

Graphing Calculator

0

Generated by Doxygen 1.10.0

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 Application Class Reference	5
3.1.1 Detailed Description	6
3.1.2 Constructor & Destructor Documentation	6
3.1.2.1 Application()	6
3.1.2.2 ~Application()	6
3.1.3 Member Function Documentation	6
3.1.3.1 initializeVariables()	6
3.1.3.2 initializeWindow()	7
3.1.3.3 pollEvents()	7
3.1.3.4 render()	8
3.1.3.5 update()	8
3.1.3.6 windowIsOpen()	8
3.1.4 Member Data Documentation	8
3.1.4.1 event	8
3.1.4.2 graph	8
3.1.4.3 videoMode	9
3.1.4.4 window	9
3.2 Graph Class Reference	9
3.2.1 Detailed Description	10
3.2.2 Constructor & Destructor Documentation	10
3.2.2.1 Graph()	10
3.2.3 Member Function Documentation	11
3.2.3.1 decXRange()	11
3.2.3.2 decYRange()	11
3.2.3.3 drawToWindow()	11
3.2.3.4 graphToScreen()	12
3.2.3.5 gridToScreen()	12
3.2.3.6 incXRange()	12
3.2.3.7 incYRange()	12
3.2.3.8 screenToGraph()	13
3.2.3.9 setGraphMode()	13
3.2.3.10 updateGraph()	13
3.2.4 Member Data Documentation	14
3.2.4.1 background	14
3.2.4.2 graphMode	14
3.2.4.3 gridCols	14

3.2.4.4 gridRows	14
3.2.4.5 gridTile	14
3.2.4.6 gridVector	14
3.2.4.7 hMax	15
3.2.4.8 hMin	15
3.2.4.9 wMax	15
3.2.4.10 wMin	15
3.2.4.11 xMax	15
3.2.4.12 xMin	15
3.2.4.13 yMax	15
3.2.4.14 yMin	15
4 File Documentation	17
4.1 /home/jakemath/Desktop/code/SFML/GraphingApp/src/application.cpp File Reference	17
4.2 application.cpp	17
4.3 /home/jakemath/Desktop/code/SFML/GraphingApp/src/application.h File Reference	18
4.3.1 Variable Documentation	19
4.3.1.1 DEFAULT_H_MAX	19
4.3.1.2 DEFAULT_H_MIN	19
4.3.1.3 DEFAULT_W_MAX	19
4.3.1.4 DEFAULT_W_MIN	19
4.3.1.5 DEFAULT_X_MAX	19
4.3.1.6 DEFAULT_X_MIN	20
4.3.1.7 DEFAULT_Y_MAX	20
4.3.1.8 DEFAULT_Y_MIN	20
4.3.1.9 WINDOW_HEIGHT	20
4.3.1.10 WINDOW_WIDTH	20
4.4 application.h	20
4.5 /home/jakemath/Desktop/code/SFML/GraphingApp/src/graph.cpp File Reference	21
4.6 graph.cpp	21
4.7 /home/jakemath/Desktop/code/SFML/GraphingApp/src/graph.h File Reference	23
4.8 graph.h	23
4.9 /home/jakemath/Desktop/code/SFML/GraphingApp/src/header.h File Reference	24
4.10 header.h	24
4.11 /home/jakemath/Desktop/code/SFML/GraphingApp/src/main.cpp File Reference	24
4.11.1 Function Documentation	24
4.11.1.1 main()	24
4.12 main.cpp	25
4.13 /home/jakemath/Desktop/code/SFML/GraphingApp/src/test_funcs.cpp File Reference	25
4.13.1 Function Documentation	25
4.13.1.1 funcA()	25
4.13.1.2 funcB()	25

4.13.1.3 funcC()	26
4.13.1.4 funcD()	26
4.13.1.5 funcE()	26
4.13.1.6 funcF()	26
4.13.1.7 funcG()	26
4.14 test_funcs.cpp	27
4.15 /home/jakemath/Desktop/code/SFML/GraphingApp/src/test_funcs.h File Reference	27
4.15.1 Function Documentation	27
4.15.1.1 funcA()	27
4.15.1.2 funcB()	28
4.15.1.3 funcC()	28
4.15.1.4 funcD()	28
4.15.1.5 funcE()	28
4.15.1.6 funcF()	28
4.15.1.7 funcG()	28
4.16 test_funcs.h	29
Index	31

Chapter 1

Class Index

1.1 Class List

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

Application	Manages the application window	5
Graph	Manages the graph area on screen	9

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

/home/jakemath/Desktop/code/SFML/GraphingApp/src/ application.cpp	17
/home/jakemath/Desktop/code/SFML/GraphingApp/src/ application.h	18
/home/jakemath/Desktop/code/SFML/GraphingApp/src/ graph.cpp	21
/home/jakemath/Desktop/code/SFML/GraphingApp/src/ graph.h	23
/home/jakemath/Desktop/code/SFML/GraphingApp/src/ header.h	24
/home/jakemath/Desktop/code/SFML/GraphingApp/src/ main.cpp	24
/home/jakemath/Desktop/code/SFML/GraphingApp/src/ test_funcs.cpp	25
/home/jakemath/Desktop/code/SFML/GraphingApp/src/ test_funcs.h	27

Chapter 3

Class Documentation

3.1 Application Class Reference

Manages the application window.

```
#include <application.h>
```

Public Member Functions

- [Application](#) ()
Default Constructor.
- virtual [~Application](#) ()
Destructor.
- const bool [windowIsOpen](#) () const
Get whether window is open.
- void [pollEvents](#) ()
Poll for events.
- void [update](#) ()
Update application objects.
- void [render](#) ()
Draw renderable objects to the window and display.

