Forest Lifecycle Simulator

8

Generated by Doxygen 1.9.1

Chapter 1

ComputerGraphics - Forest Lifecycle Simulator

For course work for a class at Grand Canyon University

By Benjamin Carter and Josh Canode.

Watch Video Here

1.0.1 Requirements:

1. Requires OpenGL
2. GLU
3. GLUT

1.0.2 To build, use make

1. First Clone the git repository
   - this can be found at `https://github.com/BenRobotics101/ComputerGraphics`
   - or in linux can be cloned by running `git clone https://github.com/BenRobotics101/ComputerGraphics.git`
   - you can run `git checkout main` to make sure you are in the main branch.
2. Make sure the working directory is the root of the repository.
3. Run `make`.
4. The program will be `Program` under the root of the repository. Run it with `./Program`

1.0.3 Documentation

You can look at the `refman.pdf` or the `index.html` file under `doxygenOutput/html` for an interactive documentation page on the project.

1.0.4 File structure:

File Structure:

- src/
  - This is where the source files are
- include/
  - This is where the header files are
- Project8.docx/.pdf
  - The report
- Makefile
  - The makefile
- Flowchart.drawio.png
  - The flowchart
- ForestStateGraph.drawio.png
  - A map of the different forest states.

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 ForestAnimationSettings Struct Reference

The constants for the forest.
`#include <Trees.h>`

Public Attributes

- float TREE_BLANK_SIT_VACANT_UNIT_TIME = 20.0f

  This is the length of time that an empty cell will try to stay empty (with it being a linear % ramp up)
- float TREE_NEW_GROW_FLAT_RATE = 0.00004f

  This is the percent that a new tree will grow from an empty cell per time unit.
- float TREE_NEW_GROW_NEIGHBOR_RATE = 0.0025f

  This is the increase in percent that an empty spot will receive for every alive neighbor around it.
- float TREE_INITIAL_ALIVE = 0.60f

  This is the initial % of trees that are alive in the forest.
- float TREE_INITIAL_AGE = 50.0f

  This is the initial age of the trees at the start of the program.
- float ANIMATION_GROW_SPEED_SEC = 0.75f

  This is the "grow" animation in seconds.
- float TREE_MINIMUM_FLAMABILITY = 0.20f

  This is the minimal % a tree starts with in likelihood of catching fire when receiving a spark.
- float TREE_AGE_FLAMABILITY_FACTOR = 0.022f

  This is how much % the flammability increases per unit time.
- float TREE_SPONTANEOUS_IGNITE_BASE = 0.001f

  This is the % that a tree will ignite (lightning)
- float TREE_SPONTANEOUS_IGNITE_PER_ALIVE_TREES = 0.0001f

  This is how much the spontaneous percent will increase for every alive tree on the map.
- float TREE_SPONTANEOUS_FIRE_START_DIVISOR = 10.0

  For every fire started, the percent is divided by this amount to decrease the number of concurrent fires on the map.
- float TREE_NEIGHBOR_BURN_FACTOR = 0.23f

  This is % chance that a tree will receive a spark from a neighbor. Two neighbors equals double this.
- float TREE_UPHILL_BURN_BONUS = 0.1f

  If the neighboring tree is on fire and of lower elevation, increase by this uphill bonus % per 1 unit in y direction.
- float TREE_BRUN_LENGTH_START = 10.0f

  The length of time a tree is on fire (in units)
- float TREE_BURN_LENGTH_END = 0.8f

  The minimum length of time a tree is on fire (in units)
- float TREE_BURN_LENGTH_AGE_FACTOR = -0.1f

The length decreases by this amount of units per unit in age.

- float TREE_SIT_VACANT_UNIT_TIME = 20.0f

  A burned stump attempts to stay as is for this amount of time (linearly ramping up %)
- float TREE_REGROW_FLAT_RATE = 0.001f

  Regrow percentage minimum.
- float TREE_REGROW_NEIGHBOR_RATE = 0.15f

  For each neighbor alive, increment grow % by this much.
- float TREE_COMPLETE_DEATH_RATE = 0.04f

  Complete death of cell % rate.
- float LIGHTNING_FRAME_COUNT = 10.0f

  Length of lightning animation in frames.
- float LIGHTNING_HEIGHT = 20.0f

  Lightning top y-value.

