
flom

Release 0.4.2

Feb 04, 2019

Contents

1	About this project	3
1.1	Features	3
1.2	Use from other languages	3
1.3	Relationship to MonoMotion	4
2	Installation	5
2.1	Runtime requirements	5
2.2	Current versions	5
3	How to build flom manually	7
3.1	Build requirements	7
3.2	Procedure	7
4	Basic usage	9
4.1	Headers	9
4.2	Load / Dump the motion	9
4.3	Obtain a frame	9
4.4	Iterate over frames	10
5	API Documentation	11
5.1	Class Hierarchy	11
5.2	File Hierarchy	11
5.3	Full API	11

About this project

flom is a library to handle keyframed motion of robots in C++. The main issue with formerly used representation-like plain csv or json (used in DeepMimic) with keyframes-is lack of interoperability. flom resolves that problem by providing basic functionalities to handle keyframed motion (looping, interpolation, etc) in one library.

1.1 Features

- Obtain a frame correspond to arbitrary time point
- Iterate over frames at arbitrary fps
- Import / Export the motion data
 - can be converted to JSON
 - the file format specification is represented in [protobuf](#)
- Edit keyframes (insert/delete)
- Effector support
 - Effectors can express the reference pose of links

1.2 Use from other languages

Currently, these bindings are available:

- [MonoMotion/flom-py](#) (Python)

1.3 Relationship to MonoMotion

In the development of [MonoMotion](#), we needed common representation of robot motion. flom is created for that purpose, but flom can be used for anything.

CHAPTER 2

Installation

Download and install suitable package from [GitHub Releases](#)

Latest build artifacts can be found at [bintray](#)

Or you can [build manually](#)

2.1 Runtime requirements

`libprotobuf.so` is required

2.2 Current versions

Lastest release: 0.4.2

Current build version (not released): 0.4.2

How to build flom manually

3.1 Build requirements

- boost (headers)
- protobuf 3.0.0 or later
- cmake 3.13.2 or later
- c++17 compiler
 - clang 5.0 or later
 - gcc 6.1 or later
- c++17 standard library
 - libc++ 7 or later
 - libstdc++ 6 or later

3.2 Procedure

```
git clone https://github.com/monomotion/flom --recursive
cd flom
mkdir build && cd $_
cmake ..
make -j $(nproc)
sudo make install
```


4.1 Headers

```
#include <flom/flom.hpp>
```

All required headers are included in `flom.hpp`

4.2 Load / Dump the motion

Loading and dumping is easy:

```
int main() {  
    auto motion = flom::Motion::load("file.fom");  
  
    // ... Edit loaded motion  
  
    motion.dump("out.fom");  
}
```

We recommend to use `.fom` as a file extension.

You can use `Motion::load_json` or `Motion::dump_json` if you like json

4.3 Obtain a frame

Use `Motion::frame_at` to obtain a frame at arbitrary time point. For example:

```
// Assume motion is an instance of flom::Motion  
auto frame = motion.frame_at(1.5);
```

Here, `frame` is a frame at 1.5 second since the motion is started.

4.4 Iterate over frames

```
for (auto&& [t, frame] : motion.frames(10)) {  
    // Do something with frame  
}
```

In this way, frames are iterated in 10fps(not actual time, but the time in the motion!). Also `t` holds the time of current frame.

5.1 Class Hierarchy

5.2 File Hierarchy

5.3 Full API

5.3.1 Namespaces

Namespace flom

Contents

- *Namespaces*
- *Classes*
- *Enums*
- *Functions*
- *Typedefs*

Namespaces

- *Namespace flom::compat*
- *Namespace flom::constants*
- *Namespace flom::errors*

- Namespace *floam::proto_util*

Classes

- Struct *Effector*
- Struct *EffectorType*
- Struct *Frame*
- Struct *Rotation*
- Class *CheckedFrameRef*
- Class *ConstKeyframeRange*
- Class *EffectorDifference*
- Class *EffectorWeight*
- Class *frame_iterator*
- Class *frame_iterator::Impl*
- Class *FrameDifference*
- Class *FrameRange*
- Class *keyframe_iterator*
- Class *KeyframeRange*
- Class *Location*
- Class *Motion*
- Class *Motion::Impl*

Enums

- Enum *CoordinateSystem*
- Enum *LoopType*

Functions

- Function *floam::interpolate(double, Location const&, Location const&)*
- Function *floam::interpolate(double, Rotation const&, Rotation const&)*
- Function *floam::interpolate(double, Effector const&, Effector const&)*
- Function *floam::interpolate(double, Frame const&, Frame const&)*
- Function *floam::interpolate(double, double, double)*
- Template Function *floam::lerp*
- Function *floam::loose_compare*
- Template Function *floam::names_hash(const std::unordered_set<K>&)*
- Template Function *floam::names_hash(const std::unordered_map<K, V>&)*

- *Function flom::operator!=(const Frame&, const Frame&)*
- *Function flom::operator!=(const Motion&, const Motion&)*
- *Function flom::operator!=(const frame_iterator&, const frame_iterator&)*
- *Function flom::operator!=(const keyframe_iterator&, const keyframe_iterator&)*
- *Function flom::operator!=(const Effector&, const Effector&)*
- *Function flom::operator-(const Frame&, const Frame&)*
- *Function flom::operator-(const frame_iterator&, const frame_iterator&)*
- *Function flom::operator-(const Effector&, const Effector&)*
- *Function flom::operator==(const Frame&, const Frame&)*
- *Function flom::operator==(const Motion&, const Motion&)*
- *Function flom::operator==(const frame_iterator&, const frame_iterator&)*
- *Function flom::operator==(const keyframe_iterator&, const keyframe_iterator&)*
- *Function flom::operator==(const Location&, const Location&)*
- *Function flom::operator==(const Rotation&, const Rotation&)*
- *Function flom::operator==(const EffectorDifference&, const EffectorDifference&)*
- *Function flom::operator==(const Effector&, const Effector&)*
- *Function flom::operator==(const EffectorType&, const EffectorType&)*
- *Function flom::operator==(const EffectorWeight&, const EffectorWeight&)*
- *Function flom::operator==(const FrameDifference&, const FrameDifference&)*

Typedefs

- *Typedef flom::KeyRange*

Namespace flom::compat

Contents

- *Typedefs*
- *Variables*

Typedefs

- *Typedef flom::compat::optional*

Variables

- *Variable flom::compat::nullopt*

Namespace flom::constants

Contents

- *Variables*

Variables

- *Variable flom::constants::float_point_tolerance*
- *Variable flom::constants::pi*

Namespace flom::errors

Contents

- *Classes*

Classes

- *Class InitKeyframeError*
- *Class InvalidFrameError*
- *Class InvalidTimeError*
- *Class InvalidWeightError*
- *Class JSONDumpError*
- *Class JSONLoadError*
- *Class KeyframeNotFoundError*
- *Class OutOfFramesError*
- *Class ParseError*
- *Class SerializationError*

Namespace flom::proto_util

Contents

- *Functions*

Functions

- Function `flom::proto_util::pack_coord_system`
- Function `flom::proto_util::pack_effector_type`
- Function `flom::proto_util::pack_effector_weight`
- Function `flom::proto_util::pack_location`
- Function `flom::proto_util::pack_quat`
- Function `flom::proto_util::pack_rotation`
- Function `flom::proto_util::pack_vec3`
- Function `flom::proto_util::unpack_coord_system`
- Function `flom::proto_util::unpack_effector_type`
- Function `flom::proto_util::unpack_effector_weight`
- Function `flom::proto_util::unpack_location`
- Function `flom::proto_util::unpack_quat`
- Function `flom::proto_util::unpack_rotation`
- Function `flom::proto_util::unpack_vec3`

5.3.2 Classes and Structs

Struct Effector

- Defined in *File effector.hpp*

Inheritance Relationships

Base Type

- `public boost::addable< Effector, EffectorDifference >`

Struct Documentation

```
struct Effector : public boost::addable<Effector, EffectorDifference>
```

Public Functions

```
Effector ()
```

```
Effector (const compat::optional<Location>&, const compat::optional<Rotation>&)
```

```
const compat::optional<Location> &location () const
```

```
compat::optional<Location> location ()
```

```
void set_location (const compat::optional<Location>&)
```

```
void clear_location ()

const compat::optional<Rotation> &rotation () const

compat::optional<Rotation> rotation ()

void set_rotation (const compat::optional<Rotation>&)

void clear_rotation ()

Effector new_compatible_effector () const

bool is_compatible (const Effector&) const

bool is_compatible (const EffectorDifference&) const

Effector &operator+= (const EffectorDifference&)
```

Struct EffectorType

- Defined in *File effector_type.hpp*

Inheritance Relationships

Base Type

- public boost::operators< EffectorType >

Struct Documentation

```
struct EffectorType : public boost::operators<EffectorType>
```

Public Functions

```
EffectorType ()

EffectorType (compat::optional<CoordinateSystem> location, compat::optional<CoordinateSystem>
              rotation)

compat::optional<CoordinateSystem> location () const

compat::optional<CoordinateSystem> rotation () const

void set_location (compat::optional<CoordinateSystem>)

void clear_location ()

void set_rotation (compat::optional<CoordinateSystem>)

void clear_rotation ()

Effector new_effector () const

bool is_compatible (const Effector&) const
```

Struct Frame

- Defined in *File frame.hpp*

Inheritance Relationships

Base Type

- `public boost::addable< Frame, FrameDifference >`

Struct Documentation

```
struct Frame : public boost::addable<Frame, FrameDifference>
```

Public Functions

```
Frame ()

Frame (const PositionsMap&, const EffectorsMap&)

const PositionsMap &positions () const
PositionsMap positions ()

void set_positions (const PositionsMap&)

void set_position (const std::string&, double)

const EffectorsMap &effectors () const
EffectorsMap effectors ()

void set_effectors (const EffectorsMap&)

void set_effector (const std::string&, const Effector&)

KeyRange<std::string> joint_names () const

KeyRange<std::string> effector_names () const

Frame new_compatible_frame () const

bool is_compatible (const Frame&) const

bool is_compatible (const FrameDifference&) const

Frame &operator+= (const FrameDifference&)
```

Struct Rotation

- Defined in *File effector.hpp*

Inheritance Relationships

Base Type

- `public boost::addable< Rotation, boost::subtractable< Rotation, boost::equality_comparable< Rotation, boost::multipliable< Rotation, std::size_t > > > >`

Struct Documentation

struct `Rotation` : **public** `boost::addable<Rotation, boost::subtractable<Rotation, boost::equality_comparable<Rotation, boost:`

Public Types

using `value_type` = `Eigen::Quaternion<double>`

Public Functions

Rotation ()

Rotation (double *w*, double *x*, double *y*, double *z*)

Rotation (const *value_type*&)

const *value_type* &quaternion () const

void **set_quaternion** (const *value_type*&)

double **w** () const

double **x** () const

double **y** () const

double **z** () const

std::tuple<double, double, double, double> **wxyz** () const

void **set_wxyz** (double, double, double, double)

Rotation &**operator+=** (const *Rotation*&)

Rotation &**operator-=** (const *Rotation*&)

Rotation &**operator*=** (std::size_t)

Class CheckedFrameRef

- Defined in *File range.hpp*

Class Documentation

class CheckedFrameRef

Public Types

`using reference_type = Frame&`

Public Functions

`CheckedFrameRef (reference_type value_, const Motion *motion_)`

`CheckedFrameRef &operator= (const Frame &frame)`

`operator reference_type () const`

Class ConstKeyframeRange

- Defined in *File range.hpp*

Class Documentation

class ConstKeyframeRange

Public Types

`using value_type = Frame`

`using const_iterator = typename std::map::const_iterator`

Public Functions

`ConstKeyframeRange ()`

`ConstKeyframeRange (const_iterator begin, const_iterator end)`

`ConstKeyframeRange (const ConstKeyframeRange&)`

`ConstKeyframeRange (ConstKeyframeRange&&)`

`ConstKeyframeRange &operator= (const ConstKeyframeRange&)`

`ConstKeyframeRange &operator= (ConstKeyframeRange&&)`

`const_iterator begin () const`

`const_iterator end () const`

`const_iterator cbegin () const`

`const_iterator cend () const`

Class EffectorDifference

- Defined in *File effector.hpp*

Inheritance Relationships

Base Type

- ```
private boost::addable< EffectorDifference, boost::equality_comparable<
EffectorDifference, boost::multipliable< EffectorDifference, std::size_t >
> >
```