Private Member Functions

- void [initializeVariables](#) ()
Initialzie member variables.
- void [initializeWindow](#) ()
Initialize window member variable.

Private Attributes

- sf::RenderWindow * [window](#)
- sf::VideoMode [videoMode](#)
- sf::Event [event](#)
- [Graph](#) * [graph](#)

3.1.1 Detailed Description

Manages the application window.

Definition at line 22 of file [application.h](#).

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Application()

```
Application::Application ( )
```

Default Constructor.

Definition at line 21 of file [application.cpp](#).

```
00021     {
00022         this->initializeVariables();
00023         this->initializeWindow();
00024     }
```

3.1.2.2 ~Application()

```
Application::~~Application ( ) [virtual]
```

Destructor.

Definition at line 26 of file [application.cpp](#).

```
00026     {
00027         delete this->window;
00028         delete this->graph;
00029     }
```

3.1.3 Member Function Documentation

3.1.3.1 initializeVariables()

```
void Application::initializeVariables ( ) [private]
```

Initialzie member variables.

Definition at line 3 of file [application.cpp](#).

```
00003     {
00004         this->window = nullptr;
00005         sf::Vector2f xRange(DEFAULT_X_MIN, DEFAULT_X_MAX);
00006         sf::Vector2f yRange(DEFAULT_Y_MIN, DEFAULT_Y_MAX);
00007         sf::Vector2f wRange(DEFAULT_W_MIN, DEFAULT_W_MAX);
00008         sf::Vector2f hRange(DEFAULT_H_MIN, DEFAULT_H_MAX);
00009         this->graph = new Graph(xRange, yRange, wRange, hRange);
00010     }
```

3.1.3.2 initializeWindow()

```
void Application::initializeWindow ( ) [private]
```

Initialize window member variable.

Definition at line 12 of file [application.cpp](#).

```
00012     {
00013         this->videoMode.width = WINDOW_WIDTH;
00014         this->videoMode.height = WINDOW_HEIGHT;
00015         this->window = new sf::RenderWindow(this->videoMode, "Application", sf::Style::None);
00016                                     // sf::Style::Titlebar | sf::Style::Close);
00017         this->window->setPosition(sf::Vector2i(0, 0));
00018         this->window->setFramerateLimit(60);
00019     }
```

3.1.3.3 pollEvents()

```
void Application::pollEvents ( )
```

Poll for events.

Definition at line 35 of file [application.cpp](#).

```
00035     {
00036         while (this->window->pollEvent(this->event)) {
00037             switch (this->event.type) {
00038                 case sf::Event::Closed:
00039                     this->window->close();
00040                     break;
00041                 case sf::Event::KeyPressed:
00042                     switch (this->event.key.code) {
00043                         case sf::Keyboard::Escape:
00044                             this->window->close();
00045                             break;
00046                         case sf::Keyboard::A:
00047                             graph->setGraphMode(1);
00048                             break;
00049                         case sf::Keyboard::B:
00050                             graph->setGraphMode(2);
00051                             break;
00052                         case sf::Keyboard::C:
00053                             graph->setGraphMode(3);
00054                             break;
00055                         case sf::Keyboard::D:
00056                             graph->setGraphMode(4);
00057                             break;
00058                         case sf::Keyboard::E:
00059                             graph->setGraphMode(5);
00060                             break;
00061                         case sf::Keyboard::F:
00062                             graph->setGraphMode(6);
00063                             break;
00064                         case sf::Keyboard::G:
00065                             graph->setGraphMode(7);
00066                             break;
00067                         case sf::Keyboard::Right:
00068                             graph->incXRange(0.1);
00069                             break;
00070                         case sf::Keyboard::Left:
00071                             graph->decXRange(0.1);
00072                             break;
00073                         case sf::Keyboard::Up:
00074                             graph->incYRange(0.1);
00075                             break;
00076                         case sf::Keyboard::Down:
00077                             graph->decYRange(0.1);
00078                             break;
00079                     }
00080                     break;
00081             }
00082         }
00083     }
```

3.1.3.4 render()

```
void Application::render ( )
```

Draw renderable objects to the window and display.

Definition at line 90 of file [application.cpp](#).

```
00091 {  
00092     window->clear();  
00093     graph->drawToWindow(window);  
00094     window->display();  
00095 }
```

3.1.3.5 update()

```
void Application::update ( )
```

Update application objects.

Definition at line 85 of file [application.cpp](#).

```
00085 {  
00086     pollEvents();  
00087     graph->updateGraph();  
00088 }
```

3.1.3.6 windowIsOpen()

```
const bool Application::windowIsOpen ( ) const
```

Get whether window is open.

Definition at line 31 of file [application.cpp](#).

```
00031 {  
00032     return this->window->isOpen();  
00033 }
```

3.1.4 Member Data Documentation

3.1.4.1 event

```
sf::Event Application::event [private]
```

Definition at line 27 of file [application.h](#).

3.1.4.2 graph

```
Graph* Application::graph [private]
```

Definition at line 28 of file [application.h](#).

3.1.4.3 videoMode

```
sf::VideoMode Application::videoMode [private]
```

Definition at line 26 of file [application.h](#).

3.1.4.4 window

```
sf::RenderWindow* Application::window [private]
```

Definition at line 25 of file [application.h](#).

3.2 Graph Class Reference

Manages the graph area on screen.

```
#include <graph.h>
```

Public Member Functions

- [Graph](#) (sf::Vector2f xRange, sf::Vector2f yRange, sf::Vector2f wRange, sf::Vector2f hRange)
Default constructor.
- void [setGraphMode](#) (int i)
Set graphMode.
- void [updateGraph](#) ()
Update gridVector based on graphMode.
- void [drawToWindow](#) (sf::RenderWindow *window)
Draw gridVector to a window.
- void [decXRange](#) (float xDec)
Decrement xMin and xMax by xDec.
- void [decYRange](#) (float yDec)
Decrement yMin and yMax by yDec.
- void [incXRange](#) (float xInc)
Increment xMin and xMax by xInc.
- void [incYRange](#) (float yInc)
Increment yMin and yMax by yInc.

