OG2D ライブラリリファレンス

OpenGL2DGameLibrary リファレンス

目次

・Texture 実装概要
　　⇒class Texture　　//p31
・glTimer 実装概要
　　⇒class Time　　//p33
・Window 実装概要
　　⇒class Window　　//p35
・OGTask 実装概要
　　⇒class OGTK　　//p37
・TaskObject 実装概要
　　⇒class TaskObject

ページ未記入のページは使用することがほとんどないものになっています。
ソースコード上にコメント文もありますので使用したいものがある場合そちらをご確認ください。
時間が空いてるときに追記していきます。

OGlib/Circle

説明
円の図形を扱うもの。

フィールド

| | |
|---|---|
| float | center_x |
| float | center_y |
| float | r |

メソッド
Circle()
Circle(float x,float y,float r)
Circle(int x,int y,int r)
Circle(Circle& c)

サンプルコード

```
Circle test(5,5,10);

std::cout << test.center_x << std::endl;
std::cout << test.center_y << std::endl;
std::cout << test.r << std::endl;
```

---

```
5
5
10
```

OGlib/Vec2

説明
2 つの float 変数を扱うもの。

フィールド

| | |
|---|---|
| float | x |
| float | y |

メソッド
Vec2()
Vec2(float x,float y)
Vec2(int x,int y)
void Normalize()
float GetLength()

サンプルコード

```
Vec2 test;
test = {650,900}; //○
//test = {0.f,1} //error

std::cout << test.x << std::endl;
std::cout << test.y << std::endl;
```

---

650
900

OGlib/Box2D

説明
矩形の図形を扱うもの。

フィールド

| | |
|---|---|
| float | x |
| float | y |
| float | w |
| float | h |

メソッド
Box2D()
Box2D(float x,float y,float w,float h)
Box2D(int x,int y,int w,int h)
Box2D(Box2D& b)
Box2D(Vec2& pos,Vec2& size)

void Offset(float x,float y)
void Offset(int x,int y)
引数　float 型変数二つ
返り値　なし
引数の値分 x と y の値をプラスします。
void OffsetSize()
引数　なし
返り値なし
w の値に x を足したものを w にいれる。
h の値に y を足したものを h にいれる。

サンプルコード

```
Box2D draw(5,5,10,10);
std::cout << draw.x << draw.y << draw.w << draw.h << std::endl;
draw.OffsetSize();
std::cout << draw.x << draw.y << draw.w << draw.h << std::endl;

Vec2 position = {10,50};
Vec2 Scale = {64,64};
Box2D test(position,Scale);
test.OffsetSize();
```

---

```
551010
551515
```

OGlib/Color

説明
色を保持、OpenGL へ送るもの。

フィールド

| | |
|---|---|
| float | red |
| float | green |
| float | blue |
| float | alpha |

メソッド
Color()
Color(float,float,float,float)
Color(int,int,int,int)

unsigned int Getcolor();
引数　なし
戻り値　Color 情報
システム内部でパソコン側に送るときに使用するデータ

サンプルコード

```
Color color = {1,1,1,1};

glColor4f(color.red,color.green,color.blue,color.alpha);
```

OGlib/OG

メソッド

float ToRadian(float)
引数　float 角度
戻り値　radian 値
角度の radian を求めます。

_OGSystem/OG

メソッド

void _Rotate(float,Vec2*);
引数　float 角度 Vec2* 頂点情報
戻り値　なし
頂点情報を角度に合わせて変更する

void LineHitDraw(Vec2*)
引数　Vec* 頂点情報
戻り値　なし
頂点４点を白い線で引きます。

void LineHitDraw(Vec2*,Color&)
引数　Vec* 頂点情報 Color& 線の色
戻り値　なし
頂点４点を引数の色で引きます。

OGSystem/EngineSystem

説明
ゲームにて使用する各種機能を保持管理しているもの。

宣言
EngineSystem* OGge
グローバルポインタ
Winmain.cpp/int main で new をしている。
主にこれにアクセスして EngineSystem の機能を使用する

フィールド

| | |
|---|---|
| Camera2D* | camera |
| Window* | window |
| FPS* | fps |
| Audio* | audiodevice |
| SoundManager* | soundManager |
| Input* | in |

