



ユーザインタフェースを持つアプリの作成





プロジェクトの作成

projectGeneratorを起動する

windows 版のパッケージ



macOS 版のパッケージ



空のプロジェクトの作成

*	create / update	Project nar クトを作る (自分で記	me は フ るたびに 役定して	プロジェ ニ 変わる こも可)	
	Project name:				
	myCasualSketch	24	import		
	Project path:				
	<openframework< td=""><td>sの展開場所>¥apps</td><td>s¥myApps</td><td>Q</td><td></td></openframework<>	sの展開場所>¥apps	s¥myApps	Q	
	Addons:	そのまま	E		
	Addons		1	_	
	ofxAssimpModelLo	ader	-	A	
	ofxGPS of	xAssimp	/odell	Loade	rを追加
	ofxGui				
	ofxKinect				
	ofxNetwork				
	ofxOpenCv				
	ofxOsc			-	

	create / update			
	Project name:			
	myCasualSketch	x\$ import	t	
	Project path:			
	<openframeworksの展開場< td=""><td>訮>¥apps¥myApps</td><td>s Q</td><td></td></openframeworksの展開場<>	訮>¥apps¥myApps	s Q	
	Addons: ~その)まま		
	ofxAssimpModelLoader ×		•	
	Platforms: OfxA	ssimpMo	odelLoad	ler
	Windows (Visual Studio 2017) その Generate	×)まま	•	
フ	ロジェクト作	■成		

ofxAssimpModelLoader アドオンについて

- 3D モデルやアニメーションの読み込みと表示を行う
 - Open Asset Import Library" (assimp) <u>https://www.assimp.org/</u>
- マニュアル
 - https://openframeworks.cc/documentation/ofxAssimpModelLoader/ofx AssimpModelLoader/
 - 現時点で 3DS, ASE, DXF, HMP, MD2, MD3, MD5, MDC, MDL, NFF, PLY, STL, X, LWO, OBJ, SMD, Collada, Ogre XML, partly LWS の読み込みに対応

プロジェクトの作成成功

Suce You	C:\of_v0.11.2_vs2017_rele Addons: cess!	paselapps\imyAj	ックし	て開く
<ope [noti [noti [noti [noti [noti</ope 	ice]	T>¥apps¥myA 1.2_vs2017_release _vs2017_release\app	pps¥myCasualS	iketch
[noti h	ice] setting up new project C:\of_v	0.11.2_vs2017_releas	se\apps\myApps\myCasu	lose

📙 🛛 🚽 📙 🗢 🛛 myCasualSketch		—	
ファイル ホーム 共有 表示			×
← → • ↑ 📙 « myA → myCas	sualS ∨ ē	, P myCasualSketcl	hの検索
名前 ^	更新日時	種類	サイズ
📊 bin	2021/07/25 10:42	ファイル フォルダー	
src 🔤	2021/07/25 10:42	ファイル フォルダー	
📄 addons.make	2021/07/25 10:42	MAKE ファイル	1 KB
🛃 icon.rc	2021/03/24 19:29	Resource Script	1 KB
📳 myCasualSketch.sln	2021/07/25 10:42	Microsoft Visual S	з КВ
🛐 myCasualSketch.vcxproj	2021/07/25 10:42	VC++ Project	19 KB
🗊 myCasualSketch.vcxproj.filters	2021/07/25 10:42	VC++ Project Filte	12 KB
myCasualSketch.vcxproj.user	2021/07/25 10:42	USER ファイル	2 KB

Visual Studio は まだ起動しない

8個の項目 |

model.zip のダウンロードと展開



bin > data にファイルを配置

📙 🛛 🛃 📊 🛨 🛛 model			_			data			
ファイル ホーム 共有	表示			× ?	ファイル ホーム	共有 表示			~
← → • ↑	7トップ → model	م ن	modelの検索		$\leftarrow \rightarrow \cdot \uparrow$	<mark>_</mark> ≪ bin → data	ڻ ~	♀ dataの検索	
	^ 名前	更新日時	種類	サイズ	名前	^	更新日時	種類	サイズ
オ クイックアクセス	l-hand.mtl	2021/07/24 17:08	MTL ファイル	1 KB	.gitkeep		2021/03/24 19:29	GITKEEP ファイル	0 KB
📥 OneDrive - Personal	o Direction in the second seco	2021/07/24 17:08	3D Object	61 KB					
🔷 OpeDrive - 和歌山大!	📄 I-hinge.mtl	2021/07/24 17:09	MTL ファイル	1 KB					
опериме - днад д Д.	🙆 I-hinge.obj	2021/07/24 17:09	3D Object	31 KB					
💻 PC	📑 I-lower.mtl	2021/07/24 17:09	MTL ファイル	1 KB		20			
🧊 3D オブジェクト	o I-lower.obj	2021/07/24 17:09	CONTRACTOR OF	58 KB					
🖊 ダウンロード	📄 I-shoulder.mtl	2021/07/24 17:24		1 KB		\checkmark			
💻 デスクトップ	💩 l-shoulder.obj	2021/07/24 17:24		209 KB	_	→ data へ移動			
model	l-upper.mtl	2021/07/24 17:09		1 KB					
ドキュメント	🧭 l-upper.obj	2021/07/24 17:09	3D Object	131 KB					
■ パユ/21	r-hand.mtl	2021/07/24 17:10	MTL 77'	B			L キープロミッエ	クトの	
E C777	🧭 r-hand.obj	2021/07/24 17:10	3D Object	61 KB		ТЕЛЖ			
E77	r-hinge.mtl	2021/07/24 17:10	MTL ファイル	1 KB		bir	ヽ> data フォル	タに	
🎝 ミュージック	🧭 r-hinge.obj	2021/07/24 17:10	3D Object	31 KB			置いてくださ	(L)	
🏪 Windows (C:)	r-lower.mtl	2021/07/24 17:10	MTL ファイル	1 KB					
👝 Backup (D:)	🧭 r-lower.obj	2021/07/24 17:10	3D Object	58 KB					
	r-shoulder.mtl	2021/07/24 17:23	MTL ファイル	1 KB					
イットワーク	Ø r-shoulder.obj	2021/07/24 17:23	3D Object	203 KB					
	r-upper.mtl	2021/07/24 17:11	MTL ファイル	1 KB	1個の項目				