Private Member Functions

- sf::Vector2f [gridToScreen](#) (sf::Vector2u grid_pos)
Convert grid (i,j) coordinates to screen (w,h) coordinates.
- sf::Vector2f [screenToGraph](#) (sf::Vector2f screen_pos)
Convert screen (w,h) coordinates to graph (x,y) coordinates.
- sf::Vector2f [graphToScreen](#) (sf::Vector2f graph_pos)
Convert graph (x,y) coordinates to screen (w,h) coordinates.

Private Attributes

- float [xMin](#)
Minimum x value (graph)
- float [xMax](#)
Maximum x value (graph)
- float [yMin](#)
Minimum y value (graph)
- float [yMax](#)
Maximum y value (graph)
- float [wMin](#)
Minimum x value (screen)
- float [wMax](#)
Maximum x value (screen)
- float [hMin](#)
Minimum y value (screen)
- float [hMax](#)
Maximum y value (screen)
- int [gridRows](#)
Rows in rectangle grid.
- int [gridCols](#)
Columns in rectangle grid.
- sf::RectangleShape [gridTile](#)
Rectangle prefab.
- std::vector< sf::RectangleShape > [gridVector](#)
Collection of rectangles to be drawn.
- sf::RectangleShape [background](#)
Background rectangle.
- int [graphMode](#)
Temp variable for declaring graph type.

3.2.1 Detailed Description

Manages the graph area on screen.

Definition at line 9 of file [graph.h](#).

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Graph()

```
Graph::Graph (
    sf::Vector2f xRange,
    sf::Vector2f yRange,
    sf::Vector2f wRange,
    sf::Vector2f hRange )
```

Default constructor.

Definition at line 27 of file [graph.cpp](#).


```

00028 {
00029     xMin = xRange.x;
00030     xMax = xRange.y;
00031     yMin = yRange.x;
00032     yMax = yRange.y;
00033     wMin = wRange.x;
00034     wMax = wRange.y;
00035     hMin = hRange.x;
00036     hMax = hRange.y;
00037     gridRows = 1000;
00038     gridCols = 1000;
00039     graphMode = 1;
00040     gridTile.setSize(sf::Vector2f((wMax-wMin)/gridCols, (hMax-hMin)/gridRows));
00041     background.setSize(sf::Vector2f(wMax-wMin, hMax-hMin));
00042     background.setPosition(sf::Vector2f(wMin, hMin));
00043     background.setFillColor(sf::Color::White);
00044 }

```

3.2.3 Member Function Documentation

3.2.3.1 decXRange()

```

void Graph::decXRange (
    float xDec )

```

Decrement xMin and xMax by xDec.

Definition at line 108 of file [graph.cpp](#).

```

00109 {
00110     xMin -= xDec;
00111     xMax -= xDec;
00112     return;
00113 }

```

3.2.3.2 decYRange()

```

void Graph::decYRange (
    float yDec )

```

Decrement yMin and yMax by yDec.

Definition at line 115 of file [graph.cpp](#).

```

00116 {
00117     yMin -= yDec;
00118     yMax -= yDec;
00119     return;
00120 }

```

3.2.3.3 drawToWindow()

```

void Graph::drawToWindow (
    sf::RenderWindow * window )

```

Draw gridVector to a window.

Definition at line 98 of file [graph.cpp](#).

```

00099 {
00100     window->draw(background);
00101     for (auto r : gridVector)
00102     {
00103         window->draw(r);
00104     }
00105     return;
00106 }

```

3.2.3.4 graphToScreen()

```
sf::Vector2f Graph::graphToScreen (
    sf::Vector2f graph_pos ) [private]
```

Convert graph (x,y) coordinates to screen (w,h) coordinates.

Definition at line 19 of file [graph.cpp](#).

```
00020 {
00021     sf::Vector2f screen_pos;
00022     screen_pos.x = wMin + (graph_pos.x-xMin) * (wMax-wMin) / (xMax-xMin);
00023     screen_pos.y = hMin + (yMax-graph_pos.y) * (hMax-hMin) / (yMax-yMin);
00024     return screen_pos;
00025 }
```

3.2.3.5 gridToScreen()

```
sf::Vector2f Graph::gridToScreen (
    sf::Vector2u grid_pos ) [private]
```

Convert grid (i,j) coordinates to screen (w,h) coordinates.

Definition at line 3 of file [graph.cpp](#).

```
00004 {
00005     sf::Vector2f screen_pos;
00006     screen_pos.x = wMin + grid_pos.x * (wMax-wMin) / gridCols;
00007     screen_pos.y = hMin + grid_pos.y * (hMax-hMin) / gridRows;
00008     return screen_pos;
00009 }
```

3.2.3.6 incXRange()

```
void Graph::incXRange (
    float xInc )
```

Increment xMin and xMax by xInc.

Definition at line 122 of file [graph.cpp](#).

```
00123 {
00124     xMin += xInc;
00125     xMax += xInc;
00126     return;
00127 }
```

3.2.3.7 incYRange()

```
void Graph::incYRange (
    float yInc )
```

Increment yMin and yMax by yInc.

Definition at line 129 of file [graph.cpp](#).

```
00130 {
00131     yMin += yInc;
00132     yMax += yInc;
00133     return;
00134 }
```

3.2.3.8 screenToGraph()

```
sf::Vector2f Graph::screenToGraph (
    sf::Vector2f screen_pos ) [private]
```

Convert screen (w,h) coordinates to graph (x,y) coordinates.

