL_GEOMETRY_SHADER, GL_FRAGMENT_SHADER
const	char* subroutineName - name of the subroutine to get the location for

Returns

GLuint - index of the given subroutine for the shader stage in this shader program

7.2.3.9 getUniformBlockBuffer()

```
GLubyte * CSCI441::ShaderProgram::getUniformBlockBuffer (
            const char * uniformBlockName )
```

Returns an allocated buffer for the given uniform block in this shader program.

Note

Prints an error message to standard error stream if the uniform block is not found

Parameters

const	char* uniformBlockName - name of the uniform block to allocate a buffer for

Returns

GLubyte* - allocated buffer for the given uniform block in this shader program

7.2.3.10 getUniformBlockIndex()

```
GLint CSCI441::ShaderProgram::getUniformBlockIndex (
            const char * uniformBlockName )
```

Returns the index of the given uniform block in this shader program.

Note

Prints an error message to standard error stream if the uniform block is not found

Parameters

const	char* uniformBlockName - name of the uniform block to get the index for

Returns

GLint - index of the given uniform block in this shader program

7.2.3.11 getUniformBlockOffsets() [1/2]

```
GLint * CSCI441::ShaderProgram::getUniformBlockOffsets (
            const char * uniformBlockName )
```

Returns an array of offsets into the buffer for the given uniform block in this shader program.

Note

Prints an error message to standard error stream if the uniform block is not found

Parameters

const	char* uniformBlockName - name of the uniform block to return offsets for

Returns

GLint* - array of offsets for the given uniform block in this shader program

7.2.3.12 getUniformBlockOffsets() [2/2]

```
GLint * CSCI441::ShaderProgram::getUniformBlockOffsets (
            const char * uniformBlockName,
            const char * names[] )
```

Returns an array of offsets into the buffer for the given uniform block and names in this shader program.

Note

Prints an error message to standard error stream if the uniform block is not found

Parameters

const	char* uniformBlockName - name of the uniform block to return offsets for
const	char* names[] - names of the uniform block components to get offsets for

Returns

GLint* - array of offsets for the given uniform block in this shader program

7.2.3.13 getUniformBlockSize()

```
GLint CSCI441::ShaderProgram::getUniformBlockSize (
            const char * uniformBlockName )
```

Returns the size of the given uniform block in this shader program.

Note

Prints an error message to standard error stream if the uniform block is not found

Parameters

const	char* uniformBlockName - name of the uniform block to get the size for

Returns

GLint - size of the given uniform block in this shader program

7.2.3.14 getUniformLocation()

```
GLint CSCI441::ShaderProgram::getUniformLocation (
            const char * uniformName )
```

Returns the location of the given uniform in this shader program.

Note

Prints an error message to standard error stream if the uniform is not found

Parameters

const	char* uniformName - name of the uniform to get the location for

Returns

GLint - location of the given uniform in this shader program

7.2.3.15 setUniformBlockBinding()

```
void CSCI441::ShaderProgram::setUniformBlockBinding (
            const char * uniformBlockName,
            GLuint binding )
```

Set the binding point for the given uniform block in this shader program.

Note

Prints an error message to standard error stream if the uniform block is not found

Parameters

const	char* uniformBlockName - name of the uniform block to bind
GLuint	binding - binding point for this uniform block

The documentation for this class was generated from the following file:

- CSCI441/ShaderProgram3.hpp

Chapter 8

File Documentation

8.1 CSCI441/FramebufferUtils3.hpp File Reference

Helper functions to work with OpenGL Framebuffers.

```
#include <GL/glew.h>
#include <stdio.h>
```

Namespaces

- CSCI441

  CSCI441 Helper Functions for OpenGL.
- FramebufferUtils

  OpenGL Texture Utility functions.

8.1.1 Detailed Description

Helper functions to work with OpenGL Framebuffers.

Author

Dr. Jeffrey Paone

Date

Last Edit: 14 Nov 2017

Version

1.6

Copyright

These functions, classes, and constants help minimize common code that needs to be written.

8.2 CSCI441/modelLoader3.hpp File Reference

Helper functions to draw 3D OpenGL 3.0+ objects.

```
#include <GL/glew.h>
#include <glm/glm.hpp>
#include <SOIL/SOIL.h>
#include <fstream>
#include <map>
#include <string>
#include <vector>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <CSCI441/modelMaterial.hpp>
#include <CSCI441/TextureUtils.hpp>
```

Classes

- class CSCI441::ModelLoader

  Loads object models from file and renders using VBOs/VAOs.

Namespaces

- CSCI441

  CSCI441 Helper Functions for OpenGL.