## Class Documentation

**class EffectorDifference** : boost::addable<*EffectorDifference*, boost::equality\_comparable<*EffectorDifference*, boost::multipliable<*EffectorDifference*, std::size\_t>>>

### Public Functions

**EffectorDifference** (const *Effector*&, const *Effector*&)

**EffectorDifference** ()

**EffectorDifference** (const *EffectorDifference*&)

**EffectorDifference** (*EffectorDifference*&&)

*EffectorDifference* &operator= (const *EffectorDifference*&)

*EffectorDifference* &operator= (*EffectorDifference*&&)

const compat::optional<*Location*> &location () const

compat::optional<*Location*> location ()

const compat::optional<*Rotation*> &rotation () const

compat::optional<*Rotation*> rotation ()

*EffectorDifference* &operator\*= (std::size\_t)

*EffectorDifference* &operator+= (const *EffectorDifference*&)

bool is\_compatible (const *EffectorDifference*&) const

## Class EffectorWeight

- Defined in *File effector\_weight.hpp*



## Inheritance Relationships

### Base Type

- `private boost::operators< EffectorWeight >`

### Class Documentation

**class EffectorWeight** : `boost::operators<EffectorWeight>`

#### Public Functions

**EffectorWeight** ()

**EffectorWeight** (double *location*, double *rotation*)

double **location** () **const**

double **rotation** () **const**

void **set\_location** (double)

void **set\_rotation** (double)

### Class InitKeyframeError

- Defined in *File errors.hpp*

## Inheritance Relationships

### Base Type

- `public exception`

### Class Documentation

**class InitKeyframeError** : `public exception`

#### Public Functions

**InitKeyframeError** ()

**virtual const** char \***what** () **const**

### Class InvalidFrameError

- Defined in *File errors.hpp*

## Inheritance Relationships

### Base Type

- `public exception`

### Class Documentation

`class InvalidFrameError : public exception`

#### Public Functions

`InvalidFrameError (const std::string&)`

`virtual const char *what () const`

`std::string status_message () const`

#### Public Members

`std::string status`

### Class InvalidTimeError

- Defined in *File errors.hpp*

## Inheritance Relationships

### Base Type

- `public exception`

### Class Documentation

`class InvalidTimeError : public exception`

#### Public Functions

`InvalidTimeError (double)`

`virtual const char *what () const`

`double time () const`

## Class InvalidWeightError

- Defined in *File errors.hpp*

## Inheritance Relationships

### Base Type

- `public exception`

## Class Documentation

```
class InvalidWeightError : public exception
```

### Public Functions

```
InvalidWeightError (double)

virtual const char *what () const

double weight () const
```

## Class JSONDumpError

- Defined in *File errors.hpp*

## Inheritance Relationships

### Base Type

- `public exception`

## Class Documentation

```
class JSONDumpError : public exception
```

### Public Functions

```
JSONDumpError (const std::string&)

virtual const char *what () const

std::string status_message () const
```

## Public Members

`std::string status`

## Class JSONLoadError

- Defined in *File errors.hpp*

## Inheritance Relationships

### Base Type

- `public exception`

## Class Documentation

```
class JSONLoadError : public exception
```

### Public Functions

```
JSONLoadError (const std::string&)
```

```
virtual const char *what () const
```

```
std::string status_message () const
```

### Public Members

`std::string status`

## Class KeyframeNotFoundError

- Defined in *File errors.hpp*

## Inheritance Relationships

### Base Type

- `public exception`

## Class Documentation

```
class KeyframeNotFoundError : public exception
```

### Public Functions

```
KeyframeNotFoundError (double)
virtual const char *what () const
double time () const
```

### Class OutOfFramesError

- Defined in *File errors.hpp*

### Inheritance Relationships

#### Base Type

- public exception

### Class Documentation

```
class OutOfFramesError : public exception
```

### Public Functions

```
OutOfFramesError (double)
virtual const char *what () const
double time () const
```

### Class ParseError

- Defined in *File errors.hpp*

### Inheritance Relationships

#### Base Type

- public exception

### Class Documentation

```
class ParseError : public exception
```

## Public Functions

`ParseError()`

`virtual const char *what() const`

## Class SerializationError

- Defined in *File errors.hpp*

## Inheritance Relationships

### Base Type

- public exception

## Class Documentation

`class SerializationError : public exception`

## Public Functions

`SerializationError()`

`virtual const char *what() const`

## Class frame\_iterator

- Defined in *File range.hpp*

## Nested Relationships

### Nested Types

- *Class frame\_iterator::Impl*

## Class Documentation

`class frame_iterator`

## Public Types

```
using iterator_category = std::input_iterator_tag
using value_type = std::pair<double, Frame>
using difference_type = double
using pointer = Frame *
using reference = Frame&
```

## Public Functions

```
frame_iterator()
frame_iterator(Motion const &motion, double fps)
frame_iterator(const frame_iterator&)
frame_iterator(frame_iterator&&)
frame_iterator &operator=(const frame_iterator&)
frame_iterator &operator=(frame_iterator&&)
~frame_iterator()
value_type operator*() const
frame_iterator &operator++()
frame_iterator operator++(int)
frame_iterator &operator--()
frame_iterator operator--(int)
double current_time() const
class Impl
```

## Public Functions

```
Impl(const Motion &motion_, double fps_)
double current_time() const
bool check_is_end() const
```

## Public Members

```
const Motion *motion
double fps
long t_index = 0
```

## Class `frame_iterator::Impl`

- Defined in *File range.impl.hpp*

## Nested Relationships

This class is a nested type of *Class frame\_iterator*.

## Class Documentation

```
class Impl
```

### Public Functions

```
Impl (const Motion &motion_, double fps_)
```

```
double current_time () const
```

```
bool check_is_end () const
```

### Public Members

```
const Motion *motion
```

```
double fps
```

```
long t_index = 0
```

## Class `FrameDifference`

- Defined in *File frame.hpp*

## Inheritance Relationships

### Base Type

- ```
private boost::addable< FrameDifference, boost::equality_comparable<
    FrameDifference, boost::multipliable< FrameDifference, std::size_t > >
    >
```

Class Documentation

```
class FrameDifference : boost::addable<FrameDifference, boost::equality_comparable<FrameDifference, boost::multipliable<
```


Public Functions

```

FrameDifference (const Frame&, const Frame&)

FrameDifference ()

FrameDifference (const FrameDifference&)

FrameDifference (FrameDifference&&)

FrameDifference &operator= (const FrameDifference&)

FrameDifference &operator= (FrameDifference&&)

const std::unordered_map<std::string, double> &positions () const

std::unordered_map<std::string, double> positions ()

const std::unordered_map<std::string, EffectorDifference> &effectors () const

std::unordered_map<std::string, EffectorDifference> effectors ()

FrameDifference &operator*= (std::size_t)

FrameDifference &operator+= (const FrameDifference&)

bool is_compatible (const FrameDifference&) const

```

Class FrameRange

- Defined in *File range.hpp*

Class Documentation

```
class FrameRange
```

Public Types

```

using value_type = Frame

using iterator = frame_iterator

```

Public Functions

```

FrameRange ()

FrameRange (Motion const &motion_, double fps_)

FrameRange (const FrameRange&)

FrameRange (FrameRange&&)

FrameRange &operator= (const FrameRange&)

```

```
FrameRange &operator= (FrameRange&&)

iterator begin () const

iterator end () const

iterator cbegin () const

iterator cend () const
```

Class `keyframe_iterator`

- Defined in *File range.hpp*

Class Documentation

```
class keyframe_iterator
```

Public Types

```
using base_iterator = std::map<double, Frame>::iterator
using iterator_category = std::bidirectional_iterator_tag
using value_type = std::iterator_traits::value_type
using difference_type = std::iterator_traits::difference_type
using pointer = std::iterator_traits::pointer
using reference = std::iterator_traits::reference
using checked_value_type = std::pair<const double, CheckedFrameRef>
```

Public Functions

```
keyframe_iterator ()

keyframe_iterator (base_iterator it_, const Motion &motion_)

keyframe_iterator (const keyframe_iterator&)

keyframe_iterator (keyframe_iterator&&)

keyframe_iterator &operator= (const keyframe_iterator&)

keyframe_iterator &operator= (keyframe_iterator&&)

const value_type &operator* () const

checked_value_type operator* ()

const value_type &operator-> () const

checked_value_type operator-> ()

keyframe_iterator &operator++ ()
```

```
keyframe_iterator operator++ (int)
```

```
keyframe_iterator &operator-- ()
```

```
keyframe_iterator operator-- (int)
```

Class KeyframeRange

- Defined in *File range.hpp*

Class Documentation

```
class KeyframeRange
```

Public Types

```
using value_type = Frame
```

```
using iterator = keyframe_iterator
```

```
using base_iterator = typename std::map<double, Frame>::iterator
```

Public Functions

```
KeyframeRange ()
```

```
KeyframeRange (base_iterator begin_, base_iterator end_, const Motion &motion_)
```

```
KeyframeRange (const KeyframeRange&)
```

```
KeyframeRange (KeyframeRange&&)
```

```
KeyframeRange &operator= (const KeyframeRange&)
```

```
KeyframeRange &operator= (KeyframeRange&&)
```

```
iterator begin ()
```

```
iterator end ()
```

Class Location

- Defined in *File effector.hpp*

Inheritance Relationships

Base Type

- private boost::addable< Location, boost::subtractable< Location, boost::equality_comparable< Location, boost::multipliable< Location, std::size_t > > > >

Class Documentation

class **Location** : boost::addable<*Location*, boost::subtractable<*Location*, boost::equality_comparable<*Location*, boost::multiplia

Public Types

using **value_type** = Eigen::Matrix<double, 3, 1>

Public Functions

Location ()

Location (double *x*, double *y*, double *z*)

Location (const *value_type*&)

const *value_type* &**vector** () const

void **set_vector** (const *value_type*&)

double **x** () const

double **y** () const

double **z** () const

std::tuple<double, double, double> **xyz** () const

void **set_x** (double)

void **set_y** (double)

void **set_z** (double)

void **set_xyz** (double, double, double)

Location &**operator+=** (const *Location*&)

Location &**operator-=** (const *Location*&)

Location &**operator*=** (std::size_t)

Class Motion

- Defined in *File motion.hpp*

Nested Relationships

Nested Types

- *Class Motion::Impl*

Class Documentation

class Motion

Public Functions

Motion (**const** std::unordered_set<std::string> &*joint_names*, **const** std::unordered_map<std::string, *EffectorType*> &*effector_types*, **const** std::string &*model* = "")

Motion (*Motion* **const**&)

~Motion ()

bool **is_valid** () **const**

bool **is_valid_frame** (**const** *Frame*&) **const**

Frame **frame_at** (double *t*) **const**

FrameRange **frames** (double *fps*) **const**

bool **is_in_range_at** (double *t*) **const**

void **dump** (std::ostream&) **const**

void **dump_json** (std::ostream&) **const**

std::string **dump_json_string** () **const**

LoopType **loop** () **const**

void **set_loop** (*LoopType*)

std::string **model_id** () **const**

void **set_model_id** (std::string **const**&)

Frame **new_keyframe** () **const**

void **insert_keyframe** (double *t*, **const** *Frame*&)

void **delete_keyframe** (double *t*, bool *loose* = true)

KeyframeRange **keyframes** ()

ConstKeyframeRange **keyframes** () **const**

ConstKeyframeRange **const_keyframes** () **const**

void **clear_keyframes** ()

EffectorType **effector_type** (**const** std::string&) **const**

EffectorWeight **effector_weight** (**const** std::string&) **const**

void **set_effector_weight** (**const** std::string&, *EffectorWeight*)

double **length** () **const**

```
KeyRange<std::string> joint_names () const  
KeyRange<std::string> effector_names () const
```

Public Static Functions

```
static Motion load (std::istream&)  
static Motion load_json (std::istream&)  
static Motion load_json_string (std::string const&)  
class Impl
```

Public Functions

```
Impl (const std::unordered_set<std::string> &joints, const std::unordered_map<std::string, EffectorType> &effectors, const std::string &model = "")  
void add_initial_frame ()  
Frame new_keyframe () const  
proto::Motion to_protobuf () const  
bool is_valid () const  
bool is_valid_frame (const Frame&) const
```

Public Members

```
std::string model_id  
LoopType loop  
std::map<double, Frame> raw_frames  
const std::unordered_set<std::string> joint_names  
const std::unordered_map<std::string, EffectorType> effector_types  
std::unordered_map<std::string, EffectorWeight> effector_weights  
const std::size_t joints_hash  
const std::size_t effectors_hash
```

Public Static Functions

```
static Motion from_protobuf (proto::Motion const&)
```

Class Motion::Impl

- Defined in *File motion.impl.hpp*

Nested Relationships

This class is a nested type of *Class Motion*.

