# ArteOGL.framework

# reference

TSB program system

# 目次

# 1. 概要

## 1.1. このドキュメントについて

このドキュメントは、ArteOGL.framwork（以下、ArteOGLと呼称）について記述されるものです。
別冊のチュートリアルと合わせてお読みください。

以降の記述については特筆のない場合、このドキュメントに対応するリビジョンのArteOGLについて記述されてます。具体的な値については文末の奥付を参照ください。

## 1.2. 想定するユーザ

このドキュメント及びArteOGLは、ある程度の技術力と意欲を持っているけれど、実際なかなか難しくて、みたいな人をユーザとして想定しています。もちろんそれ以外の方でもウエルカムです。
十分に技術がある人は是非ご自分でお作りください。

## 1.3. ArteOGL.framwork

ArteOGLは、MacOS X向けのゲーム開発に便利なフレームワークです。
OpenGL poweredで、Cocoa-likeな、
一般的にはスプライトシステムと呼ばれるように思います。
当然、それに留まる気はありませんが。

## 1.4. project Artesema / Artesema OGL

project ArtesemaとはTSBpsのゲーム開発プロジェクトであり
Artesema OGLはその成果物です。
ArteOGL.frameworkはArtesema OGLの基礎部分を抽出したものです。

## 1.5. OpenGL

OpenGLは、3Dグラフィックスの為のインターフェイス。砕いて言えば、GPUに計算をさせる為の規格化されたAPIです。これを使ってやると、処理のほとんどをGPUで行う事が出来ます。ただし、ハードウェアに近い設計なのでそれなりに扱いにくいです。
ArteOGLはOpenGLの命令をラップして使い易くしたものになります。

OpenGLをご存知の方は混ぜ込んで使うとさらに効果的かと思います。

## 1.6. ライセンス条項

ArteOGLは以下の用途に限ってその利用を許可されています。

- 非営利目的（フリーウェアを含む）
- 学術目的
- 同人即売会での販売目的

これは予告無く変更される事があります。
詳しくは最新のライセンス条項をお読みください。

http://tsbps.com/document/arte/fw/

## 1.7. 免責

ArteOGLは基本的に、コードが汚いでしょうし、メモリも漏れているかもしれません。突貫工事なので、あまり気にしないであげてください。

もしくは、ここ変じゃね？と突っ込んでください。

お決まり文句になりますが、このフレームワークにより不都合、損害は発生してもこちらは責任を持てません。
各自の任意と責任でご利用ください。

# 2. ArteOGL.framework

この章ではArteOGL.frameworkの内部構造について記していきます。

## 2.1. 基本構造

ArteOGLは、表示の核となるArteOGLViewを中心に、それを取り巻くクラス群により構成されています。

利用者は、nib/xibにArteOGLViewを一つとArteOGLController（のサブクラス）を一つ配置、Outletで結び使う事を前提にしています。

## 2.2. 座標系

ArteOGLは、内部に2つの座標系をもちます。
スクリーン座標系と、仮想座標系です。

- スクリーン座標系（ArteSystemCoord）
  画面の大きさと等しい座標系。
  texpointで保持される
-
- 仮想座標系（ArteVirtualCoord）
  仮想座標軸。
  texvirtualで保持される

この2つの座標系は仮想軸原点を元に相互変換されて運用されます。
また、全てのArteOGLObjectはこの２つの座標系のいずれかに属します。
よくわからないうちはスクリーン座標系だけ用いていてください。

（追記予定地）

初期化処理
毎回の処理
オブジェクトの誕生、死、ライフサイクル
イベント処理

# 3. クラス

この章では、クラスのリファレンスを記述します。

## 3.1. ArteOGLView

| superclass | NSOpenGLView |
|---|---|
| protocol | ArteOGLViewProtocol |

ArteOGLViewはArteOGL.frameworkの基幹となるビュークラスです。
このビューにスプライトオブジェクトをセットする事で画面表示を行います。
通常、nib/xibにてインスタンス化と配置を行います。

### 3.1.1.screen manage

ArteOGLViewは1枚のOpenGLContextをスクリーンとして持ちます。
またそのフルスクリーン表示をサポートします。

```
- (NSSize)screenSize;
```

スクリーンの大きさを得る。

```
- (void)swichScreenMode;
- (BOOL)isFullScreenMode;
- (void)startFullScreen;
- (void)endFullScreen;
```

画面のウインドウ/フルスクリーン切替を行う。

### 3.1.2. object manage

ArteOGLViewは表示するオブジェクト（スプライト）をリストで保持します。

```
- (BOOL)hasObject:(id)obj;
- (BOOL)setObject:(id)obj;
- (BOOL)removeObject:(id)obj;
- (void)removeAllObjects;
```

