OG2DLibrary

AUTHOR

バージョン

2018年10月07日(日)

目次

Table of contents

OG2DLibrary version 0.75

自作OpenGL2Dゲームライブラリ

開発実績

開発実績

このライブラリを使用して開発したゲーム一覧

teamTrickle Trickle

https://www.youtube.com/watch?v=0EuPpEUQvic

名前空間索引

名前空間一覧

全名前空間の一覧です。

階層索引

クラス階層

クラス階層一覧です。大雑把に文字符号順で並べられています。

クラス索引

クラス一覧

クラス・構造体・共用体・インターフェースの一覧です。

ファイル索引

ファイル一覧

ファイル一覧です。

名前空間詳解

CT 名前空間

生成するときに使用する型

列挙型

- enum **CollisionType** { **BOX**, **CIRCLE**, **POINTER**, **CAPSULE**, **LINE**, **NON** }

詳解

生成するときに使用する型

namespace **CT**

列挙型詳解

enum CT::CollisionType

列挙値:

BOX	矩形
CIRCLE	円
POINTER	点
CAPSULE	カプセル
LINE	線
NON	無

GO 名前空間

GameObjectに関するデータ

列挙型

- enum **Mode** { **NORMAL**, **PAUSE**, **STOP**, **ALLSTOP**, **KILL** }
状態の設定

詳解

GameObjectに関するデータ

namespace **GO**

列挙型詳解

enum GO::Mode

状態の設定

enum Mode

列挙値:

NORMAL	通常処理
PAUSE	ポーズ用処理
STOP	更新停止
ALLSTOP	描画更新共に停止
KILL	削除

In 名前空間

簡易引数用

列挙型

- enum **Button** { **BUTTON_A**, **BUTTON_B**, **BUTTON_X**, **BUTTON_Y**, **BUTTON_L1**, **BUTTON_R1**, **BUTTON_BACK**, **BUTTON_START**, **BUTTON_L3**, **BUTTON_R3**, **BUTTON_U**, **BUTTON_R**, **BUTTON_D**, **BUTTON_L** }
- *仮装コントローラの入力設定* enum **AXIS** { **AXIS_LEFT_X**, **AXIS_LEFT_Y**, **AXIS_RIGHT_X**, **AXIS_RIGHT_Y**, **AXIS_R2**, **AXIS_L2**, **AXIS_BUTTON_NUM** }
- *コントローラのスティックと押し込み* enum **StickButton** { **LSTICK_LEFT**, **LSTICK_RIGHT**, **LSTICK_UP**, **LSTICK_DOWN**, **RSTICK_LEFT**, **RSTICK_RIGHT**, **RSTICK_UP**, **RSTICK_DOWN**, **BUTTON_R2**, **BUTTON_L2**, **STICK_NUM** }
- *スティックをボタンButton入力にも対応* enum **IN** { **B1**, **B2**, **B3**, **B4**, **CD**, **CU**, **CR**, **CL**, **L1**, **R1**, **D1**, **D2**, **SR**, **SL**, **LD**, **LU**, **LR**, **LL**, **RD**, **RU**, **RR**, **RL**, **L2**, **R2** }
- *仮想入力* enum **Key** { **A**, **S**, **D**, **W**, **Q**, **E**, **Z**, **X**, **C**, **R**, **F**, **V**, **T**, **G**, **B**, **Y**, **H**, **N**, **U**, **J**, **M**, **I**, **K**, **O**, **L**, **P**, **SPACE**, **ENTER**, **ESCAPE**, **UP**, **DOWN**, **LEFT**, **RIGHT** }

キーボード入力

詳解

簡易引数用

namespace **In**

列挙型詳解

enum In::AXIS

コントローラのスティックと押し込み

enum AXIS

列挙値:

AXIS_LEFT_X	左スティックX値
AXIS_LEFT_Y	左スティックY値
AXIS_RIGHT_X	右スティックX値
AXIS_RIGHT_Y	右スティックY値
AXIS_R2	R2
AXIS_L2	L2
AXIS_BUTTON_NUM	AXISNumber

enum In::Button

仮装コントローラの入力設定

enum **Button**

列挙値:

BUTTON_A	1
BUTTON_B	2
BUTTON_X	3
BUTTON_Y	4
BUTTON_L1	5
BUTTON_R1	6
BUTTON_BACK	7
BUTTON_START	8
BUTTON_L3	9
BUTTON_R3	10
BUTTON_U	11
BUTTON_R	12
BUTTON_D	13
BUTTON_L	14

enum In::IN

仮想入力

enum IN

列挙値:

B1	
B2	
B3	
B4	
CD	
CU	

CR	
CL	
L1	
R1	
D1	
D2	
SR	
SL	
LD	
LU	
LR	
LL	
RD	
RU	
RR	
RL	
L2	
R2	

enum In::Key

キーボード入力

enum Key

列挙値:

A	
S	
D	
W	
Q	
E	
Z	
X	
C	
R	
F	
V	
T	
G	
B	
Y	
H	
N	
U	
J	
M	
I	
K	
O	
L	
P	
SPACE	
ENTER	
ESCAPE	
UP	

DOWN	
LEFT	
RIGHT	

enum In::StickButton

スティックをボタンButton入力にも対応

enum StickButton

列挙値:

LSTICK_LEFT	左スティック左入力
LSTICK_RIGHT	左スティック右入力
LSTICK_UP	左スティック上入力
LSTICK_DOWN	左スティック下入力
RSTICK_LEFT	右スティック左入力
RSTICK_RIGHT	右スティっク右入力
RSTICK_UP	右スティック上入力
RSTICK_DOWN	右スティック下入力
BUTTON_R2	R2
BUTTON_L2	L2
STICK_NUM	スティック数

Mouse 名前空間

マウス用簡易引数

列挙型

- enum **Button** { **LEFT**, **RIGTH**, **CENTER**, **BUTTON_4**, **BUTTON_5**, **BUTTON_6**, **BUTTON_7**, **BUTTON_8** }

Mouseのボタン

詳解

マウス用簡易引数

namespace **Mouse**

列挙型詳解

enum Mouse::Button

Mouseのボタン

enum **Button**

列挙値:

LEFT	左
RIGTH	右
CENTER	中心
BUTTON_4	
BUTTON_5	
BUTTON_6	
BUTTON_7	
BUTTON_8	

OG 名前空間

便利機能関数や数学計算

関数

- void **MulitMatrixf** (GLfloat *src1, GLfloat *src2, GLfloat *dst)
- void **Normalize** (GLfloat *v)
- void **LineBoxDraw** (const **Box2D** *_b, const **Color** &color, const float linewidth)
- void **LineBoxDraw** (const **Box2D** *_b, const float linewidth, const **Color** &color)
- void **LineBoxDraw** (const **Vec2** *_b, const **Color** &color, const float linewidth)
- void **LineBoxDraw** (const **Vec2** *_b, const float linewidth, const **Color** &color)
- void **LineDraw** (const **Vec2** *_b, const **Color** &color, const float linewidth)
- void **LineDraw** (const **Vec2** *_b, const float linewidth, const **Color** &color)
- void **LineOvalDraw** (const **Vec2** *pos, const **Vec2** *scale, const float linewidth, const **Color** &color)
- void **LineOvalDraw** (const int x, const int y, const float ovalx, const float ovaly, const float linewidth, const **Color** &color)
- void **PointDraw** (const **Vec2** *pos, const **Color** &color, const float linewidth)
- void **PointDraw** (const **Vec2** *pos, const float linewidth, const **Color** &color)
- void **_Rotate** (const float _ang, **Vec2** *_b)
- void **BackColor** (const **Color** &color)
- void **BackColor** (const float &red, const float &green, const float &blue, const float &alpha)
- int **mbclen** (const char *c)
- void **cout** (const **Box2D** &)
- void **cout** (const **Vec2** &)
- void **cout** (const **Color** &)
- bool **Data_Cipher** (const std::string &in_path, const std::string &out_path)
- std::string **Data_Composite** (std::ifstream &ifs)
- void **OutDebugData** (const std::string &out_path, const std::string &text)
- void **DataClear** (const std::string &path)
- template<class T > bool **Destroy** (T *t)
- template<class T > bool **Destroy** (const T *t)
- float **ToRadian** (const float degree_)
- void **Cross** (float *, float *, float *)
- float **inner** (const **Vec2** &, const **Vec2** &)
- float **inner** (const float, const float, const float, const float)
- float **inner** (const int, const int, const int, const int)
- float **cross** (const **Vec2** &, const **Vec2** &)
- float **cross** (const float, const float, const float, const float)
- float **cross** (const int, const int, const int, const int)
- float **doubleinner** (const **Vec2** &)
- float **doubleinner** (const float, const float)
- float **doubleinner** (const int, const int)
- float **get_distance** (const float, const float, const float, const float, const float, const float)
- bool **innerJudge** (const **Vec2** *line, const **Vec2** *point)

詳解

便利機能関数や数学計算

数学計算

namespace **OG**

関数詳解

void OG::_Rotate (const float _*ang*, Vec2 * _*b*)

void OG::BackColor (const Color & *color*)

void OG::BackColor (const float & *red*, const float & *green*, const float & *blue*, const float & *alpha*)

void OG::cout (const Box2D & *b*)

void OG::cout (const Vec2 & *v*)

void OG::cout (const Color & *c*)

void OG::Cross (float * *src1*, float * *src2*, float * *dst*)

float OG::cross (const Vec2 & _*v1*, const Vec2 & _*v2*)

float OG::cross (const float _*x1*, const float _*y1*, const float _*x2*, const float _*y2*)

float OG::cross (const int _*x1*, const int _*y1*, const int _*x2*, const int _*y2*)

bool OG::Data_Cipher (const std::string & *in_path*, const std::string & *out_path*)

std::string OG::Data_Composite (std::ifstream & *ifs*)

void OG::DataClear (const std::string & *path*)

template<class T > bool OG::Destroy (T * *t*)

template<class T > bool OG::Destroy (const T * *t*)

float OG::doubleinner (const Vec2 & _*v*)

float OG::doubleinner (const float _*x*, const float _*y*)

float OG::doubleinner (const int _*x*, const int _*y*)

float OG::get_distance (const float *x*, const float *y*, const float *x1*, const float *y1*, const float *x2*, const float *y2*)

float OG::inner (const Vec2 & _*v1*, const Vec2 & _*v2*)

float OG::inner (const float _*x1*, const float _*y1*, const float _*x2*, const float _*y2*)

float OG::inner (const int _*x1*, const int _*y1*, const int _*x2*, const int _*y2*)

bool OG::innerJudge (const Vec2 * *line*, const Vec2 * *point*)

void OG::LineBoxDraw (const Box2D * _*b*, const Color & *color*, const float *linewidth*)

void OG::LineBoxDraw (const Box2D * *_b*, const float *linewidth*, const Color & *color*)

void OG::LineBoxDraw (const Vec2 * *_b*, const Color & *color*, const float *linewidth*)

void OG::LineBoxDraw (const Vec2 * *_b*, const float *linewidth*, const Color & *color*)

void OG::LineDraw (const Vec2 * *_b*, const Color & *color*, const float *linewidth*)

void OG::LineDraw (const Vec2 * *_b*, const float *linewidth*, const Color & *color*)

void OG::LineOvalDraw (const Vec2 * *pos*, const Vec2 * *scale*, const float *linewidth*, const Color & *color*)

void OG::LineOvalDraw (const int *x*, const int *y*, const float *ovalx*, const float *ovaly*, const float *linewidth*, const Color & *color*)

int OG::mbclen (const char * *c*)

void OG::MulitMatrixf (GLfloat * *src1*, GLfloat * *src2*, GLfloat * *dst*)

void OG::Normalize (GLfloat * *v*)

void OG::OutDebugData (const std::string & *out_path*, const std::string & *text*)

void OG::PointDraw (const Vec2 * *pos*, const Color & *color*, const float *linewidth*)

void OG::PointDraw (const Vec2 * *pos*, const float *linewidth*, const Color & *color*)

float OG::ToRadian (const float *degree_*)

random 名前空間

ランダム生成名前空間

関数

- void **Init** ()
  初期化
- int **GetRand** (const int min_, const int max_)
  整数のランダム
- float **GetRand** (const float min_, const float max_)
  浮動小数点のランダム
- std::string **GetRand** (const std::string &text, const std::size_t size)
  文字列のランダム

詳解

ランダム生成名前空間

namespace random 既存GameEngineを使用している場合は自動で生成、初期化される

関数詳解

int random::GetRand (const int *min_*, const int *max_*)

整数のランダム

引数:

in	*int*	min_ 最低値
in	*int*	max_ 最大値　ランダム値

float random::GetRand (const float *min_*, const float *max_*)

浮動小数点のランダム

引数:

in	*float*	min_ 最低値
in	*float*	max_ 最大値

戻り値:

float ランダム値

std::string random::GetRand (const std::string & *text*, const std::size_t *size*)

文字列のランダム

引数:

in	*string*	text 使用する文字たち
in	*size_t*	size 返す文字のサイズ

戻り値:

string ランダム文字列

void random::Init ()

初期化

Shader 名前空間

未実装

関数

- GLuint **compile** (GLuint type, const std::string &text)
- void **setup** (const GLuint program, const std::string &v_source, const std::string &f_source)
- GLuint **read** (const std::string &file)
- GLint **attrib** (const GLint program, const std::string &name)
- GLint **uniform** (const GLuint program, const std::string &name)
- void **use** (const GLuint program)
- void **SetViewPort** (float cl, float cr, float cb, float ct, float cn, float cf)

詳解

未実装

関数詳解

GLint Shader::attrib (const GLint *program*, const std::string & *name*)

GLuint Shader::compile (GLuint *type*, const std::string & *text*)

GLuint Shader::read (const std::string & *file*)

void Shader::setup (const GLuint *program*, const std::string & *v_source*, const std::string & *f_source*)

void Shader::SetViewPort (float *cl*, float *cr*, float *cb*, float *ct*, float *cn*, float *cf*)

GLint Shader::uniform (const GLuint *program*, const std::string & *name*)

void Shader::use (const GLuint *program*)

ST 名前空間

SceneTaskに関するデータ

列挙型

- enum **Mode** { **NORMAL**, **PAUSE**, **STOP**, **KILL** }

詳解

SceneTaskに関するデータ

namespace **ST**

列挙型詳解

enum ST::Mode

enum Mode 状態の設定

列挙値:

NORMAL	通常処理
PAUSE	ポーズ用処理
STOP	更新停止
KILL	削除

UO 名前空間

UIObjectに関するデータ

列挙型

- enum **Mode** { **NORMAL**, **PAUSE**, **STOP**, **ALLSTOP**, **KILL** }

状態の設定

詳解

UIObjectに関するデータ

namespace **UO**

列挙型詳解

enum UO::Mode

状態の設定

enum Mode

列挙値:

NORMAL	通常処理
PAUSE	ポーズ用処理
STOP	更新停止
ALLSTOP	描画更新共に停止
KILL	削除

クラス詳解

Audio クラス

デバイスを設定するためのclass
`#include <Audio.h>`

公開メンバ関数

- **Audio** ()
- **~Audio** ()

詳解

デバイスを設定するためのclass

構築子と解体子

Audio::Audio ()

Audio::~Audio ()

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Audio/**Audio.h**
- src/lib/OGSystem/Audio/**Audio.cpp**

Easing::Back クラス

```
#include <easing.hpp>
```

公開メンバ関数

- float **In** (float t, float b, float c, float d)
- float **Out** (float t, float b, float c, float d)
- float **InOut** (float t, float b, float c, float d)

関数詳解

float Easing::Back::In (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Back::InOut (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Back::Out (float *t*, float *b*, float *c*, float *d*) `[inline]`

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Easing/**easing.hpp**

Easing::Bounce クラス

```
#include <easing.hpp>
```

公開メンバ関数

- float **Out** (float t, float b, float c, float d)
- float **In** (float t, float b, float c, float d)
- float **InOut** (float t, float b, float c, float d)

関数詳解

float Easing::Bounce::In (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Bounce::InOut (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Bounce::Out (float *t*, float *b*, float *c*, float *d*) `[inline]`

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Easing/**easing.hpp**

Box2D クラス

2Dデータ型
`#include <OGlib.h>`

公開メンバ関数

- **Box2D** ()
- **Box2D** (const float x_, const float y_, const float w_, const float h_)
- **Box2D** (const int x_, const int y_, const int w_, const int h_)
- **Box2D** (const **Box2D** &b_)
- **Box2D** (const **Vec2** &p, const **Vec2** &s)
- void **Offset** (const float x_, const float y_)
- void **Offset** (const int x_, const int y_)
- void **Offset** ()
- void **OffsetSize** ()
- void **OffsetCenterSize** ()
- **Box2D operator+** (const **Box2D** &b)
- **Box2D operator-** (const **Box2D** &b)
- **Box2D operator*** (const **Box2D** &b)
- void **operator+=** (const **Box2D** &b)
- void **operator-=** (const **Box2D** &b)
- void **operator*=** (const **Box2D** &b)

公開変数類

- float **x**
- float **y**
- float **w**
- float **h**

詳解

2Dデータ型

構築子と解体子

Box2D::Box2D ()

Box2D::Box2D (const float *x_*, const float *y_*, const float *w_*, const float *h_*)

Box2D::Box2D (const int *x_*, const int *y_*, const int *w_*, const int *h_*)

Box2D::Box2D (const Box2D & *b_*)

Box2D::Box2D (const Vec2 & *p*, const Vec2 & *s*)

関数詳解

void Box2D::Offset (const float *x_*, const float *y_*)

void Box2D::Offset (const int *x_*, const int *y_*)

void Box2D::Offset ()

void Box2D::OffsetCenterSize ()

void Box2D::OffsetSize ()

Box2D Box2D::operator* (const Box2D & *b*)

void Box2D::operator*= (const Box2D & *b*)

Box2D Box2D::operator+ (const Box2D & *b*)

void Box2D::operator+= (const Box2D & *b*)

Box2D Box2D::operator- (const Box2D & *b*)

void Box2D::operator-= (const Box2D & *b*)

メンバ詳解

float Box2D::h

float Box2D::w

float Box2D::x

float Box2D::y

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/OG/**OGlib.h**
- src/lib/OGSystem/OG/**OGlib.cpp**

Box3D クラス

3DBoxデータ型
`#include <OGlib.h>`

公開メンバ関数

- **Box3D** ()
- **Box3D** (const float ex, const float ey, const float ez, const float ew, const float eh, const float ed)
- **Box3D** (const **Box3D** &_e)

公開変数類

- float **x**
- float **y**
- float **z**
- float **w**
- float **h**
- float **d**

詳解

3DBoxデータ型

構築子と解体子

Box3D::Box3D ()

Box3D::Box3D (const float *ex*, const float *ey*, const float *ez*, const float *ew*, const float *eh*, const float *ed*)

Box3D::Box3D (const Box3D & *_e*)

メンバ詳解

float Box3D::d

float Box3D::h

float Box3D::w

float Box3D::x

float Box3D::y

float Box3D::z

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/OG/**OGlib.h**

- src/lib/OGSystem/OG/**OGlib.cpp**

Buffer クラス

SoundDataのBufferDataを扱う
`#include <Audio.h>`

公開メンバ関数

- **Buffer** ()
- **Buffer** (const std::string &path_)
- virtual **~Buffer** ()
- float **GetTime** () const
- ALuint **GetID** () const
- void **Bind** (const bool stereo, const void *data, const **u_int** size, const **u_int** rate) const

公開変数類

- ALuint **id_**
- float **nowTime**

詳解

SoundDataのBufferDataを扱う

構築子と解体子

Buffer::Buffer () `[explicit]`

Buffer::Buffer (const std::string & *path_*) `[explicit]`

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

関数詳解

void Buffer::Bind (const bool *stereo*, const void * *data*, const u_int *size*, const u_int *rate*) const

