

EvolutionProject

1.9.953

Generated by Doxygen 1.8.14

Contents

1	Evolution WildFire	1
1.1	Introduction	1
1.2	Usage	1
2	Todo List	3
3	Bug List	5
4	Namespace Index	7
4.1	Namespace List	7
5	Hierarchical Index	9
5.1	Class Hierarchy	9
6	Class Index	11
6.1	Class List	11
7	File Index	13
7.1	File List	13

8 Namespace Documentation	15
8.1 AutoVersion Namespace Reference	15
8.1.1 Variable Documentation	15
8.1.1.1 BUILD	15
8.1.1.2 BUILD_HISTORY	15
8.1.1.3 BUILDS_COUNT	16
8.1.1.4 DATE	16
8.1.1.5 FULLVERSION_STRING	16
8.1.1.6 MAJOR	16
8.1.1.7 MINOR	16
8.1.1.8 MONTH	16
8.1.1.9 REVISION	16
8.1.1.10 STATUS	16
8.1.1.11 STATUS_SHORT	17
8.1.1.12 UBUNTU_VERSION_STYLE	17
8.1.1.13 YEAR	17
8.2 ConfigValueConverters Namespace Reference	17
8.2.1 Detailed Description	17
8.2.2 Function Documentation	18
8.2.2.1 getValue_()	18
8.2.2.2 getValue_< bool >()	18
8.2.2.3 getValue_< double >()	19
8.2.2.4 getValue_< int >()	19
8.2.2.5 getValue_< std::string >()	20
8.3 glutCB Namespace Reference	20
8.3.1 Detailed Description	21
8.3.2 Function Documentation	21
8.3.2.1 callMouse()	21
8.3.2.2 changeSize()	22
8.3.2.3 keyPressed()	22

8.3.2.4	keyUp()	23
8.3.2.5	mouseMove()	23
8.3.2.6	passiveMouse()	24
8.3.2.7	pressSpecialKey()	24
8.3.2.8	releaseSpecialKey()	25
8.3.2.9	renderScene()	25
8.3.2.10	update()	26
8.4	utility Namespace Reference	26
8.4.1	Detailed Description	27
8.4.2	Function Documentation	27
8.4.2.1	getCurrentDate()	27
8.4.2.2	getCurrentTime()	27
8.4.2.3	numToStr()	28
8.4.2.4	replaceCharSet()	28
8.4.2.5	replaceString()	29
8.4.2.6	split()	29
8.4.2.7	toLowerCase()	30
8.4.2.8	toUpperCase()	30
9	Class Documentation	31
9.1	Audio Class Reference	31
9.1.1	Detailed Description	31
9.1.2	Constructor & Destructor Documentation	32
9.1.2.1	Audio()	32
9.1.2.2	~Audio()	32
9.1.3	Member Function Documentation	32
9.1.3.1	clearStoppedSounds()	32
9.1.3.2	playSound() [1/3]	32
9.1.3.3	playSound() [2/3]	33
9.1.3.4	playSound() [3/3]	33
9.1.4	Member Data Documentation	33

9.1.4.1	AUDIO_DIRECTORY	33
9.1.4.2	COULD_NOT_LOAD_MUSIC_MESSAGE	33
9.1.4.3	COULD_NOT_LOAD_SOUND_MESSAGE	33
9.1.4.4	music	34
9.1.4.5	soundBuffers	34
9.1.4.6	sounds	34
9.2	Camera Struct Reference	34
9.2.1	Detailed Description	35
9.2.2	Constructor & Destructor Documentation	35
9.2.2.1	Camera()	35
9.2.3	Member Data Documentation	35
9.2.3.1	ang	35
9.2.3.2	DEFAULT_HEIGHT	35
9.2.3.3	DEFAULT_R_SPEED	36
9.2.3.4	DEFAULT_T_SPEED	36
9.2.3.5	del	36
9.2.3.6	dir	36
9.2.3.7	mov	36
9.2.3.8	pos	36
9.2.3.9	rotationSpeed	37
9.2.3.10	translationSpeed	37
9.3	Config Class Reference	37
9.3.1	Detailed Description	38
9.3.2	Constructor & Destructor Documentation	38
9.3.2.1	~Config()	38
9.3.2.2	Config()	38
9.3.3	Member Function Documentation	38
9.3.3.1	getFileValue()	38
9.3.3.2	getValue()	39
9.3.4	Member Data Documentation	40

9.3.4.1	CONFIG_FILE	40
9.4	DrawCircle< A > Struct Template Reference	40
9.4.1	Constructor & Destructor Documentation	41
9.4.1.1	DrawCircle() [1/2]	41
9.4.1.2	DrawCircle() [2/2]	42
9.4.2	Member Function Documentation	42
9.4.2.1	draw()	42
9.4.3	Member Data Documentation	42
9.4.3.1	dimension	43
9.5	Drawing Struct Reference	43
9.5.1	Member Enumeration Documentation	44
9.5.1.1	Dimension	44
9.5.2	Member Function Documentation	44
9.5.2.1	changeColor() [1/2]	44
9.5.2.2	changeColor() [2/2]	44
9.5.2.3	changeTexture() [1/2]	45
9.5.2.4	changeTexture() [2/2]	45
9.5.2.5	enable2D()	46
9.5.2.6	enable3D()	46
9.5.2.7	enableND()	46
9.5.2.8	isColor()	47
9.5.2.9	isTexture()	47
9.5.3	Member Data Documentation	48
9.5.3.1	INVALID_COLOR_MESSAGE	48
9.5.3.2	INVALID_DRAWING_ORDER_MESSAGE	48
9.5.3.3	INVALID_TEXTURE_MESSAGE	48
9.5.3.4	UNKNOWN_APPROVED_COLOR_MESSAGE	48
9.5.3.5	UNKNOWN_APPROVED_TEXTURE_MESSAGE	48
9.5.3.6	UNKNOWN_DIMENSION_MESSAGE	49
9.6	DrawItem< A > Class Template Reference	49

9.6.1	Detailed Description	49
9.6.2	Constructor & Destructor Documentation	50
9.6.2.1	DrawItem()	50
9.6.3	Member Data Documentation	50
9.6.3.1	INVALID_APPEARANCE_MESSAGE	51
9.7	DrawMySpecificObject< A > Struct Template Reference	51
9.7.1	Constructor & Destructor Documentation	52
9.7.1.1	DrawMySpecificObject()	52
9.7.2	Member Function Documentation	52
9.7.2.1	draw()	53
9.7.3	Member Data Documentation	53
9.7.3.1	dimension	53
9.8	DrawPlane< A > Struct Template Reference	53
9.8.1	Constructor & Destructor Documentation	54
9.8.1.1	DrawPlane() [1/6]	54
9.8.1.2	DrawPlane() [2/6]	55
9.8.1.3	DrawPlane() [3/6]	55
9.8.1.4	DrawPlane() [4/6]	55
9.8.1.5	DrawPlane() [5/6]	55
9.8.1.6	DrawPlane() [6/6]	56
9.8.2	Member Function Documentation	56
9.8.2.1	draw()	56
9.8.3	Member Data Documentation	56
9.8.3.1	dimension	57
9.9	DrawRectangle< A > Struct Template Reference	57
9.9.1	Constructor & Destructor Documentation	58
9.9.1.1	DrawRectangle()	58
9.9.2	Member Function Documentation	58
9.9.2.1	draw()	59
9.9.3	Member Data Documentation	59

9.9.3.1 dimension	59
9.10 DrawString< A > Struct Template Reference	59
9.10.1 Constructor & Destructor Documentation	60
9.10.1.1 DrawString()	60
9.10.2 Member Function Documentation	61
9.10.2.1 draw()	61
9.10.3 Member Data Documentation	61
9.10.3.1 dimension	61
9.11 GFramework Class Reference	62
9.11.1 Detailed Description	63
9.11.2 Constructor & Destructor Documentation	63
9.11.2.1 ~GFramework()	64
9.11.2.2 GFramework() [1/2]	64
9.11.2.3 GFramework() [2/2]	64
9.11.3 Member Function Documentation	65
9.11.3.1 initializeGlut()	65
9.11.3.2 loadColors()	65
9.11.3.3 loadTextures()	66
9.11.3.4 operator=()	66
9.11.3.5 showScene()	66
9.11.3.6 startup()	67
9.11.4 Member Data Documentation	67
9.11.4.1 audio	67
9.11.4.2 camera	67
9.11.4.3 colorMap	67
9.11.4.4 display	68
9.11.4.5 drawingState	68
9.11.4.6 FPS	68
9.11.4.7 get	68
9.11.4.8 INIT_WINDOW_HEIGHT	68

9.11.4.9	INIT_WINDOW_WIDTH	68
9.11.4.10	INIT_WINDOW_X	69
9.11.4.11	INIT_WINDOW_Y	69
9.11.4.12	mouse	69
9.11.4.13	RENDERING_DISTANCE	69
9.11.4.14	simulation	69
9.11.4.15	textureMap	69
9.11.4.16	userInput	70
9.11.4.17	WINDOW_TITLE	70
9.11.4.18	windowSize	70
9.12	Logger Class Reference	70
9.12.1	Detailed Description	71
9.12.2	Constructor & Destructor Documentation	72
9.12.2.1	~Logger()	72
9.12.2.2	Logger() [1/2]	72
9.12.2.3	Logger() [2/2]	72
9.12.3	Member Function Documentation	72
9.12.3.1	get()	72
9.12.3.2	getTimeStamp()	73
9.12.3.3	log()	73
9.12.3.4	logMessage()	74
9.12.3.5	normalExit()	75
9.12.3.6	operator=()	76
9.12.3.7	toString() [1/2]	76
9.12.3.8	toString() [2/2]	76
9.12.4	Member Data Documentation	76
9.12.4.1	DEFAULT_DEGREE	76
9.12.4.2	DEFAULT_TYPE	76
9.12.4.3	FAILED_TO_OPEN_AFTER_FAILURE_MESSAGE	77
9.12.4.4	LOG_FILE_TITLE	77

9.12.4.5	logFile	77
9.12.4.6	logs	77
9.12.4.7	OPENED_LOG_AFTER_FAILURE_MESSAGE	77
9.12.4.8	PROGRAM_EXIT_MESSAGE	77
9.12.4.9	UNKNOWN_LOG_DEGREE_MESSAGE	78
9.13	Mouse Struct Reference	78
9.13.1	Detailed Description	78
9.13.2	Constructor & Destructor Documentation	78
9.13.2.1	Mouse()	78
9.13.3	Member Data Documentation	79
9.13.3.1	clicked	79
9.13.3.2	heldDown	79
9.13.3.3	x	79
9.13.3.4	y	79
9.14	Simulation Class Reference	79
9.14.1	Constructor & Destructor Documentation	80
9.14.1.1	Simulation()	80
9.14.1.2	~Simulation()	80
9.14.2	Member Function Documentation	80
9.14.2.1	init()	81
9.14.2.2	loadGameModeKeyboard()	81
9.14.2.3	run()	81
9.14.2.4	setGameMode()	82
9.14.2.5	setInputType()	82
9.14.2.6	setInputTypeKeyboard()	82
9.14.3	Member Data Documentation	83
9.14.3.1	gameMode	83
9.14.3.2	INITIAL_GAME_MODE	83
9.14.3.3	INITIAL_INPUT_TYPE	83
9.14.3.4	inputType	83

9.14.3.5 UNKNOWN_GAME_MODE_MESSAGE	83
9.14.3.6 UNKNOWN_INPUT_TYPE_MESSAGE	83
9.15 UserFunction Struct Reference	84
9.15.1 Constructor & Destructor Documentation	84
9.15.1.1 UserFunction() [1/4]	84
9.15.1.2 UserFunction() [2/4]	84
9.15.1.3 UserFunction() [3/4]	84
9.15.1.4 UserFunction() [4/4]	85
9.15.1.5 ~UserFunction()	85
9.15.2 Member Data Documentation	85
9.15.2.1 action	85
9.15.2.2 key	85
9.15.2.3 release	85
9.15.2.4 specialKey	85
9.16 UserInput Class Reference	86
9.16.1 Constructor & Destructor Documentation	86
9.16.1.1 UserInput()	86
9.16.1.2 ~UserInput()	86
9.16.2 Member Function Documentation	86
9.16.2.1 drawUserString()	87
9.16.2.2 isInputStringSubmitted()	87
9.16.2.3 setToDefault()	87
9.16.2.4 submitInputString()	87
9.16.3 Member Data Documentation	88
9.16.3.1 functions	88
9.16.3.2 inputString	88
9.16.3.3 keyInputsHeld	88
9.16.3.4 keyStates	88
9.16.3.5 NUM_KEYS	88
9.16.3.6 TERMINATING_CHAR	88

10 File Documentation	89
10.1 Audio.cpp File Reference	89
10.2 Audio.h File Reference	89
10.3 Config.cpp File Reference	90
10.4 Config.h File Reference	91
10.5 Draw.cpp File Reference	93
10.6 Draw.h File Reference	93
10.6.1 Enumeration Type Documentation	95
10.6.1.1 Appearance	95
10.7 GFramework.cpp File Reference	98
10.8 GFramework.h File Reference	99
10.9 GlutCallbacks.cpp File Reference	100
10.10GlutCallbacks.h File Reference	102
10.11LoadColors.cpp File Reference	103
10.12LoadTextures.cpp File Reference	103
10.13Logger.cpp File Reference	104
10.14Logger.h File Reference	105
10.14.1 Macro Definition Documentation	106
10.14.1.1 LOG	106
10.14.1.2 NORMAL_EXIT	106
10.14.2 Enumeration Type Documentation	107
10.14.2.1 LogDegree	107
10.14.2.2 LogType	107
10.15main.cpp File Reference	108
10.15.1 Function Documentation	109
10.15.1.1 main()	109
10.16MyGlut.cpp File Reference	109
10.16.1 Function Documentation	110
10.16.1.1 glLoadTexture()	110
10.16.1.2 glTexVert2f() [1/4]	110

10.16.1.3 glTexVert2f() [2/4]	111
10.16.1.4 glTexVert2f() [3/4]	112
10.16.1.5 glTexVert2f() [4/4]	113
10.16.1.6 glTexVert3f() [1/4]	113
10.16.1.7 glTexVert3f() [2/4]	114
10.16.1.8 glTexVert3f() [3/4]	115
10.16.1.9 glTexVert3f() [4/4]	116
10.16.1.10 saveScreenShot()	116
10.17 MyGlut.h File Reference	117
10.17.1 Detailed Description	118
10.17.2 Typedef Documentation	119
10.17.2.1 Tex	119
10.17.3 Function Documentation	119
10.17.3.1 glLoadTexture()	119
10.17.3.2 glTexVert2f() [1/4]	119
10.17.3.3 glTexVert2f() [2/4]	120
10.17.3.4 glTexVert2f() [3/4]	121
10.17.3.5 glTexVert2f() [4/4]	121
10.17.3.6 glTexVert3f() [1/4]	122
10.17.3.7 glTexVert3f() [2/4]	123
10.17.3.8 glTexVert3f() [3/4]	123
10.17.3.9 glTexVert3f() [4/4]	124
10.17.3.10 saveScreenShot()	125
10.18 MyMath.cpp File Reference	126
10.19 MyMath.h File Reference	126
10.19.1 Macro Definition Documentation	127
10.19.1.1 PI	127
10.19.2 Typedef Documentation	127
10.19.2.1 Matrix	127
10.19.2.2 Matrix2	128

10.19.2.3 Vec	128
10.19.2.4 Vec2	128
10.20Objects.cpp File Reference	128
10.21Objects.h File Reference	128
10.22Shapes.cpp File Reference	129
10.23Shapes.h File Reference	129
10.24Simulation.cpp File Reference	130
10.25Simulation.h File Reference	131
10.25.1 Enumeration Type Documentation	131
10.25.1.1 GameMode	131
10.25.1.2 InputType	131
10.26Text.cpp File Reference	132
10.27Text.h File Reference	132
10.28UserFunction.cpp File Reference	133
10.29UserFunction.h File Reference	133
10.29.1 Macro Definition Documentation	134
10.29.1.1 BACKSPACE	135
10.29.1.2 ENTER	135
10.29.1.3 ESC	135
10.29.1.4 TAB	135
10.29.2 Typedef Documentation	135
10.29.2.1 Action	135
10.30UserInput.cpp File Reference	136
10.31UserInput.h File Reference	136
10.32utility.cpp File Reference	137
10.33utility.h File Reference	138
10.34version.h File Reference	140
10.34.1 Macro Definition Documentation	140
10.34.1.1 RC_FILEVERSION	140
10.34.1.2 RC_FILEVERSION_STRING	140

Chapter 1

Evolution WildFire

1.1 Introduction

This is a 3D evolution simulator. The hope is that in future, you will be able to control evolved 3D creatures in a dynamic environment. Their evolution will change their morphology and their neural network brain to accomplish the goals you set out for them. This includes fighting other creatures, navigating a complex landscape, and completing meta-challenges to collect evolution points and evolve the best creature!

1.2 Usage

The program can be run immediately once the copy has been obtained by running the .exe in the bin/debug folder. Right now, there is no game play at all, as the software is in development.

Chapter 2

Todo List

Class [Audio](#)

This class does not clear sounds when they are done. This means that the sounds vector will grow unbounded. This is wasteful and should be resolved.

Protect / warn against using stereo sounds in 3D. (not allowed according to SFML).

Class [Camera](#)

In future, the camera implementation should be removed from the graphics class and put here.

Class [Config](#)

Allow changes to config file based on program state.

Class [Logger](#)

Right now, there isn't a way to propagate warning and error messages to the screen. Maybe this could be a variable "lastLogged" or "lastError" or "lastWarning" OR "lastSignificantLog".

Chapter 3

Bug List

Member `glutCB::renderScene ()`

The frame rate has some issues and is not consistent on some other machines.

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

AutoVersion	15
ConfigValueConverters	
This namespace contains the functions which convert values to their respective type	17
glutCB	
This namespace contains the glut callback functions	20
utility	
This namespace contains functions which you might expect to be standard in cpp	26

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Audio	31
Camera	34
Config	37
Drawing	43
DrawItem< A >	49
DrawCircle< A >	40
DrawMySpecificObject< A >	51
DrawPlane< A >	53
DrawRectangle< A >	57
DrawString< A >	59
GFramework	62
Logger	70
Mouse	78
Simulation	79
UserFunction	84
UserInput	86

Chapter 6

Class Index

6.1 Class List

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

Audio	The audio engine	31
Camera	This class describes the camera placed at the head of the user	34
Config	This class allows variables to be loaded from a file	37
DrawCircle< A >	40
Drawing	43
DrawItem< A >	The parent class of items which can drawn	49
DrawMySpecificObject< A >	51
DrawPlane< A >	53
DrawRectangle< A >	57
DrawString< A >	59
GFramework	This class holds all the functions required for a 3D simulator to be run	62
Logger	This class houses the logging functionality	70
Mouse	This class contains information about the user's mouse	78
Simulation	79
UserFunction	84
UserInput	86

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Audio.cpp	89
Audio.h	89
Config.cpp	90
Config.h	91
Draw.cpp	93
Draw.h	93
GFramework.cpp	98
GFramework.h	99
GlutCallbacks.cpp	100
GlutCallbacks.h	102
LoadColors.cpp	103
LoadTextures.cpp	103
Logger.cpp	104
Logger.h	105
main.cpp	108
MyGlut.cpp	109
MyGlut.h	
This file contains some useful extension for glut	117
MyMath.cpp	126
MyMath.h	126
Objects.cpp	128
Objects.h	128
Shapes.cpp	129
Shapes.h	129
Simulation.cpp	130
Simulation.h	131
Text.cpp	132
Text.h	132
UserFunction.cpp	133
UserFunction.h	133
UserInput.cpp	136
UserInput.h	136
utility.cpp	137
utility.h	138
version.h	140

Chapter 8

Namespace Documentation

8.1 AutoVersion Namespace Reference

Variables

- static const char `DATE []` = "24"
- static const char `MONTH []` = "01"
- static const char `YEAR []` = "2018"
- static const char `UBUNTU_VERSION_STYLE []` = "18.01"
- static const char `STATUS []` = "Alpha"
- static const char `STATUS_SHORT []` = "a"
- static const long `MAJOR` = 0
- static const long `MINOR` = 0
- static const long `BUILD` = 0
- static const long `REVISION` = 0
- static const long `BUILDS_COUNT` = 1
- static const char `FULLVERSION_STRING []` = "0.0.0.0"
- static const long `BUILD_HISTORY` = 0

8.1.1 Variable Documentation

8.1.1.1 BUILD

```
const long AutoVersion::BUILD = 0 [static]
```

8.1.1.2 BUILD_HISTORY

```
const long AutoVersion::BUILD_HISTORY = 0 [static]
```

8.1.1.3 BUILDS_COUNT

```
const long AutoVersion::BUILDS_COUNT = 1 [static]
```

8.1.1.4 DATE

```
const char AutoVersion::DATE[] = "24" [static]
```

8.1.1.5 FULLVERSION_STRING

```
const char AutoVersion::FULLVERSION_STRING[] = "0.0.0.0" [static]
```

8.1.1.6 MAJOR

```
const long AutoVersion::MAJOR = 0 [static]
```

8.1.1.7 MINOR

```
const long AutoVersion::MINOR = 0 [static]
```

8.1.1.8 MONTH

```
const char AutoVersion::MONTH[] = "01" [static]
```

8.1.1.9 REVISION

```
const long AutoVersion::REVISION = 0 [static]
```

8.1.1.10 STATUS

```
const char AutoVersion::STATUS[] = "Alpha" [static]
```


8.1.1.11 STATUS_SHORT

```
const char AutoVersion::STATUS_SHORT[] = "a" [static]
```

8.1.1.12 UBUNTU_VERSION_STYLE

```
const char AutoVersion::UBUNTU_VERSION_STYLE[] = "18.01" [static]
```

8.1.1.13 YEAR

```
const char AutoVersion::YEAR[] = "2018" [static]
```

8.2 ConfigValueConverters Namespace Reference

This namespace contains the functions which convert values to their respective type.

Functions

- `template<typename T>`
`T getValue_ (std::string value)`
Templated function to return the value as the specified type T.
- `template<>`
`bool getValue_< bool > (std::string value)`
Template specialization, returns value as boolean.
- `template<>`
`int getValue_< int > (std::string value)`
Template specialization, returns value as boolean.
- `template<>`
`double getValue_< double > (std::string value)`
Template specialization, returns value as boolean.
- `template<>`
`std::string getValue_< std::string > (std::string value)`
Template specialization, returns value as boolean.

8.2.1 Detailed Description

This namespace contains the functions which convert values to their respective type.

In the [Config](#) class, a key is specified which is looked up in the [Config](#) file. In these functions, the value is converted from a `std::string` to the specified template type.

Warning

This should not be used outside of the [Config](#) class.

Note

Ideally these functions would be in the `config` class, however for templating reasons, it must be in a separate namespace.

See also

[Config](#)

8.2.2 Function Documentation

8.2.2.1 `getValue_()`

```
template<typename T >
T ConfigValueConverters::getValue_ (
    std::string value )
```

Templated function to return the value as the specified type T.

Parameters

<i>value</i>	const std::string The value to convert.
--------------	---

Returns

bool The value as a boolean.

See also

[Config::getFileValue](#)

Note

This should only be called by [Config::getValue](#).

8.2.2.2 `getValue_< bool >()`

```
template<>
bool ConfigValueConverters::getValue_< bool > (
    std::string value )
```

Template specialization, returns value as boolean.

Parameters

<i>value</i>	const std::string The value to convert.
--------------	---

Returns

bool The value as a boolean.

See also

[Config::getFileValue](#), [ConfigValueConverters::getValue_](#)

Note

This should only be called by [Config::getValue](#).

8.2.2.3 `getValue_< double >()`

```
template<>
double ConfigValueConverters::getValue_< double > (
    std::string value )
```

Template specialization, returns value as boolean.

Parameters

<i>value</i>	const std::string The value to convert.
--------------	---

Returns

double The value as a double.

See also

[Config::getFileValue](#), [ConfigValueConverters::getValue_](#)

Note

This should only be called by [Config::getValue](#).