ソリューションファイルを開く

_	Addons:
Suce You <ope< td=""><td>cess! r can now find your project in enFrameworks の展開場所>¥apps¥myApps¥myCasualSketch</td></ope<>	cess! r can now find your project in enFrameworks の展開場所>¥apps¥myApps¥myCasualSketch
[noti [noti [noti [noti [noti h	<pre>ce] ce] setting OF path to: C:\of_v0.11.2_vs2017_release ce] from -o option ce] target platform is: vs ce] project path is: C:\of_v0.11.2_vs2017_release\apps\myApps\myCasualSketch ce] setting up new project C:\of_v0.11.2_vs2017_release\apps\myApps\myCasualSketc</pre>
	IDE で開く → Open in IDE Close

📙 🛛 💆 📙 🛨 🛛 myCası		- C	×		
ファイル ホーム 共有	ī 表示				×
← → • ↑ <mark> </mark> «	myA → myCasu →	~	ඊ / myCa	asualSketchの検	
名前 ^	これをダ	ブ	種類	של: של:	x
📑 bin	ルクリッ	ク 0:42	ファイル フォノ	ダー	
🔄 src		0:42	ファイル フォノ	レダー	
addons.make	してもよ	0:42	MAKE ファイ	JL	1 KB
🖾 icon.rc	EVE	, voj z4 19:29	Resource S	cript	1 KB
🖉 myCasualSketch.sln	2021	/07/25 10:42	Microsoft \	/isual S	3 KB
🛐 myCasualSketch.vcx	proj 2021	/07/25 10:42	VC++ Proje	ct	19 KB
nyCasualSketch.vcx	proj.filters 2021	/07/25 10:42	VC++ Proje	ect Filte	12 KB
myCasualSketch.vcx	proj.user 2021	/07/25 10:42	USER ファイノ	ŀ	2 KB

8個の項目 | 1個の項目を選択 2.01 KB |

Visual Studio が起動する



ソリューションの再ターゲット

ソリューション操作の再ターゲット ×	
プロジェクトの再ターゲット	
次のプロジェクトは、以前のバージョンの Visual C++ プラットフォーム ツールセットを使用しています。これらのプロ ジェクトは、最新の Microsoft ツールセットをターゲットとするようにアップグレードできます。また、お使いのマシン にインストールされているものからターゲットの Windows SDK バージョンを選択することもできます。	
Windows SDK バージョン: 10.0 (最新のインストールされているバージョン) 、	
プラットフォーム ツールセット: v142 へのアップグレード v	
 ¥myCasualSketch¥myCasualSketch.vcxproj ¥vs¥openframeworksLib.vcxproj 	
OK キャンセル	

Visual Studio は頻繁に更新しているので皆さんがお使いの Visual Studio SDK のバージョンと合わない場合がある

Visual Studio 起動		
 マァイル(F) 編集(E) 表示(V) Git(G) プロジェクト(P) ビルド(B) デバッグ(D) テスト(S) 分析(N) ツール(T) 拡張機能(X) ウィンドウ(W) ヘルプ(H) … P myCetch ○ • ○ 潤 • 論 単 ♡ • ♡ • ▼ Debug • Win32 • ▶ ローカル Windows デバッガー・ 自動 • ■ 函 - 	— □	× R
ソリューション エクスプローラー ・ ・ ・ ソリューション エクスプローラーの検索 (Ctrl+:) ・ ・ コン ソリューション 'myCasualSketch' (2/2 プロジェクト) ・ ・ ● 動 myCasualSketch ・ ● ● openframeworksLib ・ ・		サーバー エクスプローラー ツールボックス 通知 プロパテ
出力 出力元(S): 全般 構成 'Debug x64': プラットフォーム ツールセットを 'v142' に変更しています (以前は 'v141')。 構成 'Release ₩in32': プラットフォーム ツールセットを 'v142' に変更しています (以前は 'v141')。	- ₽ >	×
構成 'Release x84': ブラットフォーム ツールセットを 'v142' に変更しています (以前は 'v141')。 再ターゲットの終了: 完了 2、失敗 0、スキッブ 0 4 Git 変更 <u>ソリューショ クラス ビュー プロパティ リソース ビュー</u> 出力 シンボルの検索結果 エラー一覧		•
□ 準備完了 ↑ ソース	(管理に追加 🔺 🛛 🦂	1 4

3D モデルの読み込みと表示

ofxAssimpModelLoader を使う



カメラ, ライト, モデルのメンバ変数を追加

#pragma once

#include "ofMain.h"
#include "ofxAssimpModelLoader.h"

```
class ofApp : public ofBaseApp{
   ofEasyCam camera;
   ofLight light;
   ofxAssimpModelLoader model;
```

public:

```
void setup();
void update();
void draw();
```

(以下略)

3D 表示に必要なメンバを追加 する

- ofCamera クラスのメンバ変数を追 加する
 - ofEasyCam クラスはマウス操作が可能
- ofLight クラスのメンバ変数を追加 する
- 3D モデルの読み込みと表示を 行う
 - ofxAssimpModelLoader クラスのメンバ変数を追加する



3D 表示の設定と 3D モデルファイルの読み込み

```
#include "ofApp.h"
```

```
//----
void ofApp::setup(){
    ofEnableDepthTest();
    light.enable();
    light.setPosition(300.0f, 400.0f, 500.0f);
    model.loadModel("l-hand.obj");
```

}

(以下略)

ofEnableDepthTest();