ALuint Buffer::GetID () const

float Buffer::GetTime () const

メンバ詳解

ALuint Buffer::id_

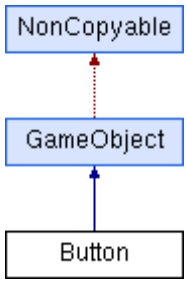
float Buffer::nowTime

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Audio/**Audio.h**
- src/lib/OGSystem/Audio/**Audio.cpp**

Button クラス

マウスに反応するButtonclass
`#include <Button.h>`
Button の継承関係図



公開メンバ関数

- virtual **~Button** ()
  destructor
- bool **IsMouseHit** () const
  マウスポインタとの判定を返す
- void **SetText** (const std::string &text)
  Buttonに設置するText

静的公開メンバ関数

- static **Button** * **Create** (const **Vec2** &pos, const **Vec2** &size, const std::string &name)
  Button生成

詳解

マウスに反応するButtonclass

マウスと当たっていてクリックされている判定を行ってくれる

構築子と解体子

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

destructor

関数詳解

Button * Button::Create (const Vec2 & *pos*, const Vec2 & *size*, const std::string & *name*) `[static]`

Button生成

引数:

in	*const*	**Vec2**& pos 位置
in	*const*	**Vec2**& size 大きさ
in	*const*	string& name 登録名

戻り値:

Button* 生成したButton

bool Button::IsMouseHit () const

マウスポインタとの判定を返す

戻り値:

bool Hit true

void Button::SetText (const std::string & *text*)

Buttonに設置するText

引数:

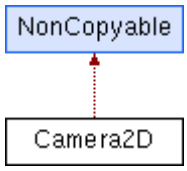
in	*const*	string& text

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Button/**Button.h**
- src/lib/OGSystem/Button/**Button.cpp**

Camera2D クラス

2DCamera
`#include <Camera.h>`
Camera2D の継承関係図



公開メンバ関数

- **Camera2D** ()
  constructor
- **Camera2D** (const **Box2D** &b)
  constructor
- virtual **~Camera2D** ()
  destructor
- void **Initialize** (const **Box2D** &b)
  初期化処理
- void **Update** () const
  更新処理
- void **MovePos** (const **Vec2** &move)
  位置を移動させる
- void **SetPos** (const **Vec2** &pos)
  位置を設定する
- void **SetSize** (const **Vec2** &size)
  サイズを設定する
- void **MoveSize** (const **Vec2** &move)
  サイズを動かす
- void **SetPos_x** (const float &x)
  カメラ位置のX座標を設定する
- void **SetPos_y** (const float &y)
  カメラ位置のY座標を設定する
- void **SetSize_w** (const float &w)
  カメラの横サイズを変える
- void **SetSize_h** (const float &h)
  カメラの縦サイズを変える
- **Vec2 GetPos** () const
  カメラの位置を返す
- **Vec2 GetSize** () const
  カメラのサイズを返す

詳解

2DCamera

GL機能を使用し2D上でのカメラを扱うclass

既存GameEngineを使用している場合は内部で宣言されている
使用しない場合は2Dの描画のために１つ生成する必要がある

構築子と解体子

Camera2D::Camera2D () `[explicit]`

constructor

Camera2D::Camera2D (const Box2D &　*b*) `[explicit]`

constructor

引数:

in	*const*	**Box2D**& b 位置とサイズ

Camera2D::~Camera2D () `[virtual]`

destructor

関数詳解

Vec2 Camera2D::GetPos () const

カメラの位置を返す

戻り値:

Vec2 カメラ位置

Vec2 Camera2D::GetSize () const

カメラのサイズを返す

戻り値:

Vec2 カメラの大きさ

void Camera2D::Initialize (const Box2D &　*b*)

初期化処理

引数:

in	*const*	**Box2D**& b 位置とサイズ

void Camera2D::MovePos (const Vec2 &　*move*)

位置を移動させる

引数:

in	*const*	**Vec2**& move 移動値

void Camera2D::MoveSize (const Vec2 &　*move*)

サイズを動かす

引数:

in	*const*	**Vec2**& move 移動値

void Camera2D::SetPos (const Vec2 &　*pos*)

位置を設定する

引数:

in	*const*	**Vec2**& pos 位置

void Camera2D::SetPos_x (const float &　*x*)

カメラ位置のX座標を設定する

引数:

in	*float*	x 座標値

void Camera2D::SetPos_y (const float &　*y*)

カメラ位置のY座標を設定する

引数:

in	*float*	y 座標値

void Camera2D::SetSize (const Vec2 &　*size*)

サイズを設定する

引数:

in	*const*	**Vec2**& size 大きさ

void Camera2D::SetSize_h (const float &　*h*)

カメラの縦サイズを変える

引数:

in	*float*	w 縦サイズ

void Camera2D::SetSize_w (const float & *w*)

カメラの横サイズを変える

引数:

in	*float*	w 横サイズ

void Camera2D::Update () const

更新処理

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Camera/**Camera.h**
- src/lib/OGSystem/Camera/**Camera.cpp**

Easing::Circ クラス

```
#include <easing.hpp>
```

公開メンバ関数

- float **In** (float t, float b, float c, float d)
- float **Out** (float t, float b, float c, float d)
- float **InOut** (float t, float b, float c, float d)

関数詳解

float Easing::Circ::In (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Circ::InOut (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Circ::Out (float *t*, float *b*, float *c*, float *d*) `[inline]`

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Easing/**easing.hpp**

Circle クラス

円データ型
`#include <OGlib.h>`

公開メンバ関数

- **Circle** ()
- **Circle** (const float, const float, const float)
- **Circle** (const int, const int, const int)
- **Circle** (const **Circle** &)

公開変数類

- float **center_x**
- float **center_y**
- float **r**

詳解

円データ型

構築子と解体子

Circle::Circle ()

Circle::Circle (const float _*x*, const float _*y*, const float _*r*)

Circle::Circle (const int _*x*, const int _*y*, const int _*r*)

Circle::Circle (const Circle & _*c*)

メンバ詳解

float Circle::center_x

float Circle::center_y

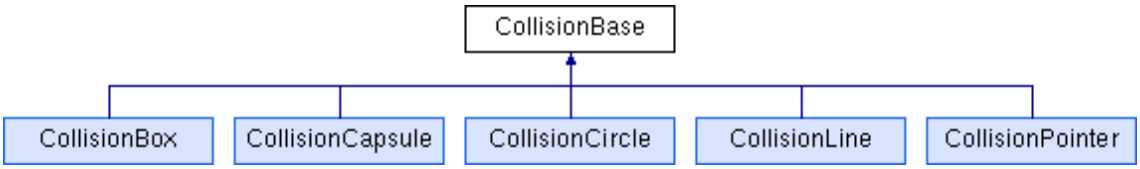
float Circle::r

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/OG/**OGlib.h**
- src/lib/OGSystem/OG/**OGlib.cpp**

CollisionBase クラス

判定の元
`#include <Collision.h>`
CollisionBase の継承関係図



公開メンバ関数

- **CollisionBase** (const unsigned short vertex)
  constructor
- virtual void **CreateHitBase** (**Vec2** *pos, **Vec2** *scale, **Vec2** *radius, float *angle)
  当たり判定を生成する
- virtual bool **Hit** (**CollisionBase** *collision)=0
  当たり判定を返す
- virtual bool **GetHit** (**CollisionBox** *b)=0
  矩形との判定
- virtual bool **GetHit** (**CollisionCircle** *b)=0
  円との判定
- virtual bool **GetHit** (**CollisionPointer** *b)=0
  点との判定
- virtual bool **GetHit** (**CollisionCapsule** *b)=0
  カプセルとの判定
- virtual bool **GetHit** (**CollisionLine** *b)=0
  線との判定

公開変数類

- const unsigned short **VERTEX_NUM**
  頂点数
- **Vec2** * **_pos**
  Position
- **Vec2** * **_scale**
  Scale
- **Vec2** * **_radius**
  Radius
- float * **_rotate**
  Angle

詳解

判定の元

構築子と解体子

CollisionBase::CollisionBase (const unsigned short *vertex*) [explicit]

constructor

引数:

in	*unsigned*	short vertex 頂点数

関数詳解

void CollisionBase::CreateHitBase (Vec2 * *pos*, Vec2 * *scale*, Vec2 * *radius*, float * *angle*) [virtual]

当たり判定を生成する

引数:

in	*Vec2**	pos Position
in	*Vec2**	scale Scale
in	*Vec2**	radius Radius
in	*float**	angle Angle

virtual bool CollisionBase::GetHit (CollisionBox * *b*) [pure virtual]

矩形との判定

引数:

in	*CollisionBox**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionCapsule (*p.51*), **CollisionLine** (*p.58*), **CollisionPointer** (*p.61*), **CollisionCircle** (*p.55*), **CollisionBox** (*p.48*)で実装されています。

virtual bool CollisionBase::GetHit (CollisionCircle * *b*) [pure virtual]

円との判定

引数:

in	*CollisionCircle**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionCapsule (*p.52*), **CollisionLine** (*p.58*), **CollisionPointer** (*p.61*), **CollisionCircle** (*p.55*), **CollisionBox** (*p.48*)で実装されています。

virtual bool CollisionBase::GetHit (CollisionPointer * *b*) [pure virtual]

点との判定

引数:

in	*CollisionPointer**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionCapsule (*p.52*), **CollisionLine** (*p.58*), **CollisionPointer** (*p.61*), **CollisionCircle** (*p.55*), **CollisionBox** (*p.48*)で実装されています。

virtual bool CollisionBase::GetHit (CollisionCapsule * *b*) [pure virtual]

カプセルとの判定

引数:

in	*CollisionCapsule**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionCapsule (*p.52*), **CollisionLine** (*p.59*), **CollisionPointer** (*p.62*), **CollisionCircle** (*p.56*), **CollisionBox** (*p.49*)で実装されています。

virtual bool CollisionBase::GetHit (CollisionLine * *b*) [pure virtual]

線との判定

引数:

in	*CollisionLine**	b 相手のオブジェクト　当たっていればtrue

CollisionCapsule (*p.52*), **CollisionLine** (*p.58*), **CollisionPointer** (*p.61*), **CollisionCircle** (*p.55*), **CollisionBox** (*p.48*)で実装されています。

virtual bool CollisionBase::Hit (CollisionBase * *collision*) [pure virtual]

当たり判定を返す

引数:

in	*CollisionBase**	collision 判定相手

戻り値:

bool true hit

CollisionCapsule (*p.53*), **CollisionLine** (*p.59*), **CollisionPointer** (*p.62*), **CollisionCircle** (*p.56*), **CollisionBox** (*p.49*)で実装されています。

メンバ詳解

Vec2* CollisionBase::_pos

Position

Vec2* CollisionBase::_radius

Radius

float* CollisionBase::_rotate

Angle

Vec2* CollisionBase::_scale

Scale

const unsigned short CollisionBase::VERTEX_NUM

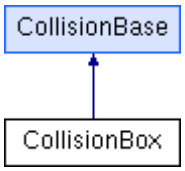
頂点数

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Collision/**Collision.h**
- src/lib/OGSystem/Collision/**Collision.cpp**

CollisionBox クラス

矩形判定
`#include <Collision.h>`
CollisionBox の継承関係図



公開メンバ関数

- **CollisionBox** ()
  constructor
- void **Rotate** (const float _angle)
  回転値を変更する
- float **Rotate** () const
  回転値を取得する
- void **CreateCollision** ()
  当たり判定を生成する
- bool **Hit** (**CollisionBase** *collision) override
  当たり判定を返す
- virtual bool **GetHit** (**CollisionBox** *b) override
  矩形との判定
- virtual bool **GetHit** (**CollisionCircle** *b) override
  円との判定
- virtual bool **GetHit** (**CollisionPointer** *b) override
  点との判定
- virtual bool **GetHit** (**CollisionLine** *b) override
  点との判定(未完成)
- virtual bool **GetHit** (**CollisionCapsule** *b) override
  カプセルとの判定(未完成)

公開変数類

- **Box2D hitBase**
  当たり判定

詳解

矩形判定

構築子と解体子

CollisionBox::CollisionBox () `[explicit]`

constructor

関数詳解

void CollisionBox::CreateCollision ()

当たり判定を生成する

bool CollisionBox::GetHit (CollisionBox * *b*) [override], [virtual]

矩形との判定

引数:

in	*CollisionBox**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.44*)を実装しています。

bool CollisionBox::GetHit (CollisionCircle * *b*) [override], [virtual]

円との判定

引数:

in	*CollisionCircle**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.44*)を実装しています。

bool CollisionBox::GetHit (CollisionPointer * *b*) [override], [virtual]

点との判定

引数:

in	*CollisionPointer**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.45*)を実装しています。

bool CollisionBox::GetHit (CollisionLine * *b*) [override], [virtual]

点との判定(未完成)

引数:

in	*CollisionLine**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.45*)を実装しています。

bool CollisionBox::GetHit (CollisionCapsule * *b*) [override], [virtual]

カプセルとの判定(未完成)

引数:

in	*CollisionCapsule**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.45*)を実装しています。

bool CollisionBox::Hit (CollisionBase * *collision*) [override], [virtual]

当たり判定を返す

引数:

in	*CollisionBase**	collision 判定相手

戻り値:

bool true hit

CollisionBase (*p.45*)を実装しています。

void CollisionBox::Rotate (const float _*angle*)

回転値を変更する

引数:

in	*float*	_angle 回転値

float CollisionBox::Rotate () const

回転値を取得する

戻り値:

float 回転値

メンバ詳解

Box2D CollisionBox::hitBase

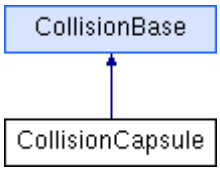
当たり判定

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Collision/**Collision.h**
- src/lib/OGSystem/Collision/**Collision.cpp**

CollisionCapsule クラス

カプセル判定 ！ 未完成
`#include <Collision.h>`
CollisionCapsule の継承関係図



公開メンバ関数

- **CollisionCapsule** ()
  constructor
- bool **Hit** (**CollisionBase** *collision) override
  当たり判定を返す
- virtual bool **GetHit** (**CollisionBox** *b) override
  矩形との判定
- virtual bool **GetHit** (**CollisionCircle** *b) override
  円との判定
- virtual bool **GetHit** (**CollisionPointer** *b) override
  点との判定
- virtual bool **GetHit** (**CollisionLine** *b) override
  線との判定
- virtual bool **GetHit** (**CollisionCapsule** *b) override
  カプセルとの判定

公開変数類

- **Vec2 hitBase** [2]
  当たり判定

詳解

カプセル判定 ！ 未完成

構築子と解体子

CollisionCapsule::CollisionCapsule () `[explicit]`

constructor

関数詳解

bool CollisionCapsule::GetHit (CollisionBox * *b*) `[override]`, `[virtual]`

矩形との判定

引数:

in	*CollisionBox**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.44*)を実装しています。

bool CollisionCapsule::GetHit (CollisionCircle * *b*) [override], [virtual]

円との判定

引数:

in	*CollisionCircle**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.44*)を実装しています。

bool CollisionCapsule::GetHit (CollisionPointer * *b*) [override], [virtual]

点との判定

引数:

in	*CollisionPointer**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.45*)を実装しています。

bool CollisionCapsule::GetHit (CollisionLine * *b*) [override], [virtual]

線との判定

引数:

in	*CollisionLine**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.45*)を実装しています。

bool CollisionCapsule::GetHit (CollisionCapsule * *b*) [override], [virtual]

カプセルとの判定

引数:

in	*CollisionCapsule**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.45*)を実装しています。

bool CollisionCapsule::Hit (CollisionBase * *collision*) [override], [virtual]

当たり判定を返す

引数:

in	*CollisionBase**	collision 判定相手

戻り値:

bool true hit

CollisionBase (*p.45*)を実装しています。

メンバ詳解

Vec2 CollisionCapsule::hitBase[2]

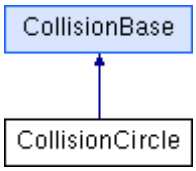
当たり判定

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Collision/**Collision.h**
- src/lib/OGSystem/Collision/**Collision.cpp**

CollisionCircle クラス

円判定
`#include <Collision.h>`
CollisionCircle の継承関係図



公開メンバ関数

- **CollisionCircle** ()
  constructor
- void **CreateCollision** ()
  当たり判定を生成する
- bool **Hit** (**CollisionBase** *collision) override
  当たり判定を返す
- virtual bool **GetHit** (**CollisionBox** *b) override
  矩形との判定
- virtual bool **GetHit** (**CollisionCircle** *b) override
  円との判定
- virtual bool **GetHit** (**CollisionPointer** *b) override
  点との判定
- virtual bool **GetHit** (**CollisionLine** *b) override
  線との判定(未完成)
- virtual bool **GetHit** (**CollisionCapsule** *b) override
  カプセルとの判定(未完成)

公開変数類

- **Circle hitBase**
  当たり判定

詳解

円判定

構築子と解体子

CollisionCircle::CollisionCircle () `[explicit]`

constructor

関数詳解

void CollisionCircle::CreateCollision ()

当たり判定を生成する

bool CollisionCircle::GetHit (CollisionBox * *b*) [override], [virtual]

矩形との判定

引数:

in	*CollisionBox**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.44*)を実装しています。

bool CollisionCircle::GetHit (CollisionCircle * *b*) [override], [virtual]

円との判定

引数:

in	*CollisionCircle**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.44*)を実装しています。

bool CollisionCircle::GetHit (CollisionPointer * *b*) [override], [virtual]

点との判定

引数:

in	*CollisionPointer**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.45*)を実装しています。

bool CollisionCircle::GetHit (CollisionLine * *b*) [override], [virtual]

線との判定(未完成)

引数:

in	*CollisionLine**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.45*)を実装しています。

bool CollisionCircle::GetHit (CollisionCapsule * *b*) [override], [virtual]

カプセルとの判定(未完成)

引数:

in	*CollisionCapsule**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.45*)を実装しています。

bool CollisionCircle::Hit (CollisionBase * *collision*) [override], [virtual]

当たり判定を返す

引数:

in	*CollisionBase**	collision 判定相手

戻り値:

bool true hit

CollisionBase (*p.45*)を実装しています。

メンバ詳解

Circle CollisionCircle::hitBase

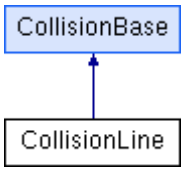
当たり判定

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Collision/**Collision.h**
- src/lib/OGSystem/Collision/**Collision.cpp**

CollisionLine クラス

線判定
`#include <Collision.h>`
CollisionLine の継承関係図



公開メンバ関数

- **CollisionLine** ()
  constructor
- void **CreateCollision** ()
  当たり判定を生成する
- bool **Hit** (**CollisionBase** *collision) override
  当たり判定を返す
- virtual bool **GetHit** (**CollisionBox** *b) override
  矩形との判定
- virtual bool **GetHit** (**CollisionCircle** *b) override
  円との判定
- virtual bool **GetHit** (**CollisionPointer** *b) override
  点との判定
- virtual bool **GetHit** (**CollisionLine** *b) override
  線との判定
- virtual bool **GetHit** (**CollisionCapsule** *b) override
  カプセルとの判定

公開変数類

- **Vec2 hitBase** [2]
  当たり判定

詳解

線判定

構築子と解体子

CollisionLine::CollisionLine () `[explicit]`

constructor

関数詳解

void CollisionLine::CreateCollision ()

当たり判定を生成する

bool CollisionLine::GetHit (CollisionBox * *b*) [override], [virtual]