描画オブジェクトをリストに設定、削除します。
設定されたオブジェクトはキューに保持され、次の処理時に追加されます。

```
- (void)killObject:(id)obj;
- (void)killThemAll;
```

特定のオブジェクトや全てのオブジェクトを殺します。
特定のオブジェクトについては[obj kill];した方が速いです。

### 3.1.3. virtual coord

ArteOGLVewはスクリーン座標軸と同時に仮想座標軸を持ちます。
仮想軸（vx,vy）は左下原点。
この仮想軸はOpenGLにより提供されるものではなく、ArteOGLの実装によるものです。仮想軸座標は描画前に実軸座標へ変換されます。

```
- (NSPoint)virtualDatum;
- (void)resetVirtualCoordOrigin;
```

仮想軸原点は実軸座標上に配置され、その交差座標はvirtualDatumで表現されます。
仮想軸の初期位置は(0,0)です。
virtualDatum値の変更は座標変換に影響を与えるため、ユーザ側から行うべきではありません。後述のスクロールを用いてください。

```
- (float)xyAtVxy:(float) vx;
- (float)xAtVx:(float) vx;
- (float)yAtVy:(float) vy;
- (float)vxyAtXY:(float) vx;
- (float)vxAtX:(float) x;
- (float)vyAtY:(float) y;
- (void)resetVirtualCoordOrigin;
```

必要に応じて相互変換を行えます。

### 3.1.4. scrolling

ArteOGLは仮想軸を動かす事により画面のスクロールを実現します。

```
- (void)resetScrollMode;
- (int)horizontalScrollMode;
- (int)verticalScrollMode;
- (void)setHorizontalScrollMode:(int)mode;
- (void)setVerticalScrollMode:(int)mode;
```

スクロールの種類については5. 定数をお読みください。

初期値は`ArteOGLNonScrollMode`です。

```
- (NSPoint)scrollSpeed;
- (void)setScrollSpeed:(NSPoint)speeds;
```

スクロールモードが`ArteOGLForceScrollMode`時の速度を設定します。

スクロール速度の単位はpixel per frameになります。

```
- (void)setScrollFocusToObject:(id)obj;
```

フォーカススクロールモードの対象を設定します。

### 3.1.5.pause

```
- (void)togglePause;
- (BOOL)pause;
- (void)setPause:(BOOL)bl;
```

ポーズを管理します。

ポーズモード中は描画が停止され、入力が制限されます。

### 3.1.6.stage

```
- (void)setStageBoundToScreenSize;
- (void)setStageBounds:(NSRect)rect;
- (void)setStageBoundsLeft:(float)f;
- (void)setStageBoundsRight:(float)f;
- (void)setStageBoundsBottom:(float)f;
- (void)setStageBoundsTop:(float)f;
- (void)clearStageBounds;
```

ステージの大きさを設定します。
ステージは仮想軸上に展開され、仮想軸上の利用範囲を示します。

これは表示範囲を大きく逸脱したオブジェクトの回収などの為に存在します。

### 3.1.7.collision

衝突演算の計算補助用のメソッド群です。

```
- (id)nearestCollisionObject:(id)obj;
- (NSArray *)collisionsAtRect:(NSRect)rect;
- (NSArray *)objectsAtRect:(NSRect)rect;
- (NSMutableIndexSet *)objectIndexesAtRect:(NSRect)rect;
```

### 3.1.8. drawing support

画面描画の補助メソッドです。
ArteOGLObjectのサブクラスを実装する場合に有用です。

```
- (void)fillRect:(NSRect)rect;
```

赤色の線で指定領域を縁取ります。
デバッグなどに便利です。

```
- (void)frameRect:(NSRect)rect;
```
現在の指定色で指定領域を塗り潰します。

```
- (void)setBlendMode:(ArteOGLViewBlendMode)mode;
- (void)revertBlendMode;
```
描画の合成モードを切り替えます。
切替を行った後はrevertで戻してください。

```
- (long)framecount;
```
描画の総フレーム数です。
描画回数ごとに挙動を換えたい場合などに参照します。

## 3.2. ArteOGLObject

| superclass | NSObject |
|---|---|

ArteOGLObjectはArteOGLシステムの基礎となるオブジェクトです。
ArteOGLViewにセットされて動作する為の挙動が実装されています。

### 3.2.1. Coordinates

座標処理系のメソッドに着いて述べます。

```
- (void)setCoordMode:(int)mode;
- (int)coordMode;
```
座標モードを設定・呼び出します。