メソッド
EngineSystem()
EngineSystem(int,int,char*bool = false)
~EngineSystem()

bool Initialize()
引数　なし
戻り値 true false
初期化処理、new の後に行わなければならない処理がここで行われている。

void UpDate()
引数　なし
返り値　なし
更新処理、フィールドごとに必要な更新をここで呼び出している。

void SetWindow(int,int,char*,bool = false)
引数　Window の幅　Window の高さ　Window の名　Window モード
返り値　なし
生成する Window の情報を与える関数
これがない場合はこちらで設定している初期状態のものが出現する。
これを呼び出す場所は OGTask/OGTK/_myGameInitialize()内。
この関数が唯一 Window を生成する前に呼び出される関数になっている。
サンプルコード

```
OGTK::_myGameInitialize()
{
	OGge->SetWindow(960,540,"WindowName",false);
}
```

void SetWindowPos(Vec2&)
引数　生成する Window の位置。
戻り値　なし
生成する Window の位置を指定するもの。
フルスクリーンでの起動では無効化される。
こちらも_myGameInitialize()内で呼ぶことを想定して作られている。
サンプルコード

```
OGTK::_myGameInitialize()
{
	Vec2 winpos = {0,0};
	OGge->SetWindowPos(winpos);
}
```

void SetCursorOn(bool)
引数　見える状態 true　隠す場合 false
返り値　なし
生成した Window 上でマウスポインタを隠すかどうかを決める関数。
行われない場合 true のまま Window が生成される。
こちらも_muyGameInitialize()内で呼ぶ。
サンプルコード

```
OGTK::_myGameInitialize()
{
	OGge->SetCursorOn(false);
}
```

void SetIcon(std::string&)
引数　使用する画像のファイル名(パスは不要)
戻り値　なし
生成した Window に Icon を設定する関数。
設定しない場合こちらで準備したものを初期化に使われる。
画像は 16*16,32*32,48*48 が適正サイズとされている。
こちらも_myGameInitialize()内で呼ぶ。
サンプルコード

```
OGTK::_myGameInitialize()
{
	OGge->SetIcon((std::string)"testIcon.png");
}
```

void SetPause(bool)
引数　ポーズ処理を行うならば true　通常更新を行うならば false
戻り値　なし
bool GetPause()
引数　なし
戻り値　true false
更新を行う関数を変更する、その状態を持ってくるもの。

サンプルコード

```
if(OGge->in,down(In::B1))
{
	std::cout << OGge->GetPause();
	OGge->SetPause(true);
	std::cout << OGge->GetPause();
}
```

---

void GameEnd();
引数　なし
戻り値　なし
行うとアプリケーションを終了する関数。
bool GetEnd()
引数　なし
戻り値　true　false
アプリケーション終了を行うかどうかを返す関数。

サンプルコード

```
std::cout << OGge->GetEnd() ;
if(OGge->in.down(In::B1))
{
	OGge->GameEnd();
	std::cout << OGge->GetEnd();
}
```

---

01

void ChengeTask()
引数　なし
戻り値　なし
scene を移行するときなどに変更したカメラの位置やポーズ情報をもとに戻す関数。

サンプルコード

```
Title::~Title
{
	OGge->ChengeTask();
}
```

void SetTaskObject(TaskObject::SP&)
引数　生成したタスクオブジェクトのスマートポインタ
戻り値　なし

サンプルコード

```
Sample::SP test = Sample::SP(new Sample());
OGge->SetTaskObject(test);
```

void SetDeleteEngine(bool)
引数　終了 true　以外 false
戻り値
なし
bool GetDeleteEngine()
引数　なし
戻り値　true false
ゲームエンジンの終了確認。
使用用途はタスク scene 終了時に次のタスクを生成するかを決める時。

サンプルコード

```
Sample::Finalize()
{
	if(!GetDeleteEngine())
	{
		auto next = Sample2::Create();
	}
}
```

template <class T>std::shared_ptr<T> GetTask(std::string)
引数　探したいタスクの名
戻り値　TaskObject nullptr
引数の名のタスクを全タスクから検索をかける。同名の最初タスクが返される。もし全タスクの中に１つも存在しなかった場合は nullptr を返す

