

COLA

Generated by Doxygen 1.8.7

Tue Apr 29 2014 17:53:13

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	Class Documentation	5
3.1	COLA::DrawTools Class Reference	5
3.1.1	Detailed Description	5
3.1.2	Member Function Documentation	5
3.1.2.1	DrawFlowPoint	5
3.1.2.2	DrawMatches	5
3.1.2.3	DrawTau	6
3.2	COLA::FeatureTracker Class Reference	6
3.2.1	Detailed Description	6
3.2.2	Constructor & Destructor Documentation	6
3.2.2.1	FeatureTracker	7
3.2.3	Member Function Documentation	8
3.2.3.1	frameMatcher	8
3.2.3.2	generateDescriptors	8
3.3	COLA::FlowField Class Reference	8
3.3.1	Detailed Description	9
3.3.2	Constructor & Destructor Documentation	9
3.3.2.1	FlowField	9
3.3.3	Member Data Documentation	9
3.3.3.1	matches	9
3.4	COLA::FlowPoint Class Reference	9
3.4.1	Detailed Description	10
3.4.2	Constructor & Destructor Documentation	10
3.4.2.1	FlowPoint	10
3.4.2.2	FlowPoint	10
3.4.3	Member Function Documentation	10

3.4.3.1	operator==	10
3.4.4	Member Data Documentation	10
3.4.4.1	location	10
3.4.4.2	magnitude	10
3.5	COLA::FrameDescriptor Class Reference	11
3.5.1	Detailed Description	11
3.5.2	Constructor & Destructor Documentation	11
3.5.2.1	FrameDescriptor	11
3.5.3	Member Function Documentation	12
3.5.3.1	normalizeKeypoints	12
3.5.3.2	reset	12
3.5.4	Member Data Documentation	12
3.5.4.1	descriptors	12
3.5.4.2	featurePoints	12
3.5.4.3	frame_number	12
3.5.4.4	process_frame	12
3.5.4.5	refFrame	12
3.5.4.6	roi_offset	12
3.5.4.7	roi_rect	12
3.5.4.8	roi_set	12
3.5.4.9	timestamp	13
3.6	COLA::GlobalFlow Class Reference	13
3.6.1	Detailed Description	13
3.6.2	Member Function Documentation	13
3.6.2.1	CalculateGlobalFlow	13
3.7	COLA::Tau Class Reference	13
3.7.1	Detailed Description	14
3.7.2	Constructor & Destructor Documentation	14
3.7.2.1	Tau	14
3.7.2.2	Tau	14
3.7.3	Member Data Documentation	15
3.7.3.1	location	15
3.7.3.2	nodal	15
3.7.3.3	tau	15
3.8	COLA::TauMat Class Reference	15
3.8.1	Detailed Description	15
3.8.2	Constructor & Destructor Documentation	15
3.8.2.1	TauMat	15
3.9	COLA::Time Class Reference	17
3.9.1	Detailed Description	17

3.9.2	Constructor & Destructor Documentation	18
3.9.2.1	Time	18
3.9.3	Member Function Documentation	18
3.9.3.1	delay	18
3.9.3.2	Instance	18
3.9.3.3	setTime	18
3.9.3.4	timeElapsed	18
3.9.4	Member Data Documentation	20
3.9.4.1	isLag	20

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

COLA::DrawTools	5
COLA::FeatureTracker	6
COLA::FlowPoint	9
COLA::FrameDescriptor	11
COLA::GlobalFlow	13
Mat	
COLA::TauMat	15
Point3f	
COLA::Tau	13
COLA::Time	17
vector	
COLA::FlowField	8

Chapter 2

Class Index

2.1 Class List

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

COLA::DrawTools	
DrawTools is a class containing useful utilities for drawing outputs from COLA	5
COLA::FeatureTracker	
Track features between frame pairs	6
COLA::FlowField	
A FlowField represents a collection of FlowPoints from the same frame set	8
COLA::FlowPoint	
FlowPoint is an object representation of an optical flow vector	9
COLA::FrameDescriptor	
An object to contain all the relevant feature information for a frame	11
COLA::GlobalFlow	
Calculates the global flow from a COLA::FlowField	13
COLA::Tau	
Calculate the Tau value for a FlowPoint	13
COLA::TauMat	
Tau Mat UNSTABLE	15
COLA::Time	
How COLA views time	17