4.1.1 Detailed Description

The constants for the forest.
The documentation for this struct was generated from the following file:

- include/Trees.h

4.2 Mesh Class Reference

The Mesh class creates "ground" terrain given starting x, y, and a function.
`#include <Mesh.h>`

Public Member Functions

- Mesh ()

  Empty constructor. Should not be used. Exists only for the "new" keyword.
- Mesh (float division, float start, float end, float floor, float(∗yFunc)(float, float))

  Construct a new Mesh.
- ∼Mesh ()

  Destroy the Mesh object.
- void setup ()

  Setup the mesh. This creates the mesh and performs the calculations. This MUST be called before further use.
- glm::vec3 ∗ getTopPoints ()

  Get an array of vec3 of the top square points (for drawing)
- int numberTopPoints ()

  Return the number of elements in the getTopPoints() array.
- glm::vec3 ∗ getCubePoints ()

  Get an array of vec3 of the center of the top of each cell.
- int numberCubePoints ()

  Return the number of elements in the getCubePoints() array.
- glm::vec3 ∗ getSidePoints ()

  Get an array of vec3 of all the side points in drawing order.
- int numberSidePoints ()

  Return the number of points in the getSidePoints() array.
- float getSquareWidth ()

  Get the width of a cell.

Static Public Member Functions

- static float mapF (float x, float in_min, float in_max, float out_min, float out_max)

  Basic linear mapping function.
- static void mixColor (float change[3], float x, int r, int g, int b, int r2, int g2, int b2)

  The maping function, but interpolates between two different colors.

4.2.1 Detailed Description

The Mesh class creates "ground" terrain given starting x, y, and a function.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Mesh()

```
Mesh::Mesh (
            float division,
            float start,
            float end,
            float floor,
            float(*)(float, float) yFunc )
```

Construct a new Mesh.

Parameters

division	- How many "squares" per side of the mesh?
start	- start at what x value (x and z both start here)
end	- end at what x value (x and z both end here)
floor	- the floor y value that the rectangles are drawn to
yFunc	- a function pointer that returns the y value given the x,z coordinate.

4.2.3 Member Function Documentation

4.2.3.1 getCubePoints()

```
glm::vec3 * Mesh::getCubePoints ( )
```

Get an array of vec3 of the center of the top of each cell.

Returns

glm::vec3*

4.2.3.2 getSidePoints()

```
glm::vec3 * Mesh::getSidePoints ( )
```

Get an array of vec3 of all the side points in drawing order.

Returns

glm::vec3*

4.2.3.3 getSquareWidth()

```
float Mesh::getSquareWidth ( )
```

Get the width of a cell.

Returns

float

4.2.3.4 getTopPoints()

```
glm::vec3 * Mesh::getTopPoints ( )
```

Get an array of vec3 of the top square points (for drawing)

Returns

glm::vec3∗

4.2.3.5 mapF()

```
static float Mesh::mapF (
            float x,
            float in_min,
            float in_max,
            float out_min,
            float out_max )  [inline], [static]
```

Basic linear mapping function.

Parameters

x	
in_min	
in_max	
out_min	
out_max	

Returns

float

4.2.3.6 mixColor()

```
static void Mesh::mixColor (
            float change[3],
            float x,
            int r,
            int g,
            int b,
            int r2,
            int g2,
            int b2 )  [inline], [static]
```

The maping function, but interpolates between two different colors.

Parameters

change	

Parameters

x	- 0 to 1. 0 is first color, 1 is second.
r	
g	
b	
r2	
g2	
b2	

4.2.3.7 numberCubePoints()

`int Mesh::numberCubePoints ( )`
Return the number of elements in the getCubePoints() array.

Returns

int

4.2.3.8 numberSidePoints()

`int Mesh::numberSidePoints ( )`
Return the number of points in the getSidePoints() array.

Returns

int

4.2.3.9 numberTopPoints()

`int Mesh::numberTopPoints ( )`
Return the number of elements in the getTopPoints() array.

Returns

int

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

- include/Mesh.h
- src/Mesh.cpp

4.3 TextWriter Class Reference

The StringDraw class. Draws a string to a window.
`#include <TextWriter.h>`

Public Member Functions

- TextWriter (void *font, int screenWidth, int screenHeight)

  Construct a Text Writer object.
- void write (float x, float y, const char *string)

  Write the string onto the screen at given point. Point is in normalized -1.0 to 1.0 coordinates, with center being 0,0.
- void close ()

  Reset and cleanup after glut operations.

