DiploidEngineApp

```
void FileCreate(DiploidEngineApp* app);
void SoundLoad(DiploidEngineApp* app);//一回だけ音を読み込む
void GraphicsLoad(DiploidEngineApp* app);//一回だけ画像を読み込む(解像度を変更した際にも呼ばれます。)
void SoundInit(DiploidEngineApp* app);//一回だけ音を初期化
void GraphicsInit(DiploidEngineApp* app);//一回だけ画像を初期化(解像度を変更した際にも呼ばれます。再読み込み時に値を変更したくない場合はUpdata()関数で処理をすること。)
void Updata(DiploidEngineApp* app);//ループ処理(fps依存)
void Draw(DiploidEngineApp* app);//描画
void Destory(DiploidEngineApp* app);//ver1.0の機能でのオブジェクト削除
void End(DiploidEngineApp* app);//ゲーム終了時の処理
```

DiploidEngineInput

```
void Init();
void Update();
void Draw(int x, int y, bool debug = false);
bool GetKey(int DXLIB_KEY_CODE);//キーを押したか押していないかの判定。(押していたらtrueが返る)
bool GetPressKey(int DXLIB_KEY_CODE);//キーを押した瞬間の判定。(押し瞬間にtrueが返る)
bool GetReleaseKey(int DXLIB_KEY_CODE);//キーを離された瞬間の判定。(離した瞬間にtrueが返る)
bool GetMouse(int DXLIB_MOUSE_CODE);//マウスを押したか押していないかの判定。(押していたらtrueが返る)
bool GetPressMouse(int DXLIB_MOUSE_CODE);//マウスを押した瞬間の判定。(押した瞬間にtrueが返る)
bool GetReleaseMouse(int DXLIB_MOUSE_CODE);//マウスを離した瞬間の判定。(離した瞬間にtrueが返る)
int GetWhellVolume();//マウスホイールの回転数を瞬間的に取得します。
float GetWhellVolume_Scale(float max = 5.0f, float min = 0.2f, float scale = 0.1f);//マウスホイールの回転数を元にスケール値を取得します。(初期値は1.0f、maxにはスケール最大値を指定、minには最小値を指定,scaleには回転数の倍率を指定)
VECTOR GetMousePressPosition();//押した瞬間のマウス座標の取得。
VECTOR GetMouseReleasePosition();//離した瞬間のマウス座標の取得。
VECTOR GetMousePosition();//現在のマウス座標の取得。
```

DiploidEngineMath

```
float VectorAdd(float a, float b);//ベクトル加算(一次式) aからbに加算減算等を行っています。
float VectorSub(float a, float b);//ベクトル減算(一次式) aからbに加算減算等を行っています。
float ThreeSquares(float a, float b);//直角三角形の斜辺の値を取得します。(aとbは直角に交わる。)
float ToDegree(float radian_angle);//ラジアンを度数に変換
float ToRadian(float degree_angle);//度数をラジアンに変換
float EquilateralTriangleHeight(float length);//斜辺の長さから正三角形の高さを計算します。
float EquilateralTriangleLength(float height);//高さから正三角形の斜辺の長さを計算します。
```

DiploidEngineSetting

```
void Init();//DxLib初期化処理。初期化に失敗(-1)した場合はアプリケーションを終了します。
void SetBegin();//DxLib初期化処理より前に設定する項目。
void SetEnd();//DxLib初期化処理後に設定する項目。
void End();//ゲーム終了をするときに呼ぶDxLib終了処理。
void Updata();
void SetExit(int flag);//終了フラグを設定する。
void SetWindowSize(int new_window_x, int new_window_y);
int GetExit();//終了フラグを取得する。
bool GetReloadFlag();//再度画像を読み込むflagを取得する。
void SetReloadFlag(bool new_flag);//再度画像を読み込むflagを変更する。
```

DiploidBoxV2

```
void Init(VECTOR pos, VECTOR size, unsigned int color, float scale = 1.0f, bool fill = FALSE, float thickness = 1.0f);
void MoveUpdate();//設定した移動速度を反映します。
void Draw(bool draw = true);//円を描画します。(drawにfalseを入れることで描画しない)

void SetColor(unsigned int new_color);//新しい色を設定します。
void SetPosition(VECTOR new_pos);//新しい位置を設定します。
void SetSize(VECTOR new_size);//新しい大きさを設定します。
void SetFill(bool new_fill);//新たに塗りつぶしを設定します。
void SetThickness(float new_thickness);//新たに線の太さを設定します。
void SetScale(float new_scale_x, float new_scale_y);//新たにスケール値を設定します。

void SetName(string new_name);//新たにオブジェクトの名前を設定します。
void SetObjectNumber(int new_number);//新しいオブジェクト番号を設定します。
void SetHitFlag(bool new_hit_flag);//新しくヒットフラグを設定します。
void SetMoveSpeed(VECTOR new_move_speed);//新しく移動する速度を設定します。(3軸指定版)
void SetMoveSpeed(float angle, float new_move_speed);//新しく移動する速度を設定します。(ラジアン角指定版)
void SetLife(float new_life);//新しく寿命を設定します。
void SetDestoryFlag(bool new_flag);//新しく削除フラグを設定します。
void SetMainCameraFlag(bool new_flag);//新しくメインカメラフラグを設定します。(Camera関数以外からの使用は控える)

void AddLife(float add_val);//寿命カウントを足します。
void SubLife(float sub_val);//寿命カウントを引きます。

VECTOR GetPosition();//現在の位置を取得します。
VECTOR GetSize();//現在の大きさを取得します。
float GetScaleX();//現在のX方向のスケール値を取得します。
float GetScaleY();//現在のX方向のスケール値を取得します。
float GetThickness();//現在の線の太さを取得します。
unsigned int GetColor();//現在の色を取得します。
bool GetFill();//現在の塗りつぶしの設定を取得します。

string GetName();//現在のオブジェクトの名前を取得します。
int GetObjectNumber();//現在のオブジェクトの番号を取得します。
bool GetHitFlag();//現在のオブジェクトのヒットフラグを取得します。
VECTOR GetMoveSpeed();//現在のオブジェクトの移動速度を取得します。
float GetMoveAngle();//現在のオブジェクトの移動している方角を取得します。(ラジアン)
float GetLife();//現在のオブジェクトの寿命を取得します。
bool GetDestoryFlag();//現在の削除フラグを取得します。(trueで削除対象オン)
bool GetMainCameraFlag();//現在のオブジェクトがメインカメラの対象になっているかのフラグを取得します。
```