Chapter 3

Class Documentation

3.1 COLA::DrawTools Class Reference

[DrawTools](#) is a class containing useful utilities for drawing outputs from COLA.

```
#include <DrawTools.h>
```

Public Member Functions

- void [DrawFlowPoint](#) (cv::Mat &output, cv::Mat &input_frame, [FlowPoint](#) &flow_vector)
DrawFlowPoint is used to plot a plot point on an image.
- void [DrawMatches](#) (cv::Mat &output, [FrameDescriptor](#) &train, [FrameDescriptor](#) &query, [FlowField](#) &field)
DrawMatches is used to plot matches from a [COLA::FlowField](#) onto an output buffer.
- void [DrawTau](#) (cv::Mat &output, [COLA::TauMat](#) &tau_field)
DrawTau is used to plot a [Tau](#) point on an image.

3.1.1 Detailed Description

[DrawTools](#) is a class containing useful utilities for drawing outputs from COLA.

The Drawtools class provides functions for drawing matches, flow points, and tau fields

3.1.2 Member Function Documentation

3.1.2.1 void COLA::DrawTools::DrawFlowPoint (cv::Mat & output, cv::Mat & input_frame, [FlowPoint](#) & flow_vector)

DrawFlowPoint is used to plot a plot point on an image.

Parameters

<i>&output</i>	Contains a reference to the output frame we will write to.
<i>&input_frame</i>	Contains a reference to the output buffer
<i>&flow_vector</i>	Contains a flow vector reference that will be plotted on the output buffer.

3.1.2.2 void COLA::DrawTools::DrawMatches (cv::Mat & output, [FrameDescriptor](#) & train, [FrameDescriptor](#) & query, [FlowField](#) & field)

DrawMatches is used to plot matches from a [COLA::FlowField](#) onto an output buffer.

Parameters

<i>output</i>	contains a reference to the output frame we will write to.
<i>train</i>	Source of the train portion of the match set from
<i>query</i>	Source of the query portion of the match set from
<i>field</i>	The field that will be used to source the matches for the given frame set.

3.1.2.3 void COLA::DrawTools::DrawTau (cv::Mat & output, COLA::TauMat & tau_field)

DrawTau is used to plot a [Tau](#) point on an image.

Parameters

<i>&output</i>	Contains a reference to the output frame we will write to.
<i>&tau_field</i>	A reference to the tau field to be plotted

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

- include/COLA/DrawTools.h
- src/DrawTools.cpp

3.2 COLA::FeatureTracker Class Reference

Track features between frame pairs.

```
#include <FeatureTracker.h>
```

Public Member Functions

- [FeatureTracker](#) (unsigned int maxFeatures=1000)
FeatureTracker Constructor, initializes the image algorithms and sets max features to track. We use max features as a means of constraining the computational complexity of the algorithm. *FeatureTracker* will save the first n features until maxFeatures is reached.
- virtual [~FeatureTracker](#) ()
Destructor.
- bool [generateDescriptors](#) (COLA::FrameDescriptor &frameDescriptor)
Generate the descriptors for a given frame.
- bool [frameMatcher](#) (COLA::FrameDescriptor &train, COLA::FrameDescriptor &query, COLA::FlowField &field)
frameMatcher creates a [COLA::FlowField](#) for a pair of COLA::FrameDescriptors

3.2.1 Detailed Description

Track features between frame pairs.

This is the feature tracker used by COLA. We can select different algorithms for use in the tracker by assigning an instance to the relevant pointer in the constructor. It is assumed that all descriptors inherit the relevant interfaces from OpenCV.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 COLA::FeatureTracker::FeatureTracker (unsigned int *maxFeatures* = 1000)

[FeatureTracker](#) Constructor, initializes the image algorithms and sets max features to track. We use max features as a means of constraining the computational complexity of the algorithm. [FeatureTracker](#) will save the first n features until maxFeatures is reached.

