Plan 2D

BIM++

Mar 18, 2021

TABLE OF CONTENTS:

- Document
- Repository

This project is for plan in 2D. And it is inspired by IFC.

CHAPTER

ONE

FEATURES

- Header-Only
- **Define some data structure for the 2D plan**
  - Node
  - Wall
  - Hole - Door or Window or Hole
  - Room
- **Implement some algorithms for the 2D plan**
  - Compute the edges of the room by some dispersed walls

1.1 Data structures

1.1.1 Basic

template<typename **T** = double>
class point
Define a point for the 2D plan, it has two dimensions.

template<typename **TPrecision** = double, typename **TPoint** = *point*<*TPrecision*>>
class bimpp::plan2d::**constant**
Define some classes and declare some constant values

Public Types

typedef *TPoint* **point_type**
Define the point type.

Public Static Functions

static inline bool **isValid**(id_type *_id*)
Is id valid?

Public Static Attributes

static const *point_type* **zero_point**
Origin point.

static const *point_type* **unit_point**
Unit point.

static const id_type **none_id**
Invalid id.

1.1.2 Node

template<typename **TConstant** = *constant*<>>
class bimpp::plan2d::**node**
A point or a joint in the plan.

A node represents a point or a joint with two walls in the 2D plan

Public Functions

inline node(precision_type *_x* = 0, precision_type *_y* = 0)
A constructor by a 2D coordinate

Parameters

- _x: The value in x-axis, default is 0.
- _y: The value in y-axis, default is 0.

inline node(**const** point_type &*_point* = *TConstant*::zero_point)
A constructor by a 2D point

Parameters

- _point: A 2D point, default is origin point.

1.1.3 Wall

template<typename **TConstant** = *constant*<>>
class bimpp::plan2d::**wall**
A wall represents a wall in the 2D plan

Public Members

std::string **kind**
The kind of wall.

id_type **start_node_id**
The start of wall.

id_type **end_node_id**
The end of wall.

precision_type **thickness**
The thickness of wall.

1.1.4 Hole

template<typename **TConstant** = *constant*<>>
class hole
A hole represents a hole in the 2D plan, it might be a window or a door or a hole in a wall.

1.1.5 Room

template<typename **TConstant** = *constant*<>>
class room

1.2 Algorithms

1.2.1 Enumerations

enum bimpp::plan2d::algorithm::**room_side**
An enum, it means the direction of the room's edges.

Values:

enumerator room_side_both
Don't know the room faces inside or outside.

enumerator room_side_in
The room faces inside.

enumerator room_side_out
The room faces outside.

1.2.2 Data structures

Define some data for algorithm.

class wall_ex

class node_ex

class room_ex

1.2.3 Functions

They implement some algorithms that.

template<typename **TItem**>
static inline bool bimpp::plan2d::algorithm::**isContainsForBiggerVector**(**const** std::vector<*TItem*> &*_all*, **const** std::vector<*TItem*> &*_sub*)

Is the `_all` contains `_sub`?

Importance: The `_all` and `_sub` must be sorted from small to big.

Parameters

- `_all`: A vector contains all items
- `_sub`: A vector contains a part of items

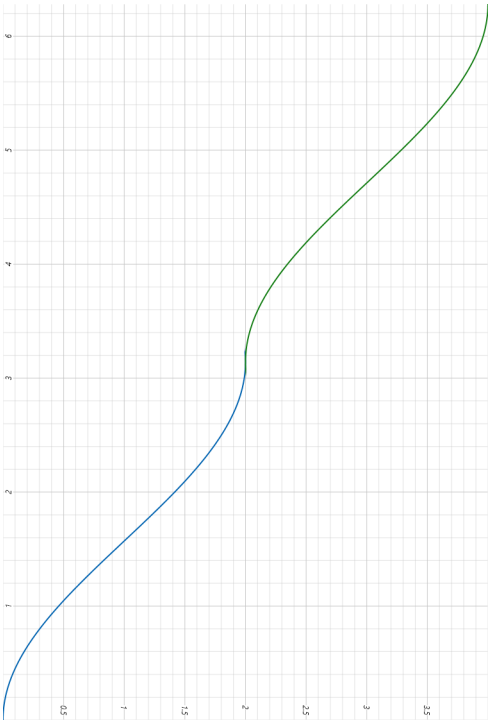
static inline precision_type bimpp::plan2d::algorithm::**calculateAngleEx**(**const** node_type &*_o*, **const** node_type &*_a*, **const** node_type &*_b*)

