Game Framework

リファレンス・マニュアル

Satoshi Numata - 2017年6月29日

目次

1. はじめに

1.1. Game Frameworkについて

何か書く。

1.2. Unityへの準拠

何か書く。

2. 関数の一覧

2.1. 基本のグラフィック関数

画面のクリア

void Clear(color)　　色を指定して画面をクリア

2.2. 単純図形の線描画

円の描画

void DrawCircle(center, radius_f, color)　　1色指定して円を描画
void DrawCircle(center, radius_f, width, color)　　1色指定して円を描画
void DrawCircle(center, radius_f, width, startRad, endRad, color)
開始角・終了角と1色を指定して円を描画
void DrawCircle(center, radius_f, width, startRad, endRad, startColor, endColor)
開始角・終了角と2色を指定して円を描画

void DrawCircle(center, radius_v2, color)　　XY別の半径と1色指定して円を描画
void DrawCircle(center, radius_v2, width, color)　XY別の半径と1色指定して円を描画
void DrawCircle(center, radius_v2, width, startRad, endRad, color)
XY別の半径と開始角・終了角と1色を指定して円を描画
void DrawCircle(center, radius_v2, width, startRad, endRad, startColor, endColor)
XY別の半径と開始角・終了角と2色を指定して円を描画

直線の描画

void DrawLine(p1, p2, color)　　1色指定して直線を描画
void DrawLine(p1, p2, color1, color2)　　2色指定して直線を描画
void DrawLine(p1, p2, width, color)　　1色指定して直線を描画
void DrawLine(p1, p2, width, color1, color2)　　2色指定して直線を描画

点の描画

void DrawPoint(pos, color)　　点を描画

四角形（矩形）の描画

void DrawRect(rect, color)　　1色指定して矩形を描画
void DrawRect(rect, color1, color2, color3, color4)　　4色指定して矩形を描画
void DrawRect(rect, width, color)　　1色指定して矩形を描画
void DrawRect(rect, width, color1, color2, color3, color4)　　4色指定して矩形を描画

三角形の描画

void DrawTriangle(p1, p2, p3, color)　　1色指定して三角形を描画
void DrawTriangle(p1, p2, p3, color1, color2, color3)　　3色指定して三角形を描画

void DrawTriangle(p1, p2, p3, width, color)　　1色指定して三角形を描画
void DrawTriangle(p1, p2, p3, width, color1, color2, color3)　3色指定して三角形を描画

2.3. 単純図形の塗りつぶし描画

円の塗りつぶし

void FillCircle(center, radius_f, color)　　1色指定して円を塗りつぶし
void FillCircle(center, radius_f, startRad, endRad, color)
　　開始角・終了角と1色を指定して円を塗りつぶし
void FillCircle(center, radius_f, startRad, endRad, startColor, endColor)
　　開始角・終了角と2色を指定して円を塗りつぶし
void FillCircle(center, radius_v2, color)　　XY別の半径と1色を指定して円を塗りつぶし
void FillCircle(center, radius_v2, startRad, endRad, color)
　　XY別の半径と開始角・終了角と1色を指定して円を塗りつぶし
void FillCircle(center, radius_v2, startRad, endRad, startColor, endColor)
　　XY別の半径と開始角・終了角と2色を指定して円を塗りつぶし

四角形（矩形）の塗りつぶし

void FillRect(rect, color)　　1色指定して矩形を塗りつぶし
void FillRect(rect, color1, color2, color3, color4)　　4色指定して矩形を塗りつぶし

三角形の塗りつぶし

void FillTriangle(p1, p2, p3, color)　　1色指定して三角形を塗りつぶし
void FillTriangle(p1, p2, p3, color1, color2, color3)　3色指定して三角形を塗りつぶし