Parameters

<i>maxFeatures</i>	integer representation of the max features to track
--------------------	---

3.2.3 Member Function Documentation

3.2.3.1 `bool COLA::FeatureTracker::frameMatcher (COLA::FrameDescriptor & train, COLA::FrameDescriptor & query, COLA::FlowField & field)`

frameMatcher creates a [COLA::FlowField](#) for a pair of COLA::FrameDescriptors

Parameters

<i>&train</i>	COLA::FrameDescriptor the set of train features
<i>&query</i>	COLA::FrameDescriptor the set of query features
<i>&field</i>	COLA::FlowField the data structure we store the flow field in (the difference between tracked points in train -> query)

3.2.3.2 `bool COLA::FeatureTracker::generateDescriptors (COLA::FrameDescriptor & frameDescriptor)`

Generate the descriptors for a given frame.

Parameters

<i>&frameDescriptor</i>	COLA::FrameDescriptor the query frame to generate descriptors for
-----------------------------	---

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

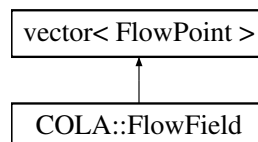
- include/COLA/FeatureTracker.h
- src/FeatureTracker.cpp

3.3 COLA::FlowField Class Reference

A [FlowField](#) represents a collection of FlowPoints from the same frame set.

```
#include <FlowField.h>
```

Inheritance diagram for COLA::FlowField:



Public Member Functions

- [FlowField](#) (int size)
Constructor.
- void [reset](#) (void)

Reset the [FlowField](#) This allows us to reset the flow field without releasing the memory allocated to the vector. This is useful when we have an upper bound on the number of possible flow points and do not want to have to reallocate memory in between matchings.

Public Attributes

- `vector< vector< cv::DMatch > > matches`

3.3.1 Detailed Description

A [FlowField](#) represents a collection of FlowPoints from the same frame set.

This is a vector grouping of FlowPoints for a given frame set. This is also used to carry the match vector used by the matcher for determining feature matches from the descriptors

3.3.2 Constructor & Destructor Documentation

3.3.2.1 COLA::FlowField::FlowField (int size)

Constructor.

Parameters

<i>size</i>	int the expected size of the flow field (used to pre-allocate the vector)
-------------	---

3.3.3 Member Data Documentation

3.3.3.1 vector<vector<cv::DMatch> > COLA::FlowField::matches

the vector of matches (for use by the matching algorithm)

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

- include/COLA/FlowField.h
- src/FlowField.cpp

3.4 COLA::FlowPoint Class Reference

[FlowPoint](#) is an object representation of an optical flow vector.

```
#include <FlowField.h>
```

Public Member Functions

- [FlowPoint](#) (cv::Point2f root, cv::Vec2f [magnitude](#))
FlowPoint Constructor (Pre-Calculated flow vector)
- [FlowPoint](#) (cv::Point2f start, cv::Point2f end, float time_delta)
FlowPoint Constructor (Point Delta form)
- bool [operator==](#) (const [FlowPoint](#) &other) const
FlowPoint equivalence operator This operator allows us to compare COLA::FlowPoints.

Public Attributes

- cv::Point2f [location](#)
- cv::Vec2f [magnitude](#)

3.4.1 Detailed Description

[FlowPoint](#) is an object representation of an optical flow vector.

A [FlowPoint](#) object represent the location and magnitude of an optical flow point.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 COLA::FlowPoint::FlowPoint (cv::Point2f *root*, cv::Vec2f *magnitude*)

[FlowPoint](#) Constructor (Pre-Calculated flow vector)

This constructor is used if we have a flow point we already calculated the magnitude for.

Parameters

<i>root</i>	cv::Point2f the location the flow vector represent the flow for. This is the current location of the feature.
<i>magnitude</i>	cv::Vec2f floating point representation of the x and y portions of the optical flow vector magnitude

3.4.2.2 COLA::FlowPoint::FlowPoint (cv::Point2f *start*, cv::Point2f *end*, float *time_delta*)

[FlowPoint](#) Constructor (Point Delta form)

Parameters

<i>start</i>	cv::Point2f the initial feature location
<i>end</i>	cv::Point2f the current (final) location of the feature
<i>time_delta</i>	float the elapsed time (in seconds) between start and end

3.4.3 Member Function Documentation

3.4.3.1 bool COLA::FlowPoint::operator== (const [FlowPoint](#) & *other*) const

[FlowPoint](#) equivalence operator This operator allows us to compare COLA::FlowPoints.

