CS193P - Lecture 4

iPhone Application Development

Building an Application	アプリをビルドする
Model, View, Controller	モデル=ビュー=コントローラ（MVC）
Nib Files	nibファイル（xibファイル）
Controls and Target-Action	コントロールとターゲット=アクション

Announcements おしらせ（省略）

- Friday sections
  - Friday, 4-5: 260-113

- Invites to Developer Program will go out this weekend
  - Sign up and get your certificate when you get it
  - Start making apps that will run on Hardware!!

- Waiting for a couple students to reply about P/NC spots
  - If we don't hear back today, we're giving them away

Today's Topics 今日のトピックス

- Application Lifecycle アプリの一生
- Model, View, Controller design MVCデザイン（パターン）
- Interface Builder and Nib Files IBと nibファイル
- Controls and Target-Action コントロールとターゲット=アクション
- HelloPoly demo HelloPoly（宿題アプリ）デモ

Review おさらい

Memory Management メモリ管理

- Alloc/Init （メモリの）確保を初期化
  - -alloc assigns memory; -init sets up the object
  - Override -init, not -alloc
    上書き定義するなら -init

-alloc でメモリを割り当て
-init でオブジェクトに仕立てる

- Retain/Release retain（保持）/ release（解放）
  - Increment and decrement retainCount retainCount を +1 / -1
  - When retainCount is 0, object is deallocated
  - Don’t call -dealloc!
    -dealloc を（直接）呼ばない
    （例外はカスタムクラスを作ったとき）

retainCount が 0 になったら
オブジェクトは（自動的に）

- Autorelease オートリリース
  - Object is released when run loop completes

（autorelease指定の）オブジェクトは
run-loop（イベント処理のループ１回分）が終わるごとに release

セッタ　ゲッタ　プロパティ

Setters, Getters, and Properties

標準書式（名前づけの慣習）

- Setters and Getters have a standard format:

```
- (int)age;
- (void)setAge:(int)age;
```

ゲッタはインスタンス変数名そのまま

セッタは set変数名 とする

value

- Properties allow access to setters and getters through dot syntax:

プロパティによってドット記法でセッタ/ゲッタを呼び出せる

```
@property age;

int theAge = person.age;
person.age = 21;
```

int theAge = [person age];

[person setAge:21]