2.4. テキストの描画

文字列の描画

void DrawText(pos, str, color)　　テキストを描画
void DrawText(pos, str, color1, color2)　　テキストを上下2色指定して描画
void DrawTextCenter(pos, str, color)　　中央位置を指定して中央寄せでテキストを描画
void DrawTextCenter(pos, str, color1, color2)　中央寄せでテキストを上下2色指定して描画
void DrawTextRight(pos, str, color)　　右端位置を指定して右寄せでテキストを描画
void DrawTextRight(pos, str, color1, color2)　右寄せでテキストを上下2色指定して描画

描画フォントの指定

void SetFont(fontName, fontSize)　　テキスト描画用のフォントを指定
　　（システムフォントはPostScript名で指定）
　　（**カスタムフォント**は **[Add to targets] のチェックボックスを必ず付けてプロジェクトに追加**した後、**拡張子を含めてファイル名を指定**）

文字描画サイズの取得

Vector2 GetTextSize()　　DrawText()で描画されるサイズ

2.5. 画像の描画

単純な画像の描画

void DrawImage(filename, pos)	座標指定で画像を描画
void DrawImage(filename, pos, center)	座標と中心指定で画像を描画
void DrawImage(filename, pos, color)	座標と色指定で画像を描画
void DrawImage(filename, pos, center, color)	座標, 中心, 色指定で画像を描画
void DrawImage(filename, pos, color1, color2, color3, color4)	座標と4色指定で画像を描画
void DrawImage(filename, pos, center, color1, color2, color3, color4)	座標, 中心, 4色指定で画像を描画

描画元矩形を指定する画像の描画

void DrawImage(filename, pos, srcRect)	描画元矩形を指定して画像を描画
void DrawImage(filename, pos, srcRect, center)	描画元矩形, 中心を指定して画像を描画
void DrawImage(filename, pos, srcRect, color)	描画元矩形と色を指定して画像を描画
void DrawImage(filename, pos, srcRect, center, color)	描画元矩形, 中心, 色を指定して画像を描画
void DrawImage(filename, pos, srcRect, color1, color2, color3, color4)	描画元矩形と4色を指定して画像を描画
void DrawImage(filename, pos, srcRect, center, color1, color2, color3, color4)	描画元矩形, 中心, 4色を指定して画像を描画

描画先矩形を指定する画像の描画

void DrawImage(filename, destRect)	描画先矩形を指定して画像を描画 画像サイズは長辺に合わせてリサイズ
void DrawImage(filename, destRect, color)	描画先矩形と色を指定して画像を描画 画像サイズは長辺に合わせてリサイズ
void DrawImageFit(filename, destRect)	描画先矩形を指定して画像を描画 画像サイズは矩形に合わせてリサイズ
void DrawImageFit(filename, destRect, color)	描画先矩形と色を指定して画像を描画 画像サイズは矩形に合わせてリサイズ

描画先の中心座標を指定する画像の描画

void DrawImageCenter(filename, center)	描画先の中心座標を指定して画像を描画
void DrawImageCenter(filename, center, color)	描画先の中心座標と色を指定して画像を描画
void DrawImageCenter(filename, center, scale_f)	描画先の中心座標とスケールを指定して描画
void DrawImageCenter(filename, center, scale_f, color)	描画先の中心座標, スケール, 色を指定して描画

画像サイズの取得

Vector2 GetImageSize(filename)	画像のサイズを取得

2.6. 描画の調整

ブレンドモードの設定

void SetBlendMode(mode)	ブレンドモードの設定（「5. ブレンドモード」を参照）

レイヤー深度の設定

void SetLayer(int)	描画時のレイヤー深度の設定（高いほど手前に描画）

2.7. カメラの設定

カメラ位置の設定

void SetCameraTranslation(x, y)	カメラの位置を設定
void SetCameraTranslation(vec)	カメラの位置を設定

カメラ回転角の設定

void SetCameraRotation(rad)	カメラの回転角を設定（ラジアン単位）

カメラのスケールの設定

void SetCameraScale(scale_f)	カメラのスケールを設定
void SetCameraScale(scale, scale)	カメラのスケールを設定
void SetCameraScale(scale_v)	カメラのスケールを設定