Class Documentation

class Impl

Public Functions

Impl (**const** std::unordered_set<std::string> &*joints*, **const** std::unordered_map<std::string, *EffectorType*> &*effectors*, **const** std::string &*model* = "")

void **add_initial_frame** ()

Frame **new_keyframe** () **const**

proto::Motion **to_protobuf** () **const**

bool **is_valid** () **const**

bool **is_valid_frame** (**const** *Frame*&) **const**

Public Members

std::string **model_id**

LoopType **loop**

std::map<double, *Frame*> **raw_frames**

const std::unordered_set<std::string> **joint_names**

const std::unordered_map<std::string, *EffectorType*> **effector_types**

std::unordered_map<std::string, *EffectorWeight*> **effector_weights**

const std::size_t **joints_hash**

const std::size_t **effectors_hash**

Public Static Functions

static *Motion* **from_protobuf** (proto::Motion **const**&)

5.3.3 Enums

Enum CoordinateSystem

- Defined in *File effector_type.hpp*

Enum Documentation

enum flom::CoordinateSystem

Values:

World

Local

Enum LoopType

- Defined in *File motion.hpp*

Enum Documentation

enum flom::LoopType

Values:

None

Wrap

5.3.4 Functions

Function flom::interpolate(double, Location const&, Location const&)

- Defined in *File interpolation.hpp*

Function Documentation

Location flom::interpolate (double *t*, *Location* const &*a*, *Location* const &*b*)

Function flom::interpolate(double, Rotation const&, Rotation const&)

- Defined in *File interpolation.hpp*

Function Documentation

Rotation flom::interpolate (double *t*, *Rotation* const &*a*, *Rotation* const &*b*)

Function flom::interpolate(double, Effector const&, Effector const&)

- Defined in *File interpolation.hpp*

Function Documentation

Effector flom::interpolate (double *t*, *Effector* const &*a*, *Effector* const &*b*)

Function flom::interpolate(double, Frame const&, Frame const&)

- Defined in *File interpolation.hpp*

Function Documentation

Frame flom::interpolate (double *t*, *Frame const* &*a*, *Frame const* &*b*)

Function flom::interpolate(double, double, double)

- Defined in *File interpolation.hpp*

Function Documentation

double flom::interpolate (double *t*, double *a*, double *b*)

Template Function flom::lerp

- Defined in *File interpolation.hpp*

Function Documentation

template <typename *T*, typename *U*, std::enable_if_t< std::is_floating_point< *U* >::value > * = nullptr>
T flom::lerp (*U t*, *T a*, *T b*)

Function flom::loose_compare

- Defined in *File loose_compare.hpp*

Function Documentation

bool flom::loose_compare (double, double)

Template Function flom::names_hash(const std::unordered_set<K>&)

- Defined in *File motion.impl.hpp*

Function Documentation

template <typename *K*>
 std::size_t flom::names_hash (const std::unordered_set<*K*> &*s*)

Template Function flom::names_hash(const std::unordered_map<K, V>&)

- Defined in *File motion.impl.hpp*

Function Documentation

```
template <typename K, typename V>
std::size_t flom::names_hash (const std::unordered_map<K, V> &m)
```

Function flom::operator!=(const Effector&, const Effector&)

- Defined in *File effector.hpp*

Function Documentation

```
bool flom::operator!= (const Effector&, const Effector&)
```

Function flom::operator!=(const Frame&, const Frame&)

- Defined in *File frame.hpp*

Function Documentation

```
bool flom::operator!= (const Frame&, const Frame&)
```

Function flom::operator!=(const Motion&, const Motion&)

- Defined in *File motion.hpp*

Function Documentation

```
bool flom::operator!= (const Motion&, const Motion&)
```

Function flom::operator!=(const frame_iterator&, const frame_iterator&)

- Defined in *File range.hpp*

Function Documentation

```
bool flom::operator!= (const frame_iterator&, const frame_iterator&)
```

Function flom::operator!=(const keyframe_iterator&, const keyframe_iterator&)

- Defined in *File range.hpp*

Function Documentation

```
bool flom::operator!= (const keyframe_iterator&, const keyframe_iterator&)
```

Function flom::operator-(const Effector&, const Effector&)

- Defined in *File effector.hpp*

Function Documentation

EffectorDifference flom::operator- (const *Effector*&, const *Effector*&)

Function flom::operator-(const Frame&, const Frame&)

- Defined in *File frame.hpp*

Function Documentation

FrameDifference flom::operator- (const *Frame*&, const *Frame*&)

Function flom::operator-(const frame_iterator&, const frame_iterator&)

- Defined in *File range.hpp*

Function Documentation

frame_iterator::difference_type flom::operator- (const *frame_iterator*&, const *frame_iterator*&)

Function flom::operator==(const Location&, const Location&)

- Defined in *File effector.hpp*

Function Documentation

bool flom::operator== (const *Location*&, const *Location*&)

Function flom::operator==(const Rotation&, const Rotation&)

- Defined in *File effector.hpp*

Function Documentation

bool flom::operator== (const *Rotation*&, const *Rotation*&)

Function flom::operator==(const EffectorDifference&, const EffectorDifference&)

- Defined in *File effector.hpp*

Function Documentation

bool flom::operator==(const EffectorDifference&, const EffectorDifference&)

Function flom::operator==(const Effector&, const Effector&)

- Defined in *File effector.hpp*

Function Documentation

bool flom::operator==(const Effector&, const Effector&)

Function flom::operator==(const EffectorType&, const EffectorType&)

- Defined in *File effector_type.hpp*

Function Documentation

bool flom::operator==(const EffectorType&, const EffectorType&)

Function flom::operator==(const EffectorWeight&, const EffectorWeight&)

- Defined in *File effector_weight.hpp*

Function Documentation

bool flom::operator==(const EffectorWeight&, const EffectorWeight&)

Function flom::operator==(const FrameDifference&, const FrameDifference&)

- Defined in *File frame.hpp*

Function Documentation

bool flom::operator==(const FrameDifference&, const FrameDifference&)

Function flom::operator==(const Frame&, const Frame&)

- Defined in *File frame.hpp*

Function Documentation

bool flom::operator==(const Frame&, const Frame&)

Function flom::operator==(const Motion&, const Motion&)

- Defined in *File motion.hpp*

Function Documentation

```
bool flom::operator==(const Motion&, const Motion&)
```

Function flom::operator==(const frame_iterator&, const frame_iterator&)

- Defined in *File range.hpp*

Function Documentation

```
bool flom::operator==(const frame_iterator&, const frame_iterator&)
```

Function flom::operator==(const keyframe_iterator&, const keyframe_iterator&)

- Defined in *File range.hpp*

Function Documentation

```
bool flom::operator==(const keyframe_iterator&, const keyframe_iterator&)
```

Function flom::proto_util::pack_coord_system

- Defined in *File proto_util.hpp*

Function Documentation

```
proto::EffectorType::Type flom::proto_util::pack_coord_system (compat::optional<CoordinateSystem>  
const&)
```

Function flom::proto_util::pack_effector_type

- Defined in *File proto_util.hpp*

Function Documentation

```
void flom::proto_util::pack_effector_type (EffectorType const&, proto::EffectorType *)
```

Function flom::proto_util::pack_effector_weight

- Defined in *File proto_util.hpp*

Function Documentation

void flom::proto_util::pack_effector_weight (*EffectorWeight* const&, proto::EffectorWeight *)

Function flom::proto_util::pack_location

- Defined in *File proto_util.hpp*

Function Documentation

void flom::proto_util::pack_location (*Location* const&, proto::Location *)

Function flom::proto_util::pack_quat

- Defined in *File proto_util.hpp*

Function Documentation

void flom::proto_util::pack_quat (*Rotation::value_type* const&, proto::Quaternion *)

Function flom::proto_util::pack_rotation

- Defined in *File proto_util.hpp*

Function Documentation

void flom::proto_util::pack_rotation (*Rotation* const&, proto::Rotation *)

Function flom::proto_util::pack_vec3

- Defined in *File proto_util.hpp*

Function Documentation

void flom::proto_util::pack_vec3 (*Location::value_type* const&, proto::Vec3 *)

Function flom::proto_util::unpack_coord_system

- Defined in *File proto_util.hpp*

Function Documentation

compat::optional<*CoordinateSystem*> flom::proto_util::unpack_coord_system (proto::EffectorType::Type const&)

Function flom::proto_util::unpack_effector_type

- Defined in *File proto_util.hpp*

Function Documentation

EffectorType flom::proto_util::unpack_effector_type (proto::EffectorType **const**&)

Function flom::proto_util::unpack_effector_weight

- Defined in *File proto_util.hpp*

Function Documentation

EffectorWeight flom::proto_util::unpack_effector_weight (proto::EffectorWeight **const**&)

Function flom::proto_util::unpack_location

- Defined in *File proto_util.hpp*

Function Documentation

Location flom::proto_util::unpack_location (proto::Location **const**&)

Function flom::proto_util::unpack_quat

- Defined in *File proto_util.hpp*

Function Documentation

Rotation::value_type flom::proto_util::unpack_quat (proto::Quaternion **const**&)

Function flom::proto_util::unpack_rotation

- Defined in *File proto_util.hpp*

Function Documentation

Rotation flom::proto_util::unpack_rotation (proto::Rotation **const**&)

Function flom::proto_util::unpack_vec3

- Defined in *File proto_util.hpp*

Function Documentation

Location::value_type flom::proto_util::unpack_vec3 (proto::Vec3 const&)

5.3.5 Variables

Variable flom::compat::nullopt

- Defined in *File optional.hpp*

Variable Documentation

auto flom::compat::nullopt = boost::none

Variable flom::constants::float_point_tolerance

- Defined in *File constants.hpp*

Variable Documentation

constexpr double flom::constants::float_point_tolerance = 0.00001

Variable flom::constants::pi

- Defined in *File constants.hpp*

Variable Documentation

constexpr auto flom::constants::pi = static_cast<T>(3.1415926535897932384626)

5.3.6 Typedefs

Typedef flom::compat::optional

- Defined in *File optional.hpp*

Typedef Documentation

using flom::compat::optional = typedef boost::optional<T>

Typedef flom::KeyRange

- Defined in *File frame.hpp*

Typedef Documentation

```
using flom::KeyRange = typedef boost::any_range<K, boost::forward_traversal_tag, std::add_
```

5.3.7 Directories

Directory include

Directory path: include

Subdirectories

- *Directory flom*

Files

- *File CMakeLists.txt*

Directory flom

Parent directory (include)

Directory path: include/flom

Subdirectories

- *Directory compat*

Files

- *File constants.hpp*
- *File effector.hpp*
- *File effector_type.hpp*
- *File effector_weight.hpp*
- *File errors.hpp*
- *File flom.hpp*
- *File frame.hpp*
- *File interpolation.hpp*
- *File loose_compare.hpp*
- *File motion.hpp*
- *File motion.impl.hpp*
- *File proto_util.hpp*

- *File range.hpp*
- *File range.impl.hpp*

Directory compat

Parent directory (include/flom)

