





#### プログラミングでメディアを取り扱う メディアとは media (メディア)はmedium (メディウム)の複数形 すなわち中間にあるもの・間に入って媒介するものたち コミュニケーションにおけるメディア 空間 ■ 音, 音声, 音楽 デザイン ■ 画像, 映像 ■ デザイン プロダクト 人と人の間にあるもの メッセージ



## プログラミングとは

あなたがすべきことがどこかに書いてあるわけではない

### プログラムするということ・序

- ■課題を解決する手順や方法を考える
  - つまりアルゴリズムを考える
    - ■問題を解く<u>手順</u>を定式化したもの
    - 算法ともいう
- コンピュータが取り扱い可能な形式で記述する
  - つまり**ソースプログラム**を作成する
    - アルゴリズムをもとに問題を解く<u>手順を記述</u>したもの
       原始プログラム、ソースコードともいう
    - かつては算譜という和製漢語が用いられていたが廃れた

## プログラムするということ・破

# 実行したプログラムが正しく動作するか確かめる つまりテストする

- プログラムの機能が期待通り動作するかどうか
- プログラムの性能が期待通り得られるかどうか
- プログラムが悪意のある操作によって<u>予想外の動作</u>をしないか
- プログラムが正しく動作するよう修正する

#### ■ つまり**デバッグ**する

プログラムが正しく動作しない原因や理由を見つける
 プログラムを正しく動作させる修正方法を考える

### プログラムするということ・急

■まず課題を分析して正しい設計をする

- それでも書いたプログラムはたいてい正しく動かない
- 正しく動かない理由もたいてい分からない
- どから正しく動くようになるまで試行錯誤する

## 試行錯誤できないとプログラミングできない

### プログラミングの学習

 教科書を読んだだけでは書けるようにならない ■ 文法を学んでも目的を達成する方法は分からない プログラムは書かないと書けるようにならない プログラミングが苦手という人のパラドックス 書けないのに書くことはできないと思う 最初に何をしたらいいのか分からない まず何のため(目的)に何をする(処理)のか決める プログラミングは処理の手順(手続き)を考えること

### 「プログラミング的思考」





- 弾道ミサイルを目標地点に到達 させたい
- 考えること
  - 既知のことは何か
  - 未知なことは何か
  - 求めることは何か



分析

- 弾道ミサイルの速度は v
- 目標地点までの距離は d
- 重力加速度は g
- 条件
  - 空気抵抗は考慮しない
  - 自分で速度を変えない
  - 自分で向きを変えない
- 発射角 *θ* を求める



## モデル化

- 弾道ミサイルの軌跡
  - 軌跡は放物線を描く
  - 時刻を t とする

 $x = vt\cos\theta$  $y = vt\sin\theta - \frac{1}{2}gt^2$ 





#### ■ 目標地点に到達したとき

$$x = vt\cos\theta = d$$
$$y = vt\sin\theta - \frac{1}{2}gt^2 = 0$$





 xの式より ■ (2cos<sup>2</sup> θ)/d 倍して移項  $2\sin\theta\cos\theta = \frac{gd}{m^2}$  $t = \frac{a}{v\cos\theta}$  yの式のtに代入して 二倍角の公式より  $\frac{d\sin\theta}{\cos\theta} - \frac{gd^2}{2\nu^2\cos^2\theta} = 0$  $\sin 2\theta = \frac{ga}{n^2}$  $\theta = \frac{1}{2} \sin^{-1} \frac{gd}{n^2}$ 

これをプログラムで計算するというこ とを**決めて**ようやくコーディング

実装(コーディング)

ソースプログラム

#include <iostream>
#include <cmath>
using namespace std;
int main()

// 重力, 速度, 距離 double gravity, velocity, distance;

```
// パラメータの入力
cout << "重力加速度:";
cin >> gravity;
cout << "発射速度:";
cin >> velocity;
cout << "目標までの距離:";
cin >> distance;
```

```
// パラメータのチェック
if (gravity <= 0.0) {
    cout << "重力が小さすぎます¥n";
    return 1;
}</pre>
```





#### ソースプログラムの記述に用いる言語 ■ 言語なので**文法**が決められている 文法に違反した記述はエラーになる ソースプログラムは人間が記述する ■ コンピュータが理解可能な機械語への翻訳が必要 ■ 逐次的に解釈しながら実行 インタプリタ方式

- 一括して翻訳した後に実行
  - コンパイラ方式

### インタプリタ方式とコンパイラ方式



## ソースプログラムのコンパイル

コンパイル

(一括翻訳)

#### C++ 言語のソースプログラム

// 関数 add の定義 // int add(int x, int y) { // 変数 z の宣言 int z;

}

// 引数 x と引数 y を足して z に代入する z = x + y;

// z を関数の戻り値 (関数の値) とする return z;



#### add: pushq %rbp %rsp, %rbp movq \$16, %rsp subq %ecx, 16(%rbp) movl movl %edx, 24(%rbp) 16(%rbp), %edx movl 24(%rbp), %eax movl %edx, %eax addl movl %eax, -4(%rbp) movl -4(%rbp), %eax \$16, %rsp addq %rbp popq ret アセンブリ言語は数値の羅列である 機械語の命令を文字による簡略表記 (ニーモニック) で表したもの

## インタプリタ方式のプログラミング言語

Python, Ruby, Perl, JavaScript, PHP, Prolog, ...

- 軽量言語 (Light Weight Language, LL), スクリプト言語
- Web や機械学習などサービス開発で利用されるものが多い
- 特徴
  - CPU によって実行されるのはインタプリタ
  - インタプリタがソースプログラムを解釈しながら実行する
  - コンパイル(翻訳)の必要がない

## コンパイラ方式のプログラミング言語

 C, C++, Objective-C, Swift, Go, Fortran, COBOL, ...
 OS やアプリケーションの開発に使用されるものが多い
 他のプログラミング言語の処理系の開発にも使用される
 C 言語は C 言語で開発されているが Python (のリファレンス実装の一つ) CPython は C 言語で開発されている

#### ■ 特徴

CPU は機械語のオブジェクトプログラムを直接実行する
 実行前にあらかじめコンパイル(翻訳)する必要がある
 実行時にソースプログラムの解釈を行わないため高速

### 中間言語方式のプログラミング言語

コンパイラを使って中間言語に翻訳

■ 特徴

C#, Java, Kotlin, Scala, Pascal, Smalltalk, ...
 近年のアプリケーション開発に用いられることが多い

JIT (Just In Time) コンパイラ により解釈しながら機械語 への変換を行うものが主流

- 中間言語のプログラムをインタプリタによって実行する
- インタプリタ方式とコンパイラ方式の中間的な特徴をもつ
- コンパイルが速く実行速度もコンパイル方式に迫る
- 中間言語のインタプリタが用意された多様な環境で使える

## プログラミングパラダイム

命令型プログラミング言語

 手続き型プログラミング言語
 C, C++, C#, Java, JavaScript など大半のプログラミング言語

 宣言型プログラミング言語

 関数型プログラミング言語
 LISP, Ocaml, Erlang, Scala, Haskell, F#

Lisp, Ocalin, Enang, Scala, Haskell,
 論理型プログラミング言語

Prolog, GHC

ハラタイム ある時代や分野において 支配的規範となる 「物の見方や捉え方」

## オブジェクト指向

#### プログラミングパラダイムの一つ

- 命令型・宣言型の分類などとは直交する概念
- C++, C#, Java, Python など近年のプログラミング言語で採用

#### ■ オブジェクト

- 何らかの役割を与えられた実体(メモリ)
- ■長くなるので細かい話は**割愛** 
  - 別に「オブジェクト指向」という講義がある(くらい)
  - この演習では便利な機能として利用するだけ

C++ 言語によるプログラミング

C++ 言語は初心者に向いていないらしい

### パソコンの外観と内部







#### CPU とメモリ



(前ページのマザーボードのものとは異なります)

(前ページのマザーボードに取り付けられていたものです)

## ビデオカード (GPU)







#### CPU とメモリの関係



#### メモリを使うには変数を宣言する





データ型をあまり気にしないように見える言語もある
 データ型という考え方が初学者には難しいと思われていたりする
 Python, JavaScript, Perl, PHP, ...

■ でもデータ型を明示しないだけで気を付けていないとハマる



整数型		実数型	
char	■8ビット精度符号付き整数 ■-128~127	float	■32ビット単精度浮動小数点 ■精度は10進で約6.92桁
int Iong	■ 32ビット精度符号付き整数 ■ -2,147,483,648~2,147,483,647	double	<ul> <li>■ 64ビット倍精度浮動小数点</li> <li>■ 精度は10進で約15.65桁</li> </ul>

#### 整数型のバリエーション

signed char	charと同じ
unsigned char	8ビット符号無し整数,0~255
short, short int, signed short int	16ビット符号付き整数,-32,768~32,767
unsigned short, unsigned short int	16ビット符号無し整数,0~65,535
signed, signed int, long, long int, signed long int	intと同じ
unsigned, unsigned int, unsigned long, unsigned long int	32ビット符号無し整数,0~4,294,967,2965

#### そのほかのデータ型(基本)

bool	論理型, true (真) と false (偽) の値だけを持つ
enum	列挙型,名前を付けた定数の組のうちのどれか一つの値を持つ
long long, signed long long	64ビット精度符号付き整数, -9,223,372,036,854,775,808~9,223,372,036,854,775,807
unsigned long long	64ビット精度符号無し整数, 0~18,446,744,073,709,551,615
long double	double と同じ

#### int と long が同じ理由

初期のパソコンの CPU は 8 ビットで、それが 16 ビット、32 ビット、64 ビットと発展してきました。C++ 言語のもとに なった C 言語がパソコンで使われ始めたのは 16 ビットの頃 だったため、その int は 16 ビット、long は 32 ビットになっ ていました。パソコンの CPU が 32 ビット化したことで C 言 語の int にも 32 ビット割り当てられるようになりましたが、 long は互換性のために 32 ビットのままに据え置かれました。 CPU が 64 ビット化したときも互換性のために int、long とも に 32 ビットに据え置かれました。なお、64 ビットの整数を 扱う場合は long long、unsigned long long が使用されます。

#### long double と double が同じ理由

これは Visual Studio の C++ 言語処理系である Visual C++ の仕様で(int と long の関係も Visual C と x86 系の CPU に依存した話)、他の言語処理系では同じ x86 系でも long double が80 ビットになっているものもあります。これは初期の x86 系 CPU にオプションで追加する浮動小数点演算八ードウェアが内部的に80 ビットで計算していたのをそのまま使えるようにするためですが、データサイズは128 ビットとなり48 ビット無駄に使います。128 ビットをフルに使い切る精度の実数データを扱えるように考えられてはいるものの、現時点では実装がまちまちの状態になっているようです。