矩形との判定

引数:

in	*CollisionBox**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.44*)を実装しています。

bool CollisionLine::GetHit (CollisionCircle * *b*) [override], [virtual]

円との判定

引数:

in	*CollisionCircle**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.44*)を実装しています。

bool CollisionLine::GetHit (CollisionPointer * *b*) [override], [virtual]

点との判定

引数:

in	*CollisionPointer**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.45*)を実装しています。

bool CollisionLine::GetHit (CollisionLine * *b*) [override], [virtual]

線との判定

引数:

in	*CollisionLine**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.45*)を実装しています。

bool CollisionLine::GetHit (CollisionCapsule * *b*) [override], [virtual]

カプセルとの判定

引数:

in	*CollisionCapsule**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.45*)を実装しています。

bool CollisionLine::Hit (CollisionBase * *collision*) [override], [virtual]

当たり判定を返す

引数:

in	*CollisionBase**	collision 判定相手

戻り値:

bool true hit

CollisionBase (*p.45*)を実装しています。

メンバ詳解

Vec2 CollisionLine::hitBase[2]

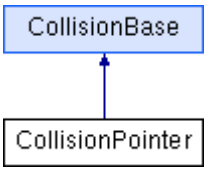
当たり判定

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Collision/**Collision.h**
- src/lib/OGSystem/Collision/**Collision.cpp**

CollisionPointer クラス

点判定
`#include <Collision.h>`
CollisionPointer の継承関係図



公開メンバ関数

- **CollisionPointer** ()
  constructor
- void **CreateCollision** ()
  当たり判定を生成する
- bool **Hit** (**CollisionBase** *collision) override
  当たり判定を返す
- virtual bool **GetHit** (**CollisionBox** *b) override
  矩形との判定
- virtual bool **GetHit** (**CollisionCircle** *b) override
  円との判定
- virtual bool **GetHit** (**CollisionPointer** *b) override
  点との判定
- virtual bool **GetHit** (**CollisionLine** *b) override
  線との判定
- virtual bool **GetHit** (**CollisionCapsule** *b) override
  カプセルとの判定(未完成)

公開変数類

- **Vec2 hitBase**
  当たり判定

詳解

点判定

構築子と解体子

CollisionPointer::CollisionPointer () `[explicit]`

constructor

関数詳解

void CollisionPointer::CreateCollision ()

当たり判定を生成する

bool CollisionPointer::GetHit (CollisionBox * *b*) [override], [virtual]

矩形との判定

引数:

in	*CollisionBox**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.44*)を実装しています。

bool CollisionPointer::GetHit (CollisionCircle * *b*) [override], [virtual]

円との判定

引数:

in	*CollisionCircle**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.44*)を実装しています。

bool CollisionPointer::GetHit (CollisionPointer * *b*) [override], [virtual]

点との判定

引数:

in	*CollisionPointer**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.45*)を実装しています。

bool CollisionPointer::GetHit (CollisionLine * *b*) [override], [virtual]

線との判定

引数:

in	*CollisionLine**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.45*)を実装しています。

bool CollisionPointer::GetHit (CollisionCapsule * *b*) [override], [virtual]

カプセルとの判定(未完成)

引数:

in	*CollisionCapsule**	b 相手のオブジェクト

戻り値:

bool 当たっていればtrue

CollisionBase (*p.45*)を実装しています。

bool CollisionPointer::Hit (CollisionBase * *collision*) [override], [virtual]

当たり判定を返す

引数:

in	*CollisionBase**	collision 判定相手

戻り値:

bool true hit

CollisionBase (*p.45*)を実装しています。

メンバ詳解

Vec2 CollisionPointer::hitBase

当たり判定

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Collision/**Collision.h**
- src/lib/OGSystem/Collision/**Collision.cpp**

Color クラス

色データ
`#include <OGlib.h>`

公開メンバ関数

- **Color** ()
- **Color** (const float r, const float g, const float b, const float a)
- **Color** (const int r, const int g, const int b, const int a)
- unsigned int **Getcolor** () const
- void **operator+=** (const **Color** &b)
- void **operator*=** (const **Color** &b)
- void **operator-=** (const **Color** &b)
- **Color operator+** (const **Color** &)
- **Color operator-** (const **Color** &)
- **Color operator*** (const **Color** &)

公開変数類

- float **red**
- float **green**
- float **blue**
- float **alpha**

詳解

色データ

構築子と解体子

Color::Color ()

Color::Color (const float *r*, const float *g*, const float *b*, const float *a*)

Color::Color (const int *r*, const int *g*, const int *b*, const int *a*)

関数詳解

unsigned int Color::Getcolor () const

Color Color::operator* (const Color & *b*)

void Color::operator*= (const Color & *b*)

Color Color::operator+ (const Color & *b*)

void Color::operator+= (const Color & *b*)

Color Color::operator- (const Color & *b*)

void Color::operator-= (const Color & *b*)

メンバ詳解

float Color::alpha

float Color::blue

float Color::green

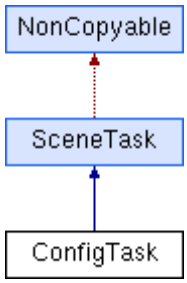
float Color::red

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/OG/**OGlib.h**
- src/lib/OGSystem/OG/**OGlib.cpp**

ConfigTask クラス

WindowをフルスクリーンかWindowで開くか設定を行うScene
`#include <WinConfig.h>`
ConfigTask の継承関係図



公開メンバ関数

- virtual **~ConfigTask** ()
  destructor

静的公開メンバ関数

- static **ConfigTask** * **Create** ()
  Scene生成

詳解

WindowをフルスクリーンかWindowで開くか設定を行うScene

構築子と解体子

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

destructor

関数詳解

ConfigTask * ConfigTask::Create () `[static]`

Scene生成

戻り値:
ConfigTask* 生成Scene

このクラス詳解は次のファイルから抽出されました:

- src/lib/Task/**WinConfig.h**
- src/lib/Task/**WinConfig.cpp**

Easing::Cubic クラス

```
#include <easing.hpp>
```

公開メンバ関数

- float **In** (float t, float b, float c, float d)
- float **Out** (float t, float b, float c, float d)
- float **InOut** (float t, float b, float c, float d)

関数詳解

float Easing::Cubic::In (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Cubic::InOut (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Cubic::Out (float *t*, float *b*, float *c*, float *d*) `[inline]`

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Easing/**easing.hpp**

Easing クラス

Easingを扱うclass
`#include <easing.hpp>`

クラス

- class **Back**
- class **Bounce**
- class **Circ**
- class **Cubic**
- class **Elastic**
- class **Expo**
- class **Linear**
- class **Quad**
- class **Quart**
- class **Quint**
- class **Sine**

公開型

- enum **Name** { **Back**, **Bounce**, **Circ**, **Cubic**, **Elastic**, **Expo**, **Linear**, **Quad**, **Quart**, **Quint**, **Sine** }
- *EasingName* enum **Mode** { **In**, **Out**, **InOut** }

EasingMode 公開メンバ関数

- float **Time** (const float duration)
  イージング用カウンタ
- bool **IsPlay** () const
  実行中か取得
- void **ResetTime** ()
  タイム初期化
- **Easing** ()
  constructor

公開変数類

- class **Easing::Linear linear**
- class **Easing::Back back**
- class **Easing::Bounce bounce**
- class **Easing::Circ circ**
- class **Easing::Cubic cubic**
- class **Easing::Elastic elastic**
- class **Easing::Expo expo**
- class **Easing::Quad quad**
- class **Easing::Quart quart**
- class **Easing::Quint quint**
- class **Easing::Sine sine**

詳解

Easingを扱うclass

各詳細はEasing表を確認

列挙型メンバ詳解

enum Easing::Mode

EasingMode
enum Mode

列挙値:

In	
Out	
InOut	

enum Easing::Name

EasingName
enum Name

列挙値:

Back	
Bounce	
Circ	
Cubic	
Elastic	
Expo	
Linear	
Quad	
Quart	
Quint	
Sine	

構築子と解体子

Easing::Easing () `[inline]`

constructor

関数詳解

bool Easing::IsPlay () const `[inline]`

実行中か取得

戻り値:
bool true 実行中

void Easing::ResetTime ()[inline]

タイム初期化

float Easing::Time (const float *duration*)[inline]

イージング用カウンタ

引数:

in	*float*	duration 設定タイム

戻り値:

現在タイム

メンバ詳解

class Easing::Back Easing::back

class Easing::Bounce Easing::bounce

class Easing::Circ Easing::circ

class Easing::Cubic Easing::cubic

class Easing::Elastic Easing::elastic

class Easing::Expo Easing::expo

class Easing::Linear Easing::linear

class Easing::Quad Easing::quad

class Easing::Quart Easing::quart

class Easing::Quint Easing::quint

class Easing::Sine Easing::sine

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Easing/**easing.hpp**

Easing::Elastic クラス

```
#include <easing.hpp>
```

公開メンバ関数

- float **In** (float t, float b, float c, float d)
- float **Out** (float t, float b, float c, float d)
- float **InOut** (float t, float b, float c, float d)

関数詳解

float Easing::Elastic::In (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Elastic::InOut (float *t*, float *b*, float *c*, float *d*) `[inline]`

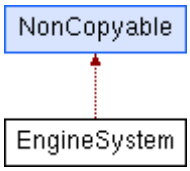
float Easing::Elastic::Out (float *t*, float *b*, float *c*, float *d*) `[inline]`

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Easing/**easing.hpp**

EngineSystem クラス

ゲームエンジン
`#include <OGsystem.h>`
EngineSystem の継承関係図



公開メンバ関数

- **EngineSystem** ()
  constructor
- **EngineSystem** (const int x, const int y, const char *name, const bool flag=false)
  constructor
- virtual **~EngineSystem** ()
  destructor
- bool **Initialize** ()
  初期化処理
- void **Update** ()
  System更新処理
- void **SetWindow** (const int x, const int y, const char *name, const bool flag=false)
  Window情報登録
- void **GetWindow** (int &x, int &y, bool &flag, **Vec2** &pos)
  Window情報取得
- void **SetWindowPos** (const **Vec2** &pos)
  Window生成位置登録
- void **SetCursorOn** (const bool flag)
  カーソル可視化設定
- void **SetIcon** (const std::string &path)
  アイコン画像設定
- void **WindowConfig** ()
  Window設定処理
- void **AllObjectStop** (const bool flag=true)
  全オブジェクトの停止設定
- void **AllUIStop** (const bool flag=true)
  全UIの停止設定
- void **AllSceneStop** (const bool flag=true)
  全シーンの停止設定
- void **AllStop** (const bool flag=true)
  全タスクの停止設定
- void **AllObjectPause** (const bool flag=true)
  全オブジェクトのポーズ設定
- void **AllScenePause** (const bool flag=true)
  全シーンのポーズ設定
- void **AllUIPause** (const bool flag=true)
  全UIのポーズ設定

- void **AllPause** (const bool flag=true)
  全タスクのポーズ設定
- void **AllObjectKill** ()
  全オブジェクトの削除命令
- void **AllUIKill** ()
  全UIの削除命令
- void **AllSceneKill** ()
  全シーンの削除命令
- void **AllKill** ()
  全タスク削除命令
- void **GameEnd** ()
  アプリケーション終了
- bool **GetEnd** () const
  終了設定を返す
- bool **GetNextWindowCreateEnable** () const
  次Window生成を行うかを返す
- void **ChengeTask** ()
  タスク変更時処理
- void **TaskGameUpdate** ()
  タスク達の更新処理
- void **SetTask** (**SceneTask** *task)
  タスクを登録する
- void **SetGameObject** (**GameObject** *object)
  GameObjectを登録する
- void **SetUI** (**UIObject** *ui)
  UIを登録する
- void **SetStartTask** (**SceneTask** *task)
  開始タスクを登録する
- void **Reset** ()
  値リセット
- void **WindowChenge** (const **Vec2** &pos, const **Vec2** &size, const char *name, const bool screen)
  WindowChenge
- void **WindowChenge** (const int x, const int y, const int w, const int h, const char *name, const bool screen)
  WindowChenge
- std::vector< **GameObject** * > **GetAllObject** () const
  登録されているオブジェクトすべてを取得する
- std::vector< **GameObject** * > **GetAllAddObject** () const
  登録予定オブジェクトすべてを取得する
- std::vector< **UIObject** * > **GetAllUI** () const
  登録されているUIすべてを取得する
- std::vector< **UIObject** * > **GetAllAddUI** () const
  登録予定UIすべてを取得する
- std::vector< **SceneTask** * > **GetAllOtherScenes** () const
  他Sceneを全て取得する
- bool **GetDeleteEngine** ()
  エンジン終了を返す
- void **SetDeleteEngine** (const bool flag)

エンジン終了を登録

- void **ShowNameAddedObject** ()
  登録されているタスクを表示する
- template<class T > T * **GetObject** (const std::string &objectName) const
  オブジェクト検索(最初の同名のオブジェクトを返す)
- template<class T > std::vector< T * > **GetObjects** (const std::string &objectName) const
  オブジェクト検索(同名すべてを返す)
- template<class T > T * **GetUI** (const std::string &tag) const
  UI検索(最初の同名のUIを返す)
- template<class T > std::vector< T * > **GetUIs** (const std::string &tag) const
  UI検索(同名すべてを返す)
- template<class T > T * **GetNowScene** () const
  NowScene取得
- template<class T > T * **GetNextScene** () const
  NextScene取得
- template<class T > std::vector< T * > **GetSceneOthers** (const std::string &name) const
  OtherScene検索

公開変数類

- **Camera2D * camera**
  カメラ2D
- **Window * window**
  window
- **FPS * fps**
  フレームレート
- **SoundManager * soundManager**
  サウンド管理
- **Input * in**
  入力管理
- bool **DebugFunction**
  デバッグ機能

詳解

ゲームエンジン

構築子と解体子

EngineSystem::EngineSystem () `[explicit]`

constructor

EngineSystem::EngineSystem (const int *x*, const int *y*, const char * *name*, const bool *flag* = `false`) `[explicit]`

constructor

引数:

in	*int*	x WindowSizeX
in	*int*	y WindowSizeY
in	*char**	name WindowName
in	*bool*	flag ScreenMode

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

destructor

関数詳解

void EngineSystem::AllKill ()

全タスク削除命令

void EngineSystem::AllObjectKill ()

全オブジェクトの削除命令

void EngineSystem::AllObjectPause (const bool *flag* = `true`)

全オブジェクトのポーズ設定

引数:

in	*bool*	flag ポーズ設定

void EngineSystem::AllObjectStop (const bool *flag* = `true`)

全オブジェクトの停止設定

引数:

in	*bool*	flag 停止設定

void EngineSystem::AllPause (const bool *flag* = `true`)

全タスクのポーズ設定

引数:

in	*bool*	flag ポーズ設定

void EngineSystem::AllSceneKill ()

全シーンの削除命令

void EngineSystem::AllScenePause (const bool *flag* = `true`)

全シーンのポーズ設定

引数:

in	*bool*	flag ポーズ設定

void EngineSystem::AllSceneStop (const bool *flag* = `true`)

全シーンの停止設定

引数:

in	*bool*	flag 停止設定

void EngineSystem::AllStop (const bool *flag* = `true`)

全タスクの停止設定

引数:

in	*bool*	flag 停止設定

void EngineSystem::AllUIKill ()

全UIの削除命令

void EngineSystem::AllUIPause (const bool *flag* = `true`)

全UIのポーズ設定

引数:

in	*bool*	flag ポーズ設定

void EngineSystem::AllUIStop (const bool *flag* = `true`)

全UIの停止設定

引数:

in	*bool*	flag 停止設定

void EngineSystem::ChengeTask ()

タスク変更時処理

void EngineSystem::GameEnd ()

アプリケーション終了

std::vector< GameObject * > EngineSystem::GetAllAddObject () const

登録予定オブジェクトすべてを取得する

戻り値:
vector<GameObject*> 登録予定全オブジェクト

std::vector< UIObject * > EngineSystem::GetAllAddUI () const

登録予定UIすべてを取得する

戻り値:
vector<UIObject*> 登録予定全UI

std::vector< GameObject * > EngineSystem::GetAllObject () const

登録されているオブジェクトすべてを取得する

戻り値:
vector<GameObject*> 登録全オブジェクト

std::vector< SceneTask * > EngineSystem::GetAllOtherScenes () const

他Sceneを全て取得する

戻り値:
vector<SceneTask*> 他Scene達

std::vector< UIObject * > EngineSystem::GetAllUI () const

登録されているUIすべてを取得する

戻り値:
vector<UIObject*> 登録全UI

bool EngineSystem::GetDeleteEngine ()

エンジン終了を返す

戻り値:

bool エンジン終了設定

bool EngineSystem::GetEnd () const

終了設定を返す

戻り値:

bool 終了設定

template<class T > T* EngineSystem::GetNextScene () const [inline]

NextScene取得

戻り値:

template<class*> 次Scene

bool EngineSystem::GetNextWindowCreateEnable () const

次Window生成を行うかを返す

戻り値:

bool trueなら再びWindowを生成する

template<class T > T* EngineSystem::GetNowScene () const [inline]

NowScene取得

戻り値:

template<class*> 現在Scene

template<class T > T* EngineSystem::GetObject (const std::string & *objectName*) const [inline]

オブジェクト検索(最初の同名のオブジェクトを返す)

引数:

in	*string*	objectName オブジェクト名

戻り値:

template<class*> 指定単体タスクclass

template<class T > std::vector<T*> EngineSystem::GetObjects (const std::string & *objectName*) const [inline]

オブジェクト検索(同名すべてを返す)

引数:

in	*string*	objectName オブジェクト名

戻り値:

vector<template<class*>> 指定複数タスクclass

template<class T > std::vector<T*> EngineSystem::GetSceneOthers (const std::string & *name*) const[inline]

OtherScene検索

引数:

in	*std::string*	name Scene名

戻り値:

vector<template class*> 該当Scene達

template<class T > T* EngineSystem::GetUI (const std::string & *tag*) const[inline]

UI検索(最初の同名のUIを返す)

引数:

in	*string*	tag UITag

戻り値:

template<class*> 指定単体タスクclass

template<class T > std::vector<T*> EngineSystem::GetUIs (const std::string & *tag*) const[inline]

UI検索(同名すべてを返す)

引数:

in	*string*	tag UItag

戻り値:

vector<template<class*>> 指定複数タスクclass

void EngineSystem::GetWindow (int & *x*, int & *y*, bool & *flag*, Vec2 & *pos*)

Window情報取得

引数:

out	*int**	x WindowSizeX
out	*int**	y WindowSizeY
out	*bool**	flag ScreenMode
out	*Vec2**	pos WindowPosition

bool EngineSystem::Initialize ()

初期化処理

戻り値:
成功でtrue

void EngineSystem::Reset ()

値リセット

void EngineSystem::SetCursorOn (const bool *flag*)

カーソル可視化設定

引数:

in	*bool*	flag 可視化設定

void EngineSystem::SetDeleteEngine (const bool *flag*)

エンジン終了を登録

引数:

in	*bool*	falg 終了設定

void EngineSystem::SetGameObject (GameObject * *object*)

GameObjectを登録する

引数:

in	*GameObject**	object **GameObject**

void EngineSystem::SetIcon (const std::string & *path*)

アイコン画像設定

引数:

in	*string*	path ファイルパスファイルパス

void EngineSystem::SetStartTask (SceneTask * *task*)

開始タスクを登録する

引数:

in	*SceneTask**	task タスク

void EngineSystem::SetTask (SceneTask * *task*)

タスクを登録する

引数:

in	*SceneTask**	task タスク

void EngineSystem::SetUI (UIObject * *ui*)

UIを登録する

引数:

in	*UIObject**	ui UI

void EngineSystem::SetWindow (const int *x*, const int *y*, const char * *name*, const bool *flag* = `false`)