Parameters

<i>&other</i>	COLA::FlowPoint a reference to the FlowPoint we wish to compare this too.
-------------------	---

3.4.4 Member Data Documentation

3.4.4.1 cv::Point2f COLA::FlowPoint::location

The location of the flow point (currently)

3.4.4.2 cv::Vec2f COLA::FlowPoint::magnitude

The magnitude of the flow

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

- include/COLA/FlowField.h
- src/FlowField.cpp

3.5 COLA::FrameDescriptor Class Reference

An object to contain all the relevant feature information for a frame.

```
#include <FrameDescriptor.h>
```

Public Member Functions

- [FrameDescriptor](#) (int numberOfFeatures=1000, cv::Rect [roi_rect](#)=cv::Rect())
Constructor.
- void [normalizeKeypoints](#) (void)
Transforms the KeyPoint vector to real-space.
- void [reset](#) (void)
prepare for a new reference frame without deallocating memory

Public Attributes

- cv::Rect [roi_rect](#)
- bool [roi_set](#)
- cv::Point2f [roi_offset](#)
- std::vector< cv::KeyPoint > [featurePoints](#)
- cv::Mat [descriptors](#)
- int [frame_number](#)
- struct timespec [timestamp](#)
- cv::Mat [refFrame](#)
- cv::Mat * [process_frame](#)

3.5.1 Detailed Description

An object to contain all the relevant feature information for a frame.

The [FrameDescriptor](#) class is a data structure that represents everything we know about a given frame. It contains the ROI information, source frame, Key Points (where interesting features are located) and the descriptors for those features. It also contains timestamp information useful for determining the elapsed time between frames. This is represented by both timestamp (for real-time implementations) and frame number (for offline video processing). See [COLA::Time](#) for more information on how time is tracked by the program. The ROI rectangle is used to create an ROI image from the original frame. This way, we can track the normalization of keypoints with respect to the original image.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 COLA::FrameDescriptor::FrameDescriptor (int *numberOfFeatures* = 1000, cv::Rect *roi_rect* = cv::Rect ())

Constructor.

We initialize a frame descriptor with number of feature and an optional ROI rectangle.

Parameters

<i>numberOfFeatures</i>	int the maximum number of features to track
-------------------------	---

<i>roi_rect</i>	cv::Rect the ROI rectangle we will use to generate the sub-image for processing.
-----------------	--

3.5.3 Member Function Documentation

3.5.3.1 void COLA::FrameDescriptor::normalizeKeypoints (void)

Transforms the KeyPoint vector to real-space.

This function allows us to transform the keypoints from ROI space to the full image space. This function will only carry out the transform once to prevent accidentally breaking the keypoint vector.

3.5.3.2 void COLA::FrameDescriptor::reset (void)

prepare for a new reference frame without deallocating memory

This resets the state of the frame descriptor by clearing the keypoint vector

3.5.4 Member Data Documentation

3.5.4.1 cv::Mat COLA::FrameDescriptor::descriptors

The matrix of descriptors for the frame keypoints

3.5.4.2 std::vector<cv::KeyPoint> COLA::FrameDescriptor::featurePoints

A vector of keypoints generated by the feature extractor.

3.5.4.3 int COLA::FrameDescriptor::frame_number

A number representing the number of frames since the start of the program

3.5.4.4 cv::Mat* COLA::FrameDescriptor::process_frame

A pointer to the memory location of the frame (or subframe) we are going to run the algorithm on.

3.5.4.5 cv::Mat COLA::FrameDescriptor::refFrame

The original image this descriptor set belongs too.

3.5.4.6 cv::Point2f COLA::FrameDescriptor::roi_offset

The vector representing the offset between the original image and the ROI rectangle

3.5.4.7 cv::Rect COLA::FrameDescriptor::roi_rect

a rectangle representing the Region of Interest (ROI)

3.5.4.8 bool COLA::FrameDescriptor::roi_set

if true, we have set a Region of Interest

3.5.4.9 struct timespec COLA::FrameDescriptor::timestamp

Represents the CPU time when the frame was created. We use this in determining the real elapsed time.