## くどいようだが整数型と実数型がある

#### 整数型(char, int, long 等)

- (主に)数を数えるのに使う
  - 小数点以下を持たない
  - ■隣り合う数値の間隔は1
  - 除算は小数部が切り捨てられる
    - **5** / 3 == 1
    - -5 / 3 == -1
    - 1/2 == 0
  - 実数と混在するときは実数になる
    - 5 / 3.0f ≒ 1.66667
    - 1.0f / 2 == 0.5f

#### 実数型(float, double 等)

- 数値計算に使う
  - 小数点以下を持つ
  - 隣り合う数値の間隔が一定でない
    - 0.1f は実は 0.1 の近似値
  - 非常に大きな数や非常に小さな数 が表現できる
  - 指数表記が可能
    - 1.23 × 10<sup>5</sup> は 1.23e5f
    - 末尾にfが付いているものはfloat型の 定数、無ければ double 型の定数

#### データを記憶するには変数に代入する



#### 手続き(処理)は関数として記述する

■ 数学 ■ 関数を定義する *f*(*x*) = *x*<sup>2</sup>

関数を評価する
 y = f(2)



```
// 関数名が f の関数の定義
double f(double x)
 // 関数の戻り値として引数 x の二乗を返す
 return x * x;
int main()
 double y;
 // 関数 f を評価した値を y に代入する
 y = f(2.0);
  これは y に f(2.0) の値を
  代入(格納)するという意味
```
#### 関数の定義と関数の呼出し





#### "//"より右はコメント



#### 最初に評価される関数 main()

```
関数名が f の関数の定義
double f(double x)
í
   関数の戻り値として引数 x の二乗を返す
 return x * x;
}
/ メインプログラム
int main()
 // 変数宣言
 double y;
 // 関数 f を評価した値を y に代入する
  = f(2.0);
}
```

- プログラムは main() 関数から実 行を開始する
- main()から関数 f()が呼び出される
- 関数 f() が実行された後 main()
   に戻る
- main()のreturn 文を省略すると
   0を返す



Visuall Studio による C++ プログラム作成

#### Visual Studio を起動する



# 「新しいプロジェクトの作成 (N)」を選ぶ

- 🗆 🗙

#### Visual Studio 2019 開始する 最近開いた項目(R) Visual Studio を使用するとき、ユーザーが開くプロジェクト、フォルダー、ファイルはここに表示されるので、すばやくアク リポジトリのクローン(C) セスできます。 GitHub や Azure DevOps などのオンライン リポジトリか 頻繁に開く項目は、ピン留めして常に一覧の先頭に表示することができます。 らコードを取得します プロジェクトやソリューションを開く(P) ମ୍ବି କି ローカルの Visual Studio プロジェクトまたは .sln ファイル を開きます ローカル フォルダーを開く(F) 2 任意のフォルダー内のコードに移動して編集します \* 新しいプロジェクトの作成(N) 開始するには、コードスキャフォールディング付きのプロジェ クトテンプレートを選択します コードなしで続行(W) →

## 「空のプロジェクト」を作成する

新しいプロジェクトの作成	「空の」で検索すれば見つけやすい - · · · · · · · · · · · · · · · · · ·
最近使用したプロジェクト テンプレート(R)	すべての言語(L) ▼ すべてのプラットフォーム(P) ▼ すべてのプロジェクトの種類 ▼
最近アクセスしたテンプレートの一覧は、ここに表示されます。	200 Django Web プロジェクト Django プロジェクト作成用のプロジェクト Linux macOS Python Web Windows
	空のプロジェクト Windows 用に C++ で最初から始めます。開始ファイルを提供しません。 コンソール C++ Windows
	空白のアプリ(ユニバーサル Windows) 定義済みのコントロールまたはレイアウトのない単一ページ ユニバーサル Windows プラットフォーム (UWP) アプリ用のプロジェクト。 C# デスクトップ UWP Windows Xbox XAML
	▼「 空白のアプリ (ユニバーサル Windows) 定義済みのコントロールまたはレイアウトのない単一ページ ユニバーサル Windows プラットフォーム (UWP) アプリ用のプロジェクト。
	マのソリューション プロジェクトを含まない空のソリューションを作成します。 その他
	戻る(B) 次へ(N)

# 「プロジェクト名 (N)」を入力する

- 🗆 🗙

#### 新しいプロジェクトを構成します

空のプロジェクト コンソール C++ Windows
<sup>カジェクト</sup> 名は適当に決めてください
sample
場所(L)
C:¥Users¥tokoi¥source¥repos •
ソリューション名(M) <b>()</b>
sample
ロッリュ-ションとプロジェクトを同じディレクトリに配置する(D) プロジェクトを一つしか作らないときはチェックを入れても大丈夫
戻る(B) 作成(C)

## 新しいプロジェクトが作成される

ズ ファイル(E) 編集(E) 表示(V) プロジェクト(P) ビルド(	<u>B)</u> デバッグ( <u>D</u> ) テスト( <u>S</u> ) タ	分析( <u>№</u> ) ツール( <u>T</u> ) 拡張機能( <u>X</u> )	ウィンドウ( <u>W)</u> ヘルプ( <u>H</u> ) 検索 (Ctrl+Q) 🖌	sample — 🗆	$\times$
🖁 © - ©   惣 - 🍅 💾 📽   ウ - ペ -   Debug -	×86 - Dーカル	レ Windows デバッガー マ 自動	•   🏓   🖾 📮	🖻 Live Share	Ŕ
ソリューション エクスプローラー       ・ 単 ×         ○ ○ △ 部・       ○ ・ □ 回 ▶ ●         ソリューション エクスプローラーの検索 (Ctrl+:)       ♪ ・         □ ソリューション 'sample' (1/1 プロジェクト)       ▲         ▲ 国 sample       ▶ ●●         ▶ ●●       参照         ● ♪ ●       小部依存関係         ● ソース ファイル         ■ リソース ファイル					サーバー エクスプローラー ツールボックス プロパティ
	出力			<b>▼</b> [	ļΧ
	出力元( <u>S</u> ):		·   &   & &   X =   20		- 64
<u> ソリューション エ</u> クラス ビュー プロパティ マネ チーム エクスプ					
□ 準備完了				↑ ソース管理に追加 🔺	<b>.</b>

## 「新しい項目の追加(W)」を選ぶ

📢 ファイル(F) 編集(E) 表示(V)	プロ	コジェクト(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)	検索 (Ctrl+Q)	P	sample	—		$\times$
0 - 0   🎦 - 🏠 💾 🚰   🐓	<b>↑</b>	ソリューションの再ターゲッ	٢		do	ows デバッガー	▼ 自動	•	🏓 🖾 🖕				🖄 Live	Share	ନ୍ଦ
	**	クラスの追加(C)							, -						4
ション エクスノローフー	B.	クラス ウィザード(Z)		Ctrl+Sh	nift+X										-7
	1 74;	リソースの追加(R)													÷1
ソリューション エクスプローラー の検索 (Ctrl+;)	1	新しい項目の追加(W)		Ctrl+Sh	nift+A										ילגי
フリューション 'sample' (1/1 プロジェク)	10	既存の項目の追加(G)		Shift+A	Alt+A										
▲ w sample	*	新しいフィルター(F)													1
▶ 外部依存関係	ß	すべてのファイルを表示(C	))												y−J/7
📒 ソース ファイル ━ ヘッダー ファイル		プロジェクトのアンロード(L)	)												ドック
= リソース ファイル		ソリューションの再スキャン	(S)												R L
		参照データベース エラーを	表示												לםול
		参照データベース エラーを	クリア											- 1 \	र्भ
		参照の追加(R)						- 5 4	⇒_ X=   a	10 m				* * *	
	t∌	接続済みサービスの追加	(C)					•   =   =		с <b>н</b>					-
	ø	スタートアップ プロジェクト	に設定(A)												
		ビルドのカスタマイズ(B)													
		テンプレートのエクスポート	(E)												
	Ħ	NuGet パッケージの管理	(N)												
	بو	プロパティ(P)		Alt+F7											
<u>ソリューション エ</u> クラス ビュー プロパティ	6	エクスプローラーでフォルダー	-を開く(X)												
□ 準備完了	-											↑ ソ-	-ス管理に追加	n 🔺 🖣	<u>ا</u> ر ا

## 「C++ ファイル (.cpp)」を追加する

新しい項目の追加 - sam	ple					?	×
◢ インストール済み		並べ替え: 既定	• # E		検索 (Ctrl+E)		ρ-
✓ Visual C++ コード 者式設定 ATL データ リソース Web ユーティリティ プロパティシート HLSL Test グラフィックス		<ul> <li>C++ ファイル (.cpp</li> <li>小ッダー ファイル (.h</li> <li>・</li> <li>・</li></ul>	o) I)	Visual C++ Visual C++ Visual C++	種類: Visual C++ 1203		
▶ オンライン	、ファ	イル名を半角	英数字で指定	する			
名前(N): 場所(L):	main C:¥Users¥tokoi¥sour	ce¥repos¥sample¥sample¥		<b></b>	参照(B) 追加(A)	++>>t	JU

# ソースプログラムの編集

▶ ファイル(E) 編集(E) 表示(Y) プロジェクト(P) ビルド(B)	<u>B</u> ) デバッグ( <u>D</u> ) テスト( <u>S</u> ) 分析( <u>N</u> ) ツール( <u>T</u> ) 拡張機能( <u>X</u> ) ウィンドウ( <u>W</u> ) ヘルプ( <u>H</u> ) 検索 (Ctrl+Q) <b>A</b> sample — □ ×
🕺 😋 🕶 📄 📸 🕶 🔛 🔐 🦻 🝷 🦿 🖓 🕶 🖓 🕶	x86 🔹 🕨 ローカル Windows デバッガー 🔹 自動 🔹 🐂 🔎 🖾 🥫 🔚 🌾 📄 🦉 📕 🌂 🌂 🦉 🖕 Live Share 🔗
ソリューション エクスプローラー       マロ・ちの       ロロロロ       マロ・ちの         ソリューション エクスプローラーの検索(Ctrl+:)       ア・         マリリューション 'sample' (1/1 プロジェクト)         ・       ●         ●       ●         ●       ●         ● <td>main.cpp* + ×       ・ (グローバル スコーカ)         1       int main()         2       .         3       .         4       .         CCICVースプログラムを書く       .</td>	main.cpp* + ×       ・ (グローバル スコーカ)         1       int main()         2       .         3       .         4       .         CCICVースプログラムを書く       .
- UV-X JP1JV	100% • ② 問題は見つかりませんでした · 行:4 文字:1 SPC CRLF
	出力
	出力元(5): -   2-  14 14 14 14 14 14 14 14 14 14 14 14 14
<u>ソリューション エ クラス ビュー プロパティ マネ チーム エクスプ</u>	
□ 準備完了	