8.2.1 Detailed Description

Helper functions to draw 3D OpenGL 3.0+ objects.

Author

Dr. Jeffrey Paone

Date

Last Edit: 15 Nov 2017

Version

1.7

Copyright

This class will load and render object files. Currently supports: .obj .off .stl

Warning

NOTE: This header file will only work with OpenGL 3.0+
NOTE: This header file depends upon GLEW

8.3 CSCI441/objects.hpp File Reference

Helper functions to draw 3D OpenGL 2.1 objects.

```
#include <GL/gl.h>
#include <assert.h>
#include <math.h>
#include <CSCI441/teapot.hpp>
```

Namespaces

- CSCI441

  CSCI441 Helper Functions for OpenGL.

Functions

- void CSCI441::drawSolidCone (GLdouble base, GLdouble height, GLint stacks, GLint slices)

  Draws a solid cone.
- void CSCI441::drawWireCone (GLdouble base, GLdouble height, GLint stacks, GLint slices)

  Draws a wireframe cone.
- void CSCI441::drawSolidCube (GLdouble sideLength)

  Draws a solid cube.
- void CSCI441::drawWireCube (GLdouble sideLength)

  Draws a wireframe cube.
- void CSCI441::drawSolidCylinder (GLdouble base, GLdouble top, GLdouble height, GLint stacks, GLint slices)

  Draws a solid open ended cylinder.
- void CSCI441::drawWireCylinder (GLdouble base, GLdouble top, GLdouble height, GLint stacks, GLint slices)

  Draws a wireframe open ended cylinder.
- void CSCI441::drawSolidDisk (GLdouble inner, GLdouble outer, GLint slices, GLint rings)

  Draws a solid disk.
- void CSCI441::drawWireDisk (GLdouble inner, GLdouble outer, GLint slices, GLint rings)

  Draws a wireframe disk.
- void CSCI441::drawSolidPartialDisk (GLdouble inner, GLdouble outer, GLint slices, GLint rings, GLdouble start, GLdouble sweep)

  Draws part of a solid disk.
- void CSCI441::drawWirePartialDisk (GLdouble inner, GLdouble outer, GLint slices, GLint rings, GLdouble start, GLdouble sweep)

  Draws part of a wireframe disk.
- void CSCI441::drawSolidSphere (GLdouble radius, GLint stacks, GLint slices)

  Draws a solid sphere.
- void CSCI441::drawWireSphere (GLdouble radius, GLint stacks, GLint slices)

  Draws a wireframe sphere.
- void CSCI441::drawSolidTeapot (GLdouble size)

  Draws a solid teapot.
- void CSCI441::drawWireTeapot (GLdouble size)

  Draws a wireframe teapot.
- void CSCI441::drawSolidTorus (GLdouble innerRadius, GLdouble outerRadius, GLint sides, GLint rings)

  Draws a solid torus.
- void CSCI441::drawWireTorus (GLdouble innerRadius, GLdouble outerRadius, GLint sides, GLint rings)

  Draws a wireframe torus.

8.3.1 Detailed Description

Helper functions to draw 3D OpenGL 2.1 objects.

Author

Dr. Jeffrey Paone

Date

Last Edit: 25 Oct 2017

Version

1.3

Copyright

These functions draw solid (or wireframe) 3D closed OpenGL objects. All objects are constructed using triangles that have normals and texture coordinates properly set.

Warning

NOTE: This header file will only work with OpenGL 2.1

8.4 CSCI441/objects3.hpp File Reference

Helper functions to draw 3D OpenGL 3.0+ objects.

```
#include <GL/glew.h>
#include <assert.h>
#include <math.h>
#include <CSCI441/teapot3.hpp>
#include <map>
```

Namespaces

- CSCI441

  CSCI441 Helper Functions for OpenGL.

Functions

- void CSCI441::setVertexAttributeLocations (GLint positionLocation, GLint normalLocation=-1, GLint texCoordLocation=-1)

  Sets the attribute locations for vertex positions, normals, and texture coordinates.
- void CSCI441::drawSolidCone (GLdouble base, GLdouble height, GLint stacks, GLint slices)

  Draws a solid cone.
- void CSCI441::drawWireCone (GLdouble base, GLdouble height, GLint stacks, GLint slices)

  Draws a wireframe cone.
- void CSCI441::drawSolidCube (GLdouble sideLength)

  Draws a solid cube.
- void CSCI441::drawWireCube (GLdouble sideLength)

  Draws a wireframe cube.
- void CSCI441::drawSolidCylinder (GLdouble base, GLdouble top, GLdouble height, GLint stacks, GLint slices)