Definition at line 11 of file [graph.cpp](#).

```
00012 {
00013     sf::Vector2f graph_pos;
00014     graph_pos.x = xMin + (screen_pos.x-wMin) * (xMax-xMin) / (wMax-wMin);
00015     graph_pos.y = yMax - (screen_pos.y-hMin) * (yMax-yMin) / (hMax-hMin);
00016     return graph_pos;
00017 }
```

3.2.3.9 setGraphMode()

```
void Graph::setGraphMode (
    int i )
```

Set graphMode.

Definition at line 46 of file [graph.cpp](#).

```
00047 {
00048     graphMode = i;
00049     return;
00050 }
```

3.2.3.10 updateGraph()

```
void Graph::updateGraph ( )
```

Update gridVector based on graphMode.

Definition at line 52 of file [graph.cpp](#).

```
00053 {
00054     gridVector.clear();
00055     for (int i=0; i<gridRows; i++)
00056     {
00057         for (int j=0; j<gridCols; j++)
00058         {
00059             sf::Vector2f screenC = gridToScreen(sf::Vector2u(i, j));
00060             sf::Vector2f graphC = screenToGraph(screenC);
00061             float x = graphC.x;
00062             float y = graphC.y;
00063             float fx = 0;
00064             switch (graphMode)
00065             {
00066                 case 1:
00067                     fx = funcA(x);
00068                     break;
00069                 case 2:
00070                     fx = funcB(x);
00071                     break;
00072                 case 3:
00073                     fx = funcC(x);
00074                     break;
00075                 case 4:
00076                     fx = funcD(x);
00077                     break;
00078                 case 5:
00079                     fx = funcE(x);
00080                     break;
00081                 case 6:
00082                     fx = funcF(x);
00083                     break;
00084                 case 7:
00085                     fx = funcG(x);
00086                     break;
00087             }
00088             if (std::abs(y-fx) < 0.01)
00089             {
00090                 gridTile.setPosition(screenC);
00091                 gridTile.setFillColor(sf::Color::Red);
00092                 gridVector.push_back(gridTile);
00093             }
00094         }
00095     }
00096 }
```

3.2.4 Member Data Documentation

3.2.4.1 background

```
sf::RectangleShape Graph::background [private]
```

Background rectangle.

Definition at line 38 of file [graph.h](#).

3.2.4.2 graphMode

```
int Graph::graphMode [private]
```

Temp variable for declaring graph type.

Definition at line 40 of file [graph.h](#).

3.2.4.3 gridCols

```
int Graph::gridCols [private]
```

Columns in rectangle grid.

Definition at line 32 of file [graph.h](#).

3.2.4.4 gridRows

```
int Graph::gridRows [private]
```

Rows in rectangle grid.

Definition at line 30 of file [graph.h](#).

3.2.4.5 gridTile

```
sf::RectangleShape Graph::gridTile [private]
```

Rectangle prefab.

Definition at line 34 of file [graph.h](#).

3.2.4.6 gridVector

```
std::vector<sf::RectangleShape> Graph::gridVector [private]
```

Collection of rectangles to be drawn.

Definition at line 36 of file [graph.h](#).

3.2.4.7 hMax

```
float Graph::hMax [private]
```

Maximum y value (screen)

Definition at line 28 of file [graph.h](#).

3.2.4.8 hMin

```
float Graph::hMin [private]
```

Minimum y value (screen)

Definition at line 26 of file [graph.h](#).

3.2.4.9 wMax

```
float Graph::wMax [private]
```

Maximum x value (screen)

Definition at line 24 of file [graph.h](#).

3.2.4.10 wMin

```
float Graph::wMin [private]
```

Minimum x value (screen)

Definition at line 22 of file [graph.h](#).

3.2.4.11 xMax

```
float Graph::xMax [private]
```

Maximum x value (graph)

Definition at line 16 of file [graph.h](#).

3.2.4.12 xMin

```
float Graph::xMin [private]
```

Minimum x value (graph)

Definition at line 14 of file [graph.h](#).

3.2.4.13 yMax

```
float Graph::yMax [private]
```

Maximum y value (graph)

Definition at line 20 of file [graph.h](#).

3.2.4.14 yMin

```
float Graph::yMin [private]
```

Minimum y value (graph)

Definition at line 18 of file [graph.h](#).

Chapter 4

File Documentation

4.1 /home/jakemath/Desktop/code/SFML/Graphing↵ App/src/application.cpp File Reference

```
#include "application.h"
```