4.3.1 Detailed Description

The StringDraw class. Draws a string to a window.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 TextWriter()

```
TextWriter::TextWriter (
            void * font,
            int screenWidth,
            int screenHeight )
```

Construct a Text Writer object.

Parameters

font	- The font to use. Use GLUT fonts.
screenWidth	- The screen width
screenHeight	- The screen height

4.3.3 Member Function Documentation

4.3.3.1 write()

```
void TextWriter::write (
            float x,
            float y,
            const char * string )
```

Write the string onto the screen at given point. Point is in normalized -1.0 to 1.0 coordinates, with center being 0,0.

Parameters

x	
y	
string	

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

- include/TextWriter.h
- src/TextWriter.cpp

4.4 Tree Class Reference

The Tree class manages what each cell does, broken up into four states.
`#include <Trees.h>`

Public Member Functions

- Tree ()

  Construct a new Tree object. Do not use! Only for mem allocations.
- Tree (glm::vec3 position, glm::vec3 dimensions, ForestAnimationSettings *forest, float *fps, float unitsPer↩ Second, float *unitClockI)

Construct a new Tree cell.

- void lightningStrike (glm::vec3 position, glm::vec3 dimensions)

  Draw Lightning Strike.
- void setNeighborData (Tree **neighborTrees, int numberOFneighbors)

  Update the tree with pointers to its neighbors. This must be called before anything else!
- void incrementAge (float t)

  Increment the age of the tree (already automatically internally increments age, but this is for extra aging)
- void draw ()

  Draw the tree.
- bool randomIF (float percent)

  Return true or false randomly based on a percent. FPS adjusted.
- int getStatus ()

  Get the status of the tree. 0 - no tree. 1 - alive tree. 2 - burning tree. 3 - burned tree.
- float getElevation ()

  Get the Elevation of the tree.
- void simulate ()

  Simulate the tree.

Static Public Attributes

- static int **numberOfTreesAlive** = 0
- static int **numberOfTreesBurning** = 0
- static int **numberOfNoTrees** = 0
- static int **numberOfTreesBurned** = 0
- static int **numberOfFireStarts** = 0

4.4.1 Detailed Description

The Tree class manages what each cell does, broken up into four states.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Tree()

```
Tree::Tree (
            glm::vec3 position,
            glm::vec3 dimensions,
            ForestAnimationSettings * forest,
            float * fps,
            float unitsPerSecond,
            float * unitClockI )
```

Construct a new Tree cell.

Parameters

position	- the center position of the cell
dimensions	- the dimensions of the cell
forest	- the forest settings
fps	- a pointer to the FPS counter
unitsPerSecond	- units per second.
unitClockI	- a pointer to the unit counter.

4.4.3 Member Function Documentation

4.4.3.1 getElevation()

```
float Tree::getElevation ( )
```

Get the Elevation of the tree.

Returns

float

4.4.3.2 getStatus()

```
int Tree::getStatus ( )
```

Get the status of the tree. 0 - no tree. 1 - alive tree. 2 - burning tree. 3 - burned tree.

Returns

int

4.4.3.3 incrementAge()

```
void Tree::incrementAge (
            float t )
```

Increment the age of the tree (already automatically internally increments age, but this is for extra aging)

Parameters

t	- time unit

4.4.3.4 lightningStrike()

```
void Tree::lightningStrike (
            glm::vec3 position,
            glm::vec3 dimensions )
```

Draw Lightning Strike.

Parameters

position	
dimensions	

4.4.3.5 randomIF()

```
bool Tree::randomIF (
            float percent )
```

Return true or false randomly based on a percent. FPS adjusted.

Parameters

percent	

Returns

true

false

4.4.3.6 setNeighborData()

```
void Tree::setNeighborData (
            Tree ** neighborTrees,
            int numberOFneighbors )
```

Update the tree with pointers to its neighbors. This must be called before anything else!