Calculate the angle α between $\vec{OA}$ and $\vec{OB}$.

$$f(\vec{OA}, \vec{OB}) = \begin{cases} \cos^{-1}(\vec{OA} \cdot \vec{OB}) & \text{if } 0° \leq \alpha \leq 180° \\ \pi * 2 - \cos^{-1}(\vec{OA} \cdot \vec{OB}) & \text{if } 180° < \alpha < 360° \end{cases}$$

Parameters

- `_o`: The O node(point)
- `_a`: The A node(point)
- `_b`: The B node(point)



x-axis: $cos(\alpha)$; **y-axis**: α in radians

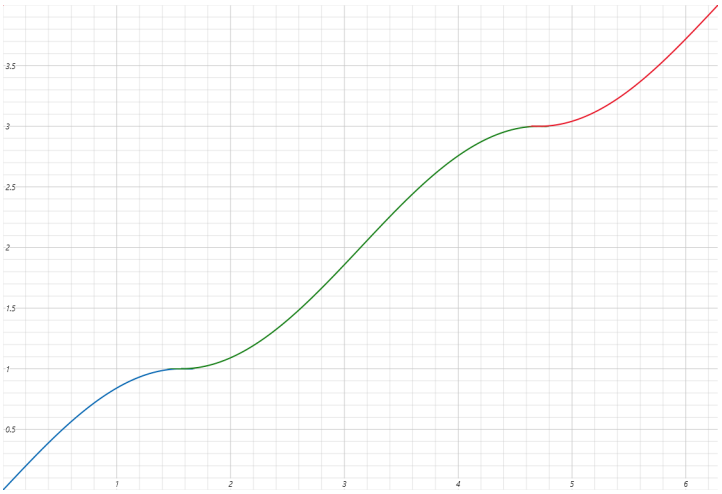
static inline precision_type `bimpp::plan2d::algorithm::`**calculateSinAngleEx**(**const** node_type &*_o*, **const** node_type &*_a*, **const** node_type &*_b*)

Calculate the increased sine of a angle α between $\vec{OA}$ and $\vec{OB}$.

$$f(\vec{OA}, \vec{OB}) = \begin{cases} |\vec{OA} \times \vec{OB}| & \text{if } 0° \leq \alpha \leq 90° \\ 2 - |\vec{OA} \times \vec{OB}| & \text{if } 90° < \alpha < 270° \\ 4 + |\vec{OA} \times \vec{OB}| & \text{if } 270° \leq \alpha < 360° \end{cases}$$

Parameters

- `_o`: The O node(point)
- `_a`: The A node(point)
- `_b`: The B node(point)



x-axis: α in radians; **y-axis**: increased $sin(\alpha)$

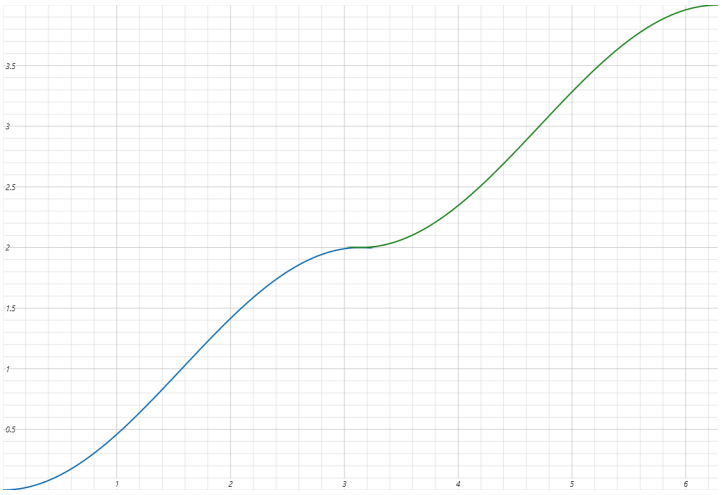
static inline precision_type bimpp::plan2d::algorithm::**calculateCosAngleEx**(**const** node_type &*_o*, **const** node_type &*_a*, **const** node_type &*_b*)