4.2 application.cpp

[Go to the documentation of this file.](#)

```
00001 #include "application.h"
00002
00003 void Application::initializeVariables() {
00004     this->window = nullptr;
00005     sf::Vector2f xRange(DEFAULT_X_MIN, DEFAULT_X_MAX);
00006     sf::Vector2f yRange(DEFAULT_Y_MIN, DEFAULT_Y_MAX);
00007     sf::Vector2f wRange(DEFAULT_W_MIN, DEFAULT_W_MAX);
00008     sf::Vector2f hRange(DEFAULT_H_MIN, DEFAULT_H_MAX);
00009     this->graph = new Graph(xRange, yRange, wRange, hRange);
00010 }
00011
00012 void Application::initializeWindow() {
00013     this->videoMode.width = WINDOW_WIDTH;
00014     this->videoMode.height = WINDOW_HEIGHT;
00015     this->window = new sf::RenderWindow(this->videoMode, "Application", sf::Style::None);
00016                                     // sf::Style::Titlebar | sf::Style::Close);
00017     this->window->setPosition(sf::Vector2i(0, 0));
00018     this->window->setFramerateLimit(60);
00019 }
00020
00021 Application::Application() {
00022     this->initializeVariables();
00023     this->initializeWindow();
00024 }
00025
00026 Application::~Application() {
00027     delete this->window;
00028     delete this->graph;
00029 }
00030
00031 const bool Application::windowIsOpen() const {
00032     return this->window->isOpen();
00033 }
00034
00035 void Application::pollEvents() {
00036     while (this->window->pollEvent(this->event)) {
00037         switch (this->event.type) {
00038             case sf::Event::Closed:
00039                 this->window->close();
00040                 break;
00041             case sf::Event::KeyPressed:
```

```

00042         switch (this->event.key.code) {
00043             case sf::Keyboard::Escape:
00044                 this->window->close();
00045                 break;
00046             case sf::Keyboard::A:
00047                 graph->setGraphMode(1);
00048                 break;
00049             case sf::Keyboard::B:
00050                 graph->setGraphMode(2);
00051                 break;
00052             case sf::Keyboard::C:
00053                 graph->setGraphMode(3);
00054                 break;
00055             case sf::Keyboard::D:
00056                 graph->setGraphMode(4);
00057                 break;
00058             case sf::Keyboard::E:
00059                 graph->setGraphMode(5);
00060                 break;
00061             case sf::Keyboard::F:
00062                 graph->setGraphMode(6);
00063                 break;
00064             case sf::Keyboard::G:
00065                 graph->setGraphMode(7);
00066                 break;
00067             case sf::Keyboard::Right:
00068                 graph->incXRange(0.1);
00069                 break;
00070             case sf::Keyboard::Left:
00071                 graph->decXRange(0.1);
00072                 break;
00073             case sf::Keyboard::Up:
00074                 graph->incYRange(0.1);
00075                 break;
00076             case sf::Keyboard::Down:
00077                 graph->decYRange(0.1);
00078                 break;
00079         }
00080         break;
00081     }
00082 }
00083 }
00084
00085 void Application::update() {
00086     pollEvents();
00087     graph->updateGraph();
00088 }
00089
00090 void Application::render()
00091 {
00092     window->clear();
00093     graph->drawToWindow(window);
00094     window->display();
00095 }
00096
00097

```

4.3 /home/jakemath/Desktop/code/SFML/GraphingApp/src/application.h

File Reference

```

#include "header.h"
#include "graph.h"

```

Classes

- class [Application](#)
Manages the application window.

Variables

- const float [WINDOW_WIDTH](#) = 1920
- const float [WINDOW_HEIGHT](#) = 1080
- const float [DEFAULT_X_MIN](#) = -2
- const float [DEFAULT_X_MAX](#) = 2
- const float [DEFAULT_Y_MIN](#) = -2
- const float [DEFAULT_Y_MAX](#) = 2
- const float [DEFAULT_W_MIN](#) = 560
- const float [DEFAULT_W_MAX](#) = 1360
- const float [DEFAULT_H_MIN](#) = 140
- const float [DEFAULT_H_MAX](#) = 940

4.3.1 Variable Documentation

4.3.1.1 [DEFAULT_H_MAX](#)

```
const float DEFAULT_H_MAX = 940
```

Definition at line 17 of file [application.h](#).

4.3.1.2 [DEFAULT_H_MIN](#)

```
const float DEFAULT_H_MIN = 140
```

Definition at line 16 of file [application.h](#).

4.3.1.3 [DEFAULT_W_MAX](#)

```
const float DEFAULT_W_MAX = 1360
```

Definition at line 15 of file [application.h](#).

4.3.1.4 [DEFAULT_W_MIN](#)

```
const float DEFAULT_W_MIN = 560
```

Definition at line 14 of file [application.h](#).

4.3.1.5 [DEFAULT_X_MAX](#)

```
const float DEFAULT_X_MAX = 2
```

Definition at line 10 of file [application.h](#).

4.3.1.6 DEFAULT_X_MIN

```
const float DEFAULT_X_MIN = -2
```

Definition at line 9 of file [application.h](#).

4.3.1.7 DEFAULT_Y_MAX

```
const float DEFAULT_Y_MAX = 2
```

Definition at line 12 of file [application.h](#).

4.3.1.8 DEFAULT_Y_MIN

```
const float DEFAULT_Y_MIN = -2
```

Definition at line 11 of file [application.h](#).

4.3.1.9 WINDOW_HEIGHT

```
const float WINDOW_HEIGHT = 1080
```

Definition at line 7 of file [application.h](#).

4.3.1.10 WINDOW_WIDTH

```
const float WINDOW_WIDTH = 1920
```

Definition at line 6 of file [application.h](#).

4.4 application.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "header.h"
00004 #include "graph.h"
00005
00006 const float WINDOW_WIDTH = 1920;
00007 const float WINDOW_HEIGHT = 1080;
00008
00009 const float DEFAULT_X_MIN = -2;
00010 const float DEFAULT_X_MAX = 2;
00011 const float DEFAULT_Y_MIN = -2;
00012 const float DEFAULT_Y_MAX = 2;
00013
00014 const float DEFAULT_W_MIN = 560;
00015 const float DEFAULT_W_MAX = 1360;
00016 const float DEFAULT_H_MIN = 140;
00017 const float DEFAULT_H_MAX = 940;
00018
00022 class Application {
00023 private:
00024     // Variables
00025     sf::RenderWindow* window;
00026     sf::VideoMode videoMode;
```

```

00027     sf::Event event;
00028     Graph* graph;
00029
00030     // Private Member Functions
00032     void initializeVariables();
00034     void initializeWindow();
00035 public:
00036     // Constructors / Destructors
00038     Application();
00040     virtual ~Application();
00041     // Accessors
00043     const bool windowIsOpen() const;
00044     // Functions
00046     void pollEvents();
00048     void update();
00050     void render();
00051 };

```

4.5 /home/jakemath/Desktop/code/SFML/GraphingApp/src/graph.cpp File Reference