  Draws a solid open ended cylinder.
- void CSCI441::drawWireCylinder (GLdouble base, GLdouble top, GLdouble height, GLint stacks, GLint slices)

  Draws a wireframe open ended cylinder.
- void CSCI441::drawSolidDisk (GLdouble inner, GLdouble outer, GLint slices, GLint rings)

  Draws a solid disk.
- void CSCI441::drawWireDisk (GLdouble inner, GLdouble outer, GLint slices, GLint rings)

  Draws a wireframe disk.
- void CSCI441::drawSolidPartialDisk (GLdouble inner, GLdouble outer, GLint slices, GLint rings, GLdouble start, GLdouble sweep)

  Draws part of a solid disk.
- void CSCI441::drawWirePartialDisk (GLdouble inner, GLdouble outer, GLint slices, GLint rings, GLdouble start, GLdouble sweep)

  Draws part of a wireframe disk.
- void CSCI441::drawSolidSphere (GLdouble radius, GLint stacks, GLint slices)

  Draws a solid sphere.
- void CSCI441::drawWireSphere (GLdouble radius, GLint stacks, GLint slices)

  Draws a wireframe sphere.
- void CSCI441::drawSolidTeapot (GLdouble size)

  Draws a solid teapot.
- void CSCI441::drawWireTeapot (GLdouble size)

  Draws a wireframe teapot.
- void CSCI441::drawSolidTorus (GLdouble innerRadius, GLdouble outerRadius, GLint sides, GLint rings)

  Draws a solid torus.
- void CSCI441::drawWireTorus (GLdouble innerRadius, GLdouble outerRadius, GLint sides, GLint rings)

  Draws a wireframe torus.

8.4.1 Detailed Description

Helper functions to draw 3D OpenGL 3.0+ objects.

Author

Dr. Jeffrey Paone

Date

Last Edit: 26 Oct 2017

Version

1.3

Copyright

These functions draw solid (or wireframe) 3D closed OpenGL objects. All objects are constructed using triangles that have normals and texture coordinates properly set.

Warning

NOTE: This header file will only work with OpenGL 3.0+
NOTE: This header file depends upon GLEW

8.5 CSCI441/OpenGLUtils.hpp File Reference

Helper functions to work with OpenGL 2.1.

```
#include <GL/gl.h>
#include <glm/glm.hpp>
#include <stdio.h>
```

Namespaces

- CSCI441

  CSCI441 Helper Functions for OpenGL.
- OpenGLUtils

  contains OpenGL Utility functions

Functions

- void CSCI441::pushMatrix (glm::mat4 mtx)

  Multiplies current matrix by given matrix.
- void CSCI441::popMatrix (glm::mat4 mtx)

  Multiplies current matrix by inverse of given matrix.
- void CSCI441::OpenGLUtils::printOpenGLInfo ()

  Prints information about our OpenGL context.
- void CSCI441::setMaterial (MaterialStruct material)

  sets all material properties at once

8.5.1 Detailed Description

Helper functions to work with OpenGL 2.1.

Author

Dr. Jeffrey Paone

Date

Last Edit: 25 Oct 2017

Version

1.3

Copyright

MIT License Copyright (c) 2017 Dr. Jeffrey Paone

These functions, classes, and constants help minimize common code that needs to be written.

Warning

NOTE: This header file depends upon glm

8.5.2 Function Documentation

8.5.2.1 printOpenGLInfo()

```
void CSCI441::OpenGLUtils::printOpenGLInfo ( )  [inline]
```

Prints information about our OpenGL context.

8.6 CSCI441/OpenGLUtils3.hpp File Reference

Helper functions to work with OpenGL 3.0+.

```
#include <GL/glew.h>
#include <glm/glm.hpp>
#include <stdio.h>
```

Namespaces

- CSCI441

  CSCI441 Helper Functions for OpenGL.
- OpenGLUtils

  contains OpenGL Utility functions

Functions

- CSCI441::DEPRECATED (void pushMatrix(glm::mat4 mtx))

  Multiplies current matrix by given matrix.
- void CSCI441::OpenGLUtils::printOpenGLInfo ()

  Prints information about our OpenGL context.
- CSCI441::DEPRECATED (void setMaterial(MaterialStruct material))

  sets all material properties at once

8.6.1 Detailed Description

Helper functions to work with OpenGL 3.0+.

Author

Dr. Jeffrey Paone

Date

Last Edit: 25 Oct 2017

Version

1.3

Copyright

MIT License Copyright (c) 2017 Dr. Jeffrey Paone

These functions, classes, and constants help minimize common code that needs to be written.