普通は動的に変更する事は無いです。

```
- (void)setX:(float)x y:(float)y z:(float)z;
- (void)setXY:(NSPoint)p;
- (void)setScreenXY:(NSPoint)sp;
- (void)setVirtualXY:(NSPoint)vp;
- (void)setVirtualX:(float)vx;
- (void)setVirtualY:(float)vy;
- (void)setX:(float)x;
- (void)setY:(float)y;
- (void)setZ:(float)z;
- (void)addXY:(NSPoint)p;
- (void)addX:(float)x;
- (void)addY:(float)y;
- (void)addZ:(float)z;
- (NSPoint)xy;
- (NSPoint)vxy;
- (float)x;
- (float)y;
- (float)vx;
- (float)vy;
- (float)z;
```

座標を設定します。

### 3.2.2. Size

```
- (void)setSize:(NSSize)s;
- (void)setWidth:(float)w;
- (void)setHeight:(float)h;
- (NSSize)size;
- (float)width;
- (float)height;
```

テクスチャの大きさを所得、設定します。

### 3.2.3. fraction

```
- (void)setFraction:(float)f;
- (float)fraction;
```

透過度をを所得、設定します。

### 3.2.4. Blend mode

```
- (void)setBlendMode:(ArteOGLViewBlendMode)mode;
- (float)blendMode;
```

合成モードを所得、設定します。

### 3.2.5. Speed

速度は毎回座標に加算されます。単位はpixel/frameです。

```
- (void)setSpeed:(NSPoint)p;
- (void)setSpeedX:(float)x;
- (void)setSpeedY:(float)y;
- (void)addSpeed:(NSPoint)p;
- (void)addSpeedX:(float)x;
- (void)addSpeedY:(float)y;
- (NSPoint)speed;
```

速度を所得、設定します。

### 3.2.6. Accela

加速度は毎回速度に加算されます。単位はpixel/frame$^2$です。

```
- (void)setAccela:(NSPoint)p;
- (void)setAccelaX:(float)x;
- (void)setAccelaY:(float)y;
- (void)addAccela:(NSPoint)p;
- (void)addAccelaX:(float)x;
- (void)addAccelaY:(float)y;
- (NSPoint)accela;
```

加速度を所得、設定します。

### 3.2.7. Force

```
- (void)setForce:(NSPoint)p;
- (void)pushToX:(float)x;
- (void)pushToY:(float)y;
- (NSPoint)force;
```

力の大きさを所得、設定します。

### 3.2.8. set Property

```
- (void)setProperty:(NSDictionary *)dic;
    - (void)setBasicProperty:(NSDictionary *)dic;
    - (void)setAdvancedProperty:(NSDictionary *)dic;
    - (void)setOLEProperty:(NSDictionary *)dic;
```

プロパティを設定します。
主に定義ファイルからオブジェクトを作成する場合に使います。

### 3.2.9. Class/Team/Life

```
- (void)setArteTeam:(int)t;
- (int)arteTeam;
```

チームを所得、設定します。
これは当たり判定処理に使います。

```
- (void)setLifecount:(int)t;
- (int)lifecount;
```

ライフカウントを所得、設定します。
ライフカウントは、その名の通りに各々のオブジェクトに設定された、各々の寿命です。単位はフレームです。毎回、計算/描画処理でデクリメントされ、0になった時点でそのオブジェクトは「死亡」（[self die];）します。

「死亡」後は「走馬灯」状態を経た後、再び「死亡」し、「完全死」します。詳しくは2章をどうぞ。

### 3.2.10. Flag/HitCheck/Physics

各種フラグを所得します。
サブクラスの初期化時に設定してください。

```
- (BOOL)attackFlag;
- (NSRect)attackRect;
- (BOOL)targetFlag;
- (NSRect)existRect;
```

当たり判定のフラグです。
attackFlag: 攻撃を行うか否か
attackRect: 攻撃範囲（スクリーン座標）
targetFlag: 攻撃を受けるか否か
existRect: 存在する（攻撃を受ける）範囲（スクリーン座標）
＊ existRectは物理演算でも使います。

```
- (BOOL)livingFlag;
- (BOOL)physicalFlag;
- (BOOL)staticFlag;
- (NSRect)movingRange;
- (int)score;
```