<pre>int main()</pre>
A second
}

■ main() **関数**を定義する

- このプログラムには中身がないの で何もしない
- main() 関数の戻り値のデータ型は int 型
- main() 関数に限り return を省略 しても戻り値として 0 を返す

# プログラムのビルドと実行



# プログラムのコンソール出力



#### コンソールに文字を出力する

#### #include <iostream>

int main()

```
std::cout << "hello, world¥n";</pre>
```

#### ■ <mark>マーカー</mark>部分を追加する

- #include <iostream>
  - iostream という標準ライブラリの定義をソースプログラムのこの部分に埋め込む
  - ■標準ライブラリは C++ 言語に最初から用意されている機能
- std::cout << "hello, world¥n";</pre>
  - コンソールに hello, world を表示
  - std::cout は標準ライブラリに含 まれるコンソール出力の機能

# 修正したプログラムのビルドと実行

ズ ファイル(F) 編集(E) 表示(V) 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) 検索 (Ctrl+Q) 🔎	sample	- 0	$\times$
🍈 🗢 🗢 🗎 🏜 🗳 🗳 🖓 🗸 🖓 🕶 🕨 Debug 🕞	x86 - トローカル Windows デバッガー - 自動 - 「声」 🖾 🚽 🔚 🌾 🌾 🦉 🖉	ા લાગા ગા	🗧 📝 Live Share	ጽ
ソリューション エクスプローラー       ・ 早 ×         ○ ○ 合 品・       ○ ・ 与 回 回 ◇          ソリューション エクスプローラーの検索 (Ctrl+:)       ・         ソリューション 'sample' (1/1 プロジェクト)         ▲ 雪 sample         ▶ ••• 参照         ▶ ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ● □         ● ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ● ••• ●         ●         ●         ●         ●         ●         ●         ●         ●         ●         ●         ●         ●         ●	main.cpp ⇒ × Sample (グローバルスコープ) ・ int main() f std::cout << "hello, world¥n"; 7			サーバーエクスプローラー ツールボックス プロ
- リソース ファイル - リソース ファイル ソリューション エ クラス ビュー プロパティ マネ チーム エクスプ	<ul> <li>100% ● 問題は見つかりませんでした</li> <li>出力元(S): デバッグ ● 金 金 空 認</li> <li>sample.exe (Wins2): U:#Windows#SysWUWb4#vcruntime1400.dii か読み込まれました。</li> <li>'sample.exe' (Wins2): 'C:#Windows#SysWUWb4#vcruntime1400.dii が読み込まれました。</li> <li>スレッド 0x59c0 はコード 0 (0x0) で終了しました。</li> <li>'sample.exe' (Wins2): 'C:#Windows#SysWUWb4#kernel.appcore.dll' が読み込まれました。</li> </ul>	▶ 行:7 :	文字:1 SPC CF ▼ ↓	U/(Fr
□ 準備完了		<b>↑</b> צי	−ス管理に追加 🔺	<b>ب</b>

# 修正したプログラムのコンソール出力

#### Microsoft Visual。コンソールに出力した文字

hello, world

C:¥Users¥tokoi¥source¥repos¥sample¥Debug¥sample.exe(ブロセス 15488)は、コード 0 で終了しました。 デバッグが停止したときに自動的にコンソールを閉じるには、[ツール] -> [オブション] -> [デバッグ] -> [デバッグの停止時に自 動的にコンソールを閉じる] を有効にします。 このウィンドウを閉じるには、任意のキーを押してください...

×





#### ■ <mark>マーカー</mark>部分を追加する

- double f(double x)
  - 戻り値のデータ型 double の関数 f() を定義する
  - 仮引数 x のデータ型は double
  - 戻り値として x \* x を返す
- y = f(2.0);
  - 実引数に 2.0 を指定して関数 f() を呼び出す
  - 関数 f()の戻り値は変数 y に代入 する

# ブレークポイントを設定してビルドと実行



## 一時停止したらステップインする

<ul> <li>ステイル(E) 編集(E) 表示(M) プロジェクト(P) ビルド(B)</li> <li>マ マ ○   裕 - 論 単 単   ウ - ペ -   Debug - x</li> <li>プロセス: [22904] sample.exe</li> </ul>	デバッグ(D)       テスト(S)       分析(N)       ツール(D)       拡張機能(X)       ウィンドウ(W)       ヘルプ(H)       検索 (Ctrl+Q)       タ       sample       —       □         36 <ul> <li> <li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></ul>	× Æ
ソリューション エクスプローラー       ・ ↓ ×         ○ ○ 合 部 ・ ○ ・ ち 回 回 ◇ ♪ -         ソリューション エクスプローラーの検索 (Ctrl+:)         □ ソリューション 'sample' (1/1 プロジェクト)         ▲ 国 sample         ▶ ••• 参照         ▶ ••• 参照         ▶ ••• 参照         ▶ ••• 参照         ▶ ••• ● **         ▶ ••• ● **         ▶ ••• ● **         ▶ ••• ● **         ▶ ••• ● **         ▶ ••• ● **	main.cpp +# ×       ◆       参断ツール         Sample       (グローバルスコープ)       ② main()       ③       〇	<ul> <li>₽</li> <li>₽</li> <li>×</li> <li>×</li></ul>
■ ヘッシー ファイル ■ リソース ファイル 時停止している	100%     ● 問題は見つかりませんでした     ● 行:12 文字:1 SPC CRLF     ●● イベントの表示 (1の1)       自動     ● 早 ×     呼び出し履歴       検索 (Ctrl+E)     ● ● ← ●     ***       タカ     体     毎#5	▼
実行中は赤い		不
	日動 ローカル スレット モジュール ジォッチ 1 「「「「「」」」」 「「「」」」」 「「」」」 「「」」」 「「」」」」 「「」」」」 「」」 「」」 「」」」 「」」 「」」」 「」」 「」」」 「」」 「」」」 「」」 「」」」 「」」 「」」」 「」」 「」」」 「」 「	La

## 関数 f() に移動するのでステップオーバー

<ul> <li>✓ ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(E)</li> <li>◎ • ○   裕 • △ 単 単   ウ • ペ •   Debug • [</li> <li>⑦ プロセス: [22904] sample.exe</li> <li>▼ 〒 ライフサイク.</li> </ul>	) 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) 検索 (Ctrl+Q) <b>P</b> sample - □ (86 ・ ▶ 続行(C) ・ 自動 ・   ■   圖 <sub>-</sub> □ <b>・</b> ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	× Æ
yリューション エクスプローラー ・ ↓ × ・ ↓ * ・ ↓ *	main.cpp +> ×       ・       ②       診断ツール            Sample        ・       (グローバルスコープ)         ・       ③       「       ④       □       ①       ②       ①       ①       ②       ①       ③       ⑤       ④       ④       ④       ④       ④       ④       ④       ④       ④       ④       ④       ④       ④       ④       ④       ④       ●       ●       <	• # ×
₩12-5/27 T/771-5- 757 ビュ-	自動	▼
□ 準備完了	▲ <u>10 10 10 10 10 10 10 10 10 10 10 10 10 1</u>	<b>.</b>

# 次の行に進んだらステップオーバー

<ul> <li>マァイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B)</li> <li>○ • ○   浴 • △ 単 単   ウ • ペ •   Debug • x</li> <li>プロセス: [22904] sample.exe</li> <li>▼ ライフサイクル</li> </ul>	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) 検索 (Ctrl+Q) ♀ sample - □ 56	× Æ
ソリューション エクスプローラー       ・ ち 回 回 い チ         ソリューション エクスプローラーの検索 (Ctrl+:)       ク・         ソリューション 'sample' (1/1 プロジェクト)       ・         ・・       参照         ・・       参照         ・・       参照         ・・       参照         ・       ・         ・ <td< td=""><td>main.cpp +&gt; ×       ・ (グローパルスコーカ) ・ © f(double x)         1       #include <iostream>         2       edouble f (double x)         3       edouble f (double x)         4       {         1       return x * x; sizUW###         1       int main()         3       edouble y;         10          10</iostream></td><td>▼ 平 × ▲ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓</td></td<>	main.cpp +> ×       ・ (グローパルスコーカ) ・ © f(double x)         1       #include <iostream>         2       edouble f (double x)         3       edouble f (double x)         4       {         1       return x * x; sizUW###         1       int main()         3       edouble y;         10          10</iostream>	▼ 平 × ▲ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
ソリューション エクスプローラー クラス ビュー	自動 ローカル スレッド モジュール ウォッチ1 呼び出し履歴 ブレークポイント 例外設定 出力 ↑ ソース管理に追加 ヘ	<b>↓</b>

# もう一度ステップオーバー

マアイル(F)         編集(E)         表示(V)         プロジェクト(P)         ビルド(B)           ● <td< th=""><th>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) 検索 (Ctrl+Q) P sample — □ 86  ► 続行(C) モ 自動  ►   戸   図 - □ = 〇 → * ○ → * ○ □   液 - ◎ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □</th><th>× &amp;</th></td<>	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) 検索 (Ctrl+Q) P sample — □ 86  ► 続行(C) モ 自動  ►   戸   図 - □ = 〇 → * ○ → * ○ □   液 - ◎ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	× &
> ソリューション エクスプローラー マ 単 × C O 合 首 ・ で ち 回 回 ◇ チ ー ソリューション エクスプローラーの検索 (Ctrl+:) マ マ マ × マ リリューション 'sample' (1/1 プロジェクト) ▲ Sample ▶ ・■ 参照 ▶ ● ● ◆ ● ◆ ● ◆ ● ◆ ● ◆ ● ◆ ● ◆ ● ◆ ● ◆	main.cpp キン     マローバルスコープ)     ゆ f(double x)     びローバルスコープ)     ゆ f(double x)       1     #include <iostream>     *     *       2     3     曰double f (double x)     *     *       3     □double f (double x)     *     *       4     「     *     *       5     □     ●     ●       6     □     *     *       10     □     □     ●</iostream>	<ul> <li>₽ ×</li> <li>^</li> <li></li> <li><!--</td--></li></ul>
	100%       ● 問題は見つかりませんでした       ● 行:6 文字:1 SPC CRLF       ●● イベントの表示 (4 の 4)         自動       ● 早 ×       ● び出し履歴         検索 (Ctrl+E)       ● ◆ ◆ ◆       ●         名前       値       種類         ● x       2.000000000000000000000000000000000000	▼ 早 × 言語 ▲ C++ C++ 不
□ 準備完了	▲ 2.3 - 7.6 4 0 km 2 m 2 m 2 m 2 m 2 m 2 m 2 m 2 m 2 m	🐥 a