Window情報登録

引数:

in	*int*	x WindowSizeX
in	*int*	y WindowSizeY
in	*char**	name WindowName
in	*bool*	flag ScreenMode

void EngineSystem::SetWindowPos (const Vec2 & *pos*)

Window生成位置登録

引数:

in	***Vec2***	pos 位置情報

void EngineSystem::ShowNameAddedObject ()

登録されているタスクを表示する

void EngineSystem::TaskGameUpdate ()

タスク達の更新処理

void EngineSystem::Update ()

System更新処理

void EngineSystem::WindowChenge (const Vec2 &　*pos*, const Vec2 &　*size*, const char *　*name*, const bool　*screen*)

WindowChenge

引数:

in	*const*	**Vec2**& pos WindowPosition
in	*const*	**Vec2**& size WindowSize
in	*char**	name WindowTitleName
in	*bool*	screen WindowMode

void EngineSystem::WindowChenge (const int　*x*, const int　*y*, const int　*w*, const int *h*, const char *　*name*, const bool　*screen*)

WindowChenge

引数:

in	*int*	x WindowPosition_x
in	*int*	y WindowPosition_y
in	*int*	w WindowSize_w
in	*int*	h WindowSize_h
in	*char**	name WindowTitleName
in	*bool*	screen WindowMode

void EngineSystem::WindowConfig ()

Window設定処理

メンバ詳解

Camera2D* EngineSystem::camera

カメラ2D

bool EngineSystem::DebugFunction

デバッグ機能

FPS* EngineSystem::fps

フレームレート

Input* EngineSystem::in

入力管理

SoundManager* EngineSystem::soundManager

サウンド管理

Window* EngineSystem::window

window

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/OG/**OGsystem.h**
- src/lib/OGSystem/OG/**OGsystem.cpp**

EventTask クラス

イベントタスク
`#include <Event.h>`

公開メンバ関数

- **EventTask** ()
  constructor
- virtual **~EventTask** ()
  destructor
- bool **Load** (const **Event** &eventType, const std::string &path)
  イベントを読み込む

詳解

イベントタスク

構築子と解体子

EventTask::EventTask () `[explicit]`

constructor

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

destructor

関数詳解

bool EventTask::Load (const Event & *eventType*, const std::string & *path*)

イベントを読み込む

引数:

in	*Event*	eventType 生成するEventの種類
in	*string*	path 読み込むEventファイルのパス

戻り値:

bool 成功true

このクラス詳解は次のファイルから抽出されました:

- src/lib/Event/**Event.h**
- src/lib/Event/**Event.cpp**

Easing::Expo クラス

```
#include <easing.hpp>
```

公開メンバ関数

- float **In** (float t, float b, float c, float d)
- float **Out** (float t, float b, float c, float d)
- float **InOut** (float t, float b, float c, float d)

関数詳解

float Easing::Expo::In (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Expo::InOut (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Expo::Out (float *t*, float *b*, float *c*, float *d*) `[inline]`

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Easing/**easing.hpp**

Font クラス

フォントの描画class
`#include <TextureFont.h>`

公開メンバ関数

- **Font** ()
  constructor
- virtual **~Font** ()
  destructor
- void **Draw** (const std::string &text, const **Vec2** &pos, const int fontSize, const **Color** &color={ 1, 1, 1, 1 })
  フォント描画
- void **SetTexture** (const std::string &path)
  登録画像の変更

詳解

フォントの描画class

現在対応しているもの

アルファベット

ひらがな

カタカナ

数字

構築子と解体子

Font::Font () `[explicit]`

constructor

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

destructor

関数詳解

void Font::Draw (const std::string & *text*, const Vec2 & *pos*, const int *fontSize*, const Color & *color* = `{ 1,1,1,1 }`)

フォント描画

引数:

in	*string*	text 描画文字列
in	***Vec2***	pos 描画位置
in	*int*	fontSize 描画文字サイズ
in	***Color***	color 色指定

void Font::SetTexture (const std::string & *path*)

登録画像の変更

引数:

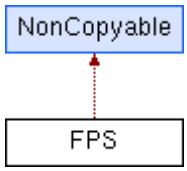
in	*string*	path ファイルパス

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Font/**TextureFont.h**
- src/lib/OGSystem/Font/**TextureFont.cpp**

FPS クラス

フレームレートを計算,制限するclass
`#include <FPS.h>`
FPS の継承関係図



公開メンバ関数

- **FPS** ()
  constructor
- virtual **~FPS** ()
  destructor
- void **Update** ()
  更新処理
- void **SetFrameRate** (const int rate)
- bool **FrameCheck** ()
  フレームチェック

詳解

フレームレートを計算,制限するclass

既存GameEngineを使用している場合はUpdateをDEBUG時のみ行う

構築子と解体子

FPS::FPS () `[explicit]`

constructor

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

destructor

関数詳解

bool FPS::FrameCheck ()

フレームチェック

戻り値:

秒間フレームに達しているならtrue

void FPS::SetFrameRate (const int *rate*)

フレームレート指定

引数:

in	*int*	rate frame rate

void FPS::Update ()

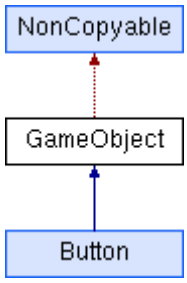
更新処理

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/FPS/**FPS.h**
- src/lib/OGSystem/FPS/**FPS.cpp**

GameObject クラス

GameObject
`#include <GameObject.h>`
GameObject の継承関係図



公開メンバ関数

- **GameObject** ()
  constructor
- **GameObject** (const **Objform** &form, const std::string &tag, const **Vec2** &pos, const **Vec2** &scale, const float angle=0.0f)
  constructor
- virtual **~GameObject** ()
  destructor
- void **Init** (const **Objform** &form, const std::string &tag, const **Vec2** &pos, const **Vec2** &scale, const float angle=0.0f)
  オブジェクト初期化
- bool **Hit** (**GameObject** *object)
  当たり判定
- bool **Hit** (**CollisionBase** *object)
  当たり判定
- virtual void **Update** ()
  更新処理
- virtual void **Pause** ()
  停止処理
- virtual void **Render2D** ()
  描画処理
- void **UpdateManager** ()
  更新管理
- void **RenderManager** ()
  描画管理
- void **LineDraw** (const float lineWidth=1.0f)
  当たり判定確認
- bool **IsObjectDistanceCheck** (**GameObject** *object)
  大まかな範囲の判定を返す
- bool **IsObjectDistanceCheck** (**CollisionCircle** *object)
  大まかな範囲の判定を返す
- void **LineDistanceDraw** (const float lineWidth=1.0f)
  大きな判定範囲を表示する
- void **CollisionConfig** (const bool flag)

当たり判定設定

- bool **IsCollision** () const
  当たり判定設定の確認
- void **SetPosition** (const **Vec2** &pos)
  位置設定
- void **SetPosition** (const float x, const float y)
  位置設定
- **Vec2 GetPosition** () const
  位置情報取得
- void **SetScale** (const **Vec2** &scale)
  拡縮設定
- void **SetScale** (const float x, const float y)
  拡縮設定
- **Vec2 GetScale** () const
  拡縮情報取得
- void **SetRotate** (const float angle)
  回転設定
- float **GetRotate** () const
  回転情報取得
- void **SetRadius** (const **Vec2** &radius)
  判定倍率
- void **SetRadius** (const float x, const float y)
  判定倍率
- **Vec2 GetRadius** () const
  判定倍率情報取得
- void **SetTag** (const std::string &name)
  タグ設定
- std::string **GetTag** () const
  タグ情報取得
- void **SetMass** (const float mass)
  質量設定
- float **GetMass** () const
  質量情報取得
- void **Kill** ()
  オブジェクトを削除する
- void **CancelKill** ()
  削除命令をキャンセルする
- void **SetPause** (const bool flag=true)
  Pause設定
- void **SetStop** (const bool flag=true)
  停止設定
- void **SetAllStop** (const bool flag=true)
  全機能停止
- **GO::Mode GetMode** () const
  現状の状態を返す
- bool **ModeCheck** (const **GO::Mode** &mode) const
  モード確認 [in] Mode mode 比較対象
- **Objform Getform** () const

現在のformを返す

- void **SetDrawOrder** (const unsigned int order)
  描画順指定 高い方が前に描画される
- unsigned int **GetDrawOrder** () const
  描画順を取得

詳解

GameObject

当たり判定や更新、描画処理を行うことが出来る

構築子と解体子

GameObject::GameObject () `[explicit]`

constructor

GameObject::GameObject (const Objform & *form*, const std::string & *tag*, const Vec2 & *pos*, const Vec2 & *scale*, const float *angle* = `0.0f`) `[explicit]`

constructor

引数:

in	*Objform*	form オブジェクトの状態
in	*string*	tag オブジェクトタグ名
in	***Vec2***	pos 位置
in	***Vec2***	scale 拡縮
in	*float*	angle 回転値

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

destructor

関数詳解

void GameObject::CancelKill ()

削除命令をキャンセルする

void GameObject::CollisionConfig (const bool *flag*)

当たり判定設定

引数:

in	*bool*	flag 判定有無

unsigned int GameObject::GetDrawOrder () const

描画順を取得

Objform GameObject::Getform () const

現在のformを返す

戻り値:

Objform 現在のform

float GameObject::GetMass () const

質量情報取得

戻り値:

float mass

GO::Mode GameObject::GetMode () const

現状の状態を返す

戻り値:

Mode 状態

Vec2 GameObject::GetPosition () const

位置情報取得

戻り値:

Vec2 Position

Vec2 GameObject::GetRadius () const

判定倍率情報取得

戻り値:

[in] **Vec2** radius

float GameObject::GetRotate () const

回転情報取得

戻り値:

float angle

Vec2 GameObject::GetScale () const

拡縮情報取得

戻り値:

Vec2 Scale

std::string GameObject::GetTag () const

タグ情報取得

戻り値:

string objecttag

bool GameObject::Hit (GameObject * *object*)

当たり判定

引数:

in	*GameObjet**	object 相手のオブジェクト

戻り値:

bool hit true

bool GameObject::Hit (CollisionBase * *object*)

当たり判定

引数:

in	*CollisionBase**	object 相手のオブジェクト

戻り値:

bool hit true

void GameObject::Init (const Objform & *form*, const std::string & *tag*, const Vec2 & *pos*, const Vec2 & *scale*, const float *angle* = 0.0f)

オブジェクト初期化

引数:

in	*Objform*	form オブジェクトの状態
in	*string*	tag オブジェクトタグ名
in	***Vec2***	pos 位置

in	***Vec2***	scale 拡縮
in	*float*	angle 回転値

bool GameObject::IsCollision () const

当たり判定設定の確認

戻り値:

bool 判定有無

bool GameObject::IsObjectDistanceCheck (GameObject * *object*)

大まかな範囲の判定を返す

引数:

in	*GameObject**	object 相手のオブジェクト

戻り値:

bool 判定内であればtrue

bool GameObject::IsObjectDistanceCheck (CollisionCircle * *object*)

大まかな範囲の判定を返す

引数:

in	*CollisionCircle**	object 相手のオブジェクト

戻り値:

bool 判定内であればtrue

void GameObject::Kill ()

オブジェクトを削除する

void GameObject::LineDistanceDraw (const float *lineWidth* = `1.0f`)

大きな判定範囲を表示する

引数:

in	*float*	lineWidth Lineの太さ

void GameObject::LineDraw (const float *lineWidth* = `1.0f`)

当たり判定確認

引数:

in	*float*	lineWidth Lineの太さ

bool GameObject::ModeCheck (const GO::Mode & *mode*) const

モード確認 [in] Mode mode 比較対象

戻り値:

bool 比較対象と同じならtrue

void GameObject::Pause () `[virtual]`

停止処理

void GameObject::Render2D () `[virtual]`

描画処理

void GameObject::RenderManager ()

描画管理

void GameObject::SetAllStop (const bool *flag* = `true`)

全機能停止

引数:

in	*bool*	flag trueで全停止

void GameObject::SetDrawOrder (const unsigned int *order*)

描画順指定 高い方が前に描画される

引数:

in	*unsigned*	int order 描画順値

void GameObject::SetMass (const float *mass*)

質量設定

引数:

in	*float*	mass 質量

void GameObject::SetPause (const bool *flag* = `true`)

Pause設定

引数:

in	*bool*	flag trueでPause化

void GameObject::SetPosition (const Vec2 & *pos*)

位置設定

引数:

in	***Vec2***	pos 位置

void GameObject::SetPosition (const float *x*, const float *y*)

位置設定

引数:

in	*float*	x 位置X座標
in	*float*	y 位置Y座標

void GameObject::SetRadius (const Vec2 & *radius*)

判定倍率

引数:

in	***Vec2***	radius 倍率

void GameObject::SetRadius (const float *x*, const float *y*)

判定倍率

引数:

in	*float*	x x座標倍率
in	*float*	y y座標倍率

void GameObject::SetRotate (const float *angle*)

回転設定

引数:

in	*float*	angle 回転角度

void GameObject::SetScale (const Vec2 & *scale*)

拡縮設定

引数:

in	***Vec2***	scale 拡縮

void GameObject::SetScale (const float *x*, const float *y*)

拡縮設定

引数:

in	*float*	x 拡縮X
in	*float*	y 拡縮Y

void GameObject::SetStop (const bool *flag* = `true`)

停止設定

引数:

in	*bool*	flag trueで停止

void GameObject::SetTag (const std::string & *name*)

タグ設定

引数:

in	*string*	name tagname

void GameObject::Update () `[virtual]`

更新処理

void GameObject::UpdateManager ()

更新管理

このクラス詳解は次のファイルから抽出されました:

- src/lib/Object/**GameObject.h**
- src/lib/Object/**GameObject.cpp**

Input::GamePad クラス

ゲームパッド入力
`#include <Input.h>`

公開型

- enum **Pad** { **BUTTON_A**, **BUTTON_B**, **BUTTON_X**, **BUTTON_Y**, **BUTTON_L1**, **BUTTON_R1**, **BUTTON_BACK**, **BUTTON_START**, **BUTTON_L3**, **BUTTON_R3**, **BUTTON_U**, **BUTTON_R**, **BUTTON_D**, **BUTTON_L** }
- *仮装コントローラの入力設定* enum **AXIS** { **AXIS_LEFT_X**, **AXIS_LEFT_Y**, **AXIS_RIGHT_X**, **AXIS_RIGHT_Y**, **AXIS_R2**, **AXIS_L2**, **AXIS_BUTTON_NUM** }
- *仮装コントローラの入力設定* enum **AXISBUTTON** { **LSTICK_LEFT**, **LSTICK_RIGHT**, **LSTICK_UP**, **LSTICK_DOWN**, **RSTICK_LEFT**, **RSTICK_RIGHT**, **RSTICK_UP**, **RSTICK_DOWN**, **BUTTON_R2**, **BUTTON_L2**, **STICK_NUM** }

仮装コントローラの入力設定 公開メンバ関数

- **GamePad** (const int id)
  constructor
- bool **on** (const int index) const
  押している判定を返す
- bool **down** (const int index) const
  押した瞬間の判定を返す
- bool **up** (const int index) const
  離した瞬間の判定を返す
- bool **EitherDown** () const
  全てのdown入力のうち1つでも判定があるか調べる
- bool **EitherOn** () const
  全てのon入力のうち1つでも判定があるか調べる
- bool **EitherUp** () const
  全てのup入力のうち1つでも判定があるか調べる
- float **axis** (const int index) const
  指定値のスティックの傾きを返す
- bool **axis_on** (const int index) const
  指定スティックの押し状態を返す
- bool **axis_down** (const int index) const
  指定スティックの押し状態を返す
- bool **axis_up** (const int index) const
  指定スティックの押し状態を返す
- bool **isPresent** () const
  ゲームパッドの有無を返す
- void **Update** ()
  入力状況の更新
- void **Initialize** ()
  各値の初期化
- void **Reset** ()
  入力状態のリセット
- bool **registAxisButton** (const float axis_threshold_)
  スティックの範囲外処理

- const char * **GetName** () const
  ゲームパッド名を返す

詳解

ゲームパッド入力

列挙型メンバ詳解

enum Input::GamePad::AXIS

仮装コントローラの入力設定

enum AXIS

列挙値:

AXIS_LEFT_X	左スティックX値
AXIS_LEFT_Y	左スティックY値
AXIS_RIGHT_X	右スティックX値
AXIS_RIGHT_Y	右スティックY値
AXIS_R2	R2
AXIS_L2	L2
AXIS_BUTTON_NUM	ButtonNumber

enum Input::GamePad::AXISBUTTON

仮装コントローラの入力設定

enum AXISBUTTON

列挙値:

LSTICK_LEFT	左スティック左入力
LSTICK_RIGHT	左スティック右入力
LSTICK_UP	左スティック上入力
LSTICK_DOWN	左スティック下入力

RSTICK_LEFT	右スティック左入力
RSTICK_RIGHT	右スティっク右入力
RSTICK_UP	右スティック上入力
RSTICK_DOWN	右スティック下入力
BUTTON_R2	R2
BUTTON_L2	L2
STICK_NUM	スティック数

enum Input::GamePad::Pad

仮装コントローラの入力設定

enum Pad

列挙値:

BUTTON_A	1
BUTTON_B	2
BUTTON_X	3
BUTTON_Y	4
BUTTON_L1	5
BUTTON_R1	6
BUTTON_BACK	7
BUTTON_START	8
BUTTON_L3	9
BUTTON_R3	10
BUTTON_U	11
BUTTON_R	12

BUTTON_D	13
BUTTON_L	14

構築子と解体子

Input::GamePad::GamePad (const int *id*)[explicit]

constructor

引数:

in	*int*	id ゲームパッド番号

関数詳解

float Input::GamePad::axis (const int *index*) const

指定値のスティックの傾きを返す

引数:

in	*int*	index スティック指定

戻り値:

float 傾き度(0~1)

bool Input::GamePad::axis_down (const int *index*) const

指定スティックの押し状態を返す

引数:

in	*int*	index スティック指定

戻り値:

bool 指定側に倒された瞬間であればtrue

bool Input::GamePad::axis_on (const int *index*) const

指定スティックの押し状態を返す

引数:

in	*int*	index スティック指定

戻り値:

bool 指定側に倒れていればtrue

bool Input::GamePad::axis_up (const int *index*) const

指定スティックの押し状態を返す

引数:

in	*int*	index スティック指定

戻り値:

bool 指定側から上がった瞬間であればtrue

bool Input::GamePad::down (const int *index*) const

押した瞬間の判定を返す

引数:

in	*int*	index 判定を行いたい入力番号

戻り値:

bool 押した瞬間であればtrue

bool Input::GamePad::EitherDown () const

全てのdown入力のうち１つでも判定があるか調べる

戻り値:

1つ以上入力されているとtrue

bool Input::GamePad::EitherOn () const

全てのon入力のうち１つでも判定があるか調べる

戻り値:

1つ以上入力されているとtrue

bool Input::GamePad::EitherUp () const

全てのup入力のうち１つでも判定があるか調べる

戻り値:

1つ以上入力されているとtrue

const char * Input::GamePad::GetName () const

ゲームパッド名を返す

戻り値:

char* ゲームパッド名

void Input::GamePad::Initialize ()

各値の初期化

bool Input::GamePad::isPresent () const

ゲームパッドの有無を返す

戻り値:

bool 存在すればtrue

bool Input::GamePad::on (const int *index*) const

押している判定を返す

引数:

in	*int*	index 判定を行いたい入力番号

戻り値:

bool 押していればtrue

bool Input::GamePad::registAxisButton (const float *axis_threshold_*)