8.2.2.4 `getValue_< int >()`

```
template<>
int ConfigValueConverters::getValue_< int > (
    std::string value )
```

Template specialization, returns value as boolean.

Parameters

<i>value</i>	const std::string The value to convert.
--------------	---

Returns

int The value as an integer.

See also

[Config::getFileValue](#), [ConfigValueConverters::getValue_](#)

Note

This should only be called by [Config::getValue](#).

8.2.2.5 `getValue_< std::string >()`

```
template<>
std::string ConfigValueConverters::getValue_< std::string > (
    std::string value )
```

Template specialization, returns value as boolean.

Parameters

<i>value</i>	const std::string The value to convert.
--------------	---

Returns

std::string The value as a std::string.

See also

[Config::getFileValue](#), [ConfigValueConverters::getValue_](#)

Note

This should only be called by [Config::getValue](#).

8.3 glutCB Namespace Reference

This namespace contains the glut callback functions.

Functions

- void [renderScene](#) ()
Prepares the drawing state, then calls the simulation at a constant FPS.
- void [update](#) ()
Updates the audio, window size, userfunctions (ie: calls them), and camera.
- void [changeSize](#) (int w, int h)
Updates the opengl viewport etc on window resize.
- void [callMouse](#) (int button, int state, int mx, int my)
Is called on mouse click.
- void [mouseMove](#) (int mx, int my)
Is called on mouse movement.
- void [passiveMouse](#) (int mx, int my)
Continuously called to update details about the mouse.
- void [keyPressed](#) (unsigned char key, int x, int y)
Sets the state of the [UserInput](#) keystates for the pressed key and calls its associated pressed function.
- void [keyUp](#) (unsigned char key, int x, int y)
Resets the state of the [UserInput](#) keystates for the released key and calls its associated release function.
- void [pressSpecialKey](#) (int key, int kxx, int kyy)
Calls special key (ex: Arrow keys) press functions.
- void [releaseSpecialKey](#) (int key, int kx, int ky)
Calls the special key (ex: Arrow keys) release functions.

8.3.1 Detailed Description

This namespace contains the glut callback functions.

These must be bound functions and therefore cannot appear inside of a class. They are used once in the [GFramework](#) class.

8.3.2 Function Documentation

8.3.2.1 [callMouse\(\)](#)

```
void glutCB::callMouse (
    int button,
    int state,
    int mx,
    int my )
```

Is called on mouse click.

Possible buttons include GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, and GLUT_RIGHT_BUTTON.

Parameters

<i>button</i>	int Which mouse button is clicked.
<i>state</i>	int The state of the click (release, or press).
<i>mx</i>	int The x position of the mouse.
<i>my</i>	int The y position of the mouse.

Returns

void

Warning

Not yet implemented.

8.3.2.2 changeSize()

```
void glutCB::changeSize (
    int w,
    int h )
```

Updates the opengl viewport etc on window resize.

Parameters

<i>w</i>	int The new width of the window.
<i>h</i>	int The new height of the window.

Returns

void

8.3.2.3 keyPressed()

```
void glutCB::keyPressed (
    unsigned char key,
    int x,
    int y )
```

Sets the state of the [UserInput](#) keystates for the pressed key and calls its associated pressed function.

Parameters

<i>key</i>	unsigned char
<i>x</i>	int
<i>y</i>	int

Returns

void

Note

This only works for 'normal', ascii characters.

See also

[pressSpecialKey](#) for handling special characters.

8.3.2.4 keyUp()

```
void glutCB::keyUp (
    unsigned char key,
    int x,
    int y )
```

Resets the state of the [UserInput](#) keystates for the released key and calls its associated release function.

Parameters

<i>key</i>	unsigned char
<i>x</i>	int
<i>y</i>	int

Returns

void

Note

This only works for 'normal', ascii characters.

See also

[pressSpecialKey](#) for handling special characters.

8.3.2.5 mouseMove()

```
void glutCB::mouseMove (
    int mx,
    int my )
```

Is called on mouse movement.

Parameters

<i>mx</i>	int The x position of the mouse.
<i>my</i>	int The y position of the mouse.

Returns

void

Warning

Not yet implemented.

8.3.2.6 passiveMouse()

```
void glutCB::passiveMouse (
    int mx,
    int my )
```

Continuously called to update details about the mouse.

Specifically it update the [GFramework](#) mouse, and resets the [GFramework](#) camera.del in the case that the mouse is used to control the camera.

Parameters

<i>mx</i>	int The x position of the mouse.
<i>my</i>	int The y position of the mouse.

Returns

void

Note

Not fully implemented.

8.3.2.7 pressSpecialKey()

```
void glutCB::pressSpecialKey (
    int key,
    int kxx,
    int kyy )
```

Calls special key (ex: Arrow keys) press functions.

Parameters

<i>key</i>	int The glut key pressed.
<i>kxx</i>	int The x position of the mouse at the time the key is pressed.
<i>kyy</i>	int The y position of the mouse at the time the key is pressed.

Returns

void

See also

<https://www.opengl.org/resources/libraries/glut/spec3/node54.html> for the available special keys.

8.3.2.8 releaseSpecialKey()

```
void glutCB::releaseSpecialKey (  
    int key,  
    int kx,  
    int ky )
```

Calls the special key (ex: Arrow keys) release functions.

Parameters

<i>key</i>	int The glut key released.
<i>kx</i>	int The x position of the mouse at the time the key is pressed.
<i>ky</i>	int The y position of the mouse at the time the key is pressed.

Returns

void

See also

<https://www.opengl.org/resources/libraries/glut/spec3/node54.html> for the available special keys.

8.3.2.9 renderScene()

```
void glutCB::renderScene ( )
```

Prepares the drawing state, then calls the simulation at a constant FPS.

Returns

void

Bug The frame rate has some issues and is not consistent on some other machines.

8.3.2.10 update()

```
void glutCB::update ( )
```

Updates the audio, window size, userfunctions (ie: calls them), and camera.

Returns

void

Note

Kind of messy, needs clean up.

8.4 utility Namespace Reference

This namespace contains functions which you might expect to be standard in cpp.

Functions

- std::string [toUpper](#) (std::string str)
Converts a std::string to upper case.
- std::string [toLower](#) (std::string str)
Converts a std::string to lower case.
- std::string [replaceString](#) (std::string subject, const std::string &search, const std::string &replace)
Returns a string with the desired text replaced.
- std::string [replaceCharSet](#) (std::string subject, const std::string &charSet, const std::string &replace)
Replaces a set of characters with a string.
- template<typename T >
std::string [numToStr](#) (T Number)
Returns a number as its string representation.
- std::vector< std::string > [split](#) (std::string stringToBeSplit, std::string delimiter)
Splits a string based on a supplied delimiter.
- std::string [getCurrentTime](#) ()
Returns the current time.
- std::string [getCurrentDate](#) ()
Returns the current date.

8.4.1 Detailed Description

This namespace contains functions which you might expect to be standard in cpp.

Specifically functions in the standard libraries. If you ever ask "why isnt that function standard?" then it should be here.

8.4.2 Function Documentation

8.4.2.1 `getCurrentDate()`

```
std::string utility::getCurrentDate ( )
```

Returns the current date.

Returns

std::string The current date as a string in the format: mmm/dd/yyyy

Here is the call graph for this function:



8.4.2.2 `getCurrentTime()`

```
std::string utility::getCurrentTime ( )
```

Returns the current time.

Returns

std::string The current time as a string in the format: hh:mm:ss

Here is the call graph for this function:



8.4.2.3 numToStr()

```
template<typename T >
std::string utility::numToStr (
    T Number )
```

Returns a number as its string representation.

Template Parameters

<i>T</i>	The number type (eg: float, double, int, etc.)
----------	--

Parameters

<i>Number</i>	The value to be converted to a string.
---------------	--

Returns

A string representation of a number.

8.4.2.4 replaceCharSet()

```
std::string utility::replaceCharSet (
    std::string subject,
    const std::string & charSet,
    const std::string & replace )
```

Replaces a set of characters with a string.

This replacement string can often be just a character. This will the replace a set of characters with one character.

Parameters

<i>subject</i>	std::string The string to be returned with replacement.
<i>charSet</i>	const std::string& The set of characters to be replaced.
<i>replace</i>	const std::string& The text to replace with.

Returns

std::string The string with the character set replaced.

Here is the call graph for this function:

**8.4.2.5 replaceString()**

```
std::string utility::replaceString (
    std::string subject,
    const std::string & search,
    const std::string & replace )
```

Returns a string with the desired text replaced.

Parameters

<i>subject</i>	std::string The string to be returned with replacement.
<i>search</i>	const std::string& The text to replace.
<i>replace</i>	const std::string& The text to replace with.

Returns

std::string The replaced string

8.4.2.6 split()

```
std::vector< std::string > utility::split (
    std::string stringToBeSplit,
    std::string delimiter )
```

Splits a string based on a supplied delimiter.

Parameters

<i>stringToBeSplit</i>	std::string The string to be split.
<i>delimiter</i>	std::string The delimiter to split the string on.

Returns

`std::vector<std::string>` A vector of the split strings.

8.4.2.7 toLower()

```
std::string utility::toLower (
    std::string str )
```

Converts a `std::string` to lower case.

Parameters

<i>str</i>	<code>std::string</code> The string to convert the case of.
------------	---

Returns

`std::string` The string represented as lower case.

8.4.2.8 toUpper()

```
std::string utility::toUpper (
    std::string str )
```

Converts a `std::string` to upper case.

Parameters

<i>str</i>	<code>std::string</code> The string to convert the case of.
------------	---

Returns

`std::string` The string represented as lower case.

Chapter 9

Class Documentation

9.1 Audio Class Reference

The audio engine.

```
#include <Audio.h>
```

Public Member Functions

- [Audio](#) ()
- [~Audio](#) ()
- void [playSound](#) (std::string soundFile, double x, double y, double z)
- void [playSound](#) (std::string soundFile, [Vec](#) position)
- void [playSound](#) (std::string soundFile)
- void [clearStoppedSounds](#) ()

Public Attributes

- sf::Music [music](#)
- std::map< std::string, sf::SoundBuffer > [soundBuffers](#)
- std::vector< sf::Sound > [sounds](#)

Static Private Attributes

- static char constexpr const * [AUDIO_DIRECTORY](#) = "assets/Audio/"
- static char constexpr const * [COULD_NOT_LOAD_MUSIC_MESSAGE](#) = "Failed to open [music](#) file: "
- static char constexpr const * [COULD_NOT_LOAD_SOUND_MESSAGE](#) = "Failed to open sound file: "

9.1.1 Detailed Description

The audio engine.

This class is incomplete. It needs more options! Mono vs Stereo, disable spatialization, loop, attenuation, min distance, CLEAR WHEN DONE.

Todo This class does not clear sounds when they are done. This means that the sounds vector will grow unbounded. This is wasteful and should be resolved.

Protect / warn against using stereo sounds in 3D. (not allowed according to SFML).

9.1.2 Constructor & Destructor Documentation

9.1.2.1 Audio()

```
Audio::Audio ( )
```

9.1.2.2 ~Audio()

```
Audio::~~Audio ( )
```

9.1.3 Member Function Documentation

9.1.3.1 clearStoppedSounds()

```
void Audio::clearStoppedSounds ( )
```

9.1.3.2 playSound() [1/3]

```
void Audio::playSound (
    std::string soundFile,
    double x,
    double y,
    double z )
```

Parameters

<i>soundFile</i>	std::string
<i>x</i>	double
<i>y</i>	double
<i>z</i>	double

Returns

void

Warning

Stereo sounds will not be rendered in 3D space.

9.1.3.3 `playSound()` [2/3]

```
void Audio::playSound (
    std::string soundFile,
    Vec position )
```

9.1.3.4 `playSound()` [3/3]

```
void Audio::playSound (
    std::string soundFile )
```

Here is the call graph for this function:



9.1.4 Member Data Documentation

9.1.4.1 `AUDIO_DIRECTORY`

```
char constexpr const* Audio::AUDIO_DIRECTORY = "assets/Audio/" [static], [private]
```

9.1.4.2 `COULD_NOT_LOAD_MUSIC_MESSAGE`

```
char constexpr const* Audio::COULD_NOT_LOAD_MUSIC_MESSAGE = "Failed to open music file: "
[static], [private]
```

9.1.4.3 `COULD_NOT_LOAD_SOUND_MESSAGE`

```
char constexpr const* Audio::COULD_NOT_LOAD_SOUND_MESSAGE = "Failed to open sound file: "
[static], [private]
```

9.1.4.4 music

```
sf::Music Audio::music
```

9.1.4.5 soundBuffers

```
std::map<std::string, sf::SoundBuffer> Audio::soundBuffers
```

9.1.4.6 sounds

```
std::vector<sf::Sound> Audio::sounds
```

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

- [Audio.h](#)
- [Audio.cpp](#)

9.2 Camera Struct Reference

This class describes the camera placed at the head of the user.

```
#include <GFramework.h>
```

Public Member Functions

- [Camera](#) ()

Public Attributes

- double [translationSpeed](#)
How fast the camera moves forward.
- double [rotationSpeed](#)
How fast the camera rotates.
- [Vec](#) [pos](#)
The global coordinates of the camera.
- [Vec](#) [mov](#)
The next frame movement of the camera.
- [Vec](#) [dir](#)
The direction the camera is pointing to.
- [Vec](#) [ang](#)
The angle around the z axis, starting at +x. (I think)
- [Vec](#) [del](#)
The next frame rotation of the camera.

Static Public Attributes

- static double constexpr `DEFAULT_T_SPEED` = 1.0
The default translation speed.
- static double constexpr `DEFAULT_R_SPEED` = 0.04
The default rotation speed.
- static double constexpr `DEFAULT_HEIGHT` = 1.8
The default height of the camera (wrt z=0)

9.2.1 Detailed Description

This class describes the camera placed at the head of the user.

Todo In future, the camera implementation should be removed from the graphics class and put here.

9.2.2 Constructor & Destructor Documentation

9.2.2.1 Camera()

```
Camera::Camera ( ) [inline]
```

9.2.3 Member Data Documentation

9.2.3.1 ang

```
Vec Camera::ang
```

The angle around the z axis, starting at +x. (I think)

9.2.3.2 DEFAULT_HEIGHT

```
double constexpr Camera::DEFAULT_HEIGHT = 1.8 [static]
```

The default height of the camera (wrt z=0)

9.2.3.3 DEFAULT_R_SPEED

```
double constexpr Camera::DEFAULT_R_SPEED = 0.04 [static]
```

The default rotation speed.

9.2.3.4 DEFAULT_T_SPEED

```
double constexpr Camera::DEFAULT_T_SPEED = 1.0 [static]
```

The default translation speed.

9.2.3.5 del

```
Vec Camera::del
```

The next frame rotation of the camera.

9.2.3.6 dir

```
Vec Camera::dir
```

The direction the camera is pointing to.

9.2.3.7 mov

```
Vec Camera::mov
```

The next frame movement of the camera.

9.2.3.8 pos

```
Vec Camera::pos
```

The global coordinates of the camera.

9.2.3.9 rotationSpeed

```
double Camera::rotationSpeed
```

How fast the camera rotates.

9.2.3.10 translationSpeed

```
double Camera::translationSpeed
```

How fast the camera moves forward.

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

- [GFramework.h](#)

9.3 Config Class Reference

This class allows variables to be loaded from a file.

```
#include <Config.h>
```

Public Member Functions

- [~Config](#) ()
Empty destructor.

Static Public Member Functions

- `template<typename T >`
`static T getValue (std::string key)`
Returns the value associated with the key as the specified type.

Protected Member Functions

- [Config](#) ()
Empty constructor.

Static Private Member Functions

- `static std::string getFileValue (std::string key)`
Returns the corresponding value to the key provided from the config file.

Static Private Attributes

- static constexpr const char * `CONFIG_FILE` = "assets/config.config"
The file containing the configuration data.

9.3.1 Detailed Description

This class allows variables to be loaded from a file.

This is very useful to set the state of the program on start up. To add a new config parameter, just append it to the config file with an associated value. Values can be parsed as boolean, int, double, std::string. Then the value can be retrieved from the key as:

```
std::string value = Config::getValue<std::string>("STRING_TEST")
```

Todo Allow changes to config file based on program state.

9.3.2 Constructor & Destructor Documentation

9.3.2.1 ~Config()

```
Config::~Config ( )
```

Empty destructor.

9.3.2.2 Config()

```
Config::Config ( ) [protected]
```

Empty constructor.

9.3.3 Member Function Documentation

9.3.3.1 getFileValue()

```
std::string Config::getFileValue (
    std::string key ) [static], [private]
```

Returns the corresponding value to the key provided from the config file.

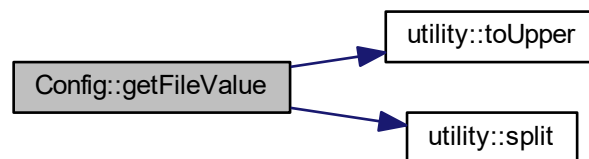
Parameters

<i>key</i>	std::string The string to be found in the configuration file.
------------	---

Returns

std::string The value of the configuration variable.

Here is the call graph for this function:

**9.3.3.2 getValue()**

```
template<typename T >
static T Config::getValue (
    std::string key ) [inline], [static]
```

Returns the value associated with the key as the specified type.

Parameters

<i>key</i>	std::string The key to be found.
------------	----------------------------------

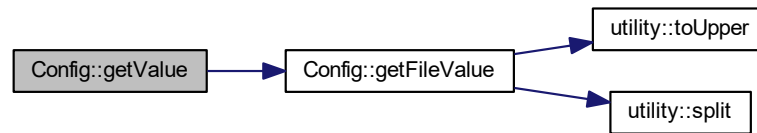
Returns

T templated parameter can be one of bool, int, double, std::string.

Warning

Fatal error if key cannot be found since configuration parameters are required.

Here is the call graph for this function:



9.3.4 Member Data Documentation

9.3.4.1 CONFIG_FILE

```
constexpr const char* Config::CONFIG_FILE = "assets/config.config" [static], [private]
```

The file containing the configuration data.

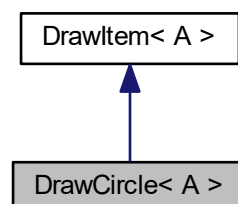
The documentation for this class was generated from the following files:

- [Config.h](#)
- [Config.cpp](#)

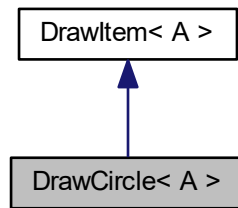
9.4 DrawCircle< A > Struct Template Reference

```
#include <Shapes.h>
```

Inheritance diagram for DrawCircle< A >:



Collaboration diagram for DrawCircle< A >:



Public Member Functions

- [DrawCircle](#) (double x, double y, double r)
- [DrawCircle](#) (double x, double y, double r, int numSegments)

Static Public Attributes

- static [Drawing::Dimension](#) constexpr [dimension](#) = static_cast<[Drawing::Dimension](#)>(2)

Private Member Functions

- void [draw](#) (double x, double y, double r, int numSegments)

Additional Inherited Members

9.4.1 Constructor & Destructor Documentation

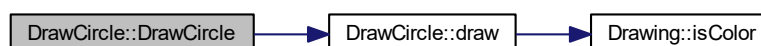
9.4.1.1 DrawCircle() [1/2]

```
template<Appearance A>
DrawCircle< A >::DrawCircle (
    double x,
    double y,
    double r ) [inline]
```

9.4.1.2 DrawCircle() [2/2]

```
template<Appearance A>
DrawCircle< A >::DrawCircle (
    double x,
    double y,
    double r,
    int numSegments ) [inline]
```

Here is the call graph for this function:



9.4.2 Member Function Documentation

9.4.2.1 draw()

```
template<Appearance A>
void DrawCircle< A >::draw (
    double x,
    double y,
    double r,
    int numSegments ) [inline], [private]
```

Here is the call graph for this function:



9.4.3 Member Data Documentation

9.4.3.1 dimension

```
template<Appearance A>
Drawing::Dimension constexpr DrawCircle< A >::dimension = static_cast<Drawing::Dimension>(2)
[static]
```

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

- [Shapes.h](#)

9.5 Drawing Struct Reference

```
#include <Draw.h>
```

Public Types

- enum [Dimension](#) : unsigned int { [Dimension::NONE](#) =0, [Dimension::TWO](#) =2, [Dimension::THREE](#) =3 }

Static Public Member Functions

- static void [enableND](#) ([Dimension](#) d)
Prepares drawing for the specified dimension.
- static void [enable2D](#) ()
Prepares the drawing for 2 dimensional drawing.
- static void [enable3D](#) ()
Prepares the drawing for 2 dimensional drawing.
- static void [changeColor](#) ([Vec](#) c)
Specifies the color for drawing.
- static void [changeColor](#) ([Appearance](#) C)
Specifies the color for drawing.
- static void [changeTexture](#) ([Tex](#) textureID)
Specifies the texture for drawing.
- static void [changeTexture](#) ([Appearance](#) C)
Specifies the texture for drawing.
- static bool [isColor](#) ([Appearance](#) A)
Determines if the specified Appearance is a color.
- static bool [isTexture](#) ([Appearance](#) A)
Determines if the specified Appearance is a texture.

Static Public Attributes

- static constexpr const char * [UNKNOWN_DIMENSION_MESSAGE](#) = "Invalid number of dimensions, defaulting to 2D."
- static constexpr const char * [INVALID_DRAWING_ORDER_MESSAGE](#) = "Cannot render 2D GFramework before 3D."
- static constexpr const char * [INVALID_COLOR_MESSAGE](#) = "Color is not valid."
- static constexpr const char * [INVALID_TEXTURE_MESSAGE](#) = "Texture is not valid."
- static constexpr const char * [UNKNOWN_APPROVED_COLOR_MESSAGE](#) = "Approved color not found."
- static constexpr const char * [UNKNOWN_APPROVED_TEXTURE_MESSAGE](#) = "Approved texture not found."

9.5.1 Member Enumeration Documentation

9.5.1.1 Dimension

```
enum Drawing::Dimension : unsigned int [strong]
```

Enumerator

NONE	
TWO	
THREE	

9.5.2 Member Function Documentation

9.5.2.1 changeColor() [1/2]

```
void Drawing::changeColor (
    Vec c ) [static]
```

Specifies the color for drawing.

Parameters

<i>c</i>	Vec The (r,g,b) values for the color.
----------	---------------------------------------

Returns

void

9.5.2.2 changeColor() [2/2]

```
void Drawing::changeColor (
    Appearance C ) [static]
```

Specifies the color for drawing.

Parameters

<i>C</i>	Appearance The Appearance enum for the color.
----------	---

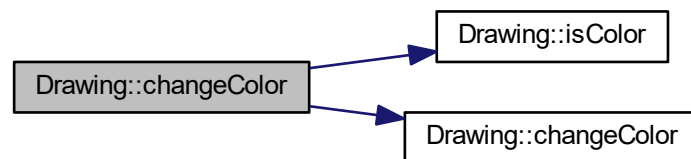
Returns

void

See also

[Appearance](#)

Here is the call graph for this function:

**9.5.2.3 `changeTexture()`** [1/2]

```
void Drawing::changeTexture (  
    Tex textureID ) [static]
```

Specifies the texture for drawing.

Parameters

<i>textureID</i>	Tex The id corresponding to the texture to draw with.
------------------	---

Returns

void

9.5.2.4 `changeTexture()` [2/2]

```
void Drawing::changeTexture (  
    Appearance C ) [static]
```

Specifies the texture for drawing.

Parameters

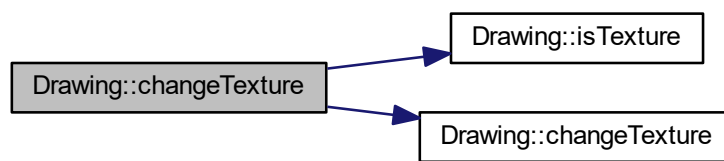
C	Appearance The Appearance enum corresponding to the texture to draw with.
----------	---

Returns

void

See also[Appearance](#)

Here is the call graph for this function:

**9.5.2.5 enable2D()**

```
void Drawing::enable2D ( ) [static]
```

Prepares the drawing for 2 dimensional drawing.