Warning

NOTE: This header file depends upon glm
NOTE: This header file depends upon GLEW

8.6.2 Function Documentation

8.6.2.1 printOpenGLInfo()

```
void CSCI441::OpenGLUtils::printOpenGLInfo ( )  [inline]
```

Prints information about our OpenGL context.

8.7 CSCI441/ShaderProgram3.hpp File Reference

Class to work with OpenGL 3.0+ Shaders.

```
#include "ShaderUtils3.hpp"
#include <stdlib.h>
```

Classes

- class CSCI441::ShaderProgram

  Handles registration and compilation of Shaders.

Namespaces

- CSCI441

  CSCI441 Helper Functions for OpenGL.

8.7.1 Detailed Description

Class to work with OpenGL 3.0+ Shaders.

Author

Dr. Jeffrey Paone

Date

Last Edit: 03 Nov 2017

Version

1.4

Copyright

These functions, classes, and constants help minimize common code that needs to be written.

8.8 CSCI441/ShaderUtils3.hpp File Reference

Helper functions to work with OpenGL Shaders.

```
#include <GL/glew.h>
#include <stdio.h>
#include <string.h>
#include <fstream>
#include <string>
```

8.8.1 Detailed Description

Helper functions to work with OpenGL Shaders.

Author

Dr. Jeffrey Paone

Date

Last Edit: 28 Oct 2017

Version

1.4

Copyright

These functions, classes, and constants help minimize common code that needs to be written.

8.9 CSCI441/teapot.hpp File Reference

Helper functions to draw teapot with OpenGL 2.1.

```
#include <GL/gl.h>
```

8.9.1 Detailed Description

Helper functions to draw teapot with OpenGL 2.1.

Date

Last Edit: 19 Sep 2017

Warning

NOTE: This header file will only work with OpenGL 2.1

Modified by Dr. Jeffrey Paone to work in Colorado School of Mines CSCI441 course context.

8.10 CSCI441/teapot3.hpp File Reference

Helper functions to draw teapot with OpenGL 3.0+.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <GL/glew.h>
```

8.10.1 Detailed Description

Helper functions to draw teapot with OpenGL 3.0+.

Date

Last Edit: 26 Oct 2017

Warning

NOTE: This header file will only work with OpenGL 3.0+

8.11 CSCI441/TextureUtils.hpp File Reference

Helper functions to work with OpenGL Textures.

```
#include <GL/gl.h>
#include <SOIL/SOIL.h>
#include <stdio.h>
#include <string>
```

Namespaces

- CSCI441

  CSCI441 Helper Functions for OpenGL.
- TextureUtils

  OpenGL Texture Utility functions.

Functions

- bool CSCI441::TextureUtils::loadBMP (const char *filename, int &imageWidth, int &imageHeight, int &imageChannels, unsigned char *imageData, const char *path="./")

  loads a BMP into memory
- bool CSCI441::TextureUtils::loadPPM (const char *filename, int &imageWidth, int &imageHeight, unsigned char *&imageData)

  loads a PPM into memory
- bool CSCI441::TextureUtils::loadTGA (const char *filename, int &imageWidth, int &imageHeight, unsigned char *&imageData, int &imageChannels)

  loads a TGA into memory
- GLuint CSCI441::TextureUtils::loadAndRegisterTexture (const char *filename, GLenum minFilter=GL_LIN↩EAR, GLenum magFilter=GL_LINEAR, GLenum wrapS=GL_REPEAT, GLenum wrapT=GL_REPEAT)

  loads and registers a texture into memory returning a texture handle
- GLuint CSCI441::TextureUtils::loadAndRegister2DTexture (const char *filename, GLenum minFilter=GL_L↩INEAR, GLenum magFilter=GL_LINEAR, GLenum wrapS=GL_REPEAT, GLenum wrapT=GL_REPEAT)

  loads and registers a texture into memory returning a texture handle

8.11.1 Detailed Description

Helper functions to work with OpenGL Textures.

Author

Dr. Jeffrey Paone

Date

Last Edit: 03 Nov 2017

Version

1.5

Copyright

These functions, classes, and constants help minimize common code that needs to be written.

8.11.2 Function Documentation

8.11.2.1 loadAndRegister2DTexture()