- 隠面消去処理を有効にする
- light.enable();
 lightによる陰影付けを有効にする
- model.loadModel("l-hand.obj");
 - プロジェクトのフォルダの bin の data の中の I-hand.obj という3D モ デルファイルを model に読み込む
 - ".obj"は Wavefront OBJ 形式の 3D モデルファイルで Blender 等で作 成できる

カメラを有効にして 3D モデルを表示する

```
void ofApp::draw(){
  camera.begin();
 model.drawFaces();
  camera.end();
}
   (以下略)
```

camera.begin() と camera.end()の 間は3次元空間の表示になる

■ 第3回参照









3D モデルの大きさを正規化しないようにする

#include "ofApp.h"

```
void ofApp::setup(){
    ofEnableDepthTest();
    light.enable();
    light.setPosition(300.0f, 400.0f, 500.0f);
    model.loadModel("1-hand.obj");
    model.setScaleNormalization(false);
```

(以下略)

model.setScaleNormalization(false)

- 引数が false なら model に読み込ん
 だ 3D モデルのサイズを正規化し
 ない
 - openFrameworksの3Dモデルのサイズ はウィンドウサイズを基準にしている (数100~数1000)
 - しかし一般的な 3D モデルのサイズは これに準じていない
 - そこで openFrameworks はデフォルト で 3D モデルのサイズを正規化する
 - そうすると複数の 3D モデルを配置する際に位置の基準がサイズと一致しないため位置合わせが難しくなる









階層化できるクラスを作る

ofNode クラスから派生する

ofxAssimpModelLoader を階層化可能にする

- ofxAssimpModelLoader は ofNode の派生クラスではない
 階層化できない
 - setParent() メソッドを持たない
 - setPosition(), setScale(), setRotation() メソッドによる設定は対象の 3D モデルのみに対して行われる
- ofNode を継承したクラスを自分で定義する
 - of3dPrimitive は ofNode の派生クラス
 - ofBoxPrimiteve, ofSpherePrimitive などは of3dPrimitive の派生クラス
 - ofNode クラスから setParent() メソッドが継承される
 - 第3回参照

ofApp.h を開く		
マファイル(F) 編集(E) 表示(V) Git(G) プロジェクト(P) ビルド(B) デパッグ(D) テスト(S) 分析(N) ツール(T) 拡張機能(X) ウィンドウ(W) ヘルプ(H) … P myCetch Win2 レーカル Windows デパッガー、自動 ■ 図 の、ジンドウ(W) ヘルプ(H) … P myCetch ソリューション I/7スプローラ- ・ # × OfApp.cpp ofApp.cpp ソリューション I/7スプローラ- ・ # × OfApp.cpp ofApp.cpp ofApp.cpp ソリューション I/7スプローラ- ・ # × OfApp.cpp ofApp.cpp ofApp.cpp ofApp.cpp ソリューション I/7スプローラ- ・ # × OfApp.cpp ofApp.cpp ofApp.cpp ofApp.cpp ofApp.cpp ソリューション I/7スプローラ- ・ # * 第 ● ● ●	− □	× P2 サーバーエクスプローラー ツールボックス 通知 プロパティ * * * *
18 void mouseMoved(int x, int y); 19 void mouseDragged(int x, int y, int button); 20 void mousePressed(int x, int y, int button); 20 void mouseReleased(int x, int y); 21 void mouseEntered(int x, int y); 23 void mouseExited(int x, int y); 24 void windowResized(int w, int h); 25 void dragEvent(ofDragInfo dragInfo); 26 void gotMessage (ofMessage msg); Git 変更 ソリューショ クラス ビュー プロパティ リソース ビュー 100 % で 問題は見つかりませんでした	列: 30 タブ	診断シール ト
□ 準備完了 ↑ ソース	管理に追加 🔺	- 1 a

ofNode を継承して Model クラスを定義する

#pragma once



class Model : public ofNode

- ofNode クラスを public 継承して派 生クラスの Model を定義する
 - 継承すると基底クラスである ofNode の機能(メソッド等)が使える
- ofxAssimpModelLoader model;
 - ofxAssimpModelLoader 型の変数 model を派生クラス Model のメン バにする
- Model hand;
 - model の代わりに自分で定義した Model 型の変数 hand を of App クラ スのメンバにする



ofNode から派生した Model クラスを使う

```
#include "ofApp.h"
```

```
void ofApp::setup(){
    ofEnableDepthTest();
    light.enable();
    light.setPosition(300.0f, 400.0f, 500.0f);
    hand.model.loadModel("1-hand.obj");
    hand.model.setScaleNormalization(false);
```

(途中略)

}

```
void ofApp::draw(){
   camera.begin();
   hand.model.drawFaces();
   camera.end();
}
```

hand は Model クラスのインスタ ンス(オブジェクト)

- model は Model クラスのメンバ で ofAssimpModelLoader 型
 - hand.model.loadModel("l-hand.obj");
 - hand.model.setScaleNormalization(fal se);
 - hand.model.drawFaces();





実行結果は変わらない



Model クラスにインスタンス lower を追加する

#pragma once

```
#include "ofMain.h"
#include "ofxAssimpModelLoader.h"
```

```
class Model : public ofNode{
```

```
public:
    ofxAssimpModelLoader model;
};
```

```
class ofApp : public ofBaseApp{
   ofEasyCam camera;
   ofLight light;
   Model hand, lower;
```

public:

(以下略)

```
void ofApp::setup(){
    ofEnableDepthTest();
    light.enable();
    light.setPosition(300.0f, 400.0f, 500.0f);
    lower.model.loadModel("1-lower.obj");
    lower.model.setScaleNormalization(false);
    hand.model.loadModel("1-hand.obj");
    hand.model.setScaleNormalization(false);
```

(途中略)

```
//----
void ofApp::draw(){
   camera.begin();
   lower.model.drawFaces();
   hand.model.drawFaces();
   camera.end();
```