生存判定、物理演算判定、スタティック判定、物理演算補助、スコア用の値。

### 3.2.11. Calc/Draw

```
- (void)OGLconvert:(id <ArteOGLViewProtocol>)arteview;
```

仮想軸座標と実軸座標の変換、もしくは仮想座標の実軸適用を行います。
毎フレーム、OGLcalcの実行前に呼ばれます。
オーバーライドする場合はスーパークラスを呼んでください。

```
- (void)OGLcalc:(id <ArteOGLViewProtocol>)arteview;
```
移動他の計算を行います。
オーバーライドする場合はスーパークラスを呼んでください。

```
- (void)OGLcalcOLE:(id <ArteOGLViewProtocol>)arteview;
```
オブジェクトレベルエフェクトの計算を行います。
オーバーライドする場合はスーパークラスを呼んでください。

```
- (void)OGLdraw:(id <ArteOGLViewProtocol>)arteview;
```
描画を行います。
オーバーライドする場合はスーパークラスを呼んでください。

```
- (void)OGLhitBy:(ArteOGLObject *)obj;
```
当たり判定のメソッドです。
他のオブジェクトから攻撃を受けた場合に呼ばれます。

```
- (NSPoint)OGLpush:(NSPoint)pushwidth;
```
物理演算のメソッドです。
他のオブジェクトから押された場合に呼ばれます。

### 3.2.12. action

```
- (void)die;
```
死にます。主に自分が呼びます。

```
- (void)kill;
```
殺されます。主に他のオブジェクトから呼ばれます。

### 3.2.13. Description

```
//- (NSString *)description;
- (NSString *)descriptionMessage;
- (NSString *)shortDescription;
```

デスクリプションメッセージを管理します。
デバッグ用に座標、速度、加速度、などの情報を表示します。

サブクラスは、特に記述する事が無い場合：

descriptionMessageのみを実装します

追加情報を加えたい場合：

```
return [[super description] stringByAppendingString:@""];
```

とするのが良いでしょう。
shortDescriptionは1行に収まる短縮表記です。
本来の用途は別にあるのですが、ログるときにも便利かもしれません。

### 3.2.14. OLE

オブジェクトが各自に演出を行います。
未整理です。

```
- (void)setOLEShakeWithX:(float)wid interval:(int)frame;
- (void)setOLEShakeWithY:(float)wid interval:(int)frame;
- (void)setOLEShakeWithFraction:(float)wid interval:(int)frame;
- (void)setOLESickly:(float)wid halfInterval:(int)hc;
- (void)setOLEZoom:(float)scale interval:(int)frame;
```

## 3.3. ArteOGLRectangle

| superclass | ArteOGLObject |
|---|---|

四角形のオブジェクトです。

```
+ (id)objectWithRectFill:(NSRect)rect;
+ (id)objectWithRectFrame:(NSRect)rect;
- (id)initWithRect:(NSRect)rect;
```

オブジェクトを生成します。

```
- (void)setColor:(NSColor *)col;
```

線色/塗り潰し色を設定します。

```
- (void)setFillModeWithName:(NSString *)modename;
- (void)setFillMode:(ArteOGLRectangleMode)mode;
- (int)fillMode;
```

塗り潰しモードを設定します。

## 3.4. ArteOGLImage

| superclass | ArteOGLObject |
|---|---|

画像のオブジェクトです。
画像オブジェクトは座標、大きさを保持する実オブジェクトと、実画像を表現するテクスチャの2つから構成されます。

テクスチャをオブジェクトから切り離してキャッシュする事で、便利です。

### 3.4.1. Texture Caching Manage

テクスチャのキャッシュ管理を行います。

```
+ (BOOL)cacheTextureNamed:(NSString *)name;
```

テクスチャをキャッシュします。
name: 画像のファイル名
return: キャッシュの成功

```
+ (ArteOGLImageTexture *)sharedTextureNamed:(NSString *)name;
```

キャッシュ済みのテクスチャを返します。

```
+ (void)clearSharedTextures;
```

キャッシュをクリアします。

### 3.4.2. initialize (with texture caching )

```
+ (id)objectWithImageNamed:(NSString *)name;
- (id)initWithImageNamed:(NSString *)name;
```

キャッシュされたテクスチャを用いて画像オブジェクトを生成します。

### 3.4.3. initialize (without caching)

```
- (id)initWithNSImage:(NSImage *)img;
- (id)initWithImageTexture:(ArteOGLImageTexture *)texobj;
```

画像オブジェクトを生成します。
キャッシングは為されません。

### 3.4.4. texture manage

```
- (void)switchTexture:(ArteOGLImageTexture *)texobj
               resize:(BOOL)resize;
```

テクスチャを入れ替えます。
resize: YESならオブジェクトのサイズをテクスチャサイズに合わせます。

### 3.4.5. values

```
- (NSSize) imageSize;
- (NSSize) textureSize;
- (ArteOGLImageTexture *) texture;
```

## 3.5. ArteOGLImageTexture

| superclass | NSObject |
|---|---|

画像テクスチャを表すクラスです。

### 3.5.1. initialize

```
- (id)initWithImageNamed:(NSString *)name
          withExtension:(NSString *)ext;
- (id)initWithNSImage:(NSImage *)img;
- (id)initWithBitmapImageRep:(NSBitmapImageRep *)rep;
```

