ballEngine

Generated by Doxygen 1.8.15

Contents

Chapter 1

ballEngine

2D physics engine in early development

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

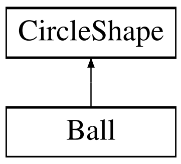
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 Ball Class Reference

Inheritance diagram for Ball:



Public Member Functions

- Ball (float radius, float mass, sf::Vector2f initPos, sf::Vector2f initVel)
- float timeToCollision (Ball &otherBall)
- void checkForBounce (int worldSizeX, int worldSizeY)
- void ballCollision (Ball &otherBall)
- void updateVelocity (float dt, Ball &otherBall)
- void applyExternalImpulse (sf::Vector2f force, float dt)
- void updatePosition (float dt)
- void sampleNextPosition ()
- void sampleCurrentPosition ()
- float getMass ()
- float getRadius ()
- sf::Vector2f getVelocity ()
- void setVelocity (sf::Vector2f vel)
- void setToCollided ()
- void resetToCollided ()
- bool getHasCollided ()
- float getKE ()
- sf::Vector2f getMomentum ()
- float getDistance (Ball &otherBall)
- float getSpeed ()
- float getRelSpeed (Ball &otherBall)
- bool getSamplePrevPosBool ()
- std::deque< sf::Vector2f > & getPreviousPositions ()

Private Member Functions

- float lenJonesForce (float x, float x_0, float r, float m)
- float **exptCollForce** (float x, float x_0, float r, float m)
- float newtonForce (float x, float x_0, float r, float G, float M)

Private Attributes

- sf::Vector2f **velocity**
- float **dampingFactor** = 1
- float **mass**
- float **radius**
- bool **collidedThisFrame** = false
- std::deque< sf::Vector2f > **previousPositions**
- bool **samplePreviousPositions** = true

5.1.1 Constructor & Destructor Documentation

5.1.1.1 Ball()

```
Ball::Ball (
             float radius,
             float mass,
             sf::Vector2f initPos,
             sf::Vector2f initVel )
```

Construct the ball.

Parameters

radius	The radius of the ball.
mass	The mass of the ball.
initPos	The initial position of the ball.
initVel	The initial velocity of the ball.

5.1.2 Member Function Documentation

5.1.2.1 applyExternalImpulse()

```
void Ball::applyExternalImpulse (
             sf::Vector2f force,
             float dt )
```

Applies an external force in the chosen direction.

Parameters

force	The directional force to apply.
dt	The simulation timestep.

5.1.2.2 ballCollision()

```
void Ball::ballCollision (
             Ball & otherBall )
```

Executes a collision with another ball.

Parameters

&otherBall	The other ball to collide with.

Returns

Void.

5.1.2.3 checkForBounce()

```
void Ball::checkForBounce (
             int worldSizeX,
             int worldSizeY )
```

Checks if the ball is about to intersect the world boundary and executes a damped collision.

Parameters

worldSizeX	The x-component size of the simulation world.
worldSizeY	The y-component size of the simulation world.

Returns

Void.

5.1.2.4 getDistance()

```
float Ball::getDistance (
             Ball & otherBall )
```

Get the distance between this ball and another ball.

Parameters

&otherBall	The other ball to calculate distance to.

Returns

The distance between this ball and the other ball.

5.1.2.5 getHasCollided()

```
bool Ball::getHasCollided ( )
```

Get the collision state of the ball.

Returns

The collision state of the ball.

5.1.2.6 getKE()

```
float Ball::getKE ( )
```

Get the kinetic energy of the ball.

Returns

The kinetic energy of the ball.

5.1.2.7 getMass()

```
float Ball::getMass ( )
```

Get the current mass of the ball.

Returns

The mass of the ball.

5.1.2.8 getMomentum()

```
sf::Vector2f Ball::getMomentum ( )
```

Get the momentum vector of the ball.

Returns

The momentum vector of the ball.

5.1.2.9 getPreviousPositions()

```
std::deque< sf::Vector2f > & Ball::getPreviousPositions ( )
```

Returns the std::deque list of previous positions of the ball.

Returns

List of previous positions of the ball.

5.1.2.10 getRadius()

```
float Ball::getRadius ( )
```

Get the current radius of the ball.

Returns

The radius of the ball.