DiploidCircleV2

```
void Init(VECTOR pos, float radi, unsigned int color, bool fill = FALSE, float thickness = 1.0f);
void MoveUpdate();//設定した移動速度を反映します。
void Draw(bool draw = true);//円を描画します。(drawにfalseを入れることで描画しない)

void SetColor(unsigned int new_color);//新しい色を設定します。
void SetPosition(VECTOR new_pos);//新しい位置を設定します。
void SetRadius(float new_radius);//新しい半径を設定します。
void SetFill(bool new_fill);//新たに塗りつぶしを設定します。
void SetThickness(float new_thickness);//新たに線の太さを設定します。

void SetName(string new_name);//新たにオブジェクトの名前を設定します。
void SetObjectNumber(int new_number);//新しいオブジェクト番号を設定します。
void SetHitFlag(bool new_hit_flag);//新しくヒットフラグを設定します。
void SetMoveSpeed(VECTOR new_move_speed);//新しく移動する速度を設定します。(3軸指定版)
void SetMoveSpeed(float angle, float new_move_speed);//新しく移動する速度を設定します。(ラジアン角指定版)
void SetLife(float new_life);//新しく寿命を設定します。
void SetDestoryFlag(bool new_flag);//新しく削除フラグを設定します。
void SetMainCameraFlag(bool new_flag);//新しくメインカメラフラグを設定します。(Camera関数以外からの使用は控える)

void AddLife(float add_val);//寿命カウントを足します。
void SubLife(float sub_val);//寿命カウントを引きます。

VECTOR GetPosition();//現在の位置を取得します。
float GetRadius();//現在の半径を取得します。
float GetThickness();//現在の線の太さを取得します。
unsigned int GetColor();//現在の色を取得します。
bool GetFill();//現在の塗りつぶしの設定を取得します。

string GetName();//現在のオブジェクトの名前を取得します。
int GetObjectNumber();//現在のオブジェクトの番号を取得します。
bool GetHitFlag();//現在のオブジェクトのヒットフラグを取得します。
VECTOR GetMoveSpeed();//現在のオブジェクトの移動速度を取得します。
float GetMoveAngle();//現在のオブジェクトの移動している方角を取得します。(ラジアン)
float GetLife();//現在のオブジェクトの寿命を取得します。
bool GetDestoryFlag();//現在の削除フラグを取得します。(trueで削除対象オン)
bool GetMainCameraFlag();//現在のオブジェクトがメインカメラの対象になっているかのフラグを取得します。

size_t GetHitPointsVolume();//現在の当たっている場所の総数を取得します。
VECTOR GetHitPosition();//現在の当たっている場所の位置を取得します。
list<VECTOR>::iterator GetHitPointsListIterator(list<VECTOR>::iterator*
pointer_to_iterator);//hit_points_listのイテレータを取得します。(使用非推奨:イテレータ指定版)
list<VECTOR>::iterator GetHitPointsListIterator();//要検証：hit_points_listのイテレータを取得します。(使用推奨：安全版。list<>::begin()が初期値)
list<VECTOR>* GetHitPointsListPointer();//hit_points_listへのポインタを取得します。
```