テクスチャオブジェクトを生成します。
初期化時にテクスチャ作成も行われます。

### 3.5.2. bmprep

ArteOGLImageTextureはBitmapRepからOpenGLテクスチャを生成します。これらのメソッドはBitmapRepを設定します。

```
- (BOOL)setImageNamed:(NSString *)name
        withExtension:(NSString *)ext;
- (BOOL)setNSImage:(NSImage *)img;
- (BOOL)setBitmapImageRep:(NSBitmapImageRep *)rep;
```

通常、ユーザがこれらのメソッドを呼び出す必要はありません。

### 3.5.3. texturize

OpenGLテクスチャを生成・破棄します。

```
- (BOOL)texturize;
- (BOOL)createTexture;
- (void)destructTexture;
- (BOOL)isTextureResident;
```

通常、ユーザがこれらのメソッドを呼び出す必要はありません。

### 3.5.4. draw

描画を行います。

```
- (void)OGLdrawAtPoint:(NSPoint)pt;
- (void)OGLdrawInRect:(NSRect)rect fraction:(float)alpha;
```

これらのテクスチャの描画命令は、ユーザーオブジェクトが画像描画を行う時などに用いられます。

### 3.5.5.values

```
- (NSSize)        imgsize;
- (NSString *)    imgname;
- (NSSize)        texsize;
- (GLuint)        texid;
```

imgsizeは読み込む前の画像の大きさ、
texsizeは読み込んだ後のテクスチャの大きさになります。

## 3.6. ArteOGLImageTexture2N

| superclass | ArteOGLImageTexture |
|---|---|

末尾の『2N』は『$2^n$』を示し、2の累乗の大きさ（2, 4, 16, 32, 64, 128, 256, 512, 1024, 2048, ... ）の画像である事を意味します。
内部的にArteOGLImageTextureが`GL_TEXTURE_RECTANGLE_EXT`を用いるのに対し、ArteOGLImageTexture2Nは`GL_TEXTURE_2D`を用います。

この違いにより
・読み込む画像の大きさが限定される
代わりに
・パターン繰り返し
・おそらくに高速
を可能にします。

Artesema OGLでは背景画像の描画などに使っています。

#### 3.6.1. draw

```
- (void)OGLdrawAtPoint:(NSPoint)pt;
- (void)OGLdrawInRect:(NSRect)rect fraction:(float)alpha;
```

継承しています。

```
- (void)OGLdrawInRect:(NSRect)rect
                shift:(NSPoint)shift
             fraction:(float)alpha;
```

繰り返し有効で指定範囲内を描きます。
パターン塗り潰し的な結果が得られます。
shift: xy方向それぞれのずらし分

### 3.7. ArteOGLString

| superclass | ArteOGLObject |
|---|---|

文字のオブジェクトです。
画像と同じく、実オブジェクトとテクスチャに分割されています。

文字は画像と違って同じ文字を使い回す事が少なく、キャッシュの必要があまりないのですが、そこはユーザーの努力まかせに積極的にキャッシュするような方向になっています。

キャッシュは文章単位になります。

### 3.7.1. texture cache manage

```
+ (BOOL)cacheTextureWithAttributedString:
                        (NSAttributedString *)as
                 forName:(NSString *)name;
+ (BOOL)cacheTextureWithSystemFontString:(NSString *)str
              grayscale:(float)gs
                   size:(int)fs
                 forName:(NSString *)name;
+ (BOOL)cacheTextureWithString:(NSString *)str
              grayscale:(float)gs
               fontname:(NSString *)fn
               fontsize:(int)fs
                forName:(NSString *)name;
```

予めにキャッシュします。

```
+ (id)cachedTextureWithAttributedString:
                        (NSAttributedString *)as
                 forName:(NSString *)name;
+ (id)cachedTextureWithSystemFontString:(NSString *)str
                 forName:(NSString *)name;
+ (id)cachedTextureWithBlackSystemFontString:(NSString *)str
                 forName:(NSString *)name;
+ (id)cachedTextureWithWhiteSystemFontString:(NSString *)str
                 forName:(NSString *)name;
+ (id)cachedTextureWithString:(NSString *)str
              grayscale:(float)gs
               fontname:(NSString *)fn
               fontsize:(int)fs
                forName:(NSString *)name;
```