サンプルコード

```
auto test = OGge->GetTask<Sample>(“sample”);
if(!auto)
{
}
```

Camera/Camera2D

説明
2D 用のカメラを管理するもの。
基本的には OGge 内部で宣言されているものを使用する

メソッド
void SetPos(Vec2&)
引数　位置
戻り値　なし
void MovePos(Vec2&)
引数　　移動値
戻り値　なし
Vec2 GetPos()
引数　なし
戻り値　位置
2D 上のカメラの位置を設定、移動、返します。

void SetSize(Vec2&)
引数　サイズ
戻り値　なし
void MoveSize(Vec2&)
引数　移動値
戻り値　なし
Vec2 GetSize()
引数　なし
戻り値　サイズ
2D 上のカメラのサイズを設定、移動、返します。

サンプルコード

```
OGge->camera->SetPos(Vec2(10,10));
Vec2 pos = OGge->camera->GetPos(); //10,10
OGge->camera->SetSize(Vec2(1080,720));
Vec2 size = OGge->camera->GetSize(); //1080,720
```

Collision/CollisionBox

説明
矩形の当たり判定を行う。

フィールド

| | |
|---|---|
| Box2D | hitBase |
| float | angle |

メソッド
bool hitBox(CollisionBox&)
bool hitCircle(CollisionCircle&)
引数　相手の情報
返り値　当たっている場合 true それ以外 false
当たり判定を計算しその結果を bool で返すもの

void Rotate(float)
引数　回転値
戻り値　なし
内部回転値に引数を適用させる。
Collision/CollisionCircle

説明
円の当たり判定を行う。
フィールド

| | |
|---|---|
| Circle | hitBase |

メソッド
bool hitBox(CollisionBox&)
bool hitCiecle(CollisionCircle&)
引数　相手の情報
返り値　当たっている場合 true それ以外 false
当たり判定を計算しその結果を bool で返すもの

この Collision は基本的に GameObject で呼ぶため
特別形式でない場合は使用することはない。

Easing/Easing

説明
Easing を扱うことができるもの。

フィールド

| | |
|---|---|
| Linear | linear |
| Back | back |
| Bounce | bounce |
| Circ | circ |
| Cubic | cubic |
| Clastic | elastic |
| Expo | expo |
| Quad | quad |
| Quart | quart |
| Quint | quint |
| Sine | sine |

メソッド
Easing()
float Time(float)
引数　イージングを行う時間
返り値　現在タイム
それぞれの関数の引数に必要な時間を計算してくれる関数

bool isplay()
引数　なし
戻り値　イージングが行われている true それ以外　false
イージングを行っているかを bool で返す。

上記フィールド全てに In,OutInOut が存在する
引数は共通して(float,float,float,float)
引数　時間　始点　終点-始点　経過時間
戻り値　その時間に応じた始点から終点までの位置
詳細は Easing の動作説明(外部サイト)を確認してください。

サンプルコード

```
Easing easing_x;
Easing easing_y;
while(1)
{
	Vec2 pos;
	pos.x = easing_x.quint.InOut(10,0,100,easing_x.Time(10));
	pos.y = easing_y,quint.InOut(10,10,300,easing_y,Time(10));
}
```

Input/Input

説明
入力を使用、管理するもの。key と Pad 両方で反応するよう設定されているものを扱える。基本的には OGge 内で宣言されているものを使用する。

フィールド

| | |
|---|---|
| std::vector<GamePad> | pad |
| KeyBoard | key |
| Mouse | mouse |
| bool | Pad_Connection |

メソッド
bool on(int,int = 0)
bool up(int,int = 0)
bool down(int,int = 0)
引数　入力簡易空間に存在するもの　ゲームパッドの番号
返り値　true false
入力に応じてそれがされているかいないかを返すもの。
引数 1 には In 空間のものを使用できるが Key,Gpad のものも存在するため下記に存在する表を確認の上使用すること。
別のものを引数に入れた場合 DebugError が起きます。
Mouse の入力は Key と適用させていないので Mouse 空間を使用すること。

| in | キーボード番地 | ゲームパッド番地 |
|---|---|---|
| B1 | Z | A |
| B2 | X | B |
| B3 | C | X |
| B4 | V | Y |
| CU | ↑ | ↑ |
| CR | → | → |
| CD | ↓ | ↓ |
| CL | ← | ← |
| L1 | Q | LB |
| R1 | E | RB |

| | | |
|---|---|---|
| D1 | ENTER | BACK |
| D2 | SPACE | START |
| SR | N | 右スティック押し込み |
| SL | B | 左スティック押し込み |