ビルドして実行すると2つのモデルが重なる





```
void ofApp::setup(){
    ofEnableDepthTest();
    light.enable();
    light.setPosition(300.0f, 400.0f, 500.0f);
    lower.model.loadModel("l-lower.obj");
    lower.model.setScaleNormalization(false);
    hand.model.loadModel("l-hand.obj");
    hand.model.setScaleNormalization(false);
    hand.model.setScaleNormalization(false);
    hand.model.setScaleNormalization(false);
    hand.model.setScaleNormalization(false);
    hand.model.setScaleNormalization(false);
    hand.model.setScaleNormalization(false);
    hand.model.setScaleNormalization(false);
    hand.setPosition(150.0f, 0.0f, 0.0f);
```

(以下略)

 ofNode を継承したクラス Model のインスタンス hand に位置を 設定する

- hand に設定した位置は親からの相
 対位置になる
 - 現時点ではまだ hand に親子関係(階 層構造)は設定していない
- hand.model.setPosition(...)のように model に直接位置を設定できるが、 model は ofAssimpModelLoader クラ スのインスタンスなので、階層構 造を持つことができない

hand に設定した座標変換を使って表示する

```
void ofApp::draw(){
  camera.begin();
  lower.model.drawFaces();
  hand.transformGL();
  hand.model.drawFaces();
  hand.restoreTransformGL();
  camera.end();
```

```
(以下略)
```

■ transformGL() メソッド

- 現在の座標変換を保存した後、それに ofNode クラスの階層構造にもとづいた座標変換を合成する
- restoreTransformGL() メソッド
 - transformGL() メソッドで保存した
 変換行列を復帰する

この2つの間で図形の表示処理 を行えば ofNode クラスに設定 した階層構造にもとづいて座標 変換を行って図形が表示される
ビルドして実行すると hand が移動している



lowerの位置を設定して図形を表示する

void ofApp::setup(){
 ofEnableDepthTest();
 light.enable();
 light.setPosition(300.0f, 400.0f, 500.0f);
 lower.model.loadModel("1-lower.obj");
 lower.model.setScaleNormalization(false);
 lower.setPosition(250.0f, 0.0f, 0.0f);
 hand.model.loadModel("1-hand.obj");
 hand.model.setScaleNormalization(false);
 hand.model.setScaleNormalization(false);
 hand.model.setScaleNormalization(false);
 hand.model.setScaleNormalization(false);
 hand.model.setScaleNormalization(false);
 hand.model.setScaleNormalization(false);
 hand.model.setScaleNormalization(false);
 hand.model.setScaleNormalization(false);
 hand.model.setScaleNormalization(false);
 hand.setPosition(150.0f, 0.0f, 0.0f);
 hand.setPosition(150.0f, 0.0f, 0.0f);
 hand.setPosition(150.0f, 0.0f, 0.0f);
}

(以下略)

void ofApp::draw(){
 camera.begin();
 lower.transformGL();
 lower.model.drawFaces();
 lower.restoreTransformGL();
 hand.transformGL();
 hand.model.drawFaces();
 hand.restoreTransformGL();
 camera.end();
}

(以下略)

ビルドして実行すると lower だけ移動している



hand の親を lower にする

```
void ofApp::setup(){
    ofEnableDepthTest();
    light.enable();
    light.setPosition(300.0f, 400.0f, 500.0f);
    lower.model.loadModel("l-lower.obj");
    lower.model.setScaleNormalization(false);
    lower.setPosition(250.0f, 0.0f, 0.0f);
    hand.model.loadModel("l-hand.obj");
    hand.model.setScaleNormalization(false);
    hand.model.setScaleNormalization(false);
    hand.model.setScaleNormalization(false);
    hand.model.setScaleNormalization(false);
    hand.model.setScaleNormalization(false);
    hand.model.setScaleNormalization(false);
    hand.setPosition(150.0f, 0.0f, 0.0f);
    hand.setParent(lower);
```

hand.setParent(lower);

- hand の親を lower に設定する
- setParent()は ofNode から継承され
 たメソッド

(以下略)

hand が lower の先に移動している



メソッドを追加する インスタンスに共通する処理はメソッド化する

Model クラスに図形表示のメソッドを追加する

#pragma once

#include "ofMain.h"
#include "ofxAssimpModelLoader.h"

class Model : public ofNode{

public:

};

ofxAssimpModelLoader model;

void drawFaces(){
 transformGL();
 model.drawFaces();
 restoreTransformGL();

drawFaces() という メソッド名に特別 な意味はないので 他の名前でもよい

- この例では drawFaces() という名前 で定義している
 - drawFaces()内にある transformGL()や restoreTransformGL()は適用するオブ ジェクトを指定していない
 - これらはこの drawFaces()を適用する
 オブジェクトに対して適用される
 - transformGL()や restoreTransformGL()は ofNodeから継承された Model クラスの メソッド
 - model も drawFaces()を適用するオブ ジェクトのメンバが対象になる
 - model は Model クラスのメンバ
 - model に適用している drawFaces() は ofAssimpModelLoader クラスのメソッド

定義した drawFaces() メソッドを使う



(以下略)

■ 定義した drawFaces() メソッド を使う

transformGL(), restoreTransformGL() および model.drawFaces() メソッド は draw() メソッドから呼び出して いる

ビルドして実行すると結果は変わらない



コンストラクタを定義する オブジェクト (インスタンス)の生成時に実行する処理

Modelのvectorである parts をメンバにする

#pragma once

```
#include "ofMain.h"
#include "ofxAssimpModelLoader.h"
```

class Model : public ofNode{

public: ofxAssimpModelLoader model; };

class ofApp : public ofBaseApp{
 ofEasyCam camera;
 ofLight light;
 Model hand, lower;
 vector<Model> parts;

public:

(以下略)

parts は Model クラスの vector

parts にオブジェクトを追加する

```
void ofApp::setup(){
    ofEnableDepthTest();
    light.enable();
    light.setPosition(300.0f, 400.0f, 500.0f);
    parts.emplace_back();
    parts.back().model.loadModel("1-lower.obj");
    parts.back().model.setScaleNormalization(false);
    parts.back().setPosition(250.0f, 0.0f, 0.0f);
    parts.back().model.loadModel("1-hand.obj");
    parts.back().model.setScaleNormalization(false);
    parts.back().model.setScaleNormalization(false);
    parts.back().setPosition(150.0f, 0.0f, 0.0f);
    parts.back().setPosition(150.0f, 0.0f, 0.0f);
    parts.back().setParent(parts[0]);
```

(以下略)

- parts.emplace_back();
 - partsの最後に空のオブジェクトを 追加する
- parts.back()
 - partsの最後のオブジェクト
- parts[0]
 - partsの最初のオブジェクト
 - parts.front() と同じ
- 第3回参照

vector に格納されている全ての図形を表示する

```
//----
void ofApp::draw(){
   camera.begin();
   for (auto& part : parts){
      part.drawFaces();
   }
   camera.end();
}
```

```
(以下略)
```

for (auto& part : parts){

- parts の一つ一つの要素を part に入れて { ... } を実行する
- これを parts のすべての要素について繰り返す

■ 第3回参照

ビルドして実行するとやっぱり変わらない



コンストラクタを定義する

#pragma once

#include "ofMain.h" #include "ofxAssimpModelLoader.h" class Model : public ofNode{ ofxAssimpModelLoader model; public を指定して いないので private public: メンバになる Model(const char* file){ model.loadModel(file); model.setScaleNormalization(false); void drawFaces(){ (途中略) (以下略)

コンストラクタはクラスと同じ 名前 (Model) のメンバ関数

- オブジェクト(インスタンス)を
 生成するときに呼び出される
- 値を返さない (戻り値はない)
- コンストラクタの引数 file で読み込むファイル名を受け取る
 - そのファイル名を使って 3D モデ ルファイルを読み込む

 メンバ変数の model はコンスト ラクタからしか操作しないので private にできる

3D モデルを emplace_back の引数に指定する



 emplace_back() はオブジェクト を一つ生成して vector の最後に 追加する

- その際にコンストラクタが実行される
- emplace_back()の引数がコンスト
 ラクタの引数に与えられる

■ 第3回参照

(以下略)

これも結果にコミットしない



位置もコンストラクタの引数で指定する

#pragma once

```
#include "ofMain.h"
#include "ofxAssimpModelLoader.h"
```

```
class Model : public ofNode{
    ofxAssimpModelLoader model;
```

public:

```
Model(const char* file, float x, float y, float z){
   model.loadModel(file);
   model.setScaleNormalization(false);
   setPosition(x, y, z);
}
void drawFaces(){
   (途中略)
}
```

(以下略)

 コンストラクタの引数でモデルの 位置 x, y, z を受け取る

- setPosition() は ofNode から継承し
 た Model クラスのメソッド
 - 生成したオブジェクト(インスタン ス)に対して適用される

emplace_back()の引数に位置を追加する



 読み込んだ 3D モデルの位置を emplace_back()の引数に指定す る

- Model クラスのコンストラクタは 引数に指定したファイル名の 3D モデルファイルを読み込む
 - ofxAssimpModelLoader クラスの loadModel() メソッド
- 読み込んだ 3D モデルを表示する 位置を引数で指定された位置に設 定する
 - ofNode クラスの setPosition() メソッド

まだ結果にコミットしない



別のオブジェクトを追加する

```
void ofApp::setup(){
    ofEnableDepthTest();
    light.enable();
    light.setPosition(300.0f, 400.0f, 500.0f);
    parts.emplace_back("l-upper.obj",
        100.0f, 0.0f, 0.0f);
    parts.emplace_back("l-lower.obj",
        250.0f, 0.0f, 0.0f);
    parts.emplace_back("l-hand.obj",
        150.0f, 0.0f, 0.0f);
    parts.back().setParent(parts[0]);
}
```

 "I-upper.obj" という 3D モデル ファイルを読み込んで追加する
 モデルの位置は (100, 0, 0) とする

(以下略)

ビルドして実行すると重なって表示される





```
void ofApp::setup(){
    ofEnableDepthTest();
    light.enable();
    light.setPosition(300.0f, 400.0f, 500.0f);
    parts.emplace_back("l-upper.obj",
        100.0f, 0.0f, 0.0f);
    parts.emplace_back("l-lower.obj",
        250.0f, 0.0f, 0.0f);
    parts.emplace_back("l-hand.obj",
        150.0f, 0.0f, 0.0f);
    parts[1].setParent(parts[0]);
    parts[2].setParent(parts[1]);
```

(以下略)

■ 親子関係を変更する

- 2つ目のパーツ (parts[1]) の親を1 つ目のパーツ (parts[0]) にする
- 3つ目のパーツ (parts[1]) の親を2
 つ目のパーツ (parts[0]) にする
- 階層構造の設定は vector への要素の追加が完了した後に行う

vectorは要素を追加する際にメモリが足らないとメモリを確保し直す場合があり、その時にオブジェクトの格納場所が移動してしまう

ビルドして実行するとひと続きになる



3D モデルの階層構造



課題7-1

ロボットの両腕を完成させる

ロボットの両腕を完成させなさい

- bin > data フォルダに置いた 3D モデルファイルを読み込み、右 の表に従って階層構造を与えて 配置しなさい
 - I-upper.obj, I-lower.obj, I-hand.objの
 読み込みと配置は完了している