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

- include/COLA/FrameDescriptor.h
- src/FrameDescriptor.cpp

3.6 COLA::GlobalFlow Class Reference

Calculates the global flow from a [COLA::FlowField](#).

```
#include <GlobalFlow.h>
```

Public Member Functions

- [GlobalFlow](#) (cv::Size frame_size)
Constructor (Pre-allocation)
- [COLA::FlowPoint CalculateGlobalFlow](#) ([COLA::FlowField](#) &flowField)
Global Flow calculator.

3.6.1 Detailed Description

Calculates the global flow from a [COLA::FlowField](#).

Sometimes we would like to see the flow of the whole image, we accomplish this by averaging an entire [FlowField](#) for a given set of frames.

3.6.2 Member Function Documentation

3.6.2.1 COLA::FlowPoint COLA::GlobalFlow::CalculateGlobalFlow ([COLA::FlowField](#) & flowField)

Global Flow calculator.

Parameters

<i>&flowField</i>	COLA::FlowField a reference to the flow field we will average to return the global flow vector.
-----------------------	---

Returns

[COLA::FlowPoint](#) the optical flow vector representing the average flow of the image. The location of this vector is assumed to be the camera nodal point.

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

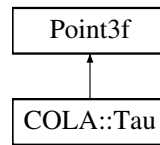
- include/COLA/GlobalFlow.h
- src/GlobalFlow.cpp

3.7 COLA::Tau Class Reference

Calculate the [Tau](#) value for a [FlowPoint](#).

```
#include <Tau.h>
```

Inheritance diagram for COLA::Tau:



Public Member Functions

- [Tau](#) (float [tau](#), cv::Point2f [location](#))
Constructor (Pre-calculated [Tau](#))
- [Tau](#) (COLA::FlowPoint &flow_pt, cv::Point2f &nodal)
Constructor (Calculate [Tau](#) from [FlowPoint](#))

Public Attributes

- float [tau](#)
- cv::Point2f [location](#)
- cv::Point2f [nodal](#)

3.7.1 Detailed Description

Calculate the [Tau](#) value for a [FlowPoint](#).

[Tau](#) is a mathematical representation of the time to collision with an object. [Tau](#) is effectively the orthogonal projection of the flow vector onto the nodal point.

$$\tau = \frac{||\vec{AN}||}{||\vec{AN}_{flow}||}$$

3.7.2 Constructor & Destructor Documentation

3.7.2.1 COLA::Tau::Tau (float [tau](#), cv::Point2f [location](#))

Constructor (Pre-calculated [Tau](#))

Parameters

<i>tau</i>	float the time until impact
<i>location</i>	cv::Point2f the location of the tau point.

3.7.2.2 COLA::Tau::Tau (COLA::FlowPoint & [flow_pt](#), cv::Point2f & [nodal](#))

Constructor (Calculate [Tau](#) from [FlowPoint](#))

This constructor calculates tau from a flow point and specified nodal (in relation to the [FlowPoint](#)).

Parameters

<i>&flow_pt</i>	COLA::FlowPoint the flow point we will calculate tau for
---------------------	--

<i>&nodal</i>	cv::Point2f the nodal point of the camera (usually the center point of the frame)
-------------------	---

3.7.3 Member Data Documentation

3.7.3.1 cv::Point2f COLA::Tau::location

The current location of the point on the image plane

3.7.3.2 cv::Point2f COLA::Tau::nodal

The nodal point of the camera (in reference to the image plane) used to calculate the [Tau](#) value

3.7.3.3 float COLA::Tau::tau

The time until collision in seconds

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

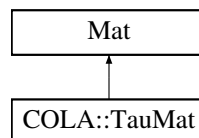
- include/COLA/Tau.h
- src/Tau.cpp

3.8 COLA::TauMat Class Reference

[Tau](#) Mat UNSTABLE.

```
#include <Tau.h>
```

Inheritance diagram for COLA::TauMat:



Public Member Functions

- [TauMat](#) (cv::Size mat_size, vector< [COLA::Tau](#) > tau_field)
Constructor (Create from vector<COLA::Tau>)

3.8.1 Detailed Description

[Tau](#) Mat UNSTABLE.