## main() 関数の f() の呼び出し位置に戻る



# ステップオーバーするとyに代入される

<ul> <li>✓ ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B</li> <li>◎ • ○   裕 • △ 単 単   ウ • ペ •   Debug • 2</li> <li>⑦ プロセス: [22904] sample.exe</li> <li>▼ 図 ライフサイクル</li> </ul>	) 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) 検索 (Ctrl+Q) <b>P</b> sample - □ 86  ► 続行(C) ► 自動 <b>F</b>   Ø <sub>=</sub> □ ■ <b>O</b>   → * <b>?</b>   Ø <sub>=</sub> □ □ □ V イベント ► スレッド: [20952] メイン スレッド <b>マ マ マ マ マ</b> マ マ マ マ マ マ マ マ マ マ マ マ マ マ	× Æ
ソリューション エクスプローラー       ・ 中 ×         ○ ○ 合 首・ ○ ・ ち 回 回 ◇ ♪ -         ソリューション エクスプローラーの検索 (Ctrl+:)         ○ ソリューション 'sample' (1/1 プロジェクト)         ▲ 雪 sample         ▶ ••■ 参照         ▶ ••■ 参照	main.cpp +p ×       ・ (グローバルスコープ) ・ (P main())       診断ツール	• # × ^ ~
	100%       ● 問題は見っかりませんでした       ◆ 行:13 文字:1 SPC CRLF       ●● イベントの表示 (6 の 6)         自動       ● 平 ×       ● び出し履歴         検索 (Ctrl+E)       ● - ← →       ●         名前       値       種類         ● y       4.0000000000000       double         変数 y の内容が 4.0 になった       ● の内容が 4.0 になった       ● 取び出し履歴         ● 取び出し 履歴       ブレークボイント 例外設定       中力	▼ 早 × 言語 ▲ C++ 不

## std::cout をステップオーバーする

マアイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(E)         ● • ●          ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	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)       検索 (Ctrl+Q)       ク       sample       ー       □         86 <ul> <li> <li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></ul>	× R
ソリューション エクスプローラーの検索(Ctrl+:)       ク・         ソリューション エクスプローラーの検索(Ctrl+:)       ク・         ソリューション 'sample' (1/1 プロジェクト)       ・         ・       ●	main.cpp + x       ・ な	<ul> <li>↓ ♀ ×</li> <li>▲</li> <li>↓</li> <li>↓</li></ul>
ソリューション エクスプローラー     クラス ビュー       ロ     準備完了	自動 ローカル スレッド モジュール ウォッチ1 呼び出し履歴 ブレークポイント 例外設定 出力	÷.

#### コンソールに出力される

C:¥Users¥tokoi¥source¥repos¥sample¥Debug¥sample.exe

×

#### 「続行」すると残りを一気に実行する

🔀 ファイル(F) 編集(E) 表示(V) 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) 検索 (Ctrl+Q) 🔎 sample	- 🗆 ×
🕴 😋 🕶 🕲 🖆 💾 🔐 🤔 🤊 🗸 🖓 🚽 Debug 🕞	86 ▼ ▶ 続行(C) ▼ 自動		🖻 Live Share 🛛 🛱
プロセス: [22904] sample.exe   ▼  ぼ ライフサイク.	レイベント - スレッド: [209_] メインフレッド 続行	・ * * 打ち切りたいと	
ソリューション エクスプローラー - マーマーズ マーマーズ	main.cpp + X	きは強制終了 (新ツール	<b>-</b> ₽×
	Bel sample - (クロ−/៶)ルス		~
ソリューション エクスプローラー の検索 (Ctrl+:)	8 ⊟int main() 9  {	* 診断セッション: 0 秒 (37 ミリ秒 選択)	済み)
□ ソリューション 'sample' (1/1 プロジェクト)	10 double y;	36ミリカシ	
▲ sample ▶ ••■ 参照	• 12 y = f(2.0);		¥
▶ 💼 外部依存関係	<ul> <li>◆ 14</li> <li>▲ 14</li> </ul>	prid#n; 概要 イベント メモリ使用量 CPU	J 使用率
▲ ニュース ファイル ▶ ++ main.cop	15	イベント	^
<u>≓</u> ヘッダー ファイル	100 % - ◎ 問題は見つかりませんでした	行:14 文字:1 SPC CRLF ∞∞ イベントの表示 (7 の 7)	~
🐖 リソース ファイル		▼ 平 × 呼び出し履歴	<b>-</b> ₽ ×
	検索 (Ctrl+E) ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・	> '' 名前	<b>言語</b> ▲
	名前值	種類 ▲ Sample.exe!main() 行 14	C++
	e std::operator< {}	std::basic kernel32.dll![下のフレームは間違っているか、または見つ	つかりません… 不…
	y 4.000000000000000000000000000000000000	double	
ソリューション エクスプローラー クラス ビュー	自動 ローカル スレッド モジュール ウォッチ 1	呼び出し履歴 ブレークポイント 例外設定 出力	
□ 準備完了			理に追加 🔺 🌲 🔚

#### プログラムの実行が終了する

亟 Microsoft Visual Studio デバッグ コンソール

hello, world

C:¥Users¥tokoi¥source¥repos¥sample¥Debug¥sample.exe(ブロセス 15488)は、コード 0 で終了しました。 デバッグが停止したときに自動的にコンソールを閉じるには、[ツール] -> [オブション] -> [デバッグ] -> [デバッグの停止時に自 動的にコンソールを閉じる] を有効にします。 このウィンドウを閉じるには、任意のキーを押してください...

#### yをコンソールに出力する

```
#include <iostream>
```

```
double f(double x)
```

```
return x * x;
```

```
}
```

}

```
int main()
```

```
double y;
```

```
y = f(2.0);
std::cout << "hello, world¥n" << y;</pre>
```

マーカー 部分を追加する ■ 「std::cout << "..." | 全体も std::cout と同じ機能を持つ ■ したがって std::cout << "..." << "…"のように続けて書ける std::cout << y とすると変数 y</li> の値を文字に直してコンソー ルに出力する したがって std::cout << "..." <</li> yとすれば"…"の後ろにyの 値を文字直して出力される

#### hello, world の次に4が出力される

📧 Microsoft Visual Studio デパッグ コンソール bello, world C.¥Users¥tokoi¥source¥repos¥sample¥Debug¥sample.exe(プロセス 19980)は、コード 0 で終了しました。 デバッグが停止したときに自動的にコンソールを閉じるには、[ツール] -> [オプション] -> [デバッグ] -> [デバッグの停止時に自 ンソールを閉じる]を有効にします。 ごのウィンドウを閉じるには、任意のキーを押してください...





#### ■ <mark>マーカー</mark>部分を変更する

#### ■ ¥nは改行文字

- ¥はフォントや文字コードの環境に よっては、と表示される
- 英語圏で用いられていた文字集合 (ASCIIコード)の字形は\だったが 日本の文字集合の規格 JIS X 0201 で は字形を¥にした

#### 改行位置が移動する

📧 Microsoft Visual Studio デパッグ コンソール × squared is 4 C:¥Users¥tokoi¥source¥repos¥sample¥Debug¥sample.exe(ブロセス 3660)は、コード 0 で終了しました。 デバッグが停止したときに自動的にコンソールを閉じるには、[ツール] -> [オブション] -> [デバッグ] -> [デバッグの停止時に自 動的にコンソールを閉じる] を有効にします。 このウィンドウを閉じるには、任意のキーを押してください...

ソフトウェアを作るのは色々大変 ちゃんとしたソフトウェアを作るなら道具から揃えよう

#### メディアを扱うプログラミング

#### プログラミング言語自体はメディアを扱う機能がない

- オペレーティングシステムの機能を呼び出す
  - 音声, 音楽, 画像, 映像, グラフィックス
  - いろんなことができるが複雑で手間がかかることが多い

#### ライブラリを組み合わせる

- 音声入出力,画像入出力,映像入出力,3Dモデル入力
   音声処理,画像処理,映像処理,グラフィックス処理
   物理シミュレーション,フォント処理,…
- 一通り揃えるのは大変
## ミドルウェアを利用する

- ■特定用途向けのソフトウェア開発環境のパッケージ
  - シーングラフ/レンダリングエンジン
    - OpenSceneGraph, OGRE, Delta3D, ...
  - ツールキット
  - <u>openFrameworks</u>, <u>Cinder</u>, <u>P5.js</u>, <u>Three.js</u>, <u>freeglut</u>, <u>GLFW</u>, <u>FLTK</u>, <u>Qt</u>, … ■ プログラミング環境
  - Max/MSP, PureData, SuperCollider, vvvv, TouchDesigner, Processing, ...
     ゲームエンジン
    - Unity, Unreal Engine, CRYENGINE, Lumberyard, OROCHI4, ...
    - Irrlicht Engine, <u>Armory Engine</u>, <u>Godot</u>, <u>Xenko</u>, ...

