Xcode プロジェクトの演習

Cocoa Application を使ったプログラムの作成

H. Kawakami

November 6, 2015(H27)

Cocoa Framework を使ったプログラム

(1) Color 表示と直線の描画

Cocoa の Graphics

(2) Scalingとphase portrait

Timer を使って描画に動きを付ける

主として Graphics と Event handling の入門

演習の目的と目標

演習の目的

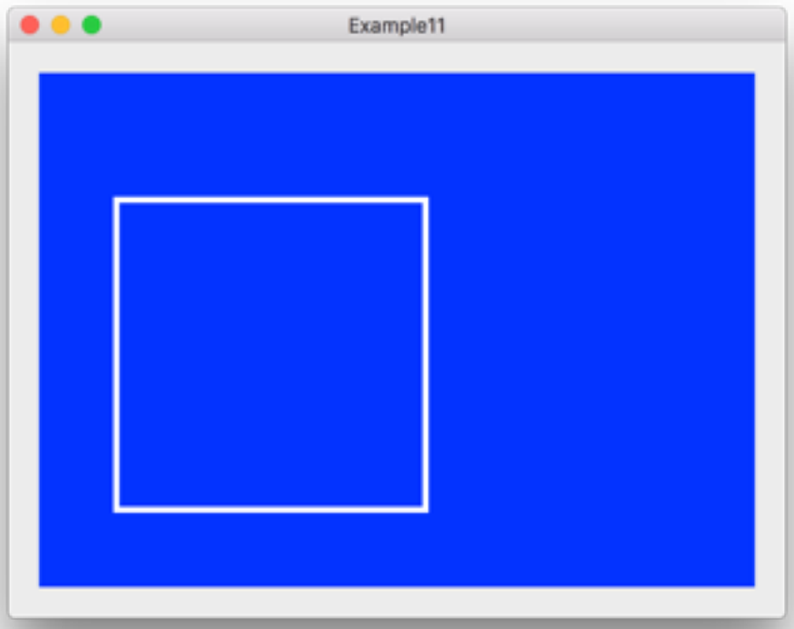
- Mac OS X のプログラム開発環境 Xcode を使って力学系のプログラムを書く

演習の目標

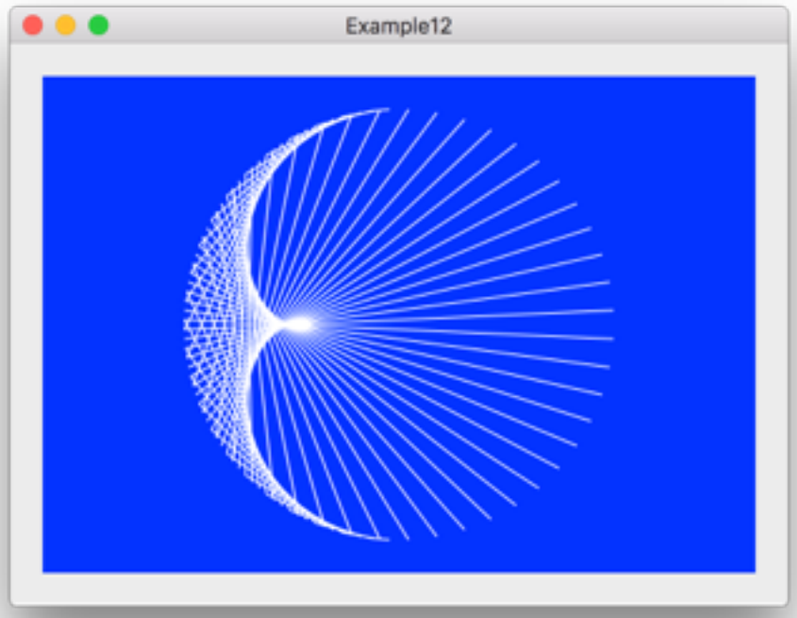
- X code 開発環境 IDE(integrated development environments)の使い方をちょっとだけ習得
- Mac OS X の Graphics を少し使ってみる
- OOP プログラムの手法を Objective-C で体験
- 力学系のプログラムを開発する