```
GLuint CSCI441::TextureUtils::loadAndRegister2DTexture (
            const char * filename,
            GLenum minFilter = GL_LINEAR,
            GLenum magFilter = GL_LINEAR,
            GLenum wrapS = GL_REPEAT,
            GLenum wrapT = GL_REPEAT )  [inline]
```

loads and registers a texture into memory returning a texture handle

This function loads a texture into memory and registers the texture with OpenGL. The provided minification and magnification filters are set for the texture. The texture coordinate wrapping parameters are also set.

Parameters

const	char* filename - name of texture to load
GLenum	minFilter - minification filter to apply (default: GL_LINEAR)
GLenum	magFilter - magnification filter to apply (default: GL_LINEAR)
GLenum	wrapS - wrapping to apply to S coordinate (default: GL_REPEAT)
GLenum	wrapT - wrapping to apply to T coordinate (default: GL_REPEAT)

Returns

GLuint - texture handle corresponding to the texture

8.11.2.2 loadAndRegisterTexture()

```
GLuint CSCI441::TextureUtils::loadAndRegisterTexture (
            const char * filename,
            GLenum minFilter = GL_LINEAR,
            GLenum magFilter = GL_LINEAR,
            GLenum wrapS = GL_REPEAT,
            GLenum wrapT = GL_REPEAT )  [inline]
```

loads and registers a texture into memory returning a texture handle

Equivalent to loadAndRegister2DTexture()

8.11.2.3 loadBMP()

```
bool CSCI441::TextureUtils::loadBMP (
            const char * filename,
            int & imageWidth,
            int & imageHeight,
            int & imageChannels,
            unsigned char * imageData,
            const char * path = "./" )  [inline]
```

loads a BMP into memory

This function reads an ASCII BMP, returning true if the function succeeds and false if it fails. If it succeeds, the variables imageWidth and imageHeight will hold the width and height of the read image, respectively.

It's not terribly robust.

Returns the image as an unsigned character array containing imageWidth∗imageHeight∗3 entries (for that many bytes of storage).

NOTE: this function expects imageData to be UNALLOCATED, and will allocate memory itself. If the function fails (returns false), imageData will be set to NULL and any allocated memory will be automatically deallocated.

Parameters

in	*const*	char∗ filename - filename of the image to load
out	*int*	&imageWidth - will contain the image width upon successful completion
out	*int*	&imageHeight - will contain the image height upon successful completion
out	*unsigned*	char∗ &imageData - will contain the RGB data upon successful completion
in	*const*	char∗ path - path to where file is stored. defaults to current directory

Precondition

imageData is unallocated

Returns

bool - true if loading succeeded, false otherwise

8.11.2.4 loadPPM()

```
bool CSCI441::TextureUtils::loadPPM (
            const char * filename,
            int & imageWidth,
            int & imageHeight,
            unsigned char *& imageData )  [inline]
```

loads a PPM into memory

This function reads an ASCII PPM, returning true if the function succeeds and false if it fails. If it succeeds, the variables imageWidth and imageHeight will hold the width and height of the read image, respectively.

It's not terribly robust.

Returns the image as an unsigned character array containing imageWidth∗imageHeight∗3 entries (for that many bytes of storage).

NOTE: this function expects imageData to be UNALLOCATED, and will allocate memory itself. If the function fails (returns false), imageData will be set to NULL and any allocated memory will be automatically deallocated.

Parameters

in	*const*	char ∗filename - filename of the image to load
out	*int*	&imageWidth - will contain the image width upon successful completion
out	*int*	&imageHeight - will contain the image height upon successful completion
out	*unsigned*	char∗ &imageData - will contain the RGB data upon successful completion

Precondition

imageData is unallocated

Returns

bool - true if loading succeeded, false otherwise

8.11.2.5 loadTGA()

```
bool CSCI441::TextureUtils::loadTGA (
            const char * filename,
            int & imageWidth,
            int & imageHeight,
            unsigned char *& imageData,
            int & imageChannels )  [inline]
```

loads a TGA into memory

This function reads an ASCII TGA, returning true if the function succeeds and false if it fails. If it succeeds, the variables imageWidth and imageHeight will hold the width and height of the read image, respectively.

It's not terribly robust.

Returns the image as an unsigned character array containing imageWidth∗imageHeight∗3 entries (for that many bytes of storage).

NOTE: this function expects imageData to be UNALLOCATED, and will allocate memory itself. If the function fails (returns false), imageData will be set to NULL and any allocated memory will be automatically deallocated.

Parameters

in	*const*	char ∗filename - filename of the image to load
out	*int*	&imageWidth - will contain the image width upon successful completion
out	*int*	&imageHeight - will contain the image height upon successful completion
out	*unsigned*	char∗ &imageData - will contain the RGB data upon successful completion
out	*int*	&imageChannels - will contain the number of channels in the image upon successful completion

Precondition

imageData is unallocated

Returns

bool - true if loading succeeded, false otherwise

Index