ゲームパッド番地名は Logicool コントローラの配置名を使って記載しております。ほかのコントローラと名前が一致しない場合がございます。

サンプルコード

```
if(OGge->in->down(In::SL))
{}
if(OGge->in->up(Input::SR))
{}
while(OGge->in->on(Input::in::B4))
{}
```

Input/Input/KeyBord

説明
Inputclass に存在するキーボードの判定を行うもの

メソッド
bool on(int)
bool up(int)
bool down(int)
引数　入力簡易空間に存在するもの
返り値　true false
キーボードだけ用の入力判定

| key | キーボード番地 |
|---|---|
| A~Z | A~Z |
| ENTER | Enter |
| SPACE | Space |
| ESCAPE | ESC |

サンプルコード
if(OGge->in->key.down(In::SPACE))
if(OGge->in->key.down(Input::KeyBoard::Key::ENTER))
Input/Input/GamePad

説明
Inputclass に存在するキーボードの判定を行うもの

メソッド
bool on(int)
bool up(int)
bool down(int)
引数　入力簡易空間に存在するもの
返り値　true false
ゲームパッドだけ用の入力判定

float axis(int)
引数　入力簡易空間のスティックのもの
返り値　傾き度(-1.0 ~ 1.0 の範囲内)
ゲームパッドのスティック判定用

| pad | ゲームパッド番地 |
|---|---|
| BUTTON_A | A |
| BUTTON_B | B |
| BUTTON_X | X |
| BUTTON_Y | Y |
| BUTTON_L1 | LB |
| BUTTON_R1 | RB |
| BUTTON_BACK | BACK |
| BUTTON_START | START |
| BUTTON_L3 | 左スティック押し込み |
| BUTTON_R3 | 右スティック押し込み |
| BUTTON_U | ↑ |
| BUTTON_R | → |
| BUTTON_D | ↓ |
| BUTTON_L | ← |
| AXIS_LEFT_X | 左スティック X 座標値 |

| | |
|---|---|
| AXIS_LEFT_Y | 左スティック Y 座標値 |
| AXIS_RIGHT_X | 右スティック X 座標値 |
| AXIS_RIGHT_Y | 右スティック Y 座標値 |

サンプルコード
```
if(OGge->in->pad[0].down(In::BUTTON_A))
Vec2 pos;
pos.x += OGge->in->pad[0].axis(In::AXIS_LEFT_Y));
pos.y += OGge->in->pad[0].axis(Input::GamePad::AXIS::AXIS_LEFT_X));
```

Input/Input/Mouse

説明
マウス操作を行うためのもの。

フィールド

| | |
|---|---|
| bool | isPresent |

メソッド
bool on(int)
bool up(int)
bool down(int)
引数　入力簡易空間に存在するもの
返り値　true false
マウスだけ用の入力判定

Vec2 GetPos()
引数　なし
返り値　Window の位置から見たマウスの位置
マウスの位置を返すためのもの。

| | |
|---|---|
| Mouse | マウスのボタン |
| LEFT | 左ボタン |
| RIGTH | 右ボタン |
| CENTER | 中央ボタン |
| BUTTON_4 | |

| | |
|---|---|
| BUTTON_5 | |
| BUTTON_6 | |
| BUTTON_7 | |
| BUTTON_8 | |

4~8 は使用しているマウスに存在すれば使える。
モノによって変わるため未記入。

サンプルコード

```
if(OGge->in->mouse.down(Mouse::LEFT))

Vec2 pos = OGge->in->mouse.GetPos();
```

Random/random

説明
ランダム値を取得する名前空間

メソッド
int GetRand(int,int);
float GetRand(float,float)
引数　最小値　最大値
返り値　その間でのランダム値
最小値と最大値を渡すことでその中でのランダムを生成し返す。

srd::string GetRand(std::string&,std::size_t)
引数　ランダムに使用する文字列　生成するランダム数
返り値　サイズ分のランダム文字列
送った文字列の中にある文字からサイズ分のランダム文字列を生成して返す。