Cocoa Application：今日の演習

view を 1 つ表示する

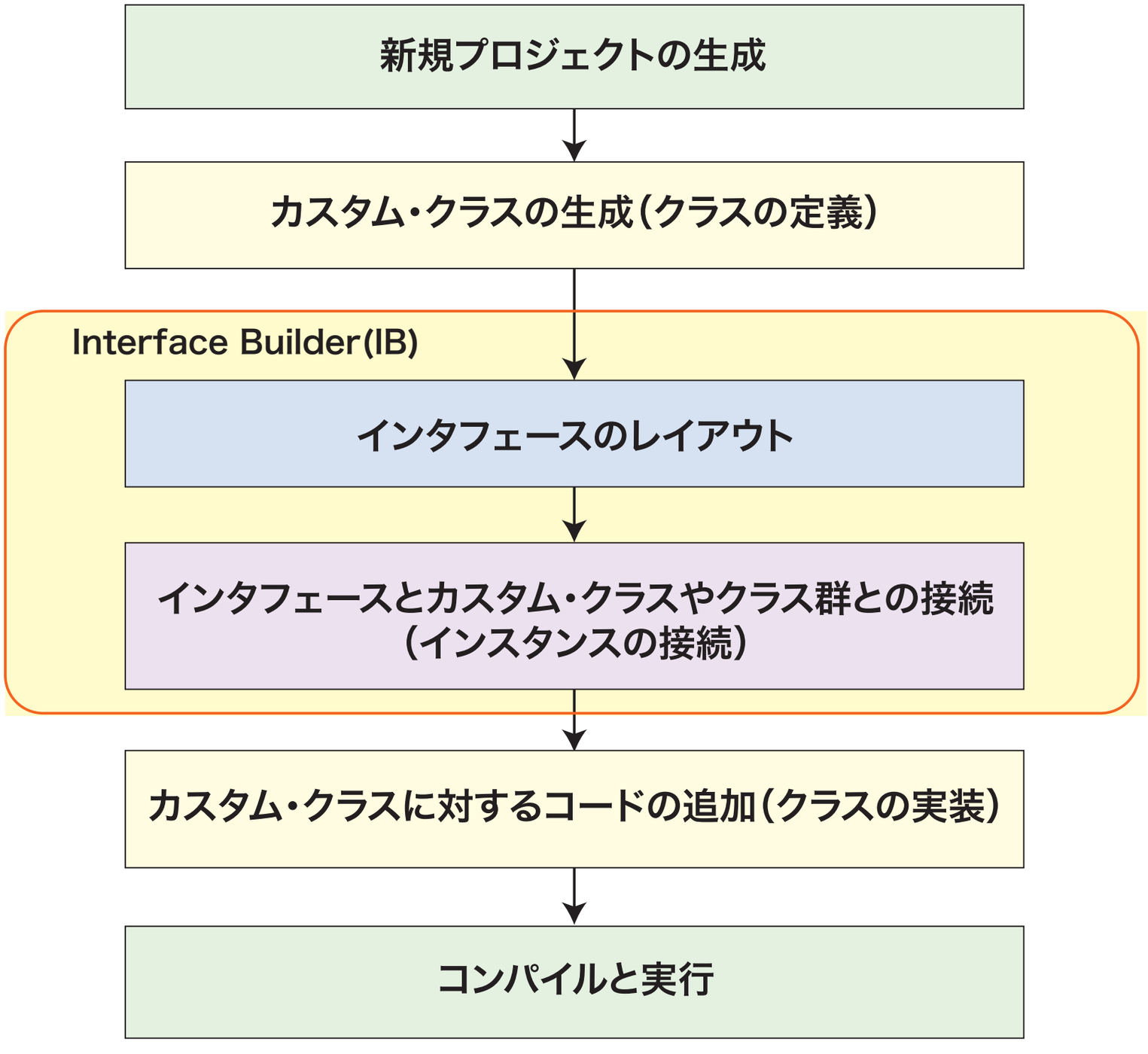


bezier path を表示する

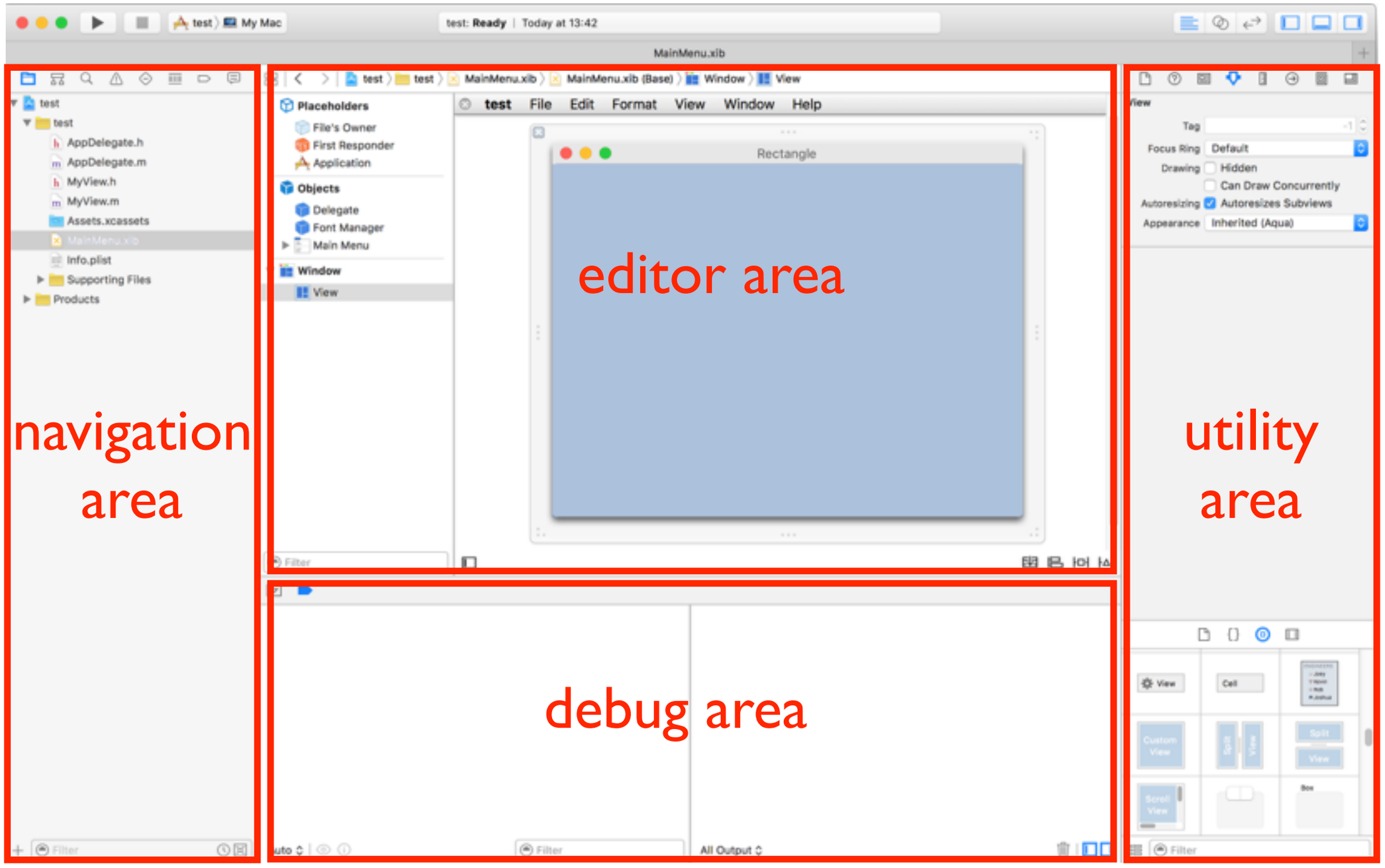


Cocoa Application の作り方を習得

プログラムをつくる流れ



Workspace window



Xcode プロジェクト: Cocoa Application