## クリエーティブコーディング

- 映像・音響による表現のためのプログラミング
  - メディアアート, インタラクティブアート
    - "<u>メディアアートの教科書</u>" (多摩美術大学)
    - 関連企業
      - <u>teamLab</u> (チームラボ)
      - <u>Rhizomatiks</u> (ライゾマティクス)
      - BACKSPACE Productions Inc. (バックスペースプロダクションズインク)
      - <u>Takram</u> (タクラム)
      - MontBlanc Pictures (モンブラン・ピクチャーズ)
      - THE EUGENE Studio (ザ・ユージーン・スタジオ)
      - 株式会社白

### Processing

### ■ プログラミングによる創作を行うためのツール

- 誰でも簡単に視覚表現を行うプログラムを作れる
- MIT Media Lab. の John Maeda の下で "<u>Design by Numbers</u>" の開発に 携わった二人の大学院生 Casey Reas と Ben Fry によりオープンソー スプロジェクトとして開発

#### Java で実装されている

プログラムは Java でも Python でも書ける

プログラミングで絵を描く

## Processingの sketchbook (開発環境)



## iPhone でも動く



📲 UQ mobile 🗢	0:20		<u> </u>	7 💋
<b>〈</b> My Projects	sample		+	
sample.pde				
<pre>void setup()    size(300, }</pre>	{ 300);			
<pre>void draw()     ellipse(1) }</pre>	{ 50, 150,	100,	100)	;
Format Code			Refere	ence



## しかし, この演習では openFrameworks を使います メディアプログラミング演習

### openFrameworks

## クリエーティブコーディングのための C++ 言語によるオープ ンソースのフレームワーク

- 様々なライブラリを統合
  - グラフィクス: <u>OpenGL</u>, <u>GLEW</u>, <u>GLUT</u>, <u>libtess2</u>, <u>cairo</u>
  - オーディオ: <u>rtAudio</u>, <u>PortAudio</u>, <u>OpenAL</u>, <u>Kiss FFT</u> または <u>FMOD</u>
  - フォント: <u>FreeType</u>
  - イメージの読込と保存: <u>FreeImage</u>
  - 動画の再生と取込: <u>Quicktime</u>, <u>GStreamer</u>, <u>videoInput</u>
  - 様々なユーティリティー: <u>Poco</u>
  - コンピュータビジョン: <u>OpenCV</u>
  - 3Dモデルの読み込み:<u>Assimp</u>

## 実は何を使うかすごく悩んだ

#### Processing

■ Java ベースで簡単・処理系も軽い・Python でも書ける

#### Python cgkit

■ Python で 3D CG を扱うパッケージやプラグインを集めた

#### ■ <u>Three.js</u>

■ JavaScript で WebGL を使うグラフィックスライブラリ

#### Unity

■ Unreal Engine と並んでゲーム開発ミドルウェアの標準



## プログラミングを 知ること

## アプリケーション プログラムを 作ること

## openFrameworks の特徴

- openFrameworks は C++ 用のツールキット
  - 既存のライブラリや SDK がそのまま使える
    - openFrameworks 自体はそれらをくっつける糊 (glue)の役割
  - 開発環境には一般のもの (Visual Studio, Xcode など) を使う
    - Processing は Processing 自体が開発環境(なので手軽ではある)
- 同様なものとして,他に <u>Cinder</u> がある
  - ■より新しい機能を使って作られている
    - 他に C# 用の <u>OpenTK</u> というツールキットがある

## なぜ openFrameworks か

- 一般的な開発環境に追加して使える
  - Visual Studio, Xcode, eclipse など
- 他のライブラリや SDK と組み合わせやすい
  - デバイスの SDK は C や C++ で用意されていることが多い
     研究用としても使える?
- C++ ベースである
   プログラミングの初心者には向いていない
  - まあ何とかなるでしょう?

## 課題1-1

openFrameworks のパッケージのダウンロードと展開

## <u>https://openframeworks.cc/ja/</u>を開く



about download documentation learning gallery community development > forum > github > addons > slack > blog > donations English 한국어 简体中文



openFrameworksは創 造的なコーディングの ためのC++のオープン ソースツールキットで す



最新のリリース(0.11.2)の入手と、 openFrameworksを作動させるためのセット アップガイド。

#### ドキュメント

openFramewroksのクラス、関数、アドオン のリファレンス資料。ガイドやチュートリア ルは、tutorialを参照してください。

フォーラム

## Download (寄付募集がポップアップする)

> forum >	aithub > addons > slack > bloa > dona	donations te	English 한국	elopmeni 어 简体中文
<b>0.11.2</b> は最新のリレ イスが追加さ 全に互換性があります。バ openFrameworksを使用するため アップガイドが必要となるでしょ ムに投稿してください。openFra	openFrameworks is developed and maint contributors. Donations help support the development documentation and pay for third party set If you are using openFrameworks comme support openFrameworks development, p project.	ained by several voluntary of openFrameworks, improve the vices needed for the project. ercially or would simply like to please consider donating to the	ーフェ ンと完 セット フォーラ	
OSX	donate	now		
ダウンロード openFrameworks for osx セットアップガイド xcode qt creator (experimental) emscripten	ダウンロード openFrameworks for linux gcc6 or later セットアップガイド linux install qt creator emscripten	ダウンロード openFrameworks for visual studio (2017) qt creator / msys2 32 qt creator / msys2 64 セットアップガイト visual studio qt creator msys2	or 2bit ŀbit ≲	

## 様々なプラットフォームのアプリが作れる

#### OSX / linux / windows に対応している

about	download	documentation	learning	gallery	community
> forum	> github > adda	ons > slack > blog	> donations		English

**0.11.2** は最新のリリースです。これはマイナーバージョンです。様々な新機能や新しいインターフェ イスが追加されています。ですので、このバージョンは0.11.0やさらに新しいバージョンと完 全に互換性があります。バージョン間の違いの一覧は、changelogを参照してください。

openFrameworksを使用するためにはIDE(統合開発環境)が必要です。また、実際に試していくにはプラットフォームごとのセット アップガイドが必要となるでしょう。もしバグをみつけたら問題点のページに投稿してください。その他質問があれば、フォーラ ムに投稿してください。openFrameworksは、MITライセンスで配布されています。

#### スマートフォンや Raspberry Pi にも 対応している

ity	mobile モバイル版のopenFrameworks	ios osx only	android ダウンロード
フェ 注完	は、デスクトップ版と同等の機能 に加えて、加速度系、コンパス、 GPSなど、モバイル端末固有の機 能をサポートしています。	ダウンロード openFrameworks for xcode	openFrameworks for android セットアップガイド android studio
ット ーラ		セットアップガイド <mark>xcode</mark>	
	<b>linux arm</b> Raspberry Pi, Beaglebone (black), Pandaboard, BeagleBoardといっ た、Linuxの作動するARMベース のボードのための openFrameworksです。 セットアップガイドは主要なボー ドのみしか用意されていません が、ARM6かARM7のボードであれ ば作動するはずです。	linux armv6 ダウンロード openFrameworks for linux armv6 セットアップガイド raspberry pi	<b>linux armv7</b> ダウンロード openFrameworks for linux armv7 セットアップガイド pandaboard generic armv7

**OSX** ダウンロード openFrameworks for

OSX

セットアップガイド xcode qt creator (experimental) emscripten **linux** ダウンロード openFrameworks for linux gcc6 or later

#### セットアップガイド linux install qt creator emscripten

windows ダウンロード openFrameworks for visual studio (2017) qt creator / msys2 32bit qt creator / msys2 64bit

セットアップガイド visual studio qt creator msys2

## 使用する PC に合ったものをダウンロード

#### Windows 版のダウンロード

ダウンロード openFrameworks for visual studio (2017) qt creator / msys2 32bit qt creator / msys2 64bit

セットアップガイド

#### macOS 版のダウンロード

OSX ダウンロード openFrameworks for osx セットアップガイド xcode qt creator (experimental)

## ダウンロードした ZIP ファイルを展開

#### ■ 展開先のフォルダの条件

- フォルダのパスに空白文字が含まれていないこと
  - ユーザ名に空白文字を含んでいると「ドキュメント」や「デスクトップ」 のパスに空白文字が含まれることがある
- フォルダのパスに全角文字が含まれていないこと
  - ユーザ名が日本語だと「ドキュメント」や「デスクトップ」のパスに全角 文字が含まれることがある
- 個人所有の PC なら C: ドライブの直下 (C:¥) が確実
  - 条件を満たすことができるなら他の場所でも可

## 展開先として不適当なフォルダの例



"ドキュメント"は実体が "Documents" なので可
"デスクトップ"も実体が "Desktop" なので可
ユーザ名に空白文字を含む場合 ("Taro Yamada" など) 不可
ユーザ名に全角文字を含む場合 ("山田太郎" など) 不可

## ZIP ファイルの中身を C:¥ にコピー

🖊    💆 📙 🗕	展開 ダウンロード	– 🗆 X	🖳 U 🎽 🛄 🛨 U	展開	of_v0.11.0_vs2017_release.zip	– 🗆 X
ファイル ホーム 共有	表示 圧縮フォルダー ツール	~ ( <b>?</b> )	ファイル ホーム 共有 表	示 圧縮フォルダー ツール		¥ 😢
$\leftrightarrow$ $\rightarrow$ $\checkmark$ $\uparrow$ $\downarrow$ $\rightarrow$ PC $\rightarrow$	ダウンロード 🗸 🗟 🔎	ダウンロードの検索	🗲 🔶 👻 🛧 📙 « ダウン >	of_v0.11.0_vs2017_release.zip	• • • • • • • • • • • • • • • • • • •	
- אלשת אינא אינא	今日 (2)			<b>が</b>		圧縮サイズ パスワード保
	🔳 of v0.11.0 osx release.zip 右クリン	ソク		of_v0.11.0_vs2017_release	ファイル フォルダー	
I http://www.action.com/action/act	0f_v0.11.0_vs2017_release.zip	更新日時: 2020/06/27 15:22	🛱 ドキュメント 🔹 🖈			
- = ピクチャ 🖌	開く(O)		- 			
📱 ビデオ 🛛 🖈	新しいウィンドウで開く(E)		📲 ビデオ 🛛 🖈			
🔤 Projects 🛛 🖈	S Skype で共有		🔒 Projects 🛛 🖈			
📥 OneDrive - Personal	■ Code で開く		📏 📥 OneDrive - Personal			
🔷 OpeDrive - 和歌山士(	すべて展開(T)		、 🗢 OpeDrive - 和歌山士	$\sim 1^{\circ} = 2 - 1^{\circ} 1^{\circ}$		
Опертие - днахш х-	スタート Xニューにとン留のする Microsoft Defender でつちれいする			ニトフイノル	トフック & ト	コツノ
PC			V 💻 PC			
🚽 ネットワーク	プログラムから開く(H)	👃 VLC media player	> <b>3D</b> オフジェクト			
	アクヤスを許可する(G) >	- エクスプロー <del>ラ</del> -				
	📲 WinMerge	💼 Microsoft Store を検索する(S)	→ <u>→</u> ) スクトラク ■ ドキュメント			
	以前のパージョンの復元(V)	 別のプログラムを選択(C)	> = P0fv			
	送3(N) >		> 📓 ビデオ			
	切り取り(T)		> ) ミュージック		使い慣れた	-アーカイ
	コピー(C)		> 告 Windows (C:)		バた店。フ	
	ショートカットの作成(S)		> 👝 Backup (D:) → Wi	ndows (C:) へ移動	へを使う(	、伸行ノよい
	削除(D)		> 🪅 ネットワーク			
	名前の変更(M) 					
	プロパティ(R)		<			>
2 個の項目   1 個の項目を選択	510 MB		1個の項目   1個の項目を選択			