キャッシュされたテクスチャを得ます。

ラベルの表示などにはこれを用いるべきかと思います。

```
+ (ArteOGLStringTexture *)sharedTextureNamed:(NSString *)name;
+ (void)uncacheTextureForName:(NSString *)name;
+ (void)clearSharedTextures;
```

キャッシュ済みのテクスチャ管理します
文字のキャッシュは画像とは別個に行われます。

### 3.7.2. initialize

```
+ (id)objectWithAttributedString:(NSAttributedString *)name;
+ (id)objectWithString:(NSString *)str
                attributes:(NSDictionary *)dic;
+ (id)objectWithString:(NSString *)str
                 grayscale:(float)gs
                  fontsize:(int)fs;
```

```
- (id)initWithAttributedString:(NSAttributedString *)str;
- (id)initWithString:(NSString *)str
                attributes:(NSDictionary *)att;
- (id)initWithProperty:(NSDictionary *)prop;
```

初期化します。キャッシュは行われません。

### 3.7.3. texture manage

```
- (void)switchTexture:(id)texobj;
- (void)changeString:(NSString *)str
                attributes:(NSDictionary *)dic;
```

テクスチャを置き換え、あるいは新しい文字からのテクスチャで置き換えます。

## 3.8. ArteOGLStringTexture

| superclass | ArteOGLImageTexture |
|---|---|

文字のテクスチャは、文章単位で画像に起こしてからOpenGLテクスチャ化されます。

### 3.8.1. initialize

```
+ (id)textureWithAttributedString:(NSAttributedString *)aS;
+ (id)textureWithString:(NSString *)str
             attributes:(NSDictionary *)dic;
```

テクスチャを置き換え、あるいは新しい文字からのテクスチャで置き換えます。

### 3.8.2. draw

```
- (void)OGLdrawAtPoint:(NSPoint)pt
                     z:(float)texzdepth;
- (void)OGLdrawAtRightBottomPoint:(NSPoint)pt
                     z:(float)texzdepth;
```

ArteOGLは左下原点ですので、指定点の右上に描画されます。
〜RightBottomPointでは、指定点の左上に描画されます。

### 3.8.3. values

```
- (NSSize)              framesize;
//- (NSSize)            texsize;
//- (GLuint)            texid;
- (NSString *)          string;
- (NSDictionary *)      attributes;
```

texsize,texidは継承されています。

## 3.9. ArteOGLController

ArteOGLControllerはArteOGLViewのdelegateとして動作するコントローラクラスです。

ユーザーはこれのサブクラスを作成し、必要に応じてメソッドをオーバライドする事でArteOGLを利用したアプリケーションを効率的に作成出来ます。

### 3.9.1. ArteOGLViewDelegateProtocol, regist

```
- (void)ArteOGL:(id <ArteOGLViewProtocol>)view
        registerAsDelegate:(id)controller;
```

ArteOGLViewのawakeFromNib:で発行されます。
ArteOGLControllerは、第一引数を自らの管理するビューとしてインスタンス変数arteviewに保持します。
オーバーライドする場合はスーパークラスを呼ぶのを忘れないでください。

### 3.9.2. ArteOGLViewDelegateProtocol, event

キー/マウスイベントを提供します。
ここに提供される以上のイベントが必要な場合はArteOGLViewのサブクラスを作成してください。

```
- (void)ArteOGL:(id <ArteOGLViewProtocol>)view
           keyPressed:(unichar)key;
- (void)ArteOGL:(id <ArteOGLViewProtocol>)view
          keyReleased:(unichar)key;
```

キーが押された/放された場合に呼ばれます。
第二引数は押されたキーです。

```
- (void)ArteOGL:(id <ArteOGLViewProtocol>)view
          mousePressed:(NSPoint)pt;
- (void)ArteOGL:(id <ArteOGLViewProtocol>)view
         mouseReleased:(NSPoint)pt;
```

マウスボタンが押された/放された場合に呼ばれます。

第二引数は実軸座標になります。仮想軸座標が必要な場合は
[view vxyAtXY:pt];で所得してください。

### 3.9.3. ArteOGLViewDelegateProtocol, interrupt

処理途中で呼ばれます。
ArteOGLViewの計算・描画処理の流れを把握してから使ってください。

```
- (void)ArteOGLdidCalculate:(id <ArteOGLViewProtocol>)view;
```

計算が終わった時点で呼ばれます。

```
- (void)ArteOGLdidDraw:(id <ArteOGLViewProtocol>)view;
```

描画が終わった時点で呼ばれます。

### 3.9.4. ArteOGLViewDelegateProtocol, delegate

処理を委譲します。

```
- (void)ArteOGL:(id <ArteOGLViewProtocol>)view
              objectDied:(id)obj;
```