Directory path: include/flom/compat

Files

- *File optional.hpp*

5.3.8 Files

File CMakeLists.txt

Parent directory (include)

Contents

- *Definition (include/CMakeLists.txt)*

Definition (include/CMakeLists.txt)

Program Listing for File CMakeLists.txt

Return to documentation for file (include/CMakeLists.txt)

```
#
# Copyright 2018 coord.e
#
# This file is part of Flom.
#
# Flom is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# Flom is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with Flom. If not, see <http://www.gnu.org/licenses/>.
#

file(GLOB HEADER_FILES flom/*.hpp)
add_custom_target(flom_headers SOURCES ${HEADER_FILES})
```

(continues on next page)

(continued from previous page)

```
enable_clang_format(flom_headers)
enable_clang_tidy(flom_headers)

install(DIRECTORY . DESTINATION include FILES_MATCHING PATTERN "*.hpp")
```

File constants.hpp

Parent directory (include/flom)

Contents

- *Definition* (include/flom/constants.hpp)
- *Included By*
- *Namespaces*
- *Variables*

Definition (include/flom/constants.hpp)

Program Listing for File constants.hpp

Return to documentation for file (include/flom/constants.hpp)

```
//
// Copyright 2018 coord.e
//
// This file is part of Flom.
//
// Flom is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// Flom is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
// along with Flom. If not, see <http://www.gnu.org/licenses/>.
//

#ifndef FLOM_CONSTANTS_HPP
#define FLOM_CONSTANTS_HPP

namespace flom::constants {

template <typename T>
static constexpr auto pi = static_cast<T>(3.1415926535897932384626);
```

(continues on next page)

(continued from previous page)

```
static constexpr double float_point_tolerance = 0.00001;

} // namespace flom::constants

#endif
```

Included By

- *File flom.hpp*
- *File loose_compare.hpp*

Namespaces

- *Namespace flom::constants*

Variables

- *Variable flom::constants::float_point_tolerance*
- *Variable flom::constants::pi*

File effector.hpp

Parent directory (include/flom)

Contents

- *Definition* (include/flom/effector.hpp)
- *Includes*
- *Included By*
- *Namespaces*
- *Classes*
- *Functions*

Definition (include/flom/effector.hpp)

Program Listing for File effector.hpp

Return to documentation for file (include/flom/effector.hpp)

```
//
// Copyright 2018 coord.e
//
```

(continues on next page)

(continued from previous page)

```

// This file is part of Flom.
//
// Flom is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// Flom is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
// along with Flom. If not, see <http://www.gnu.org/licenses/>.
//

#ifndef FLOM_EFFECTOR_HPP
#define FLOM_EFFECTOR_HPP

#include "flom/compat/optional.hpp"
#include <type_traits>

#include <boost/operators.hpp>

#include <Eigen/Dense>
#include <Eigen/Geometry>

namespace flom {

class Location
    : boost::addable<
        Location,
        boost::subtractable<
            Location,
            boost::equality_comparable<
                Location, boost::multipliable<Location, std::size_t>>>> {
public:
    using value_type = Eigen::Matrix<double, 3, 1>;

private:
    value_type vector_;

public:
    Location();
    Location(double x, double y, double z);

    explicit Location(const value_type &);

    const value_type &vector() const;
    void set_vector(const value_type &);

    double x() const;
    double y() const;
    double z() const;

    std::tuple<double, double, double> xyz() const;

```

(continues on next page)

(continued from previous page)

```

void set_x(double);
void set_y(double);
void set_z(double);

void set_xyz(double, double, double);

Location &operator+=(const Location &);
Location &operator-=(const Location &);
Location &operator*=(std::size_t);
};

bool operator==(const Location &, const Location &);

struct Rotation
    : boost::addable<
        Rotation,
        boost::subtractable<
            Rotation,
            boost::equality_comparable<
                Rotation, boost::multipliable<Rotation, std::size_t>>>> {
public:
    using value_type = Eigen::Quaternion<double>;

private:
    value_type quat_;

public:
    Rotation();
    Rotation(double w, double x, double y, double z);

    explicit Rotation(const value_type &);

    const value_type &quaternion() const;
    void set_quaternion(const value_type &);

    double w() const;
    double x() const;
    double y() const;
    double z() const;

    std::tuple<double, double, double, double> wxyz() const;

    void set_wxyz(double, double, double, double);

    Rotation &operator+=(const Rotation &);
    Rotation &operator-=(const Rotation &);
    Rotation &operator*=(std::size_t);
};

bool operator==(const Rotation &, const Rotation &);

struct Effector;

class EffectorDifference
    : boost::addable<
        EffectorDifference,
        boost::equality_comparable<

```

(continues on next page)

(continued from previous page)

```

        EffectorDifference,
        boost::multipliable<EffectorDifference, std::size_t>>> {
private:
    compat::optional<Location> location_;
    compat::optional<Rotation> rotation_;

public:
    EffectorDifference(const Effector &, const Effector &);

    EffectorDifference() = delete;

    EffectorDifference(const EffectorDifference &) = default;
    EffectorDifference(EffectorDifference &&) = default;

    EffectorDifference &operator=(const EffectorDifference &) = default;
    EffectorDifference &operator=(EffectorDifference &&) = default;

    const compat::optional<Location> &location() const &;
    compat::optional<Location> location() &&;

    const compat::optional<Rotation> &rotation() const &;
    compat::optional<Rotation> rotation() &&;

    EffectorDifference &operator*=(std::size_t);
    EffectorDifference &operator+=(const EffectorDifference &);

    bool is_compatible(const EffectorDifference &) const;
};

bool operator==(const EffectorDifference &, const EffectorDifference &);

struct Effector : boost::addable<Effector, EffectorDifference> {
private:
    compat::optional<Location> location_;
    compat::optional<Rotation> rotation_;

public:
    Effector();
    Effector(const compat::optional<Location> &,
             const compat::optional<Rotation> &);

    const compat::optional<Location> &location() const &;
    compat::optional<Location> location() &&;

    void set_location(const compat::optional<Location> &);
    void clear_location();

    const compat::optional<Rotation> &rotation() const &;
    compat::optional<Rotation> rotation() &&;

    void set_rotation(const compat::optional<Rotation> &);
    void clear_rotation();

    Effector new_compatible_effector() const;
    bool is_compatible(const Effector &) const;
    bool is_compatible(const EffectorDifference &) const;

```

(continues on next page)

(continued from previous page)

```
Effector &operator+=(const EffectorDifference &);  
};  
  
bool operator==(const Effector &, const Effector &);  
bool operator!=(const Effector &, const Effector &);  
EffectorDifference operator-(const Effector &, const Effector &);  
  
} // namespace flom  
  
#endif
```

Includes

- Eigen/Dense
- Eigen/Geometry
- boost/operators.hpp
- flom/compat/optional.hpp (*File optional.hpp*)
- type_traits

Included By

- *File effector_type.hpp*
- *File flom.hpp*
- *File frame.hpp*
- *File interpolation.hpp*
- *File proto_util.hpp*

Namespaces

- *Namespace flom*

Classes

- *Struct Effector*
- *Struct Rotation*
- *Class EffectorDifference*
- *Class Location*

Functions

- *Function flom::operator!=(const Effector&, const Effector&)*
- *Function flom::operator-(const Effector&, const Effector&)*

- Function `flom::operator==(const Location&, const Location&)`
- Function `flom::operator==(const Rotation&, const Rotation&)`
- Function `flom::operator==(const EffectorDifference&, const EffectorDifference&)`
- Function `flom::operator==(const Effector&, const Effector&)`

File `effector_type.hpp`

Parent directory (`include/flom`)

Contents

- *Definition* (`include/flom/effector_type.hpp`)
- *Includes*
- *Included By*
- *Namespaces*
- *Classes*
- *Enums*
- *Functions*

Definition (`include/flom/effector_type.hpp`)

Program Listing for File `effector_type.hpp`

Return to documentation for file (`include/flom/effector_type.hpp`)

```
//
// Copyright 2018 coord.e
//
// This file is part of Flom.
//
// Flom is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// Flom is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
// along with Flom. If not, see <http://www.gnu.org/licenses/>.
//

#ifndef FLOM_EFFECTOR_TYPE_HPP
#define FLOM_EFFECTOR_TYPE_HPP
```

(continues on next page)

(continued from previous page)

```
#include "flom/effector.hpp"

#include <boost/operators.hpp>

#include "flom/compat/optional.hpp"

namespace flom {

enum class CoordinateSystem { World, Local };

struct EffectorType : boost::operators<EffectorType> {
private:
    compat::optional<CoordinateSystem> location_;
    compat::optional<CoordinateSystem> rotation_;

public:
    EffectorType() = delete;
    EffectorType(compat::optional<CoordinateSystem> location,
                 compat::optional<CoordinateSystem> rotation);

    compat::optional<CoordinateSystem> location() const;
    compat::optional<CoordinateSystem> rotation() const;

    void set_location(compat::optional<CoordinateSystem>);
    void clear_location();
    void set_rotation(compat::optional<CoordinateSystem>);
    void clear_rotation();

    Effector new_effector() const;
    bool is_compatible(const Effector &) const;
};

bool operator==(const EffectorType &, const EffectorType &);

} // namespace flom

#endif
```

Includes

- `boost/operators.hpp`
- `flom/compat/optional.hpp` (*File optional.hpp*)
- `flom/effector.hpp` (*File effector.hpp*)

Included By

- *File motion.hpp*

Namespaces

- *Namespace flom*

Classes

- *Struct EffectorType*

Enums

- *Enum CoordinateSystem*

Functions

- *Function flom::operator==(const EffectorType&, const EffectorType&)*

File effector_weight.hpp

Parent directory (include/flom)

Contents

- *Definition* (include/flom/effector_weight.hpp)
- *Includes*
- *Included By*
- *Namespaces*
- *Classes*
- *Functions*

Definition (include/flom/effector_weight.hpp)

Program Listing for File effector_weight.hpp

Return to documentation for file (include/flom/effector_weight.hpp)

```
//
// Copyright 2018 coord.e
//
// This file is part of Flom.
//
// Flom is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// Flom is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
```

(continues on next page)

(continued from previous page)

```
// You should have received a copy of the GNU General Public License
// along with Flom.  If not, see <http://www.gnu.org/licenses/>.
//

#ifdef FLOM_EFFECTOR_WEIGHT_HPP
#define FLOM_EFFECTOR_WEIGHT_HPP

#include <boost/operators.hpp>

namespace flom {

class EffectorWeight : boost::operators<EffectorWeight> {
private:
    double location_;
    double rotation_;

    static double validate_weight(double);
public:
    EffectorWeight() = delete;

    EffectorWeight(double location, double rotation);

    double location() const noexcept;
    double rotation() const noexcept;

    void set_location(double);
    void set_rotation(double);
};

bool operator==(const EffectorWeight &, const EffectorWeight &);

} // namespace flom

#endif
```

Includes

- `boost/operators.hpp`

Included By

- *File motion.hpp*

Namespaces

- *Namespace flom*

Classes

- *Class EffectorWeight*

Functions

- Function `flom::operator==(const EffectorWeight&, const EffectorWeight&)`

File errors.hpp

Parent directory ([include/flom](#))

Contents

- [Definition \(`include/flom/errors.hpp`\)](#)
- [Includes](#)
- [Included By](#)
- [Namespaces](#)
- [Classes](#)

Definition ([include/flom/errors.hpp](#))

Program Listing for File errors.hpp

[Return to documentation for file \(`include/flom/errors.hpp`\)](#)

```
//
// Copyright 2018 coord.e
//
// This file is part of Flom.
//
// Flom is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// Flom is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
// along with Flom. If not, see <http://www.gnu.org/licenses/>.
//

#ifdef FLOM_ERRORS_HPP
#define FLOM_ERRORS_HPP

#include <exception>
#include <string>

namespace flom::errors {

class InvalidTimeError : public std::exception {
```

(continues on next page)