## C: ドライブに展開したパッケージの中身

🏪 🎚 🛃 📙 🖛 🛛 Windows (C:)			—	
<b>ファイル ホーム 共有 表示</b>	t			~
← → ∽ ↑ 👆 > PC > Wir	ndows (C:)	ي ~	ሥ Windows (C:)ወ	
1 h/wh anha	名前 ^	更新日時	種類	サイズ
X 2192 アクゼス	Allegorithmic	2019/08/26 8:57	ファイル フォルダー	
📥 OneDrive - Personal	🚽 Autodesk	2019/08/26 8:53	ファイル フォルダー	
👝 OneDrive - 和歌山大学	Brother	2019/02/05 14:40	ファイル フォルダー	
	- gstreamer	2020/01/15 17:47	ファイル フォルダー	
🛗 和歌山大学	LeapSDK	2019/02/04 10:49	ファイル フォルダー	
PC	Libovr	2019/08/26 10:19	ファイル フォルダー	
<u></u> 、 = 3D オブジェクト	librealsense2	2020/06/25-15:22	ファイルフォルダー	
	- maxima-5.42.2	2019/02/05 14:05	<u></u> フォルダー	
⇒ 9990-F	nsys64	2020/06/19 8:16	ファイル フォルダー	
📃 デスクトップ	of_v0.11.0_vs2017_release	2020/06/29 15:18	ファイル フォルダー	
🛗 ドキュメント	- OpenCV	2020/06/19 10:06	ファイル フォルダー	
📰 ピクチャ	PerfLogs	2020/05/07 8:36	ファイル フォルダー	
🛒 ビデオ	📊 Program Files	2020/07/02 9:21	ファイル フォルダー	
シュージック	Program Files (x86)	2020/06/25 12:13	ファイル フォルダー	
Windows (C:)	ProgramData	2020/03/06 7:30	ファイル フォルダー	
	solidangle	2019/08/26 8:57	ファイル フォルダー	
	texlive	2019/02/04 13:45	ファイル フォルダー	
👝 Backup (E:)	Windows	2020/06/25 12:10	ファイル フォルダー	
鹶 ネットワーク	<mark>_</mark> ユーザー	2019/08/06 19:35	ファイル フォルダー	

🛃 📙 ╤    of_v0.11.0_vs201	7_release		- 🗆	×
イル ホーム 共有 表:	<b>T</b>			<b>~</b> 🕐
→ • ↑ 📙 > PC > W	indows (C:) > of_v0.11.0_vs2017_release >	• ٩ 🗸		
		更新日時	種類	サイ
* 2199 222	addons	2020/07/06 11:35	ファイル フォルダー	
📤 OneDrive - Personal	apps	2020/06/29 15:17	ファイル フォルダー	
🔷 OpeDrive,和歌山士学	docs	2020/06/29 15:17	ファイル フォルダー	
	🔄 examples	2020/06/29 15:17	ファイル フォルダー	
📓 和歌山大学	🔄 libs	2020/06/29 15:18	ファイル フォルダー	
PC		2020/06/29 15:17	ファイル フォルダー	
<u>-</u> r⊂ 	projectGenerator-vs	2020/06/30 20:45	ファイル フォルダー	
	scripts	2020/06/29 15:17	ファイル フォルダー	
♣ 9ウンロード	CHANGELOG.md	2019/11/30 20:47	MD ファイル	
📃 デスクトップ	CODE_OF_CONDUCT.md	2019/11/30 20:47	MD ファイル	
🗮 ドキュメント	INSTALL.md	2019/11/30 20:53	MD ファイル	
📰 ピクチャ	INSTALL_FROM_GITHUB.md	2019/11/30 20:47	MD ファイル	
📱 ビデオ	LICENSE.md	2019/11/30 20:47	MD ファイル	
▶ ≂¬_>ĵ₩⊅	README.md	2019/11/30 20:47	MD ファイル	
🖕 Windows (C:)	THANKS.md	2019/11/30 20:47	MD ファイル	
👝 Data (D:)				

👝 Backup (E:)

💣 ネットワーク

openFrameworks のパッケージの中身

## パッケージの内容

#### windows 版のパッケージ



#### osx 版のパッケージ



## 配置したパッケージはむやみに移動しない

- openFrameworks を使って作ったプログラム(プロジェクト)は openFrameworks のフォルダ内にある
  - 何かのプロジェクトをビルド後にパッケージ自体を別のところに 移動すると以降ビルドに失敗するようになる
    - 最初にビルドしたときに作られる openFrameworks のライブラリファイルの場所を絶対パスで記録している



# 課題1-2 サンプルプロジェクトのビルド

## (windows) 3DPrimitivesExample.sln

📙    💆 📙 🗧    3DPrimitivesExample					— 🛛	×
ファイル ホーム 共有 表示						× 🕐
← → ∽ ↑ 📙 → PC → Data (D:) → User:	s > tokoi > デスクトップ > of_v0.10.1_vs2017_rele	ease > examples > 3d >	3DPrimitivesExample		✓ ひ 3DPrimitivesExampleの検索	م
of_v0.10.1_vs2017_release			種類	サイズ	3DPrimitivesExample sln	
addons		2019/11/26 11:57	' ファイル フォルダー		Visual Studio Solution	
apps		2019/11/26 11:57	ファイル フォルダー			
docs	3DPrimitivesExample.png	2018/11/15 0:31	PNG ファイル	781 KB	15	
examples	3DPrimitivesExample.sln	2018/11/15 0:33	Visual Studio Solu	зKB		
 3d	3DPrimitivesExample.vcxproj	2018/11/15 0:33	VC++ Project	11 KB		
	3DPrimitivesExample.vcxproj.filters	2018/11/15 0:33	VC++ Project Filte	1 KB		
3DPrimitivesEvample	3DPrimitivesExample.vcxproj.user	2018/11/15 0:33	Per-User Project O	2 KB	更新日時: 2018/11/15 0:33	
bin	addons.make	2018/11/15 0:33	MAKE ファイル	O KB	サイズ: 2.02 KB	
	icon.rc	2018/11/15 0:33	Resource Script	1 KB	作成日時: 2018/11/15 0:33	
src	README.md	2018/11/15 0:31	MDファイル	2 KB		
advanced3dExample						
assimpExample						
cameraLensOffsetExample						
cameraParentingExample						
cameraRibbonExample						
easyCamExample						
meshFromCameraExample						
modelNoiseExample						
normalsExample						
ofBoxExample						
ofNodeExample						
orientationExample						
n n int Claud Funnale						
10 個の項目   1 個の項目を選択 2.02 KB						

## セキュリティ警告が出ても「OK」

3DPrimitivesExample に対するセキュリティ警告	?	×
信頼性の高い発行元からのプロジェクトのみを開くようにしてください。		
プロジェクト ファイル 3DPrimitivesExample は、完全に信頼された発行元からのファイルではない可能性がありませ Visual Studio で開いて、カスタムビルドステップを実行すると、セキュリティ上の危険を引き起こすおそれがあります が信頼されない発行元から提供されている場合、お使いのコンピューターが破損したり、個人情報が侵害されたりす があります。	す。 Micros た。このプロジ する原因にな	oft ジェクト なること
このプロジェクトを開きますか?		
✓ ソリューション内のすべてのプロジェクトに対して確認メッセージを表示する(A) OK	キャン	リセル
openframeworksLib に対するセキュリティ警告	?	×
信頼性の高い発行元からのプロジェクトのみを開くようにしてください。		
プロジェクトファイル openframeworksLib は、完全に信頼された発行元からのファイルではない可能性があります	. Microso	oft

フロジェクト ファイル openframeworksLib は、完全に信頼された発行元からのファイル ぐはない 可能性があります。Microsoft Visual Studio で開いて、カスタム ビルドステップを実行すると、セキュリティ上の危険を引き起こすおそれがあります。このプロジェクト が信頼されない発行元から提供されている場合、お使いのコンピューターが破損したり、個人情報が侵害されたりする原因になること があります。

このプロジェクトを開きますか?

✓ ソリューション内のすべてのプロジェクトに対して確認メッセージを表示する(A)

ОК	キャンセ

## 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 パージョン: 10.0 (最新のインストールされているパージョン) 、	
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	
<ul> <li>¥3DPrimitivesExample¥3DPrimitivesExample.vcxproj</li> <li>¥vs¥openframeworksLib.vcxproj</li> </ul>	
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