1．新しいプロジェクトをつくる

2．クラスを定義し，クラスファイルをつくる

3．View の作成･定義（xib ファイルの編集）

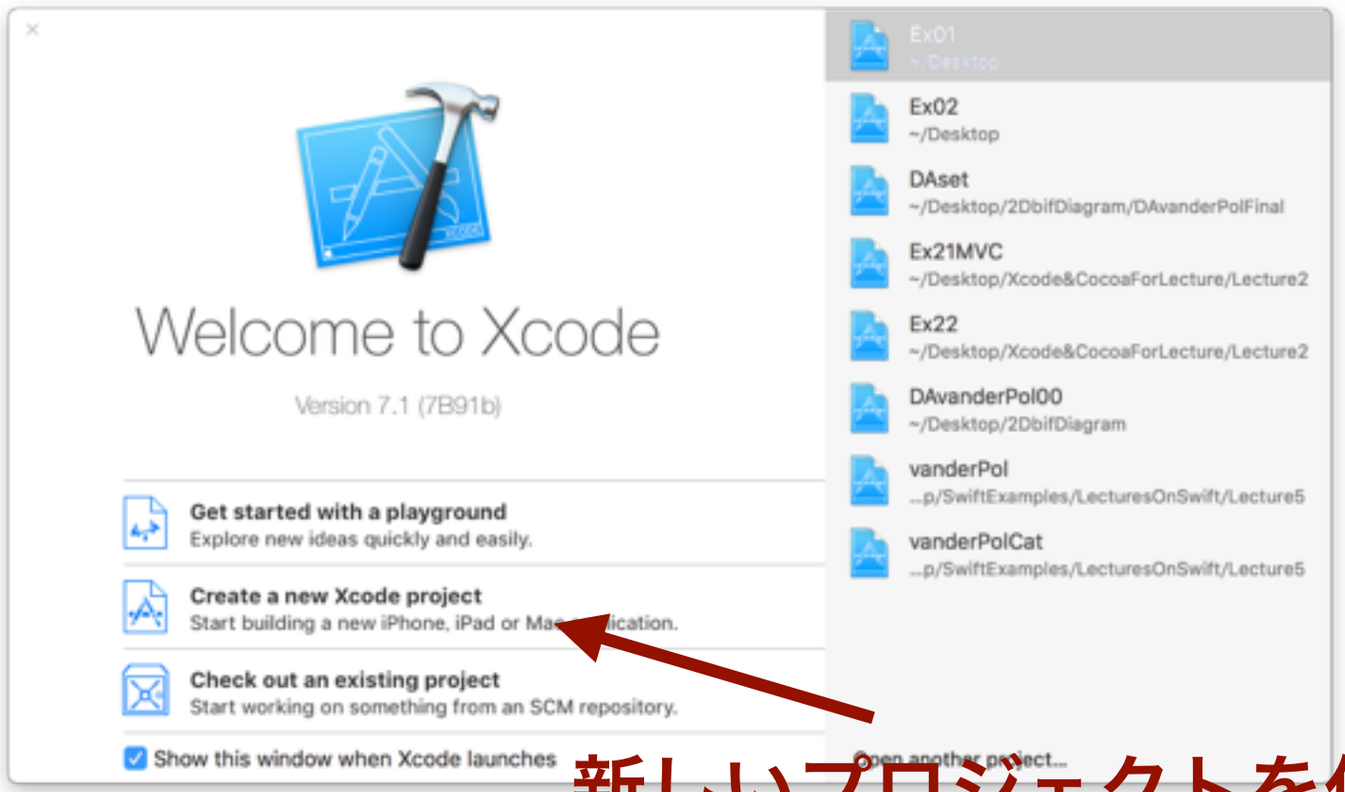
4．定義したクラスとViewの接続

5．プログラムのコードを書く

6．実行

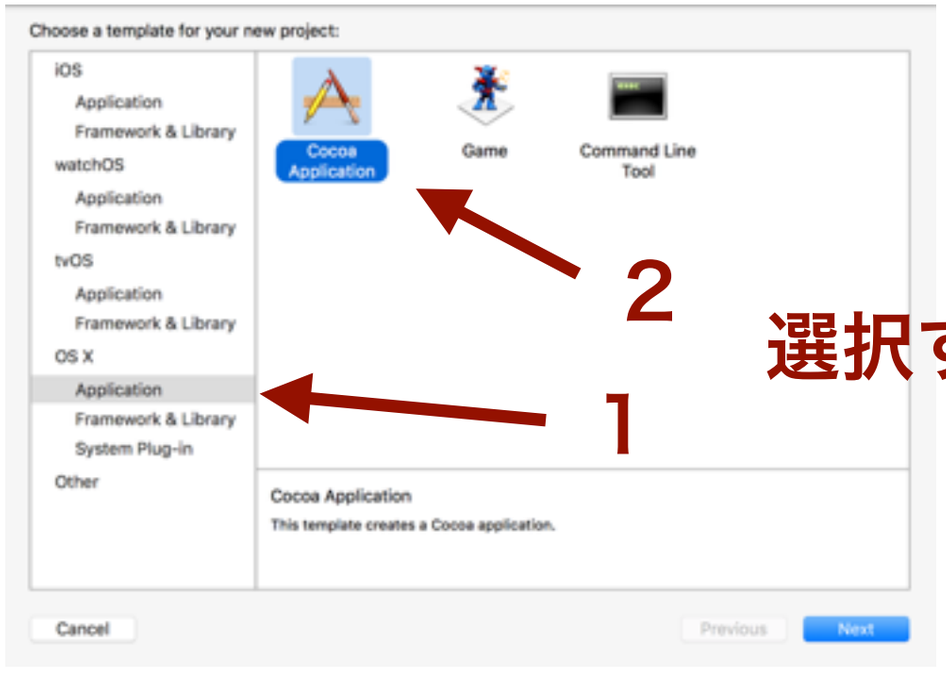
1．新しいプロジェクトをつくる(Xcode)