5.1.2.11 getRelSpeed()

```
float Ball::getRelSpeed (
            Ball & otherBall )
```

Get the relative speed between this ball and another ball.

Parameters

&otherBall	The other ball to calculate relative speed to.

Returns

Relative speed between this ball and the other ball.

5.1.2.12 getSamplePrevPosBool()

```
bool Ball::getSamplePrevPosBool ( )
```

Returns the state of whether the ball samples previous positions.

Returns

The state of whether the ball samples previous positions.

5.1.2.13 getSpeed()

```
float Ball::getSpeed ( )
```

Get the current speed of the ball.

Returns

The current speed of the ball.

5.1.2.14 getVelocity()

```
sf::Vector2f Ball::getVelocity ( )
```

Get the current velocity of the ball.

Returns

The velocity vector of the ball.

5.1.2.15 lenJonesForce()

```
float Ball::lenJonesForce (
             float x,
             float x_0,
             float r,
             float m )  [private]
```

Returns change in velocity of a Lennard-Jones based force. This function must be called twice; once for each orthogonal co-ordinate x and y.

Parameters

x	The current position of the particle/ball.
x↩ *_0*	The current position of the other particle/ball to interact with.
r	The distance between the current and other particle/ball.
m	The mass of the particle/ball.

Returns

Infinitesimal change in velocity.

5.1.2.16 newtonForce()

```
float Ball::newtonForce (
             float x,
             float x_0,
             float r,
             float G,
             float M )  [private]
```

Returns change in velocity of a gravitational force. This function must be called twice; once for each orthogonal co-ordinate x and y.

Parameters

x	The current position of the particle/ball.
x↩ *_0*	The current position of the other particle/ball to interact with.
r	The distance between the current and other particle/ball.
G	The gravitational constant.
M	The mass of the OTHER particle/ball.

Returns

Infinitesimal change in velocity.

5.1.2.17 resetToCollided()

```
void Ball::resetToCollided ( )
```

Sets the ball collision state to false.

Returns

Void.

5.1.2.18 sampleCurrentPosition()

```
void Ball::sampleCurrentPosition ( )
```

Stores the current position of the ball to a std::deque containing the "history" of the ball positions.

Returns

Void.

5.1.2.19 sampleNextPosition()

```
void Ball::sampleNextPosition ( )
```

Stores the current position of the ball to a std::vector containing the "history" of the ball positions. Also erases the oldest entry in the history vector given by positionSize.

Returns

Void.

5.1.2.20 setToCollided()

```
void Ball::setToCollided ( )
```

Sets the ball collision state to true.

Returns

Void.

5.1.2.21 setVelocity()

```
void Ball::setVelocity (
             sf::Vector2f vel )
```

Set the current velocity of the ball.

Parameters

vel	The velocity vector to set the ball to.

Returns

Void.

5.1.2.22 timeToCollision()

```
float Ball::timeToCollision (
             Ball & otherBall )
```

Calculate the time to collision with another ball.

Parameters

&otherBall	Reference to the other ball.

Returns

Time to collision.

5.1.2.23 updatePosition()

```
void Ball::updatePosition (
             float dt )
```

Updates the ball position based on the current velocity.

Parameters

dt	The simulation timestep.

Returns

Void.

5.1.2.24 updateVelocity()

```
void Ball::updateVelocity (
             float dt,
             Ball & otherBall )
```

Updates the velocity of the current ball by calculating forces on the ball.

Parameters

dt	The simulation timestep.
&otherBall	The other ball to interact with.

Returns

Void.