ファイル名	相対位置	親
l-shoulder.obj	(100, 0, 0)	(指定しない)
l-hinge.obj	(170, 0, 0)	l-shoulder.obj
l-upper.obj	(100, 0, 0)	l-hinge.obj
l-lower.obj	(250, 0, 0)	l-upper.obj
l-hand.obj	(100, 0, 0)	l-lower.obj
r-shoulder.obj	(-100, 0, 0)	(指定しない)
r-hinge.obj	(-170, 0, 0)	r-shoulder.obj
r-upper.obj	(-100, 0, 0)	r-hinge.obj
r-lower.obj	(-250, 0, 0)	r-upper.obj
r-hand.obj	(-100, 0, 0)	r-lower.obj





課題のアップロード

- 作成したプログラムの実行結果のスクリーンショットを撮って 7-1.png というファイル名で保存し、Moodle の第7回課題にアップロードしてください
 - マウスを使ってウィンドウ内に両腕が全部収まるようにしてくだ さい



課題7-2

GUIで操作する

ロボットの腕のすべての関節の角度を操作する ユーザインタフェースを追加しなさい

- パーツの角度を変更するには rotate メソッドを使う // 0 番目のパーツの角度を Y 軸中心に現状から 1 度回転する
 - parts[0].rotateDeg(1.0f, 0.0f, 1.0f, 0.0f);
 - <u>https://openframeworks.cc/documentation/3d/ofNode/#sho</u> <u>w_rotateDeg</u>
- '+' キーか '-' キーかで回転方向を変えられると良い
- パーツの番号 (0~9) もキー操作 ('0'~'9') で切り替えられると良い
 - これは static 変数に保存しておく必要があると思う

課題のアップロード

- 作成したプログラムの実行結果のスクリーンショットを撮って 7-2.png というファイル名で保存し、Moodle の第7回課題にアップロードしてください
 - GUI を操作してロボットの腕の姿勢を初期状態から変えてください
 マウスを使ってウィンドウ内に両腕が全部収まるようにしてください
- Uースプログラム of App.h と of App.cpp を Moodle の第7回
 課題にアップロードしてください

ユーザインタフェース

ofxImGui の使い方

ofxImGuiを入手する

(第7回) ユーザインタフェース (※) 第7回資料 (※) model.zip (※) ofxlmGui-1.75.zip (※) 第7回課題 (※) 第7回課題 Moodle から へいていたいたのを ダウンロードしてください

正式なものは GitHub にあります

<u> https://github.com/jvcleave/ofxImGui/releases</u>



addons に追加する

ofxImGui-1.75.zip を展開した中のフォルダを openFrameworks
 を展開したフォルダの中の
 addonsの中に移す

📦 🛃 📗 🖛	展開 ofx	lmGui-1.75.zip	– 🗆 X
ファイル ホーム 共有 表示	圧縮フォルダー ツール		~ 🕐
← → マ ↑	ード > ofxlmGui-1.75.zip >	5 v	𝒫 ofxlmGui-1.75.zip
	名前	種類	Æ
→ 🖈 シ1ツク ሥクセス	ofxImGui-1.75	ファイ	ル フォルダー
> 😹 Creative Cloud Files			
> 🥌 OneDrive - Personal			
> 📥 OneDrive - 和歌山大学			
> 📓 和歌山大学			
🗸 💻 bc			
> 🧊 3D オブジェクト			
> 🦊 ダウンロード	<		>
1 個の項目			i 🖬



projectGenerator を起動する

	管理 projectGenerator	- 🗆 X
ファイル ホーム 共有 表示	アプリケーション ツール	~ (
← → • ↑ <mark>·</mark> « of_v0.11.2_vs2	017 > projectGenerator > 🛛 🗸 💍	𝒫 projectGenerator
🏪 Windows (C:) 🔥	名前 ^	更新日時 /
Allegorithmic	locales	2021/07/26 7:40
	resources	2021/07/26 7:40
		2021/03/24 19:33
gstreamer	content_shell.pak	2021/03/24 19:33
	🚳 d3dcompiler_47.dll	2021/03/24 19:33
	📓 ffmpegsumo.dll	2021/03/24 19:33
	📄 icudtl.dat	2021/03/24 19:33
Ibrealsense2	🗟 libEGL.dll	2021/03/24 19:33
maxima-5.44.0	🗟 libGLESv2.dll	2021/03/24 19:33
msys64		2021/03/24 19:33
of_v0.11.2_vs2017_release	🚳 msvcp120.dll	2021/03/24 19:33
addons	🚳 msvcr120.dll	2021/03/24 19:33
	📄 natives_blob.bin	2021/03/24 19:33
den	🚳 node.dll	2021/03/24 19:33
docs	📑 pdf.dll	2021/03/24 19:33
examples	🛞 projectGenerator.exe 🥌	2021/03/24 19:33
libs	snapshot_blob.bin	2021/03/24 19:33
🔥 🔤 other	ui_resources_200_percent.pak	2021/03/24 19:33
projectGenerator	🧕 vccorlib120.dll	2021/03/24 19:33
🚽 scripts 🗸 🗸	<	2021/03/24 19:33
21 個の項目 1 個の項目を選択 45.5 M	1B	1

create / update import をクリック Project name: D¢ mySketch import Project path: C:\of_v0.11.2_vs2017_release\apps\myApps Q Addons: Addons... Ŧ Platforms: Windows (Visual Studio 2017) 🗶 Generate
このプロジェクトのフォルダを選択する