・Xcodeを開いて，　メニュー：ファイル > 新規プロジェクト

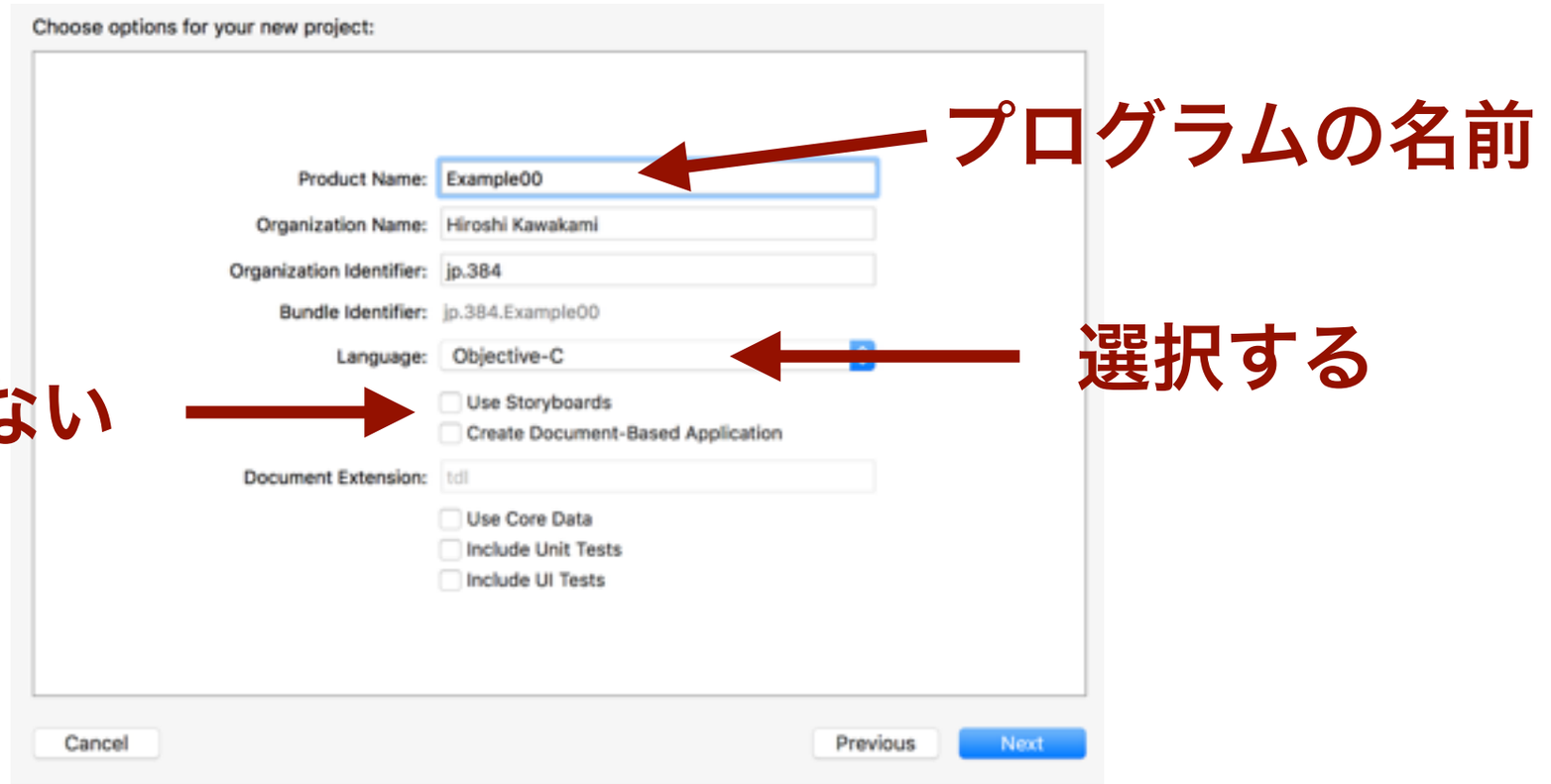


新しいプロジェクトを作るを選択

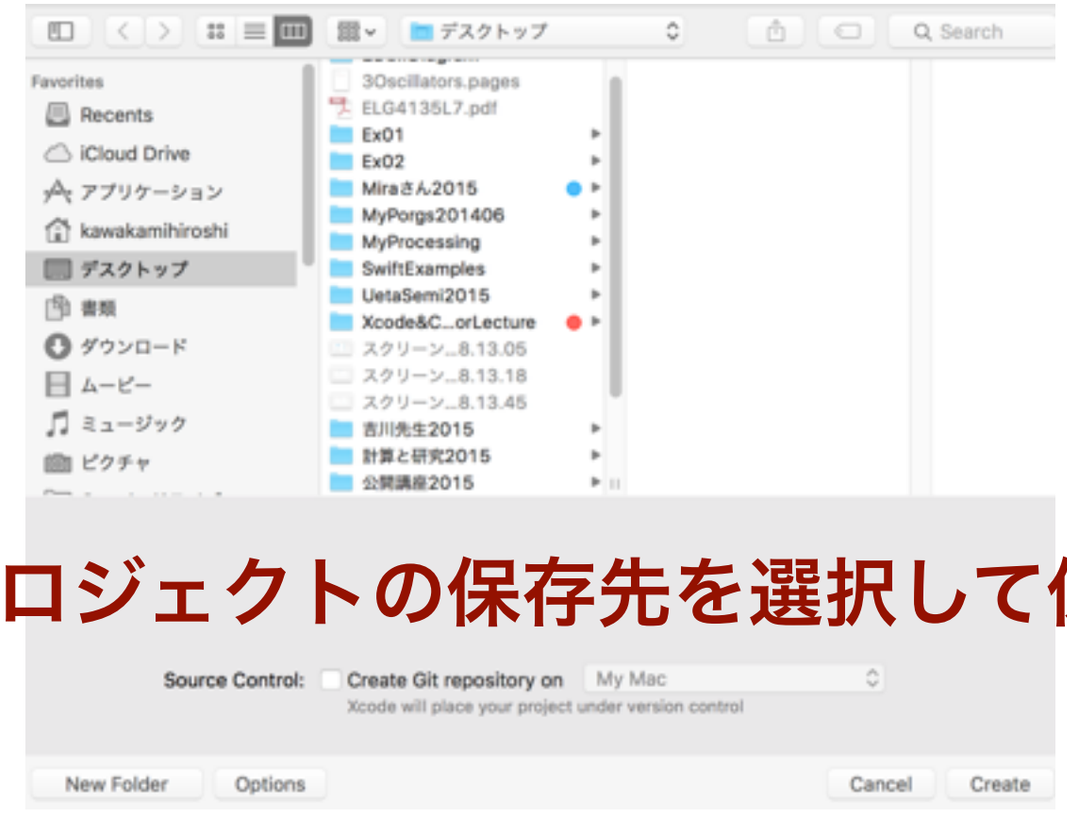
・Mac OSX の Application ， Cocoa Application を選択