Returns

void

9.5.2.6 enable3D()

```
void Drawing::enable3D ( ) [static]
```

Prepares the drawing for 2 dimensional drawing.

Returns

void

9.5.2.7 enableND()

```
void Drawing::enableND (
    Dimension d ) [static]
```

Prepares drawing for the specified dimension.

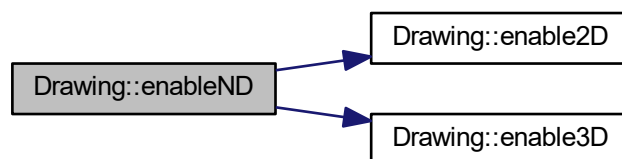
Parameters

<i>d</i>	Dimension The desired drawing dimension.
----------	--

Returns

void

Here is the call graph for this function:

**9.5.2.8 isColor()**

```
bool Drawing::isColor (  
    Appearance A ) [static]
```

Determines if the specified Appearance is a color.

Parameters

<i>A</i>	Appearance The Appearance to check.
----------	-------------------------------------

Returns

bool True if the Appearance is a color. False otherwise.

9.5.2.9 isTexture()

```
bool Drawing::isTexture (  
    Appearance A ) [static]
```

Determines if the specified Appearance is a texture.

Parameters

A	Appearance The Appearance to check.
---	-------------------------------------

Returns

bool True if the Appearance is a texture. False otherwise.

9.5.3 Member Data Documentation

9.5.3.1 INVALID_COLOR_MESSAGE

```
constexpr const char* Drawing::INVALID_COLOR_MESSAGE = "Color is not valid." [static]
```

9.5.3.2 INVALID_DRAWING_ORDER_MESSAGE

```
constexpr const char* Drawing::INVALID_DRAWING_ORDER_MESSAGE = "Cannot render 2D GFramework  
before 3D." [static]
```

9.5.3.3 INVALID_TEXTURE_MESSAGE

```
constexpr const char* Drawing::INVALID_TEXTURE_MESSAGE = "Texture is not valid." [static]
```

9.5.3.4 UNKNOWN_APPROVED_COLOR_MESSAGE

```
constexpr const char* Drawing::UNKNOWN_APPROVED_COLOR_MESSAGE = "Approved color not found."  
[static]
```

9.5.3.5 UNKNOWN_APPROVED_TEXTURE_MESSAGE

```
constexpr const char* Drawing::UNKNOWN_APPROVED_TEXTURE_MESSAGE = "Approved texture not found."  
[static]
```


9.5.3.6 UNKNOWN_DIMENSION_MESSAGE

```
constexpr const char* Drawing::UNKNOWN_DIMENSION_MESSAGE = "Invalid number of dimensions,
defaulting to 2D." [static]
```

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

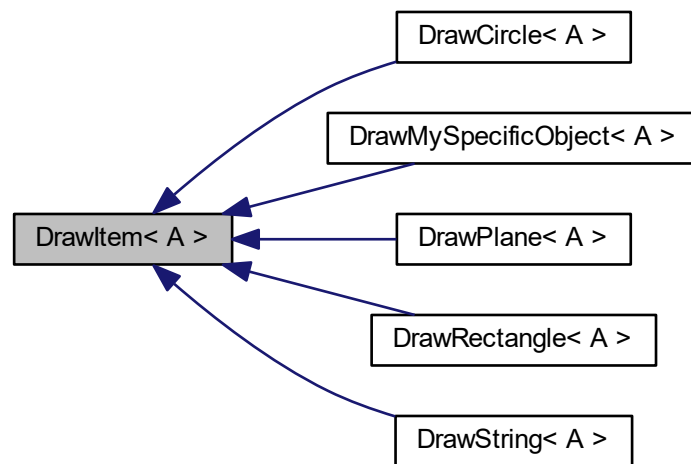
- [Draw.h](#)
- [Draw.cpp](#)

9.6 DrawItem< A > Class Template Reference

The parent class of items which can drawn.

```
#include <Draw.h>
```

Inheritance diagram for DrawItem< A >:



Protected Member Functions

- [DrawItem](#) ([Drawing::Dimension](#) dim)
Applies the appearance. The subclass is used to perform the drawing.

Static Private Attributes

- static constexpr const char * [INVALID_APPEARANCE_MESSAGE](#) = "Appearance is not valid."

9.6.1 Detailed Description

```
template<Appearance A>
class DrawItem< A >
```

The parent class of items which can drawn.

Template Parameters

<i>A</i>	The appearance to be used when drawing.
----------	---

Warning

Not all children will know how to use textures. This may be fine, but it might also log a warning or crash the program.

9.6.2 Constructor & Destructor Documentation

9.6.2.1 DrawItem()

```
template<Appearance A>
DrawItem< A >::DrawItem (
    Drawing::Dimension dim ) [inline], [protected]
```

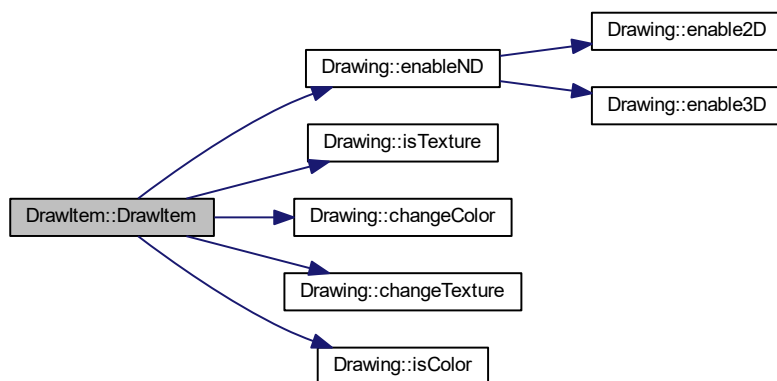
Applies the appearance. The subclass is used to perform the drawing.

Constructor is used as decorator. This will apply the texture before the drawing is performed.

Parameters

<i>dim</i>	Drawing::Dimension
------------	------------------------------------

Here is the call graph for this function:



9.6.3 Member Data Documentation

9.6.3.1 INVALID_APPEARANCE_MESSAGE

```
template<Appearance A>
constexpr const char* DrawItem< A >::INVALID_APPEARANCE_MESSAGE = "Appearance is not valid."
[static], [private]
```

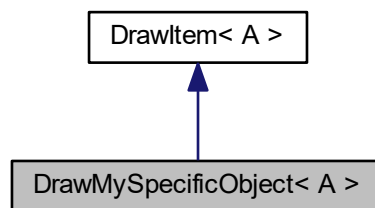
The documentation for this class was generated from the following file:

- [Draw.h](#)

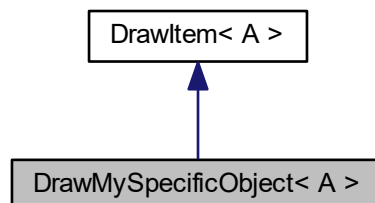
9.7 DrawMySpecificObject< A > Struct Template Reference

```
#include <Objects.h>
```

Inheritance diagram for DrawMySpecificObject< A >:



Collaboration diagram for DrawMySpecificObject< A >:



Public Member Functions

- [DrawMySpecificObject](#) (double x, double y, double X, double Y)

Static Public Attributes

- static `Drawing::Dimension` constexpr `dimension` = `static_cast<Drawing::Dimension>(3)`

Private Member Functions

- void `draw` (double x, double y, double X, double Y)

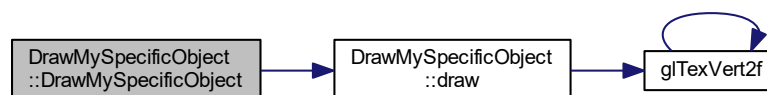
Additional Inherited Members

9.7.1 Constructor & Destructor Documentation

9.7.1.1 DrawMySpecificObject()

```
template<Appearance A>
DrawMySpecificObject< A >::DrawMySpecificObject (
    double x,
    double y,
    double X,
    double Y ) [inline]
```

Here is the call graph for this function:

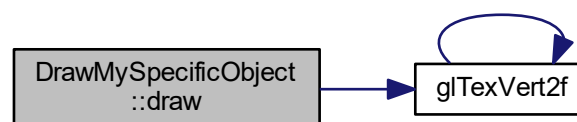


9.7.2 Member Function Documentation

9.7.2.1 draw()

```
template<Appearance A>
void DrawMySpecificObject< A >::draw (
    double x,
    double y,
    double X,
    double Y ) [inline], [private]
```

Here is the call graph for this function:



9.7.3 Member Data Documentation

9.7.3.1 dimension

```
template<Appearance A>
Drawing::Dimension constexpr DrawMySpecificObject< A >::dimension = static_cast<Drawing::Dimension>(3)
[static]
```

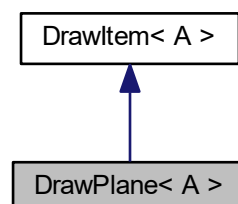
The documentation for this struct was generated from the following file:

- [Objects.h](#)

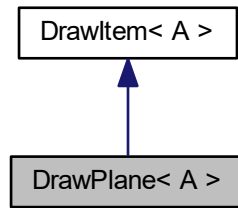
9.8 DrawPlane< A > Struct Template Reference

```
#include <Shapes.h>
```

Inheritance diagram for DrawPlane< A >:



Collaboration diagram for DrawPlane< A >:



Public Member Functions

- [DrawPlane](#) (double length)
- [DrawPlane](#) (double length, double xAngle)
- [DrawPlane](#) ([Vec](#) pos, double length)
- [DrawPlane](#) (double x, double y, double z, double length)
- [DrawPlane](#) (double x, double y, double z, double length, double xAngle)
- [DrawPlane](#) ([Vec](#) pos, double length, double xAngle)

Static Public Attributes

- static [Drawing::Dimension](#) constexpr [dimension](#) = static_cast<[Drawing::Dimension](#)>(3)

Private Member Functions

- void [draw](#) ([Vec](#) pos, double length, double xAngle)

Additional Inherited Members

9.8.1 Constructor & Destructor Documentation

9.8.1.1 DrawPlane() [1/6]

```

template<Appearance A>
DrawPlane< A >::DrawPlane (
    double length ) [inline]
  
```

9.8.1.2 DrawPlane() [2/6]

```
template<Appearance A>
DrawPlane< A >::DrawPlane (
    double length,
    double xAngle ) [inline]
```

9.8.1.3 DrawPlane() [3/6]

```
template<Appearance A>
DrawPlane< A >::DrawPlane (
    Vec pos,
    double length ) [inline]
```

9.8.1.4 DrawPlane() [4/6]

```
template<Appearance A>
DrawPlane< A >::DrawPlane (
    double x,
    double y,
    double z,
    double length ) [inline]
```

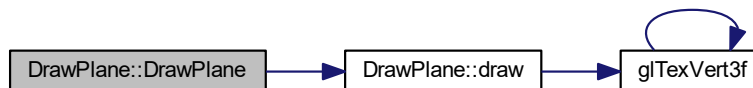
9.8.1.5 DrawPlane() [5/6]

```
template<Appearance A>
DrawPlane< A >::DrawPlane (
    double x,
    double y,
    double z,
    double length,
    double xAngle ) [inline]
```

9.8.1.6 DrawPlane() [6/6]

```
template<Appearance A>
DrawPlane< A >::DrawPlane (
    Vec pos,
    double length,
    double xAngle ) [inline]
```

Here is the call graph for this function:

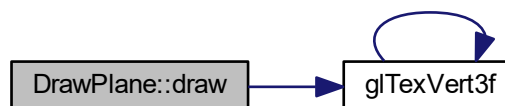


9.8.2 Member Function Documentation

9.8.2.1 draw()

```
template<Appearance A>
void DrawPlane< A >::draw (
    Vec pos,
    double length,
    double xAngle ) [inline], [private]
```

Here is the call graph for this function:



9.8.3 Member Data Documentation

9.8.3.1 dimension

```
template<Appearance A>
Drawing::Dimension constexpr DrawPlane< A >::dimension = static_cast<Drawing::Dimension>(3)
[static]
```

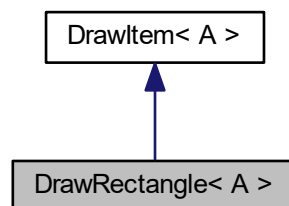
The documentation for this struct was generated from the following file:

- [Shapes.h](#)

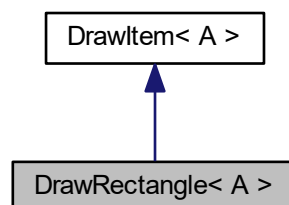
9.9 DrawRectangle< A > Struct Template Reference

```
#include <Shapes.h>
```

Inheritance diagram for DrawRectangle< A >:



Collaboration diagram for DrawRectangle< A >:



Public Member Functions

- [DrawRectangle](#) (double x, double y, double X, double Y)

Static Public Attributes

- static [Drawing::Dimension](#) constexpr [dimension](#) = static_cast<[Drawing::Dimension](#)>(2)

Private Member Functions

- void [draw](#) (double x, double y, double X, double Y)

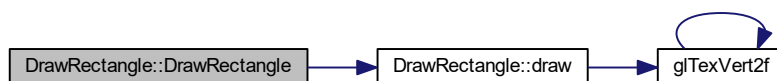
Additional Inherited Members

9.9.1 Constructor & Destructor Documentation

9.9.1.1 DrawRectangle()

```
template<Appearance A>
DrawRectangle< A >::DrawRectangle (
    double x,
    double y,
    double X,
    double Y ) [inline]
```

Here is the call graph for this function:

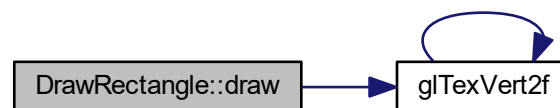


9.9.2 Member Function Documentation

9.9.2.1 draw()

```
template<Appearance A>
void DrawRectangle< A >::draw (
    double x,
    double y,
    double X,
    double Y ) [inline], [private]
```

Here is the call graph for this function:



9.9.3 Member Data Documentation

9.9.3.1 dimension

```
template<Appearance A>
Drawing::Dimension constexpr DrawRectangle< A >::dimension = static_cast<Drawing::Dimension>(2)
[static]
```

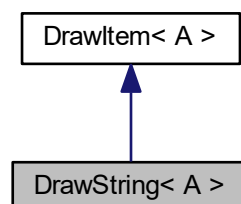
The documentation for this struct was generated from the following file:

- [Shapes.h](#)

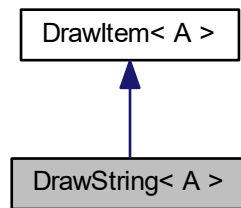
9.10 DrawString< A > Struct Template Reference

```
#include <Text.h>
```

Inheritance diagram for DrawString< A >:



Collaboration diagram for DrawString< A >:



Public Member Functions

- [DrawString](#) (std::string s, float x, float y, bool centerX=true, bool centerY=true)

Static Public Attributes

- static [Drawing::Dimension](#) constexpr [dimension](#) = static_cast<[Drawing::Dimension](#)>(2)

Private Member Functions

- [draw](#) (std::string s, float x, float y, bool centerX=true, bool centerY=true)

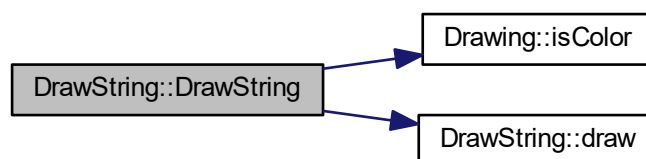
Additional Inherited Members

9.10.1 Constructor & Destructor Documentation

9.10.1.1 DrawString()

```
template<Appearance A>
DrawString< A >::DrawString (
    std::string s,
    float x,
    float y,
    bool centerX = true,
    bool centerY = true ) [inline]
```

Here is the call graph for this function:



9.10.2 Member Function Documentation

9.10.2.1 draw()

```
template<Appearance A>
DrawString< A >::draw (
    std::string s,
    float x,
    float y,
    bool centerX = true,
    bool centerY = true ) [inline], [private]
```

9.10.3 Member Data Documentation

9.10.3.1 dimension

```
template<Appearance A>
Drawing::Dimension constexpr DrawString< A >::dimension = static_cast<Drawing::Dimension>(2)
[static]
```

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

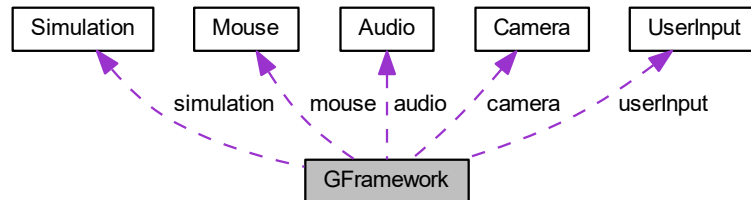
- [Text.h](#)

9.11 GFramework Class Reference

This class holds all the functions required for a 3D simulator to be run.

```
#include <GFramework.h>
```

Collaboration diagram for GFramework:



Public Member Functions

- void [startup](#) ()
Initialize simulation and start the glutmainloop.
- void [showScene](#) ()
Sets the proper display settings & calls glutswapbuffers.
- [~GFramework](#) ()
Destructor deletes singleton instance.
- [GFramework](#) ([GFramework](#) const &)=delete
Copy constructor is deleted in singleton.
- void [operator=](#) ([GFramework](#) const &)=delete
Assignment operator is deleted in singleton.

Public Attributes

- [Vec2](#) [windowSize](#)
(width, height) dimensions of the window.
- [Drawing::Dimension](#) [drawingState](#)
What dimension the [GFramework](#) is prepared to draw in.
- [Camera](#) [camera](#)
The camera on the users head.
- [Mouse](#) [mouse](#)
The mouse the user controls.
- [UserInput](#) [userInput](#)
Set of user inputs.
- [Audio](#) * [audio](#)
Audio system for the engine.
- [Simulation](#) * [simulation](#)
The simulation to be run in this framework.
- bool [display](#) = true
Whether the scene is being shown to the user. (set to false for maximum speed)
- std::map< enum [Appearance](#), [Tex](#) > [textureMap](#) = {}
Maps an appearance to a texture for drawing.
- std::map< enum [Appearance](#), [Vec](#) > [colorMap](#) = {}
Maps an appearance to a color for drawing.

Static Public Attributes

- static double constexpr `FPS` = 60
Frames per second for the simulation to run at.
- static int constexpr `RENDERING_DISTANCE` = 1000
How far objects should be rendered.
- static std::unique_ptr< `GFramework` > const `get`
Singleton instance. Unique_ptr insures destruction.

Private Member Functions

- `GFramework` ()
Constructor initializes freeglut and loads appearances into maps. Private for singleton pattern.
- void `initializeGlut` ()
Intializes all the freeglut functions with callbacks.
- void `loadTextures` ()
Loads all the textures into the framework.
- void `loadColors` ()
Loads all the colors into the framework.

Static Private Attributes

- static char constexpr const * `WINDOW_TITLE` = "Evolution WildFire"
Name of the window / game.
- static int constexpr `INIT_WINDOW_X` = 600
Initial x position of the window on start up.
- static int constexpr `INIT_WINDOW_Y` = 100
Initial y position of the window on start up.
- static int constexpr `INIT_WINDOW_WIDTH` = 1920 / 2.0
Initial width of the window on start up.
- static int constexpr `INIT_WINDOW_HEIGHT` = 1080 / 2.0
Initial height of the window on start up.

9.11.1 Detailed Description

This class holds all the functions required for a 3D simulator to be run.

It includes camera, audio, user input among other things. This class implements the singleton pattern and as a result only one instance is ever allowed.

9.11.2 Constructor & Destructor Documentation

9.11.2.1 ~GFramework()

```
GFramework::~~GFramework ( )
```

Destructor deletes singleton instance.

9.11.2.2 GFramework() [1/2]

```
GFramework::GFramework (
    GFramework const & ) [delete]
```

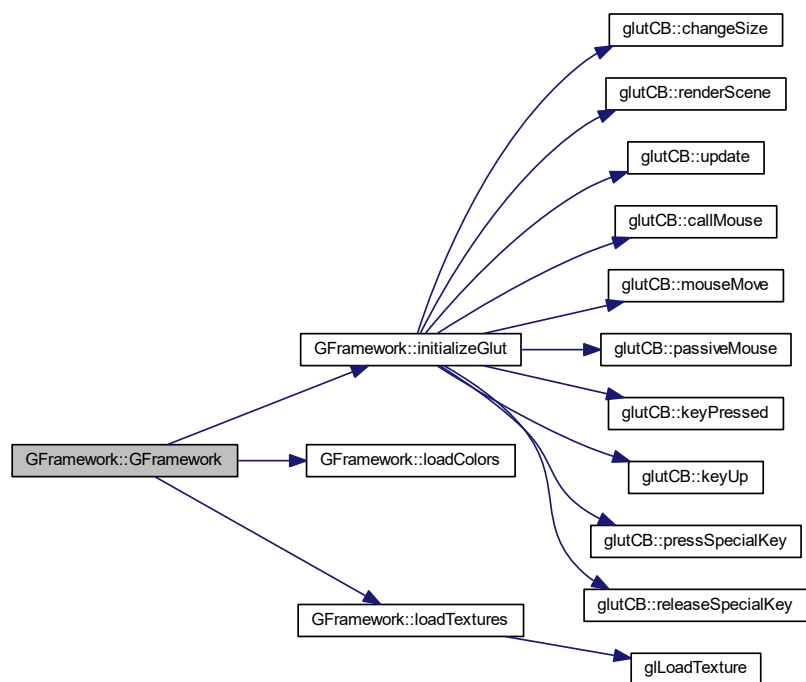
Copy constructor is deleted in singleton.

9.11.2.3 GFramework() [2/2]

```
GFramework::GFramework ( ) [private]
```

Constructor initializes freeglut and loads appearances into maps. Private for singleton pattern.

Here is the call graph for this function:



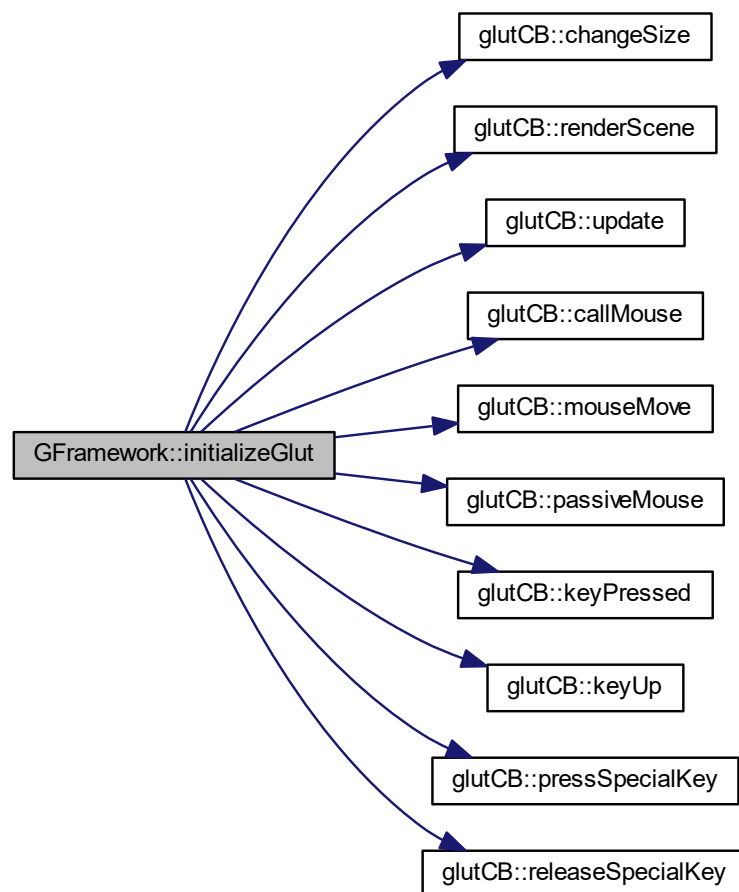
9.11.3 Member Function Documentation

9.11.3.1 initializeGlut()

```
void GFramework::initializeGlut ( ) [private]
```

Initializes all the freeglut functions with callbacks.