```
#include "graph.h"
```

4.6 graph.cpp

[Go to the documentation of this file.](#)

```

00001 #include "graph.h"
00002
00003 sf::Vector2f Graph::gridToScreen(sf::Vector2u grid_pos)
00004 {
00005     sf::Vector2f screen_pos;
00006     screen_pos.x = wMin + grid_pos.x * (wMax-wMin) / gridCols;
00007     screen_pos.y = hMin + grid_pos.y * (hMax-hMin) / gridRows;
00008     return screen_pos;
00009 }
00010
00011 sf::Vector2f Graph::screenToGraph(sf::Vector2f screen_pos)
00012 {
00013     sf::Vector2f graph_pos;
00014     graph_pos.x = xMin + (screen_pos.x-wMin) * (xMax-xMin) / (wMax-wMin);
00015     graph_pos.y = yMax - (screen_pos.y-hMin) * (yMax-yMin) / (hMax-hMin);
00016     return graph_pos;
00017 }
00018
00019 sf::Vector2f Graph::graphToScreen(sf::Vector2f graph_pos)
00020 {
00021     sf::Vector2f screen_pos;
00022     screen_pos.x = wMin + (graph_pos.x-xMin) * (wMax-wMin) / (xMax-xMin);
00023     screen_pos.y = hMin + (yMax-graph_pos.y) * (hMax-hMin) / (yMax-yMin);
00024     return screen_pos;
00025 }
00026
00027 Graph::Graph(sf::Vector2f xRange, sf::Vector2f yRange, sf::Vector2f wRange, sf::Vector2f hRange)
00028 {
00029     xMin = xRange.x;
00030     xMax = xRange.y;
00031     yMin = yRange.x;
00032     yMax = yRange.y;
00033     wMin = wRange.x;
00034     wMax = wRange.y;
00035     hMin = hRange.x;
00036     hMax = hRange.y;
00037     gridRows = 1000;
00038     gridCols = 1000;
00039     graphMode = 1;
00040     gridTile.setSize(sf::Vector2f((wMax-wMin)/gridCols, (hMax-hMin)/gridRows));
00041     background.setSize(sf::Vector2f(wMax-wMin, hMax-hMin));
00042     background.setPosition(sf::Vector2f(wMin, hMin));
00043     background.setFillColor(sf::Color::White);
00044 }
00045
00046 void Graph::setGraphMode(int i)

```

```

00047 {
00048     graphMode = i;
00049     return;
00050 }
00051
00052 void Graph::updateGraph()
00053 {
00054     gridVector.clear();
00055     for (int i=0; i<gridRows; i++)
00056     {
00057         for (int j=0; j<gridCols; j++)
00058         {
00059             sf::Vector2f screenC = gridToScreen(sf::Vector2u(i, j));
00060             sf::Vector2f graphC = screenToGraph(screenC);
00061             float x = graphC.x;
00062             float y = graphC.y;
00063             float fx = 0;
00064             switch (graphMode)
00065             {
00066                 case 1:
00067                     fx = funcA(x);
00068                     break;
00069                 case 2:
00070                     fx = funcB(x);
00071                     break;
00072                 case 3:
00073                     fx = funcC(x);
00074                     break;
00075                 case 4:
00076                     fx = funcD(x);
00077                     break;
00078                 case 5:
00079                     fx = funcE(x);
00080                     break;
00081                 case 6:
00082                     fx = funcF(x);
00083                     break;
00084                 case 7:
00085                     fx = funcG(x);
00086                     break;
00087             }
00088             if (std::abs(y-fx) < 0.01)
00089             {
00090                 gridTile.setPosition(screenC);
00091                 gridTile.setFillColor(sf::Color::Red);
00092                 gridVector.push_back(gridTile);
00093             }
00094         }
00095     }
00096 }
00097
00098 void Graph::drawToWindow(sf::RenderWindow* window)
00099 {
00100     window->draw(background);
00101     for (auto r : gridVector)
00102     {
00103         window->draw(r);
00104     }
00105     return;
00106 }
00107
00108 void Graph::decXRange(float xDec)
00109 {
00110     xMin -= xDec;
00111     xMax -= xDec;
00112     return;
00113 }
00114
00115 void Graph::decYRange(float yDec)
00116 {
00117     yMin -= yDec;
00118     yMax -= yDec;
00119     return;
00120 }
00121
00122 void Graph::incXRange(float xInc)
00123 {
00124     xMin += xInc;
00125     xMax += xInc;
00126     return;
00127 }
00128
00129 void Graph::incYRange(float yInc)
00130 {
00131     yMin += yInc;
00132     yMax += yInc;
00133     return;

```

```
00134 }
```

4.7 /home/jakemath/Desktop/code/SFML/GraphingApp/src/graph.h File Reference

```
#include "header.h"
#include "test_funcs.h"
```

Classes

- class [Graph](#)

Manages the graph area on screen.

4.8 graph.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "header.h"
00004 #include "test_funcs.h"
00005
00009 class Graph
00010 {
00011 private:
00012     // Variables
00014     float xMin;
00016     float xMax;
00018     float yMin;
00020     float yMax;
00022     float wMin;
00024     float wMax;
00026     float hMin;
00028     float hMax;
00030     int gridRows;
00032     int gridCols;
00034     sf::RectangleShape gridTile;
00036     std::vector<sf::RectangleShape> gridVector;
00038     sf::RectangleShape background;
00040     int graphMode;
00041
00042     // Private Member Functions
00044     sf::Vector2f gridToScreen(sf::Vector2u grid_pos);
00046     sf::Vector2f screenToGraph(sf::Vector2f screen_pos);
00048     sf::Vector2f graphToScreen(sf::Vector2f graph_pos);
00049 public:
00051     Graph(sf::Vector2f xRange, sf::Vector2f yRange, sf::Vector2f wRange, sf::Vector2f hRange);
00053     void setGraphMode(int i);
00055     void updateGraph();
00057     void drawToWindow(sf::RenderWindow* window);
00059     void decXRange(float xDec);
00061     void decYRange(float yDec);
00063     void incXRange(float xInc);
00065     void incYRange(float yInc);
00066 };
```