カメラの位置・回転角・スケールの同時設定

void SetCameraTRS(pos, rad, scale_f)	カメラの位置・回転角・スケールを設定
void SetCameraTRS(pos, rad, scale_v)	カメラの位置・回転角・スケールを設定
void ResetCameraTRS()	カメラの設定をリセット

2.8. モデル描画の設定

モデル位置の設定

void SetModelTranslation(x, y)	モデルの位置を設定
void SetModelTranslation(vec)	モデルの位置を設定

モデル回転角の設定

void SetModelRotation(rad)	モデルの回転角を設定（ラジアン単位）
void SetModelScale(scale_f)	モデルのスケールを設定

モデルのスケールの設定

void SetModelScale(x, y)	モデルのスケールを設定
void SetModelScale(scale_v)	モデルのスケールを設定

モデルの位置・回転角・スケールの同時設定

void SetModelTRS(pos, rad, scale_f)	モデルの位置・回転角・スケールを設定
void SetModelTRS(pos, rad, scale_v)	モデルの位置・回転角・スケールを設定

void ResetModelTRS()	モデルの描画位置をリセット

2.9. ユーザ入力の取得（キーボード）

キーボード状態のチェック

bool Input::GetKey(keyMask)	キーボードの押下状態をチェック
bool Input::GetKeyDown(keyMask)	キーボードが直前フレームで押されたかチェック
bool Input::GetKeyUp()	キーボードが直前フレームで離されたかチェック

仮想軸の値の取得

float Input::GetAxis(axisName)	仮想軸の値を取得
float Input::GetAxisRaw(axisName)	仮想軸の値を取得（平滑化フィルターなし）
void Input::ResetInputAxes()	仮想軸の値をリセット

2.10. ユーザ入力の取得（マウス）

マウスボタンの押下チェック

bool Input::GetMouseButton(int)	マウスボタンの押下状態をチェック
bool Input::GetMouseButtonDown(int)	マウスボタンが直前フレームで押されたかチェック
bool Input::GetMouseButtonUp(int)	マウスボタンが直前フレームで離されたかチェック

カーソル位置の取得

Vector2 Input::MousePosition()	マウスの現在位置を取得（カメラ設定考慮なし）
Vector2 GetMousePosition()	マウスの現在位置を取得（カメラ設定反映）

2.11. BGMの再生

void PlayBGM(filename)	BGMを再生（ループ再生）
void PlayBGM()	一時停止していたBGMの再生を再開
void PauseBGM()	BGMを一時停止
void SetBGMVolume(volume)	BGMの音量を設定
void StopBGM()	BGMの再生を停止（再生再開時には冒頭から開始）

2.12. 効果音の再生

void PlaySound(filename)	効果音を再生（再生中の場合、中断されて冒頭から開始）
void PlaySound(filename, volume)	効果音を再生（再生中の場合、中断されて冒頭から開始）

2.13. 画面サイズの取得

《変数》float Screen::width	画面の横のサイズ（ピクセル単位）
《変数》float Screen::height	画面の縦のサイズ（ピクセル単位）
《関数》Vector2 Screen::size()	画面サイズの取得（ピクセル単位）
《関数》Rect GetScreenBounds()	画面枠の取得（ピクセル単位）

2.14.時間の操作

《変数》int Time::frameCount　　ゲーム開始時からの経過フレーム数
《変数》float Time::time　　ゲーム開始時からの経過時間
《変数》float Time::deltaTime　　直前フレームから現在のフレームまでの経過時間
《変数》float Time::timeScale　　時間経過のスケール値
《変数》float Time::unscaledDeltaTime
　　スケール値の影響を受けない直前フレームから現在のフレームまでの経過時間
《変数》float Time::unscaledTime
　　スケール値の影響を受けないゲーム開始時からの経過時間