Here is the call graph for this function:



9.11.3.2 loadColors()

```
void GFramework::loadColors ( ) [private]
```

Loads all the colors into the framework.

9.11.3.3 loadTextures()

```
void GFramework::loadTextures ( ) [private]
```

Loads all the textures into the framework.

Here is the call graph for this function:



9.11.3.4 operator=()

```
void GFramework::operator= (
    GFramework const & ) [delete]
```

Assignment operator is deleted in singleton.

9.11.3.5 showScene()

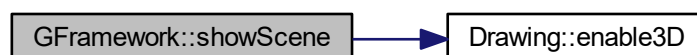
```
void GFramework::showScene ( )
```

Sets the proper display settings & calls glutswapbuffers.

Returns

void

Here is the call graph for this function:



9.11.3.6 startup()

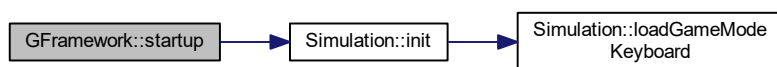
```
void GFramework::startup ( )
```

Initialize simulation and start the glutmainloop.

Returns

void

Here is the call graph for this function:



9.11.4 Member Data Documentation

9.11.4.1 audio

```
Audio* GFramework::audio
```

Audio system for the engine.

9.11.4.2 camera

```
Camera GFramework::camera
```

The camera on the users head.

9.11.4.3 colorMap

```
std::map<enum Appearance, Vec> GFramework::colorMap = {}
```

Maps an appearance to a color for drawing.

9.11.4.4 display

```
bool GFramework::display = true
```

Whether the scene is being shown to the user. (set to false for maximum speed)

9.11.4.5 drawingState

```
Drawing::Dimension GFramework::drawingState
```

What dimension the [GFramework](#) is prepared to draw in.

9.11.4.6 FPS

```
double constexpr GFramework::FPS = 60 [static]
```

Frames per second for the simulation to run at.

9.11.4.7 get

```
std::unique_ptr< GFramework > const GFramework::get [static]
```

Singleton instance. Unique_ptr insures destruction.

9.11.4.8 INIT_WINDOW_HEIGHT

```
int constexpr GFramework::INIT_WINDOW_HEIGHT = 1080 / 2.0 [static], [private]
```

Initial height of the window on start up.

9.11.4.9 INIT_WINDOW_WIDTH

```
int constexpr GFramework::INIT_WINDOW_WIDTH = 1920 / 2.0 [static], [private]
```

Initial width of the window on start up.

9.11.4.10 INIT_WINDOW_X

```
int constexpr GFramework::INIT_WINDOW_X = 600 [static], [private]
```

Initial x position of the window on start up.

9.11.4.11 INIT_WINDOW_Y

```
int constexpr GFramework::INIT_WINDOW_Y = 100 [static], [private]
```

Initial y position of the window on start up.

9.11.4.12 mouse

```
Mouse GFramework::mouse
```

The mouse the user controls.

9.11.4.13 RENDERING_DISTANCE

```
int constexpr GFramework::RENDERING_DISTANCE = 1000 [static]
```

How far objects should be rendered.

9.11.4.14 simulation

```
Simulation* GFramework::simulation
```

The simulation to be run in this framework.

9.11.4.15 textureMap

```
std::map<enum Appearance, Tex> GFramework::textureMap = {}
```

Maps an appearance to a texture for drawing.

9.11.4.16 userInput

`UserInput` `GFramework::userInput`

Set of user inputs.

9.11.4.17 WINDOW_TITLE

`char constexpr const*` `GFramework::WINDOW_TITLE` = "Evolution WildFire" [static], [private]

Name of the window / game.

9.11.4.18 windowSize

`Vec2` `GFramework::windowSize`

(width, height) dimensions of the window.

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

- [GFramework.h](#)
- [LoadColors.cpp](#)
- [LoadTextures.cpp](#)
- [GFramework.cpp](#)

9.12 Logger Class Reference

This class houses the logging functionality.

```
#include <Logger.h>
```

Public Member Functions

- void `log` (int line, std::string file, std::string func, std::string msg, `LogDegree` d=DEFAULT_DEGREE, `LogType` t=DEFAULT_TYPE)
Logs a descriptive message to the appropriate stream.
- void `normalExit` (int line, std::string file, std::string func)
Logs a descriptive normal exit message.
- `~Logger` ()
Cleans up the logger.
- `Logger` (`Logger` const &)=delete
Deleted for singleton pattern.
- void `operator=` (`Logger` const &)=delete
Deleted for singleton pattern.

Static Public Member Functions

- static [Logger](#) & [get](#) ()
Returns the singleton instance.

Private Member Functions

- [Logger](#) ()
Private constructor for singleton pattern.
- void [logMessage](#) (std::string msg, bool writeToConsole=true)
Logs the message to the appropriate stream.
- std::string [getTimeStamp](#) ()
Returns the current time.
- std::string [toString](#) ([LogDegree](#) s)
Converts a LogDegree to a string (Verbatim)
- std::string [toString](#) ([LogType](#) e)
Converts a LogType to a string (Verbatim)

Private Attributes

- std::unordered_set< std::string > [logs](#)
Stores the unique warning log messages.
- std::ofstream [logFile](#)
The ofstream corresponding to the logging file.

Static Private Attributes

- static constexpr const [LogDegree](#) [DEFAULT_DEGREE](#) = [LogDegree::DEBUG](#)
The LogDegree to use if none is specified.
- static constexpr const [LogType](#) [DEFAULT_TYPE](#) = [LogType::GENERAL](#)
The LogType to use if none is specified.
- static constexpr const char * [LOG_FILE_TITLE](#) = "assets/logger.log"
- static constexpr const char * [PROGRAM_EXIT_MESSAGE](#) = "The program has exited successfully."
- static constexpr const char * [OPENED_LOG_AFTER_FAILURE_MESSAGE](#) = "Initialization could not open logging file. Succeeded in reopening.\n"
- static constexpr const char * [FAILED_TO_OPEN_AFTER_FAILURE_MESSAGE](#) = "Initialization could not open logging file. Failed in reopening.\n"
- static constexpr const char * [UNKNOWN_LOG_DEGREE_MESSAGE](#) = "Invalid Log Degree Specified.\n"

9.12.1 Detailed Description

This class houses the logging functionality.

It allows you to log messages of various severity (degree) and provides detailed output which is very useful for debugging.

Todo Right now, there isn't a way to propagate warning and error messages to the screen. Maybe this could be a variable "lastLogged" or "lastError" or "lastWarning" OR "lastSignificantLog".

9.12.2 Constructor & Destructor Documentation

9.12.2.1 ~Logger()

```
Logger::~~Logger ( )
```

Cleans up the logger.

Closes the log file.

9.12.2.2 Logger() [1/2]

```
Logger::Logger (
    Logger const & ) [delete]
```

Deleted for singleton pattern.

9.12.2.3 Logger() [2/2]

```
Logger::Logger ( ) [private]
```

Private constructor for singleton pattern.

9.12.3 Member Function Documentation

9.12.3.1 get()

```
static Logger& Logger::get ( ) [inline], [static]
```

Returns the singleton instance.

Returns

[Logger](#)& Reference to the [Logger](#) singleton instance.

9.12.3.2 `getTimeStamp()`

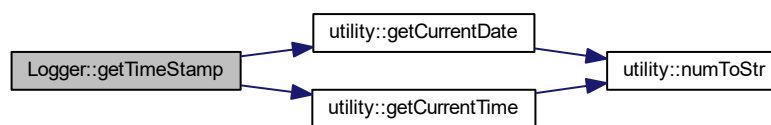
```
std::string Logger::getTimeStamp ( ) [private]
```

Returns the current time.

Returns

std::string Time in the format MMM/dd/yyyy-(hh:mm:ss)

Here is the call graph for this function:

9.12.3.3 `log()`

```
void Logger::log (
    int line,
    std::string file,
    std::string func,
    std::string msg,
    LogDegree d = DEFAULT_DEGREE,
    LogType t = DEFAULT_TYPE )
```

Logs a descriptive message to the appropriate stream.

The logged string is in the format [msg] [line] [file] [function] [date] [time] in a human readable format. The LogDegree specifies where and how the message should be logged.

- **Debug:** Show to console, log to file
- **Warning:** Show to console, log to file, display to user (blocks input until user dismisses message). Only logs unique warnings.
- **Fatal:** Show to console, log to file, display to user (EXIT_FAILURE on dismissal)

Parameters

<i>line</i>	int The line that called the logger.
<i>file</i>	std::string The file that called the logger.
<i>func</i>	std::string The function that called the logger.
<i>msg</i>	std::string The message to be logged.
<i>d</i>	LogDegree The specified LogDegree.
<i>t</i>	LogType The specified LogType.

Returns

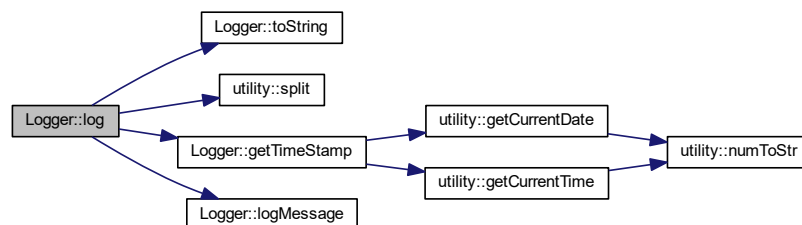
void

Warning

This function is not meant to be called manually. It should be called through the [LOG](#) macro.

See also[LOG](#)

Here is the call graph for this function:

**9.12.3.4 logMessage()**

```
void Logger::logMessage (
    std::string msg,
    bool writeToConsole = true ) [private]
```

Logs the message to the appropriate stream.

Tries to open the log file for writing. If it is unsuccessful, every next log attempt will try to reopen the file. Warnings and Errors will only be logged once. Errors should alert the user and terminate the program. Warnings should just prompt the user. This handling is implemented in the simulation. Debug messages are written to the console based on [Config](#) file parameter "DEBUG_LOG_TO_CONSOLE", while Warnings and Errors are.

Parameters

<i>msg</i>	std::string The message to be logged.
<i>writeToConsole</i>	bool Should the message print to the console.

Returns

void

Note

Default arguments are not obvious. I always have to come back here. Maybe fix this?

9.12.3.5 normalExit()

```
void Logger::normalExit (
    int line,
    std::string file,
    std::string func )
```

Logs a descriptive normal exit message.

The logged string is in the format [msg] [line] [file] [function] [date] [time] in a human readable format. Normal exit is logged as debug message.

Parameters

<i>line</i>	int The line that called the logger.
<i>file</i>	std::string The file that called the logger.
<i>func</i>	std::string The function that called the logger.

Returns

void

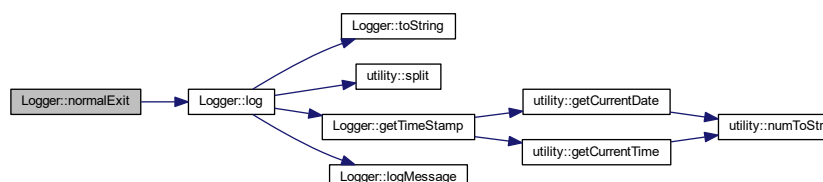
Warning

This function is not meant to be called manually. It should be called through the [NORMAL_EXIT](#) macro.

See also

[NORMAL_EXIT](#)

Here is the call graph for this function:



9.12.3.6 operator=()

```
void Logger::operator= (
    Logger const & ) [delete]
```

Deleted for singleton pattern.

9.12.3.7 toString() [1/2]

```
std::string Logger::toString (
    LogDegree s ) [private]
```

Converts a LogDegree to a string (Verbatim)

9.12.3.8 toString() [2/2]

```
std::string Logger::toString (
    LogType e ) [private]
```

Converts a LogType to a string (Verbatim)

9.12.4 Member Data Documentation

9.12.4.1 DEFAULT_DEGREE

```
constexpr const LogDegree Logger::DEFAULT_DEGREE = LogDegree::DEBUG [static], [private]
```

The LogDegree to use if none is specified.

9.12.4.2 DEFAULT_TYPE

```
constexpr const LogType Logger::DEFAULT_TYPE = LogType::GENERAL [static], [private]
```

The LogType to use if none is specified.

9.12.4.3 FAILED_TO_OPEN_AFTER_FAILURE_MESSAGE

```
constexpr const char* Logger::FAILED_TO_OPEN_AFTER_FAILURE_MESSAGE = "Initialization could not  
open logging file. Failed in reopening.\n" [static], [private]
```

9.12.4.4 LOG_FILE_TITLE

```
constexpr const char* Logger::LOG_FILE_TITLE = "assets/logger.log" [static], [private]
```

9.12.4.5 logFile

```
std::ofstream Logger::logFile [private]
```

The ofsteam corresponding to the logging file.

9.12.4.6 logs

```
std::unordered_set<std::string> Logger::logs [private]
```

Stores the unique warning log messages.

This is used to prevent duplicate warnings which would likely lead to spam.

9.12.4.7 OPENED_LOG_AFTER_FAILURE_MESSAGE

```
constexpr const char* Logger::OPENED_LOG_AFTER_FAILURE_MESSAGE = "Initialization could not  
open logging file. Succeeded in reopening.\n" [static], [private]
```

9.12.4.8 PROGRAM_EXIT_MESSAGE

```
constexpr const char* Logger::PROGRAM_EXIT_MESSAGE = "The program has exited successfully."  
[static], [private]
```

9.12.4.9 UNKNOWN_LOG_DEGREE_MESSAGE

```
constexpr const char* Logger::UNKNOWN_LOG_DEGREE_MESSAGE = "Invalid Log Degree Specified.\n"
[static], [private]
```

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

- [Logger.h](#)
- [Logger.cpp](#)

9.13 Mouse Struct Reference

This class contains information about the user's mouse.

```
#include <GFramework.h>
```

Public Member Functions

- [Mouse](#) ()

Public Attributes

- int [x](#)
x coordinate of mouse (from left to right)
- int [y](#)
y coordinate of mouse (from top to bottom)
- bool [clicked](#)
Is the mouse clicked?
- bool [heldDown](#)
Is the mouse button held down?

9.13.1 Detailed Description

This class contains information about the user's mouse.

Specifically, its position, whether it has been clicked, and if it is held down. It is not fully implemented.

9.13.2 Constructor & Destructor Documentation

9.13.2.1 Mouse()

```
Mouse::Mouse ( ) [inline]
```

9.13.3 Member Data Documentation

9.13.3.1 clicked

```
bool Mouse::clicked
```

Is the mouse clicked?

9.13.3.2 heldDown

```
bool Mouse::heldDown
```

Is the mouse button held down?

9.13.3.3 x

```
int Mouse::x
```

x coordinate of mouse (from left to right)

9.13.3.4 y

```
int Mouse::y
```

y coordinate of mouse (from top to bottom)

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

- [GFramework.h](#)

9.14 Simulation Class Reference

```
#include <Simulation.h>
```

Public Member Functions

- void [run](#) ()
- void [init](#) ()
- void [setGameMode](#) ([GameMode](#) g)
- void [setInputType](#) ([InputType](#) t)
- [Simulation](#) ()
- [~Simulation](#) ()

Public Attributes

- [GameMode](#) [gameMode](#)
- [InputType](#) [inputType](#)

Private Member Functions

- void [setInputTypeKeyboard](#) ()
- void [loadGameModeKeyboard](#) ()

Static Private Attributes

- static constexpr [GameMode](#) [INITIAL_GAME_MODE](#) = [GameMode::SIMULATION](#)
- static constexpr [InputType](#) [INITIAL_INPUT_TYPE](#) = [InputType::DEFAULT](#)
- static constexpr const char * [UNKNOWN_INPUT_TYPE_MESSAGE](#) = "UNKNOWN [INPUT](#) TYPE KEYB↵
OARD SELECTED."
- static constexpr const char * [UNKNOWN_GAME_MODE_MESSAGE](#) = "UNKNOWN GAMEMODE SELE↵
CTED."

9.14.1 Constructor & Destructor Documentation

9.14.1.1 [Simulation](#)()

```
Simulation::Simulation ( )
```

9.14.1.2 [~Simulation](#)()

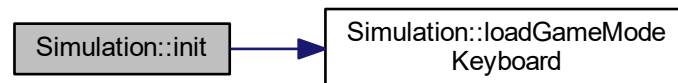
```
Simulation::~Simulation ( )
```

9.14.2 Member Function Documentation

9.14.2.1 init()

```
void Simulation::init ( )
```

Here is the call graph for this function:



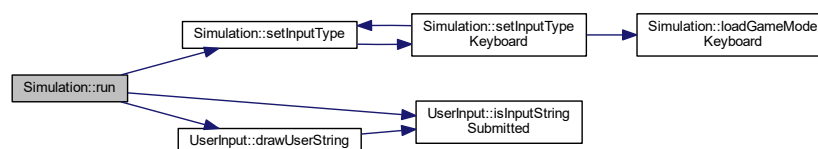
9.14.2.2 loadGameModeKeyboard()

```
void Simulation::loadGameModeKeyboard ( ) [private]
```

9.14.2.3 run()

```
void Simulation::run ( )
```

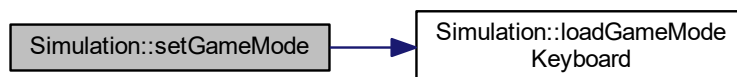
Here is the call graph for this function:



9.14.2.4 setGameMode()

```
void Simulation::setGameMode (
    GameMode g )
```

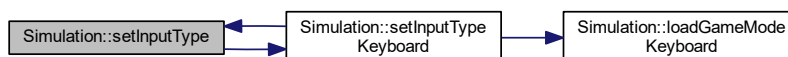
Here is the call graph for this function:



9.14.2.5 setInputType()

```
void Simulation::setInputType (
    InputType t )
```

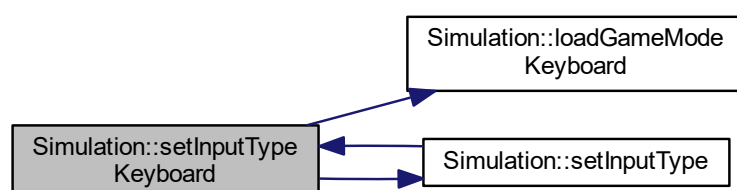
Here is the call graph for this function:



9.14.2.6 setInputTypeKeyboard()

```
void Simulation::setInputTypeKeyboard ( ) [private]
```

Here is the call graph for this function:



9.14.3 Member Data Documentation

9.14.3.1 gameMode

`GameMode` `Simulation::gameMode`

9.14.3.2 INITIAL_GAME_MODE

`constexpr GameMode` `Simulation::INITIAL_GAME_MODE` = `GameMode::SIMULATION` `[static]`, `[private]`

9.14.3.3 INITIAL_INPUT_TYPE

`constexpr InputType` `Simulation::INITIAL_INPUT_TYPE` = `InputType::DEFAULT` `[static]`, `[private]`

9.14.3.4 inputType

`InputType` `Simulation::inputType`

9.14.3.5 UNKNOWN_GAME_MODE_MESSAGE

`constexpr const char*` `Simulation::UNKNOWN_GAME_MODE_MESSAGE` = `"UNKNOWN GAMEMODE SELECTED."`
`[static]`, `[private]`

9.14.3.6 UNKNOWN_INPUT_TYPE_MESSAGE

`constexpr const char*` `Simulation::UNKNOWN_INPUT_TYPE_MESSAGE` = `"UNKNOWN INPUT TYPE KEYBOARD S↵
ELECTED."` `[static]`, `[private]`

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

- [Simulation.h](#)
- [Simulation.cpp](#)

9.15 UserFunction Struct Reference

```
#include <UserFunction.h>
```

Public Member Functions

- [UserFunction](#) ()
- [UserFunction](#) (char key_t, [Action](#) action_t)
- [UserFunction](#) (int specialKey_t, [Action](#) action_t)
- [UserFunction](#) (int specialKey_t, [Action](#) action_t, [Action](#) release_t)
- [~UserFunction](#) ()

Public Attributes

- int [specialKey](#)
- char [key](#)
- [Action](#) [action](#)
- [Action](#) [release](#)

9.15.1 Constructor & Destructor Documentation

9.15.1.1 [UserFunction\(\)](#) [1/4]

```
UserFunction::UserFunction ( )
```

9.15.1.2 [UserFunction\(\)](#) [2/4]

```
UserFunction::UserFunction (
    char key_t,
    Action action_t )
```

9.15.1.3 [UserFunction\(\)](#) [3/4]

```
UserFunction::UserFunction (
    int specialKey_t,
    Action action_t )
```

9.15.1.4 UserFunction() [4/4]

```
UserFunction::UserFunction (
    int specialKey_t,
    Action action_t,
    Action release_t )
```

9.15.1.5 ~UserFunction()

```
UserFunction::~~UserFunction ( )
```

9.15.2 Member Data Documentation

9.15.2.1 action

```
Action UserFunction::action
```

9.15.2.2 key

```
char UserFunction::key
```

9.15.2.3 release

```
Action UserFunction::release
```

9.15.2.4 specialKey

```
int UserFunction::specialKey
```

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

- [UserFunction.h](#)
- [UserFunction.cpp](#)

9.16 UserInput Class Reference

```
#include <UserInput.h>
```

Public Member Functions

- [UserInput](#) ()
- [~UserInput](#) ()
- void [setDefault](#) ()
- bool [isInputStringSubmitted](#) ()
- void [submitInputString](#) ()
- void [drawUserString](#) (double x, double y, bool xCenter=true, bool yCenter=true)

Public Attributes

- std::string [inputString](#)
- std::vector< [UserFunction](#) > [functions](#)
- bool [keyStates](#) [NUM_KEYS] = {false}
- bool [keyInputsHeld](#)

Static Private Attributes

- static int constexpr [NUM_KEYS](#) = 256
- static char constexpr [TERMINATING_CHAR](#) = '#'

9.16.1 Constructor & Destructor Documentation

9.16.1.1 UserInput()

```
UserInput::UserInput ( )
```

9.16.1.2 ~UserInput()

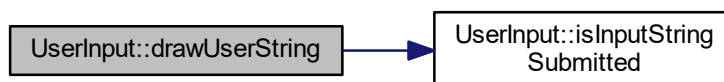
```
UserInput::~~UserInput ( )
```

9.16.2 Member Function Documentation

9.16.2.1 drawUserString()

```
void UserInput::drawUserString (
    double x,
    double y,
    bool xCenter = true,
    bool yCenter = true )
```

Here is the call graph for this function:



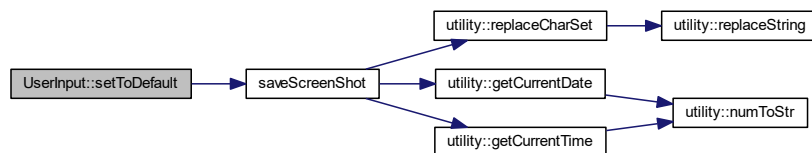
9.16.2.2 isInputStringSubmitted()

```
bool UserInput::isInputStringSubmitted ( )
```

9.16.2.3 setToDefault()

```
void UserInput::setToDefault ( )
```

Here is the call graph for this function:



9.16.2.4 submitInputString()

```
void UserInput::submitInputString ( )
```

9.16.3 Member Data Documentation

9.16.3.1 functions

```
std::vector<UserFunction> UserInput::functions
```

9.16.3.2 inputString

```
std::string UserInput::inputString
```

9.16.3.3 keyInputIsHeld

```
bool UserInput::keyInputIsHeld
```

9.16.3.4 keyStates

```
bool UserInput::keyStates[NUM_KEYS] = {false}
```

9.16.3.5 NUM_KEYS

```
int constexpr UserInput::NUM_KEYS = 256 [static], [private]
```

9.16.3.6 TERMINATING_CHAR

```
char constexpr UserInput::TERMINATING_CHAR = '#' [static], [private]
```

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

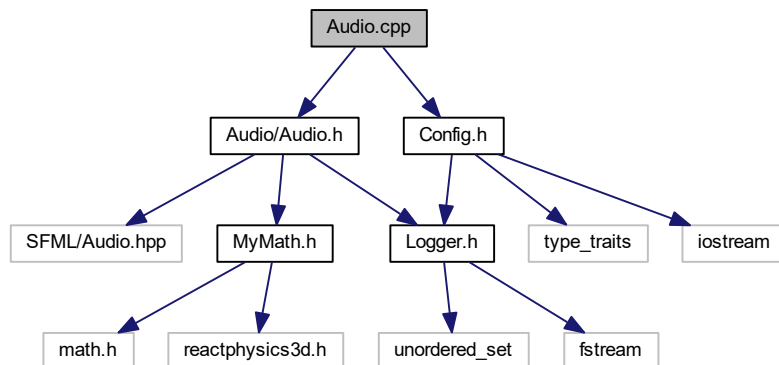
- [UserInput.h](#)
- [UserInput.cpp](#)

Chapter 10

File Documentation

10.1 Audio.cpp File Reference

```
#include "Audio/Audio.h"  
#include "Config.h"  
Include dependency graph for Audio.cpp:
```

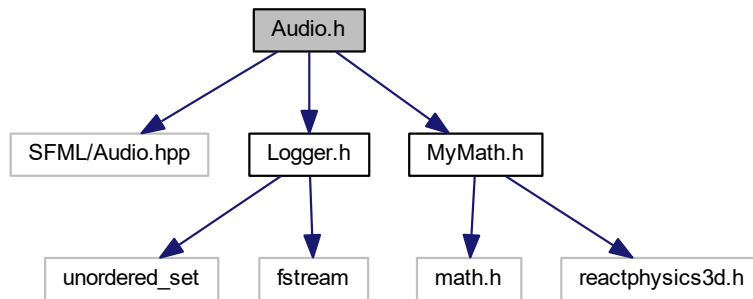


10.2 Audio.h File Reference

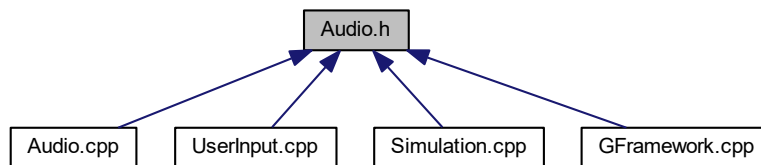
```
#include <SFML/Audio.hpp>  
#include "Logger.h"
```

```
#include "MyMath.h"
```

Include dependency graph for Audio.h:



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



Classes

- class [Audio](#)

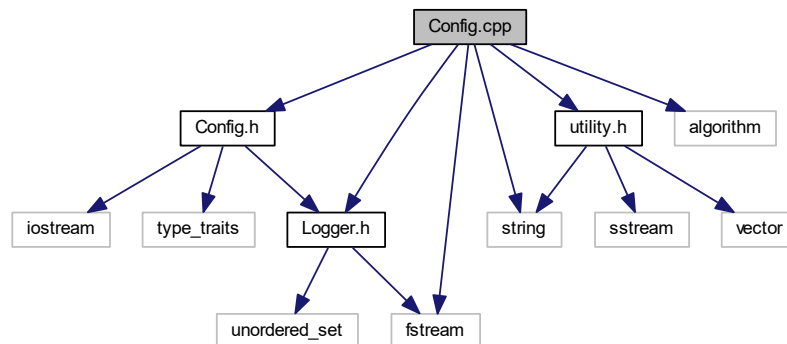
The audio engine.