Building an Application

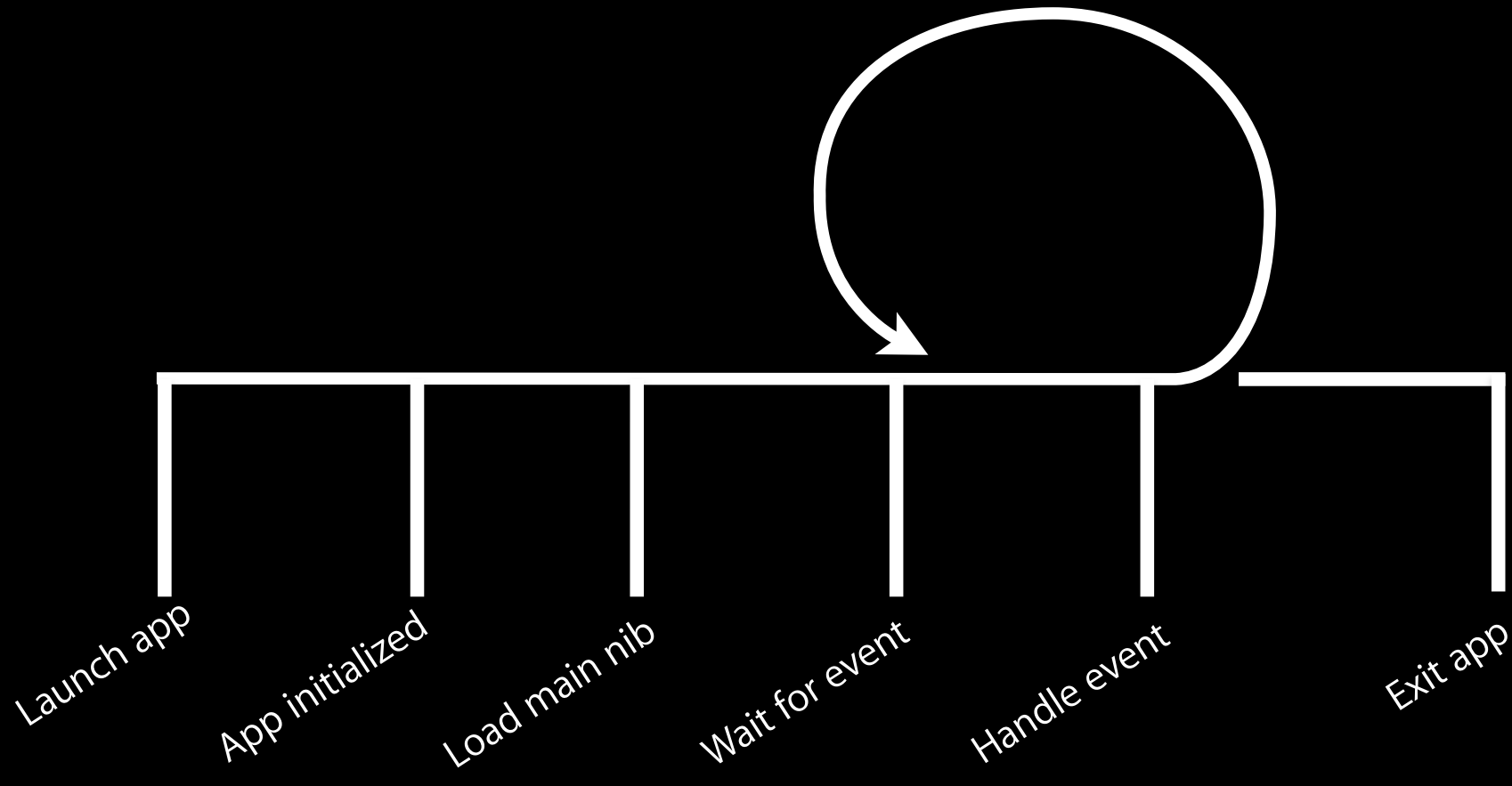
アプリをビルドする

解剖
Anatomy of an Application アプリの解剖

- Compiled code コンパイルされたコード
  - Your code あなたの（書いた）コード
  - Frameworks フレームワーク（UIKit や Foundation 等）
- Nib files nibファイル（xibファイル）
  - UI elements and other objects UI 部品や他のオブジェクト
  - Details about object relationships オブジェクト間のつながりの詳細
- Resources (images, sounds, strings, etc)
  リソース（画像，音，文字列など）
- Info.plist file (application configuration)
  info.plist（アプリ構成情報）

アプリの一生

App Lifecycle



UIKit Framework UIKit フレームワーク

- Provides standard interface elements 標準インターフェス部品を提供（ボタン，スライダなど）
- UIKit and you
  - Don’t fight the frameworks
  - Understand the designs and how you fit into them

UIKit と あなた（は仲良くしよう）
・格闘しないように（素直に使おう）．
・デザインを理解して，うまく使おう．

UIKit Framework

- Starts your application あなたのアプリを起動
- Every application has a single instance of UIApplication
  - Singleton design pattern あなたのアプリは UIApplication クラスのただ一つのインスタンス

```
@interface UIApplication
+ (UIApplication *)sharedApplication
@end
```

あなたのアプリは [UIApplication sharedApplication]; で取得できる.

  - Orchestrates the lifecycle of an application
  - Dispatches events
  - Manages status bar, application icon badge
  - Rarely subclassed
    - Uses delegation instead

・アプリの一生を指揮
・イベントをさばく
・…などを管理する
・サブクラス化はめったにしない（代わりに delegation する）

Delegation デリゲーション（委託：[誰かに]仕事を頼むこと）

- Control passed to delegate objects to perform application-specific behavior
- Avoids need to subclass complex objects
- Many UIKit classes use delegates
  - UIApplication
  - UITableView
  - UITextField

※デリゲーションについての詳細は後日扱います.

UIApplicationDelegate

- Xcode project templates have one set up by default
- Object you provide that participates in application lifecycle
- Can implement various methods which UIApplication will call
- Examples:

※デリゲーションについての詳細は後日扱います.

UIApplicationDelegate

- Xcode project templates have one set up by default
- Object you provide that participates in application lifecycle
- Can implement various methods which UIApplication will call
- Examples:

```
- (void)applicationDidFinishLaunching:(UIApplication *)application;
- (void)applicationWillTerminate:(UIApplication *)application;

- (void)applicationWillResignActive:(UIApplication *)application;
- (BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url;

- (void)applicationDidReceiveMemoryWarning:(UIApplication *)application;
```

※デリゲーションについての詳細は後日扱います.