2.15.文字列操作のヘルパー関数

C言語書式でのC++文字列の作成

string FormatString(format, ...)　　C言語の書式に従ってC++文字列を生成（1024バイトまで）

部分文字列への分解

vector<string> Split(str, separator)　　指定された文字に基づいて文字列を分解

先頭・末尾の文字列のチェック

bool StartsWith(str, value)　　文字列の先頭をチェック
bool StartsWith(str, value, ignoreCase)　　文字列の先頭をチェック
bool EndsWith(str, value)　　文字列の末尾をチェック
bool EndsWith(str, value, ignoreCase)　　文字列の末尾をチェック

大文字・小文字の変換

string ToLower(str)　　空白文字を先頭と末尾から削除した文字列を作成
string ToUpper(str)　　指定した文字を先頭と末尾から削除した文字列を作成

先頭・末尾からの指定文字の削除

string Trim(str)　　空白文字を先頭と末尾から削除した文字列を作成
string Trim(str, trimChars)　　指定した文字を先頭と末尾から削除した文字列を作成

2.16.デバッグ出力

void Debug()　　画面上に描画するデバッグ文字列を1行分改行
void Debug(const char* format, ...)　　画面上にデバッグ文字列を左上から描画
　　（書式はC言語標準のprintf()関数に準拠）

void DebugLog(...)　　デバッグ出力領域にデバッグ文字列を出力します。
　　（書式はC言語標準のprintf()関数に準拠）
void DebugLogWarning(...)　　デバッグ出力領域に警告用のデバッグ文字列を出力します。
　　（書式はC言語標準のprintf()関数に準拠）
void DebugLogError(...)　　デバッグ出力領域にエラー用のデバッグ文字列を出力します。
　　（書式はC言語標準のprintf()関数に準拠）

3. 乱数の発生

3.1. 乱数シードの取得と設定

unsigned Random::GetSeed()

現在設定されている乱数シードを取得します。

void Random::SetSeed(unsigned seed)

新たな乱数シードを設定します。

3.2. 浮動小数点値の乱数発生

float Random::FloatRange(float min, float min)

min 以上 max 以下の浮動小数点型の乱数を発生させます。

float Random::FloatValue()

0.0f 以上 1.0f 以下の浮動小数点型の乱数を発生させます。

3.3. 整数値の乱数発生

int Random::IntRange(int min, int max)

min 以上 max 以下の整数型の乱数を発生させます。

int Random::IntValue()

0 以上 INT_MAX 以下の整数型の乱数を発生させます。

int Random::IntValue(int upper)

0 以上 upper 未満の整数型の乱数を発生させます。

4. テキスト情報の読み込み

4.1. テキストファイルの読み込み

基本的なテキストファイルの読み込みには、ファイル名を指定してFileオブジェクトを作成し、ReadLine()関数を使います。

【テキストファイルの読み込み例】

```
void Start()
{
    File file("test.txt");
    string str;
    while (file.ReadLine(str)) {
        DebugLog(str);
    }
}
```

4.2. XMLファイルの読み込み

XmlDocumentクラスを使用することで、構造化されたXMLドキュメントの情報が読み込めます。

【XMLファイルの読み込み例】

```
///// test.xml
// <item x="30" y="20.5" />
// <item x="10" y="30" />
void Start()
{
    vector<Vector2> list;

    XmlDocument doc("test.xml");
    while (!doc.IsAtEnd()) {
        float x = doc.GetFloatAttribute("x");
        float y = doc.GetFloatAttribute("y");
        list.push_back(Vector2(x, y));
        doc.MoveToNextSibling();
    }
    ...
}
```

5. 数学関数

5.1. C言語標準の数学関数

各種の定数

《変数》FLT_EPSILON	ごくわずかな値を表す浮動小数点の定数です。
《変数》INFINITY	無限大を表す定数です。負の無限大は -INFINITY で表します。
《変数》M_PI	円周率を表す定数です。

三角関数

float acosf(float value)	value のアークコサインを計算します。
float asinf(float value)	value のアークサインを計算します。
float atanf(float value)	value のアークタンジェントを計算します。
float atan2f(float y, float x)	タンジェントが y/x になる角度をラジアンで返します。
float cosf(float rad)	ラジアン単位の角度に対するコサインを計算します。
float sinf(float rad)	ラジアン単位の角度に対するサインを計算します。
float tanf(float rad)	ラジアン単位の角度に対するタンジェントを計算します。