Parameters

neighborTrees	
numberOFneighbors	

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

- include/Trees.h
- src/Trees.cpp

Chapter 5

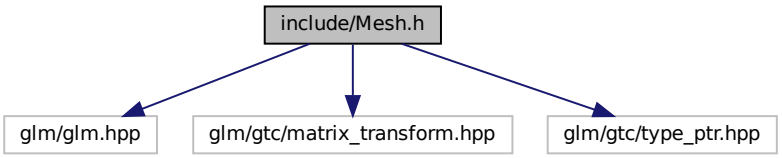
File Documentation

5.1 include/Mesh.h File Reference

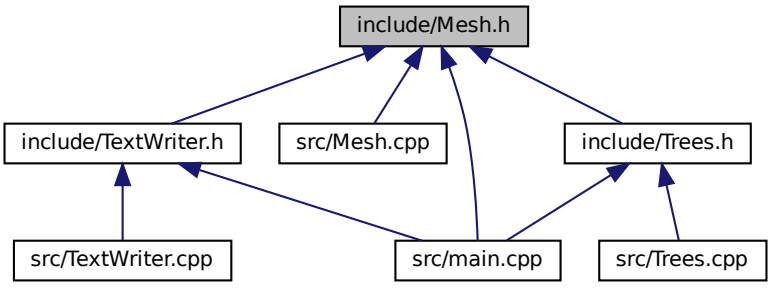
This holds the Mesh class definitions.

```
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp>
```

Include dependency graph for Mesh.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Mesh

  The Mesh class creates "ground" terrain given starting x, y, and a function.

5.1.1 Detailed Description

This holds the Mesh class definitions.

Author

Benjamin Carter and Josh Canode

Version

1.0

Date

2023-11-18

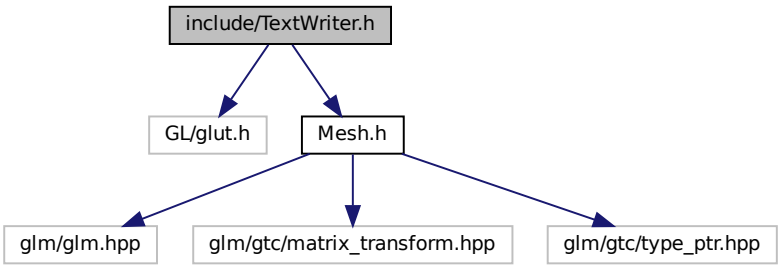
Copyright

5.2 include/TextWriter.h File Reference

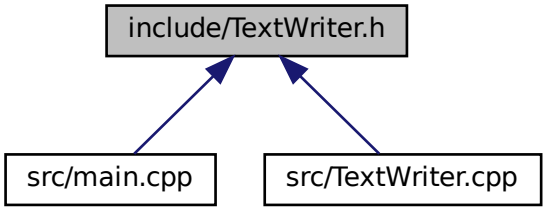
This holds the Text Writer Definitions.

```
#include <GL/glut.h>
#include "Mesh.h"
```

Include dependency graph for TextWriter.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TextWriter

  The StringDraw class. Draws a string to a window.

5.2.1 Detailed Description

This holds the Text Writer Definitions.

Author

Benjamin Carter and Josh Canode

Version

1.0

Date

2023-11-18

Copyright

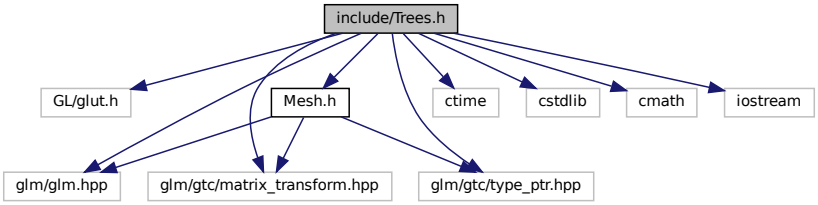
Copyright (c) 2023

5.3 include/Trees.h File Reference

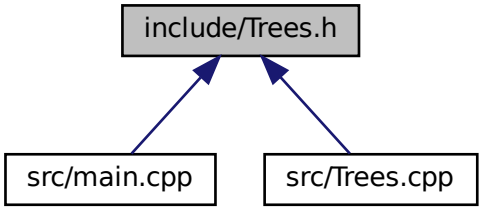
This holds the tree class definitions.

```
#include <GL/glut.h>
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp>
#include <ctime>
#include <cstdlib>
#include <cmath>
#include <iostream>
#include "Mesh.h"
```

Include dependency graph for Trees.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct ForestAnimationSettings

  The constants for the forest.
- class Tree

  The Tree class manages what each cell does, broken up into four states.