DiploidImageV2

```
void Load(const char* path);//画像を読み込みます。
void Init(VECTOR pos, bool shift_flag = false);//読み込んだ画像を初期化します。(shift_flagがtrueで画像を中心にして描画します。)
void MoveUpdate();//設定した移動速度を反映します。
void Draw(bool draw = true);//画像を描画します。
void SetHandl(int new_handl);//画像ハンドルを設定します。
void SetScale(float scale_x, float scale_y);//画像の表示倍率を変更します。
void SetAngle(float new_angle);//新しく回転角を設定します。
void SetPosition(VECTOR new_position);//新しく表示位置を設定します。
void SetTurnFlag(bool new_flag);//新しく画像反転フラグを設定します。
void SetRotatePosition(VECTOR new_position);//新しく回転軸を設定します。
void SetBright(float scale = 1.0f);//明るさを設定します。(1.0fで100%)
void SetBright(int red, int green, int blue);//明るさを指定します。
void SetAlpha(int pal);//透過度を設定します。
void SetActive(bool flag);//アクティブフラグを設定します。
void SetMoveSpeed(VECTOR new_move_speed);//新しく移動する速度を設定します。(3軸指定版)
void SetMoveSpeed(float angle, float new_move_speed);//新しく移動する速度を設定します。(ラジアン角指定版)
void SetName(string new_name);//新たにオブジェクトの名前を設定します。
void SetLife(float new_life);//新しく寿命を設定します。
void SetDestoryFlag(bool new_flag);//新しく削除フラグを設定します。
void SetMainCameraFlag(bool new_flag);//新しくメインカメラフラグを設定します。(Camera関数以外からの使用は控える)
void SetObjectNumber(int new_number);//新しいオブジェクト番号を設定します。
void SetHitFlag(bool new_hit_flag);//新しくヒットフラグを設定します。
void AddLife(float add_val);//寿命カウントを足します。
void SubLife(float sub_val);//寿命カウントを引きます。
int GetGraphicsHandl();//読み込まれた現在のグラフィックハンドルを取得します。
VECTOR GetScale();//現在の画像の表示倍率を取得します。
VECTOR GetSize();//現在の画像の大きさを取得します。
VECTOR GetPosition();//現在の画像の左上の位置を取得します。
VECTOR GetCenterPosition();//現在の画像の中心位置を取得します。
VECTOR GetRotatePosition();//現在の画像の回転軸座標を取得します。
VECTOR GetDownCenterPosition();//現在の画像の左下の座標を取得します。
float GetAngle();//現在の画像の回転角を取得します。(ラジアン)
bool GetTurnFlag();//現在の画像反転フラグを取得します。(TRUEで反転)
int GetAplha();
VECTOR GetMoveSpeed();//現在のオブジェクトの移動速度を取得します。
float GetMoveAngle();//現在のオブジェクトの移動している方角を取得します。(ラジアン)
string GetName();//現在のオブジェクトの名前を取得します。
float GetLife();//現在のオブジェクトの寿命を取得します。
bool GetDestoryFlag();//現在の削除フラグを取得します。(trueで削除対象オン)
bool GetMainCameraFlag();//現在のオブジェクトがメインカメラの対象になっているかのフラグを取得します。
int GetObjectNumber();//現在のオブジェクトの番号を取得します。
bool GetHitFlag();//現在のオブジェクトのヒットフラグを取得します。
bool GetActiveFlag();//現在のオブジェクトのアクティブフラグを取得します。
```

DiploidTriangle(一部関数は未実装)

```
void Init(TRIANGLE pos, unsigned int color, float rotation = 0, bool fill = FALSE, float thickness = 1.0f);
void MoveUpdate();//設定した移動速度を反映します。
void Draw(bool draw = true);//円を描画します。(draw に false を入れることで描画しない)
void SetColor(unsigned int new_color);//新しい色を設定します。
void SetPosition(TRIANGLE new_pos);//新しい位置を設定します。
void SetRadius(float new_radius);//新しい半径を設定します。
void SetFill(bool new_fill);//新たに塗りつぶしを設定します。
void SetThickness(float new_thickness);//新たに線の太さを設定します。
void SetName(string new_name);//新たにオブジェクトの名前を設定します。
void SetObjectNumber(int new_number);//新しいオブジェクト番号を設定します。
void SetHitFlag(bool new_hit_flag);//新しくヒットフラグを設定します。
void SetMoveSpeed(VECTOR new_move_speed);//新しく移動する速度を設定します。(3軸指定版)
void SetMoveSpeed(float angle, float new_move_speed);//新しく移動する速度を設定します。(ラジアン角指定版)
void SetLife(float new_life);//新しく寿命を設定します。
void SetDestoryFlag(bool new_flag);//新しく削除フラグを設定します。
void SetMainCameraFlag(bool new_flag);//新しくメインカメラフラグを設定します。(Camera 関数以外からの使用は控える)
void AddLife(float add_val);//寿命カウントを足します。
void SubLife(float sub_val);//寿命カウントを引きます。
TRIANGLE GetPosition();//現在の位置を取得します。
float GetRadius();//現在の半径を取得します。
float GetThickness();//現在の線の太さを取得します。
unsigned int GetColor();//現在の色を取得します。
bool GetFill();//現在の塗りつぶしの設定を取得します。
VECTOR GetCenterPosition();//現在の重心を得ます。
string GetName();//現在のオブジェクトの名前を取得します。
int GetObjectNumber();//現在のオブジェクトの番号を取得します。
bool GetHitFlag();//現在のオブジェクトのヒットフラグを取得します。
VECTOR GetMoveSpeed();//現在のオブジェクトの移動速度を取得します。
float GetMoveAngle();//現在のオブジェクトの移動している方角を取得します。(ラジアン)
float GetLife();//現在のオブジェクトの寿命を取得します。
bool GetDestoryFlag();//現在の削除フラグを取得します。(true で削除対象オン)
bool GetMainCameraFlag();//現在のオブジェクトがメインカメラの対象になっているかのフラグを取得します。
size_t GetHitPointsVolume();//現在の当たっている場所の総数を取得します。
VECTOR GetHitPosition();//現在の当たっている場所の位置を取得します。
list<VECTOR>::iterator GetHitPointsListIterator(list<VECTOR>::iterator* pointer_to_iterator);//hit_points_list のイテレータを取得します。(使用非推奨:イテレータ指定版)
list<VECTOR>::iterator GetHitPointsListIterator();//要検証：hit_points_list のイテレータを取得します。(使用推奨：安全版。list<>::begin()が初期値)
list<VECTOR>* GetHitPointsListPointer();//hit_points_list へのポインタを取得します。
```