スティックの範囲外処理

引数:

in	*float*	axis_threshold_ 区切る値

戻り値:

成功true

void Input::GamePad::Reset ()

入力状態のリセット

bool Input::GamePad::up (const int *index*) const

離した瞬間の判定を返す

引数:

in	*int*	index 判定を行いたい入力番号

戻り値:

bool 離した瞬間であればtrue

void Input::GamePad::Update ()

入力状況の更新

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Input/**Input.h**
- src/lib/OGSystem/Input/**Input.cpp**

Wav::Info 構造体

```
#include <Audio.h>
```

公開変数類

- **u_int id**
- **u_int ch**
- **u_int sample_rate**
- **u_int bit**
- **u_int size**

メンバ詳解

u_int Wav::Info::bit

u_int Wav::Info::ch

u_int Wav::Info::id

u_int Wav::Info::sample_rate

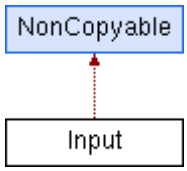
u_int Wav::Info::size

この構造体詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Audio/**Audio.h**

Input クラス

ゲームパッド、キーボード、マウスの入力を扱うclass
`#include <Input.h>`
Input の継承関係図



クラス

- class **GamePad**
  ゲームパッド入力
- struct **InputData**
  ゲームパッドとキーボードを区別する
- class **KeyBoard**
  キーボード入力
- class **Mouse**
  マウス入力

公開型

- enum **in** { **B1**, **B2**, **B3**, **B4**, **CD**, **CU**, **CR**, **CL**, **L1**, **R1**, **D1**, **D2**, **SR**, **SL**, **LD**, **LU**, **LR**, **LL**, **RD**, **RU**, **RR**, **RL**, **L2**, **R2** }

入力用仮想入力 公開メンバ関数

- void **Inputinit** (GLFWwindow *w)
  入力初期化
- bool **on** (const int index, const int padNum=0) const
  押している判定を返す
- bool **down** (const int index, const int padNum=0) const
  押した瞬間の判定を返す
- bool **up** (const int index, const int padNum=0) const
  離した瞬間の判定を返す
- float **axis** (const int index, const int padNum=0) const
  指定値のスティックの傾きを返す
- void **registAxis** (const float regist)
  全てのゲームパッドのスティックの傾き範囲を制限する
- void **Update** ()
  入力状況の更新
- bool **EitherDown** () const
  全てのdown入力のうち１つでも判定があるか調べる
- bool **EitherOn** () const
  全てのon入力のうち１つでも判定があるか調べる
- bool **EitherUp** () const
  全てのup入力のうち１つでも判定があるか調べる
- virtual **~Input** ()
  destructor

公開変数類

- std::vector< **GamePad** * > **pad**
  ゲームパッド配列
- **KeyBoard** * **key**
  キーボード
- **Mouse** * **mouse**
  マウス
- bool **pad_Connection**
  ゲームパッドの存在有無

詳解

ゲームパッド、キーボード、マウスの入力を扱うclass

既存GameEngineを使用している場合は自動で生成される

列挙型メンバ詳解

enum Input::in

入力用仮想入力

enum in

列挙値:

B1	
B2	
B3	
B4	
CD	
CU	
CR	
CL	
L1	
R1	
D1	
D2	
SR	
SL	
LD	
LU	
LR	
LL	
RD	
RU	
RR	
RL	
L2	
R2	

構築子と解体子

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

destructor

関数詳解

float Input::axis (const int *index*, const int *padNum* = 0) const

指定値のスティックの傾きを返す

引数:

in	*int*	index スティック指定
in	*int*	padNum 判定を行うゲームパッドの番号

戻り値:

float 傾き度(0~1)

bool Input::down (const int *index*, const int *padNum* = 0) const

押した瞬間の判定を返す

引数:

in	*int*	index 判定を行いたい入力番号
in	*int*	padNum 判定を行うゲームパッドの番号

戻り値:

bool 押した瞬間であればtrue

bool Input::EitherDown () const

全てのdown入力のうち１つでも判定があるか調べる

戻り値:

bool 1つ以上入力されているとtrue

bool Input::EitherOn () const

全てのon入力のうち１つでも判定があるか調べる

戻り値:

bool 1つ以上入力されているとtrue

bool Input::EitherUp () const

全てのup入力のうち１つでも判定があるか調べる

戻り値:

bool 1つ以上入力されているとtrue

void Input::Inputinit (GLFWwindow * *w*)

入力初期化

引数:

in	*GLFWWindow**	w 指定するWindowのポインタ

bool Input::on (const int *index*, const int *padNum* = 0) const

押している判定を返す

引数:

in	*int*	index 判定を行いたい入力番号
in	*int*	padNum 判定を行うゲームパッドの番号

戻り値:

bool 押していればtrue

void Input::registAxis (const float *regist*)

全てのゲームパッドのスティックの傾き範囲を制限する

引数:

in	*float*	regist 指定値

bool Input::up (const int *index*, const int *padNum* = 0) const

離した瞬間の判定を返す

引数:

in	*int*	index 判定を行いたい入力番号
in	*int*	padNum 判定を行うゲームパッドの番号

戻り値:

bool 離した瞬間であればtrue

void Input::Update ()

入力状況の更新

メンバ詳解

KeyBoard* Input::key

キーボード

Mouse* Input::mouse

マウス

std::vector<GamePad*> Input::pad

ゲームパッド配列

bool Input::pad_Connection

ゲームパッドの存在有無

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Input/**Input.h**
- src/lib/OGSystem/Input/**Input.cpp**

Input::InputData 構造体

ゲームパッドとキーボードを区別する
`#include <Input.h>`

公開変数類

- int **button**
  ゲームパッドのボタン
- int **key**
  キーボードのキー

詳解

ゲームパッドとキーボードを区別する

メンバ詳解

int Input::InputData::button

ゲームパッドのボタン

int Input::InputData::key

キーボードのキー

この構造体詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Input/**Input.h**

Input::KeyBoard クラス

キーボード入力
`#include <Input.h>`

公開型

- enum **Key** { **A**, **S**, **D**, **W**, **Q**, **E**, **Z**, **X**, **C**, **R**, **F**, **V**, **T**, **G**, **B**, **Y**, **H**, **N**, **U**, **J**, **M**, **I**, **K**, **O**, **L**, **P**, **SPACE**, **ENTER**, **ESCAPE**, **UP**, **DOWN**, **LEFT**, **RIGHT** }

キーボードの仮装キー設定 公開メンバ関数

- **KeyBoard** ()
  constructor
- bool **up** (const int index) const
  離した瞬間の判定を返す
- bool **down** (const int index) const
  押した瞬間の判定を返す
- bool **on** (const int index) const
  押している判定を返す
- bool **EitherDown** () const
  全てのdown入力のうち1つでも判定があるか調べる
- bool **EitherOn** () const
  全てのon入力のうち1つでも判定があるか調べる
- bool **EitherUp** () const
  全てのup入力のうち1つでも判定があるか調べる
- void **Update** ()
  入力状況の更新
- void **SetWindow** (GLFWwindow *w)
  反映させるWindowを登録する

公開変数類

- bool **isPresent**
  キーボードの有無
- std::vector< **u_char** > **button_on**
  buttonのonを格納する変数
- std::vector< **u_char** > **button_down**
  buttonのdownを格納する変数
- std::vector< **u_char** > **button_up**
  buttonのupを格納する変数

詳解

キーボード入力

列挙型メンバ詳解

enum Input::KeyBoard::Key

キーボードの仮装キー設定

enum Key

列挙値:

A	
S	
D	
W	
Q	
E	
Z	
X	
C	
R	
F	
V	
T	
G	
B	
Y	
H	
N	
U	
J	
M	
I	
K	
O	
L	
P	
SPACE	
ENTER	
ESCAPE	
UP	
DOWN	
LEFT	
RIGHT	

構築子と解体子

Input::KeyBoard::KeyBoard () `[explicit]`

constructor

関数詳解

bool Input::KeyBoard::down (const int *index*) const

押した瞬間の判定を返す

引数:

in	*int*	index 判定を行いたい入力番号

戻り値:

bool 押した瞬間であればtrue

bool Input::KeyBoard::EitherDown () const

全てのdown入力のうち１つでも判定があるか調べる

戻り値:

1つ以上入力されているとtrue

bool Input::KeyBoard::EitherOn () const

全てのon入力のうち１つでも判定があるか調べる

戻り値:

1つ以上入力されているとtrue

bool Input::KeyBoard::EitherUp () const

全てのup入力のうち１つでも判定があるか調べる

戻り値:

1つ以上入力されているとtrue

bool Input::KeyBoard::on (const int *index*) const

押している判定を返す

引数:

in	*int*	index 判定を行いたい入力番号

戻り値:

bool 押していればtrue

void Input::KeyBoard::SetWindow (GLFWwindow * *w*)

反映させるWindowを登録する

引数:

in	*GLFWWindow**	w Windowのポインタ

bool Input::KeyBoard::up (const int *index*) const

離した瞬間の判定を返す

引数:

in	*int*	index 判定を行いたい入力番号

戻り値:

bool 離した瞬間であればtrue

void Input::KeyBoard::Update ()

入力状況の更新

メンバ詳解

std::vector<u_char> Input::KeyBoard::button_down

buttonのdownを格納する変数

std::vector<u_char> Input::KeyBoard::button_on

buttonのonを格納する変数

std::vector<u_char> Input::KeyBoard::button_up

buttonのupを格納する変数

bool Input::KeyBoard::isPresent

キーボードの有無

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Input/**Input.h**
- src/lib/OGSystem/Input/**Input.cpp**

Easing::Linear クラス

```
#include <easing.hpp>
```

公開メンバ関数

- float **None** (float t, float b, float c, float d)
- float **In** (float t, float b, float c, float d)
- float **Out** (float t, float b, float c, float d)
- float **InOut** (float t, float b, float c, float d)

関数詳解

float Easing::Linear::In (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Linear::InOut (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Linear::None (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Linear::Out (float *t*, float *b*, float *c*, float *d*) `[inline]`

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Easing/**easing.hpp**

Mat4 クラス

2*2行列
`#include <OGlib.h>`

公開メンバ関数

- **Mat4** (const float, const float, const float, const float)
- **Mat4** (const int, const int, const int, const int)

公開変数類

- float **mat4** [4]

詳解

2*2行列

構築子と解体子

Mat4::Mat4 (const float *ex*, const float *ey*, const float *ez*, const float *ew*)

Mat4::Mat4 (const int *ex*, const int *ey*, const int *ez*, const int *ew*)

メンバ詳解

float Mat4::mat4[4]

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/OG/**OGlib.h**
- src/lib/OGSystem/OG/**OGlib.cpp**

Mat4x4 クラス

4*4行列
`#include <OGlib.h>`

公開メンバ関数

- **Mat4x4** (const float ex, const float ey, const float ez, const float ew, const float sx, const float sy, const float sz, const float sw, const float dx, const float dy, const float dz, const float dw, const float rx, const float ry, const float rz, const float rw)
- **Mat4x4** (const int ex, const int ey, const int ez, const int ew, const int sx, const int sy, const int sz, const int sw, const int dx, const int dy, const int dz, const int dw, const int rx, const int ry, const int rz, const int rw)

公開変数類

- float **mat4** [16]

詳解

4*4行列

構築子と解体子

Mat4x4::Mat4x4 (const float *ex*, const float *ey*, const float *ez*, const float *ew*, const float *sx*, const float *sy*, const float *sz*, const float *sw*, const float *dx*, const float *dy*, const float *dz*, const float *dw*, const float *rx*, const float *ry*, const float *rz*, const float *rw*)

Mat4x4::Mat4x4 (const int *ex*, const int *ey*, const int *ez*, const int *ew*, const int *sx*, const int *sy*, const int *sz*, const int *sw*, const int *dx*, const int *dy*, const int *dz*, const int *dw*, const int *rx*, const int *ry*, const int *rz*, const int *rw*)

メンバ詳解

float Mat4x4::mat4[16]

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/OG/**OGlib.h**
- src/lib/OGSystem/OG/**OGlib.cpp**

Input::Mouse クラス

マウス入力
`#include <Input.h>`

公開型

- enum **Mouse_** { **LEFT**, **RIGHT**, **CENTER**, **BUTTON_4**, **BUTTON_5**, **BUTTON_6**, **BUTTON_7**, **BUTTON_8** }

マウスの入力設定 公開メンバ関数

- **Mouse** ()
  constructor
- virtual **~Mouse** ()
  destructor
- void **Update** ()
  入力状況の更新
- void **SetWindow** (GLFWwindow *w)
  反映させるWindowを登録する
- **Vec2 GetPos** () const
  Windowからのマウスの位置を返す
- bool **on** (const int index) const
  押している判定を返す
- bool **down** (const int index) const
  押した瞬間の判定を返す
- bool **up** (const int index) const
  離した瞬間の判定を返す
- bool **EitherDown** () const
  全てのdown入力のうち1つでも判定があるか調べる
- bool **EitherOn** () const
  てのon入力のうち1つでも判定があるか調べる
- bool **EitherUp** () const
  全てのup入力のうち1つでも判定があるか調べる
- **Vec2 GetScroll** () const
  マウスのホイール値を返す
- void **ResetMouse** ()
  マウスの入力状況をリセットする
- **CollisionPointer** * **GetCollision** () const
  マウスの判定を取得する

公開変数類

- bool **isPresent**
  マウスの有無

詳解

マウス入力

列挙型メンバ詳解

enum Input::Mouse::Mouse_

マウスの入力設定

enum Mouse_

列挙値:

LEFT	右
RIGHT	左
CENTER	中心
BUTTON_4	
BUTTON_5	
BUTTON_6	
BUTTON_7	
BUTTON_8	

構築子と解体子

Input::Mouse::Mouse () `[explicit]`

constructor

Input::Mouse::~Mouse () `[virtual]`

destructor

関数詳解

bool Input::Mouse::down (const int *index*) const

押した瞬間の判定を返す

引数:

in	*int*	index 判定を行いたい入力番号　押した瞬間であればtrue

bool Input::Mouse::EitherDown () const

全てのdown入力のうち１つでも判定があるか調べる

戻り値:

1つ以上入力されているとtrue

bool Input::Mouse::EitherOn () const

てのon入力のうち１つでも判定があるか調べる

戻り値:

1つ以上入力されているとtrue

bool Input::Mouse::EitherUp () const

全てのup入力のうち１つでも判定があるか調べる

戻り値:

1つ以上入力されているとtrue

CollisionPointer * Input::Mouse::GetCollision () const

マウスの判定を取得する

戻り値:

CollisionPointer* マウスのCollision

Vec2 Input::Mouse::GetPos () const

Windowからのマウスの位置を返す

戻り値:

Vec2 マウスの位置

Vec2 Input::Mouse::GetScroll () const

マウスのホイール値を返す

戻り値:

Vec2 ホイールの値

bool Input::Mouse::on (const int *index*) const

押している判定を返す

引数:

in	*int*	index 判定を行いたい入力番号　押していればtrue

void Input::Mouse::ResetMouse ()

マウスの入力状況をリセットする

void Input::Mouse::SetWindow (GLFWwindow * *w*)

反映させるWindowを登録する

引数:

in	*GLFWWindow**	w Windowのポインタ

bool Input::Mouse::up (const int *index*) const

離した瞬間の判定を返す

引数:

in	*int*	index 判定を行いたい入力番号　離した瞬間であればtrue

void Input::Mouse::Update ()

入力状況の更新

メンバ詳解

bool Input::Mouse::isPresent

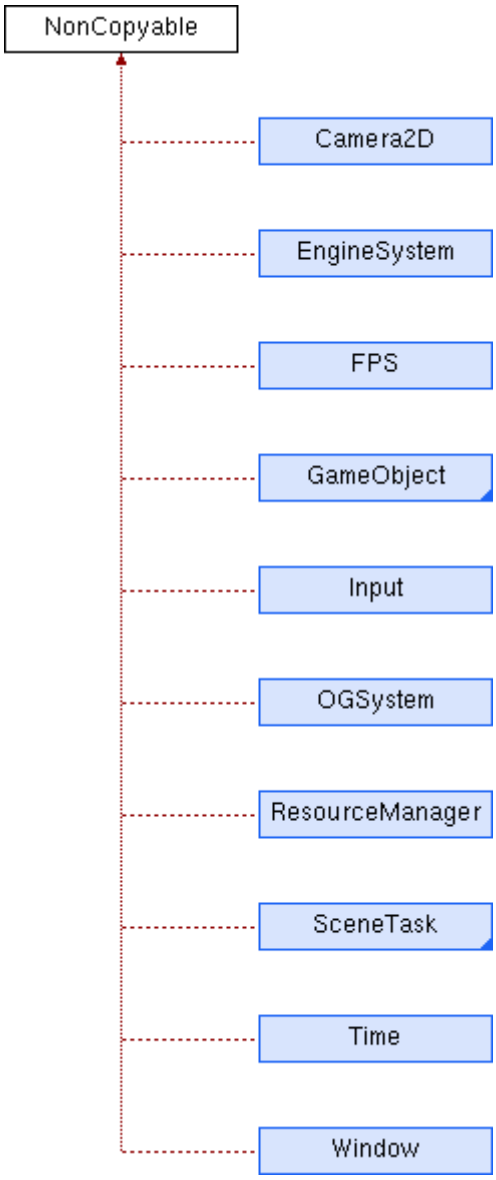
マウスの有無

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Input/**Input.h**
- src/lib/OGSystem/Input/**Input.cpp**

NonCopyable クラス

コピーを禁止するclass
`#include <OGlib.h>`
NonCopyable の継承関係図



限定公開メンバ関数

- **NonCopyable** ()
- **~NonCopyable** ()

詳解

コピーを禁止するclass

このclassを継承したclassはコピーコンストラクタと代入演算を禁止されます

構築子と解体子

NonCopyable::NonCopyable () [inline], [protected]

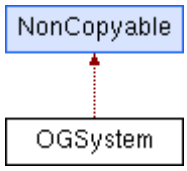
NonCopyable::~NonCopyable () [inline], [protected]

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/OG/**OGlib.h**

OGSystem クラス

System
`#include <System.h>`
OGSystem の継承関係図



公開メンバ関数

- **OGSystem** ()
  constructor
- bool **Create** ()
  システム生成
- bool **LibConfirmation** ()
  更新チェック
- virtual **~OGSystem** ()
  destructor

詳解

System

既存GameEngineClassを使用する場合OGSystem

使用しない場合_OGSystem(こちら用は未制作)

構築子と解体子

OGSystem::OGSystem () `[explicit]`

constructor

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

destructor

関数詳解

bool OGSystem::Create ()

システム生成

戻り値:

bool 成功true

bool OGSystem::LibConfirmation ()

更新チェック

戻り値:

bool 続行true

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/OG/**System.h**
- src/lib/OGSystem/OG/**System.cpp**

OGTK クラス

Engine生成に関する処理を行うclass
`#include <OGTask.h>`

公開メンバ関数

- void **Init** ()
  GammeEngineの初期化時に設定したい処理を行う初期化関数
- void **StartTaskObject** ()
  開始時に生成したいタスクを指定する関数
- virtual **~OGTK** ()
  destructor

詳解

Engine生成に関する処理を行うclass

構築子と解体子

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

destructor

関数詳解

void OGTK::Init ()

GammeEngineの初期化時に設定したい処理を行う初期化関数

void OGTK::StartTaskObject ()

開始時に生成したいタスクを指定する関数

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/OG/**OGTask.h**
- src/lib/OGSystem/OG/**OGTask.cpp**