Macros

- #define **CC**(arg) (arg / 255.0f)

5.3.1 Detailed Description

This holds the tree class definitions.

Author

Benjamin Carter and Josh Canode

Version

1.0

Date

2023-11-18

Copyright

Copyright (c) 2023

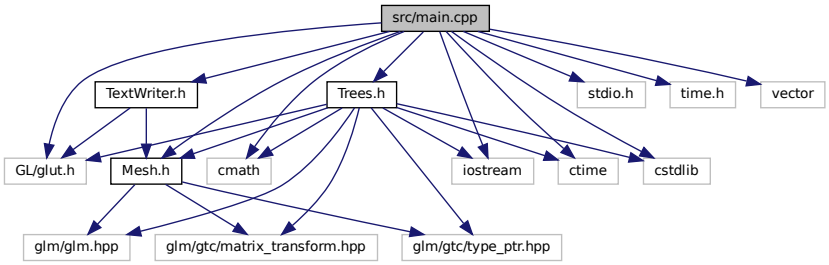
5.4 src/main.cpp File Reference

The Main Program. Sets the meshes/trees and contains the GLUT windowing functions.

```
#include <GL/glut.h>
#include <cmath>
#include <iostream>
#include "Mesh.h"
#include <stdio.h>
#include <time.h>
#include <ctime>
#include <cstdlib>
```

```
#include "TextWriter.h"
#include "Trees.h"
#include <vector>
```

Include dependency graph for main.cpp:



Macros

- #define **CC**(arg) (arg / 255.0f)
- #define **TARGET_FPS** 120.0f
- #define **MESH_DIVISIONS** 40
- #define **MESH_START** -12
- #define **MESH_END** 12
- #define **MESH_DEPTH** -3
- #define **MESH_TRANSLATE_X** 0
- #define **MESH_TRANSLATE_Z** 0

Functions

- float groundFunction (float x, float z)

  Ground Function. Returns a y value for every x,z coordinate.
- void reshape (int width, int height)

  Run when the window is reshaped.
- void renderAxes ()

  Render axes lines for debugging.
- void renderGround ()

  Render the ground mesh.
- void renderCamera (void)

  Render the camera.
- void frame ()

  The draw loop. Render everything on screen.
- void timer (int a)

  The timer function. This calculates FPS and triggers redraw events.
- void idleFunction ()

  This runs during "downtime". Updates the writeoutFPS counter.
- void get_neighbors (int i, std::vector< Tree * > &foundNeighbors, Tree **allTrees, int allTreesLength)

  Calculate neighbors for trees. Operates "soft-returns" on the foundNeighbors vector object.
- void setupCalculations ()

  Initialize the trees and the mesh.
- int main (int argc, char **argv)

  Main program.

Variables

- float **UNITS_PER_SECOND** = 0.8f
- Mesh **groundMesh** = Mesh()
- glm::vec4 **camPos**
- unsigned int **frameCount** = 0
- float **currentTime** = 0
- float **timeFPSOffset** = 0
- float **fps** = 0.0f
- float **writeoutFPS** = fps
- float **unitCounter** = 0.0f
- int **screenHeight** = 0
- int **screenWidth** = 0
- ForestAnimationSettings **forestSettings**
- float **prevCamY** = 0.0
- Tree ** **allTrees**

5.4.1 Detailed Description

The Main Program. Sets the meshes/trees and contains the GLUT windowing functions.

Author

Benjamin Carter and Josh Canode

Version

1.0

Date

2023-11-18

Copyright

Copyright (c) 2023

5.4.2 Function Documentation

5.4.2.1 get_neighbors()

```
void get_neighbors (
            int i,
            std::vector< Tree * > & foundNeighbors,
            Tree ** allTrees,
            int allTreesLength )
```

Calculate neighbors for trees. Operates "soft-returns" on the foundNeighbors vector object.

Parameters

i	
foundNeighbors	
allTrees	
allTreesLength	

5.4.2.2 groundFunction()

```
float groundFunction (
            float x,
            float z )
```

Ground Function. Returns a y value for every x,z coordinate.