Calculate the increased cosine of a angle α between $\vec{OA}$ and $\vec{OB}$.

$$f(\vec{OA}, \vec{OB}) = \begin{cases} 1 - \vec{OA} \cdot \vec{OB} & \text{if } 0° \leq \alpha \leq 180° \\ 3 + \vec{OA} \cdot \vec{OB} & \text{if } 180° < \alpha < 360° \end{cases}$$

Parameters

- _o: The O node(point)
- _a: The A node(point)
- _b: The B node(point)



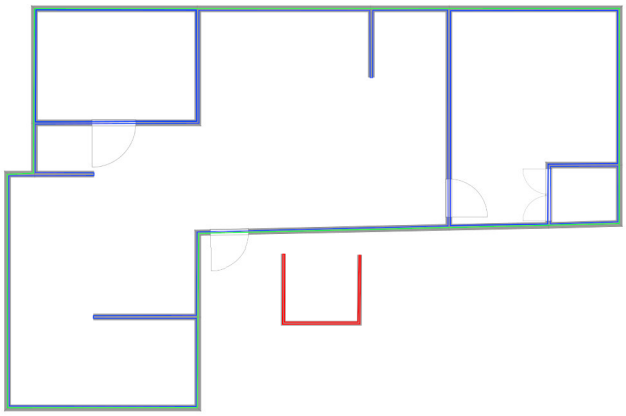
x-axis: α in radians; **y-axis**: increased $cos(\alpha)$

static inline bool bimpp::plan2d::algorithm::**computeRoomExs**(**const** house_type &*_house*, room_ex_vector &*_room_exs*, id_type *_room_id* = TConstant::none_id)

Compute all room's edges by all walls, and don't use recursion.

Parameters

- _house: The house
- _room_exs: Output the room's edge list
- _room_id: The special id of room, find all rooms if it is none



This function computes some edges of all areas that are wrapped by colourful lines.

- Blue lines: Means the *room_ex* is inside of the lines, and the *side_type* is *bimpp::plan2d::algorithm::room_side_in*.
- Green lines: Means the *room_ex* is outside of the lines, and the *side_type* is *bimpp::plan2d::algorithm::room_side_out*.
- Red lines: Means the *room_ex* is both inside and outside of the lines, and the *side_type* is *bimpp::plan2d::algorithm::room_side_both*.

1.3 Usage

1.3.1 bimpp::plan2d::algorithm<>::computeRoomExs

```
// Create a house
bimpp::plan2d::house<> bimpp_house;
// ... add some nodes ...
// ... add some walls ...
// ... add some rooms ...
```

(continues on next page)

(continued from previous page)

```
// Create a `room_ex` vector for the output result
bimpp::plan2d::algorithm<>::room_ex_vector bimpp_room_exs;
// Compute all edges of rooms by all or a specialed room
bimpp::plan2d::algorithm<>::computeRoomExs(bimpp_house, bimpp_room_exs/*, or set a␣
↪room id */);
```

CHAPTER

TWO

LICENSE

```
/*
 * The MIT License (MIT)
 * Copyright © 2020 BIM++
 *
 * Permission is hereby granted, free of charge, to any person obtaining
 * a copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
 * in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
 * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
 * OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
 * ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
 * OTHER DEALINGS IN THE SOFTWARE.
 */
```

INDEX

B