10.3 Config.cpp File Reference

```
#include "Config.h"
#include "Logger.h"
#include "utility.h"
#include <fstream>
#include <algorithm>
```

```
#include <string>
```

Include dependency graph for Config.cpp:



Namespaces

- [ConfigValueConverters](#)

This namespace contains the functions which convert values to their respective type.

Functions

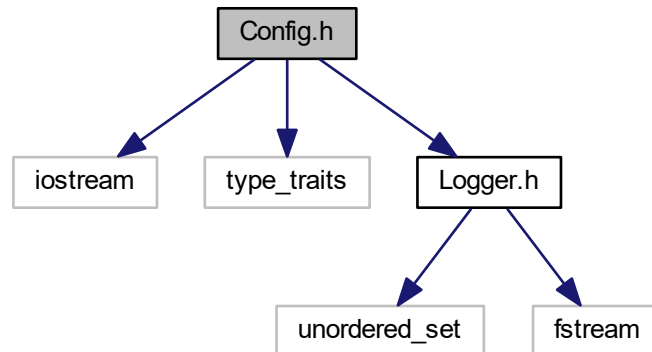
- `template<>`
`bool ConfigValueConverters::getValue_< bool > (std::string value)`
Template specialization, returns value as boolean.
- `template<>`
`int ConfigValueConverters::getValue_< int > (std::string value)`
Template specialization, returns value as boolean.
- `template<>`
`double ConfigValueConverters::getValue_< double > (std::string value)`
Template specialization, returns value as boolean.
- `template<>`
`std::string ConfigValueConverters::getValue_< std::string > (std::string value)`
Template specialization, returns value as boolean.

10.4 Config.h File Reference

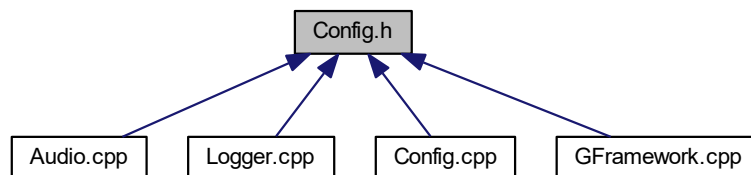
```
#include <iostream>
#include <type_traits>
```

```
#include "Logger.h"
```

Include dependency graph for Config.h:



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



Classes

- class [Config](#)
This class allows variables to be loaded from a file.

Namespaces

- [ConfigValueConverters](#)
This namespace contains the functions which convert values to their respective type.

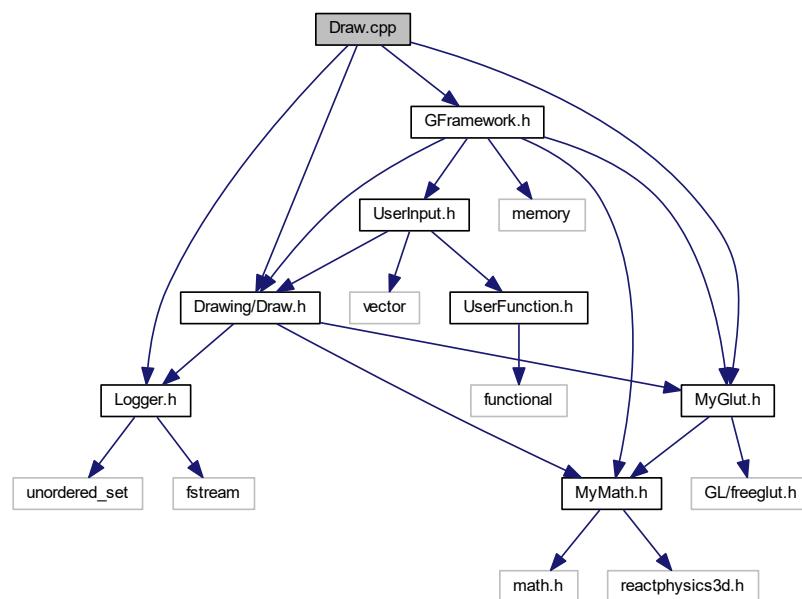
Functions

- `template<typename T>`
`T ConfigValueConverters::getValue_(std::string value)`
Templated function to return the value as the specified type T.

- `template<>`
`bool ConfigValueConverters::getValue_< bool > (std::string value)`
Template specialization, returns value as boolean.
- `template<>`
`int ConfigValueConverters::getValue_< int > (std::string value)`
Template specialization, returns value as boolean.
- `template<>`
`double ConfigValueConverters::getValue_< double > (std::string value)`
Template specialization, returns value as boolean.
- `template<>`
`std::string ConfigValueConverters::getValue_< std::string > (std::string value)`
Template specialization, returns value as boolean.

10.5 Draw.cpp File Reference

```
#include "Drawing/Draw.h"
#include "MyGlut.h"
#include "GFramework.h"
#include "Logger.h"
Include dependency graph for Draw.cpp:
```

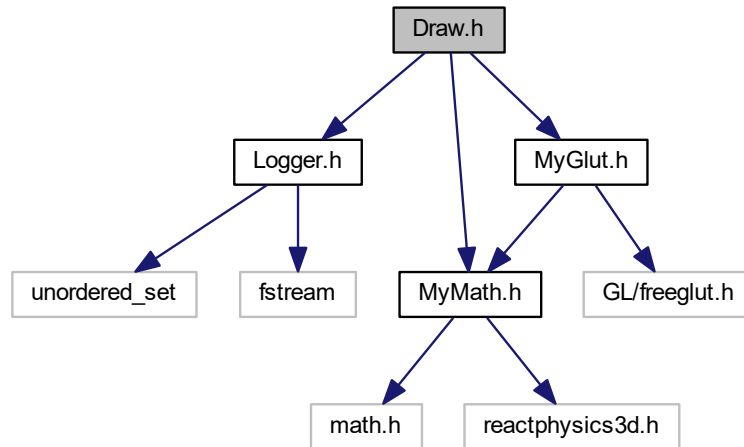


10.6 Draw.h File Reference

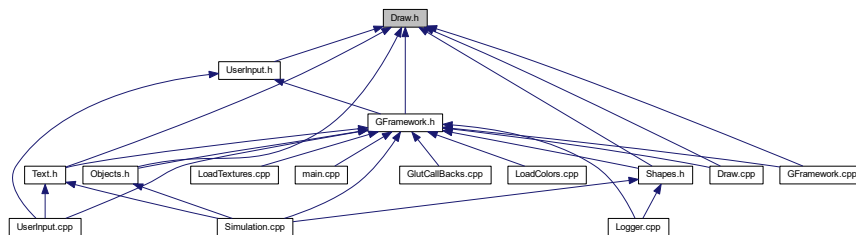
```
#include "Logger.h"
#include "MyMath.h"
```

```
#include "MyGlut.h"
```

Include dependency graph for Draw.h:



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



Classes

- struct [Drawing](#)
- class [DrawItem< A >](#)

The parent class of items which can drawn.

Enumerations

- enum [Appearance](#) {
[MAROON](#), [DARK_RED](#), [BROWN](#), [FIREBRICK](#),
[CRIMSON](#), [RED](#), [TOMATO](#), [CORAL](#),
[INDIAN_RED](#), [LIGHT_CORAL](#), [DARK_SALMON](#), [SALMON](#),
[LIGHT_SALMON](#), [ORANGE_RED](#), [DARK_ORANGE](#), [ORANGE](#),
[GOLD](#), [DARK_GOLDEN_ROD](#), [GOLDEN_ROD](#), [PALE_GOLDEN_ROD](#),
[DARK_KHAKI](#), [KHAKI](#), [OLIVE](#), [YELLOW](#),
[YELLOW_GREEN](#), [DARK_OLIVE_GREEN](#), [OLIVE_DRAB](#), [LAWN_GREEN](#),

```

CHART_REUSE, GREEN_YELLOW, DARK_GREEN, GREEN,
FOREST_GREEN, LIME, LIME_GREEN, LIGHT_GREEN,
PALE_GREEN, DARK_SEA_GREEN, MEDIUM_SPRING_GREEN, SPRING_GREEN,
SEA_GREEN, MEDIUM_AQUA_MARINE, MEDIUM_SEA_GREEN, LIGHT_SEA_GREEN,
DARK_SLATE_GRAY, TEAL, DARK_CYAN, AQUA,
CYAN, LIGHT_CYAN, DARK_TURQUOISE, TURQUOISE,
MEDIUM_TURQUOISE, PALE_TURQUOISE, AQUA_MARINE, POWDER_BLUE,
CADET_BLUE, STEEL_BLUE, CORN_FLOWER_BLUE, DEEP_SKY_BLUE,
DODGER_BLUE, LIGHT_BLUE, SKY_BLUE, LIGHT_SKY_BLUE,
MIDNIGHT_BLUE, NAVY, DARK_BLUE, MEDIUM_BLUE,
BLUE, ROYAL_BLUE, BLUE_VIOLET, INDIGO,
DARK_SLATE_BLUE, SLATE_BLUE, MEDIUM_SLATE_BLUE, MEDIUM_PURPLE,
DARK_MAGENTA, DARK_VIOLET, DARK_ORCHID, MEDIUM_ORCHID,
PURPLE, THISTLE, PLUM, VIOLET,
MAGENTA, ORCHID, MEDIUM_VIOLET_RED, PALE_VIOLET_RED,
DEEP_PINK, HOT_PINK, LIGHT_PINK, PINK,
ANTIQUE_WHITE, BEIGE, BISQUE, BLANCHED_ALMOND,
WHEAT, CORN_SILK, LEMON_CHIFFON, LIGHT_GOLDEN_ROD_YELLOW,
LIGHT_YELLOW, SADDLE_BROWN, SIENNA, CHOCOLATE,
PERU, SANDY_BROWN, BURLY_WOOD, TAN,
ROSY_BROWN, MOCCASIN, NAVAJO_WHITE, PEACH_PUFF,
MISTY_ROSE, LAVENDER_BLUSH, LINEN, OLD_LACE,
PAPAYA_WHIP, SEA_SHELL, MINT_CREAM, SLATE_GRAY,
LIGHT_SLATE_GRAY, LIGHT_STEEL_BLUE, LAVENDER, FLORAL_WHITE,
ALICE_BLUE, GHOST_WHITE, HONEYDEW, IVORY,
AZURE, SNOW, BLACK, DIM_GRAY,
GRAY, DARK_GRAY, SILVER, LIGHT_GRAY,
GAINSBORO, WHITE_SMOKE, WHITE, LAST_COLOR__,
GRASS, BUTTON, LAST_TEXTURE__}

```

Enums corresponding to the appearance of drawn objects.

10.6.1 Enumeration Type Documentation

10.6.1.1 Appearance

```
enum Appearance
```

Enums corresponding to the appearance of drawn objects.

The desired drawing call is

```
DrawObject<Appearance::STYLE> (...)
```

To allow for this, these appearances are stored in this enum and are passed to drawing functions as a template parameter. I'm not sure if this is the ideal implementation method.

To allow for both colors and textures to be applied, they are stored in the same enum despite being treated differently. The enums "LAST_COLOR_" and "LAST_TEXTURE_" are ONLY to be used by the [Drawing::isColor](#), and [Drawing::isTexture](#) methods. Unfortunately this implementation detail can't be hidden with this infrastructure.

See also

[Drawing::isColor](#), [Drawing::isTexture](#).

Enumerator

MAROON	
DARK_RED	
BROWN	
FIREBRICK	
CRIMSON	
RED	
TOMATO	
CORAL	
INDIAN_RED	
LIGHT_CORAL	
DARK_SALMON	
SALMON	
LIGHT_SALMON	
ORANGE_RED	
DARK_ORANGE	
ORANGE	
GOLD	
DARK_GOLDEN_ROD	
GOLDEN_ROD	
PALE_GOLDEN_ROD	
DARK_KHAKI	
KHAKI	
OLIVE	
YELLOW	
YELLOW_GREEN	
DARK_OLIVE_GREEN	
OLIVE_DRAB	
LAWN_GREEN	
CHART_REUSE	
GREEN_YELLOW	
DARK_GREEN	
GREEN	
FOREST_GREEN	
LIME	
LIME_GREEN	
LIGHT_GREEN	
PALE_GREEN	
DARK_SEA_GREEN	
MEDIUM_SPRING_GREEN	
SPRING_GREEN	
SEA_GREEN	
MEDIUM_AQUA_MARINE	
MEDIUM_SEA_GREEN	
LIGHT_SEA_GREEN	
DARK_SLATE_GRAY	
TEAL	
DARK_CYAN	
AQUA	
CYAN	
LIGHT_CYAN	

Enumerator

DARK_TURQUOISE	
TURQUOISE	
MEDIUM_TURQUOISE	
PALE_TURQUOISE	
AQUA_MARINE	
POWDER_BLUE	
CADET_BLUE	
STEEL_BLUE	
CORN_FLOWER_BLUE	
DEEP_SKY_BLUE	
DODGER_BLUE	
LIGHT_BLUE	
SKY_BLUE	
LIGHT_SKY_BLUE	
MIDNIGHT_BLUE	
NAVY	
DARK_BLUE	
MEDIUM_BLUE	
BLUE	
ROYAL_BLUE	
BLUE_VIOLET	
INDIGO	
DARK_SLATE_BLUE	
SLATE_BLUE	
MEDIUM_SLATE_BLUE	
MEDIUM_PURPLE	
DARK_MAGENTA	
DARK_VIOLET	
DARK_ORCHID	
MEDIUM_ORCHID	
PURPLE	
THISTLE	
PLUM	
VIOLET	
MAGENTA	
ORCHID	
MEDIUM_VIOLET_RED	
PALE_VIOLET_RED	
DEEP_PINK	
HOT_PINK	
LIGHT_PINK	
PINK	
ANTIQUE_WHITE	
BEIGE	
BISQUE	
BLANCHED_ALMOND	
WHEAT	
CORN_SILK	
LEMON_CHIFFON	
LIGHT_GOLDEN_ROD_YELLOW	

Enumerator

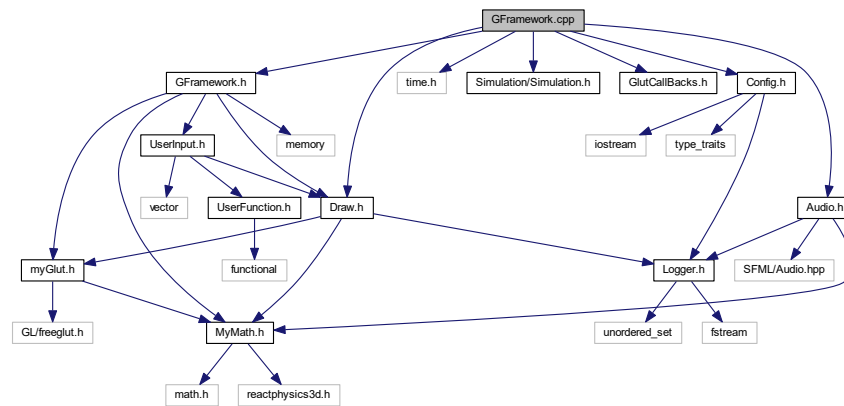
LIGHT_YELLOW	
SADDLE_BROWN	
SIENNA	
CHOCOLATE	
PERU	
SANDY_BROWN	
BURLY_WOOD	
TAN	
ROSY_BROWN	
MOCCASIN	
NAVAJO_WHITE	
PEACH_PUFF	
MISTY_ROSE	
LAVENDER_BLUSH	
LINEN	
OLD_LACE	
PAPAYA_WHIP	
SEA_SHELL	
MINT_CREAM	
SLATE_GRAY	
LIGHT_SLATE_GRAY	
LIGHT_STEEL_BLUE	
LAVENDER	
FLORAL_WHITE	
ALICE_BLUE	
GHOST_WHITE	
HONEYDEW	
IVORY	
AZURE	
SNOW	
BLACK	
DIM_GRAY	
GRAY	
DARK_GRAY	
SILVER	
LIGHT_GRAY	
GAINSBORO	
WHITE_SMOKE	
WHITE	
LAST_COLOR__	
GRASS	
BUTTON	
LAST_TEXTURE__	

10.7 GFramework.cpp File Reference

```
#include "GFramework.h"
#include <time.h>
```

```
#include "Simulation/Simulation.h"
#include "GlutCallbacks.h"
#include "Drawing/Draw.h"
#include "Config.h"
#include "Audio.h"
```

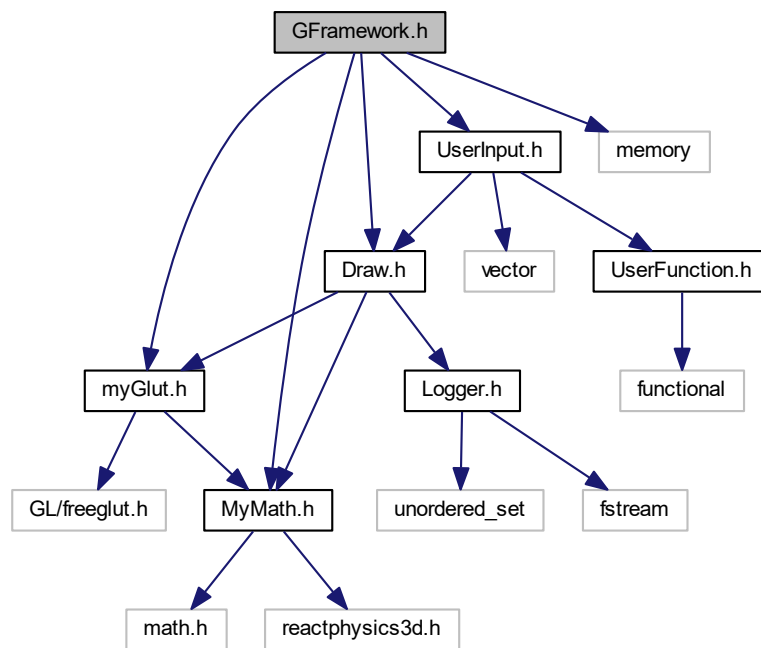
Include dependency graph for GFramework.cpp:



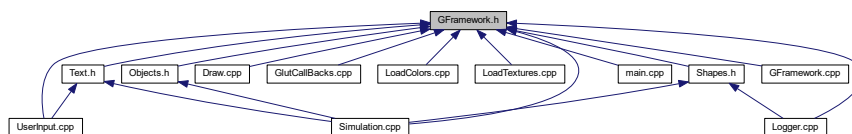
10.8 GFramework.h File Reference

```
#include "myGlut.h"
#include "Math/MyMath.h"
#include "UserInput.h"
#include "Drawing/Draw.h"
#include <memory>
```

Include dependency graph for GFramework.h:



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



Classes

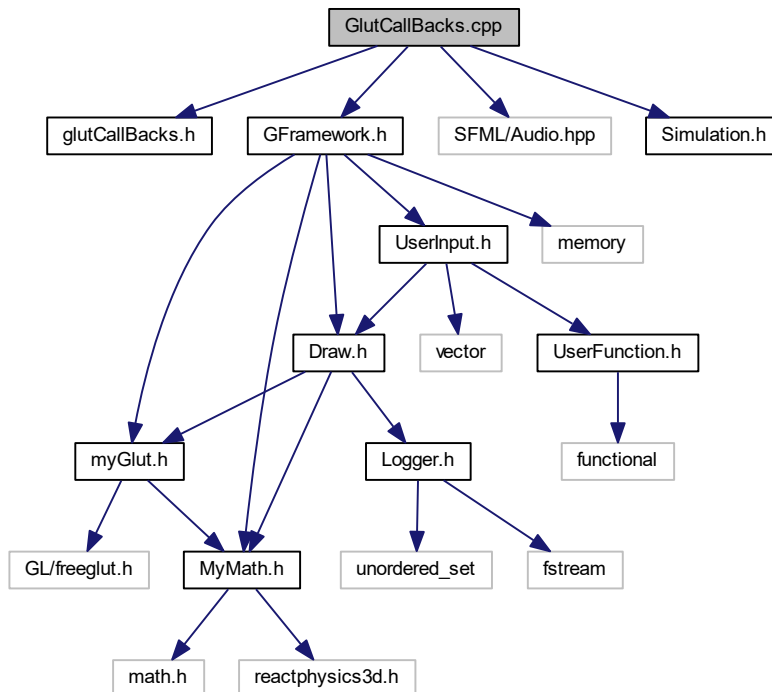
- struct [Camera](#)
This class describes the camera placed at the head of the user.
- struct [Mouse](#)
This class contains information about the user's mouse.
- class [GFramework](#)
This class holds all the functions required for a 3D simulator to be run.

10.9 GlutCallbacks.cpp File Reference

```
#include "glutCallbacks.h"
#include "GFramework.h"
```

```
#include <SFML/Audio.hpp>
#include "Simulation.h"
```

Include dependency graph for GlutCallbacks.cpp:



Namespaces

- [glutCB](#)

This namespace contains the glut callback functions.