4.9 /home/jakemath/Desktop/code/SFML/GraphingApp/src/header.h File Reference

```
#include <iostream>
#include <vector>
#include <ctime>
#include <SFML/Graphics.hpp>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Audio.hpp>
#include <SFML/Network.hpp>
```

4.10 header.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00006
00007 #include <iostream>
00008 #include <vector>
00009 #include <ctime>
00010
00011 #include <SFML/Graphics.hpp>
00012 #include <SFML/System.hpp>
00013 #include <SFML/Window.hpp>
00014 #include <SFML/Audio.hpp>
00015 #include <SFML/Network.hpp>
```

4.11 /home/jakemath/Desktop/code/SFML/GraphingApp/src/main.cpp File Reference

```
#include "application.h"
```

Functions

- `int main ()`
Program entry point.

4.11.1 Function Documentation

4.11.1.1 main()

```
int main ( )
```

Program entry point.

Seed random. Create an instance of [Application](#). Update app and render to the screen while the window is open.

Definition at line 9 of file [main.cpp](#).

```
00010 {
00011     std::srand(std::time(nullptr));
00012     Application app;
00013     while (app.windowIsOpen()) {
00014         app.update();
00015         app.render();
00016     }
00017     return 0;
00018 }
```

4.12 main.cpp

[Go to the documentation of this file.](#)

```
00001 #include "application.h"
00002
00008
00009 int main()
00010 {
00011     std::srand(std::time(nullptr));
00012     Application app;
00013     while (app.windowIsOpen()) {
00014         app.update();
00015         app.render();
00016     }
00017     return 0;
00018 }
```

4.13 /home/jakemath/Desktop/code/SFML/GraphingApp/src/test_↵ funcs.cpp File Reference

```
#include "test_funcs.h"
```

Functions

- float [funcA](#) (float x)
- float [funcB](#) (float x)
- float [funcC](#) (float x)
- float [funcD](#) (float x)
- float [funcE](#) (float x)
- float [funcF](#) (float x)
- float [funcG](#) (float x)

4.13.1 Function Documentation

4.13.1.1 funcA()

```
float funcA (
    float x )
```

Definition at line 3 of file [test_funcs.cpp](#).

```
00003     {
00004         return x;
00005     }
```

4.13.1.2 funcB()

```
float funcB (
    float x )
```

Definition at line 7 of file [test_funcs.cpp](#).

```
00007     {
00008         return x * x;
00009     }
```

4.13.1.3 funcC()

```
float funcC (  
    float x )
```

Definition at line 11 of file [test_funcs.cpp](#).

```
00011     {  
00012     return x * x * x;  
00013 }
```

4.13.1.4 funcD()

```
float funcD (  
    float x )
```

Definition at line 15 of file [test_funcs.cpp](#).

```
00015     {  
00016     return std::abs(x);  
00017 }
```

4.13.1.5 funcE()

```
float funcE (  
    float x )
```

Definition at line 19 of file [test_funcs.cpp](#).

```
00019     {  
00020     return std::sin(x);  
00021 }
```

4.13.1.6 funcF()

```
float funcF (  
    float x )
```

Definition at line 23 of file [test_funcs.cpp](#).

```
00023     {  
00024     return std::cos(x);  
00025 }
```

4.13.1.7 funcG()

```
float funcG (  
    float x )
```

Definition at line 27 of file [test_funcs.cpp](#).

```
00027     {  
00028     return std::tan(x);  
00029 }
```


4.14 test_funcs.cpp

[Go to the documentation of this file.](#)

```
00001 #include "test_funcs.h"
00002
00003 float funcA(float x) {
00004     return x;
00005 }
00006
00007 float funcB(float x) {
00008     return x * x;
00009 }
00010
00011 float funcC(float x) {
00012     return x * x * x;
00013 }
00014
00015 float funcD(float x) {
00016     return std::abs(x);
00017 }
00018
00019 float funcE(float x) {
00020     return std::sin(x);
00021 }
00022
00023 float funcF(float x) {
00024     return std::cos(x);
00025 }
00026
00027 float funcG(float x) {
00028     return std::tan(x);
00029 }
```

4.15 /home/jakemath/Desktop/code/SFML/GraphingApp/src/test_funcs.h File Reference

```
#include "math.h"
```

Functions

- float [funcA](#) (float)
- float [funcB](#) (float)
- float [funcC](#) (float)
- float [funcD](#) (float)
- float [funcE](#) (float)
- float [funcF](#) (float)
- float [funcG](#) (float)

4.15.1 Function Documentation

4.15.1.1 funcA()

```
float funcA (
    float x )
```

Definition at line 3 of file [test_funcs.cpp](#).

```
00003     {
00004         return x;
00005     }
```

4.15.1.2 funcB()

```
float funcB (  
    float x )
```

Definition at line 7 of file [test_funcs.cpp](#).

```
00007     {  
00008     return x * x;  
00009 }
```

4.15.1.3 funcC()

```
float funcC (  
    float x )
```

Definition at line 11 of file [test_funcs.cpp](#).

```
00011     {  
00012     return x * x * x;  
00013 }
```

4.15.1.4 funcD()

```
float funcD (  
    float x )
```

Definition at line 15 of file [test_funcs.cpp](#).