DiploidSoftImage

```
void Load(const char* path);
int GetGraphicsHandle();//一枚の画像としてのハンドルを得る。
void CreatSoftImage(float x = 1, float y = 1);//BOXを用いて描画する際のデータを作成します。(x,yには１ドットの大きさを指定,resizeは1以上を指定)
void CreatGraphicsImage(int x = 1, int y = 1);//Imageを用いて描画する際のデータを作成します。(x,yには1ドットの大きさを指定,resizeは1以上を指定)
void SetSoftPixelSize(float x, float y);//描画で使用しているBOXの大きさを変更します。(非推奨)
void SetSoftPosition(VECTOR pos);
void SetSoftFill(bool fill);
void SetSoftThickness(float thickness);
void SetGraphicsScale(float x, float y);
void Draw(bool debug = false);//GPU経由で描画します。(未実装)
void SoftwareDraw(bool debug = false);//BOXを用いて描画します。
void GraphicsDraw(bool debug = false);//DrawGraphを用いて描画します。
void SoftImageDelet();
```

DiploidStrings

```
int CreateFontData(int Size, int Thick, int FontType, char *FontName = NULL);//DXLIBを参照
int GetHandle();
void Load(const char* str = "");//¥nで改行を表す。(1行512文字、５改行まで)
void Init(float x, float y, int new_font_handle);
void Init(float x, float y);
void ChangeFont(int handle);
void Reset();//文字配列の中を削除し、再び文字送り描画をします。(バグにつき使用不可)
int GetEnd();//文字列の描画が終わったかのステータスを取得します。(1で描画終了、0で描画中)
std::string GetSceneName();
void SetSceneName(std::string name);
void AllIn();//全ての文字を表示します。(読み込んだ文字列を全て表示用配列にコピーします。)
bool GetCompleteFlag();//既読かどうかのフラグを得ます。
void SetCompleteFlag(bool new_flag);
void SetLineSpaceing(int space);
void SetSpeed(int speed);//文字送りの速度を設定します。
void Draw();//Load関数で読み込んだ文字列を表示する。(1行512文字)
```

DiploidCamera

```
void SetMainCameraPosition(VECTOR* target);//targetの存在する位置にカメラを移動させます。
void SetMainCameraPosition_DiploidCircleV2(DiploidCircleV2* target);//targetの存在する位置にカメラを移動させます。(Circle版)
void SetOtherLookObjectPosition_DiploidCircleV2(DiploidCircleV2* target);//メインカメラの位置に移動していないオブジェクトを正しく移動させます。(Circle版)
void SetMainCameraPosition_DiploidBoxV2(DiploidBoxV2* target);//targetの存在する位置にカメラを移動させます。(Box版)
void SetOtherLookObjectPosition_DiploidBoxV2(DiploidBoxV2* target);//メインカメラの位置に移動していないオブジェクトを正しく移動させます。(Box版)
void SetMainCameraCenterPosition(VECTOR new_position);//ターゲットを画面のどこに移動させるのか設定します。
VECTOR GetMainCameraCenterPosition();//メインカメラの位置を取得します。
```

DiploidCollision

```
bool CircleAndCircleCollisionUpdate(DiploidCircleV2* circle_one, DiploidCircleV2* circle_two, int updata_rate = 0);//円と円の当たり判定を実行します。(戻り値は二つの円がヒットしていたらtrueを返します。)
VECTOR CircleAndCircleCollisionPointsUpdate(DiploidCircleV2* circle_one, DiploidCircleV2* circle_two, int updata_rate = 0);//円と円が当たった瞬間の位置を得ます。(戻り値は当たった場所の位置を返します。zには半径が入っています。)
bool BoxAndMouseCollisionUpdate(DiploidBoxV2* box, int mouse_x, int mouse_y, int updata_rate = 0);//四角とマウスの当たり判定を実行します。(戻り値はマウスと四角がヒットしていたらtrueを返します。)
int GetUpdateCounter();//更新頻度を数えるカウンターを取得します。
```