Parameters

x	
z	

Returns

float

5.4.2.3 main()

```
int main (
            int argc,
            char ** argv )
```

Main program.

Parameters

argc	- number of arguments
argv	- array of arguments

Returns

int

5.4.2.4 reshape()

```
void reshape (
            int width,
            int height )
```

Run when the window is reshaped.

Parameters

width	
height	

5.4.2.5 timer()

```
void timer (
            int a )
```

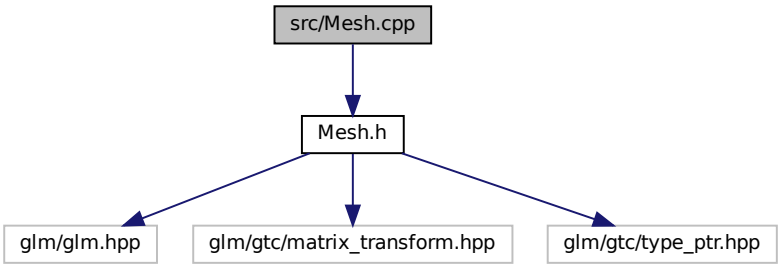
The timer function. This calculates FPS and triggers redraw events.

Parameters

a	

5.5 src/Mesh.cpp File Reference

The Mesh Class Source File. Creates a mesh terrain.
`#include "Mesh.h"`
Include dependency graph for Mesh.cpp:



5.5.1 Detailed Description

The Mesh Class Source File. Creates a mesh terrain.

Author

Benjamin Carter and Josh Canode

Version

1.0

Date

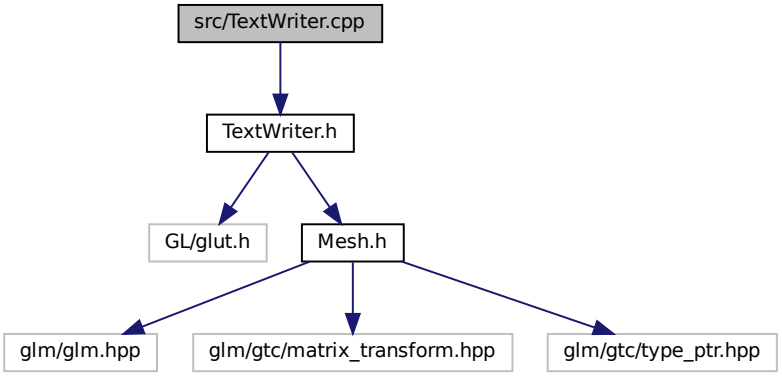
2023-11-18

Copyright

Copyright (c) 2023

5.6 src/TextWriter.cpp File Reference

Text Writer Class. Writes text to a screen using GLUT primitatives.
`#include "TextWriter.h"`
Include dependency graph for TextWriter.cpp:



5.6.1 Detailed Description

Text Writer Class. Writes text to a screen using GLUT primitatives.

Author

Benjamin Carter and Josh Canode

Version

1.0

Date

2023-11-18

Copyright

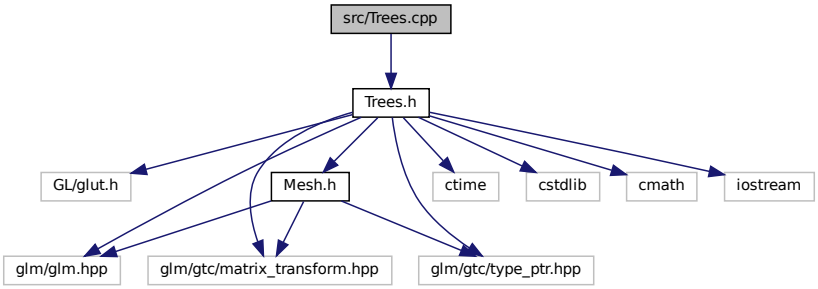
Copyright (c) 2023

5.7 src/Trees.cpp File Reference

Tree class source file.

```
#include "Trees.h"
```

Include dependency graph for Trees.cpp:



5.7.1 Detailed Description

Tree class source file.

Author

Benjamin Carter and Josh Canode

Version

1.0

Date

2023-11-18

Copyright

Copyright (c) 2023

Index