数値計算

float ceilf(float value)	value 以上の最小の整数を返します。
float expf(float x)	ネイピア数 e を x 乗した数を計算します。
float fabsf(float value)	value の絶対値を計算します。
float floorf(float value)	value 以下の最大の整数を返します。
float logf(float x)	自然対数の底 e に対する数 x の対数を計算します。
float log2f(float x)	2を底とする数 x の対数を計算します。
float log10f(float x)	常用対数の底10に対する数xの対数を計算します。
const T& max(const T& a, const T& b)	a と b のうち最大の値をリターンします。
const T& min(const T& a, const T& b)	a と b のうち最小の値をリターンします。
float powf(float x, float y)	x の y 乗を計算します。
float roundf(float value)	value に最も近い整数を返します（四捨五入）。
float sqrtf(float value)	value の平方根を計算します。

5.2. Unity準拠の数学関数

度とラジアンの変換のための定数

《変数》const float Mathf::Deg2Rad	度数法の値に掛けてラジアン単位の値に変換
《変数》const float Mathf::Rad2Deg	ラジアン単位の値に掛けて度数法の値に変換

数値補完

float Mathf::InverseLerp(a, b, value)	[a, b]の範囲内で補間された値valueを生成する線形パラメータを計算
float Mathf::Lerp(a, b, t)	aとbの間でtによる線形補間を計算（tは[0, 1]の範囲に制限）
float Mathf::LerpAngle(a, b, t)	度単位の角度a, bの間でtによる線形補間を計算（tは[0, 1]の範囲に制限）

float Mathf::LerpUnclamped(a, b, t)　度単位の角度a, bの間でtによる線形補間を計算
（tの範囲は制限なし）
float Mathf::MoveTowards(current, target, maxDelta)
Lerp()と同様に線形補間を行うが、速度がmaxDeltaを
超えないよう調整。maxDeltaが負の場合targetから遠ざかる。
float Mathf::MoveTowardsAngle(current, target, maxDelta)
LerpAngle()と同様に線形補間を行うが、速度がmaxDeltaを
超えないよう調整。maxDeltaが負の値はサポートされていない。
ターゲットからcurrentを遠ざけるには角度に180を加える。
float Mathf::SmoothDamp(current, target, outCurrentVelocity, smoothTime, maxSpeed, deltaTime)
徐々に時間をかけて望む目標に向かって値を変更。
値はバネとダンパーのような関数で滑らかに計算される。
float Mathf::SmoothDampAngle(current, target, outCurrentVelocity, smoothTime, maxSpeed, deltaTime)
度単位の角度を徐々に時間をかけて望む目標に向かって変更。
値はバネとダンパーのような関数によって滑らかに計算される。
float Mathf::SmoothStep(from, to, t)　fromからtoの値を、開始時には徐々に加速し、終了時に
徐々に減速していくやり方で補間。tは[0, 1]の範囲に制限。

数値計算

bool Mathf::Approximately(a, b)	2つの値がほぼ等しいかどうかを判定
float Mathf::Clamp(value, min, max)	value の値を min～max の範囲に制限
float Mathf::Clamp01(value)	value の値を 0.0f～1.0f の範囲に制限
int Mathf::ClosestPowerOfTwo(value)	value にもっとも近い2のべき乗の値を計算
float Mathf::DeltaAngle(current, target)	度単位の2つの角度間の最小の角度の差を計算
bool Mathf::IsPowerOfTwo(value)	値が2の乗数かどうかをチェック
float Mathf::Log(f, p)	底pに対する数pの対数を計算
int Mathf::NextPowerOfTwo(value)	value 以上の2のべき乗の値を計算
float Mathf::PingPong(t, length)	0以上length以下の範囲で値tを行き来させる
float Mathf::Repeat(t, length)	0以上length以下の範囲で値tをループさせる
float Mathf::Sign(float value)	ゼロ以上の数であれば1、そうでなければ-1