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

- headers/classBall.h
- source/simulation/classBall.cpp

5.2 BallUniverse Class Reference

Public Member Functions

- **BallUniverse** (int worldSizeX, int worldSizeY, float dt, bool force=true, bool collision=true)
- void **universeLoop** ()
- void **spawnNewBall** (sf::Vector2f position, sf::Vector2f velocity, float radius=1, float mass=1)
- void **createBallGrid** (int numWide, int numHigh, float spacing, sf::Vector2f centralPosition, sf::Vector2f init←_velocity={0, 0}, float ballMass=1, float ballRadius=1)
- void **drawBalls** (sf::RenderWindow &windowRef)
- sf::Vector2i **getWorldSize** ()
- void **incSimStep** (float delta)
- void **decSimStep** (float delta)
- void **setSimStep** (float delta)
- void **toggleSimPause** ()
- void **toggleCollisions** ()
- void **toggleForces** ()
- void **clearSimulation** ()
- void **changeBallColour** ()
- const int & **getWorldSizeX** ()
- int & **getNumOfBalls** ()
- bool & **getCollisionsEnabled** ()
- bool & **getForcesEnabled** ()
- float & **getTotalKE** ()
- sf::Vector2f & **getTotalMomentum** ()
- void **sampleAllPositions** ()
- void **drawSampledPositions** (sf::RenderWindow &window)
- void **toggleTrajectories** ()
- sf::Vector2f **getBallPosition** (int i)
- void **pushBall** (float force, float relDirection, int i)

Private Member Functions

- std::tuple< int, int, float > **findShortestCollTime** (float t_coll, std::vector< Ball > &ballArray, float dt=1e+10)
- void **calcTotalKE** (std::vector< Ball > &ballArray)
- void **calcTotalMomentum** (std::vector< Ball > &ballArray)
- void **physicsLoop** ()

Private Attributes

- std::vector< Ball > **ballArray**
- int **worldSizeX**
- int **worldSizeY**
- int **numOfBalls** = 0
- int **collider1** = 0
- int **collider2** = 0
- bool **enable_forces**
- bool **enable_collisions**
- float **currentTime** = 0
- float **timeToNextColl** = 1e+15
- float **dt**
- bool **isPaused** = false
- float **sampledt** = 5∗dt
- float **timeToNextSample** = sampledt
- bool **enable_trajectories**
- float **totalKE** = 0
- sf::Vector2f **totalMomentum** = sf::Vector2f{0,0}

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

- headers/classUniverse.h
- source/simulation/classUniverse.cpp

5.3 classBall Class Reference

5.3.1 Detailed Description

Purpose: A class which represents a physical simulation ball. Contains functions for forces, updating velocity and position as well as basic collisions.

Author

mjjq

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

- source/simulation/classBall.cpp

5.4 NonSimulationStuff Class Reference

Public Member Functions

- **NonSimulationStuff** (int m_windowSizeX, int m_windowSizeY, int spawnVelFactor, float spawnRadius, float spawnMass, float ballGridSpacing, int ballGridHeight, int ballGridWidth, bool simFitsInWindow, BallUniverse sim)
- void **mainLoop** ()

Private Member Functions

- void increaseMass ()
- void zoomToMouse (float zoomFactor)
- sf::Vector2f getEffectiveZoom (int worldSizeX, int worldSizeY)
- void checkForViewPan (sf::Vector2i initialPos, sf::Vector2f originalView, int worldSizeX, int worldSizeY, bool keyBool)
- void focusOnBall (int ballIndex, bool keyBool)
- void resetView ()
- void adjustViewSize (int sizeX, int sizeY, int worldSizeX, int worldSizeY)
- void toggleFullScreen ()
- void checkMBPress (sf::Vector2i &initPos, bool mouseType)
- sf::Vector2f velocityFromMouse (sf::Vector2i mousePosOnClick, int spawnVelFactor)
- void changeBoundaryRect (sf::Vector2i worldSize)
- void mouseWheelZoom (bool keyPress, float delta)
- void resetUIClick ()
- void clickOnUI ()
- void mouseWorldEvents (sf::Event &event)
- void mouseViewEvents (sf::Event &event)
- void mouseUIEvents (sf::Event &event)
- void keyEvents (sf::Event &event)
- void resizeEvents (sf::Event &event)
- void playerKeysDown (int player)
- void incTimeStep (sf::Time delta)
- void decTimeStep (sf::Time delta)
- void newLayerEvent (std::vector< bool > &newLayerKeys, sf::Event &event)
- float getWindowSizeX ()