Select the folder of your project, typically apps/myApps/myGeniusApp					
← → × 📙 « apps	s → myApps	ڻ ~			
整理 ▼ 新しいフォルダー				HII 👻 ?	
	名前 ^	更新日時	種類	サイズ	
X 0199 792X	emptyExample	2021/07/26 7:40	ファイル フォルダー		
📥 OneDrive - Persor	📙 myCasualSketch	2021/07/26 8:02	ファイル フォルダー		
📥 OneDrive - 和歌山					
🏪 和歌山大学					
💻 PC					
🧊 3D オブジェクト					
🖊 ダウンロード					
📃 デスクトップ					
🛗 ドキュメント					
 ピクチャ					
🚆 ಲೆಸ್					
🍌 ミュージック					
🏪 Windows (C:)					
👝 Data (D:)					
D! /F.)	- myCasualSketch				
74762-					
			フォルダーの選択	キャンセル	

myCasualSketch	24	import	
Project path:			
C:\of_v0.11.2_vs2017_release\a	pps\myApps		Q
Addons:			
ofxAssimpModelLoader ×			-
Platforms:			
Windows (Visual Studio 2017)			Ŧ
Update			

ofxImGui-1.75 を addon として追加

*	create / update			
	Project name: 📝			
	myCasualSketch	24	import	
	Project path:			
	C:\of_v0.11.2_vs2017_release\a	apps\myApp	5	Q
	Addons:			
	ofxAssimpModelLoader ×			-
	ofxGPS			^
	ofxGui			
	ofxImGui-1.75			
	ofxKinect			
	ofxNetwork			
	ofxOpenCv			
	ofxOsc			•

* create/update		
Project name: 📝		
myCasualSketch 24	import	
Project path:		
C:\of_v0.11.2_vs2017_release\apps\myAp	ops Q	
Addons:		
ofxAssimpModelLoader × ofxImGui-	1.75 × •	
Platforms:		
Windows (Visual Studio 2017) 😠	Ŧ	
Update プロジェクト更新		

プロジェクト更新成功

				0
				۹
Suco You <ope< td=""><td>Addons: cess! r can now find your project in enFrameworks の展開場所>¥a</td><td>pps¥myAp</td><td>ps¥myCasu</td><th>alSketch</th></ope<>	Addons: cess! r can now find your project in enFrameworks の展開場所>¥a	pps¥myAp	ps¥myCasu	alSketch
[notis [notis [notis [notis [notis [notis	<pre>ice] ice] setting OF path to: C:\of_v0.11.2_vs; ice] from -o option ice] target platform is: vs ice] updating project C:\of_v0.11.2_vs201; ice] adding addon: ofxAssimpModelLoader ice] adding addon: ofxImGui-1.75</pre>	2017_release 7_release\apps\	, myApps∖myCasual≤	Sketch
		Open in	IDE	Close

既に Visual Studio でプロジェク トを開いているときは Close

 Visual Studio に戻るとこのスライ ドの次のページに示す警告が出る

 現在 Visual Studio でプロジェク トを開いていなければ Open in IDE で構わない

プロジェクトを開いているときは再読み込み







	🔀 ファイル(F) 編集(E) 表示(V) Git(G)	プロ	ジェクト(P) ビルド(B) デバッグ(D)	テスト(S) 分析(N)) ツール(T)	拡張機能(X)	ウィンドウ(W)	ヘルプ(H)	P	myCetch	- 0		\times
	◎ - © 🎦 - 🏠 🔛 🔐 🤊 - ୯ -	Ŷ	ソリューションの再ターゲット		ッガー 👻 自動		- 🛋 🖾	_	恒 浩 📕	위 끝	🖄 Live Share	= }	₹
j	ソリューション エクスプローラー	*.	モジュールの追加(M)…									- ¢	4
		-715 m+	クラスの追加(C)	Ctul - Chift - V	- 🛞 Mod	el		- 6	draw()			- ÷	->
		010 +,,,,	クリス ワイリート(Z)	Cur+Shirt+X									17J
	ソリューション 'myCasualSketch' (2/2 プロジェクト	*-		Chul - Chift - A									12
	 myCasualSketch (Visual Studio 2017) 	1 *-	新しい項目の追加(W)…	Curi+Shirt+A	lelLoader.h								÷
				Shint+Ait+A	ofNode {								4
	▶ ■ 外部低仔関係		新しいフイルター(F)		mouer,								ールボ
	 Image: A state of the state of	(f)	すべてのファイルを表示(O)										דלע
	ofxImGui-1.75		プロジェクトのアンロード(L)		le, float :	x, float y,	float z)						Ч
	A 🙀 src		ソリューションの再スキャン(S)		alization(false);							ם\לַ∍
	▶ ++ ofApp.cpp	1	参照データベース エラーを表示		z);								×.
	▶ 🗈 ofApp.h		参照データベース エラーをクリア										
	icon.rc		参照の追加(R)										
		t ې	接続済みサービスの管理(C)										
		₽	スタートアップ プロジェクトに設定(A)		0:								
			プロジェクトの依存関係(S)		-(),								
I			プロジェクトのビルド順序(I)										
I	Git 恋面 <u>ソルコーション エクス クラス ビュー プロパ</u>		ビルドのカスタマイズ(B)		ofBaseApp{					- 40 + - 40	51.44 h-1		
			テンフレートのエクスボート(E)…		-				P T	J:16 文字:13	911:14 97	LF	
			NUGet ハックーシの官埋(N)							↑ y-x	倉埋に追加 ▲	-1	.d
		عر	プロパティ(P)	Alt+F7									
		0	エクフプローラーでフォルダーを閂/(V)										

ソリューションの再ターゲット

ソリューション操作の再ターゲット X	
プロジェクトの再ターゲット	
次のプロジェクトは、以前のバージョンの Visual C++ プラットフォーム ツールセットを使用しています。これらのプロ ジェクトは、最新の Microsoft ツールセットをターゲットとするようにアップグレードできます。また、お使いのマシン にインストールされているものからターゲットの Windows SDK バージョンを選択することもできます。	
Windows SDK バージョン: 10.0 (最新のインストールされているバージョン) >	
プラットフォーム ツールセット: v142 へのアップグレード v	
 ¥3DPrimitivesExample¥3DPrimitivesExample.vcxproj ¥vs¥openframeworksLib.vcxproj 	
OK キャンセル	