5.3. Unityに準拠しない数学関数

数値補完

float Mathf::EaseIn(a, b, t)	aとbの間でパラメータtによるEase-In補完を計算
float Mathf::EaseInOut(a,b, t)	aとbの間でパラメータtによるEase-InOut補完を計算
float Mathf::EaseOut(a, b, t)	aとbの間でパラメータtによるEase-Out補完を計算

6. 基本の型

6.1. Color型

色を表す構造体です。

色を表す定数については「6. 色の指定」を参照してください。

コンストラクタ

Color()	(0, 0, 0, 1)の色を作成
Color(float r, float g, float b)	赤、緑、青の各色成分を指定して作成
Color(float r, float g, float b, float a)	赤、緑、青、アルファ値の各色成分を指定して作成
Color(int color)	0xffcc99のようなHTML形式の16進数表記の値で色を作成
Color(string& str)	"ffcc99"のようなHTML形式の色指定文字列で色を作成
Color(const Color& color)	コピーコンストラクタ

Public変数

float a	0.0f～1.0f の範囲で表されるアルファ値です。
float b	0.0f～1.0f の範囲で表される青の色要素です。
float g	0.0f～1.0f の範囲で表される緑の色要素です。
float r	0.0f～1.0f の範囲で表される赤の色要素です。

Public関数

Color Alpha(float alpha)	アルファ値を変更した色を作成
Color Blue(float blue)	青の色要素を変更した色を作成
Color Green(float green)	緑の色要素を変更した色を作成
Color Red(float red)	赤の色要素を変更した色を作成
const char* c_str()	デバッグ文字列（C言語）を取得
const char* c_str(const string& format)	デバッグ文字列（C言語）を取得（書式指定）
string ToString()	デバッグ文字列（C++）を取得
string ToString(const string& format)	デバッグ文字列（C++）を取得（書式指定）

Static関数

Color EaseIn(a, b, t)	2つの色の間を Ease-In 補間した色を作成
Color EaseInOut(a, b, t)	2つの色の間を Ease-InOut 補間した色を作成
Color EaseOut(a, b, t)	2つの色の間を Ease-Out 補間した色を作成
Color HSVToRGB(H, S, V)	HSV 色空間の値から RGB 色空間の色を作成
Color HSVToRGB(H, S, V, hdr)	HSV 色空間の値から RGB 色空間の色を作成 （hdrがtrueなら各色要素のクランプなし）
Color Lerp(a, b, t)	2つの色の間を線形補間した色を作成（t は[0, 1]の範囲に制限）
Color LerpUnclamped(a, b, t)	2つの色の間を線形補間した色を作成（tの範囲制限なし）
void RGBToHSV(rgbColor, outH, outS, outV)	RGB 色空間の値から HSV 色空間の値を取得
Color SmoothStep(a, b, t)	2つの色を、開始時には徐々に加速し、終了時に徐々に減速していくやり方で補間。tは[0, 1]の範囲に制限。

6.2. Rect型

矩形を表す構造体です。

コンストラクタ

Rect()	(0, 0, 0, 0)で初期化
Rect(float x, float y, float width, float height)	始点とサイズを指定して作成
Rect(const Vector2& pos, const Vector2& size)	始点とサイズを指定して作成
Rect(const Rect& rect)	コピーコンストラクタ