Private Attributes

- int **windowSizeX**
- int **windowSizeY**
- sf::RenderWindow **window** {sf::VideoMode(windowSizeX,windowSizeY), "N-body"}
- sf::View **worldView** = window.getDefaultView()
- sf::View **GUIView** = window.getDefaultView()
- float **currentZoom** = 1.0f
- sf::Time **timestep** = sf::milliseconds(1000.0/60.0)
- sf::Time **minTimeToNextSpawn** = sf::milliseconds(500)
- sf::Time **timeToNextSpawn** = sf::milliseconds(0)
- sf::RectangleShape **boundaryRect**
- bool **simFitsInWindow**
- int **prevWindowSizeX**
- int **prevWindowSizeY**
- sf::Vector2i **prevWindowPosition**
- bool **isFullScreen** = false
- sf::Vector2i **mousePosOnClick**
- sf::Vector2i **mousePosOnPan**
- sf::Vector2i **mousePosOnRelease**
- std::pair< bool, int > **mouseOnUIWhenClicked** {false, -1}
- bool **clickedWindowToDrag** = false
- sf::Vector2f **recentViewCoords**
- sf::Vector2i **wSize**
- int **spawnVelFactor**
- float **spawnRadius**

- float **spawnMass**
- float **ballGridSpacing**
- int **ballGridHeight**
- int **ballGridWidth**
- int **playerBallIndex** = 0
- BallUniverse **ballSim**
- UIContainer **container**

5.4.1 Detailed Description

Purpose: Handle all aspects of the program which are not involved in simulation. This includes event handling of user inputs, and functions to do with rendering.

Author

mjjq

5.4.2 Member Function Documentation

5.4.2.1 adjustViewSize()

```
void NonSimulationStuff::adjustViewSize (
            int sizeX,
            int sizeY,
            int worldSizeX,
            int worldSizeY )  [private]
```

Scales the view to the window based on the current window and world sizes. This function ensure the world level zoom is preserved as well as accounting for vertical aspect ratios by checking for effective zooms in each direction.

Parameters

sizeX	The x-direction size of the window.
sizeY	The y-direction size of the window.
worldSizeX	The x-direction size of the simulation world.
worldSizeY	The y-direction size of the simulation world.

Returns

Void.

5.4.2.2 changeBoundaryRect()

```
void NonSimulationStuff::changeBoundaryRect (
            sf::Vector2i worldSize )  [private]
```

Changes the drawn rectangle which is the boundary of the simulation.

Parameters

worldSize	The size of the simulation world.

Returns

Void.

5.4.2.3 checkForViewPan()

```
void NonSimulationStuff::checkForViewPan (
             sf::Vector2i initialPos,
             sf::Vector2f originalView,
             int worldSizeX,
             int worldSizeY,
             bool keyBool )  [private]
```

Checks if a chosen key is pressed. If so, the mouse is made invisible and the view is panned based on the relative position between the mouse's position on the key press and its current position.

Parameters

initialPos	The position of the mouse when the chosen key is pressed.
originalView	The central position of the view when the chosen key is pressed.
worldSizeX	The x-direction size of the simulation world.
worldSizeY	The y-direction size of the simulation world.
keyBool	The boolean value of the chosen key, isKeyPressed.

Returns

Void.

5.4.2.4 checkMBPress()

```
void NonSimulationStuff::checkMBPress (
             sf::Vector2i & initPos,
             bool mouseType )  [private]
```

If the mouse button is pressed, draws a line between position where mouse was clicked and the current position of the mouse.

Parameters

&initPos	The click position of the mouse.
mouseType	Boolean which draws line if set to true.

5.4.2.5 clickOnUI()

```
void NonSimulationStuff::clickOnUI ( )  [private]
```

Clicks on a User Interface window.

Returns

Void.

5.4.2.6 decTimeStep()

```
void NonSimulationStuff::decTimeStep (
             sf::Time delta )  [private]
```

Decreases the rendering timestep.

Parameters

delta	The amount to decrease the rendering timestep by.

Returns

Void.

5.4.2.7 focusOnBall()

```
void NonSimulationStuff::focusOnBall (
             int ballIndex,
             bool keyBool )  [private]
```

Adjusts the camera to center on a chosen ball within the simulation if a chosen key is held.

Parameters

ballIndex	The ball to focus on, specified by an integer index.
keyBool	The boolean isKeyPressed value of the chosen key.

Returns

Void.

5.4.2.8 getEffectiveZoom()

```
sf::Vector2f NonSimulationStuff::getEffectiveZoom (
             int worldSizeX,
             int worldSizeY )  [private]
```

Generates effective zooms in the x and y directions. The effective zoom is the zoom factor in world co-ordinates as opposed to window co-ordinates, and so will differ in the x and y directions if the window is not square but rectangular. If the simulation is set to not fit within the window (simFitsInWindow== false), then the window and world are 1:1 in co-ordinates, thus the effective zoom is 1.