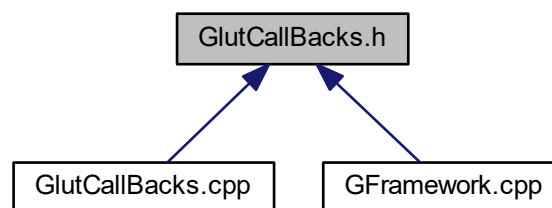
Functions

- void [glutCB::changeSize](#) (int w, int h)
Updates the opengl viewport etc on window resize.
- void [glutCB::renderScene](#) ()
Prepares the drawing state, then calls the simulation at a constant FPS.
- void [glutCB::update](#) ()
Updates the audio, window size, userfunctions (ie: calls them), and camera.
- void [glutCB::callMouse](#) (int button, int state, int mx, int my)
Is called on mouse click.
- void [glutCB::mouseMove](#) (int mx, int my)
Is called on mouse movement.
- void [glutCB::passiveMouse](#) (int mx, int my)
Continuously called to update details about the mouse.
- void [glutCB::keyPressed](#) (unsigned char key, int x, int y)
Sets the state of the [UserInput](#) keystates for the pressed key and calls its associated pressed function.

- void `glutCB::keyUp` (unsigned char key, int x, int y)
Resets the state of the `UserInput` keystates for the released key and calls its associated release function.
- void `glutCB::pressSpecialKey` (int key, int kxx, int kyy)
Calls special key (ex: Arrow keys) press functions.
- void `glutCB::releaseSpecialKey` (int key, int kx, int ky)
Calls the special key (ex: Arrow keys) release functions.

10.10 GlutCallbacks.h File Reference

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



Namespaces

- `glutCB`
This namespace contains the glut callback functions.

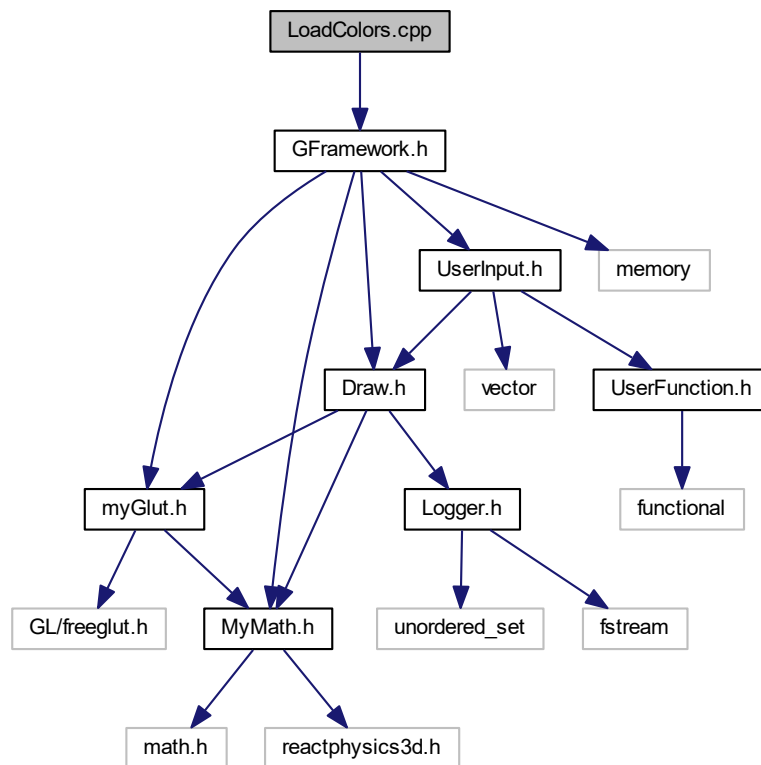
Functions

- void `glutCB::renderScene` ()
Prepares the drawing state, then calls the simulation at a constant FPS.
- void `glutCB::update` ()
Updates the audio, window size, userfunctions (ie: calls them), and camera.
- void `glutCB::changeSize` (int w, int h)
Updates the opengl viewport etc on window resize.
- void `glutCB::callMouse` (int button, int state, int mx, int my)
Is called on mouse click.
- void `glutCB::mouseMove` (int mx, int my)
Is called on mouse movement.
- void `glutCB::passiveMouse` (int mx, int my)
Continuously called to update details about the mouse.
- void `glutCB::keyPressed` (unsigned char key, int x, int y)
Sets the state of the `UserInput` keystates for the pressed key and calls its associated pressed function.
- void `glutCB::keyUp` (unsigned char key, int x, int y)
Resets the state of the `UserInput` keystates for the released key and calls its associated release function.
- void `glutCB::pressSpecialKey` (int key, int kxx, int kyy)
Calls special key (ex: Arrow keys) press functions.
- void `glutCB::releaseSpecialKey` (int key, int kx, int ky)
Calls the special key (ex: Arrow keys) release functions.

10.11 LoadColors.cpp File Reference

```
#include "GFramework.h"
```

Include dependency graph for LoadColors.cpp:



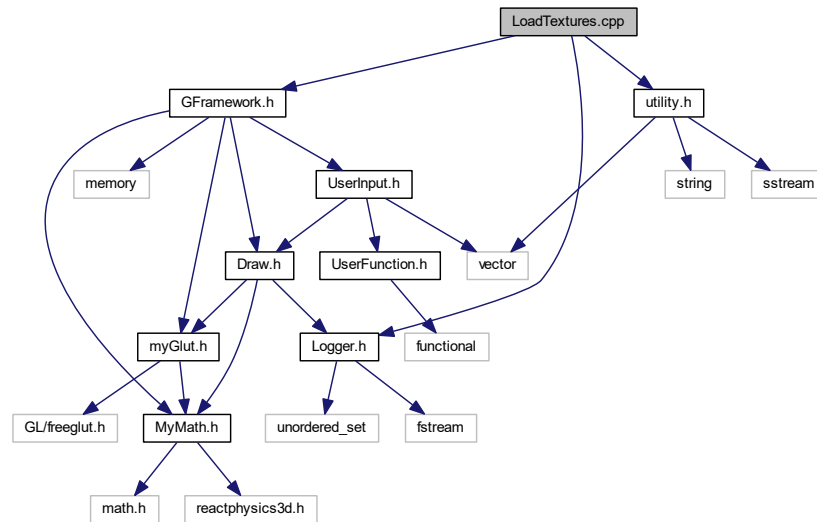
10.12 LoadTextures.cpp File Reference

```
#include "GFramework.h"
```

```
#include "Logger.h"
```

```
#include "utility.h"
```

Include dependency graph for LoadTextures.cpp:



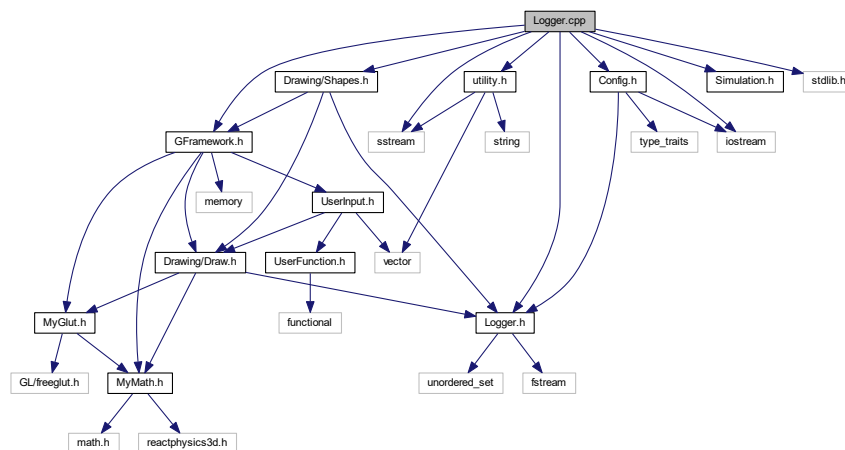
10.13 Logger.cpp File Reference

```

#include "Logger.h"
#include "Drawing/Shapes.h"
#include "GFramework.h"
#include "utility.h"
#include "Config.h"
#include "Simulation.h"
#include <iostream>
#include <stdlib.h>
#include <sstream>

```

Include dependency graph for Logger.cpp:



10.14.1 Macro Definition Documentation

10.14.1.1 LOG

```
#define LOG(  
    ... ) Logger::get().log(__LINE__, __FILE__, __FUNCTION__, __VA_ARGS__)
```

This macro passes the line, file, and function to the logger, along with additional parameters.

It effectively acts like a decorator and allows more detailed logging information to be passed to the logger. Should be called like so (the last two parameters are optional):

```
LOG("This is my message.", LogDegree::DEBUG, LogType::GENERAL);
```

Parameters

<i>msg</i>	The message to be printed with the log.
<i>You</i>	may optionally pass a LogDegree, and also LogType.

Returns

void

Warning

Type and argument checking is lacking due to macro implementation. It may be difficult to determine the incorrect call to this macro if incorrect parameters are passed. This is because the compiler redirects you to this definition instead of the culprit.

See also

[Logger::log](#)

Note

Although the input is '...' you are required to specify a message. This is to prevent 'c99 requires rest arguments to be used' error.

10.14.1.2 NORMAL_EXIT

```
#define NORMAL_EXIT( ) Logger::get().normalExit(__LINE__, __FILE__, __FUNCTION__)
```

Exits the program normally, but also logs the line, file and function where the exiting call came from.

Returns

void

Warning

Type and argument checking is lacking due to macro implementation. It may be difficult to determine the incorrect call to this macro if incorrect parameters are passed. This is because the compiler redirects you to this definition instead of the culprit.

See also

[Logger::normalExit](#)

10.14.2 Enumeration Type Documentation

10.14.2.1 LogDegree

```
enum LogDegree : unsigned int [strong]
```

Used to specify different levels of logging severity.

See also

[Logger::log](#)

Enumerator

DEBUG	
WARNING	
FATAL	

10.14.2.2 LogType

```
enum LogType : unsigned int [strong]
```

Used to specify different types of logging message (ie: the reason for the log).

See also

[Logger::log](#)

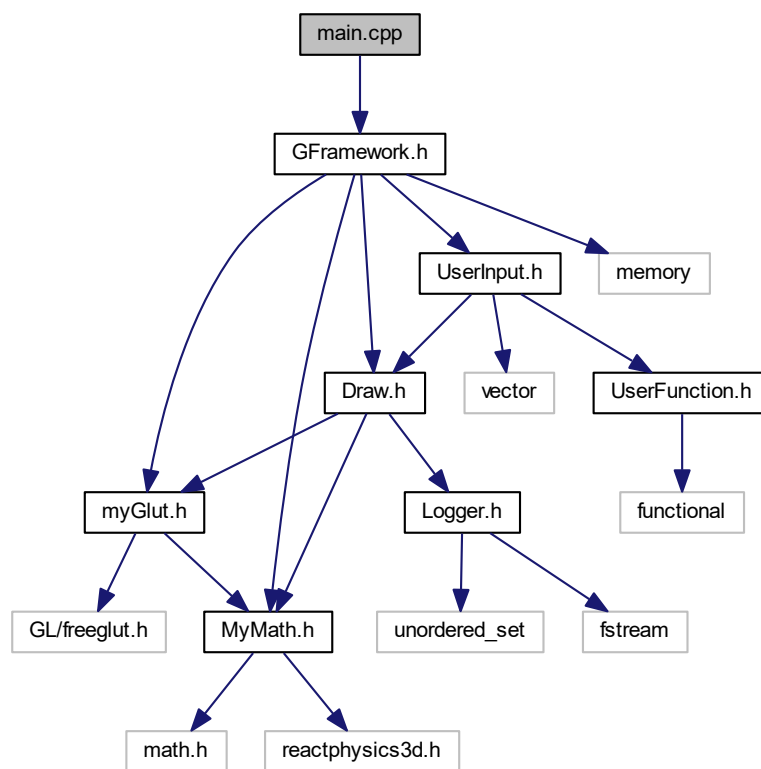
Enumerator

GENERAL	
GRAPHICS	
DISPLAY	
INPUT	
AUDIO	
CONFIG	

10.15 main.cpp File Reference

```
#include "GFramework.h"
```

Include dependency graph for main.cpp:



Functions

- `int main ()`

Starts up the glutmainloop through [GFramework](#) `startup()`.

10.15.1 Function Documentation

10.15.1.1 main()

```
int main ( )
```

Starts up the glutmainloop through [GFramework](#) startup().

The opengl framework works by running a set of disconnected functions in a loop. To do this, the functions are divided and only specific variables can be passed between them. This forces a new way of programming moving away from the typical gameloop constructs. Instead the use of (a attempted minimal) usage of singletons, static and global variables must be implemented. It seems that the best way forward is to create a singleton instance (of the [GFramework](#) class) which encapsulates all the opengl callbacks. This class effectively acts as a new main.

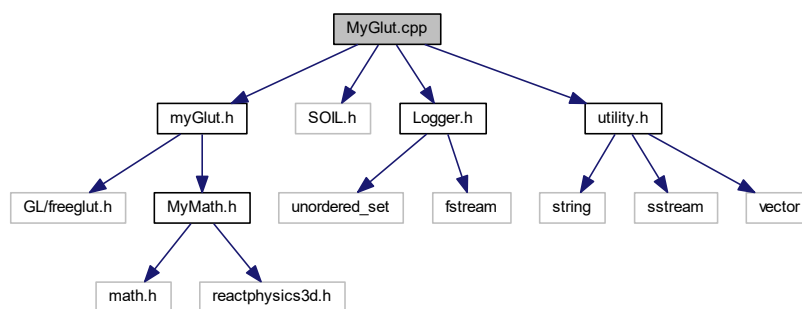
Returns

int Exit code, 0 for normal termination, non-zero value otherwise.

10.16 MyGlut.cpp File Reference

```
#include "myGlut.h"
#include "SOIL.h"
#include "Logger.h"
#include "utility.h"
```

Include dependency graph for MyGlut.cpp:



Functions

- [Tex glLoadTexture](#) (std::string fileName)
Loads a texture using SOIL and returns a reference to it.
- void [saveScreenShot](#) ()
Saves the current scene as a .png and stores it in the screenshots folder.
- void [glTexVert2f](#) (double tx, double ty, double vx, double vy)
Draws a 2D textured vertex.

- void `glTexVert2f` (`Vec2` t, double vx, double vy)
Draws a 2D textured vertex.
- void `glTexVert2f` (double tx, double ty, `Vec2` v)
Draws a 2D textured vertex.
- void `glTexVert2f` (`Vec2` t, `Vec2` v)
Draws a 2D textured vertex.
- void `glTexVert3f` (double tx, double ty, double vx, double vy, double vz)
Draws a 2D textured vertex.
- void `glTexVert3f` (`Vec2` t, double vx, double vy, double vz)
Draws a 2D textured vertex.
- void `glTexVert3f` (double tx, double ty, `Vec` v)
Draws a 2D textured vertex.
- void `glTexVert3f` (`Vec2` t, `Vec` v)
Draws a 3D textured vertex.

10.16.1 Function Documentation

10.16.1.1 `glLoadTexture()`

```
Tex glLoadTexture (
    std::string fileName )
```

Loads a texture using SOIL and returns a reference to it.

Parameters

<code>fileName</code>	std::string The name (path) of the image file to be loaded.
-----------------------	---

Returns

Tex The reference to the texture.

10.16.1.2 `glTexVert2f()` [1/4]

```
void glTexVert2f (
    double tx,
    double ty,
    double vx,
    double vy )
```

Draws a 2D textured vertex.

Parameters

<i>tx</i>	double The x texture coordinate (should be [0, 1]).
<i>ty</i>	double The y texture coordinate (should be [0, 1]).
<i>vx</i>	double The relative window x coordinate of the vertex (should be within the window width).
<i>vy</i>	double The relative window y coordinates of the vertex (should be within the window height).

Returns

void

Note

No bounds checking on input parameters since shapes might start out of bounds for a reason, but still be drawn in the scene.

Here is the call graph for this function:

**10.16.1.3 glTexVert2f()** [2/4]

```
void glTexVert2f (  
    Vec2 t,  
    double vx,  
    double vy )
```

Draws a 2D textured vertex.

Parameters

<i>t</i>	Vec2 The texture coordinates (both should be [0, 1]).
<i>vx</i>	double The relative window x coordinate of the vertex (should be within the window width).
<i>vy</i>	double The relative window y coordinates of the vertex (should be within the window height).

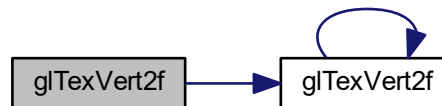
Returns

void

Note

No bounds checking on input parameters since shapes might start out of bounds for a reason, but still be drawn in the scene.

Here is the call graph for this function:

**10.16.1.4 glVertex2f()** [3/4]

```
void glVertex2f (  
    double tx,  
    double ty,  
    Vec2 v )
```

Draws a 2D textured vertex.

Parameters

<i>tx</i>	double The x texture coordinate (should be [0, 1]).
<i>ty</i>	double The y texture coordinate (should be [0, 1]).
<i>v</i>	Vec2 The relative window coordinates of the vertex (should be within the window dimensions).

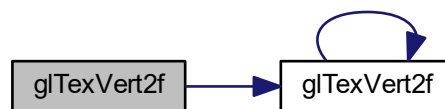
Returns

void

Note

No bounds checking on input parameters since shapes might start out of bounds for a reason, but still be drawn in the scene.

Here is the call graph for this function:

**10.16.1.5 `glTexVert2f()`** [4/4]

```
void glVertex2f (  
    Vec2 t,  
    Vec2 v )
```

Draws a 2D textured vertex.

Parameters

<i>t</i>	Vec2 The texture coordinates (both should be [0, 1]).
<i>v</i>	Vec2 The relative window coordinates (should be within the window dimensions).

Returns

void

Note

No bounds checking on input parameters since shapes might start out of bounds for a reason, but still be drawn in the scene.

10.16.1.6 `glTexVert3f()` [1/4]

```
void glVertex3f (  
    double tx,  
    double ty,  
    double vx,  
    double vy,  
    double vz )
```

Draws a 2D textured vertex.

Parameters

<i>tx</i>	double The x texture coordinates (should be [0, 1]).
<i>ty</i>	double The y texture coordinates (should be [0, 1]).
<i>vx</i>	double The x world coordinate of the vertex.
<i>vy</i>	double The y world coordinate of the vertex.
<i>vz</i>	double The z world coordinate of the vertex.

Returns

void

Note

No bounds checking on input parameters since shapes might start out of bounds for a reason, but still be drawn in the scene.

Here is the call graph for this function:

**10.16.1.7 glTexVert3f()** [2/4]

```

void glTexVert3f (
    Vec2 t,
    double vx,
    double vy,
    double vz )
  
```

Draws a 2D textured vertex.

Parameters

<i>t</i>	Vec2 The texture coordinates (both should be [0, 1]).
<i>vx</i>	double The x world coordinate of the vertex.
<i>vy</i>	double The y world coordinate of the vertex.
<i>vz</i>	double The z world coordinate of the vertex.

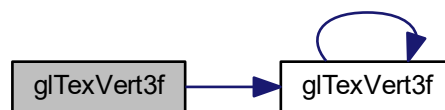
Returns

void

Note

No bounds checking on input parameters since shapes might start out of bounds for a reason, but still be drawn in the scene.

Here is the call graph for this function:

**10.16.1.8 glVertex3f()** [3/4]

```
void glVertex3f (  
    double tx,  
    double ty,  
    Vec v )
```

Draws a 2D textured vertex.

Parameters

<i>tx</i>	double The texture x coordinates (both should be [0, 1]).
<i>ty</i>	double The texture y coordinates (both should be [0, 1]).
<i>v</i>	Vec3 The world coordinates of the vertex.

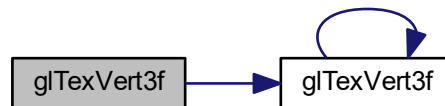
Returns

void

Note

No bounds checking on input parameters since shapes might start out of bounds for a reason, but still be drawn in the scene.

Here is the call graph for this function:

**10.16.1.9 glTexVert3f()** [4/4]

```
void glTexVert3f (
    Vec2 t,
    Vec v )
```

Draws a 3D textured vertex.

Parameters

<i>t</i>	Vec2 The texture coordinates (both should be [0, 1]).
<i>v</i>	Vec3 The world coordinates of the vertex.

Returns

void

Note

No bounds checking on input parameters since shapes might start out of bounds for a reason, but still be drawn in the scene.

10.16.1.10 saveScreenShot()

```
void saveScreenShot ( )
```

Saves the current scene as a .png and stores it in the screenshots folder.

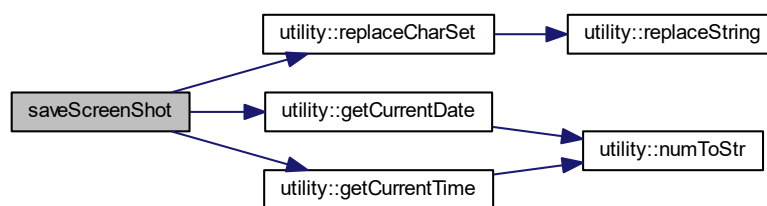
Returns

void

Note

Although it is saved as a .png, the SOIL implementation saves it as a BMP. This doesn't seem to cause any problems.

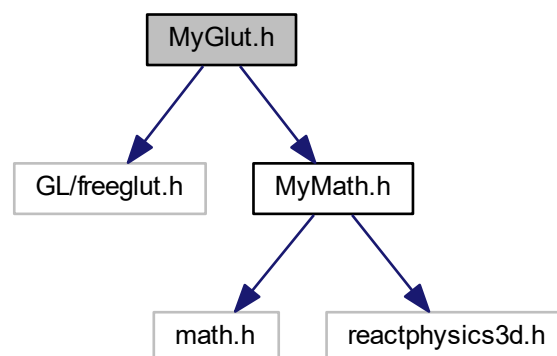
Here is the call graph for this function:



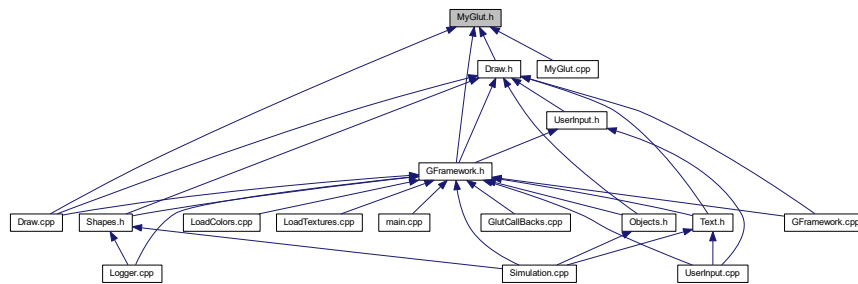
10.17 MyGlut.h File Reference

This file contains some useful extension for glut.

```
#include <GL/freeglut.h>
#include "MyMath.h"
Include dependency graph for MyGlut.h:
```



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



Typedefs

- typedef GLuint [Tex](#)
Used to refer to textures (used in the SOIL library).

Functions

- [Tex glLoadTexture](#) (std::string fileName)
Loads a texture using SOIL and returns a reference to it.
- void [saveScreenShot](#) ()
Saves the current scene as a .png and stores it in the screenshots folder.
- void [glTexVert2f](#) (Vec2 t, [Vec2](#) v)
Draws a 2D textured vertex.
- void [glTexVert2f](#) (double tx, double ty, [Vec2](#) v)
Draws a 2D textured vertex.
- void [glTexVert2f](#) ([Vec2](#) t, double vx, double vy)
Draws a 2D textured vertex.
- void [glTexVert2f](#) (double tx, double ty, double vx, double vy)
Draws a 2D textured vertex.
- void [glTexVert3f](#) ([Vec2](#) t, [Vec](#) v)
Draws a 3D textured vertex.
- void [glTexVert3f](#) (double tx, double ty, [Vec](#) v)
Draws a 2D textured vertex.
- void [glTexVert3f](#) ([Vec2](#) t, double vx, double vy, double vz)
Draws a 2D textured vertex.
- void [glTexVert3f](#) (double tx, double ty, double vx, double vy, double vz)
Draws a 2D textured vertex.

10.17.1 Detailed Description

This file contains some useful extension for glut.

Specifically SOIL and texture wrappers which make certain tasks much easier.

10.17.2 Typedef Documentation

10.17.2.1 Tex

```
typedef GLuint Tex
```

Used to refer to textures (used in the SOIL library).