(continued from previous page)

```

public:
    explicit InvalidTimeError(double);
    virtual const char *what() const noexcept;

    double time() const noexcept;

private:
    double t;
};

class OutOfFramesError : public std::exception {
public:
    explicit OutOfFramesError(double);
    virtual const char *what() const noexcept;

    double time() const noexcept;

private:
    double t;
};

class KeyframeNotFoundError : public std::exception {
public:
    explicit KeyframeNotFoundError(double);
    virtual const char *what() const noexcept;

    double time() const noexcept;

private:
    double t;
};

class ParseError : public std::exception {
public:
    // TODO: include additional information
    ParseError();
    virtual const char *what() const noexcept;
};

class SerializationError : public std::exception {
public:
    // TODO: include additional information
    SerializationError();
    virtual const char *what() const noexcept;
};

class JSONLoadError : public std::exception {
public:
    explicit JSONLoadError(const std::string &);
    virtual const char *what() const noexcept;

    std::string status_message() const noexcept;

public:
    std::string status;
};

```

(continues on next page)

(continued from previous page)

```

class JSONDumpError : public std::exception {
public:
    explicit JSONDumpError(const std::string &);
    virtual const char *what() const noexcept;

    std::string status_message() const noexcept;

public:
    std::string status;
};

class InvalidFrameError : public std::exception {
public:
    InvalidFrameError(const std::string &);
    virtual const char *what() const noexcept;

    std::string status_message() const noexcept;

public:
    std::string status;
};

class InitKeyframeError : public std::exception {
public:
    InitKeyframeError();
    virtual const char *what() const noexcept;
};

class InvalidWeightError : public std::exception {
public:
    explicit InvalidWeightError(double);
    virtual const char *what() const noexcept;

    double weight() const noexcept;

private:
    double weight_;
};

} // namespace flom::errors

#endif

```

Includes

- exception
- string

Included By

- File *flom.hpp*
- File *range.hpp*

Namespaces

- *Namespace flom::errors*

Classes

- *Class InitKeyframeError*
- *Class InvalidFrameError*
- *Class InvalidTimeError*
- *Class InvalidWeightError*
- *Class JSONDumpError*
- *Class JSONLoadError*
- *Class KeyframeNotFoundError*
- *Class OutOfFramesError*
- *Class ParseError*
- *Class SerializationError*

File flom.hpp

Parent directory (include/flom)

Contents

- *Definition* (include/flom/flom.hpp)
- *Includes*

Definition (include/flom/flom.hpp)

Program Listing for File flom.hpp

Return to documentation for file (include/flom/flom.hpp)

```
//  
// Copyright 2018 coord.e  
//  
// This file is part of Flom.  
//  
// Flom is free software: you can redistribute it and/or modify  
// it under the terms of the GNU General Public License as published by  
// the Free Software Foundation, either version 3 of the License, or  
// (at your option) any later version.  
//  
// Flom is distributed in the hope that it will be useful,  
// but WITHOUT ANY WARRANTY; without even the implied warranty of  
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
```

(continues on next page)

(continued from previous page)

```
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
// along with Flom. If not, see <http://www.gnu.org/licenses/>.
//

#ifndef FLOM_FLOM_HPP
#define FLOM_FLOM_HPP

#include "flom/constants.hpp"
#include "flom/effector.hpp"
#include "flom/errors.hpp"
#include "flom/frame.hpp"
#include "flom/interpolation.hpp"
#include "flom/motion.hpp"
#include "flom/range.hpp"

#endif
```

Includes

- flom/constants.hpp (*File constants.hpp*)
- flom/effector.hpp (*File effector.hpp*)
- flom/errors.hpp (*File errors.hpp*)
- flom/frame.hpp (*File frame.hpp*)
- flom/interpolation.hpp (*File interpolation.hpp*)
- flom/motion.hpp (*File motion.hpp*)
- flom/range.hpp (*File range.hpp*)

File frame.hpp

Parent directory (include/flom)

Contents

- *Definition* (include/flom/frame.hpp)
- *Includes*
- *Included By*
- *Namespaces*
- *Classes*
- *Functions*
- *Typedefs*

Definition (include/flom/frame.hpp)**Program Listing for File frame.hpp**

Return to documentation for file (include/flom/frame.hpp)

```
//  
// Copyright 2018 coord.e  
//  
// This file is part of Flom.  
//  
// Flom is free software: you can redistribute it and/or modify  
// it under the terms of the GNU General Public License as published by  
// the Free Software Foundation, either version 3 of the License, or  
// (at your option) any later version.  
//  
// Flom is distributed in the hope that it will be useful,  
// but WITHOUT ANY WARRANTY; without even the implied warranty of  
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
// GNU General Public License for more details.  
//  
// You should have received a copy of the GNU General Public License  
// along with Flom. If not, see <http://www.gnu.org/licenses/>.  
//  
  
#ifndef FLOM_FRAME_HPP  
#define FLOM_FRAME_HPP  
  
#include "flom/effector.hpp"  
  
#include <string>  
#include <unordered_map>  
  
#include <boost/operators.hpp>  
#include <boost/range/any_range.hpp>  
  
namespace flom {  
  
// TODO: Hide Boost.Range  
template <typename K>  
using KeyRange =  
    boost::any_range<K, boost::forward_traversal_tag,  
                    std::add_lvalue_reference_t<K>, std::ptrdiff_t>;  
  
struct Frame;  
  
class FrameDifference  
    : boost::addable<FrameDifference,  
                    boost::equality_comparable<  
                        FrameDifference,  
                        boost::multipliable<FrameDifference, std::size_t>>> {  
  
private:  
    std::unordered_map<std::string, double> positions_;  
    std::unordered_map<std::string, EffectorDifference> effectors_;  
  
public:
```

(continues on next page)

(continued from previous page)

```

FrameDifference(const Frame &, const Frame &);

FrameDifference() = delete;

FrameDifference(const FrameDifference &) = default;
FrameDifference(FrameDifference &&) = default;

FrameDifference &operator=(const FrameDifference &) = default;
FrameDifference &operator=(FrameDifference &&) = default;

const std::unordered_map<std::string, double> &positions() const &;
std::unordered_map<std::string, double> positions() &&;

const std::unordered_map<std::string, EffectorDifference> &
effectors() const &;
std::unordered_map<std::string, EffectorDifference> effectors() &&;

FrameDifference &operator*=(std::size_t);
FrameDifference &operator+=(const FrameDifference &);

bool is_compatible(const FrameDifference &) const;
};

bool operator==(const FrameDifference &, const FrameDifference &);

struct Frame : boost::addable<Frame, FrameDifference> {
private:
    using PositionsMap = std::unordered_map<std::string, double>;
    using EffectorsMap = std::unordered_map<std::string, Effector>;

    PositionsMap positions_;
    EffectorsMap effectors_;

public:
    Frame();
    Frame(const PositionsMap &, const EffectorsMap &);

    const PositionsMap &positions() const &;
    PositionsMap positions() &&;

    void set_positions(const PositionsMap &);
    void set_position(const std::string &, double);

    const EffectorsMap &effectors() const &;
    EffectorsMap effectors() &&;

    void set_effectors(const EffectorsMap &);
    void set_effector(const std::string &, const Effector &);

    KeyRange<std::string> joint_names() const;
    KeyRange<std::string> effector_names() const;

    Frame new_compatible_frame() const;
    bool is_compatible(const Frame &) const;
    bool is_compatible(const FrameDifference &) const;

    Frame &operator+=(const FrameDifference &);

```

(continues on next page)

(continued from previous page)

```
};

FrameDifference operator-(const Frame &, const Frame &);
bool operator==(const Frame &, const Frame &);
bool operator!=(const Frame &, const Frame &);

} // namespace flom

#endif
```

Includes

- `boost/operators.hpp`
- `boost/range/any_range.hpp`
- `flom/effector.hpp` (*File effector.hpp*)
- `string`
- `unordered_map`

Included By

- *File flom.hpp*
- *File interpolation.hpp*
- *File motion.hpp*
- *File motion.impl.hpp*
- *File range.hpp*

Namespaces

- *Namespace flom*

Classes

- *Struct Frame*
- *Class FrameDifference*

Functions

- *Function flom::operator!=(const Frame&, const Frame&)*
- *Function flom::operator-(const Frame&, const Frame&)*
- *Function flom::operator==(const Frame&, const Frame&)*
- *Function flom::operator==(const FrameDifference&, const FrameDifference&)*

Typedefs

- *Typedef flom::KeyRange*

File interpolation.hpp

Parent directory (include/flom)

Contents

- *Definition* (include/flom/interpolation.hpp)
- *Includes*
- *Included By*
- *Namespaces*
- *Functions*

Definition (include/flom/interpolation.hpp)

Program Listing for File interpolation.hpp

Return to documentation for file (include/flom/interpolation.hpp)

```
//
// Copyright 2018 coord.e
//
// This file is part of Flom.
//
// Flom is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// Flom is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
// along with Flom. If not, see <http://www.gnu.org/licenses/>.
//

#ifndef FLOM_INTERPOLATION_HPP
#define FLOM_INTERPOLATION_HPP

#include "flom/effector.hpp"
#include "flom/frame.hpp"

namespace flom {

template <typename T, typename U,
```

(continues on next page)

(continued from previous page)

```
        std::enable_if_t<std::is_floating_point<U>::value> * = nullptr>
T lerp(U t, T a, T b) {
    return a + t * (b - a);
}

Location interpolate(double t, Location const &a, Location const &b);
Rotation interpolate(double t, Rotation const &a, Rotation const &b);
Effector interpolate(double t, Effector const &a, Effector const &b);
Frame interpolate(double t, Frame const &a, Frame const &b);
double interpolate(double t, double a, double b);

} // namespace flom

#endif
```

Includes

- flom/effector.hpp (*File effector.hpp*)
- flom/frame.hpp (*File frame.hpp*)

Included By

- *File flom.hpp*

Namespaces

- *Namespace flom*

Functions

- *Function flom::interpolate(double, Frame const&, Frame const&)*
- *Function flom::interpolate(double, double, double)*
- *Function flom::interpolate(double, Location const&, Location const&)*
- *Function flom::interpolate(double, Rotation const&, Rotation const&)*
- *Function flom::interpolate(double, Effector const&, Effector const&)*
- *Template Function flom::lerp*

File loose_compare.hpp

Parent directory (include/flom)

Contents

- *Definition* (include/flom/loose_compare.hpp)

- *Includes*
- *Namespaces*
- *Functions*

Definition (`include/flom/loose_compare.hpp`)

Program Listing for File `loose_compare.hpp`

Return to documentation for file (`include/flom/loose_compare.hpp`)

```
//
// Copyright 2018 coord.e
//
// This file is part of Flom.
//
// Flom is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// Flom is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
// along with Flom. If not, see <http://www.gnu.org/licenses/>.
//

#ifndef FLOM_LOOSE_COMPARE_HPP
#define FLOM_LOOSE_COMPARE_HPP

#include "flom/constants.hpp"

namespace flom {

bool loose_compare(double, double);

}

#endif
```

Includes

- `flom/constants.hpp` (*File constants.hpp*)

Namespaces

- *Namespace flom*

Functions

- *Function* `flom::loose_compare`

File `motion.hpp`

Parent directory (`include/flom`)

Contents

- *Definition* (`include/flom/motion.hpp`)
- *Includes*
- *Included By*
- *Namespaces*
- *Classes*
- *Enums*
- *Functions*

Definition (`include/flom/motion.hpp`)

Program Listing for File `motion.hpp`

Return to documentation for file (`include/flom/motion.hpp`)

```
//  
// Copyright 2018 coord.e  
//  
// This file is part of Flom.  
//  
// Flom is free software: you can redistribute it and/or modify  
// it under the terms of the GNU General Public License as published by  
// the Free Software Foundation, either version 3 of the License, or  
// (at your option) any later version.  
//  
// Flom is distributed in the hope that it will be useful,  
// but WITHOUT ANY WARRANTY; without even the implied warranty of  
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
// GNU General Public License for more details.  
//  
// You should have received a copy of the GNU General Public License  
// along with Flom. If not, see <http://www.gnu.org/licenses/>.  
//  
  
#ifndef FLOM_MOTION_HPP  
#define FLOM_MOTION_HPP  
  
#include "flom/effector_type.hpp"  
#include "flom/effector_weight.hpp"  
#include "flom/frame.hpp"
```