Public変数

float x	始点のX座標の値
float y	始点のY座標の値
float width	横方向の大きさ
float height	縦方向の大きさ

Public関数

Vector2 Center()	中心点の座標を取得
bool Contains(pos)	点が内部にあるかをチェック
bool Contains(pos)	点が内部にあるかをチェック
bool Contains(rect)	矩形が内部にあるかをチェック
Vector2 Max()	最大の点を取得
Vector2 Min()	最小の点を取得
bool Overlaps(rect)	矩形が重なっているかをチェック
Vector2 Position()	始点の座標を取得
void Set(x, y, width, height)	各成分を設定
Vector2 Size()	矩形のサイズを取得
float xMax()	X方向の最大の座標を取得
float yMax()	Y方向の最大の座標を取得
float xMin()	X方向の最小の座標を取得
float yMin()	Y方向の最小の座標を取得
const char* c_str()	デバッグ文字列（C言語）の取得
const char* c_str(format)	デバッグ文字列（C言語）の取得（書式指定）
string ToString()	デバッグ文字列（C++）の取得
string ToString(format)	デバッグ文字列（C++）の取得（書式指定）

Static関数

Rect MinMaxRect(xmin, ymin, xmax, ymax)	最大・最小のX・Yの座標値を指定して矩形を作成
NormalizedToPoint(rect, normCoord)	正規化座標を指定して、長方形内部の位置を計算
Vector2 PointToNormalized(rect, point)	ポイントに対応する正規化座標を計算

6.3. Vector2型

コンストラクタ

Vector2()	(0, 0)で初期化
Vector2(float x, float y)	X座標とY座標を指定して初期化
Vector2(const Vector2& vec)	コピーコンストラクタ

Public変数

float x	X座標の値
float y	Y座標の値

Public関数

float Magnitude()	ベクトルの長さを取得します。
Vector2 Normalized()	大きさを1に正規化したベクトルを返します。
void Normalize()	このベクトルの大きさを1に正規化します。
float SqrMagnitude()	ベクトルの長さの2乗を取得します。
void Set(float x, float y)	既存の Vector2 に x、y の成分を設定します。
const char* c_str()	デバッグ文字列（C言語）の取得
const char* c_str(format)	デバッグ文字列（C言語）の取得（書式指定）
string ToString()	デバッグ文字列（C++）の取得
string ToString(format)	デバッグ文字列（C++）の取得（書式指定）

Static関数

float Angle(from, to)	2点間の角度を返します。
Vector2 ClampMagnitude(vec, maxLen)	大きさをmaxLenにクランプしたベクトルを作成
float Cross(a, b)	2つのベクトルの外積を計算
float Distance(a, b)	2つのベクトル間の距離を計算
float Dot(a, b)	2つのベクトルの内積を計算
Vector2 EaseIn(a, b, t)	ベクトルa, b間でEase-In 補間を計算
Vector2 EaseInOut(a, b, t)	ベクトルa, b間でEase-InOut 補間を計算
Vector2 EaseOut(a, b, t)	ベクトルa, b間でEase-Out 補間を計算
Vector2 Lerp(a, b, t)	ベクトルa, b間で線形補間を計算（tは[0, 1]に制限）
Vector2 LerpUnclamped(a, b, t)	ベクトルa, b間で線形補間を計算（tの制限なし）
Vector2 Max(a, b)	ベクトルa,bの各成分の最大の要素からなるベクトル
Vector2 Min(a, b)	ベクトルa,bの各成分の最小の要素からなるベクトル
Vector2 MoveTowards(current, target, maxDelta)	現在位置をターゲットの方向に移動
Vector2 Reflect(inDirection, inNormal)	法線を使って反射させたベクトルを計算
Vector2 Scale(a, b)	2つのベクトルの各成分を乗算
Vector2 SmoothDamp(current, target, currentVelocity, smoothTime, maxSpeed, deltaTime)	ベクトルを指定されたゴールに徐々に近づける
Vector2 SmoothStep(a, b, t)	2つのベクトルを、開始時には徐々に加速し、終了時に徐々に減速していくやり方で補間。tは[0, 1]の範囲に制限。

7. 色の指定

7.1. 色の名前

Game Frameworkで利用可能な色の名前の一覧です。

白・黒・透明色

Color::black	(0, 0, 0, 1) の黒
Color::white	(1, 1, 1, 1) の白
Color::clear	(0, 0, 0, 0) の透明色を表す色