死亡したオブジェクトについての処置を求めます。
スコア計算などを行う場合はこの時点で行うのが良いと思います。
デフォルトの挙動では何もしません。

```
- (void)ArteOGL:(ArteOGLView *)view
           objectOvered:(id)obj;
```

設定されたステージ領域を逸脱しているオブジェクトについての処置を求めます。
デフォルトの挙動は何もしません。が、それだと負荷がかかるので、何かしらに設定してください。

この委譲はオブジェクトがステージを逸脱している限りに何度でも呼び出されます。killする、移動する、などを施して逸脱状況を解除してください。もしくは、逸脱しないようにステージを変更してください。

```
- (void)ArteOGLdrawPause:(id <ArteOGLViewProtocol>)view;
```

ポーズ画面の描画を求めます。
これはポーズへ移行した直後に一度だけ呼ばれます。

### 3.9.5. ArteOGLViewDelegateProtocol, behavior

ArteOGLViewの挙動について問われます。
これらの多くは情報が必要になった初回のみ呼び出されます。

> これらは初期値として利用されます。動的に変化させたい場合は対応するメソッドを自分で呼び出してください。

```
- (uint)ArteOGLfps:(id <ArteOGLViewProtocol>)view;
```

フレームレートを指定します。
単位はFrame per Secondsです。
30FPSか60FPSがデファクトスタンダートらしいです。

```
- (BOOL)ArteOGLkeepAspectRatioWhenFullScreen:
                        (id <ArteOGLViewProtocol>)view;
```

フルスクリーン時に画面のアスペクト比（縦横比）を保持するかどうかを返します。
YESを返すと、縦横比が保たれるように左右もしくは上下に黒の帯が入ります。
この実装は正方形ピクセル環境のみを想定しています。

```
- (BOOL)ArteOGLhandlePauseKeyEventInView:
                        (id <ArteOGLViewProtocol>)view;
```

ポーズのキーイベントをArteOGLView側で処理するかどうかを返します。

YESを返した場合、pキーがポーズキーに割り当てられ、ポーズ開始/解除が自動で行われるようになります。
ただし、その場合は
. ポーズキーがpキーに固定される
. ポーズ中のキーイベントが渡ってこない
になりますので、キー割り当てを自由に行いたい場合、ポーズ中にメニューを操作を行いたい場合などはNOを返し、ポーズ処理を自前で実装してください。

```
- (BOOL)ArteOGLhandleFullScreenKeyEventInView:
                        (id <ArteOGLViewProtocol>)view;
```

スクリーン切替のキーイベントをArteOGLView側で処理するかどうかを返します。
YESを返した場合、^キーがフルスクリーン開始、-キー/ESCキーがフルスクリーン解除に割り当てられ、フルスクリーンの開始/解除が自動で行われるようになります。
ただし、ポーズキーと同じく、
. スクリーン切替のキーが固定される
ので、自由に設定したい場合は自前で実装してください。

## 3.10. ArteMisya

| superclass | ArteOGLObject |
|---|---|

project Shiaに内包されるMisya/SceenObject技術のArte移植版です。
Misyaは動的コンテンツを含むMake-up Languageの表示／実行環境であり、ソフトコーディングでインターフェイスを作成できます。

整備中です。
詳しくは、後日。

# 4. 定数

以下に定数を示します。

## 4.1. ArteOGLObjectCoordMode

オブジェクトの座標軸モードを示します。

```
ArteOGLObjectSystemCoordMode
```

実軸（スクリーン座標軸）モードです。

( x, y )

```
ArteOGLObjectVirtualHorizontalCoordMode
```

横軸のみ仮想軸のモードです。

( vx, y )

```
ArteOGLObjectVirtualVerticalCoordMode
```

縦軸のみ仮想軸のモードです。

( x, vy )

```
ArteOGLObjectVirtualCoordMode
```

仮想軸のモードです。

( vx, vy )

## 4.2. ArteOGLScrollMode

スクロールのモードを示します。

```
ArteOGLNonScrollMode
```

スクロールをしません。

```
ArteOGLForceScrollMode
```

強制スクロールを行います。
スクロール速度を別途指定の事。

```
ArteOGLFocusScrollMode
```

特定オブジェクトにフォーカスしてスクロールを行います。

```
ArteOGLRandomScrollMode
```

一定時間毎にスクロールモードを変更します。
未実装です。

## 4.3. ArteOGLObjectTeam

オブジェクトの所属を示します。

```
ArteNonTeam
```

どこにも所属していない事を示します。

```
ArteAlphaTeam
ArteBetaTeam
ArteGammaTeam
ArteDeltaTeam
ArteEpsilonTeam
```