Parameters

worldSizeX	The x-direction size of the simulation world.
worldSizeY	The y-direction size of the simulation world.

Returns

The effective world zoom of each direction in vector form.

5.4.2.9 getWindowSizeX()

```
float NonSimulationStuff::getWindowSizeX ( )  [private]
```

Returns the current x-direction window size.

Returns

x-direction window size.

5.4.2.10 increaseMass()

```
void NonSimulationStuff::increaseMass ( )  [private]
```

Increases the spawn mass by 1.

5.4.2.11 incTimeStep()

```
void NonSimulationStuff::incTimeStep (
             sf::Time delta )  [private]
```

Increases the rendering timestep.

Parameters

delta	The amount to increase the rendering timestep by.

Returns

Void.

5.4.2.12 keyEvents()

```
void NonSimulationStuff::keyEvents (
            sf::Event & event )  [private]
```

Function which handles general key based events. Events are processed provided there are no key primary keys held. If these keys are held, the newLayerEvent events are processed instead.

Parameters

&event	The event case to process.

Returns

Void.

5.4.2.13 mouseUIEvents()

```
void NonSimulationStuff::mouseUIEvents (
            sf::Event & event )  [private]
```

Function which handles all events to do with generating interactions with the User Interface.

Parameters

&event	The event case to process.

Returns

Void.

5.4.2.14 mouseViewEvents()

```
void NonSimulationStuff::mouseViewEvents (
            sf::Event & event )  [private]
```

Function which handles all events to do with checking for UI interaction or adjusting the view.

Parameters

&event	The event case to process.

Returns

Void.

5.4.2.15 mouseWheelZoom()

```
void NonSimulationStuff::mouseWheelZoom (
             bool keyPress,
             float delta )  [private]
```

Zooms the world based on direction of mouse wheel scroll if a chosen key is pressed.

Parameters

keyPress	The isKeyPressed boolean value for a chosen key.
delta	The mousewheel delta value. Positive if the mousewheel is scrolled up, negative otherwise.

Returns

Void.

5.4.2.16 mouseWorldEvents()

```
void NonSimulationStuff::mouseWorldEvents (
             sf::Event & event )  [private]
```

Function which handles all events to do with generating interactions with the simulation world.

Parameters

&event	The event case to process.

Returns

Void.

5.4.2.17 newLayerEvent()

```
void NonSimulationStuff::newLayerEvent (
            std::vector< bool > & newLayerKeys,
            sf::Event & event )  [private]
```

Function which handles general key based events under some combination of primary key presses e.g. ctrl, alt, shift.

Parameters

&newLayerKeys	Vector of isKeyPressed boolean.
&event	The event case to process.

Returns

Void.

5.4.2.18 playerKeysDown()

```
void NonSimulationStuff::playerKeysDown (
            int player )  [private]
```

Function which handles player key held cases.

Parameters

player	The ball index which the player controls.

Returns

Void.

5.4.2.19 resetUIClick()

```
void NonSimulationStuff::resetUIClick ( )  [private]
```

Sets the current User Interface window to not be draggable.

Returns

Void.

5.4.2.20 resetView()

```
void NonSimulationStuff::resetView ( )  [private]
```

Resets the window view to a zoom of 1, centered on the simulation world centre.

Returns

Void.

5.4.2.21 resizeEvents()

```
void NonSimulationStuff::resizeEvents (
             sf::Event & event )  [private]
```

Function which handles events on window resize.

Parameters

&event	The event case to process.

Returns

Void.

5.4.2.22 toggleFullScreen()

```
void NonSimulationStuff::toggleFullScreen ( )  [private]
```

Toggles the window between windowed and fullscreen mode.

Returns

Void.

5.4.2.23 velocityFromMouse()

```
sf::Vector2f NonSimulationStuff::velocityFromMouse (
             sf::Vector2i mousePosOnClick,
             int spawnVelFactor )  [private]
```

Generates a spawn velocity for a new ball proportional to the distance between the mouse position on click and the current mouse position.