Visual Studio は頻繁に更新しているので皆さんがお使いの Visual Studio SDK のバージョンと合わない場合がある

ofxImGui.h を #include して ofxImGui::Gui のイン スタンスをメンバーに追加

#pragma once

```
#include "ofMain.h"
#include "ofxAssimpModelLoader.h"
#include "ofxImGui.h"
```

```
class Model : public ofNode{
   (途中略)
```

```
};
```

```
class ofApp : public ofBaseApp{
   ofCamera camera;
   ofLight light;
   vector<Model> parts;
   ofxImGui::Gui gui;
```

public:

(以下略)

- ofApp.h で ofxImGui.h を #include
 する
- ofApp クラスに ofxImGui::Gui ク ラスの変数 gui をメンバに追加 する
- camera オブジェクトのクラスを ofEasyCam から ofCamera に変更 する
 - GUIの操作で視点位置が動いてしまわないようにするため

ofxImGui を初期化する

#include "ofApp.h"

```
//---
void ofApp::setup(){
    ofEnableDepthTest();
    light.enable();
    light.setPosition(300.0f, 400.0f, 500.0f);
    gui.setup();
    (以下略)
```

 ofApp.cppのsetup()で gui.setup()を実行してguiを初期 化する

GUI のウィンドウを開く



- ofDisableLighting();
 - GUI は 2 次元なので陰影付け(ラ イティング)はオフにする
- gui.begin();~gui.end();
 - この間に GUI の設定を書く
- ImGui::Begin("Control Panerl");~
 ImGui::End();
 - この間に GUI ウィンドウの設定を 書く
 - 引数 ("Control Panerl") はウィンド ウのタイトル

GUI のウィンドウが表示される





```
Void ofApp::draw(){
  ofBackground(20, 40, 60, 0);
  ofDisableLighting();
  gui.begin();
  ImGui::Begin("Control Paner1");
  ImGui::Text("Object Pose");
  ImGui::End();
  gui.end();
  ofEnableLighting();
  camera.begin();
  camera.setPosition(0.0f, 0.0f, 1000.0f);
  for (auto& part : parts){
    part.draw();
  camera.end();
}
```

ImGui::Text("Object Pose");

- GUIのウィンドウに文字を表示する
 - 日本語も表示できるが日本語を表示できるフォントに切り替える必要がある

■ "Object Pose" は表示する文字列

GUIのウィンドウに文字が表示される



図形の位置のメンバを追加する

```
class ofApp : public ofBaseApp{
   ofCamera camera;
   ofLight light;
   vector<Model> parts;
   ofxImGui::Gui gui;
   glm::vec3 position;
   public:
```

```
void setup();
void update();
void draw();
```

(以下略)

 ofApp.h で ofApp クラスに glm::vec3 型のメンバ変数 position を追加する

図形の位置を GUI で操作する

```
void ofApp::draw(){
  ofBackground(20, 40, 60, 0);
  ofDisableLighting();
  gui.begin();
  ImGui::Begin("Control Panerl");
  ImGui::Text("Object Pose");
  ImGui::DragFloat3("Position",
    reinterpret cast<float*>(&position));
  ImGui::End();
  gui.end();
  ofEnableLighting();
  camera.begin();
  camera.setPosition(0.0f, 0.0f, 1000.0f);
  parts[0].setPosition(position);
  for (auto& part : parts){
    part.draw();
  camera.end();
}
```

- ImGui::DragFloat3("Position", reinterpret_cast<float*>(&position));
 - ImGui::DragFloat3() は3つの要素を 持つ float 型の配列の内容をマウス のドラッグで変更する
 - position は glm::vec3 型だがメモリ 上のデータの並びは 3 つの要素を 持つ float 型の配列と同じなので reinterpret_cast<float*>()を使って それに見せかけている
 - position には setup() で 0 を入れる

数字の上をドラッグすると右側だけ動く



図形の角度のメンバを追加する

```
class ofApp : public ofBaseApp{
   ofCamera camera;
   ofLight light;
   vector<Model> parts;
   ofxImGui::Gui gui;
   glm::vec3 position, orientation;
   public:
      void setup();
   void update();
   void draw();
```

(以下略)

 ofApp.h で ofApp クラスに glm::vec3 型のメンバ変数 orientation を追加する

スライダーで角度を設定する

```
void ofApp::draw(){
  ofBackground(20, 40, 60, 0);
  ofDisableLighting();
  gui.begin();
  ImGui::Begin("Control Panerl");
  ImGui::Text("Object Pose");
  ImGui::DragFloat3("Position",
    reinterpret cast<float*>(&position));
  ImGui::SliderFloat("Yaw", &orientation.y,
    -180.0f, 180.0f);
  ImGui::SliderFloat("Pitch", &orientation.z,
    -180.0f, 180.0f);
  ImGui::SliderFloat("Roll", &orientation.x,
    -180.0f, 180.0f);
  ImGui::End();
  gui.end();
  ofEnableLighting();
  camera.begin();
  camera.setPosition(0.0f, 0.0f, 1000.0f);
  parts[0].setPosition(position);
  parts[0].setOrientation(orientation);
```

 ImGui::SliderFloat("Yaw", &orientation.y, -180.0f, 180.0f);
 orientation.y に -180~180の範囲で スライダで値を設定する

課題7-3 両方とも GUI で操作する

右側だけでなく左側も GUI で動くようにする

■ これまでのプログラムでは左側の腕しか動きません

■ これを右側の腕も GUI で動かせるように修正してください



課題のアップロード

- 作成したプログラムの実行結果のスクリーンショットを撮って 7-3.png というファイル名で保存し、Moodle の第7回課題にアップロードしてください
 - GUI を操作してロボットの腕の姿勢を初期状態から変えてください
 マウスを使ってウィンドウ内に両腕が全部収まるようにしてくだ さい
- Uースプログラム of App.h と of App.cpp を Moodle の第7回
 課題にアップロードしてください
 - ■課題7-2にGUIを追加しただけなので上書きしてかまいません