DiploidAnimation

```
void Load(const char* path);
void Init(VECTOR pos, int sheet, float scale_x = 1.0f, float scale_y = 1.0f);
void Update();
void SetPosition(VECTOR  new_pos);
void SetScale(float new_x, float new_y);
void SetNextAnimationTime(int new_time);//次の画像へ行く時間を設定する。
void SetAnimationSpeed(int new_Speed);//アニメーションのスピードを設定する。
void Reset();//アニメーションを最初から再生(番号初期化)
int GetAnimationNumber();//描画しているコマが何番目なのかを調べます。
int GetAnimationSpeed();//現在のアニメーションのスピードを調べます。
void Draw(bool debug = false);//一枚づつ描画
void OneCellDraw(int number = 0, bool draw = true);//一枚だけ描画
void StackDraw(bool debug = false);//重ねて描画
void AllPop();
size_t GetImagesVectorSize();
```

DiploidNovelScene

```
void SetAlphaSpeed(int new_speed);
void Load(const char* path, std::string name);
void Init(VECTOR pos);
void SetDrawName(std::string name);//名前による画像の変更。
void Update();//場面の切り替え処理
void Draw(bool draw = true);
void AlphaMax();//アルファ値を最大にします。
void NameDraw(std::string name, bool draw = true);//オブジェクトの名前を指定して描画(非推奨)
```

DiploidEngineScreen

```
void FrameCount();
void Init(int setting_fps = 60);
void Update();
void Draw(int x = 0, int y = 0, bool debug = false);
void Wait();
float GetDeltaTime();//次のフレームに行くまでに掛かった時間を取得します。(未実装)
```

DiploidSelectedUIV2

```
void Load(int graphics_handl);
void Init(VECTOR pos, VECTOR size, float scale = 1.0f);//UIを初期化します。
void Updata(DiploidEngineInput* input);//UIを更新します。
void Draw(bool draw = true, bool debug = false);//UIを描画します。
void SetPosition(int pos_x, int pos_y);//新しく位置を設定します。(左上基準)
void SetSize(int size_x, int size_y);//新しく大きさを設定します。(バグ)
void SetScale(float new_scale);//新しくスケール値を設定します。(バグ)
void SetSelectedUI(int new_flag);
VECTOR GetPosition();//現在のUIの位置(左上)の座標を取得します。
VECTOR GetSize();//現在のUIの大きさを取得します。
float GetScale();
VECTOR GetGraphicsSize();//画像の大きさを取得します。
bool GetHit();//現在、UIにマウスがヒットしているかのフラグを取得します。(ヒットしていればtrue)
bool GetClick();//UIがクリックされたかのフラグを取得します。(クリックされるとtrue)
int GetSelectedUI();//UIが選択されているかのフラグを取得します。(選択されていればtrue)
```

DiploidSliderObject

```
void Load(int graphics_handl);
void Init(VECTOR pos, VECTOR size, int slider_type = 0, float scale = 1.0f);//UIを初期化します。
void Updata(DiploidEngineInput* input);//UIを更新します。
void Draw(bool draw = true, bool debug = false);//UIを描画します。
void SetPosition(int pos_x, int pos_y);//新しく位置を設定します。(左上基準)
void SetSize(int size_x, int size_y);//新しく大きさを設定します。(バグ)
void SetScale(float new_scale);//新しくスケール値を設定します。(バグ)
void SetSelectedUI(int new_flag);
void SetSliderType(int new_type);
VECTOR GetPosition();//現在のUIの位置(左上)の座標を取得します。
VECTOR GetSize();//現在のUIの大きさを取得します。
VECTOR GetGraphicsSize();//画像の大きさを取得します。
float GetScale();
bool GetHit();//現在、UIにマウスがヒットしているかのフラグを取得します。(ヒットしていればtrue)
bool GetClick();//UIがクリックされたかのフラグを取得します。(クリックされるとtrue)
int GetSelectedUI();//UIが選択されているかのフラグを取得します。(選択されていればtrue)
```

ContibuousTriangle

```
void Init(VECTOR pos, float length,int size_x, int size_y);
void Updata(DiploidEngineInput& input);
void Draw(bool draw = true, bool debug = false);
```

DiploidBox

```
void Init(VECTOR position = { 0,0,0 }, VECTOR size = { 0,0,0 });//マウスに追従させる場合は{0,0,0}を代入
void Update();//ポジションなどの数値を変更したら必ず呼ばないと結果が反映されない。
void Draw(bool wire = true);
VECTOR SetCenterPosition(VECTOR new_center_position);//中央ポジションを任意の数値に変更する。(戻り値はnew_center_positionを返す)
VECTOR GetSize();//縦と横の大きさを返します。
VECTOR GetPosition(VECTOR get_pos = { 0,0,0 });//指定した場所の位置を返します。例)get_posに{1,0,0}を代入→四角の右上のポジションを指定→位置をVECTOR型で返す。{0,0,0}で左上
VECTOR GetCenterPosition();//中央の位置を返します。
void TransformScale(float scale = 1.0f);//図形をスケーリングします。
void Destory() { impacted = destory = true; };//削除します。
int SetDrawNameTagFlag(int flag = TRUE);//TRUEでname_tagを表示。
std::string GetNameTag();//name_tagに設定されている名前を返します。
```

DiploidCircle

```
void Init(VECTOR position, float size);
void Update();
void Draw(bool wire = true);
float GetSize();//円の大きさを返します。
VECTOR GetCenterPosition();//中央の位置を返します。
void Destory() { impacted = destory = true; };
int SetDrawNameTagFlag(int flag = TRUE);//TRUEでname_tagを表示。
std::string GetNameTag();//name_tagに設定されている名前を返します。
```