・プロジェクト名（例えば　Example00）を入れて，保存



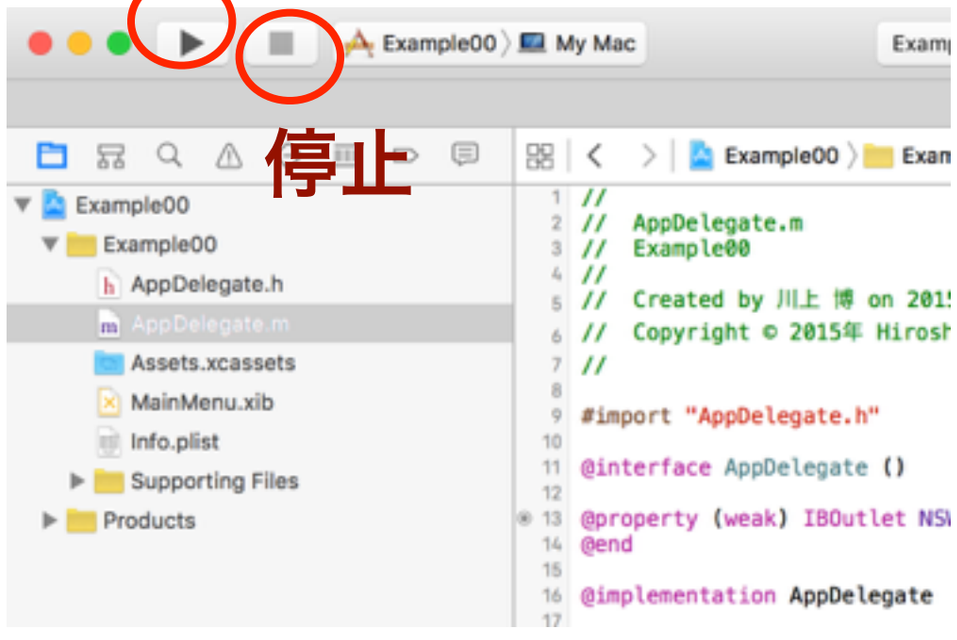
・プロジェクトを保存



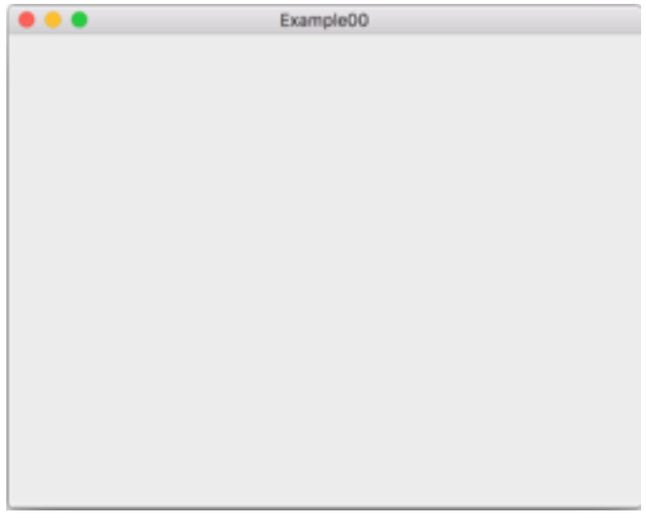
プロジェクトの保存先を選択して保存

実行

停止



ここで実行すると…

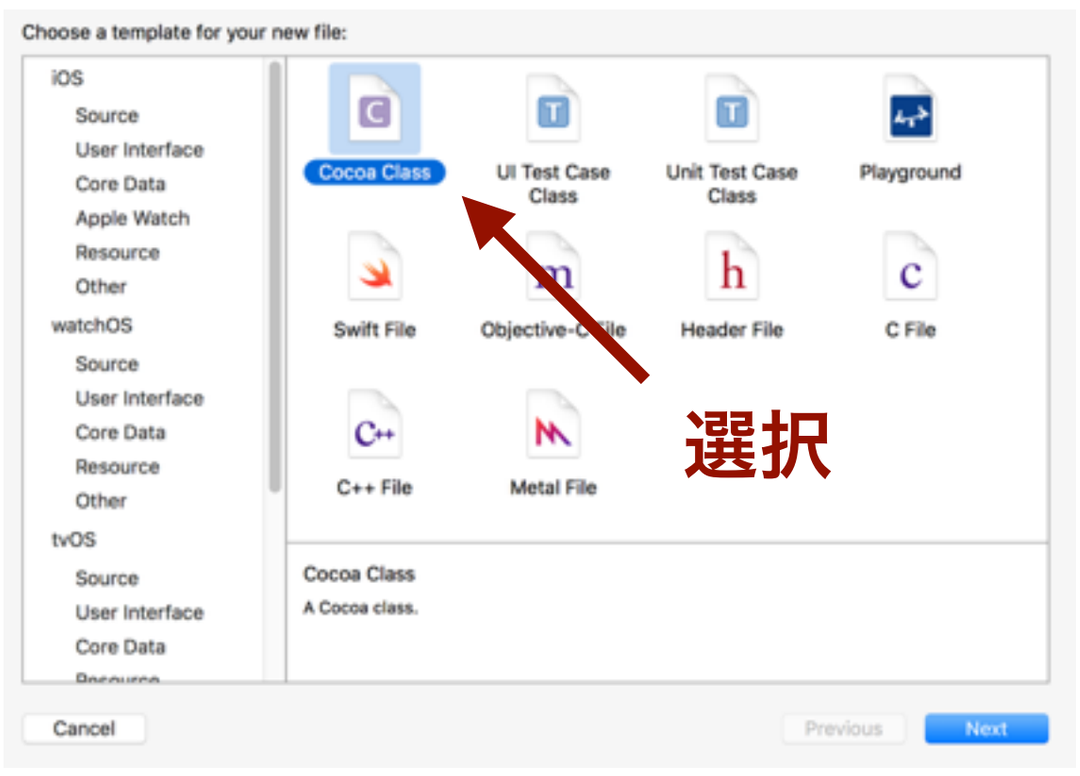


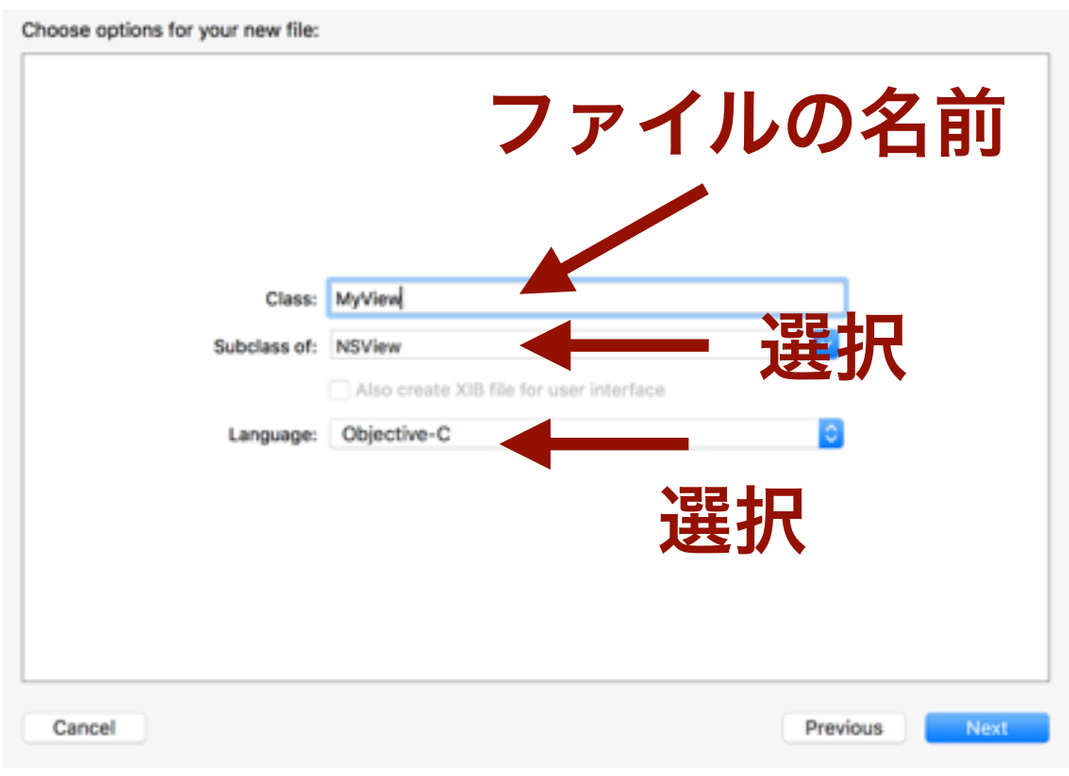
2．クラスを定義する (Xcode)

最初に必要なクラスを定義し，そのクラスファイルを作る

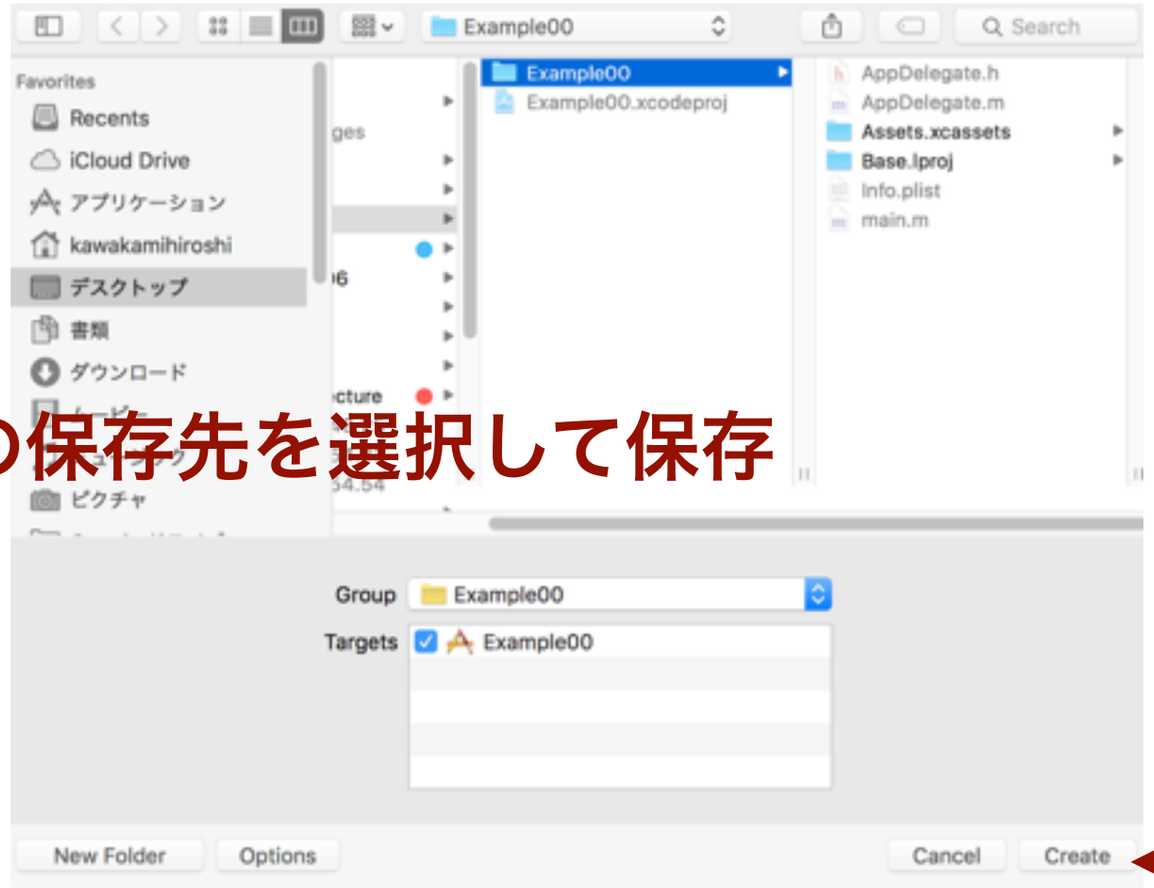
・Xcodeの File > New> File… でObjective C のクラス（例えば　MyView）を作る