OrderCheck クラス

描画順を管理するclass
`#include <OGsystem.h>`

公開メンバ関数

- **OrderCheck** ()
  constructor

公開変数類

- int **id**
  ObjectID
- unsigned int **order_s**
  描画順

詳解

描画順を管理するclass

構築子と解体子

OrderCheck::OrderCheck () `[inline]`, `[explicit]`

constructor

メンバ詳解

int OrderCheck::id

ObjectID

unsigned int OrderCheck::order_s

描画順

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/OG/**OGsystem.h**

Easing::Quad クラス

```
#include <easing.hpp>
```

公開メンバ関数

- float **In** (float t, float b, float c, float d)
- float **Out** (float t, float b, float c, float d)
- float **InOut** (float t, float b, float c, float d)

関数詳解

float Easing::Quad::In (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Quad::InOut (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Quad::Out (float *t*, float *b*, float *c*, float *d*) `[inline]`

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Easing/**easing.hpp**

Easing::Quart クラス

```
#include <easing.hpp>
```

公開メンバ関数

- float **In** (float t, float b, float c, float d)
- float **Out** (float t, float b, float c, float d)
- float **InOut** (float t, float b, float c, float d)

関数詳解

float Easing::Quart::In (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Quart::InOut (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Quart::Out (float *t*, float *b*, float *c*, float *d*) `[inline]`

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Easing/**easing.hpp**

Easing::Quint クラス

```
#include <easing.hpp>
```

公開メンバ関数

- float **In** (float t, float b, float c, float d)
- float **Out** (float t, float b, float c, float d)
- float **InOut** (float t, float b, float c, float d)

関数詳解

float Easing::Quint::In (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Quint::InOut (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Quint::Out (float *t*, float *b*, float *c*, float *d*) `[inline]`

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Easing/**easing.hpp**

ResourceLoad クラス

リソースを読み込むイベントclass
`#include <ResourceLoad.h>`

公開メンバ関数

- **ResourceLoad** (std::ifstream &ifs)
  constructor
- virtual **~ResourceLoad** ()
  destructor

詳解

リソースを読み込むイベントclass

構築子と解体子

ResourceLoad::ResourceLoad (std::ifstream & *ifs*) `[explicit]`

constructor

引数:

in	*ifstream*	ifs ファイルデータ

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

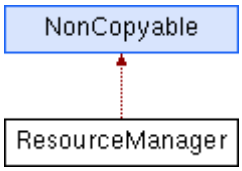
destructor

このクラス詳解は次のファイルから抽出されました:

- src/lib/Event/ResourceLoad/**ResourceLoad.h**
- src/lib/Event/ResourceLoad/**ResourceLoad.cpp**

ResourceManager クラス

リソースを生成、解放、管理を行うclass
`#include <ResourceManager.h>`
ResourceManager の継承関係図



公開メンバ関数

- void **SetSoundData** (const std::string &name, **Sound** *sound)
  存在するサウンドデータのポインタを登録する
- bool **CreateSound** (const std::string &name, const std::string &path)
  新しくサウンドを生成しデータに登録する
- **Sound** * **GetSoundData** (const std::string &name)
  登録されているサウンドデータを返す
- void **SetTextureData** (const std::string &name, **Texture** *tex)
  存在するテクスチャデータのポインタを登録する
- bool **CreateTexture** (const std::string &name, const std::string &path)
  新しくテクスチャを生成しデータを登録する
- **Texture** * **GetTextureData** (const std::string &name)
  登録されているテクスチャデータを返す
- bool **DeleteTexture** (const std::string &name)
  登録されているテクスチャデータを破棄,deleteする
- bool **DeleteSound** (const std::string &name)
  登録されているサウンドデータを破棄,deleteする
- virtual **~ResourceManager** ()
  destructor 全データを破棄,deleteする

詳解

リソースを生成、解放、管理を行うclass

既存GameEngineを使用している場合は自動で生成される

ここに登録しているものはEngine終了時自動でdeleteされるので自分でdeleteしないこと

newせず登録したものは自分でDeleteResource呼んで消しておくこと

構築子と解体子

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

destructor 全データを破棄,deleteする

関数詳解

bool ResourceManager::CreateSound (const std::string & *name*, const std::string & *path*)

新しくサウンドを生成しデータに登録する

引数:

in	*string*	name 登録名
in	*string*	path ファイルのパス

戻り値:

bool 生成に成功でtrueを返す

bool ResourceManager::CreateTexture (const std::string & *name*, const std::string & *path*)

新しくテクスチャを生成しデータを登録する

引数:

in	*string*	name 登録名
in	*string*	path ファイルのパス

戻り値:

bool 生成に成功でtrueを返す

bool ResourceManager::DeleteSound (const std::string & *name*)

登録されているサウンドデータを破棄,deleteする

引数:

in	*string*	name 登録名

戻り値:

bool 削除に成功でtrue

bool ResourceManager::DeleteTexture (const std::string & *name*)

登録されているテクスチャデータを破棄,deleteする

引数:

in	*string*	name 登録名

戻り値:

bool 削除に成功でtrue

Sound * ResourceManager::GetSoundData (const std::string & *name*)

登録されているサウンドデータを返す

引数:

in	*string*	name　登録名

戻り値:

Sound* 登録されているサウンドを返す

Texture * ResourceManager::GetTextureData (const std::string &　*name*)

登録されているテクスチャデータを返す

引数:

in	*string*	name 登録名

戻り値:

Sound* 登録されているテクスチャを返す

void ResourceManager::SetSoundData (const std::string &　*name*, Sound *　*sound*)

存在するサウンドデータのポインタを登録する

引数:

in	*string*	name 登録名
in	*Sound**	sound サウンドデータのポインタ

void ResourceManager::SetTextureData (const std::string &　*name*, Texture *　*tex*)

存在するテクスチャデータのポインタを登録する

引数:

in	*string*	登録名
in	*Texture**	tex テクスチャデータのポインタ

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/ResourceManager/**ResourceManager.h**
- src/lib/OGSystem/ResourceManager/**ResourceManager.cpp**

SceneManager クラス

sceneを管理するclass
`#include <OGsystem.h>`

公開メンバ関数

- **SceneManager** ()
  constructor
- virtual **~SceneManager** ()
  destructor
- void **SetNowTask** (**SceneTask** *task)
  タスクを登録する
- void **SetNextTask** (**SceneTask** *task)
  タスクを登録する
- void **SetOtherTask** (**SceneTask** *task)
  タスクを登録する
- void **SceneMigration** ()
  タスクを移行する
- void **OrtherSceneKillCheck** ()
  他タスクの削除管理
- **SceneTask** * **GetNowTask** () const
  現在タスクを取得
- **SceneTask** * **GetNextTask** () const
  次タスクを取得
- std::vector< **SceneTask** * > **GetOtherAllTask** () const
  他タスクを全取得

詳解

sceneを管理するclass

構築子と解体子

SceneManager::SceneManager () `[explicit]`

constructor

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

destructor

関数詳解

SceneTask * SceneManager::GetNextTask () const

次タスクを取得

戻り値:
SceneTask* 次のタスク

SceneTask * SceneManager::GetNowTask () const

現在タスクを取得

戻り値:
SceneTask* 現在のタスク

std::vector< SceneTask * > SceneManager::GetOtherAllTask () const

他タスクを全取得

戻り値:
vector<SceneTask*> 他タスク達

void SceneManager::OrtherSceneKillCheck ()

他タスクの削除管理

void SceneManager::SceneMigration ()

タスクを移行する

void SceneManager::SetNextTask (SceneTask * *task*)

タスクを登録する

引数:

in	*SceneTask**	次に登録したいタスク

void SceneManager::SetNowTask (SceneTask * *task*)

タスクを登録する

引数:

in	*SceneTask**	現在に登録したいタスク

void SceneManager::SetOtherTask (SceneTask * *task*)

タスクを登録する

引数:

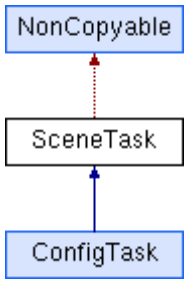
in	*SceneTask**	登録したいタスク

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/OG/**OGsystem.h**
- src/lib/OGSystem/OG/**OGsystem.cpp**

SceneTask クラス

Sceneを扱うclass
`#include <SceneTask.h>`
SceneTask の継承関係図



公開メンバ関数

- **SceneTask** ()
  constructor
- virtual **~SceneTask** ()
  destructor
- bool **Init** (const std::string &name)
  初期化処理
- void **UpdateManager** ()
  更新処理
- virtual void **Update** ()
  更新処理
- virtual void **Pause** ()
  ポーズ処理
- void **Kill** (const bool flag=true)
  削除命令
- bool **GetNextTask** () const
  次タスクの生成の許可を返す
- std::string **GetTaskName** () const
  登録されているタスク名を返す
- void **SetPause** (const bool flag=true)
  ポーズ設定を行う
- void **SetStop** (const bool flag=true)
  停止処理を行う
- **ST::Mode GetMode** () const
  現状の状態を返す
- bool **ModeCheck** (const **ST::Mode** &mode) const
  モード確認 [in] Mode mode 比較対象
- void **EnableGameObjectDestroy** (const bool flag)
  終了時にGameObjectを削除させる設定
- bool **GetAllObjectDestroy** () const
  了時にGameObjectを削除させる設定を取得

詳解

Sceneを扱うclass
更新処理のみをもっている
破棄時に全オブジェクトの削除権限をもっている

構築子と解体子

SceneTask::SceneTask () `[explicit]`

constructor

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

destructor

関数詳解

void SceneTask::EnableGameObjectDestroy (const bool *flag*)

終了時にGameObjectを削除させる設定

引数:

in	*bool*	flag trueなら削除させる

bool SceneTask::GetAllObjectDestroy () const

了時にGameObjectを削除させる設定を取得

戻り値:
bool trueなら削除させる

ST::Mode SceneTask::GetMode () const

現状の状態を返す

戻り値:
Mode 状態

bool SceneTask::GetNextTask () const

次タスクの生成の許可を返す

戻り値:

bool 生成許可

std::string SceneTask::GetTaskName () const

登録されているタスク名を返す

戻り値:

string タスク名

bool SceneTask::Init (const std::string & *name*)

初期化処理

引数:

in	*string*	name タスク名

戻り値:

bool 成功でtrue

void SceneTask::Kill (const bool *flag* = `true`)

削除命令

引数:

in	*bool*	flag 次タスクの生成を行うかの設定

bool SceneTask::ModeCheck (const ST::Mode & *mode*) const

モード確認 [in] Mode mode 比較対象

戻り値:

bool 比較対象と同じならtrue

void SceneTask::Pause () `[virtual]`

ポーズ処理

void SceneTask::SetPause (const bool *flag* = `true`)

ポーズ設定を行う

引数:

in	*bool*	flag ポーズ設定

void SceneTask::SetStop (const bool *flag* = `true`)

停止処理を行う

引数:

in	*bool*	flag 停止処理設定

void SceneTask::Update () `[virtual]`

更新処理

void SceneTask::UpdateManager ()

更新処理

このクラス詳解は次のファイルから抽出されました:

- src/lib/Object/**SceneTask.h**
- src/lib/Object/**SceneTask.cpp**

Easing::Sine クラス

```
#include <easing.hpp>
```

公開メンバ関数

- float **In** (float t, float b, float c, float d)
- float **Out** (float t, float b, float c, float d)
- float **InOut** (float t, float b, float c, float d)

関数詳解

float Easing::Sine::In (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Sine::InOut (float *t*, float *b*, float *c*, float *d*) `[inline]`

float Easing::Sine::Out (float *t*, float *b*, float *c*, float *d*) `[inline]`

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Easing/**easing.hpp**

Sound クラス

```
#include <Sound.h>
```

公開メンバ関数

- **Sound** ()
  constructor
- **Sound** (const std::string &path_, const bool loop=false)
  constructor
- virtual **~Sound** ()
- bool **Create** (const std::string &path_, const bool loop=false)
  サウンドの読み込み
- void **Play** () const
  サウンドの再生
- void **Stop** () const
  サウンドの停止
- void **Pause** () const
  サウンドの一時停止
- void **Volume** (const float value_) const
  サウンドの音量変更
- void **Pitch** (const float value_) const
  サウンドのピッチ変更
- void **Looping** (const bool loop_) const
  サウンドのループ再生
- bool **IsPlay** () const
  サウンドが再生中か調べる
- float **CurrentTime** () const
  現在の再生時間を返す
- float **Duration** () const
  サウンドの時間を返す
- void **SetVolume** (float maxVolume_)
  最大音量を指定する
- float **GetVolume** () const
  設定されている最大音量を返す

詳解

サウンドを読み込み、再生を行うclass

一括読み込みをしている

構築子と解体子

Sound::Sound () `[explicit]`

constructor

Sound::Sound (const std::string & *path_*, const bool *loop* = `false`)`[explicit]`

constructor

引数:

in	*string*	path_ ファイルのパス
in	*bool*	loop ループ再生

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

brief destructor

関数詳解

bool Sound::Create (const std::string & *path_*, const bool *loop* = `false`)

サウンドの読み込み

引数:

in	*string*	path_ ファイル名
in	*bool*	loop ループ再生

戻り値:

bool 読み込み成功true

float Sound::CurrentTime () const

現在の再生時間を返す

戻り値:

現在の再生時間

float Sound::Duration () const

サウンドの時間を返す

戻り値:

サウンドの時間

float Sound::GetVolume () const

設定されている最大音量を返す

戻り値:

float 最大音量

bool Sound::IsPlay () const

サウンドが再生中か調べる

戻り値:
再生中true

void Sound::Looping (const bool *loop_*) const

サウンドのループ再生

引数:

in	*bool*	loop_ ループの設定

void Sound::Pause () const

サウンドの一時停止

void Sound::Pitch (const float *value_*) const

サウンドのピッチ変更

引数:

in	*floar*	value_ ピッチ

void Sound::Play () const

サウンドの再生

void Sound::SetVolume (float *maxVolume_*)

最大音量を指定する

引数:

in	*float*	maxVolume_ 最大音量

void Sound::Stop () const

サウンドの停止

void Sound::Volume (const float *value_*) const

サウンドの音量変更

引数:

in	*float*	value_ 音量

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Audio/**Sound.h**
- src/lib/OGSystem/Audio/**Sound.cpp**

SoundManager クラス

Soundを管理するManager
`#include <SoundManager.h>`

公開メンバ関数

- **SoundManager** ()
- **~SoundManager** ()
- void **SetMaxVolume** (const float value_)
- float **GetMaxVolume** () const
- void **SetVolume** (const **Sound** *s, const float value_)
- void **SetSound** (**Sound** *s)
- bool **DeleteSound** (const **Sound** *s)
- void **AllDelete** ()
- void **Application** ()

詳解

Soundを管理するManager

構築子と解体子

SoundManager::SoundManager () `[explicit]`

SoundManager::~SoundManager ()

関数詳解

void SoundManager::AllDelete ()

void SoundManager::Application ()

bool SoundManager::DeleteSound (const Sound * *s*)

float SoundManager::GetMaxVolume () const

void SoundManager::SetMaxVolume (const float *value_*)

void SoundManager::SetSound (Sound * *s*)

void SoundManager::SetVolume (const Sound * *s*, const float *value_*)

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Audio/**SoundManager.h**
- src/lib/OGSystem/Audio/**SoundManager.cpp**

Source クラス

SoundDataのSourceDataを扱う
`#include <Audio.h>`

公開メンバ関数

- **Source** ()
- **~Source** ()
- void **BindBuffer** (const **Buffer** &buffer_)
- void **UnBindBuffer** () const
- ALuint **GetID** () const
- void **Play** () const
- void **Stop** () const
- void **Pause** () const
- void **Volume** (const float volume_) const
- void **Pitch** (const float value_) const
- void **Looping** (const bool loop_) const
- bool **isPlay** () const
- float **currenttime** () const
- void **queueBuffer** (const **Buffer** &buffer_) const
- ALuint **UnqueueBuffer** () const
- int **processed** () const

公開変数類

- ALuint **id_**

詳解

SoundDataのSourceDataを扱う

構築子と解体子

Source::Source ()

Source::~Source ()

関数詳解

void Source::BindBuffer (const Buffer &　*buffer_*)

float Source::currenttime () const

ALuint Source::GetID () const

bool Source::isPlay () const

void Source::Looping (const bool　*loop_*) const

void Source::Pause () const

void Source::Pitch (const float　*value_*) const

void Source::Play () const

int Source::processed () const

void Source::queueBuffer (const Buffer &　*buffer_*) const

void Source::Stop () const

void Source::UnBindBuffer () const

ALuint Source::UnqueueBuffer () const

void Source::Volume (const float　*volume_*) const

メンバ詳解

ALuint Source::id_

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Audio/**Audio.h**
- src/lib/OGSystem/Audio/**Audio.cpp**

StreamingSound クラス

`#include <StreamingSound.h>`

公開メンバ関数

- **StreamingSound** ()
- **StreamingSound** (const std::string &path, const bool loop=false)
- void **gain** (const float gain)
- void **pause** ()
- void **play** ()
- void **pitch** (const float value_) const
- void **stop** ()
- void **DeleteSound** ()
- bool **isPlaying** ()
- float **GetTime** () const
- void **debugUpdata** ()
- void **createSound** (const std::string &path, bool loop=false)

詳解

サウンドの読み込み、再生を行うclass 随時読み込み ※バグあり

構築子と解体子

StreamingSound::StreamingSound ()

StreamingSound::StreamingSound (const std::string & *path*, const bool *loop* = `false`)

関数詳解

void StreamingSound::createSound (const std::string & *path*, bool *loop* = `false`)

void StreamingSound::debugUpdata ()

void StreamingSound::DeleteSound ()

void StreamingSound::gain (const float *gain*)

float StreamingSound::GetTime () const

bool StreamingSound::isPlaying ()

void StreamingSound::pause ()

void StreamingSound::pitch (const float *value_*) const

void StreamingSound::play ()

void StreamingSound::stop ()

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Audio/**StreamingSound.h**
- src/lib/OGSystem/Audio/**StreamingSound.cpp**

StreamWav クラス

```
#include <Audio.h>
```

公開メンバ関数

- **StreamWav** (const std::string &file)
- bool **isStereo** () const
- **u_int sampleRate** () const
- void **loop** (const bool loop)
- void **toTop** ()
- bool **isEnd** () const
- size_t **GetlastSize** () const
- size_t **read** (std::vector< char > &buffer)

構築子と解体子

StreamWav::StreamWav (const std::string & *file*) `[explicit]`

関数詳解

size_t StreamWav::GetlastSize () const

bool StreamWav::isEnd () const

bool StreamWav::isStereo () const

void StreamWav::loop (const bool *loop*)

size_t StreamWav::read (std::vector< char > & *buffer*)

u_int StreamWav::sampleRate () const

void StreamWav::toTop ()

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Audio/**Audio.h**
- src/lib/OGSystem/Audio/**Audio.cpp**

Texture クラス

画像の読み込み、表示を行うclass
`#include <Texture.h>`

公開メンバ関数

- **Texture** ()
  constructor
- **Texture** (const std::string &path)
  constructor
- virtual **~Texture** ()
  destructor
- bool **Create** (const std::string &path)
  画像データの生成
- void **Draw** (const **Box2D** &draw, const **Box2D** &src, const **Color** &color={ 1.0f, 1.0f, 1.0f, 1.0f })
  描画処理
- void **Rotate** (const float angle)
  回転の適用
- **Vec2 GetTextureSize** () const
  読み込んだ画像のサイズを返す
- GLuint **GetID** () const
  登録されているIDを返す
- GLuint **CreateID** (const GLsizei &size)
  テクスチャIDを生成する
- void **DeleteID** (const GLsizei &size)
  テクスチャIDを削除する
- void **Bind** (const GLuint &id)
  テクスチャをバインドする