DiploidImage

```
void Load(const char* path);//画像読み込み
void Init(VECTOR pos = { 0.0f,0.0f,0.0f }, double size_scale = 1.0f, double angle_scale = 0.0f);//初期位置(size_scaleは1で等倍、2で二倍など)
void Updata();//位置やアニメーションなどの更新(つまり主に座標移動)
void Draw(bool draw = true);//画像を表示
int GetGraphicsSizeX();//読み込んだ画像のXの大きさを取得します。
int GetGraphicsSizeY();//読み込んだ画像のYの大きさを取得します。
void TransformScale(float scale = 1.0f);//サイズスケールを変更する。
int SetDrawNameTagFlag(int flag = FALSE);//TRUEでname_tagを表示。
std::string GetNameTag();//name_tagに設定されている名前を返します。
```

DiploidLine

```
void Init(VECTOR position_one, VECTOR position_two);
void Update();
void Draw(bool wire = false);
void Destory() { impacted = destory = true; };
double GetAngle();//線分の傾きを取得
int SetDrawNameTagFlag(int flag = TRUE);//TRUEでname_tagを表示。
std::string GetNameTag();//name_tagに設定されている名前を返します。
```

DiploidPoint

```
void Init(VECTOR position);//マウスに追従させる場合は{0,0,0}を代入
void Update();
void Draw(bool wire = true);
void Destory() { impacted = destory = true; };
std::string GetNameTag();//name_tag に設定されている名前を返します。
```