10.17.3 Function Documentation

10.17.3.1 glLoadTexture()

```
Tex glLoadTexture (
    std::string fileName )
```

Loads a texture using SOIL and returns a reference to it.

Parameters

<i>fileName</i>	std::string The name (path) of the image file to be loaded.
-----------------	---

Returns

Tex The reference to the texture.

10.17.3.2 glTexVert2f() [1/4]

```
void glTexVert2f (
    Vec2 t,
    Vec2 v )
```

Draws a 2D textured vertex.

Parameters

<i>t</i>	Vec2 The texture coordinates (both should be [0, 1]).
<i>v</i>	Vec2 The relative window coordinates (should be within the window dimensions).

Returns

void

Note

No bounds checking on input parameters since shapes might start out of bounds for a reason, but still be drawn in the scene.

10.17.3.3 glTexVert2f() [2/4]

```
void glTexVert2f (  
    double tx,  
    double ty,  
    Vec2 v )
```

Draws a 2D textured vertex.

Parameters

<i>tx</i>	double The x texture coordinate (should be [0, 1]).
<i>ty</i>	double The y texture coordinate (should be [0, 1]).
<i>v</i>	Vec2 The relative window coordinates of the vertex (should be within the window dimensions).

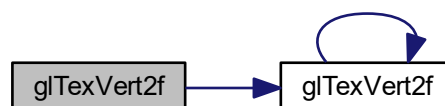
Returns

void

Note

No bounds checking on input parameters since shapes might start out of bounds for a reason, but still be drawn in the scene.

Here is the call graph for this function:



10.17.3.4 glTexVert2f() [3/4]

```
void glTexVert2f (
    Vec2 t,
    double vx,
    double vy )
```

Draws a 2D textured vertex.

Parameters

<i>t</i>	Vec2 The texture coordinates (both should be [0, 1]).
<i>vx</i>	double The relative window x coordinate of the vertex (should be within the window width).
<i>vy</i>	double The relative window y coordinates of the vertex (should be within the window height).

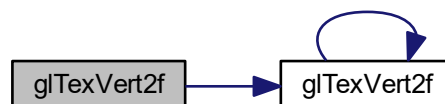
Returns

void

Note

No bounds checking on input parameters since shapes might start out of bounds for a reason, but still be drawn in the scene.

Here is the call graph for this function:



10.17.3.5 glTexVert2f() [4/4]

```
void glTexVert2f (
    double tx,
    double ty,
    double vx,
    double vy )
```

Draws a 2D textured vertex.

Parameters

<i>tx</i>	double The x texture coordinate (should be [0, 1]).
<i>ty</i>	double The y texture coordinate (should be [0, 1]).
<i>vx</i>	double The relative window x coordinate of the vertex (should be within the window width).
<i>vy</i>	double The relative window y coordinates of the vertex (should be within the window height).

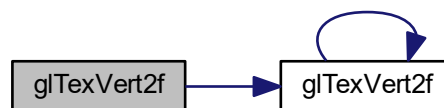
Returns

void

Note

No bounds checking on input parameters since shapes might start out of bounds for a reason, but still be drawn in the scene.

Here is the call graph for this function:

**10.17.3.6 glVertex3f()** [1/4]

```
void glVertex3f (
    Vec2 t,
    Vec v )
```

Draws a 3D textured vertex.

Parameters

<i>t</i>	Vec2 The texture coordinates (both should be [0, 1]).
<i>v</i>	Vec3 The world coordinates of the vertex.

Returns

void

Note

No bounds checking on input parameters since shapes might start out of bounds for a reason, but still be drawn in the scene.

10.17.3.7 glTexVert3f() [2/4]

```
void glTexVert3f (
    double tx,
    double ty,
    Vec v )
```

Draws a 2D textured vertex.

Parameters

<i>tx</i>	double The texture x coordinates (both should be [0, 1]).
<i>ty</i>	double The texture y coordinates (both should be [0, 1]).
<i>v</i>	Vec3 The world coordinates of the vertex.

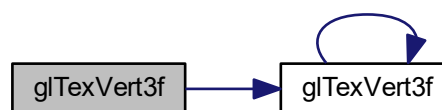
Returns

void

Note

No bounds checking on input parameters since shapes might start out of bounds for a reason, but still be drawn in the scene.

Here is the call graph for this function:

**10.17.3.8 glTexVert3f()** [3/4]

```
void glTexVert3f (
    Vec2 t,
    double vx,
    double vy,
    double vz )
```

Draws a 2D textured vertex.

Parameters

<i>t</i>	Vec2 The texture coordinates (both should be [0, 1]).
<i>vx</i>	double The x world coordinate of the vertex.
<i>vy</i>	double The y world coordinate of the vertex.
<i>vz</i>	double The z world coordinate of the vertex.

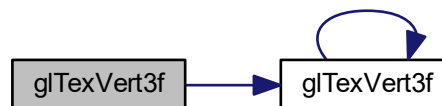
Returns

void

Note

No bounds checking on input parameters since shapes might start out of bounds for a reason, but still be drawn in the scene.

Here is the call graph for this function:

**10.17.3.9 glVertex3f()** [4/4]

```

void glVertex3f (
    double tx,
    double ty,
    double vx,
    double vy,
    double vz )
  
```

Draws a 2D textured vertex.

Parameters

<i>tx</i>	double The x texture coordinates (should be [0, 1]).
<i>ty</i>	double The y texture coordinates (should be [0, 1]).
<i>vx</i>	double The x world coordinate of the vertex.
<i>vy</i>	double The y world coordinate of the vertex.
<i>vz</i>	double The z world coordinate of the vertex.

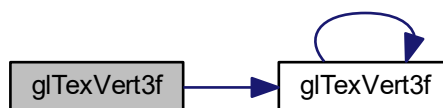
Returns

void

Note

No bounds checking on input parameters since shapes might start out of bounds for a reason, but still be drawn in the scene.

Here is the call graph for this function:

**10.17.3.10 saveScreenShot()**

```
void saveScreenShot ( )
```

Saves the current scene as a .png and stores it in the screenshots folder.

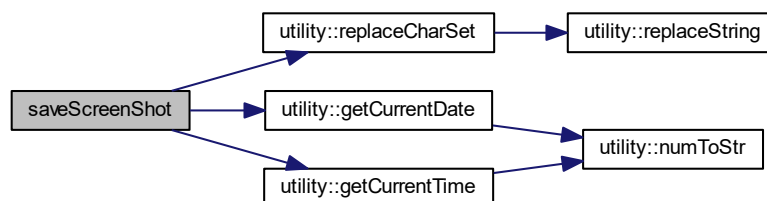
Returns

void

Note

Although it is saved as a .png, the SOIL implementation saves it as a BMP. This doesn't seem to cause any problems.

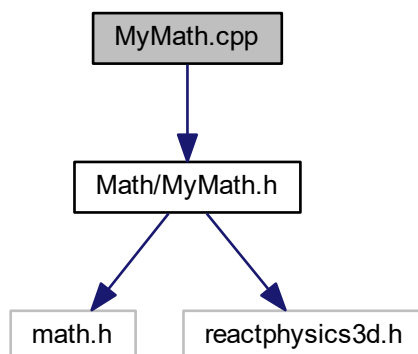
Here is the call graph for this function:



10.18 MyMath.cpp File Reference

```
#include "Math/MyMath.h"
```

Include dependency graph for MyMath.cpp:

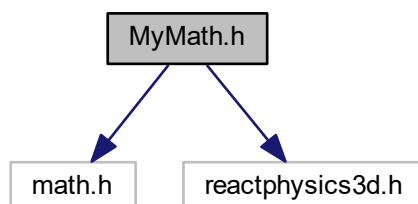


10.19 MyMath.h File Reference

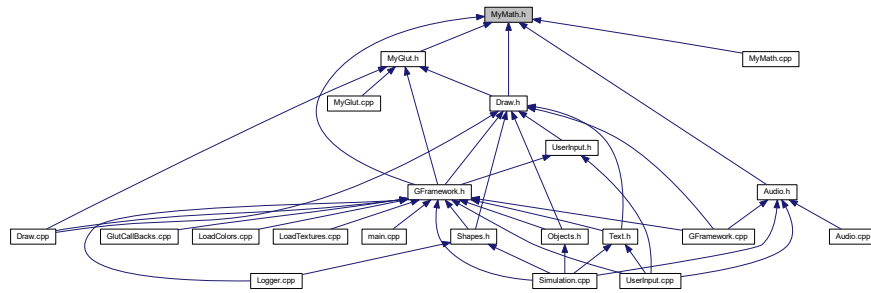
```
#include <math.h>
```

```
#include "reactphysics3d.h"
```

Include dependency graph for MyMath.h:



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



Macros

- `#define` [PI](#) 3.1415926535897932384626

Typedefs

- `typedef reactphysics3d::Matrix3x3` [Matrix](#)
- `typedef reactphysics3d::Matrix2x2` [Matrix2](#)
- `typedef reactphysics3d::Vector3` [Vec](#)
- `typedef reactphysics3d::Vector2` [Vec2](#)

10.19.1 Macro Definition Documentation

10.19.1.1 PI

```
#define PI 3.1415926535897932384626
```

10.19.2 Typedef Documentation

10.19.2.1 Matrix

```
typedef reactphysics3d::Matrix3x3 Matrix
```

10.19.2.2 Matrix2

```
typedef reactphysics3d::Matrix2x2 Matrix2
```

10.19.2.3 Vec

```
typedef reactphysics3d::Vector3 Vec
```

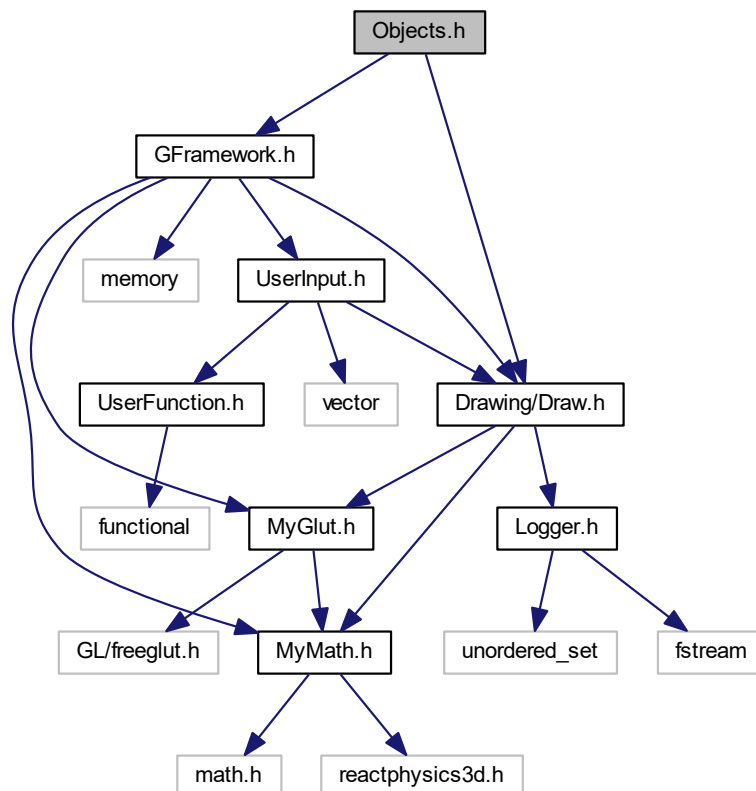
10.19.2.4 Vec2

```
typedef reactphysics3d::Vector2 Vec2
```

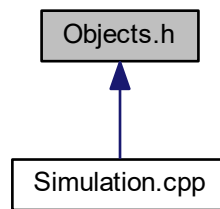
10.20 Objects.cpp File Reference

10.21 Objects.h File Reference

```
#include "Drawing/Draw.h"
#include "GFramework.h"
Include dependency graph for Objects.h:
```



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



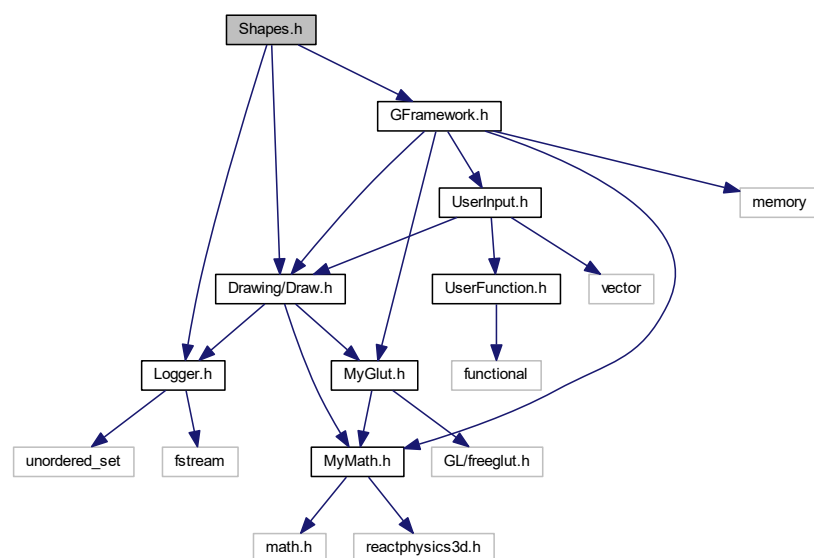
Classes

- struct [DrawMySpecificObject< A >](#)

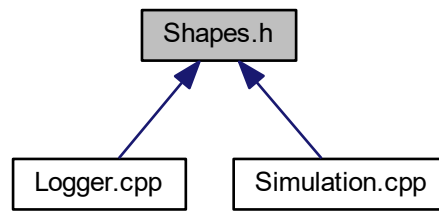
10.22 Shapes.cpp File Reference

10.23 Shapes.h File Reference

```
#include "Drawing/Draw.h"  
#include "Logger.h"  
#include "GFramework.h"  
Include dependency graph for Shapes.h:
```



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



Classes

- struct `DrawRectangle< A >`
- struct `DrawCircle< A >`
- struct `DrawPlane< A >`

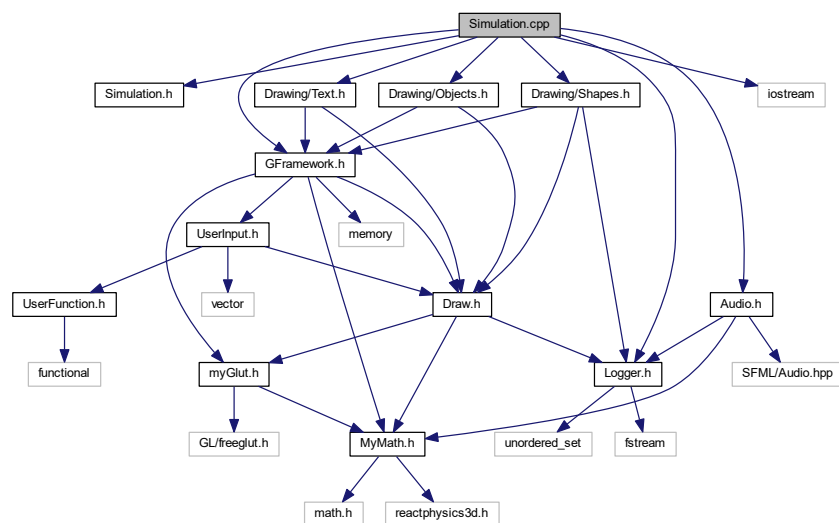
10.24 Simulation.cpp File Reference

```

#include "Simulation.h"
#include "GFramework.h"
#include "Drawing/Shapes.h"
#include "Drawing/Text.h"
#include "Drawing/Objects.h"
#include "Logger.h"
#include "Audio.h"
#include <iostream>

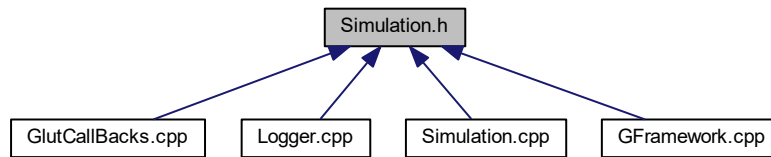
```

Include dependency graph for `Simulation.cpp`:



10.25 Simulation.h File Reference

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



Classes

- class [Simulation](#)

Enumerations

- enum [GameMode](#) : unsigned int { [GameMode::MAIN_MENU](#), [GameMode::SIMULATION](#) }
- enum [InputType](#) : unsigned int { [InputType::DEFAULT](#), [InputType::BLOCKING_MESSAGE](#), [InputType::FATAL_MESSAGE](#), [InputType::NUMERIC_INPUT](#), [InputType::ALPHA_NUMERIC_INPUT](#) }

10.25.1 Enumeration Type Documentation

10.25.1.1 GameMode

```
enum GameMode : unsigned int [strong]
```

Enumerator

MAIN_MENU	
SIMULATION	

10.25.1.2 InputType

```
enum InputType : unsigned int [strong]
```

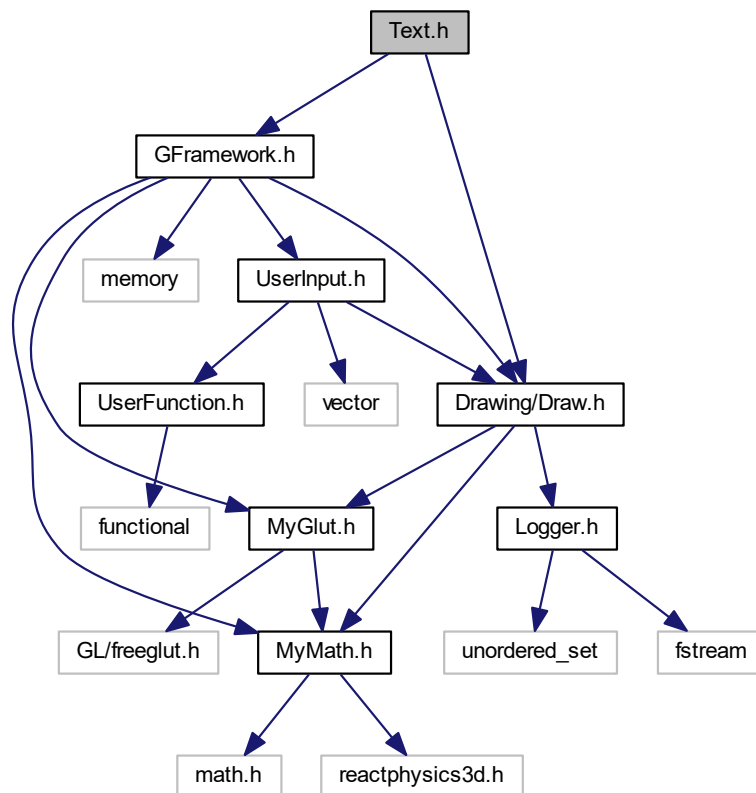
Enumerator

DEFAULT	
BLOCKING_MESSAGE	
FATAL_MESSAGE	
NUMERIC_INPUT	
ALPHA_NUMERIC_INPUT	

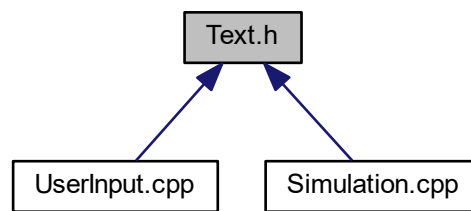
10.26 Text.cpp File Reference

10.27 Text.h File Reference

```
#include "Drawing/Draw.h"
#include "GFramework.h"
Include dependency graph for Text.h:
```



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



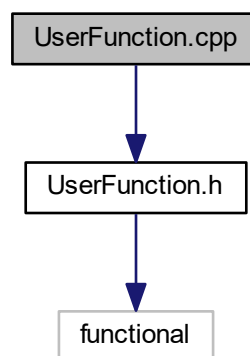
Classes

- struct `DrawString< A >`

10.28 UserFunction.cpp File Reference

```
#include "UserFunction.h"
```

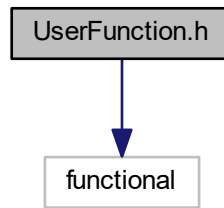
Include dependency graph for `UserFunction.cpp`:



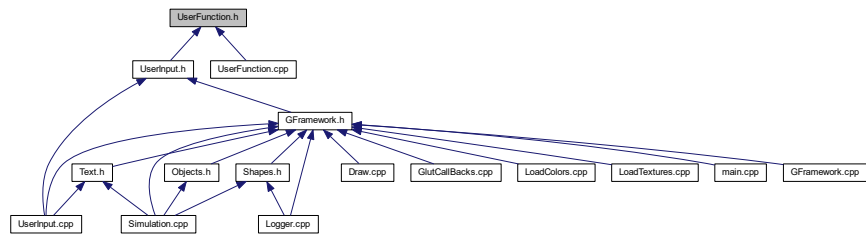
10.29 UserFunction.h File Reference

```
#include <functional>
```

Include dependency graph for UserFunction.h:



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



Classes

- struct [UserFunction](#)

Macros

- `#define` [ESC](#) (char) 27
- `#define` [TAB](#) (char) 9
- `#define` [BACKSPACE](#) (char) 8
- `#define` [ENTER](#) (char) 13

Typedefs

- `typedef std::function< void()>` [Action](#)

10.29.1 Macro Definition Documentation

10.29.1.1 BACKSPACE

```
#define BACKSPACE (char) 8
```

10.29.1.2 ENTER

```
#define ENTER (char) 13
```

10.29.1.3 ESC

```
#define ESC (char) 27
```

10.29.1.4 TAB

```
#define TAB (char) 9
```

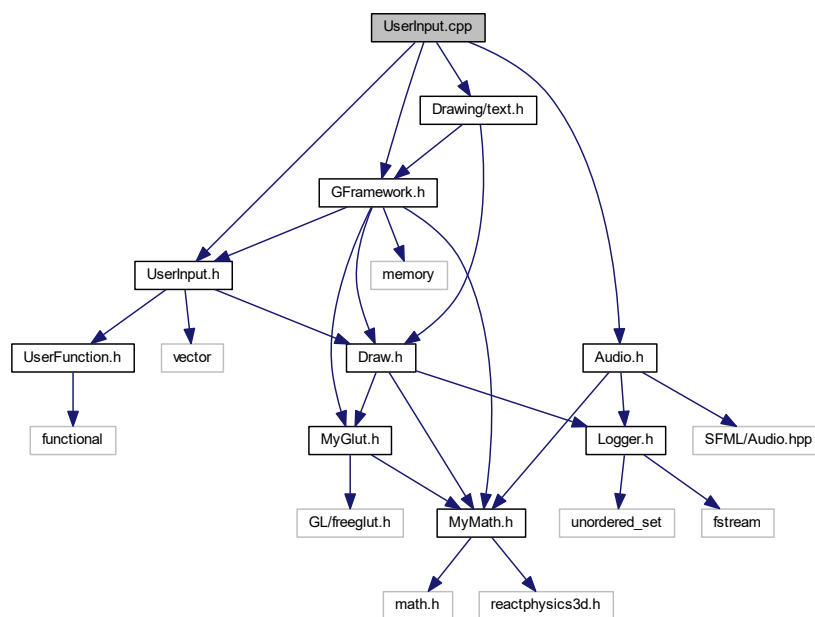
10.29.2 Typedef Documentation

10.29.2.1 Action

```
typedef std::function<void()> Action
```

10.30 UserInput.cpp File Reference

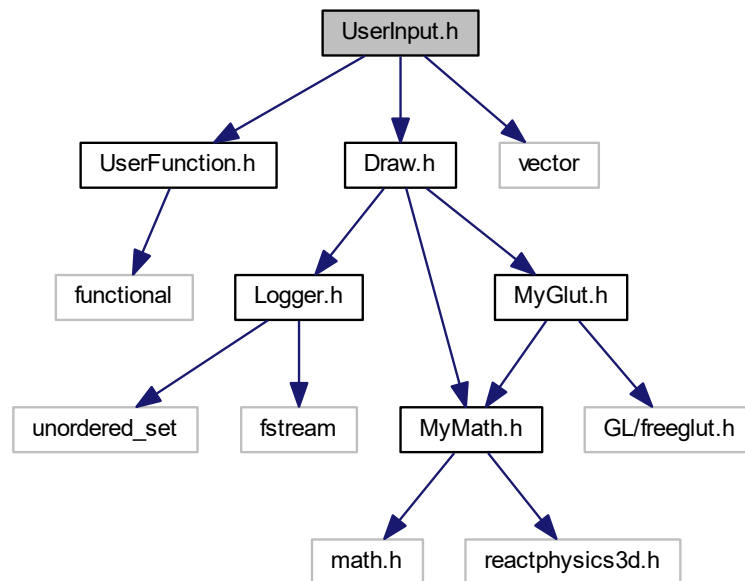
```
#include "UserInput.h"
#include "GFramework.h"
#include "Drawing/text.h"
#include "Audio.h"
Include dependency graph for UserInput.cpp:
```



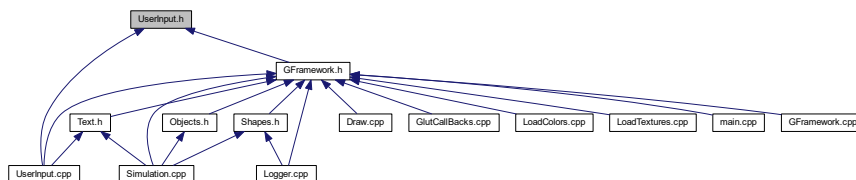
10.31 UserInput.h File Reference

```
#include "UserFunction.h"
#include "Draw.h"
#include <vector>
```


Include dependency graph for UserInput.h:



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



Classes

- class [UserInput](#)

10.32 utility.cpp File Reference

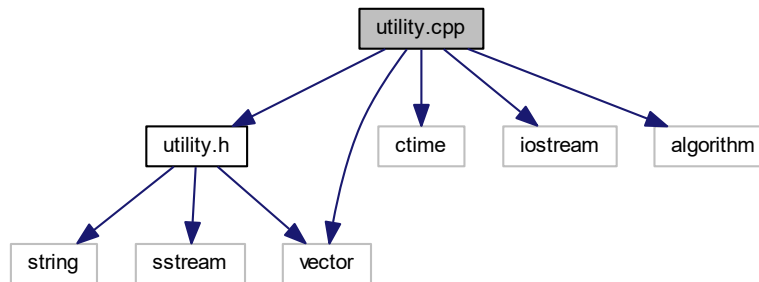
```

#include "utility.h"
#include <ctime>
#include <vector>
#include <iostream>

```

```
#include <algorithm>
```

Include dependency graph for utility.cpp:



Namespaces

- [utility](#)

This namespace contains functions which you might expect to be standard in cpp.