詳解

画像の読み込み、表示を行うclass

構築子と解体子

Texture::Texture () `[explicit]`

constructor

Texture::Texture (const std::string & *path*) `[explicit]`

constructor

引数:

in	*string*	path ファイルパス

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

destructor

関数詳解

void Texture::Bind (const GLuint & *id*)

テクスチャをバインドする

引数:

in	*GLuint*	id バインドするID 0指定で対象を無にする

bool Texture::Create (const std::string & *path*)

画像データの生成

引数:

in	*string*	path ファイルパス

GLuint Texture::CreateID (const GLsizei & *size*)

テクスチャIDを生成する

引数:

in	*GLsizei*	size 生成数

戻り値:

GLuint ID

void Texture::DeleteID (const GLsizei & *size*)

テクスチャIDを削除する

引数:

in	*GLsizei*	size 削除数

void Texture::Draw (const Box2D & *draw*, const Box2D & *src*, const Color & *color* = `{ 1.0f,1.0f,1.0f,1.0f }`)

描画処理

引数:

in	***Box2D***	draw 描画範囲
in	***Box2D***	src 画像範囲
in	***Color***	color 描画色

GLuint Texture::GetID () const

登録されているIDを返す

戻り値:

GLuint ID

Vec2 Texture::GetTextureSize () const

読み込んだ画像のサイズを返す

戻り値:

Vec2 画像の大きさ

void Texture::Rotate (const float *angle*)

回転の適用

引数:

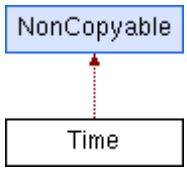
in	*float*	angle 回転値

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Texture/**Texture.h**
- src/lib/OGSystem/Texture/**Texture.cpp**

Time クラス

実時間を導くためのclass
`#include <glTimer.h>`
Time の継承関係図



公開メンバ関数

- **Time** ()
  constructor
- virtual **~Time** ()
  destructor
- void **Start** ()
  計測開始
- void **Stop** ()
  計測停止
- void **Pause** ()
  一時停止
- float **GetTime** ()
  現在タイムを返す
- void **InitTime** (const float time_)
  開始タイムの変更
- bool **IsPlay** () const
  計測判定

詳解

実時間を導くためのclass

構築子と解体子

Time::Time () `[explicit]`

constructor

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

destructor

関数詳解

float Time::GetTime ()

現在タイムを返す

戻り値:

float 現在のタイム

void Time::InitTime (const float *time_*)

開始タイムの変更

引数:

in	*float*	time_ 初期時のタイム

bool Time::IsPlay () const

計測判定

戻り値:

bool 計測を行っているか調べる

void Time::Pause ()

一時停止

void Time::Start ()

計測開始

void Time::Stop ()

計測停止

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Timer/**glTimer.h**
- src/lib/OGSystem/Timer/**glTimer.cpp**

UIObject クラス

UI表示のためのオブジェクト
`#include <UITask.h>`

公開メンバ関数

- **UIObject** ()
  constructor
- **UIObject** (const std::string &tag, const **Vec2** &pos, const **Vec2** &scale, const float angle=0.f)
  constructor
- virtual **~UIObject** ()
  [in] destructor
- void **Init** (const std::string &tag, const **Vec2** &pos, const **Vec2** &scale, const float angle=0.f)
  Initialize
- virtual void **Update** ()
  更新処理
- virtual void **Pause** ()
  停止処理
- virtual void **Render2D** ()
  描画処理
- void **UpdateManager** ()
  更新管理
- void **RenderManager** ()
  描画管理
- void **SetPosition** (const **Vec2** &pos)
  位置設定
- void **SetPosition** (const float x, const float y)
  位置設定
- **Vec2 GetPosition** ()
  位置情報取得
- void **SetScale** (const **Vec2** &scale)
  拡縮設定
- void **SetScale** (const float x, const float y)
  拡縮設定
- **Vec2 GetScale** () const
  拡縮情報取得
- void **SetRotate** (const float angle)
  回転設定
- float **GetRotate** () const
  回転情報取得
- void **SetTag** (const std::string &name)
  タグ設定
- std::string **GetTag** () const
  タグ情報取得
- void **Kill** ()
  オブジェクトを削除する
- void **CancelKill** ()

削除命令をキャンセルする

- void **SetPause** (const bool flag=true)
  Pause設定
- void **SetStop** (const bool flag=true)
  停止設定
- void **SetAllStop** (const bool flag=true)
  全機能停止
- **UO::Mode GetMode** () const
  現状の状態を返す
- bool **ModeCheck** (const **UO::Mode** &mode) const
  モード確認 [in] Mode mode 比較対象
- void **SetDrawOrder** (const unsigned int order)
  描画順指定 高い方が前に描画される
- unsigned int **GetDrawOrder** () const
  描画順を取得

詳解

UI表示のためのオブジェクト

更新と描画のみを行える

構築子と解体子

UIObject::UIObject () `[explicit]`

constructor

UIObject::UIObject (const std::string & *tag*, const Vec2 & *pos*, const Vec2 & *scale*, const float *angle* = `0.f`) `[explicit]`

constructor

引数:

in	*string&*	tag TaskName
in	*const*	**Vec2**& pos Position
in	*const*	**Vec2**& scale Scale
in	*float*	angle Angle

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

[in] destructor

関数詳解

void UIObject::CancelKill ()

削除命令をキャンセルする

unsigned int UIObject::GetDrawOrder () const

描画順を取得

UO::Mode UIObject::GetMode () const

現状の状態を返す

戻り値:
Mode 状態

Vec2 UIObject::GetPosition ()

位置情報取得

戻り値:
Vec2 Position

float UIObject::GetRotate () const

回転情報取得

戻り値:
float angle

Vec2 UIObject::GetScale () const

拡縮情報取得

戻り値:
Vec2 Scale

std::string UIObject::GetTag () const

タグ情報取得

戻り値:
string objecttag

void UIObject::Init (const std::string & *tag*, const Vec2 & *pos*, const Vec2 & *scale*, const float *angle* = `0.f`)

Initialize

引数:

in	*string&*	tag TaskName
in	*const*	**Vec2**& pos Position
in	*const*	**Vec2**& scale Scale
in	*float*	angle Angle

void UIObject::Kill ()

オブジェクトを削除する

bool UIObject::ModeCheck (const UO::Mode & *mode*) const

モード確認 [in] Mode mode 比較対象

戻り値:

bool 比較対象と同じならtrue

void UIObject::Pause () `[virtual]`

停止処理

void UIObject::Render2D () `[virtual]`

描画処理

void UIObject::RenderManager ()

描画管理

void UIObject::SetAllStop (const bool *flag* = `true`)

全機能停止

引数:

in	*bool*	flag trueで全停止

void UIObject::SetDrawOrder (const unsigned int *order*)

描画順指定 高い方が前に描画される

引数:

in	*unsigned*	int order 描画順値

void UIObject::SetPause (const bool *flag* = `true`)

Pause設定

引数:

in	*bool*	flag trueでPause化

void UIObject::SetPosition (const Vec2 & *pos*)

位置設定

引数:

in	***Vec2***	pos 位置

void UIObject::SetPosition (const float *x*, const float *y*)

位置設定

引数:

in	*float*	x 位置X座標
in	*float*	y 位置Y座標

void UIObject::SetRotate (const float *angle*)

回転設定

引数:

in	*float*	angle 回転角度

void UIObject::SetScale (const Vec2 & *scale*)

拡縮設定

引数:

in	***Vec2***	scale 拡縮

void UIObject::SetScale (const float *x*, const float *y*)

拡縮設定

引数:

in	*float*	x 拡縮X
in	*float*	y 拡縮Y

void UIObject::SetStop (const bool *flag* = `true`)

停止設定

引数:

in	*bool*	flag trueで停止

void UIObject::SetTag (const std::string & *name*)

タグ設定

引数:

in	*string*	name tagname

void UIObject::Update () `[virtual]`

更新処理

void UIObject::UpdateManager ()

更新管理

このクラス詳解は次のファイルから抽出されました:

- src/lib/Object/**UITask.h**
- src/lib/Object/**UITask.cpp**

Vec2 クラス

2次元Vector
`#include <OGlib.h>`

公開メンバ関数

- **Vec2** ()
- **Vec2** (const float, const float)
- **Vec2** (const int, const int)
- void **Normalize** ()
- float **GetLength** ()
- **Vec2 operator+** (const **Vec2** &)
- **Vec2 operator-** (const **Vec2** &)
- **Vec2 operator*** (const float)
- **Vec2 operator*** (const **Vec2** &)
- void **operator+=** (const **Vec2** &)
- void **operator*=** (const float)
- void **operator*=** (const **Vec2** &)
- void **operator-=** (const **Vec2** &)
- bool **operator==** (const **Vec2** &)

公開変数類

- float **x**
- float **y**

詳解

2次元Vector

構築子と解体子

Vec2::Vec2 ()

Vec2::Vec2 (const float *ex*, const float *ey*)

Vec2::Vec2 (const int *ex*, const int *ey*)

関数詳解

float Vec2::GetLength ()

void Vec2::Normalize ()

Vec2 Vec2::operator* (const float *n*)

Vec2 Vec2::operator* (const Vec2 & *v*)

void Vec2::operator*= (const float *n*)

void Vec2::operator*= (const Vec2 & *v*)

Vec2 Vec2::operator+ (const Vec2 & *v*)

void Vec2::operator+= (const Vec2 & *v*)

Vec2 Vec2::operator- (const Vec2 & *v*)

void Vec2::operator-= (const Vec2 & *v*)

bool Vec2::operator== (const Vec2 & *v*)

メンバ詳解

float Vec2::x

float Vec2::y

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/OG/**OGlib.h**
- src/lib/OGSystem/OG/**OGlib.cpp**

Vec3 クラス

3次元Vector
`#include <OGlib.h>`

公開メンバ関数

- **Vec3** ()
- **Vec3** (const float, const float, const float)
- **Vec3** (const int, const int, const int)

公開変数類

- float **x**
- float **y**
- float **z**

詳解

3次元Vector

構築子と解体子

Vec3::Vec3 ()

Vec3::Vec3 (const float *ex*, const float *ey*, const float *ez*)

Vec3::Vec3 (const int *ex*, const int *ey*, const int *ez*)

メンバ詳解

float Vec3::x

float Vec3::y

float Vec3::z

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/OG/**OGlib.h**
- src/lib/OGSystem/OG/**OGlib.cpp**

Wav クラス

Wavファイルのデータを扱う
`#include <Audio.h>`

クラス

- struct **Info**

公開メンバ関数

- **Wav** (const std::string &file)
- **u_int channel** () const
- bool **isStereo** () const
- **u_int sampleRate** () const
- **u_int size** () const
- float **time** () const
- const char * **data** () const

静的公開メンバ関数

- static bool **analyzeWavFile** (**Info** &info, std::ifstream &fstr)

詳解

Wavファイルのデータを扱う

構築子と解体子

Wav::Wav (const std::string & *file*) `[explicit]`

関数詳解

bool Wav::analyzeWavFile (Info & *info*, std::ifstream & *fstr*) `[static]`

u_int Wav::channel () const

const char * Wav::data () const

bool Wav::isStereo () const

u_int Wav::sampleRate () const

u_int Wav::size () const

float Wav::time () const

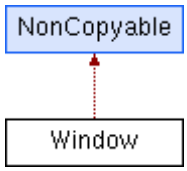
このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Audio/**Audio.h**

- src/lib/OGSystem/Audio/**Audio.cpp**

Window クラス

Windowの生成、管理を行うclass
`#include <Window.h>`
Window の継承関係図



公開メンバ関数

- **Window** ()
  constructor
- **Window** (const int x, const int y, const char *name, const bool flag, const **Vec2** &pos)
  constructor
- virtual **~Window** ()
  destructor
- bool **createWindow** (const int x, const int y, char *name, const bool flag, const **Vec2** &pos)
  constructor
- void **SetIcon** (const std::string &path)
  アイコンを設定する
- void **LimitsWindow** ()
  Windowのサイズとアスペクト比を固定する
- void **LimitsWindow** (const int w, const int h)
  Windowのサイズとアスペクト比を固定する
- void **WindowIcon** ()
  Windowをアイコン化、アイコン化から解除する
- void **Visualization** ()
  Windowを可視化、可視化から解除する
- void **InMouseMode** (const bool flag)
  マウスを表示させるかを決める
- **Vec2 GetSize** () const
  Windowのサイズを返す
- **Vec2 GetPos** () const
  Windowの位置を返す
- void **SetWindowPos** (const **Vec2** &pos)
  生成するWindowの位置を設定する
- GLFWwindow * **GetWindow** () const
  Window情報を返す
- void **SetPos** (const **Vec2** &pos)
  Windowの位置を設定する
- void **SetPos** (const int x, const int y)
  Windowの位置を設定する
- void **SetSize** (const **Vec2** &size)
  Windowのサイズを設定する
- void **SetSize** (const int w, const int h)
  Windowのサイズを設定する

- void **ChengeTitle** (const char *name)
  ChengeWindowTitle
- void **ChengeWindow** (const int x, const int y, const int w, const int h, const bool flag)
  Windowモードを変更
- void **ChengeWindow** (const **Vec2** &pos, const **Vec2** &size, const bool flag)
  Windowモードを変更

詳解

Windowの生成、管理を行うclass

構築子と解体子

Window::Window () `[explicit]`

constructor

Window::Window (const int *x*, const int *y*, const char * *name*, const bool *flag*, const Vec2 & *pos*) `[explicit]`

constructor

引数:

in	*int*	x windowXサイズ
in	*int*	y windowYサイズ
in	*char**	name window名
in	*bool*	flag フルスク設定
in	***Vec2***	pos window位置

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

destructor

関数詳解

void Window::ChengeTitle (const char * *name*)

ChengeWindowTitle

引数:

in	*const*	char* name TitleName

void Window::ChengeWindow (const int *x*, const int *y*, const int *w*, const int *h*, const bool *flag*)

Windowモードを変更

引数:

in	*int*	x WindowPosition_x
in	*int*	y WindowPosition_y
in	*int*	w WindowSize_x
in	*int*	h WindowSize_h
in	*bool*	flag true hullScreen

void Window::ChengeWindow (const Vec2 & *pos*, const Vec2 & *size*, const bool *flag*)

Windowモードを変更

引数:

in	*const*	**Vec2**& pos WindowPosition
in	*const*	**Vec2**& size WindowSize
in	*bool*	flag true hullScreen

bool Window::createWindow (const int *x*, const int *y*, char * *name*, const bool *flag*, const Vec2 & *pos*)

constructor

引数:

in	*int*	x windowXサイズ
in	*int*	y windowYサイズ
in	*char**	name window名
in	*bool*	flag フルスク設定
in	***Vec2***	pos window位置

戻り値:

bool 生成に成功でtrue

Vec2 Window::GetPos () const

Windowの位置を返す

戻り値:

Vec2 Windowの位置

Vec2 Window::GetSize () const

Windowのサイズを返す

戻り値:

Vec2 Windowのサイズ

GLFWwindow * Window::GetWindow () const

Window情報を返す

戻り値:
GLFWWinfow* Window情報

void Window::InMouseMode (const bool *flag*)

マウスを表示させるかを決める

引数:

in	*bool*	flag 表示設定

void Window::LimitsWindow ()

Windowのサイズとアスペクト比を固定する

void Window::LimitsWindow (const int *w*, const int *h*)

Windowのサイズとアスペクト比を固定する

引数:

in	*int*	w WindowSize_w
in	*int*	h WindowSize_h

void Window::SetIcon (const std::string & *path*)

アイコンを設定する

引数:

in	*string*	path ファイルパス

void Window::SetPos (const Vec2 & *pos*)

Windowの位置を設定する

引数:

in	*const*	**Vec2**& pos 位置

void Window::SetPos (const int *x*, const int *y*)

Windowの位置を設定する

引数:

in	*int*	x WindowPosition_x
in	*int*	y WindowPosition_y

void Window::SetSize (const Vec2 & *size*)

Windowのサイズを設定する

引数:

in	*const*	**Vec2**& size サイズ

void Window::SetSize (const int *w*, const int *h*)

Windowのサイズを設定する

引数:

in	*int*	w WindowSize_w
in	*int*	h WindowSize_h

void Window::SetWindowPos (const Vec2 & *pos*)

生成するWindowの位置を設定する

引数:

in	***Vec2***	pos Windowの位置

void Window::Visualization ()

Windowを可視化、可視化から解除する

void Window::WindowIcon ()

Windowをアイコン化、アイコン化から解除する

このクラス詳解は次のファイルから抽出されました:

- src/lib/OGSystem/Window/**Window.h**
- src/lib/OGSystem/Window/**Window.cpp**

ファイル詳解

src/lib/doxygen.c ファイル

src/lib/Event/Event.cpp ファイル

```
#include "Event.h"
#include "ResourceLoad¥ResourceLoad.h"
```

src/lib/Event/Event.h ファイル

```
#include "OG¥_OGsystem.h"
```

クラス

- class **EventTask**
  イベントタスク

列挙型

- enum **Event** { **RESOURCE_LOAD** }

読み込むイベントの種類

列挙型詳解

enum Event

読み込むイベントの種類

enum Event

列挙値:

RESOURCE_LOAD	リソース読み込みイベント

src/lib/Event/ResourceLoad/ResourceLoad.cpp ファイル

```
#include "ResourceLoad.h"
```

src/lib/Event/ResourceLoad/ResourceLoad.h ファイル

```
#include "OG¥OGsystem.h"
```

クラス

- class **ResourceLoad**
  リソースを読み込むイベントclass

src/lib/Object/GameObject.cpp ファイル

```
#include "GameObject.h"
```

src/lib/Object/GameObject.h ファイル

```
#include "OGSystem¥Collision¥Collision.h"
```

クラス

- class **GameObject**
  GameObject

名前空間

- **GO**

GameObjectに関するデータ 列挙型

- enum **Objform** { **Non**, **Circle**, **Box**, **Pointer**, **Line** }
- *オブジェクトの形の設定* enum **GO::Mode** { **GO::NORMAL**, **GO::PAUSE**, **GO::STOP**, **GO::ALLSTOP**, **GO::KILL** }

状態の設定

列挙型詳解

enum Objform

オブジェクトの形の設定

enum Objform

列挙値:

Non	無
Circle	円
Box	矩形
Pointer	点
Line	線

src/lib/Object/SceneTask.cpp ファイル

```
#include "SceneTask.h"
```

src/lib/Object/SceneTask.h ファイル

```
#include "OG¥_OGsystem.h"
```

クラス

- class **SceneTask**
  Sceneを扱うclass

名前空間

- **ST**

SceneTaskに関するデータ 列挙型

- enum **ST::Mode** { **ST::NORMAL**, **ST::PAUSE**, **ST::STOP**, **ST::KILL** }

src/lib/Object/UITask.cpp ファイル

```
#include "UITask.h"
#include "OG¥OGsystem.h"
```

src/lib/Object/UITask.h ファイル

```
#include "OG¥_OGsystem.h"
```

クラス

- class **UIObject**
  UI表示のためのオブジェクト

名前空間

- **UO**

UIObjectに関するデータ 列挙型

- enum **UO::Mode** { **UO::NORMAL**, **UO::PAUSE**, **UO::STOP**, **UO::ALLSTOP**, **UO::KILL** }

状態の設定

src/lib/OGSystem/Audio/Audio.cpp ファイル

```
#include "Audio.h"
```

src/lib/OGSystem/Audio/Audio.h ファイル

```
#include "OG¥_OGsystem.h"
```

クラス

- class **Audio**
  デバイスを設定するためのclass
- class **Buffer**
  SoundDataのBufferDataを扱う
- class **Source**
  SoundDataのSourceDataを扱う
- class **Wav**
  Wavファイルのデータを扱う
- struct **Wav::Info**
- class **StreamWav**

src/lib/OGSystem/Audio/Sound.cpp ファイル

```
#include "Sound.h"
```

src/lib/OGSystem/Audio/Sound.h ファイル

```
#include "Audio.h"
```

クラス

- class **Sound**

src/lib/OGSystem/Audio/SoundManager.cpp ファイル

```
#include "SoundManager.h"
```

src/lib/OGSystem/Audio/SoundManager.h ファイル

```
#include "Sound.h"
```

クラス

- class **SoundManager**
  Soundを管理するManager

src/lib/OGSystem/Audio/StreamingSound.cpp ファイル

```
#include "StreamingSound.h"
```

src/lib/OGSystem/Audio/StreamingSound.h ファイル

```
#include "Audio.h"
#include <mutex>
```