サンプルコード

```
int test_i = random::GetRand(0,10)
float test_t = random::GetRand(0.0f,10.f);
std::string test_s = random::GetRand("abcdefg",8);

//test_t = random::GetRand(0,1.5f);//error
```

---

test_i = 0 ~ 10
test_t = 0.0 ~ 10.0
test_s = a ~ g * 8

Audio/Audio

説明
オーディオのデバイスやバッファ、ソース、Wav ファイルの管理、再生を行う
OGge 内部ではデバイスの設定を行っている。
Sound ではソースとバッファを管理している。

Sound/Sound

説明
音を管理する。読み込めるファイルは.wav

メソッド
Sound()
Sound(std::string&,bool = false)
引数　ファイル名　ループ有無
戻り値　なし
初期処理に加え指定したファイルの読み込みを行う。

bool create(std::string&,bool = false)
引数　ファイル名　ループ有無
戻り値　成功 true それ以外　false
ファイルの読み込みを行う。

void play()
引数　なし
戻り値　なし
読み込んだ音ファイルを再生する。

void stop()
引数　なし
戻り値　なし
再生中の音を止める。

void pause()
引数　なし
戻り値　なし
再生中の音を一時停止する。

void volume(float)
引数　音量(0~1)
戻り値　なし
音量を設定します。最大値 1.0 最小値 0.0

void pitch(float)
引数　音程
戻り値　なし
ピッチを設定します。通常値 1.0

void looping(bool)
引数　ループの有無
戻り値　なし
ループ再生を行うかを設定します。

bool isplay()
引数　なし
戻り値　再生中 true　それ以外 false
現在再生を行っているかを返します。

float currenttime()
引数　なし
戻り値　現在の再生時間
再生時間を返します。

float duration()
引数　なし
戻り値　ファイルの再生時間
読み込んだファイルの全体の時間を返します

void setVolume(float)
引数　音量
戻り値　なし
SoundManager で行う際に設定する音量。これと volume()の関連はありません。

float getVolume()
引数　なし
戻り値　音量
SoundManager にて使用する音量。こちらも volume()との関連はありません。

サンプルコード

```
Sound sound;
//読み込み
sound.create(“test.wav”,true);
//音量 0.5
sound.volume(0.5f);
//再生
sound.play();
//一時停止
sound.pause();
//一時停止時点から再開
sound.play();
//停止
sound.stop();
//最初から再生
sound.play();
```

SoundManager/SoundManager

説明
サウンドの音量管理などを行うもの。OGge 内で宣言されており
基本的にはそれを使用する。

メソッド
SoundManager()

void SetMaxVolume(float)
引数　最大音量
戻り値　なし
最大音量を設定します。0.0 から 1.0 までが有効範囲です。

float GetMaxVolume()
引数　なし
戻り値　最大音量
現在設定されている最大音量を返します。初期設定は 1.0 です。

void SetSound(Sound*)
引数　サウンドポインタ
戻り値　なし
最大音量を設定したいサウンドを登録します。

bool DeleteSound(Sound*)
引数　削除したいサウンドポインタ
戻り値　成功 true それ以外 false
登録しているサウンドデータを削除します。

void AllDelete()
引数　なし
戻り値　なし
登録している全データを削除します。

void Application()
引数　なし
戻り値　なし
登録している Sound すべてに最大音量を適用させます。

サンプルコード

```
OGge->soundManager->SetMaxVolume(0.5f);

Sound sound("test.wav");

sound.setVolume(0.7f);

OGge->soundManager->SetSound(&sound);

OGge->soundManager->Application();

sound.play();
//この場合 0.5 が最大とした 0.7 の値である 0.35 の音量で再生されます。
```