(continues on next page)

(continued from previous page)

```

#include <iostream>
#include <memory>
#include <string>
#include <unordered_map>
#include <unordered_set>
#include <vector>

namespace flom {

enum class LoopType { None, Wrap };

class FrameRange;
class KeyframeRange;
class ConstKeyframeRange;

class Motion {
    friend bool operator==(const Motion &, const Motion &);
public:
    static Motion load(std::istream &);
    static Motion load_json(std::istream &);
    static Motion load_json_string(std::string const &);

    Motion(const std::unordered_set<std::string> &joint_names,
           const std::unordered_map<std::string, EffectorType> &effector_types,
           const std::string &model = "");

    Motion(Motion const &);
    ~Motion();

    bool is_valid() const;
    bool is_valid_frame(const Frame &) const;

    Frame frame_at(double t) const;

    FrameRange frames(double fps) const;

    bool is_in_range_at(double t) const;

    void dump(std::ostream &) const;
    void dump_json(std::ostream &) const;
    std::string dump_json_string() const;

    LoopType loop() const;
    void set_loop(LoopType);

    std::string model_id() const;
    void set_model_id(std::string const &);

    Frame new_keyframe() const;
    void insert_keyframe(double t, const Frame &);
    void delete_keyframe(double t, bool loose = true);
    KeyframeRange keyframes();
    ConstKeyframeRange keyframes() const;
    ConstKeyframeRange const_keyframes() const;
    void clear_keyframes();

```

(continues on next page)

(continued from previous page)

```
EffectorType effector_type(const std::string &) const;

EffectorWeight effector_weight(const std::string &) const;
void set_effector_weight(const std::string &, EffectorWeight);

double length() const;

KeyRange<std::string> joint_names() const;
KeyRange<std::string> effector_names() const;

private:
    class Impl;
    std::unique_ptr<Impl> impl;
};

bool operator==(const Motion &, const Motion &);
bool operator!=(const Motion &, const Motion &);

} // namespace flom

#endif
```

Includes

- flom/effector_type.hpp (*File effector_type.hpp*)
- flom/effector_weight.hpp (*File effector_weight.hpp*)
- flom/frame.hpp (*File frame.hpp*)
- iostream
- memory
- string
- unordered_map
- unordered_set
- vector

Included By

- *File flom.hpp*
- *File motion.impl.hpp*
- *File proto_util.hpp*
- *File range.hpp*

Namespaces

- *Namespace flom*

Classes

- *Class Motion*

Enums

- *Enum LoopType*

Functions

- *Function flom::operator!=(const Motion&, const Motion&)*
- *Function flom::operator==(const Motion&, const Motion&)*

File motion.impl.hpp

Parent directory (include/flom)

Contents

- *Definition* (include/flom/motion.impl.hpp)
- *Includes*
- *Namespaces*
- *Classes*
- *Functions*

Definition (include/flom/motion.impl.hpp)

Program Listing for File motion.impl.hpp

Return to documentation for file (include/flom/motion.impl.hpp)

```
//
// Copyright 2018 coord.e
//
// This file is part of Flom.
//
// Flom is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// Flom is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
```

(continues on next page)

(continued from previous page)

```

// You should have received a copy of the GNU General Public License
// along with Flom.  If not, see <http://www.gnu.org/licenses/>.
//

#ifndef FLOM_MOTION_IMPL_HPP
#define FLOM_MOTION_IMPL_HPP

#include "flom/frame.hpp"
#include "flom/motion.hpp"

#include "motion.pb.h"

#include <functional>
#include <map>
#include <numeric>
#include <string>
#include <unordered_map>
#include <unordered_set>

namespace flom {

template <typename K> std::size_t names_hash(const std::unordered_set<K> &s) {
    auto h{s.hash_function()};
    return std::accumulate(std::cbegin(s), std::cend(s),
                           static_cast<std::size_t>(0),
                           [&h](auto r, const auto &p) { return r ^ h(p); });
}

template <typename K, typename V>
std::size_t names_hash(const std::unordered_map<K, V> &m) {
    auto h{m.hash_function()};
    return std::accumulate(
        std::cbegin(m), std::cend(m), static_cast<std::size_t>(0),
        [&h](auto r, const auto &p) { return r ^ h(p.first); });
}

class Motion::Impl {
public:
    std::string model_id;
    LoopType loop;
    std::map<double, Frame> raw_frames;

    // keys of these two member must not be changed after construction
    const std::unordered_set<std::string> joint_names;
    const std::unordered_map<std::string, EffectorType> effector_types;
    std::unordered_map<std::string, EffectorWeight> effector_weights;

    // Hash of joint_names
    const std::size_t joints_hash;
    // Hash of keys of effector_types
    const std::size_t effectors_hash;

    Impl(const std::unordered_set<std::string> &joints,
         const std::unordered_map<std::string, EffectorType> &effectors,
         const std::string &model = "")
        : model_id(model), loop(LoopType::None), raw_frames(),
          joint_names(joints), effector_types(effectors),

```

(continues on next page)

(continued from previous page)

```

        joints_hash(names_hash(joints)), effectors_hash(names_hash(effectors)) {
    this->effector_weights.reserve(effectors.size());
    for (const auto &[name, e] : effectors) {
        this->effector_weights.emplace(name, EffectorWeight{0.0, 0.0});
    }
    this->add_initial_frame();
}

void add_initial_frame();
Frame new_keyframe() const noexcept;

static Motion from_protobuf(proto::Motion const &);
proto::Motion to_protobuf() const;

bool is_valid() const;
bool is_valid_frame(const Frame &) const;
};

} // namespace flom

#endif

```

Includes

- flom/frame.hpp (*File frame.hpp*)
- flom/motion.hpp (*File motion.hpp*)
- functional
- map
- motion.pb.h
- numeric
- string
- unordered_map
- unordered_set

Namespaces

- *Namespace flom*

Classes

- *Class Motion::Impl*

Functions

- *Template Function flom::names_hash(const std::unordered_set<K>&)*

- *Template Function* `flom::names_hash(const std::unordered_map<K, V>&)`

File optional.hpp

Parent directory (`include/flom/compat`)

Contents

- *Definition* (`include/flom/compat/optional.hpp`)
- *Includes*
- *Included By*
- *Namespaces*
- *Typedefs*
- *Variables*

Definition (`include/flom/compat/optional.hpp`)

Program Listing for File optional.hpp

Return to documentation for file (`include/flom/compat/optional.hpp`)

```
//  
// Copyright 2018 coord.e  
//  
// This file is part of Flom.  
//  
// Flom is free software: you can redistribute it and/or modify  
// it under the terms of the GNU General Public License as published by  
// the Free Software Foundation, either version 3 of the License, or  
// (at your option) any later version.  
//  
// Flom is distributed in the hope that it will be useful,  
// but WITHOUT ANY WARRANTY; without even the implied warranty of  
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
// GNU General Public License for more details.  
//  
// You should have received a copy of the GNU General Public License  
// along with Flom. If not, see <http://www.gnu.org/licenses/>.  
//  
  
#ifndef FLOM_COMPAT_OPTIONAL_HPP  
#define FLOM_COMPAT_OPTIONAL_HPP  
  
#include <cstddef>  
  
#include <boost/optional.hpp>  
  
namespace flom::compat {  
template<typename T>  
using optional = boost::optional<T>;
```

(continues on next page)

(continued from previous page)

```
static inline auto nullopt = boost::none;
}

#endif
```

Includes

- `boost/optional.hpp`
- `cstdint`

Included By

- *File `effector.hpp`*
- *File `effector_type.hpp`*
- *File `proto_util.hpp`*

Namespaces

- *Namespace `flom::compat`*

Typedefs

- *Typedef `flom::compat::optional`*

Variables

- *Variable `flom::compat::nullopt`*

File `proto_util.hpp`

Parent directory (`include/flom`)

Contents

- *Definition* (`include/flom/proto_util.hpp`)
- *Includes*
- *Namespaces*
- *Functions*

Definition (include/flom/proto_util.hpp)**Program Listing for File proto_util.hpp**

Return to documentation for file (include/flom/proto_util.hpp)

```
//  
// Copyright 2018 coord.e  
//  
// This file is part of Flom.  
//  
// Flom is free software: you can redistribute it and/or modify  
// it under the terms of the GNU General Public License as published by  
// the Free Software Foundation, either version 3 of the License, or  
// (at your option) any later version.  
//  
// Flom is distributed in the hope that it will be useful,  
// but WITHOUT ANY WARRANTY; without even the implied warranty of  
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
// GNU General Public License for more details.  
//  
// You should have received a copy of the GNU General Public License  
// along with Flom. If not, see <http://www.gnu.org/licenses/>.  
//  
  
#ifndef FLOM_PROTO_UTIL_HPP  
#define FLOM_PROTO_UTIL_HPP  
  
#include "flom/effector.hpp"  
#include "flom/motion.hpp"  
  
#include "frame.pb.h"  
#include "location.pb.h"  
#include "motion.pb.h"  
#include "rotation.pb.h"  
  
#include "flom/compat/optional.hpp"  
  
namespace flom::proto_util {  
  
void pack_vec3(Location::value_type const &, proto::Vec3 *);  
void pack_location(Location const &, proto::Location *);  
  
Location::value_type unpack_vec3(proto::Vec3 const &);  
Location unpack_location(proto::Location const &);  
  
void pack_quat(Rotation::value_type const &, proto::Quaternion *);  
void pack_rotation(Rotation const &, proto::Rotation *);  
  
Rotation::value_type unpack_quat(proto::Quaternion const &);  
Rotation unpack_rotation(proto::Rotation const &);  
  
void pack_effector_type(EffectorType const &, proto::EffectorType *);  
proto::EffectorType::Type  
pack_coord_system(compat::optional<CoordinateSystem> const &);  
  
EffectorType unpack_effector_type(proto::EffectorType const &);
```

(continues on next page)

(continued from previous page)

```

compat::optional<CoordinateSystem>
unpack_coord_system(proto::EffectorType::Type const &);

void pack_effector_weight(EffectorWeight const &, proto::EffectorWeight *);
EffectorWeight unpack_effector_weight(proto::EffectorWeight const &);

} // namespace flom::proto_util

#endif

```

Includes

- flom/compat/optional.hpp (*File optional.hpp*)
- flom/effector.hpp (*File effector.hpp*)
- flom/motion.hpp (*File motion.hpp*)
- frame.pb.h
- location.pb.h
- motion.pb.h
- rotation.pb.h

Namespaces

- *Namespace flom::proto_util*

Functions

- *Function flom::proto_util::pack_coord_system*
- *Function flom::proto_util::pack_effector_type*
- *Function flom::proto_util::pack_effector_weight*
- *Function flom::proto_util::pack_location*
- *Function flom::proto_util::pack_quat*
- *Function flom::proto_util::pack_rotation*
- *Function flom::proto_util::pack_vec3*
- *Function flom::proto_util::unpack_coord_system*
- *Function flom::proto_util::unpack_effector_type*
- *Function flom::proto_util::unpack_effector_weight*
- *Function flom::proto_util::unpack_location*
- *Function flom::proto_util::unpack_quat*
- *Function flom::proto_util::unpack_rotation*
- *Function flom::proto_util::unpack_vec3*

File range.hpp

Parent directory ([include/flom](#))

Contents

- *Definition* ([include/flom/range.hpp](#))
- *Includes*
- *Included By*
- *Namespaces*
- *Classes*
- *Functions*

Definition ([include/flom/range.hpp](#))

Program Listing for File range.hpp

Return to documentation for file ([include/flom/range.hpp](#))

```
//  
// Copyright 2018 coord.e  
//  
// This file is part of Flom.  
//  
// Flom is free software: you can redistribute it and/or modify  
// it under the terms of the GNU General Public License as published by  
// the Free Software Foundation, either version 3 of the License, or  
// (at your option) any later version.  
//  
// Flom is distributed in the hope that it will be useful,  
// but WITHOUT ANY WARRANTY; without even the implied warranty of  
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
// GNU General Public License for more details.  
//  
// You should have received a copy of the GNU General Public License  
// along with Flom. If not, see <http://www.gnu.org/licenses/>.  
//  
  
#ifndef FLOM_RANGE_HPP  
#define FLOM_RANGE_HPP  
  
#include "flom/errors.hpp"  
#include "flom/frame.hpp"  
#include "flom/motion.hpp"  
  
#include <iterator>  
#include <map>  
#include <memory>  
#include <utility>  
  
namespace flom {
```