MyView.h ファイルと MyView.m ファイルができる



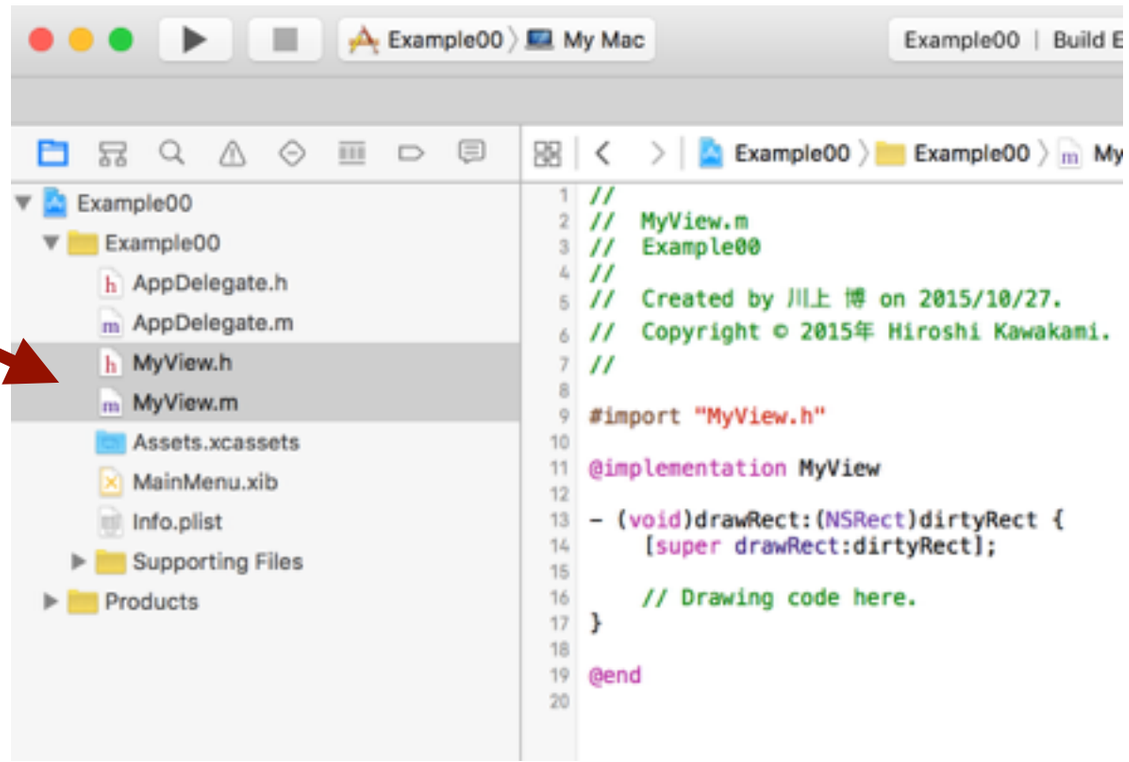


ファイルの保存先を選択して保存



選択

ファイルの保存されている

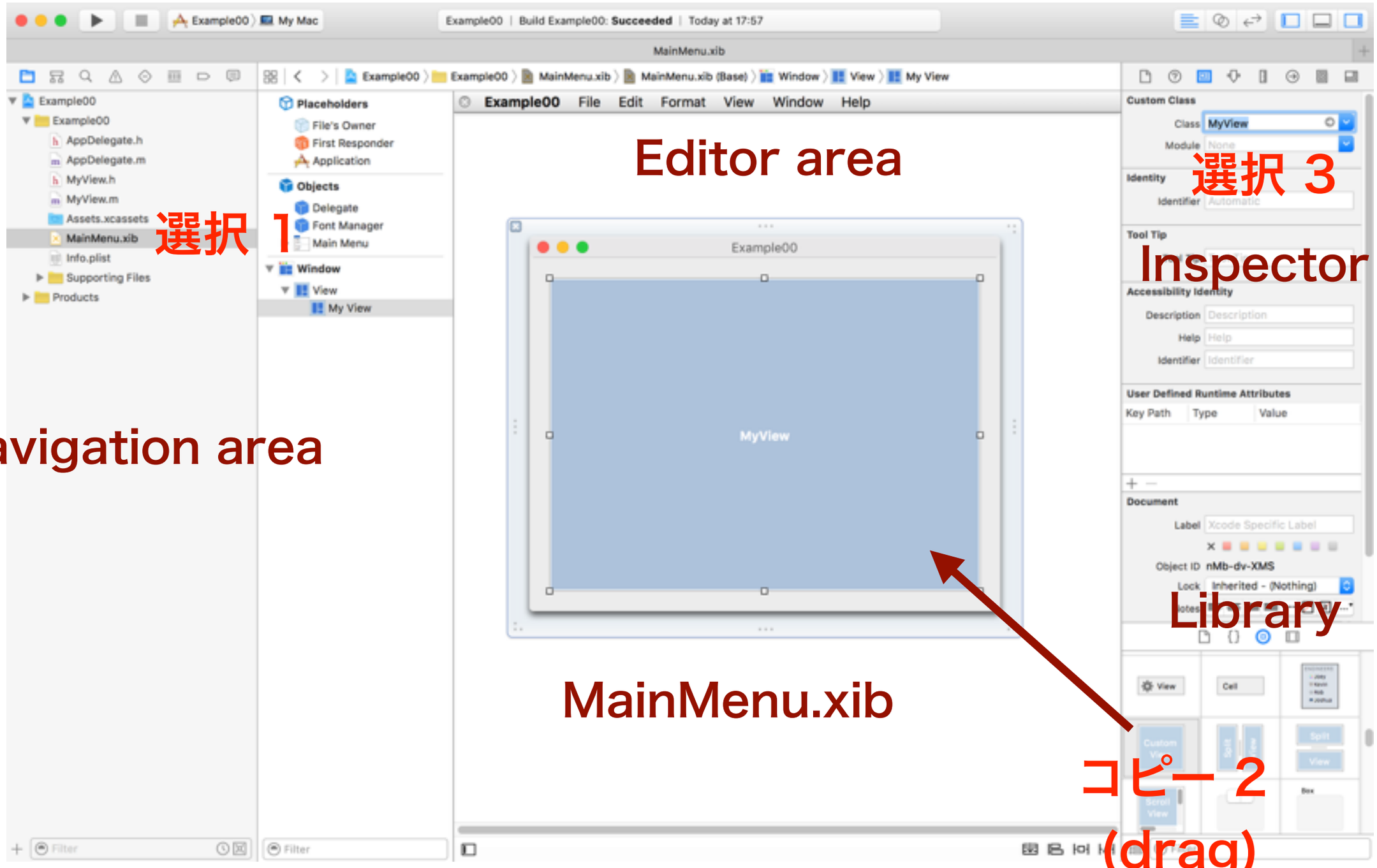


３＆４．View の作成と接続

User Interface(UI)：Custom View を配置する

- MainMenu.xib をダブルクリックして IB を立ち上げる
- ライブラリーパネルの Cocoa > View&Cells から Layout Views の Custom View を Window へ drag する
- Custom Viewを選択した状態で，InspectorのView Identity の Class よりMyViewを選択する：Custom View がMyViewのインスタンスに変わる
- MyView を選択して，Inspector でサイズを決める