Parameters

mousePosOnClick	The mouse position when a chosen mouse button was clicked.
spawnVelFactor	The factor by which the aforementioned distance is multiplied. A larger value will generate a larger spawn velocity.

Returns

The calculated velocity.

5.4.2.24 zoomToMouse()

```
void NonSimulationStuff::zoomToMouse (
            float zoomFactor )  [private]
```

Applies a zoom to the rendered view, and centres the view to the position of the mouse. The if statement imposes a maximum zoom out limit.

Parameters

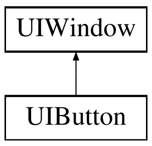
zoomFactor	The factor by which to zoom the current view.

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

- headers/classNonSim.h
- source/events/classNonSimulationStuff.cpp

5.5 UIButton Class Reference

Inheritance diagram for UIButton:



Public Member Functions

- **UIButton** (std::string font, std::string text, int fontSize, std::function< void()> const &clickFunc, sf::Vector2f position, sf::Vector2f bSize, bool fixedToWin, sf::Color color={80, 80, 80, 150})
- void **clickButton** ()
- void **releaseButton** ()
- void **renderButton** (sf::RenderWindow &window, sf::View &GUIView)
- void **updateElement** (sf::RenderWindow &window, sf::Vector2f parentPosition)

Protected Attributes

- sf::Vector2f **currPosition** = origPosition
- sf::Color **clickedColor** = {30,30,30,150}
- sf::Color **unclickedColor**
- sf::Color **mouseOverColor** = {100,100,100,150}
- std::function< void()> **clickFunc**
- bool **buttonDown** = false
- bool **displayElement** = true

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

- headers/classUIButton.h
- source/UI/classUIButton.cpp

5.6 UIContainer Class Reference

Public Member Functions

- void **addWindow** (sf::Vector2f position, float width, float height, bool fixedToWin, bool draggable=false, sf↩ ::Color color={50, 50, 50, 150})
- void **renderWindows** (sf::RenderWindow &window, sf::View &GUIView, sf::View &originalView)
- UIWindow & **getWindow** (int windowIndex)
- void **checkMouseIntersection** (sf::RenderWindow &window, sf::View &GUIView, sf::View &originalView)
- std::pair< bool, int > **doesMIntExist** ()
- void **clickOnUI** (sf::RenderWindow &window)
- void **resetUIClick** ()
- bool **isWindowDraggable** ()
- void **dragWindow** (sf::RenderWindow &window)

Private Attributes

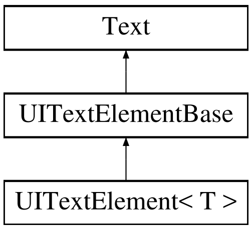
- std::vector< UIWindow > **interfaceWindows**
- std::vector< int > **interfaceWindowIDs**
- std::vector< bool > **mouseIntersectionList**
- std::pair< bool, int > **currentIntButton**
- std::pair< bool, int > **currentIntWindow**

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

- headers/classUIContainer.h
- source/UI/classUIContainer.cpp

5.7 UITextElement< T > Class Template Reference

Inheritance diagram for UITextElement< T >:



Public Member Functions

- **UITextElement** (std::string text, sf::Vector2f position, bool fixedToWin, T ∗var=nullptr, bool wrapText=false, sf::Rect< float > wrapBounds=sf::Rect< float >(0, 0, 1000, 1000))
- void **updateElement** (sf::RenderWindow &window, sf::View &GUIView, sf::Vector2f parentPosition)
- void **setOrigPosition** (sf::Vector2f newPosition)
- void **textWrap** (sf::Rect< float > parentRect)

Private Attributes

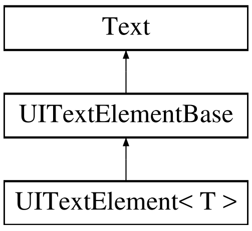
- std::string **initialText**
- std::string **wrappedText**
- sf::Font **displayFont**
- sf::Vector2f **origPosition**
- bool **displayElement** = true
- bool **fixedToWindow**
- T ∗ **variable**
- T **displayVariable**

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

- headers/classTextElement.h
- source/UI/classUITextElement.cpp

5.8 UITextElementBase Class Reference

Inheritance diagram for UITextElementBase:



Public Member Functions

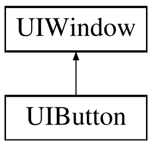
- virtual void **updateElement** (sf::RenderWindow &window, sf::View &GUIView, sf::Vector2f parentPosition)=0
- virtual void **setOrigPosition** (sf::Vector2f newPosition)=0
- virtual void **textWrap** (sf::Rect< float > parentRect)=0

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

- headers/classTextElementBase.h
- source/UI/classUITextElementBase.cpp

5.9 UIWindow Class Reference

Inheritance diagram for UIWindow:



Public Member Functions

- **UIWindow** (sf::Vector2f position, float width, float height, bool fixedToWin, bool draggable=false, sf::Color color={50, 50, 50, 150})
- template<class T >
  void **addElement** (std::string font, std::string str, int fontSize, sf::Vector2f position, T ∗var=nullptr)
- void **addButton** (std::string font, std::string text, int fontSize, sf::Vector2f position, sf::Vector2f bSize, std↩ ::function< void()> const &func, sf::Color color={80, 80, 80, 150})
- bool **ifElementsCollide** (sf::Rect< float > rectBound1, sf::Rect< float > rectBound2)
- void **renderWindow** (sf::RenderWindow &window, sf::View &GUIView)
- void **renderElements** (sf::RenderWindow &window, sf::View &GUIView)
- void **clickIntersectedButton** ()
- void **releaseClickedButton** ()
- void **checkMouseIntersection** (sf::RenderWindow &window)
- bool **getIsMouseIntersecting** ()
- void **resetButtonPair** ()
- std::pair< bool, int > **getClickedButton** ()
- void **changeOrigin** (sf::RenderWindow &window, sf::Vector2i origin)
- void **moveWindow** (sf::RenderWindow &window, sf::Vector2i newPosition)

Protected Attributes

- sf::Font **currentFont**
- sf::Vector2i **mouseOffset** = {0,0}
- sf::Vector2f **origPosition**
- float **width**
- float **height**
- sf::Rect< float > **origRect** {origPosition, {width,height}}
- sf::Color **color**
- sf::RectangleShape **windowBox**
- std::vector< UITextElementBase ∗ > **textArray**
- std::vector< UIButton ∗ > **buttonArray**
- bool **isButton** = false
- bool **fixedToWindow** = true
- bool **draggable** = true
- bool **mouseIntersecting** = false

Private Attributes

- std::pair< bool, int > **mouseOnButtonWhenClicked** {false, -1}
- std::pair< bool, int > **mouseOnButton** {false, -1}

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

- headers/classUIWindow.h
- source/UI/classUIWindow.cpp

Chapter 6

File Documentation

6.1 source/miscellaneous/sfVectorMath.cpp File Reference

```
#include <iostream>
#include <SFML/Graphics.hpp>
#include <cmath>
#include <math.h>
#include "../../headers/sfVectorMath.h"
```

Variables

- const float **PI** {3.14159265359}

6.1.1 Detailed Description

Purpose: Various mathematical vector operation functions for SFML's sf::Vector types. Includes dot product, norm etc.

Author

mjjq

6.2 source/miscellaneous/stringConversion.cpp File Reference

```
#include <iostream>
#include <sstream>
#include <SFML/Graphics.hpp>
```

Functions

- template<typename T >
  std::ostream & **operator<<** (std::ostream &outs, const sf::Vector2< T > &vec2)
- template<typename T >
  std::ostream & **operator<<** (std::ostream &outs, const bool &value)
- template<typename T >
  std::ostream & **operator<<** (std::ostream &outs, const sf::Rect< T > &rect)
- template<typename T >
  std::string **to_string** (const T &value)
- template std::string **to_string**< **sf::Vector2i** > (const sf::Vector2i &value)
- template std::string **to_string**< **sf::Vector2f** > (const sf::Vector2f &value)
- template std::string **to_string**< **sf::Rect**< **int** > > (const sf::Rect< int > &value)
- template std::string **to_string**< **sf::Rect**< **float** > > (const sf::Rect< float > &value)
- template std::string **to_string**< **bool** > (const bool &value)

6.2.1 Detailed Description

Purpose: Overloads << operator to include cases for handling SFML vectors and other SFML objects.

Author

mjjq

Index