(continues on next page)

(continued from previous page)

```

// using snake_case, following customs of iterator naming
class frame_iterator {
    friend bool operator==(const frame_iterator &,
                           const frame_iterator &) noexcept;

public:
    using iterator_category = std::input_iterator_tag;
    using value_type = std::pair<double, Frame>;
    using difference_type = double;
    using pointer = Frame *;
    using reference = Frame &;

private:
    class Impl;
    std::unique_ptr<Impl> impl;

    bool is_end = false;

public:
    frame_iterator() noexcept;
    frame_iterator(Motion const &motion, double fps) noexcept;

    frame_iterator(const frame_iterator &);
    frame_iterator(frame_iterator &&);
    frame_iterator &operator=(const frame_iterator &);
    frame_iterator &operator=(frame_iterator &&);

    ~frame_iterator();

    // This is InputIterator because operator* doesn't return reference
    value_type operator*() const;

    frame_iterator &operator++() noexcept;
    frame_iterator operator++(int) noexcept;

    frame_iterator &operator--() noexcept;
    frame_iterator operator--(int) noexcept;

    double current_time() const noexcept;
};

frame_iterator::difference_type operator-(const frame_iterator &,
                                          const frame_iterator &) noexcept;
bool operator==(const frame_iterator &, const frame_iterator &) noexcept;
bool operator!=(const frame_iterator &, const frame_iterator &) noexcept;

class FrameRange {
public:
    using value_type = Frame;
    using iterator = frame_iterator;

private:
    Motion const &motion;
    double fps;

public:

```

(continues on next page)

(continued from previous page)

```

FrameRange() = delete;
FrameRange(Motion const &motion_, double fps_) : motion(motion_), fps(fps_) {}
FrameRange(const FrameRange &) = default;
FrameRange(FrameRange &&) = default;
FrameRange &operator=(const FrameRange &) = default;
FrameRange &operator=(FrameRange &&) = default;

iterator begin() const noexcept { return {this->motion, this->fps}; }
iterator end() const noexcept { return {}; }

iterator cbegin() const noexcept { return this->begin(); }
iterator cend() const noexcept { return this->end(); }
};

class CheckedFrameRef {
public:
    using reference_type = Frame &;

    CheckedFrameRef(reference_type value_, const Motion *motion_)
        : value(value_), motion(motion_) {}

    CheckedFrameRef &operator=(const Frame &frame) & {
        if (!this->motion->is_valid_frame(frame)) {
            throw errors::InvalidFrameError{"in CheckedFrameWrapper"};
        }
        this->value = frame;
        return *this;
    }

    operator reference_type() const noexcept { return this->value; }

private:
    reference_type value;
    const Motion *motion;
};

class keyframe_iterator {
public:
    using base_iterator = std::map<double, Frame>::iterator;

    using iterator_category = std::bidirectional_iterator_tag;
    using value_type = std::iterator_traits<base_iterator>::value_type;
    using difference_type = std::iterator_traits<base_iterator>::difference_type;
    using pointer = std::iterator_traits<base_iterator>::pointer;
    using reference = std::iterator_traits<base_iterator>::reference;

    using checked_value_type = std::pair<const double, CheckedFrameRef>;

private:
    friend bool operator==(const keyframe_iterator &,
                          const keyframe_iterator &) noexcept;

    base_iterator it;
    const Motion *motion;

public:
    keyframe_iterator() noexcept : it(), motion() {}

```

(continues on next page)

(continued from previous page)

```

explicit keyframe_iterator(base_iterator it_, const Motion &motion_) noexcept
    : it(it_), motion(&motion_) {}

keyframe_iterator(const keyframe_iterator &) = default;
keyframe_iterator(keyframe_iterator &&) = default;
keyframe_iterator &operator=(const keyframe_iterator &) = default;
keyframe_iterator &operator=(keyframe_iterator &&) = default;

const value_type &operator*() const;
checked_value_type operator*();

const value_type &operator->() const;
checked_value_type operator->();

keyframe_iterator &operator++() noexcept;
keyframe_iterator operator++(int) noexcept;

keyframe_iterator &operator--() noexcept;
keyframe_iterator operator--(int) noexcept;
};

bool operator==(const keyframe_iterator &, const keyframe_iterator &) noexcept;
bool operator!=(const keyframe_iterator &, const keyframe_iterator &) noexcept;

class KeyframeRange {
public:
    using value_type = Frame;
    using iterator = keyframe_iterator;
    using base_iterator = typename std::map<double, Frame>::iterator;

private:
    base_iterator begin_it;
    base_iterator end_it;
    const Motion &motion;

public:
    KeyframeRange() = delete;
    KeyframeRange(base_iterator begin_, base_iterator end_, const Motion &motion_)
        : begin_it(begin_), end_it(end_), motion(motion_) {}
    KeyframeRange(const KeyframeRange &) = default;
    KeyframeRange(KeyframeRange &&) = default;
    KeyframeRange &operator=(const KeyframeRange &) = default;
    KeyframeRange &operator=(KeyframeRange &&) = default;

    iterator begin() noexcept { return iterator{this->begin_it, this->motion}; }
    iterator end() noexcept { return iterator{this->end_it, this->motion}; }
};

class ConstKeyframeRange {
public:
    using value_type = Frame;
    using const_iterator = typename std::map<double, Frame>::const_iterator;

private:
    const_iterator begin_;
    const_iterator end_;

```

(continues on next page)

(continued from previous page)

```
public:
    ConstKeyframeRange() = delete;
    ConstKeyframeRange(const_iterator begin, const_iterator end)
        : begin_(begin), end_(end) {}
    ConstKeyframeRange(const ConstKeyframeRange &) = default;
    ConstKeyframeRange(ConstKeyframeRange &&) = default;
    ConstKeyframeRange &operator=(const ConstKeyframeRange &) = default;
    ConstKeyframeRange &operator=(ConstKeyframeRange &&) = default;

    const_iterator begin() const noexcept { return this->begin_; }
    const_iterator end() const noexcept { return this->end_; }

    const_iterator cbegin() const noexcept { return this->begin_; }
    const_iterator cend() const noexcept { return this->end_; }
};
} // namespace flom

#endif
```

Includes

- flom/errors.hpp (*File errors.hpp*)
- flom/frame.hpp (*File frame.hpp*)
- flom/motion.hpp (*File motion.hpp*)
- iterator
- map
- memory
- utility

Included By

- *File flom.hpp*

Namespaces

- *Namespace flom*

Classes

- *Class CheckedFrameRef*
- *Class ConstKeyframeRange*
- *Class frame_iterator*
- *Class FrameRange*
- *Class keyframe_iterator*

- Class *KeyframeRange*

Functions

- Function *flom::operator!=(const frame_iterator&, const frame_iterator&)*
- Function *flom::operator!=(const keyframe_iterator&, const keyframe_iterator&)*
- Function *flom::operator-(const frame_iterator&, const frame_iterator&)*
- Function *flom::operator==(const keyframe_iterator&, const keyframe_iterator&)*
- Function *flom::operator==(const frame_iterator&, const frame_iterator&)*

File range.impl.hpp

Parent directory ([include/flom](#))

Contents

- Definition ([include/flom/range.impl.hpp](#))
- Namespaces
- Classes

Definition ([include/flom/range.impl.hpp](#))

Program Listing for File range.impl.hpp

[Return to documentation for file](#) ([include/flom/range.impl.hpp](#))

```
//
// Copyright 2018 coord.e
//
// This file is part of Flom.
//
// Flom is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// Flom is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
// along with Flom. If not, see <http://www.gnu.org/licenses/>.
//

#ifdef FLOM_RANGE_IMPL_HPP
#define FLOM_RANGE_IMPL_HPP
```

(continues on next page)

(continued from previous page)

```
namespace flom {  
  
class frame_iterator::Impl {  
public:  
    const Motion *motion;  
    double fps;  
    long t_index = 0;  
  
    Impl(const Motion &motion_, double fps_) : motion(&motion_), fps(fps_) {}  
  
    double current_time() const noexcept;  
    bool check_is_end() const noexcept;  
};  
  
} // namespace flom  
  
#endif
```

Namespaces

- *Namespace flom*

Classes

- *Class frame_iterator::Impl*

F

- flom::CheckedFrameRef (C++ class), 19
- flom::CheckedFrameRef::CheckedFrameRef (C++ function), 19
- flom::CheckedFrameRef::operator reference_type (C++ function), 19
- flom::CheckedFrameRef::operator= (C++ function), 19
- flom::CheckedFrameRef::reference_type (C++ type), 19
- flom::compat::nullopt (C++ member), 44
- flom::constants::float_point_tolerance (C++ member), 44
- flom::constants::pi (C++ member), 44
- flom::ConstKeyframeRange (C++ class), 19
- flom::ConstKeyframeRange::begin (C++ function), 19
- flom::ConstKeyframeRange::cbegin (C++ function), 19
- flom::ConstKeyframeRange::cend (C++ function), 19
- flom::ConstKeyframeRange::const_iterator (C++ type), 19
- flom::ConstKeyframeRange::ConstKeyframeRange (C++ function), 19
- flom::ConstKeyframeRange::end (C++ function), 19
- flom::ConstKeyframeRange::operator= (C++ function), 19
- flom::ConstKeyframeRange::value_type (C++ type), 19
- flom::CoordinateSystem (C++ type), 36
- flom::Effector (C++ class), 15
- flom::Effector::clear_location (C++ function), 16
- flom::Effector::clear_rotation (C++ function), 16
- flom::Effector::Effector (C++ function), 15
- flom::Effector::is_compatible (C++ function), 16
- flom::Effector::location (C++ function), 15
- flom::Effector::new_compatible_effector (C++ function), 16
- flom::Effector::operator+= (C++ function), 16
- flom::Effector::rotation (C++ function), 16
- flom::Effector::set_location (C++ function), 15
- flom::Effector::set_rotation (C++ function), 16
- flom::EffectorDifference (C++ class), 20
- flom::EffectorDifference::EffectorDifference (C++ function), 20
- flom::EffectorDifference::is_compatible (C++ function), 20
- flom::EffectorDifference::location (C++ function), 20
- flom::EffectorDifference::operator*= (C++ function), 20
- flom::EffectorDifference::operator+= (C++ function), 20
- flom::EffectorDifference::operator= (C++ function), 20
- flom::EffectorDifference::rotation (C++ function), 20
- flom::EffectorType (C++ class), 16
- flom::EffectorType::clear_location (C++ function), 16
- flom::EffectorType::clear_rotation (C++ function), 16
- flom::EffectorType::EffectorType (C++ function), 16
- flom::EffectorType::is_compatible (C++ function), 16
- flom::EffectorType::location (C++ function), 16
- flom::EffectorType::new_effector (C++ function), 16
- flom::EffectorType::rotation (C++ function), 16
- flom::EffectorType::set_location (C++ function), 16
- flom::EffectorType::set_rotation (C++ function), 16
- flom::EffectorWeight (C++ class), 21
- flom::EffectorWeight::EffectorWeight (C++ function), 21
- flom::EffectorWeight::location (C++ function), 21
- flom::EffectorWeight::rotation (C++ function), 21
- flom::EffectorWeight::set_location (C++ function), 21
- flom::EffectorWeight::set_rotation (C++ function), 21
- flom::errors::InitKeyframeError (C++ class), 21
- flom::errors::InitKeyframeError::InitKeyframeError (C++ function), 21
- flom::errors::InitKeyframeError::what (C++ function), 21
- flom::errors::InvalidFrameError (C++ class), 22
- flom::errors::InvalidFrameError::InvalidFrameError (C++ function), 22
- flom::errors::InvalidFrameError::status (C++ member), 22
- flom::errors::InvalidFrameError::status_message (C++ function), 22
- flom::errors::InvalidFrameError::what (C++ function), 22
- flom::errors::InvalidTimeError (C++ class), 22
- flom::errors::InvalidTimeError::InvalidTimeError (C++ function), 22
- flom::errors::InvalidTimeError::time (C++ function), 22
- flom::errors::InvalidTimeError::what (C++ function), 22