何れかに所属している事を示します。

## 4.4. Arte ObjectLevelEffect

未整頓に付き、詳細割愛。

## 4.5. ArteSpeedLimit

速度制限。

```
extern float ArteOGLSpeedLimitX;
extern float ArteOGLSpeedLimitY;
```

動的に変更出来るよう、定数でなく大域変数になっています。
初期値はいずれも20。単位はpixel per frame。

あまりぶっ飛んで欲しくない場合に設定してください。
＊rev.1では値が15の定数でした。

## 4.6. ArteLifeCount

lifecountの特別な値。

```
ArteOGLObjectEternalLife
```

無限の命。
時間経過では死にません。

```
ArteOGLObjectHereafterLife
```

死後の世界。
完全に死んだ後に設定されます。

## 4.7. ArteOGLBlendMode

合成モードです。
オブジェクトに設定するか、描画途中に設定してやると合成モードを変更出来ます。
詳しくは使ってみてから考えてください。

```
ArteOGLViewSourceOverBlendMode
```

ソースオーバー。デフォルト値です。
これまでの描画結果の上に重ね描きします。

```
ArteOGLViewCopyBlendMode
```

コピー。
透過度を無視して重ね描きします。その分に速いです。

```
ArteOGLViewAddBlendMode
```

加算合成。
加法混色に従い、加算合成。
黄色＋青＝緑

```
ArteOGLViewMulBlendMode
```

乗算合成。
色セロファン的な効果。

```
ArteOGLViewInvertBlendMode
```

反転合成。
ネガポジ反転。

```
ArteOGLViewScreenBlendMode
```

スクリーン合成。
加算っぽいけど、飽和しにくい。

```
ArteOGLViewXORBlendMode
```

XOR合成。

```
ArteOGLViewEraseBlendMode
```

消去。

## 4.8. ArteOGLRectangleMode

ArteOGLRectangle の塗り潰しモードです。

```
ArteOGLFrameRectangle
```

領域を縁取る枠のみを描きます。

```
ArteOGLFillRectangle
```

領域内部を塗り潰します。

```
ArteOGLDottedRectangle
```

未実装です。

```
ArteOGLEraseRectangle
```

領域内を消去します。

```
ArteOGLHighLightRectangle
```

未実装です。

```
ArteOGLInvertRectangle
```

領域内を反転します。

# 5. ArteMisya

## 5.1. ArteMisyaとは

ArteMisyaは「汎用アドベンチャーゲーム実行環境 Shia」の一部を
ArteOGL上に実装したものです。
ScreenObject形式の画面表示を可能にします。
詳しくは、Shiaのドキュメントをどうぞ。
http://tsbps.com/document/shia/v0.5/

## 5.2. ScreenObjectのメリット

GUI設計が圧倒的に楽
コーディング量、バグ量が減る

## 5.3. ArteShiaBridgeによる連携

ArteMisyaは、ArteShiaBridgeを介してArteOGLを認識します。

Misyaには元来ShiaMessageという内部通信手段が存在します。
ArteShiaBridgeはそれに接続し、ArteOGLとの橋渡しを行います。

Misyaからは、ArteOGL上の機能は「arte」の接頭詞で始まる見なされ
自由に呼び出す事が出来ます。またArteOGL側からもShiaMessageを発
行する事でMisyaへアクセスする事が出来ます。

## 5.4. Misya Delegateによる連携

ArteShiaBridge以外の選択として、Misya delegateがあります。
これはArteMisyaのスクリプトエンジンの処理の一部を委譲する事で
スクリプトコマンドを追加し、これによる連携を可能にする物です。

# 6. あとがき

以上がArteOGLの概要になります。

## 6.1. これまでのドキュメントについて

間に合わなくて申し訳ないです。

## 6.2. このrevisionのドキュメントについて

だいぶ遅れてしまいました＋中途半端＋rev.2アップデートを伴わないなどと問題が山積みなリリースになってしまいましたが
これ以上待たせるのも申し訳ないのと、何も無いよりは何かあった方がと思い、2章やインスタンス変数の情報など、色々に大事な部分が抜け落ちていますが取り急ぎにリリースをします。

## 6.3. 今後の展開

rev.3では、色々の整理の他にArteMisya周りとOLE周りが更新される予定です。

# 7. 奥付

| ArteOGL.framework | |
|---|---|
| 著者 | とうた |
| イラスト | - |
| 発行者 | TSB program system |
| 版 | revision 2 |
| 発行日 | 2009.1.10 |