DiploidEngineImpact

```
void GetSize();//各配列の大きさを取得
void PushCircle(DiploidCircle circle);//円を円配列にプッシュします。
void PushPoint(DiploidPoint point);//点を点配列にプッシュします。
void PushBox(DiploidBox box);//四角を四角配列にプッシュします。
void PushLine(DiploidLine line);//線分を線分配列にプッシュします。
void PopBackCircle();//円を一番後ろの配列から削除します。
void PopBackPoint();//点を一番後ろの配列から削除します。
void PopBackBox();//四角を一番後ろの配列から削除します。
void PopBackLine();//線分を一番後ろの配列から削除します。
void DestoryCircle();//円がヒットしていたら円配列から削除
void DestoryPoint();//点がヒットしていたら点配列から削除
void DestoryBox();//四角がヒットしていたら四角配列から削除
void DestoryLine();//線分がヒットしていたら線分配列から削除
void DestoryBox_Name_Tag(std::string name_tag);//name_tagから対象を見つけ出し四角配列から削除
void DestoryCircle_Name_Tag(std::string name_tag);//name_tagから対象を見つけ出し円配列から削除
void DestoryPoint_Name_Tag(std::string name_tag);//name_tagから対象を見つけ出し点配列から削除
void DestoryLine_Name_Tag(std::string name_tag);//name_tagから対象を見つけ出し線配列から削除
void Destory();//配列から削除
void Init();//初期化
void Updata();//衝突判定処理
void AutoNumber();//オブジェクト番号を付けるのがめんどいときの自動オブジェクト番号付与処理。(全てのオブジェクトは番号が異なるようになる。ループ処理に組み込むと動作が重くなるかも)
void Draw(bool wire = true, bool debug = false);//描写
        //Updata()を使って処理をする場合は以下のImpact~関数を使用しない。
void ImpactCirclePoint();//円と点の当たり判定処理(画面外の判定は行わない処理を実装ずみ){要バグ確認}
void ImpactCircleCircle();//円と円の当たり判定処理
void ImpactBoxPoint();//四角と点の当たり判定処理(画面外の判定は行わない処理を実装ずみ)
void ImpactBoxBox();//四角と四角の当たり判定処理(画面外の判定は行わない処理を実装ずみ)
void ImpactBoxCircle();//四角と円の当たり判定処理
void ImpactCircleLine();//円と線分の当たり判定処理
void ImpactPointLine();//点と線分の当たり判定処理
void ImpactLineLine();//線分と線分の当たり判定処理
void ImpactLineBox();//線分と箱の当たり判定処理
        //アニメーション値を変更する関数。
//BOX(Position)
void SetBoxPositionAnimation(int number, VECTOR move_speed = { 0.0f,0.0f,0.0f });//box配列から特定のboxを見つけて座標アニメーション値を変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetBoxPositionAnimationX(int number, float move_speed = 0.0f);//box配列から特定のboxを見つけてX座標アニメーション値だけを変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetBoxPositionAnimationY(int number, float move_speed = 0.0f);//box配列から特定のboxを見つけてY座標アニメーション値だけを変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetBoxPositionAnimationZ(int number, float move_speed = 0.0f);//box配列から特定のboxを見つけてZ座標アニメーション値だけを変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
```

```
void SetBoxPositionAnimation_Sreach_Object_Name(std::string name_tag, VECTOR move_speed = { 0.0f,0.0f,0.0f });//box配列から特定のboxを見つけて座標アニメーション値を変更する。(name_tagにはオブジェクト名を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetBoxPositionAnimationX_Sreach_Object_Name(std::string name_tag, float move_speed = 0.0f);//box配列から特定のboxを見つけてX座標アニメーション値だけを変更する。(name_tagにはオブジェクト名を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetBoxPositionAnimationY_Sreach_Object_Name(std::string name_tag, float move_speed = 0.0f);//box配列から特定のboxを見つけてY座標アニメーション値だけを変更する。(name_tagにはオブジェクト名を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetBoxPositionAnimationZ_Sreach_Object_Name(std::string name_tag, float move_speed = 0.0f);//box配列から特定のboxを見つけてZ座標アニメーション値だけを変更する。(name_tagにはオブジェクト名を入れること。配列に追加した後に数値を変更したい場合に使用)

//CIRCLE(Position)
void SetCirclePositionAnimation(int number, VECTOR move_speed = { 0,0,0 });//circle配列から特定のcircleを見つけて座標アニメーション値を変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetCirclePositionAnimationX(int number, float move_speed = 0.0f);//circle配列から特定のcircleを見つけてX座標アニメーション値だけを変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetCirclePositionAnimationY(int number, float move_speed = 0.0f);//circle配列から特定のcircleを見つけてY座標アニメーション値だけを変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetCirclePositionAnimation_Sreach_Object_Name(std::string name_tag, VECTOR move_speed = { 0,0,0 });//circle配列から特定のcircleを見つけて座標アニメーション値を変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetCirclePositionAnimationX_Sreach_Object_Name(std::string name_tag, float move_speed = 0.0f);//circle配列から特定のcircleを見つけてX座標アニメーション値だけを変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetCirclePositionAnimationY_Sreach_Object_Name(std::string name_tag, float move_speed = 0.0f);//circle配列から特定のcircleを見つけてY座標アニメーション値だけを変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)

//POINT(Position)
void SetPointPositionAnimation(int number, VECTOR move_speed = { 0,0,0 });//point配列から特定のpointを見つけて座標アニメーション値を変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetPointPositionAnimationX(int number, float move_speed = 0.0f);//point配列から特定のpointを見つけてX座標アニメーション値だけを変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetPointPositionAnimationY(int number, float move_speed = 0.0f);//point配列から特定のpointを見つけてY座標アニメーション値だけを変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetPointPositionAnimation_Sreach_Object_Name(std::string name_tag, VECTOR move_speed = { 0,0,0 });//point配列から特定のpointを見つけて座標アニメーション値を変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetPointPositionAnimationX_Sreach_Object_Name(std::string name_tag, float move_speed = 0.0f);//point配列から特定のpointを見つけてX座標アニメーション値だけを変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetPointPositionAnimationY_Sreach_Object_Name(std::string name_tag, float move_speed = 0.0f);//point配列から特定のpointを見つけてY座標アニメーション値だけを変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