flom::errors::InvalidWeightError (C++ class), 23
 flom::errors::InvalidWeightError::InvalidWeightError (C++ function), 23
 flom::errors::InvalidWeightError::weight (C++ function), 23
 flom::errors::InvalidWeightError::what (C++ function), 23
 flom::errors::JSONDumpError (C++ class), 23
 flom::errors::JSONDumpError::JSONDumpError (C++ function), 23
 flom::errors::JSONDumpError::status (C++ member), 24
 flom::errors::JSONDumpError::status_message (C++ function), 23
 flom::errors::JSONDumpError::what (C++ function), 23
 flom::errors::JSONLoadError (C++ class), 24
 flom::errors::JSONLoadError::JSONLoadError (C++ function), 24
 flom::errors::JSONLoadError::status (C++ member), 24
 flom::errors::JSONLoadError::status_message (C++ function), 24
 flom::errors::JSONLoadError::what (C++ function), 24
 flom::errors::KeyframeNotFoundError (C++ class), 24
 flom::errors::KeyframeNotFoundError::KeyframeNotFoundError (C++ function), 25
 flom::errors::KeyframeNotFoundError::time (C++ function), 25
 flom::errors::KeyframeNotFoundError::what (C++ function), 25
 flom::errors::OutOfFramesError (C++ class), 25
 flom::errors::OutOfFramesError::OutOfFramesError (C++ function), 25
 flom::errors::OutOfFramesError::time (C++ function), 25
 flom::errors::OutOfFramesError::what (C++ function), 25
 flom::errors::ParseError (C++ class), 25
 flom::errors::ParseError::ParseError (C++ function), 26
 flom::errors::ParseError::what (C++ function), 26
 flom::errors::SerializationError (C++ class), 26
 flom::errors::SerializationError::SerializationError (C++ function), 26
 flom::errors::SerializationError::what (C++ function), 26
 flom::Frame (C++ class), 17
 flom::Frame::effector_names (C++ function), 17
 flom::Frame::effectors (C++ function), 17
 flom::Frame::Frame (C++ function), 17
 flom::Frame::is_compatible (C++ function), 17
 flom::Frame::joint_names (C++ function), 17
 flom::Frame::new_compatible_frame (C++ function), 17
 flom::Frame::operator+= (C++ function), 17
 flom::Frame::positions (C++ function), 17
 flom::Frame::set_effector (C++ function), 17
 flom::Frame::set_effectors (C++ function), 17
 flom::Frame::set_position (C++ function), 17
 flom::Frame::set_positions (C++ function), 17
 flom::frame_iterator (C++ class), 26
 flom::frame_iterator::~~frame_iterator (C++ function), 27
 flom::frame_iterator::current_time (C++ function), 27
 flom::frame_iterator::difference_type (C++ type), 27
 flom::frame_iterator::frame_iterator (C++ function), 27
 flom::frame_iterator::Impl (C++ class), 27, 28
 flom::frame_iterator::Impl::check_is_end (C++ function), 27, 28
 flom::frame_iterator::Impl::current_time (C++ function), 27, 28
 flom::frame_iterator::Impl::fps (C++ member), 27, 28
 flom::frame_iterator::Impl::Impl (C++ function), 27, 28
 flom::frame_iterator::Impl::motion (C++ member), 27, 28
 flom::frame_iterator::Impl::t_index (C++ member), 27, 28
 flom::frame_iterator::iterator_category (C++ type), 27
 flom::frame_iterator::operator* (C++ function), 27
 flom::frame_iterator::operator++ (C++ function), 27
 flom::frame_iterator::operator- (C++ function), 27
 flom::frame_iterator::operator= (C++ function), 27
 flom::frame_iterator::pointer (C++ type), 27
 flom::frame_iterator::reference (C++ type), 27
 flom::frame_iterator::value_type (C++ type), 27
 flom::FrameDifference (C++ class), 28
 flom::FrameDifference::effectors (C++ function), 29
 flom::FrameDifference::FrameDifference (C++ function), 29
 flom::FrameDifference::is_compatible (C++ function), 29
 flom::FrameDifference::operator*= (C++ function), 29
 flom::FrameDifference::operator+= (C++ function), 29
 flom::FrameDifference::operator= (C++ function), 29
 flom::FrameDifference::positions (C++ function), 29
 flom::FrameRange (C++ class), 29
 flom::FrameRange::begin (C++ function), 30
 flom::FrameRange::cbegin (C++ function), 30
 flom::FrameRange::cend (C++ function), 30
 flom::FrameRange::end (C++ function), 30
 flom::FrameRange::FrameRange (C++ function), 29
 flom::FrameRange::iterator (C++ type), 29
 flom::FrameRange::operator= (C++ function), 29
 flom::FrameRange::value_type (C++ type), 29
 flom::interpolate (C++ function), 36, 37
 flom::keyframe_iterator (C++ class), 30
 flom::keyframe_iterator::base_iterator (C++ type), 30
 flom::keyframe_iterator::checked_value_type (C++ type), 30
 flom::keyframe_iterator::difference_type (C++ type), 30
 flom::keyframe_iterator::iterator_category (C++ type), 30
 flom::keyframe_iterator::keyframe_iterator (C++ function), 30
 flom::keyframe_iterator::operator* (C++ function), 30
 flom::keyframe_iterator::operator++ (C++ function), 30, 31
 flom::keyframe_iterator::operator- (C++ function), 31

flom::keyframe_iterator::operator-> (C++ function), 30
 flom::keyframe_iterator::operator= (C++ function), 30
 flom::keyframe_iterator::pointer (C++ type), 30
 flom::keyframe_iterator::reference (C++ type), 30
 flom::keyframe_iterator::value_type (C++ type), 30
 flom::KeyframeRange (C++ class), 31
 flom::KeyframeRange::base_iterator (C++ type), 31
 flom::KeyframeRange::begin (C++ function), 31
 flom::KeyframeRange::end (C++ function), 31
 flom::KeyframeRange::iterator (C++ type), 31
 flom::KeyframeRange::KeyframeRange (C++ function), 31
 flom::KeyframeRange::operator= (C++ function), 31
 flom::KeyframeRange::value_type (C++ type), 31
 flom::lerp (C++ function), 37
 flom::Local (C++ enumerator), 36
 flom::Location (C++ class), 32
 flom::Location::Location (C++ function), 32
 flom::Location::operator*= (C++ function), 32
 flom::Location::operator+= (C++ function), 32
 flom::Location::operator-= (C++ function), 32
 flom::Location::set_vector (C++ function), 32
 flom::Location::set_x (C++ function), 32
 flom::Location::set_xyz (C++ function), 32
 flom::Location::set_y (C++ function), 32
 flom::Location::set_z (C++ function), 32
 flom::Location::value_type (C++ type), 32
 flom::Location::vector (C++ function), 32
 flom::Location::x (C++ function), 32
 flom::Location::xyz (C++ function), 32
 flom::Location::y (C++ function), 32
 flom::Location::z (C++ function), 32
 flom::LoopType (C++ type), 36
 flom::loose_compare (C++ function), 37
 flom::Motion (C++ class), 33
 flom::Motion::~~Motion (C++ function), 33
 flom::Motion::clear_keyframes (C++ function), 33
 flom::Motion::const_keyframes (C++ function), 33
 flom::Motion::delete_keyframe (C++ function), 33
 flom::Motion::dump (C++ function), 33
 flom::Motion::dump_json (C++ function), 33
 flom::Motion::dump_json_string (C++ function), 33
 flom::Motion::effector_names (C++ function), 34
 flom::Motion::effector_type (C++ function), 33
 flom::Motion::effector_weight (C++ function), 33
 flom::Motion::frame_at (C++ function), 33
 flom::Motion::frames (C++ function), 33
 flom::Motion::Impl (C++ class), 34, 35
 flom::Motion::Impl::add_initial_frame (C++ function), 34, 35
 flom::Motion::Impl::effector_types (C++ member), 34, 35
 flom::Motion::Impl::effector_weights (C++ member), 34, 35
 flom::Motion::Impl::effectors_hash (C++ member), 34, 35
 flom::Motion::Impl::from_protobuf (C++ function), 34, 35
 flom::Motion::Impl::Impl (C++ function), 34, 35
 flom::Motion::Impl::is_valid (C++ function), 34, 35
 flom::Motion::Impl::is_valid_frame (C++ function), 34, 35
 flom::Motion::Impl::joint_names (C++ member), 34, 35
 flom::Motion::Impl::joints_hash (C++ member), 34, 35
 flom::Motion::Impl::loop (C++ member), 34, 35
 flom::Motion::Impl::model_id (C++ member), 34, 35
 flom::Motion::Impl::new_keyframe (C++ function), 34, 35
 flom::Motion::Impl::raw_frames (C++ member), 34, 35
 flom::Motion::Impl::to_protobuf (C++ function), 34, 35
 flom::Motion::insert_keyframe (C++ function), 33
 flom::Motion::is_in_range_at (C++ function), 33
 flom::Motion::is_valid (C++ function), 33
 flom::Motion::is_valid_frame (C++ function), 33
 flom::Motion::joint_names (C++ function), 33
 flom::Motion::keyframes (C++ function), 33
 flom::Motion::length (C++ function), 33
 flom::Motion::load (C++ function), 34
 flom::Motion::load_json (C++ function), 34
 flom::Motion::load_json_string (C++ function), 34
 flom::Motion::loop (C++ function), 33
 flom::Motion::model_id (C++ function), 33
 flom::Motion::Motion (C++ function), 33
 flom::Motion::new_keyframe (C++ function), 33
 flom::Motion::set_effector_weight (C++ function), 33
 flom::Motion::set_loop (C++ function), 33
 flom::Motion::set_model_id (C++ function), 33
 flom::names_hash (C++ function), 37, 38
 flom::None (C++ enumerator), 36
 flom::operator!= (C++ function), 38
 flom::operator- (C++ function), 39
 flom::operator== (C++ function), 39–41
 flom::proto_util::pack_coord_system (C++ function), 41
 flom::proto_util::pack_effector_type (C++ function), 41
 flom::proto_util::pack_effector_weight (C++ function), 42
 flom::proto_util::pack_location (C++ function), 42
 flom::proto_util::pack_quat (C++ function), 42
 flom::proto_util::pack_rotation (C++ function), 42
 flom::proto_util::pack_vec3 (C++ function), 42
 flom::proto_util::unpack_coord_system (C++ function), 42
 flom::proto_util::unpack_effector_type (C++ function), 43
 flom::proto_util::unpack_effector_weight (C++ function), 43
 flom::proto_util::unpack_location (C++ function), 43
 flom::proto_util::unpack_quat (C++ function), 43

flom::proto_util::unpack_rotation (C++ function), [43](#)
flom::proto_util::unpack_vec3 (C++ function), [44](#)
flom::Rotation (C++ class), [18](#)
flom::Rotation::operator*= (C++ function), [18](#)
flom::Rotation::operator+= (C++ function), [18](#)
flom::Rotation::operator-= (C++ function), [18](#)
flom::Rotation::quaternion (C++ function), [18](#)
flom::Rotation::Rotation (C++ function), [18](#)
flom::Rotation::set_quaternion (C++ function), [18](#)
flom::Rotation::set_wxyz (C++ function), [18](#)
flom::Rotation::value_type (C++ type), [18](#)
flom::Rotation::w (C++ function), [18](#)
flom::Rotation::wxyz (C++ function), [18](#)
flom::Rotation::x (C++ function), [18](#)
flom::Rotation::y (C++ function), [18](#)
flom::Rotation::z (C++ function), [18](#)
flom::World (C++ enumerator), [36](#)
flom::Wrap (C++ enumerator), [36](#)