## (windows) デバッグを開始する



## (windows) 実行中



## (macOS) 3DPrimitivesExample.xcodeproj



(macOS) "Open" する

# 「インターネットからダウンロードしてきたものだけど本当 に開いてよいか」って聞かれるので "Open" をクリック



## (macOS) Other Code Signing Flags (こ "--deep"

② 「BDPintitivesExample ③ ConstructivesExample ④ ConstructivesExample ④ ConstructivesExample ④ ConstructivesExample ④ ConstructivesExample ④ ConstructivesExample ④ ConstructivesExample ● ConstructivesExample	1 🔹 🔹 🕨 📄 🍌 3DPriebug 👌	🖳 My Mac 3DPrimitivesExample   Build Failed 🛕 10 🛽 1 🎓 🕂 🕂
② PerindivesExample		🗄 < > 🖹 3DPrimitivesExample < 🛆 > 🕞
② configure Accordig Project.xccordig src main.cpp c ofApp.h openFrameworks openFrameworks odApp.h openFrameworks </td <td>3DPrimitivesExample</td> <td>🔲 🗘 Jeneral Signing &amp; Capabilities Resource Tags Info Build Settings Build Phases Build</td>	3DPrimitivesExample	🔲 🗘 Jeneral Signing & Capabilities Resource Tags Info Build Settings Build Phases Build
<ul> <li>Projectacoming</li> <li>Projectacoming</li> <li>graphics</li> <li>graddons</li> <li>graphics</li> <li>graphics</li></ul>	openFrameworks-Info.plist	Basic Customized All Combined Levels + Q~ code sign (S)
Signing     Signing     Signing     Setting	src	
C of App. cpp h of App.h openFrameworks b 3d b app b communication b ovents b g d b g d b of Main.h b sound b types b utils b video v addons v iccal.addons c communication b other Code Signing Flags deep vdeep"を入力して改行 (***をクリック)	🕞 main.cpp	* signing (4) "code sign" を検索
h of App.h	G. ofApp.cpp	Setting A 3DPrimitivesExample
Code Signing Identity app app communication code Signing Style code Signing Style code Signing Flags Code Signing Fl	h ofApp.h	Code Signing Entitlements
→ app → app → code Signing Style → orMain.h → orMain.h → sound → types → utils → video ▼ addons ▼ blocal_addons ▼ blocal_addons ▼ blocal_addons ▼ blocal_addons ▼ blocal_addons ▼ blocal_addons ▼ blocal_addons ▼ blocal_addons ▼ blocal_addons ★ 3DPrimitivesExampleDebug.app + ⓒ Fiter ◆ Titer	openFrameworks	Code Signing Identity Sign to Run Locally
<ul> <li>Communication</li> <li>events</li> <li>gi</li> <li>graphics</li> <li>math</li> <li>f Main.h</li> <li>sound</li> <li>types</li> <li>utils</li> <li>video</li> <li>addons</li> <li>local_addons</li> <li>3DPrimitivesExampleDebug.app</li> <li>+ € Filter</li> </ul>		Code Signing Style Automatic
<ul> <li>▶ events</li> <li>&gt; gl</li> <li>&gt; graphics</li> <li>&gt; math</li> <li>&gt; fofMain.h</li> <li>&gt; sound</li> <li>&gt; types</li> <li>&gt; utils</li> <li>&gt; video</li> <li>▼ addons</li> <li>▼ local_addons</li> <li>★ 3DPrimitivesExampleDebug.app</li> <li>+ © Filter</li> </ul>	▶ communication	► Other Code Signing Flags
→ g g → graphics → math → ofMain.h → sound → types → utils → video ▼ addons ▼ local_addons + ⓒ Filter ① 図 + ♡ Filter ① 図 +deep"を入力して改行	events	
<ul> <li>■ graphics</li> <li>■ math</li> <li>● ofMain.h</li> <li>■ sound</li> <li>■ types</li> <li>■ utils</li> <li>■ video</li> <li>▼ addons</li> <li>▼ blocaLaddons</li> <li>* 3DPrimitivesExampleDebug.app</li> <li>+ ● Filter</li> </ul>	▶ 🛄 gi	deep
<ul> <li>in math</li> <li>in of Main.h</li> <li>is sound</li> <li>it types</li> <li>in utils</li> <li>in video</li> <li>in local_addons</li> </ul>	graphics	
<ul> <li>Name</li> <li>Sound</li> <li>types</li> <li>utils</li> <li>video</li> <li>addons</li> <li>local_addons</li> <li>A 3DPrimitivesExampleDebug.app</li> <li>+ € Filter</li> </ul>	▶ matn	
<ul> <li>▶ types</li> <li>▶ utils</li> <li>▶ video</li> <li>▼ addons</li> <li>▼ local_addons</li> <li>A 3DPrimitivesExampleDebug.app</li> <li>+ © Filter</li> </ul>	▶ sound	
<ul> <li>▶ utils</li> <li>▶ video</li> <li>▼ addons</li> <li>▼ local_addons</li> <li>★ 3DPrimitivesExampleDebug.app</li> <li>+ 등 Filter</li> </ul>	▶ <mark></mark> types	
▶ video ▼ addons ▼ local_addons A 3DPrimitivesExampleDebug.app + <a href="million">Filter</a> () <a href="million">() <a href="million">million</a></a>	▶ 📩 utils	
Addons     Addons     A 3DPrimitivesExampleDebug.app     +      Filter     C	▶ video	
A 3DPrimitivesExampleDebug.app +	V addons	
	A 3DPrimitivesExampleDebug.app	<u> "+" をクリック ⑥</u>
	+ (🖲 Filter	+ -

## (macOS) "Run" する

🗧 💿 🕨 📄 🖂 3DPriebug 👌	My Mac 3DPrimitivesExample: Ready   Today at 23:39		
	멾 < > No Selection	< 🔺 > 🕞	
🔻 🔄 3DPrimitivesExample			
openFrameworks-Info.plist			
Project.xcconfig			
▼ <mark>src</mark>			
🛃 main.cpp			
G→ ofApp.cpp			
h ofApp.h			
▼ openFrameworks			
▶ <mark></mark> 3d			
▶ 📩 app			
communication			
▶ events			
▶ <mark></mark> gl	No Editor		
▶ graphics			
▶ math			
h) ofMain.h			
▶ sound			
▶ types			
▶ utils			
▶ video			
▼ <u></u> addons			
▼ <u> </u> local_addons			
A 3DPrimitivesExampleDebug.app			
+ 🕞 Filter			

## (macOS) 実行中



## 実行結果のスクリーンショットを撮る



## "1-2.png" というファイル名で保存





## スクリーンショットのアップロード

## ■実行結果のスクリーンショットを1-2.pngというファイル名で Moodle の第1回課題にアップロードしてください




# 課題1-3 他のサンプルプロジェクトのビルド

### 他のサンプルプロジェクトをビルドする

- example のサンプルプロジェクトは openFrameworks で何か 作ろうとしたときに必ず参考になります
- 他のサンプルプロジェクトのプロジェクト名を一通り見てく
   ださい
  - 何をするものか考えてください
- これらの中から3つ以上のサンプルプロジェクトをビルド・
   実行してみてください
  - カメラやマイクが必要なものもあります

# 課題1-4<br /> 空のプロジェクトを作成してビルド

#### 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



#### Project name:

- 作成するプロジェクト(プログラム)の名前
- Project path:
  - 作成するプロジェクトのファイル を置く場所
  - openFrameworksのパッケージを展開した場所の中の apps¥myApps

### プロジェクトの作成成功

	📙    🛂 📙 🚽    mySketch		- 0	$\times$
create / update	ファイル ホーム 共有 表示			¥ 🕐
	← → ▾  🔤 « myA → mySketch →	م <b>ن ب</b>	mySketchの検索	
	名前 ^		種類	サイズ
	.vs	2020/06/29 16:47	 ファイル フォルダー	
Success! クリックタると開く	ー bin これをグブ	2020/06/29 16:48	ファイル フォルダー	
Your can now find your project in		2020/06/29 16:48	ファイル フォルダー	
<pre><openframeworks 00="" 展用场別="">¥apps¥myApps¥mySketcn</openframeworks></pre>		2020/06/29 16:48	ファイル フォルダー	
	🗋 addons.ma してもよい	2020/06/29 15:41	MAKE ファイル	C
[notice]	🖂 icon.rc	2019/11/30 20:47	Resource Script	1
<pre>[notice ] setting OF path to: D:\Users\tokoi\Desktop\of_v0.10.1_vs2017_release [notice ] from -0 ontion</pre>	🐺 mySketch.sln	2020/06/29 15:41	Microsoft Visual S	з
[notice] Hange option [notice] target platform is: vs [notice] project plath is: D:\Users\tokoj\Deskton\of v0.10.1 vs2017 release\anns\mv0nns	🐄 mySketch.vcxproj	2020/06/29 15:41	VC++ Project	11
(motice ) project path is: D. Josefs (taka) (desition (d	mySketch.vcxproj.filters	2020/06/29 15:41	VC++ Project Filte	1
[notice ] secting of new project of (ast s (const located) (a)_orizot_sects_ceres (apps)	🗿 mySketch.vcxproj.user	2020/06/29 15:41	Per-User Project O	2
Open in IDE Close				
Update				
IDE (Visual Studio) で開く				
	<			<b></b> >

#### IDE (Visual Studio, Xcode) で開く



#### 実行中のウィンドウ



#### main() 関数は(今は)いじらない



## ofApp クラス("クラス"の話は後日)

🔀 ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビル	<sup>×</sup> (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)	検索 (Ctrl+Q) 👂	mySketch	- 0	×
🖁 G - O   📸 - 🏠 💾 🔐   🤊 - C -   Debug -	Win32 -	ーカル Windows デバッガー 👻 自動	•   🎜   🖾 🖕			🖄 Live Share	Ŕ
ソリューション エクスプローラーの検索(Ctrl+:)       ク         ソリューション エクスプローラーの検索(Ctrl+:)       ク         マリューション 'mySketch' (2/2 プロジェクト)       (2/2 プロジェクト)         ● ● 参照       ● ● 参照         ● ● ● 外部依存関係       ● ● ● 参照         ● ● ● ofApp.cpp       ● ● ● ofApp.h         ● ● ● ofApp.h       ● ● ● openframeworksLib	ImpSketch         1       #pragma         2       3         3       #include         4       5         5       □class of         6       7         7       public         8       voir         9       voir         10       voir         11       12         12       voir         13       voir         14       voir         15       voir         16       voir         17       voir         18       voir         20       voir         21       voir         23       24         25       ;	<pre>     (グローパ     once         "ofMain.h"     fApp : public ofBaseApp{         c:         d setup();         d update();         d draw();         d keyPressed(int key);         d keyReleased(int x, int y);         d mouseMoved(int x, int y);         d mousePressed(int x, int y,         d mouseEntered(int x, int y);         d mouseExited(int x, int y);         d windowResized(int w, int h         d dragEvent(ofDragInfo dragI         d gotMessage(ofMessage msg);     } } </pre>	いレスコープ) int button); int button); , int button); ; nfo);			ofApp.h 🗯 🗙	サーバー エクスプローラー ツールボックス プロパティ 診断ツール
<u> ソリューション I</u> クラス ビュー   プロバティ マネ   チーム エクスプ	_ 100 % - ♥ 問題は見	しつかりませんでした <			▶ 行:1	文字:1 9ブ	LF
目準備完了					רע <u>ז</u> ר	ノ管理に追加 ▲	÷ .