```

```
void SetBoxSizeAnimation(int number, VECTOR move_size = { 0,0,0 });//box配列から特定のboxを見つけて拡大アニメーション値を変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetCircleSizeAnimation(int number, float move_size = 0.0f);//circle配列から特定のcircleを見つけて拡大アニメーション値を変更する。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetBoxPositionAdd(int number, VECTOR position = { 0,0,0 });//box配列から特定のboxを見つけて元ある座標に指定数値分だけを足す。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetBoxPositionYAdd(int number, float position = 0.0f);//box配列から特定のboxを見つけて元あるY座標に指定数値分だけを足す。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetCirclePositionAdd(int number, VECTOR position = { 0,0,0 });//circle配列から特定のcircleを見つけて元ある座標に指定数値分だけを足す。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetPointPositionAdd(int number, VECTOR position = { 0,0,0 });//point配列から特定のpointを見つけて元ある座標に指定数値分だけを足す。(numberにはオブジェクト番号を入れること。配列に追加した後に数値を変更したい場合に使用)
void SetBoxPosition(int number, VECTOR position = { 0.0f,0.0f,0.0f });
VECTOR GetBoxCenterPosition(int number);//box配列から特定のboxを見つけてそのboxの中心点の座標を得ます。
VECTOR GetCircleCenterPosition(int number);//circle配列から特定のcircleを見つけてそのcircleの中心点の座標を得ます。
VECTOR GetPointPosition(int number);//point配列から特定のpointを見つけてそのpointの座標を得ます。
//各配列の大きさを得る関数
int GetEndBox();//box配列の現在の一番最後の数を取得。(配列になにもなければ-1を返す。)
int GetEndCircle();//circle配列の現在の一番最後の数を取得。(配列になにもなければ-1を返す。)
int GetEndPoint();//point配列の現在の一番最後の数を取得。(配列になにもなければ-1を返す。)
int GetEndLine();//line配列の現在の一番最後の数を取得。(配列になにもなければ-1を返す。)
//どれかの配列を基準に他の配列と大きさを比較して大きければ基準の配列の大きさを返す。
int GetMaxBoxNumber();//box配列を基準に、一番大きい値を返す。つまり、box以外のオブジェクトと比較して大きければ最大数を返す。(配列になにもなければ-1を返す。)
int GetMaxPointNumber();//point配列を基準に、一番大きい値を返す。つまり、point以外のオブジェクトと比較して大きければ最大数を返す。(配列になにもなければ-1を返す。)
int GetMaxCircleNumber();//circle配列を基準に、一番大きい値を返す。つまり、circle以外のオブジェクトと比較して大きければ最大数を返す。(配列になにもなければ-1を返す。)
int GetMaxLineNumber();//line配列を基準に、一番大きい値を返す。つまり、line以外のオブジェクトと比較して大きければ最大数を返す。(配列になにもなければ-1を返す。)
int GetMaxArrayNumber();//box.point.circle.line配列をそれぞれ比べて一番大きい要素数を返す。
int GetAllArraySize();//全ての配列のサイズの合計を返す。
//オブジェクト番号を調べる関数。
int GetBoxNumber(int target);//box配列の最初から数えてx番目のオブジェクト番号を取得する。
int GetPointNumber(int target);//point配列の最初から数えてx番目のオブジェクト番号を取得する。
int GetCircleNumber(int target);//circle配列の最初から数えてx番目のオブジェクト番号を取得する。
int GetLineNumber(int target);//line配列の最初から数えてx番目のオブジェクト番号を取得する。
```

```
//指定のオブジェクト番号を変更する関数。(要検証)
void SetBoxNumber(int target_number, int set_number);//box配列の指定データのオブジェクト番号(number変数)を変更します。(target_numberには配列頭から数えて何番目の値を変えるのかを指定、set_numberには実際に変え値)
void SetCircleNumber(int target_number, int set_number);//circle配列の指定データのオブジェクト番号(number変数)を変更します。(target_numberには配列頭から数えて何番目の値を変えるのかを指定、set_numberには実際に変え値)
void SetPointNumber(int target_number, int set_number);//point配列の指定データのオブジェクト番号(number変数)を変更します。(target_numberには配列頭から数えて何番目の値を変えるのかを指定、set_numberには実際に変え値)
void SetLineNumber(int target_number, int set_number);//line配列の指定データのオブジェクト番号(number変数)を変更します。(target_numberには配列頭から数えて何番目の値を変えるのかを指定、set_numberには実際に変え値)
void SetNumber(int number);//未実装
//レイヤー番号を調べる関数。
int GetBoxLayerNumber(int target);//box配列の最初から数えてx番目のレイヤー番号を取得する。
int GetPointLayerNumber(int target);//point配列の最初から数えてx番目のレイヤー番号を取得する。
int GetCircleLayerNumber(int target);//circle配列の最初から数えてx番目のレイヤー番号を取得する。
int GetLineLayerNumber(int target);//line配列の最初から数えてx番目のレイヤー番号を取得する。
//指定のレイヤー番号を変更する関数。(要検証)
void SetBoxLayerNumber(int target_number, int set_number);//box配列の指定データのレイヤー番号(layer_number変数)を変更します。(target_numberには配列頭から数えて何番目の値を変えるのかを指定、set_numberには実際に変え値)
void SetCircleLayerNumber(int target_number, int set_number);//circle配列の指定データのレイヤー番号(layer_number変数)を変更します。(target_numberには配列頭から数えて何番目の値を変えるのかを指定、set_numberには実際に変え値)
void SetPointLayerNumber(int target_number, int set_number);//point配列の指定データのレイヤー番号(layer_number変数)を変更します。(target_numberには配列頭から数えて何番目の値を変えるのかを指定、set_numberには実際に変え値)
void SetLineLayerNumber(int target_number, int set_number);//line配列の指定データのレイヤー番号(layer_number変数)を変更します。(target_numberには配列頭から数えて何番目の値を変えるのかを指定、set_numberには実際に変え値)
//オブジェクトが衝突しているか調べる関数。
bool GetBoxImpactFlag(int number);//box配列のオブジェクト衝突フラグを取得する。(numberにはオブジェクト番号、0でヒットなし、1でヒットしている)
bool GetPointImpactFlag(int number);//point配列の最初から数えてx番目のオブジェクト衝突フラグ取得する。(numberにはオブジェクト番号、0でヒットなし、1でヒットしている)
bool GetCircleImpactFlag(int number);//circle配列の最初から数えてx番目のオブジェクト衝突フラグ取得する。(numberにはオブジェクト番号、0でヒットなし、1でヒットしている)
bool GetLineImpactFlag(int number);//line配列の最初から数えてx番目のオブジェクト衝突フラグ取得する。(numberにはオブジェクト番号、0でヒットなし、1でヒットしている)
bool GetBoxImpactFlag_Sreach_Name_Tag(std::string name_tag);//box配列のオブジェクト衝突フラグを取得する。(numberにはオブジェクト番号、0でヒットなし、1でヒットしている)
bool GetPointImpactFlag_Sreach_Name_Tag(std::string name_tag);//box配列のオブジェクト衝突フラグを取得する。(numberにはオブジェクト番号、0でヒットなし、1でヒットしている)
bool GetCircleImpactFlag_Sreach_Name_Tag(std::string name_tag);//box配列のオブジェクト衝突フラグを取得する。(numberにはオブジェクト番号、0でヒットなし、1でヒットしている)
```

```
bool GetLineImpactFlag_Sreach_Name_Tag(std::string name_tag);//box配列のオブジェクト衝突フラグを取得する。(numberにはオブジェクト番号、0でヒットなし、１でヒットしている)
DiploidBox GetBoxInfo(int number);
```