User Interface(UI)：Custom　Viewを配置する



5. 定義したクラスのコードを書く (Xcode)

MyView.h

```
#import <Cocoa/Cocoa.h>

@interface MyView:NSView
{

}

@end
```

MyView.m

```
#import "MyView.h"

@implementation MyView
- (id)initWithFrame:(NSRect)frame {
    self = [super initWithFrame:frame];
    if (self) {
        // Initialization code here.
    }
    return self;
}

- (void)drawRect:(NSRect)dirtyRect {
    NSRect bounds = [self bounds];

    [[NSColor blueColor] set];
    [NSBezierPath fillRect:bounds];

    [[NSColor whiteColor] set];
    NSBezierPath *path;
    path=[NSBezierPath bezierPathWithRect:
      NSMakeRect(50, 50, 200, 200)];

    [path setLineWidth:4.0f];
    [path stroke];
}
@end
```

Cocoa でのグラフィックスの基本形

- Graphics context: NSView
- Graphics path: NSBezierPath

6. 実行 (Xcode)

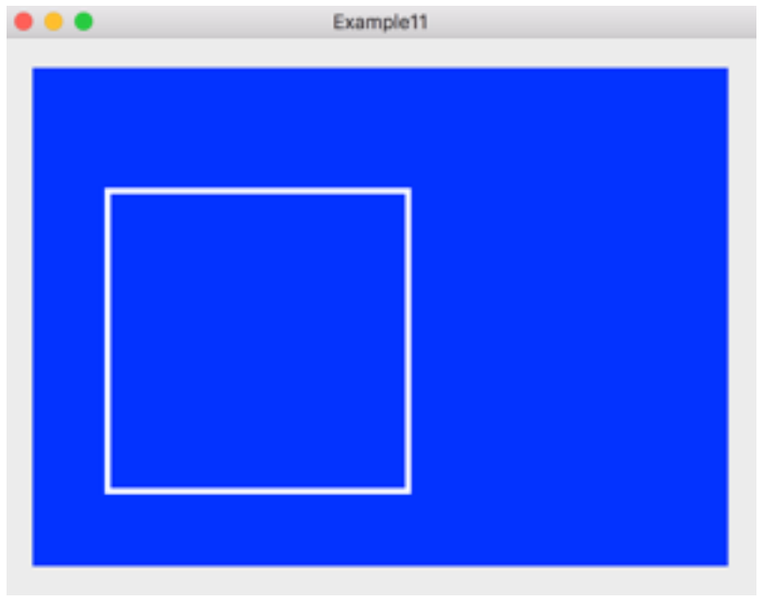
Example11 〉 My Mac | Example11 | Build Example11: **Succeeded** | Today at 12:49

MyView.m

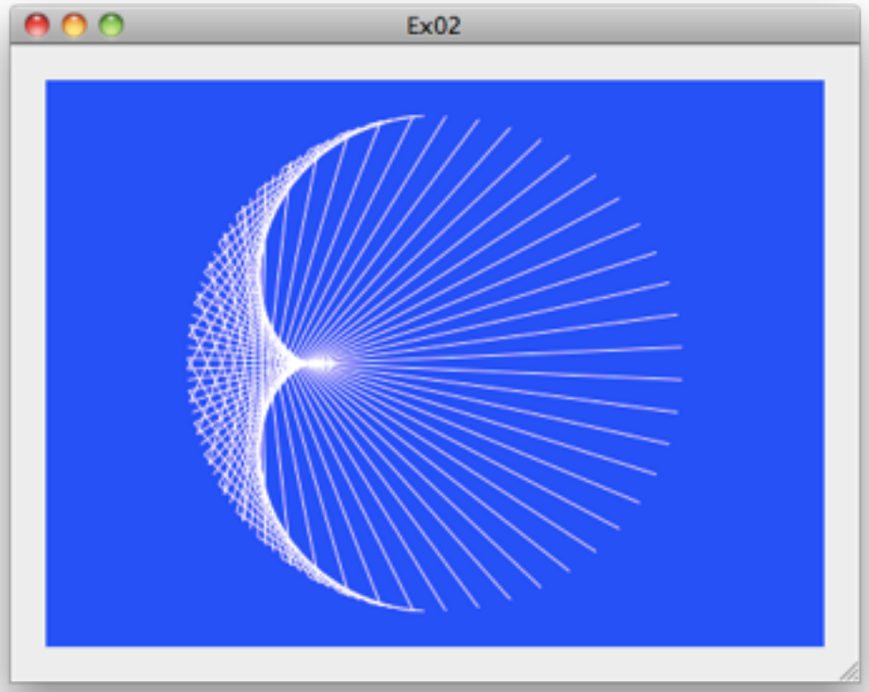
Example11 〉 Example11 〉 MyView.m 〉 -initWithFrame:

- Example11
  - Example11
    - AppDelegate.h
    - AppDelegate.m
    - MyView.h
    - MyView.m
    - Assets.xcassets
    - MainMenu.xib
    - Info.plist
  - Supporting Files
  - Products

```
1  //
2  //  MyView.m
3  //  Example11
4  //
5  //  Created by 川上 博 on 2015/10/28.
6  //  Copyright © 2015年 Hiroshi Kawakami. All rights reserved.
7  //
8
9  #import "MyView.h"
10
11 @implementation MyView
12
13 - (id)initWithFrame:(NSRect)frame {
14     self = [super initWithFrame:frame];
15     if (self) {
16         // Initialization code here.
17     }
18     return self;
19 }
20
21 - (void)drawRect:(NSRect)dirtyRect {
22     NSRect bounds = [self bounds];
23
24     [[NSColor blueColor] set];
25     [NSBezierPath fillRect:bounds];
26
27     [[NSColor whiteColor] set];
28     NSBezierPath *path;
29     path=[NSBezierPath bezierPathWithRect:NSMakeRect(50, 50, 200, 200)];
30     //  path=[NSBezierPath bezierPathWithOvalInRect:NSMakeRect(50, 50, 200, 200)];
31     [path setLineWidth:4.0f];
32     [path stroke];
33 }
34
35 @end
```



Bezier path を表示する




```
#import "MyView.h"
#include <math.h>

#define PI 3.1415926535
#define INCPI PI/71

@implementation MyView

- (id)initWithFrame:(NSRect)frame {
    self = [super initWithFrame:frame];
    if (self) {
        // Initialization code here.
    }
    return self;
}

- (void)drawRect:(NSRect)dirtyRect {
    NSRect bounds = [self bounds];

    [[NSColor blueColor] set];
    [NSBezierPath fillRect:bounds];

    [[NSColor whiteColor] set];
    NSBezierPath *path=[NSBezierPath bezierPath];
    double r, t, u, x0, y0;
    x0=bounds.size.width/2.0;
    y0=bounds.size.height/2.0;
    r=MIN(x0,y0)-20.0;
    for(t=PI/2.0; t<1.5*PI; t+=INCPI)
    {
     u=3.0*t-PI;
     [path moveToPoint:NSMakePoint(x0+r*cos(t),y0+r*sin(t))];
     [path lineToPoint:NSMakePoint(x0+r*cos(u),y0+r*sin(u))];
    }

    [path setLineWidth:1.0f];
    [path stroke];
}

@end
```