#### ofApp クラスのメンバの実装はまだ空

ズ ファイル(F) 編集(E) 表示(V) 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)	検索 (Ctrl+Q)	Р my	Sketch	- 🗆	×	C
🕺 © - ©   🏠 - 놀 💾 🚰   🤊 - 🤆 -   Debug -	Win32	<ul> <li>▶ □-カ</li> </ul>	ル Windows デバッガ・	- • 自動	•	چ 🗟				🖄 Live Share	ጽ	
ソリューション エクスプローラーの検索(Ctrl+:)       ク         ソリューション エクスプローラーの検索(Ctrl+:)       ク         ワリューション 'mySketch' (2/2 プロジェクト)       ・         ・       ・ <td><ul> <li>mySketch</li> <li>1</li> <li>2</li> <li>3</li> <li>4</li> <li>5</li> <li>6</li> <li>7</li> <li>8</li> <li>9</li> <li>10</li> <li>11</li> <li>12</li> <li>13</li> <li>14</li> <li>15</li> <li>16</li> <li>17</li> <li>18</li> <li>19</li> <li>20</li> <li>21</li> <li>22</li> <li>23</li> <li>24</li> <li>25</li> <li>26</li> </ul></td> <td><pre>#include // evoid of Ap } // evoid of Ap }</pre></td> <td><pre>"of App.h" p::setup(){ p::update(){ p::draw(){ p::keyPressed(i p::keyReleased()</pre></td> <td><pre> • (グローパリレ  nt key){ int key){ </pre></td> <td>スコープ)</td> <td></td> <td></td> <td></td> <td>ofA</td> <td>pp.cpp 🚡 🗙</td> <td></td> <td>サーバー エクスプローラー ツールボックス プロパティ 診断ツール</td>	<ul> <li>mySketch</li> <li>1</li> <li>2</li> <li>3</li> <li>4</li> <li>5</li> <li>6</li> <li>7</li> <li>8</li> <li>9</li> <li>10</li> <li>11</li> <li>12</li> <li>13</li> <li>14</li> <li>15</li> <li>16</li> <li>17</li> <li>18</li> <li>19</li> <li>20</li> <li>21</li> <li>22</li> <li>23</li> <li>24</li> <li>25</li> <li>26</li> </ul>	<pre>#include // evoid of Ap } // evoid of Ap }</pre>	<pre>"of App.h" p::setup(){ p::update(){ p::draw(){ p::keyPressed(i p::keyReleased()</pre>	<pre> • (グローパリレ  nt key){ int key){ </pre>	スコープ)				ofA	pp.cpp 🚡 🗙		サーバー エクスプローラー ツールボックス プロパティ 診断ツール
<u> </u>	100 % -	○ 問題は見つ/	かりませんでした	<				) 	行:1 : ↑ ソース	文字:1 SPC 管理に追加 <u>▲</u>	LF	

#### 課題1-5

openFrameworks を使ったプログラムの動作を確認してみよう

### of App.cpp の内容(先頭部分)

<pre>#include "ofApp.h"</pre>
<pre>// void ofApp::setup(){</pre>
}
<pre>// void ofApp::update(){</pre>
}
//void ofApp::draw(){
}
<pre>// void ofApp::keyPressed(int key){</pre>
}

#### メンバ関数 (画面表示関連)

void ofApp::setup()
 アプリケーションを起動したときに一度だけ実行される

■ void ofApp::update() ■ 画面の表示を行う前に**繰り返し**実行される

■ void ofApp::draw() ■ 画面の表示を**繰り返し**行う

> "void" は戻り値を返さないことを表す これらの関数は return を使わない

#### setup() で文字を出力してみる

<pre>#include "ofApp.h"</pre>
<pre>// void ofApp::setup(){     std::cout &lt;&lt; "setup()¥n"; }</pre>
<pre>// void ofApp::update(){</pre>
}
<pre>// void ofApp::draw(){</pre>
}

#### setup() は 最初に一度だけ実行される

setup()

## update() で文字を出力してみる

```
#include "ofApp.h"
//-----
void ofApp::setup(){
 std::cout << "setup()¥n";</pre>
}
    _____
void ofApp::update(){
 std::cout << "update()¥n";</pre>
}
void ofApp::draw(){
}
```

<pre>setup()</pre>	
update()	update() (よ
update()	繰り返し何度も実行されている
update()	
	<pre>setup() update() update()</pre>

#### draw() で文字を出力してみる

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

```
//----
void ofApp::setup(){
   std::cout << "setup()¥n";
}</pre>
```

```
//----
void ofApp::update(){
   std::cout << "update()¥n";
}</pre>
```

```
//-----
void ofApp::draw(){
    std::cout << "draw()¥n";
}</pre>
```

```
setup()
update()
draw() -
update()
draw()
•••
```

draw() は update() の 後に実行される

#### メンバ関数 (キー操作関連)

- keyPressed(int key)
   キーを押したときに実行
   keyは押したキー
- keyReleased(int key)
   キーを離したときに実行
   keyは押していたキー

#### キーは文字が表示されている**黒い**ウィンドウ(コンソー ル)ではなく**グレー**のウィンドウでタイプしてください

#### keyPressed(int key) で文字を出力してみる

•••

```
void ofApp::keyPressed(int key){
  std::cout << static_cast<char>(key)
            << " was pressed¥n";</pre>
```

```
void ofApp::keyReleased(int key){
```

}

}

update() <u>このウィンドウ(コンソール)ではな</u> draw() くアプリケーションのウィンドウで update() draw() c was pressed <u>キーを押したときに</u> update() draw()の後で実行される draw() update() draw() update() draw() d was pressed update() draw() update() draw() update() draw()

#### keyReleased(int key) で文字を出力してみる

```
void ofApp::keyPressed(int key){
                                                       update()
  std::cout << static cast<char>(key)
                                                       draw()
            << " was pressed¥n";
                                                       f was pressed
}
                                                       update()
                                                       draw()
                                                       update()
void ofApp::keyReleased(int key){
                                                       draw()
std::cout << static_cast<char>(key)
                                                        f was released
           << " was released¥n";</pre>
                                                       update()
                                                       draw()
}
                                                       g was pressed
                                                       update()
                                                       draw()
                                                       update()
                                                       draw()
                                                       g was released
                                                       update()
                                                       draw()
```

```
キーを離したときに
draw()の後で実行される
```

# 課題1-6

#### 簡単な図形を描いてみよう

129

#### setup() で背景色を指定する



void ofBackground(int r, int g, int b, int a=255)

- 背景色を指定する
  - r:背景色の赤成分の強さ、0~255
  - g:背景色の緑成分の強さ、0~255
  - b:背景色の青成分の強さ、0~255
  - a:背景色の不透明度、0(透明)~
     255(不透明)、省略時は255
- ofBackGround(255, 255, 255) なら白
- マニュアル
  - https://openframeworks.cc/documentati on/graphics/ofGraphics/#show\_ofBackgr ound





### マニュアルの見方

#### global functions

- ofBackground()
- > ofBackgroundGradient()
- > ofBackgroundHex()
- > ofBeginSaveScreenAsPDF()
- > ofBeginSaveScreenAsSVG()
- > ofBeginShape()
- > ofBezierVertex()
- > ofClear()
- > ofClearAlpha()
- > ofCurveVertex()
- > ofCurveVertices()
- > ofDisableAlphaBlending()
- > ofDisableAntiAliasing()
- > ofDisableBlendMode()
- > ofDisableDepthTest()
- > ofDisablePointSprites()
- > ofDisableSmoothing()



It takes as input r,g,b (0-255). The background is cleared automatically, just before the draw() command, so if the background color is not changing, you could call this inside of setup() (once, at the start of the application). If the background color is changing, you can call this inside of update().

```
void ofApp::setup(){
    ofBackground(255,0,0); // Sets the background color to red
}
```

使用例









void ofDrawCircle(float x, float y, float radius)

- 円を描く
  - x: 円の中心の x 座標値
  - y: 円の中心の y 座標値
  - radius: 円の半径
- ofDrawCircle(30.0f, 40.0f, 100.0f) なら (30, 40) を中心に半径 100の円
- マニュアル
  - https://openframeworks.cc//documentati on/graphics/ofGraphics/#!show\_ofDrawCi rcle

#### 左上に円が描かれる







void ofSetColor(int r, int g, int b) void ofSetColor(int r, int g, int b, int a) 以後描画するものの色を指定する ■ r: 背景色の赤成分の強さ、0~255 ■ g:背景色の緑成分の強さ、0~255 ■ b:背景色の青成分の強さ、0~255 ■ a: 背景色の不透明度、 0 (透明) ~ 255(不透明)、省略時は255 ■ マニュアル https://openframeworks.cc//documentati

on/graphics/ofGraphics/#!show ofSetCol

or







```
#include "ofApp.h"
void ofApp::setup(){
  ofBackground(0, 0, 0);
}
void ofApp::update(){
}
void ofApp::draw(){
  ofSetColor(255, 0, 0);
  ofDrawCircle(0.0f, 0.0f, 100.0f);
  ofDrawRectangle(300.0f, 200.0f, 90.0f, 60.0f);
}
```

void ofDrawRectangle(float x1, float y1, float w, float h)

- 矩形を描く
  - x1: 矩形の左端の x 座標
  - y1: 矩形の上端の y 座標
  - w: 矩形の幅
  - h:矩形の高さ
- ofDrawRectangle(10.0f, 20.0f, 100.0f, 200.0f) なら (10, 20) を左上にして 幅 100 高さ 200 の矩形
- マニュアル
  - <u>https://openframeworks.cc//documentati</u> on/graphics/ofGraphics/#!show\_ofDrawR <u>ectangle</u>

#### 矩形が追加される





```
#include "ofApp.h"
void ofApp::setup(){
  ofBackground(0, 0, 0);
}
void ofApp::update(){
}
void ofApp::draw(){
  ofSetColor(255, 0, 0);
  ofDrawCircle(0.0f, 0.0f, 100.0f);
 ofSetColor(50, 100, 240);
  ofDrawRectangle(300.0f, 200.0f, 90.0f, 60.0f);
```

 ofSetColor() で指定した色は
 ofSetColor()の呼び出し以降に描 画する図形に適用される





# 課題1-7

#### 自分で図形を作ってみよう

## 二次元図形を描く

- 以下の openFrameworks の関数を使って何か二次元の図形を描いてください
  - ofDrawCircle()
    - https://openframeworks.cc//documentation/graphics/ofGraphics/#!show\_ofDrawCircle
  - ofDrawCurve()
    - https://openframeworks.cc//documentation/graphics/ofGraphics/#!show\_ofDrawCurve
  - ofDrawEllipse()
    - https://openframeworks.cc/documentation/graphics/ofGraphics/#show\_ofDrawEllipse
  - ofDrawLine()
    - https://openframeworks.cc/documentation/graphics/ofGraphics/#show\_ofDrawLine
  - ofDrawRectRounded()
    - https://openframeworks.cc/documentation/graphics/ofGraphics/#show\_ofDrawRectRounded
  - ofDrawRectangle()
    - https://openframeworks.cc/documentation/graphics/ofGraphics/#show\_ofDrawRectangle
  - ofDrawTriangle()
    - <u>https://openframeworks.cc/documentation/graphics/ofGraphics/#show\_ofDrawTriangle</u>

#### 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 1-7.png というファイル名で保存し、Moodle の第1回課題にアップロードしてください

ソースプログラム ofApp.h と ofApp.cpp を Moodle の第1回
 課題にアップロードしてください


## of App.cpp の保存場所