Functions

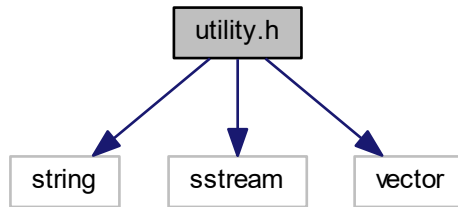
- `std::string utility::toUpper (std::string str)`
Converts a std::string to upper case.
- `std::string utility::toLower (std::string str)`
Converts a std::string to lower case.
- `std::string utility::replaceString (std::string subject, const std::string &search, const std::string &replace)`
Returns a string with the desired text replaced.
- `std::string utility::replaceCharSet (std::string subject, const std::string &charSet, const std::string &replace)`
Replaces a set of characters with a string.
- `std::vector< std::string > utility::split (std::string stringToBeSplit, std::string delimiter)`
Splits a string based on a supplied delimiter.
- `std::string utility::getCurrentDate ()`
Returns the current date.
- `std::string utility::getCurrentTime ()`
Returns the current time.

10.33 utility.h File Reference

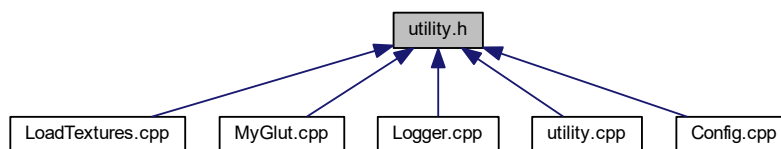
```
#include <string>
#include <sstream>
```

```
#include <vector>
```

Include dependency graph for utility.h:



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



Namespaces

- [utility](#)

This namespace contains functions which you might expect to be standard in cpp.

Functions

- `std::string utility::toUpper (std::string str)`
Converts a std::string to upper case.
- `std::string utility::toLower (std::string str)`
Converts a std::string to lower case.
- `std::string utility::replaceString (std::string subject, const std::string &search, const std::string &replace)`
Returns a string with the desired text replaced.
- `std::string utility::replaceCharSet (std::string subject, const std::string &charSet, const std::string &replace)`
Replaces a set of characters with a string.
- `template<typename T >`
`std::string utility::numToStr (T Number)`
Returns a number as its string representation.
- `std::vector< std::string > utility::split (std::string stringToBeSplit, std::string delimiter)`
Splits a string based on a supplied delimiter.
- `std::string utility::getCurrentTime ()`
Returns the current time.
- `std::string utility::getCurrentDate ()`
Returns the current date.

10.34 version.h File Reference

Namespaces

- [AutoVersion](#)

Macros

- `#define RC_FILEVERSION 0,0,0,0`
- `#define RC_FILEVERSION_STRING "0, 0, 0, 0\0"`

Variables

- static const char [AutoVersion::DATE](#) [] = "24"
- static const char [AutoVersion::MONTH](#) [] = "01"
- static const char [AutoVersion::YEAR](#) [] = "2018"
- static const char [AutoVersion::UBUNTU_VERSION_STYLE](#) [] = "18.01"
- static const char [AutoVersion::STATUS](#) [] = "Alpha"
- static const char [AutoVersion::STATUS_SHORT](#) [] = "a"
- static const long [AutoVersion::MAJOR](#) = 0
- static const long [AutoVersion::MINOR](#) = 0
- static const long [AutoVersion::BUILD](#) = 0
- static const long [AutoVersion::REVISION](#) = 0
- static const long [AutoVersion::BUILDS_COUNT](#) = 1
- static const char [AutoVersion::FULLVERSION_STRING](#) [] = "0.0.0.0"
- static const long [AutoVersion::BUILD_HISTORY](#) = 0

10.34.1 Macro Definition Documentation

10.34.1.1 RC_FILEVERSION

```
#define RC_FILEVERSION 0,0,0,0
```

10.34.1.2 RC_FILEVERSION_STRING

```
#define RC_FILEVERSION_STRING "0, 0, 0, 0\0"
```

Index

- ~Audio
 - Audio, [32](#)
- ~Config
 - Config, [38](#)
- ~GFramework
 - GFramework, [63](#)
- ~Logger
 - Logger, [72](#)
- ~Simulation
 - Simulation, [80](#)
- ~UserFunction
 - UserFunction, [85](#)
- ~UserInput
 - UserInput, [86](#)
- AUDIO_DIRECTORY
 - Audio, [33](#)
- Action
 - UserFunction.h, [135](#)
- action
 - UserFunction, [85](#)
- ang
 - Camera, [35](#)
- Appearance
 - Draw.h, [95](#)
- Audio, [31](#)
 - ~Audio, [32](#)
 - AUDIO_DIRECTORY, [33](#)
 - Audio, [32](#)
 - COULD_NOT_LOAD_MUSIC_MESSAGE, [33](#)
 - COULD_NOT_LOAD_SOUND_MESSAGE, [33](#)
 - clearStoppedSounds, [32](#)
 - music, [33](#)
 - playSound, [32](#), [33](#)
 - soundBuffers, [34](#)
 - sounds, [34](#)
- audio
 - GFramework, [67](#)
- Audio.cpp, [89](#)
- Audio.h, [89](#)
- AutoVersion, [15](#)
 - BUILD_HISTORY, [15](#)
 - BUILDS_COUNT, [15](#)
 - BUILD, [15](#)
 - DATE, [16](#)
 - FULLVERSION_STRING, [16](#)
 - MAJOR, [16](#)
 - MINOR, [16](#)
 - MONTH, [16](#)
 - REVISION, [16](#)
 - STATUS_SHORT, [16](#)
 - STATUS, [16](#)
 - UBUNTU_VERSION_STYLE, [17](#)
 - YEAR, [17](#)
- BACKSPACE
 - UserFunction.h, [134](#)
- BUILD_HISTORY
 - AutoVersion, [15](#)
- BUILDS_COUNT
 - AutoVersion, [15](#)
- BUILD
 - AutoVersion, [15](#)
- CONFIG_FILE
 - Config, [40](#)
- COULD_NOT_LOAD_MUSIC_MESSAGE
 - Audio, [33](#)
- COULD_NOT_LOAD_SOUND_MESSAGE
 - Audio, [33](#)
- callMouse
 - glutCB, [21](#)
- Camera, [34](#)
 - ang, [35](#)
 - Camera, [35](#)
 - DEFAULT_HEIGHT, [35](#)
 - DEFAULT_R_SPEED, [35](#)
 - DEFAULT_T_SPEED, [36](#)
 - del, [36](#)
 - dir, [36](#)
 - mov, [36](#)
 - pos, [36](#)
 - rotationSpeed, [36](#)
 - translationSpeed, [37](#)
- camera
 - GFramework, [67](#)
- changeColor
 - Drawing, [44](#)
- changeSize
 - glutCB, [22](#)
- changeTexture
 - Drawing, [45](#)
- clearStoppedSounds
 - Audio, [32](#)
- clicked
 - Mouse, [79](#)
- colorMap
 - GFramework, [67](#)
- Config, [37](#)
 - ~Config, [38](#)

- CONFIG_FILE, 40
- Config, 38
- getFileValue, 38
- getValue, 39
- Config.cpp, 90
- Config.h, 91
- ConfigValueConverters, 17
 - getValue_, 18
 - getValue_< bool >, 18
 - getValue_< double >, 19
 - getValue_< int >, 19
 - getValue_< std::string >, 20
- DATE
 - AutoVersion, 16
- DEFAULT_DEGREE
 - Logger, 76
- DEFAULT_HEIGHT
 - Camera, 35
- DEFAULT_R_SPEED
 - Camera, 35
- DEFAULT_T_SPEED
 - Camera, 36
- DEFAULT_TYPE
 - Logger, 76
- del
 - Camera, 36
- Dimension
 - Drawing, 44
- dimension
 - DrawCircle, 42
 - DrawMySpecificObject, 53
 - DrawPlane, 56
 - DrawRectangle, 59
 - DrawString, 61
- dir
 - Camera, 36
- display
 - GFramework, 67
- draw
 - DrawCircle, 42
 - DrawMySpecificObject, 52
 - DrawPlane, 56
 - DrawRectangle, 58
 - DrawString, 61
- Draw.cpp, 93
- Draw.h, 93
 - Appearance, 95
- DrawCircle
 - dimension, 42
 - draw, 42
 - DrawCircle, 41
- DrawCircle< A >, 40
- DrawItem
 - DrawItem, 50
 - INVALID_APPEARANCE_MESSAGE, 50
- DrawItem< A >, 49
- DrawMySpecificObject
 - dimension, 53
 - draw, 52
 - DrawMySpecificObject, 52
- DrawMySpecificObject< A >, 51
- DrawPlane
 - dimension, 56
 - draw, 56
 - DrawPlane, 54, 55
- DrawPlane< A >, 53
- DrawRectangle
 - dimension, 59
 - draw, 58
 - DrawRectangle, 58
- DrawRectangle< A >, 57
- DrawString
 - dimension, 61
 - draw, 61
 - DrawString, 60
- DrawString< A >, 59
- drawUserString
 - UserInput, 86
- Drawing, 43
 - changeColor, 44
 - changeTexture, 45
 - Dimension, 44
 - enable2D, 46
 - enable3D, 46
 - enableND, 46
 - INVALID_COLOR_MESSAGE, 48
 - INVALID_DRAWING_ORDER_MESSAGE, 48
 - INVALID_TEXTURE_MESSAGE, 48
 - isColor, 47
 - isTexture, 47
 - UNKNOWN_APPROVED_COLOR_MESSAGE, 48
 - UNKNOWN_APPROVED_TEXTURE_MESSAGE, 48
 - UNKNOWN_DIMENSION_MESSAGE, 48
- drawingState
 - GFramework, 68
- ENTER
 - UserFunction.h, 135
- ESC
 - UserFunction.h, 135
- enable2D
 - Drawing, 46
- enable3D
 - Drawing, 46
- enableND
 - Drawing, 46
- FAILED_TO_OPEN_AFTER_FAILURE_MESSAGE
 - Logger, 76
- FPS
 - GFramework, 68
- FULLVERSION_STRING
 - AutoVersion, 16
- functions
 - UserInput, 88

- GFramework, 62
 - ~GFramework, 63
 - audio, 67
 - camera, 67
 - colorMap, 67
 - display, 67
 - drawingState, 68
 - FPS, 68
 - GFramework, 64
 - get, 68
 - INIT_WINDOW_HEIGHT, 68
 - INIT_WINDOW_WIDTH, 68
 - INIT_WINDOW_X, 68
 - INIT_WINDOW_Y, 69
 - initializeGlut, 65
 - loadColors, 65
 - loadTextures, 65
 - mouse, 69
 - operator=, 66
 - RENDERING_DISTANCE, 69
 - showScene, 66
 - simulation, 69
 - startup, 66
 - textureMap, 69
 - userInput, 69
 - WINDOW_TITLE, 70
 - windowSize, 70
- GFramework.cpp, 98
- GFramework.h, 99
- GameMode
 - Simulation.h, 131
- gameMode
 - Simulation, 83
- get
 - GFramework, 68
 - Logger, 72
- getCurrentDate
 - utility, 27
- getCurrentTime
 - utility, 27
- getFileValue
 - Config, 38
- getTimeStamp
 - Logger, 72
- getValue
 - Config, 39
- getValue_
 - ConfigValueConverters, 18
- getValue_ < bool >
 - ConfigValueConverters, 18
- getValue_ < double >
 - ConfigValueConverters, 19
- getValue_ < int >
 - ConfigValueConverters, 19
- getValue_ < std::string >
 - ConfigValueConverters, 20
- glLoadTexture
 - MyGlut.cpp, 110
- MyGlut.h, 119
- glTexVert2f
 - MyGlut.cpp, 110–113
 - MyGlut.h, 119–121
- glTexVert3f
 - MyGlut.cpp, 113–116
 - MyGlut.h, 122–124
- GlutCallBacks.cpp, 100
- GlutCallBacks.h, 102
- glutCB, 20
 - callMouse, 21
 - changeSize, 22
 - keyPressed, 22
 - keyUp, 23
 - mouseMove, 23
 - passiveMouse, 24
 - pressSpecialKey, 24
 - releaseSpecialKey, 25
 - renderScene, 25
 - update, 26
- heldDown
 - Mouse, 79
- INIT_WINDOW_HEIGHT
 - GFramework, 68
- INIT_WINDOW_WIDTH
 - GFramework, 68
- INIT_WINDOW_X
 - GFramework, 68
- INIT_WINDOW_Y
 - GFramework, 69
- INITIAL_GAME_MODE
 - Simulation, 83
- INITIAL_INPUT_TYPE
 - Simulation, 83
- INVALID_APPEARANCE_MESSAGE
 - DrawItem, 50
- INVALID_COLOR_MESSAGE
 - Drawing, 48
- INVALID_DRAWING_ORDER_MESSAGE
 - Drawing, 48
- INVALID_TEXTURE_MESSAGE
 - Drawing, 48
- init
 - Simulation, 80
- initializeGlut
 - GFramework, 65
- inputString
 - UserInput, 88
- InputType
 - Simulation.h, 131
- inputType
 - Simulation, 83
- isColor
 - Drawing, 47
- isInputStringSubmitted
 - UserInput, 87
- isTexture

- Drawing, 47
- key
 - UserFunction, 85
- keyInputsHeld
 - UserInput, 88
- keyPressed
 - glutCB, 22
- keyStates
 - UserInput, 88
- keyUp
 - glutCB, 23
- LOG_FILE_TITLE
 - Logger, 77
- LOG
 - Logger.h, 106
- loadColors
 - GFramework, 65
- LoadColors.cpp, 103
- loadGameModeKeyboard
 - Simulation, 81
- loadTextures
 - GFramework, 65
- LoadTextures.cpp, 103
- log
 - Logger, 73
- LogDegree
 - Logger.h, 107
- logFile
 - Logger, 77
- logMessage
 - Logger, 74
- LogType
 - Logger.h, 107
- Logger, 70
 - ~Logger, 72
 - DEFAULT_DEGREE, 76
 - DEFAULT_TYPE, 76
 - FAILED_TO_OPEN_AFTER_FAILURE_MESSAGE, 76
 - GE, 76
 - get, 72
 - getTimeStamp, 72
 - LOG_FILE_TITLE, 77
 - log, 73
 - logFile, 77
 - logMessage, 74
 - Logger, 72
 - logs, 77
 - normalExit, 75
 - OPENED_LOG_AFTER_FAILURE_MESSAGE, 77
 - operator=, 75
 - PROGRAM_EXIT_MESSAGE, 77
 - toString, 76
 - UNKNOWN_LOG_DEGREE_MESSAGE, 77
- Logger.cpp, 104
- Logger.h, 105
 - LOG, 106
 - LogDegree, 107
 - LogType, 107
 - NORMAL_EXIT, 106
- logs
 - Logger, 77
- MAJOR
 - AutoVersion, 16
- MINOR
 - AutoVersion, 16
- MONTH
 - AutoVersion, 16
- main
 - main.cpp, 109
- main.cpp, 108
 - main, 109
- Matrix
 - MyMath.h, 127
- Matrix2
 - MyMath.h, 127
- Mouse, 78
 - clicked, 79
 - heldDown, 79
 - Mouse, 78
 - x, 79
 - y, 79
- mouse
 - GFramework, 69
- mouseMove
 - glutCB, 23
- mov
 - Camera, 36
- music
 - Audio, 33
- MyGlut.cpp, 109
 - glLoadTexture, 110
 - glTexVert2f, 110–113
 - glTexVert3f, 113–116
 - saveScreenShot, 116
- MyGlut.h, 117
 - glLoadTexture, 119
 - glTexVert2f, 119–121
 - glTexVert3f, 122–124
 - saveScreenShot, 125
 - Tex, 119
- MyMath.cpp, 126
- MyMath.h, 126
 - Matrix, 127
 - Matrix2, 127
 - PI, 127
 - Vec, 128
 - Vec2, 128
- NORMAL_EXIT
 - Logger.h, 106
- NUM_KEYS
 - UserInput, 88
- normalExit
 - Logger, 75

- numToStr
 - utility, [27](#)
- OPENED_LOG_AFTER_FAILURE_MESSAGE
 - Logger, [77](#)
- Objects.cpp, [128](#)
- Objects.h, [128](#)
- operator=
 - GFramework, [66](#)
 - Logger, [75](#)
- PROGRAM_EXIT_MESSAGE
 - Logger, [77](#)
- passiveMouse
 - glutCB, [24](#)
- PI
 - MyMath.h, [127](#)
- playSound
 - Audio, [32, 33](#)
- pos
 - Camera, [36](#)
- pressSpecialKey
 - glutCB, [24](#)
- RC_FILEVERSION_STRING
 - version.h, [140](#)
- RC_FILEVERSION
 - version.h, [140](#)
- RENDERING_DISTANCE
 - GFramework, [69](#)
- REVISION
 - AutoVersion, [16](#)
- release
 - UserFunction, [85](#)
- releaseSpecialKey
 - glutCB, [25](#)
- renderScene
 - glutCB, [25](#)
- replaceCharSet
 - utility, [28](#)
- replaceString
 - utility, [29](#)
- rotationSpeed
 - Camera, [36](#)
- run
 - Simulation, [81](#)
- STATUS_SHORT
 - AutoVersion, [16](#)
- STATUS
 - AutoVersion, [16](#)
- saveScreenShot
 - MyGlut.cpp, [116](#)
 - MyGlut.h, [125](#)
- setGameMode
 - Simulation, [81](#)
- setInputType
 - Simulation, [82](#)
- setInputTypeKeyboard
 - Simulation, [82](#)
 - setToDefault
 - UserInput, [87](#)
- Shapes.cpp, [129](#)
- Shapes.h, [129](#)
- showScene
 - GFramework, [66](#)
- Simulation, [79](#)
 - ~Simulation, [80](#)
 - gameMode, [83](#)
 - INITIAL_GAME_MODE, [83](#)
 - INITIAL_INPUT_TYPE, [83](#)
 - init, [80](#)
 - inputType, [83](#)
 - loadGameModeKeyboard, [81](#)
 - run, [81](#)
 - setGameMode, [81](#)
 - setInputType, [82](#)
 - setInputTypeKeyboard, [82](#)
 - Simulation, [80](#)
 - UNKNOWN_GAME_MODE_MESSAGE, [83](#)
 - UNKNOWN_INPUT_TYPE_MESSAGE, [83](#)
- simulation
 - GFramework, [69](#)
- Simulation.cpp, [130](#)
- Simulation.h, [131](#)
 - GameMode, [131](#)
 - InputType, [131](#)
- soundBuffers
 - Audio, [34](#)
- sounds
 - Audio, [34](#)
- specialKey
 - UserFunction, [85](#)
- split
 - utility, [29](#)
- startup
 - GFramework, [66](#)
- submitInputString
 - UserInput, [87](#)
- TAB
 - UserFunction.h, [135](#)
- TERMINATING_CHAR
 - UserInput, [88](#)
- Tex
 - MyGlut.h, [119](#)
- Text.cpp, [132](#)
- Text.h, [132](#)
- textureMap
 - GFramework, [69](#)
- toLower
 - utility, [30](#)
- toString
 - Logger, [76](#)
- toUpper
 - utility, [30](#)
- translationSpeed
 - Camera, [37](#)

- UBUNTU_VERSION_STYLE
 - AutoVersion, [17](#)
- UNKNOWN_APPROVED_COLOR_MESSAGE
 - Drawing, [48](#)
- UNKNOWN_APPROVED_TEXTURE_MESSAGE
 - Drawing, [48](#)
- UNKNOWN_DIMENSION_MESSAGE
 - Drawing, [48](#)
- UNKNOWN_GAME_MODE_MESSAGE
 - Simulation, [83](#)
- UNKNOWN_INPUT_TYPE_MESSAGE
 - Simulation, [83](#)
- UNKNOWN_LOG_DEGREE_MESSAGE
 - Logger, [77](#)
- update
 - glutCB, [26](#)
- UserFunction, [84](#)
 - ~UserFunction, [85](#)
 - action, [85](#)
 - key, [85](#)
 - release, [85](#)
 - specialKey, [85](#)
 - UserFunction, [84](#)
- UserFunction.cpp, [133](#)
- UserFunction.h, [133](#)
 - Action, [135](#)
 - BACKSPACE, [134](#)
 - ENTER, [135](#)
 - ESC, [135](#)
 - TAB, [135](#)
- UserInput, [86](#)
 - ~UserInput, [86](#)
 - drawUserString, [86](#)
 - functions, [88](#)
 - inputString, [88](#)
 - isInputStringSubmitted, [87](#)
 - keyInputsHeld, [88](#)
 - keyStates, [88](#)
 - NUM_KEYS, [88](#)
 - setToDefault, [87](#)
 - submitInputString, [87](#)
 - TERMINATING_CHAR, [88](#)
 - UserInput, [86](#)
- userInput
 - GFramework, [69](#)
- UserInput.cpp, [136](#)
- UserInput.h, [136](#)
- utility, [26](#)
 - getCurrentDate, [27](#)
 - getCurrentTime, [27](#)
 - numToStr, [27](#)
 - replaceCharSet, [28](#)
 - replaceString, [29](#)
 - split, [29](#)
 - toLower, [30](#)
 - toUpper, [30](#)
- utility.cpp, [137](#)
- utility.h, [138](#)
- Vec
 - MyMath.h, [128](#)
- Vec2
 - MyMath.h, [128](#)
- version.h, [140](#)
 - RC_FILEVERSION_STRING, [140](#)
 - RC_FILEVERSION, [140](#)
- WINDOW_TITLE
 - GFramework, [70](#)
- windowSize
 - GFramework, [70](#)
- x
 - Mouse, [79](#)
- y
 - Mouse, [79](#)
- YEAR
 - AutoVersion, [17](#)