```
00015     {  
00016     return std::abs(x);  
00017 }
```

4.15.1.5 funcE()

```
float funcE (  
    float x )
```

Definition at line 19 of file [test_funcs.cpp](#).

```
00019     {  
00020     return std::sin(x);  
00021 }
```

4.15.1.6 funcF()

```
float funcF (  
    float x )
```

Definition at line 23 of file [test_funcs.cpp](#).

```
00023     {  
00024     return std::cos(x);  
00025 }
```

4.15.1.7 funcG()

```
float funcG (  
    float x )
```

Definition at line 27 of file [test_funcs.cpp](#).

```
00027     {  
00028     return std::tan(x);  
00029 }
```

4.16 test_funcs.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "math.h"
00004
00005 float funcA(float);
00006 float funcB(float);
00007 float funcC(float);
00008 float funcD(float);
00009 float funcE(float);
00010 float funcF(float);
00011 float funcG(float);
```


Index

/home/jakemath/Desktop/code/SFML/GraphingApp/src/application.cpp, 11	Graph, 11
17	DEFAULT_H_MAX
/home/jakemath/Desktop/code/SFML/GraphingApp/src/application.h, 19	application.h, 19
18, 20	DEFAULT_H_MIN
/home/jakemath/Desktop/code/SFML/GraphingApp/src/graph.cpp, 19	graph.cpp, 19
21	DEFAULT_W_MAX
/home/jakemath/Desktop/code/SFML/GraphingApp/src/graph.h, 19	graph.h, 19
23	DEFAULT_W_MIN
/home/jakemath/Desktop/code/SFML/GraphingApp/src/header.h, 19	header.h, 19
24	DEFAULT_X_MAX
/home/jakemath/Desktop/code/SFML/GraphingApp/src/main.cpp, 19	main.cpp, 19
24, 25	DEFAULT_X_MIN
/home/jakemath/Desktop/code/SFML/GraphingApp/src/test_funcs.cpp, 19	test_funcs.cpp, 19
25, 27	DEFAULT_Y_MAX
/home/jakemath/Desktop/code/SFML/GraphingApp/src/test_funcs.h, 20	test_funcs.h, 20
27, 29	DEFAULT_Y_MIN
~Application	application.h, 20
Application, 6	drawToWindow
	Graph, 11
Application, 5	
~Application, 6	event
Application, 6	Application, 8
event, 8	
graph, 8	funcA
initializeVariables, 6	test_funcs.cpp, 25
initializeWindow, 6	test_funcs.h, 27
pollEvents, 7	funcB
render, 7	test_funcs.cpp, 25
update, 8	test_funcs.h, 27
videoMode, 8	funcC
window, 9	test_funcs.cpp, 25
windowsIsOpen, 8	test_funcs.h, 28
application.h	funcD
DEFAULT_H_MAX, 19	test_funcs.cpp, 26
DEFAULT_H_MIN, 19	test_funcs.h, 28
DEFAULT_W_MAX, 19	funcE
DEFAULT_W_MIN, 19	test_funcs.cpp, 26
DEFAULT_X_MAX, 19	test_funcs.h, 28
DEFAULT_X_MIN, 19	funcF
DEFAULT_Y_MAX, 20	test_funcs.cpp, 26
DEFAULT_Y_MIN, 20	test_funcs.h, 28
WINDOW_HEIGHT, 20	funcG
WINDOW_WIDTH, 20	test_funcs.cpp, 26
	test_funcs.h, 28
background	
Graph, 14	Graph, 9
	background, 14
decXRange	decXRange, 11
Graph, 11	decYRange, 11
decYRange	drawToWindow, 11

- Graph, 10
- graphMode, 14
- graphToScreen, 11
- gridCols, 14
- gridRows, 14
- gridTile, 14
- gridToScreen, 12
- gridVector, 14
- hMax, 14
- hMin, 15
- incXRange, 12
- incYRange, 12
- screenToGraph, 12
- setGraphMode, 13
- updateGraph, 13
- wMax, 15
- wMin, 15
- xMax, 15
- xMin, 15
- yMax, 15
- yMin, 15
- graph
 - Application, 8
- graphMode
 - Graph, 14
- graphToScreen
 - Graph, 11
- gridCols
 - Graph, 14
- gridRows
 - Graph, 14
- gridTile
 - Graph, 14
- gridToScreen
 - Graph, 12
- gridVector
 - Graph, 14
- hMax
 - Graph, 14
- hMin
 - Graph, 15
- incXRange
 - Graph, 12
- incYRange
 - Graph, 12
- initializeVariables
 - Application, 6
- initializeWindow
 - Application, 6
- main
 - main.cpp, 24
- main.cpp
 - main, 24
- pollEvents
 - Application, 7
- render
 - Application, 7
- screenToGraph
 - Graph, 12
- setGraphMode
 - Graph, 13
- test_funcs.cpp
 - funcA, 25
 - funcB, 25
 - funcC, 25
 - funcD, 26
 - funcE, 26
 - funcF, 26
 - funcG, 26
- test_funcs.h
 - funcA, 27
 - funcB, 27
 - funcC, 28
 - funcD, 28
 - funcE, 28
 - funcF, 28
 - funcG, 28
- update
 - Application, 8
- updateGraph
 - Graph, 13
- videoMode
 - Application, 8
- window
 - Application, 9
- WINDOW_HEIGHT
 - application.h, 20
- WINDOW_WIDTH
 - application.h, 20
- windowIsOpen
 - Application, 8
- wMax
 - Graph, 15
- wMin
 - Graph, 15
- xMax
 - Graph, 15
- xMin
 - Graph, 15
- yMax
 - Graph, 15
- yMin
 - Graph, 15