Texture/Texture

説明
画像を扱うもの。

メソッド
Texture()

bool Create(std::string&)
引数　画像ファイル名
戻り値　成功 true それ以外 false
画像ファイルを読み込みます。

void Draw(Box2D,Box2D,Color = {1,1,1,1});
引数　Window 上の位置　画像上の位置　色
戻り値　なし
画面上に読み込んだ画像を描画します。

void Rotate(float)
引数　角度
戻り値　なし
画像を回転させます。引数分角度を変えます。

bool Finalize()
引数　なし
戻り値　成功 true それ以外 false
画像データを破棄します。

サンプルコード

```
Texture test;

test.Create(“test.png”);

Box2D draw(10,10,64,64);
drw.OffsetSize();

Box2D src(0,64,64,64);
src.OffsetSize();

test.Rotate(35.f);

test.Draw(draw,src);

test.Finalize();

//test.png 画像を 10,10 位置から 64,64 サイズで 35 度回転して描画します。
//その画像は元画像の 0,64 ピクセルから 64,64 ピクセル分の画像データ
```

glTimer/Time

説明
現実時間を使った計測を行うもの。

メソッド
Time()

void Start()
引数　なし
戻り値　なし
計測を開始します。

void Stop()
引数　なし
戻り値　なし
計測を止めます。

void Pause()
引数　なし
戻り値　なし
計測を一時停止します。

float GetTime()
引数　なし
戻り値　現在の計測タイム
現在の経過タイムを返します。

void InitTime(float)
引数　初期タイムの値
戻り値　なし
計測開始時のタイムを 0 以外にすることができる。
初期時は 0

bool isplay()
引数　なし
戻り値　計測中 true それ以外 false
現在計測を行っているかどうかを返します。

サンプルコード

```
Time　time;

time.InitTime(5.0f);
time.Start();
while(1)
{
	if(time.GetTime() >= 10.f)
	{
		break;
	}
}
time.Stop();

//5 秒から計測を始め 10 秒以上になったとき無限ループから抜けます。
```

Window/Window

説明
生成する Window とそれに必要となる情報を保持管理を行う。
OGge 内で生成と設定を行う。基本的にはそこにあるデータを使用、変更を行う。

メソッド
Window()
Window(int,int,char*bool,Vec2&)
引数　幅　高さ　名前　モード　位置
戻り値　なし
生成する際の初期化情報を設定する場合に使用。

bool createWindow(int,int,char*bool,Vec2&)
引数　幅　高さ　名前　モード　位置
戻り値　成功 true それ以外 false
Window を実際に生成しています。コンストラクタで生成している場合は呼ばないこと。※例外エラーが起きます。

void setIcon(std::string&)
引数　ファイル名
戻り値　なし
生成する Window のアイコンを設定します。
初期設定ではコンストラクタで設定している初期情報になっています。

void LimitsWindow()
引数　なし
戻り値　なし
サイズを現在の幅と高さで固定させアスペクト比を 16*9 で固定させる。

void WindowIcon()
引数　なし
戻り値　なし
Window をアイコン化、アイコン化状態ならばそれを解除します。

void Visualization()
引数　なし
戻り値　なし
Window を可視化、可視化状態ならばそれを解除します。

void InMouseMode(bool)
引数　マウスの表示有無
戻り値　なし
true でマウスを表示、false で非表示にします。(位置情報は取得できます)

Vec2 GetSize()
引数　なし
戻り値　サイズ
生成されている Window のサイズを返します。

Vec2 GetPos()
引数　なし
戻り値　位置
生成されている Window の位置を返します。

void SetWindowPos(Vec2&)
引数　位置
戻り値　なし
生成する Window の位置を指定します。

サンプルコード
```
OGge->window->InMouseMode(false);
OGge->window->setIcon(“test.png”);
Vec2 pos = OGge->window->GetPos();
Vec2 size = OGge->window->GetSize();
```

OGTask/OGTK

説明
初期化時の値渡しを行ったり生成する初期タスクを設定したりしてくれるもの。

メソッド
void _myGameInitialize()
引数　なし
戻り値　なし
Window を生成する前に行う設定などを行う場所。

void StartTaskObject()
引数　なし
戻り値　なし
初期時に生成するタスクなんかを設定する場所。

| | | |
|---|---|---|
| 履歴 | 2018/03/21 | 記入 |
| 履歴 | 2018/03/23 | 追記 |
| 履歴 | 2018/04/01 | 追記 |
| 履歴 | 2018/04/09 | 追記 |
| 履歴 | 2018/05/10 | 追記 |