This object is currently not in a stable state, however, it is designed to report the tau values in a matrix normalized to uchar for display

3.8.2 Constructor & Destructor Documentation

3.8.2.1 COLA::TauMat::TauMat (cv::Size mat_size, vector< COLA::Tau > tau_field)

Constructor (Create from vector<COLA::Tau>)

This constructor creates a [TauMat](#) from a tau_field vector of [COLA::Tau](#) points

Parameters

<i>mat_size</i>	cv::Size the size of the output matrix (usually same as the frame size)
<i>tau_field</i>	vector<COLA::Tau> a vector of COLA::Tau points

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

- include/COLA/Tau.h
- src/Tau.cpp

3.9 COLA::Time Class Reference

How COLA views time.

```
#include <Time.h>
```

Public Member Functions

- float [timeElapsed](#) (COLA::FrameDescriptor &start, COLA::FrameDescriptor &end)
Returns the elapsed time in seconds.
- bool [delay](#) (void)
burn off remaining time
- void [setTime](#) (COLA::FrameDescriptor &frame)
sets the timestamp for NOW in a given COLA::FrameDescriptor

Static Public Member Functions

- static COLA::Time * [Instance](#) (int fps=DEFAULT_FPS, bool live=false)
Singleton Constructor/Instance Retrieve.

Public Attributes

- bool [isLag](#)

Protected Member Functions

- [Time](#) (int fps, bool live)
Constructor (PROTECTED)

3.9.1 Detailed Description

How COLA views time.

[COLA::Time](#) is a singleton object that makes a best effort at controlling the flow of time inside the program. For now, this is not written with actual real-time kernel usage. Essentially, the object "burns" time when the loop rate exceeds a pre designated FPS. It also contains a flag set when the loop time is above the current fps (the program is lagging).

[Time](#) has two modes, live and offline. The mode setting affects how time will externally report time. In offline mode, we assume that the data source is a video file. Thus, the time elapsed between frames is independent of the programs run time. Thus, time reports the elapsed time between to frames as the difference in frame number multiplied by the inverse of frames per second. In live mode, the object returns the actual clock difference between the two frames.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 COLA::Time::Time (int *fps*, bool *live*) [protected]

Constructor (PROTECTED)

Created a time object. This should never be directly used as the object is meant to be a singleton

Parameters

<i>fps</i>	the frames per second we would like Time to try to maintain
<i>live</i>	sets weather we are operating in real-time (live) mode or offline.

3.9.3 Member Function Documentation

3.9.3.1 bool COLA::Time::delay (void)

burn off remaining time

This function burns off any time remaining in the loop (to match requested FPS) and sets isLag to false. If the time is greater, it returns immediatly with true (lag) and sets isLag to true.

Returns

bool if true, the program is lagging

3.9.3.2 Time * COLA::Time::Instance (int *fps* = DEFAULT_FPS, bool *live* = false) [static]

Singleton Constructor/Instance Retrieve.

Since time is a singleton object, the constructor is made private. Instead, we use use the public Instance method to get the current instantiation, or construct a new instantiation using the supplied arguments. NOTE: if the singleton has already been Instantiated, the arguments fed into instance are ignored.

Parameters

<i>fps</i>	the frames per second we want to run at
<i>live</i>	sets whether the object is in real-time or offline mode. True sets to real-time mode False sets to offline.

Returns

[COLA::Time](#)* the pointer to the global singleton instantiation.

3.9.3.3 void COLA::Time::setTime (COLA::FrameDescriptor & *frame*)

sets the timestamp for NOW in a given [COLA::FrameDescriptor](#)

This function updates the frame number (number of frames prior since the beginning of execution) (ALWAYS) and the timestamp (ONLY IN LIVE MODE)

Parameters

<i>&frame</i>	the reference to the frame we want to set the time for.
-------------------	---

3.9.3.4 float COLA::Time::timeElapsed (COLA::FrameDescriptor & *start*, COLA::FrameDescriptor & *end*)

Returns the elapsed time in seconds.

This function returns the elapsed time in realtime or offline (frame count difference multiplied by inverse FPS) depending on how the original object was instantiated.

Parameters

<i>&start</i>	the earlier frame
<i>&end</i>	the later frame

Returns

float the elapsed time in seconds

3.9.4 Member Data Documentation**3.9.4.1 bool COLA::Time::isLag**

Flag set when the program is lagging (i.e. the main loop takes longer to run than the requested FPS).

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

- include/COLA/Time.h
- src/Time.cpp