基本の色

Color::blue	(0, 0, 1, 1) の青
Color::cyan	(0, 1, 1, 1) のシアン
Color::gray	(0.5, 0.5, 0.5, 1) のグレー
Color::green	(0, 1, 0, 1) の緑
Color::orange	(1, 0.5, 0, 1) のオレンジ
Color::pink	(1, 0.25, 0.6, 1) のピンク
Color::purple	(1, 0, 1, 1) の紫
Color::red	(1, 0, 0, 1) の赤
Color::yellow	(1, 1, 0, 1) の黄色

明るい色

Color::lightblue	(0.5, 0.5, 1, 1) の明るい青
Color::lightcyan	(0.65, 1, 1, 1) の明るいシアン
Color::lightgray	(0.75, 0.75, 0.75, 1) の明るいグレー
Color::lightgreen	(0.5, 1, 0.5, 1) の明るい緑
Color::lightorange	(1, 0.75, 0.5, 1) の明るいオレンジ
Color::lightpink	(1, 0.5, 0.8, 1) の明るいピンク
Color::lightpurple	(1, 0.5, 1, 1) の明るい紫
Color::lightred	(1, 0.5, 0.5, 1) の明るい赤
Color::lightyellow	(1, 1, 0.65, 1) の明るい黄色

暗い色

Color::darkblue	(0, 0, 0.7, 1) の暗い青
Color::darkcyan	(0, 0.5, 0.5, 1) の暗いシアン
Color::darkgray	(0.25, 0.25, 0.25, 1) の暗いグレー
Color::darkgreen	(0, 0, 0.5, 1) の暗い青
Color::darkorange	(0.5, 0.25, 0, 1) の暗いオレンジ
Color::darkpink	(0.6, 0.08, 0.3, 1) の暗いピンク
Color::darkpurple	(0.5, 0, 0.5, 1) の暗い紫
Color::darkred	(0.6, 0, 0, 1) の暗い赤
Color::darkyellow	(0.5, 0.5, 0, 1) の暗い黄色

7.2. 色のアルファ値の変更

色を表すColorクラスに用意されているPublic関数のAlpha()関数を使い、「Color::red.Alpha(0.5f)」のように書くことによって、既存の色のアルファ値を変更して、半透明な色を作ることができます。

なお、アルファ値を変更した色を作るAlpha()関数のほかに、赤要素の値を変更した色を作るRed()関数、緑要素の値を変更した色を作るGreen()関数、青要素の値を変更した色を作るBlue()関数も用意されています。

【プログラム例】

```
void Update()
{
    Clear(Color::black);

    FillCircle({-50, 0}, 100, Color::red.Alpha(0.5f));
    FillCircle({50, 0}, 100, Color::red.Alpha(0.5f));
}
```

8. ブレンドモード

Game Frameworkで利用可能なブレンドモードは、次のとおりです。

BlendModeAlpha	アルファ合成
BlendModeAdd	加算合成
BlendModeClear	クリア合成
BlendModeCopy	コピー合成
BlendModeInvert	反転合成
BlendModeMultiply	乗算合成
BlendModeScreen	スクリーン合成
BlendModeXOR	XOR合成

9. キー定数

Game Frameworkで利用可能なキー定数の一覧です。

英数字キー

KeyMask::A ～ KeyMask::Z	A～Zの英字キー
KeyMask::Alpha0 ～ KeyMask::Alpha9	0～9の数字キー

上下左右キー

KeyMask::UpArrow	上矢印キー
KeyMask::DownArrow	下矢印キー
KeyMask::LeftArrow	左矢印キー
KeyMask::RightArrow	右矢印キー

それ以外のキー

KeyMask::Space	spaceキー
KeyMask::Escape	escキー
KeyMask::Return	returnキー
KeyMask::Backspace	backspaceキー
KeyMask::Delete	deleteキー
KeyMask::Tab	tabキー
KeyMask::LeftShift	左のshiftキー
KeyMask::RightShift	右のshiftキー

あらゆるキーに対応したマスク

KeyMask::Any	`0xffffffffffffffffULL`