C の関数を混合して使える

```
#import "MyView.h"
#include <math.h>

#define PI 3.1415926535
#define INCPI PI/71

void caustic(NSRect bounds, NSBezierPath *path)
{
    double r, t, u, x0, y0;

    x0=bounds.size.width/2.0;
    y0=bounds.size.height/2.0;
    r=MIN(x0,y0)-20.0;
    for(t=PI/2.0; t<1.5*PI; t+=INCPI){
        u=3.0*t-PI;
        [path moveToPoint:NSMakePoint(x0+r*cos(t),y0+r*sin(t))];
        [path lineToPoint:NSMakePoint(x0+r*cos(u),y0+r*sin(u))];
    }
}

@implementation MyView
- (id)initWithFrame:(NSRect)frame {
    self = [super initWithFrame:frame];
    if (self) {
    }
    return self;
}

- (void)drawRect:(NSRect)dirtyRect {
   NSRect bounds = [self bounds];

    [[NSColor blueColor] set];
    [NSBezierPath fillRect:bounds];
    [[NSColor whiteColor] set];
    NSBezierPath *path=[NSBezierPath bezierPath];

    caustic(bounds, path);

    [path setLineWidth:1.0f];
    [path stroke];
}
@end
```

References

A. B. Altenberg, A. Clarke: Become an Xcoder

http://download.cocoalab.com.s3.amazonaws.com/BecomeAnXcoder.pdf

A. Hillegass著，堂阪真司訳:Objective-C プログラミング，Pearson, 2012

https://sites.google.com/site/objcprogbook/

S. Knaster, W. Malik and M. Dalrymple著，株式会社クイーブ監訳：
入門 Objective-C 2.0 [第２版]，SE, 2012

http://www.shoeisha.co.jp/book/detail/9784798129525

Cの構造体とクラスの差異？

```
#import <Cocoa/Cocoa.h>

@interface MyView : NSView
{
  NSColor *vColor;
  NSBezierPath *path;
}
- (NSColor *)vColor;
- (void)setVColor:(NSColor *)newColor;
- (NSBezierPath *)path;
- (void)setPath:(NSBezierPath *)newPath;

@end
```

instance variables

methods

class = struct(ivar) + functions(methods)

構造体と一緒に関数が定義できることに
よってプログラムの手法がどう変るのか

NSRect, NSSize, NSPoint

NSPoint

Abstract: Represents a point in a Cartesian coordinate system.

Declaration:
```
typedef struct _NSPoint {
   CGFloat x;
   CGFloat y;
} NSPoint;
```

サンプルコード: Aperture Edit Plugin – Borders & Titles

利用できるかどうか: Mac OS X 10.0 and later

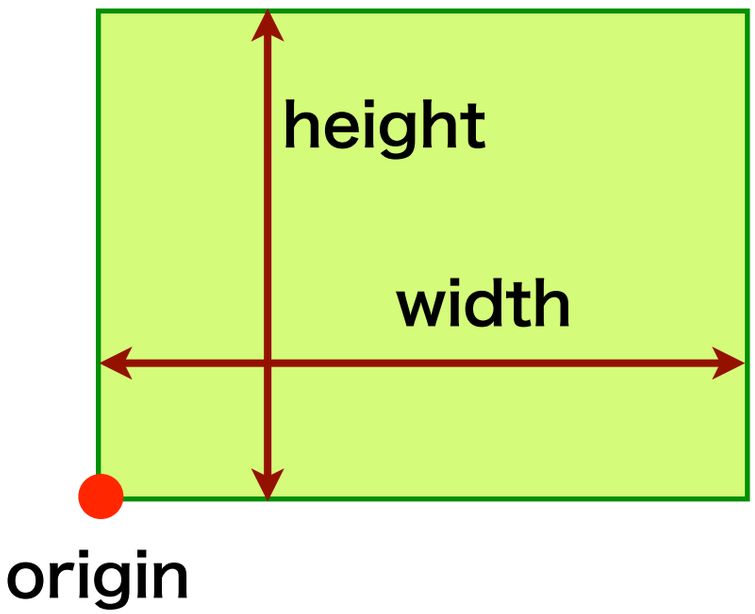
NSSize

Abstract: Represents a two-dimensional size.

Declaration:
```
typedef struct _NSSize {
   CGFloat width;
   CGFloat height;
} NSSize;
```

サンプルコード: Aperture Edit Plugin – Borders & Titles, CocoaAUHost, ScannerBrowser

利用できるかどうか: Mac OS X 10.0 and later



NSRect

Abstract: Represents a rectangle.

Declaration:
```
typedef struct _NSRect {
   NSPoint origin;
   NSSize size;
} NSRect;
```

サンプルコード: Aperture Edit Plugin – Borders & Titles, CocoaAUHost, ScannerBrowser

利用できるかどうか: Mac OS X 10.0 and later

Cocoa Graphics (Objective C)

```
- (void)drawRect:(NSRect)dirtyRect{
    NSRect bounds = [self bounds];
    double r, t, u, x0, y0;

    [[NSColor blueColor] set];
    [NSBezierPath fillRect:bounds];

    [[NSColor whiteColor] set];
    NSBezierPath *path=[NSBezierPath bezierPath];
    x0=bounds.size.width/2.0;
    y0=bounds.size.height/2.0;
    r=MIN(x0,y0)-20.0;
    for(t=PI/2.0; t<1.5*PI; t+=INCPI){
     u=3.0*t-PI;
     [path moveToPoint:NSMakePoint(x0+r*cos(t),y0+r*sin(t))];
     [path lineToPoint:NSMakePoint(x0+r*cos(u),y0+r*sin(u))];
    }
    [path setLineWidth:1.0f];
    [path stroke];
}
```

Core Graphics (C based)

```
- (void)drawRect:(NSRect)rect{
//    NSGraphicsContext *nsgc = [NSGraphicsContext currentContext];
//    CGContextRef myContext = [nsgc graphicsPort];
    CGContextRef myContext=[[NSGraphicsContext currentContext] graphicsPort];
    double r, t, u, x0, y0;

    CGContextSetRGBFillColor(myContext, 1, 1, 1, 1);
    CGContextFillRect(myContext, NSRectToCGRect(rect));

    CGContextSaveGState(myContext);
    CGContextSetRGBStrokeColor(myContext,0,0,1,1);
    x0=rect.size.width/2.0;
    y0=rect.size.height/2.0;
    r=MIN(x0,y0)-20.0;
    for(t=PI/2.0; t<1.5*PI; t+=INCPI){
     u=3.0*t-PI;
     CGContextMoveToPoint(myContext,x0+r*cos(t),y0+r*sin(t));
     CGContextAddLineToPoint(myContext, x0+r*cos(u),y0+r*sin(u));
    }
    CGContextStrokePath(myContext);
    CGContextRestoreGState(myContext);
}
```

Programming：Algorithm + Data Structure + User Interface

Policy：Reuserable programming elements

Algorithm : Object 間の接続グラフの構成