クラス

- class **StreamingSound**

src/lib/OGSystem/Button/Button.cpp ファイル

```
#include "Button.h"
```

src/lib/OGSystem/Button/Button.h ファイル

```
#include "OG¥OGsystem.h"
#include "Object¥GameObject.h"
```

クラス

- class **Button**
  マウスに反応するButtonclass

src/lib/OGSystem/Camera/Camera.cpp ファイル

```
#include "Camera.h"
```

src/lib/OGSystem/Camera/Camera.h ファイル

```
#include "OG¥_OGsystem.h"
```

クラス

- class **Camera2D**
  2DCamera

src/lib/OGSystem/Collision/Collision.cpp ファイル

```
#include "Collision.h"
```

src/lib/OGSystem/Collision/Collision.h ファイル

```
#include "OG¥_OGsystem.h"
```

クラス

- class **CollisionBase**
  判定の元
- class **CollisionBox**
  矩形判定
- class **CollisionCircle**
  円判定
- class **CollisionPointer**
  点判定
- class **CollisionLine**
  線判定
- class **CollisionCapsule**
  カプセル判定！ 未完成

名前空間

- **CT**

生成するときに使用する型 列挙型

- enum **CT::CollisionType** { **CT::BOX**, **CT::CIRCLE**, **CT::POINTER**, **CT::CAPSULE**, **CT::LINE**, **CT::NON** }

src/lib/OGSystem/Easing/easing.hpp ファイル

```
#include <math.h>
```

クラス

- class **Easing**
  Easingを扱うclass
- class **Easing::Linear**
- class **Easing::Back**
- class **Easing::Bounce**
- class **Easing::Circ**
- class **Easing::Cubic**
- class **Easing::Elastic**
- class **Easing::Expo**
- class **Easing::Quad**
- class **Easing::Quart**
- class **Easing::Quint**
- class **Easing::Sine**

マクロ定義

- #define **_USE_MATH_DEFINES**

マクロ定義詳解

#define _USE_MATH_DEFINES

src/lib/OGSystem/Font/Font.cpp ファイル

src/lib/OGSystem/Font/Font.h ファイル

src/lib/OGSystem/Font/TextureFont.cpp ファイル

```
#include "TextureFont.h"
#include "OG¥OGsystem.h"
```

src/lib/OGSystem/Font/TextureFont.h ファイル

```
#include "OGSystem¥Texture¥Texture.h"
```

クラス

- class **Font**
  フォントの描画class

src/lib/OGSystem/FPS/FPS.cpp ファイル

```
#include "FPS.h"
```

src/lib/OGSystem/FPS/FPS.h ファイル

```
#include "OG¥_OGsystem.h"
```

クラス

- class **FPS**
  フレームレートを計算,制限するclass

src/lib/OGSystem/Input/Input.cpp ファイル

```
#include "Input.h"
```

関数

- void **ResetGamePad** (std::vector< **Input::GamePad** *> &gamepad_)
- void **ResetKeyBoard** (**Input::KeyBoard** &keyboard)

関数詳解

void ResetGamePad (std::vector< Input::GamePad *> & *gamepad_*)

void ResetKeyBoard (Input::KeyBoard & *keyboard*)

src/lib/OGSystem/Input/Input.h ファイル

```
#include "OG¥_OGsystem.h"
#include "OGSystem¥Collision¥Collision.h"
```

クラス

- class **Input**
  ゲームパッド、キーボード、マウスの入力を扱うclass
- class **Input::GamePad**
  ゲームパッド入力
- class **Input::KeyBoard**
  キーボード入力
- class **Input::Mouse**
  マウス入力
- struct **Input::InputData**
  ゲームパッドとキーボードを区別する

名前空間

- **In**
- *簡易引数用* **Mouse**

マウス用簡易引数 列挙型

- enum **In::Button** { **In::BUTTON_A**, **In::BUTTON_B**, **In::BUTTON_X**, **In::BUTTON_Y**, **In::BUTTON_L1**, **In::BUTTON_R1**, **In::BUTTON_BACK**, **In::BUTTON_START**, **In::BUTTON_L3**, **In::BUTTON_R3**, **In::BUTTON_U**, **In::BUTTON_R**, **In::BUTTON_D**, **In::BUTTON_L** }
- *仮装コントローラの入力設定* enum **In::AXIS** { **In::AXIS_LEFT_X**, **In::AXIS_LEFT_Y**, **In::AXIS_RIGHT_X**, **In::AXIS_RIGHT_Y**, **In::AXIS_R2**, **In::AXIS_L2**, **In::AXIS_BUTTON_NUM** }
- *コントローラのスティックと押し込み* enum **In::StickButton** { **In::LSTICK_LEFT**, **In::LSTICK_RIGHT**, **In::LSTICK_UP**, **In::LSTICK_DOWN**, **In::RSTICK_LEFT**, **In::RSTICK_RIGHT**, **In::RSTICK_UP**, **In::RSTICK_DOWN**, **In::BUTTON_R2**, **In::BUTTON_L2**, **In::STICK_NUM** }
- *スティックをボタンButton入力にも対応* enum **In::IN** { **In::B1**, **In::B2**, **In::B3**, **In::B4**, **In::CD**, **In::CU**, **In::CR**, **In::CL**, **In::L1**, **In::R1**, **In::D1**, **In::D2**, **In::SR**, **In::SL**, **In::LD**, **In::LU**, **In::LR**, **In::LL**, **In::RD**, **In::RU**, **In::RR**, **In::RL**, **In::L2**, **In::R2** }
- *仮想入力* enum **In::Key** { **In::A**, **In::S**, **In::D**, **In::W**, **In::Q**, **In::E**, **In::Z**, **In::X**, **In::C**, **In::R**, **In::F**, **In::V**, **In::T**, **In::G**, **In::B**, **In::Y**, **In::H**, **In::N**, **In::U**, **In::J**, **In::M**, **In::I**, **In::K**, **In::O**, **In::L**, **In::P**, **In::SPACE**, **In::ENTER**, **In::ESCAPE**, **In::UP**, **In::DOWN**, **In::LEFT**, **In::RIGHT** }
- *キーボード入力* enum **Mouse::Button** { **Mouse::LEFT**, **Mouse::RIGTH**, **Mouse::CENTER**, **Mouse::BUTTON_4**, **Mouse::BUTTON_5**, **Mouse::BUTTON_6**, **Mouse::BUTTON_7**, **Mouse::BUTTON_8** }

Mouseのボタン

src/lib/OGSystem/OG/_OGsystem.cpp ファイル

```
#include "_OGsystem.h"
```

名前空間

- **OG**

便利機能関数や数学計算 関数

- void **OG::MulitMatrixf** (GLfloat *src1, GLfloat *src2, GLfloat *dst)
- void **OG::Normalize** (GLfloat *v)
- void **OG::LineBoxDraw** (const **Box2D** *_b, const **Color** &color, const float linewidth)
- void **OG::LineBoxDraw** (const **Box2D** *_b, const float linewidth, const **Color** &color)
- void **OG::LineBoxDraw** (const **Vec2** *_b, const **Color** &color, const float linewidth)
- void **OG::LineBoxDraw** (const **Vec2** *_b, const float linewidth, const **Color** &color)
- void **OG::LineDraw** (const **Vec2** *_b, const **Color** &color, const float linewidth)
- void **OG::LineDraw** (const **Vec2** *_b, const float linewidth, const **Color** &color)
- void **OG::LineOvalDraw** (const **Vec2** *pos, const **Vec2** *scale, const float linewidth, const **Color** &color)
- void **OG::LineOvalDraw** (const int x, const int y, const float ovalx, const float ovaly, const float linewidth, const **Color** &color)
- void **OG::PointDraw** (const **Vec2** *pos, const **Color** &color, const float linewidth)
- void **OG::PointDraw** (const **Vec2** *pos, const float linewidth, const **Color** &color)
- void **OG::_Rotate** (const float _ang, **Vec2** *_b)

src/lib/OGSystem/OG/_OGsystem.h ファイル

```
#include <GLFW/glfw3.h>
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include "al¥al.h"
#include "al¥alc.h"
#include <memory>
#include <cmath>
#include <fstream>
#include <sstream>
#include <cassert>
#include <string>
#include <iostream>
#include <vector>
#include <map>
#include <unordered_map>
#include <utility>
#include "OGlib.h"
#include "randmais.h"
```

名前空間

- **OG**

便利機能関数や数学計算 マクロ定義

- #define **GLEW_STATIC**
- #define **GLFW_INCLUDE_GLEXT**
- #define **GLFW_INCLUDE_GLU**
- #define **GLM_FORCE_SWIZZLE**
- #define **PI** 3.1415926f

型定義

- typedef unsigned char **u_char**
- typedef unsigned int **u_int**
- typedef unsigned long **u_long**

関数

- void **OG::MulitMatrixf** (GLfloat *src1, GLfloat *src2, GLfloat *dst)
- void **OG::Normalize** (GLfloat *v)
- void **OG::_Rotate** (const float _ang, **Vec2** *_b)
- void **OG::LineBoxDraw** (const **Box2D** *_b, const **Color** &color, const float linewidth)
- void **OG::LineBoxDraw** (const **Box2D** *_b, const float linewidth, const **Color** &color)
- void **OG::LineBoxDraw** (const **Vec2** *_b, const **Color** &color, const float linewidth)
- void **OG::LineBoxDraw** (const **Vec2** *_b, const float linewidth, const **Color** &color)
- void **OG::LineDraw** (const **Vec2** *_b, const **Color** &color, const float linewidth)
- void **OG::LineDraw** (const **Vec2** *_b, const float linewidth, const **Color** &color)
- void **OG::LineOvalDraw** (const int x, const int y, const float ovalx, const float ovaly, const float linewidth, const **Color** &color)
- void **OG::LineOvalDraw** (const **Vec2** *pos, const **Vec2** *scale, const float linewidth, const **Color** &color)
- void **OG::PointDraw** (const **Vec2** *pos, const float linewidth, const **Color** &color)
- void **OG::PointDraw** (const **Vec2** *pos, const **Color** &color, const float linewidth)
- void **OG::BackColor** (const **Color** &color)
- void **OG::BackColor** (const float &red, const float &green, const float &blue, const float &alpha)
- int **OG::mbclen** (const char *c)

- void **OG::cout** (const **Box2D** &)
- void **OG::cout** (const **Vec2** &)
- void **OG::cout** (const **Color** &)
- bool **OG::Data_Cipher** (const std::string &in_path, const std::string &out_path)
- std::string **OG::Data_Composite** (std::ifstream &ifs)
- void **OG::OutDebugData** (const std::string &out_path, const std::string &text)
- void **OG::DataClear** (const std::string &path)
- template<class T > bool **OG::Destroy** (T *t)
- template<class T > bool **OG::Destroy** (const T *t)

マクロ定義詳解

#define GLEW_STATIC

#define GLFW_INCLUDE_GLEXT

#define GLFW_INCLUDE_GLU

#define GLM_FORCE_SWIZZLE

#define PI 3.1415926f

型定義詳解

typedef unsigned char u_char

typedef unsigned int u_int

typedef unsigned long u_long

src/lib/OGSystem/OG/OGlib.cpp ファイル

```
#include "OGlib.h"
```

src/lib/OGSystem/OG/OGlib.h ファイル

```
#include <cmath>
```

クラス

- class **Circle**
  円データ型
- class **Vec3**
  3次元Vector
- class **Vec2**
  2次元Vector
- class **Mat4**
  *2*2行列*
- class **Mat4x4**
  *4*4行列*
- class **Box3D**
  3DBoxデータ型
- class **Box2D**
  2Dデータ型
- class **Color**
  色データ
- class **NonCopyable**
  コピーを禁止するclass

名前空間

- **OG**

便利機能関数や数学計算 マクロ定義

- #define **PI** 3.1415926f

関数

- float **OG::ToRadian** (const float degree_)
- void **OG::Cross** (float *, float *, float *)
- float **OG::inner** (const **Vec2** &, const **Vec2** &)
- float **OG::inner** (const float, const float, const float, const float)
- float **OG::inner** (const int, const int, const int, const int)
- float **OG::cross** (const **Vec2** &, const **Vec2** &)
- float **OG::cross** (const float, const float, const float, const float)
- float **OG::cross** (const int, const int, const int, const int)
- float **OG::doubleinner** (const **Vec2** &)
- float **OG::doubleinner** (const float, const float)
- float **OG::doubleinner** (const int, const int)
- float **OG::get_distance** (const float, const float, const float, const float, const float, const float)
- bool **OG::innerJudge** (const **Vec2** *line, const **Vec2** *point)

マクロ定義詳解

#define PI 3.1415926f

src/lib/OGSystem/OG/OGsystem.cpp ファイル

```
#include "OGsystem.h"
```

変数

- **EngineSystem * ge**
  内部システムエンジン
- **ResourceManager * rm**
  リソース管理システム

変数詳解

EngineSystem* ge

内部システムエンジン

ResourceManager* rm

リソース管理システム

src/lib/OGSystem/OG/OGsystem.h ファイル

```
#include "Camera¥Camera.h"
#include "Window¥Window.h"
#include "FPS¥FPS.h"
#include "Input¥Input.h"
#include "Texture¥Texture.h"
#include "Collision¥Collision.h"
#include "Font¥Font.h"
#include "Audio¥SoundManager.h"
#include "Timer¥glTimer.h"
#include "Easing¥easing.hpp"
#include "Audio¥StreamingSound.h"
#include "Random¥Random.h"
#include "ResourceManager¥ResourceManager.h"
#include "Font¥TextureFont.h"
#include "Object¥GameObject.h"
#include "Object¥SceneTask.h"
#include "Object¥UITask.h"
```

クラス

- class **OrderCheck**
  描画順を管理するclass
- class **SceneManager**
  sceneを管理するclass
- class **EngineSystem**
  ゲームエンジン

変数

- **EngineSystem * ge**
  内部システムエンジン
- **ResourceManager * rm**
  リソース管理システム

変数詳解

EngineSystem* ge

内部システムエンジン

ResourceManager* rm

リソース管理システム

src/lib/OGSystem/OG/OGTask.cpp ファイル

```
#include "OGTask.h"
#include "Task¥WinConfig.h"
#include "Event¥Event.h"
```

src/lib/OGSystem/OG/OGTask.h ファイル

```
#include "OG¥OGsystem.h"
```

クラス

- class **OGTK**
  Engine生成に関する処理を行うclass

src/lib/OGSystem/OG/System.cpp ファイル

```
#include "System.h"
```

src/lib/OGSystem/OG/System.h ファイル

```
#include "OGsystem.h"
```

クラス

- class **OGSystem**
  System

src/lib/OGSystem/randmais.h ファイル

変数

- const int **randomCipher** [40]

変数詳解

const int randomCipher[40]

src/lib/OGSystem/Random/Random.cpp ファイル

```
#include "Random.h"
```

src/lib/OGSystem/Random/Random.h ファイル

```
#include <random>
#include <iostream>
```

名前空間

- **random**

ランダム生成名前空間 関数

- void **random::Init** ()
  初期化
- int **random::GetRand** (const int min_, const int max_)
  整数のランダム
- float **random::GetRand** (const float min_, const float max_)
  浮動小数点のランダム
- std::string **random::GetRand** (const std::string &text, const std::size_t size)
  文字列のランダム

src/lib/OGSystem/ResourceManager/ResourceManager.cpp ファイル

```
#include "ResourceManager.h"
```

src/lib/OGSystem/ResourceManager/ResourceManager.h ファイル

```
#include "OGSystem¥Audio¥Sound.h"
#include "OGSystem¥Texture¥Texture.h"
```

クラス

- class **ResourceManager**
  リソースを生成、解放、管理を行うclass

src/lib/OGSystem/Shader/shader.cpp ファイル

```
#include "shader.h"
```

名前空間

- **Shader**

未実装 関数

- GLuint **Shader::compile** (GLuint type, const std::string &text)
- void **Shader::setup** (const GLuint program, const std::string &v_source, const std::string &f_source)
- GLuint **Shader::read** (const std::string &file)
- GLint **Shader::attrib** (const GLint program, const std::string &name)
- GLint **Shader::uniform** (const GLuint program, const std::string &name)
- void **Shader::use** (const GLuint program)
- void **Shader::SetViewPort** (float cl, float cr, float cb, float ct, float cn, float cf)

src/lib/OGSystem/Shader/shader.h ファイル

```
#include "OGSystem¥_OGsystem.h"
```

名前空間

- **Shader**

未実装 関数

- GLuint **Shader::compile** (GLuint type, const std::string &text)
- void **Shader::setup** (const GLuint program, const std::string &v_source, const std::string &f_source)
- GLuint **Shader::read** (const std::string &file)
- GLint **Shader::attrib** (const GLint program, const std::string &name)
- GLint **Shader::uniform** (const GLuint program, const std::string &name)
- void **Shader::use** (const GLuint program)
- void **Shader::SetViewPort** (float cl, float cr, float cb, float ct, float cn, float cf)

src/lib/OGSystem/Texture/Texture.cpp ファイル

```
#include "Texture.h"
#include "stb_image.h"
```

マクロ定義

- #define **STB_IMAGE_IMPLEMENTATION**

マクロ定義詳解

#define STB_IMAGE_IMPLEMENTATION

src/lib/OGSystem/Texture/Texture.h ファイル

```
#include "OG¥_OGsystem.h"
```

クラス

- class **Texture**
  画像の読み込み、表示を行うclass

src/lib/OGSystem/Thread/DataThread.cpp ファイル

src/lib/OGSystem/Thread/DataThread.h ファイル

src/lib/OGSystem/Timer/glTimer.cpp ファイル

```
#include "glTimer.h"
```

src/lib/OGSystem/Timer/glTimer.h ファイル

```
#include "OG¥_OGsystem.h"
```

クラス

- class **Time**
  実時間を導くためのclass

src/lib/OGSystem/Win/WinMain.cpp ファイル

```
#include "OG¥System.h"
#include "OG¥OGTask.h"
```

マクロ定義

- #define **_USE_MATH_DEFINES**
- #define **_OX_EPSILON_** 0.0000001f

関数

- int **main** ()
  エントリーポイント

マクロ定義詳解

#define _OX_EPSILON_ 0.0000001f

#define _USE_MATH_DEFINES

関数詳解

int main ()

エントリーポイント

src/lib/OGSystem/Window/Window.cpp ファイル

```
#include "Window.h"
#include "stb_image.h"
```

src/lib/OGSystem/Window/Window.h ファイル

```
#include "OG¥_OGsystem.h"
```

クラス

- class **Window**
  Windowの生成、管理を行うclass

src/lib/Task/WinConfig.cpp ファイル

```
#include "WinConfig.h"
#include "OGSystem¥Button¥Button.h"
#include "Task¥Task_Sample.h"
```

src/lib/Task/WinConfig.h ファイル

```
#include "OG¥OGsystem.h"
```

クラス

- class **ConfigTask**
  Windowをフルスクリーンか Windowで開くか設定を行うScene

索引
INDEX