Info.plist file

属性リスト　記述する

- Property List (often XML), describing your application
  - Icon appearance　アイコンの見せ方
  - Status bar style (default, black, hidden)　ステータスバーのスタイル
  - Orientation　向き（ポートレート / ランドスケープ）
  - Uses Wifi networking　Wifi（無線LAN）の使用
  - System Requirements　システム要件（例. iPad専用）

- Can edit most properties in Xcode　info.plist は Xcode で編集可
  - Project > Edit Active Target “Foo” menu item
  - On the properties tab

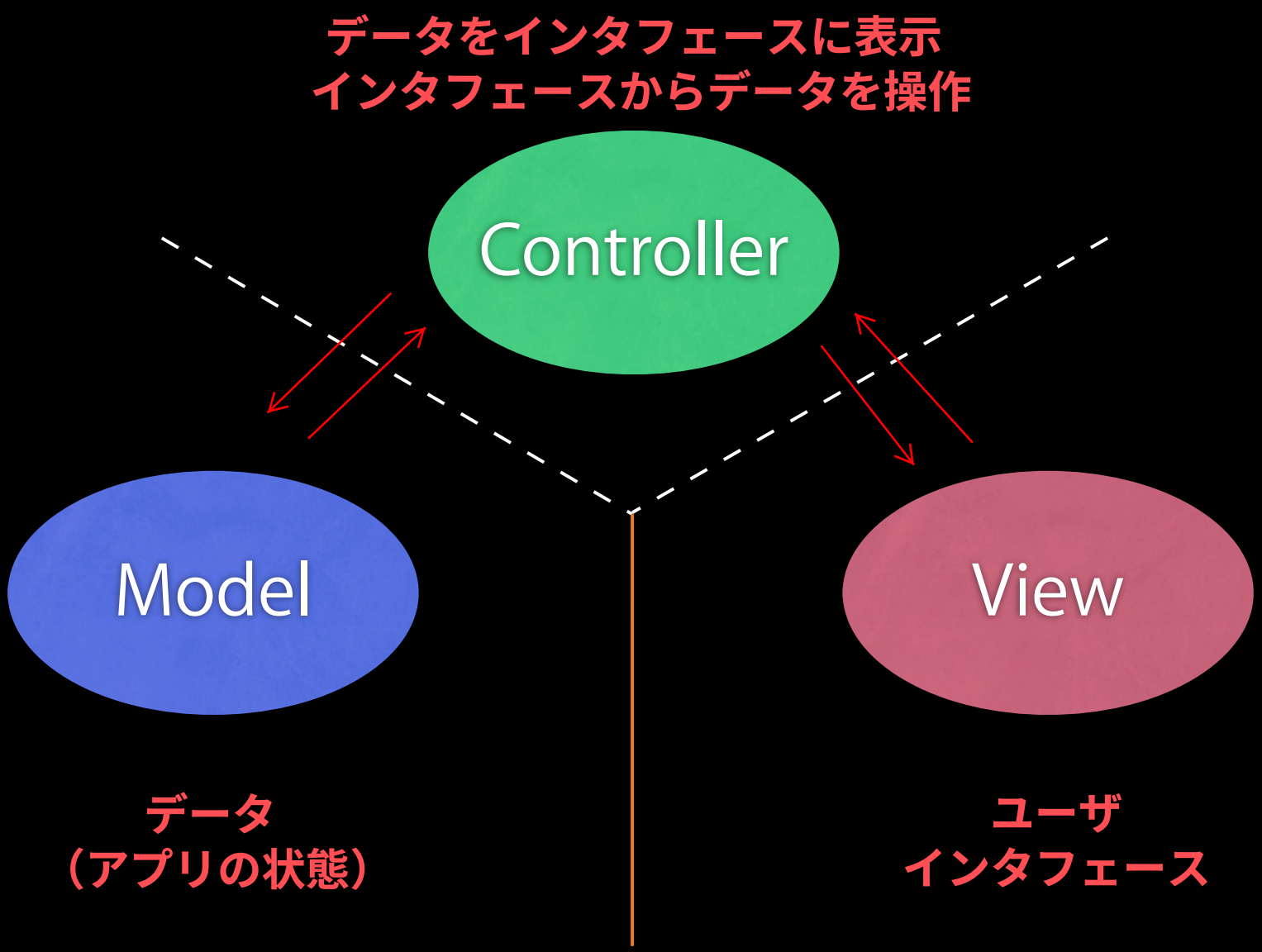
Model, View, Controller

モデル=ビュー=コントローラ（というデザインパターン）

If you take nothing else away from this class...

この授業から得るものが何もなくても…（これだけは学んで帰ってね）

Model, View, Controller



Model

- Manages the app data and state
  アプリのデータと状態を管理
- Not concerned with UI or presentation
  ユーザインタフェース（画面表示）のことは考えない
- Often persists somewhere
  どこかに「保存」されることが多い（データベース等）
- Same model should be reusable, unchanged in different interfaces

  同じモデルが異なるインタフェースでそのまま再利用されるべき

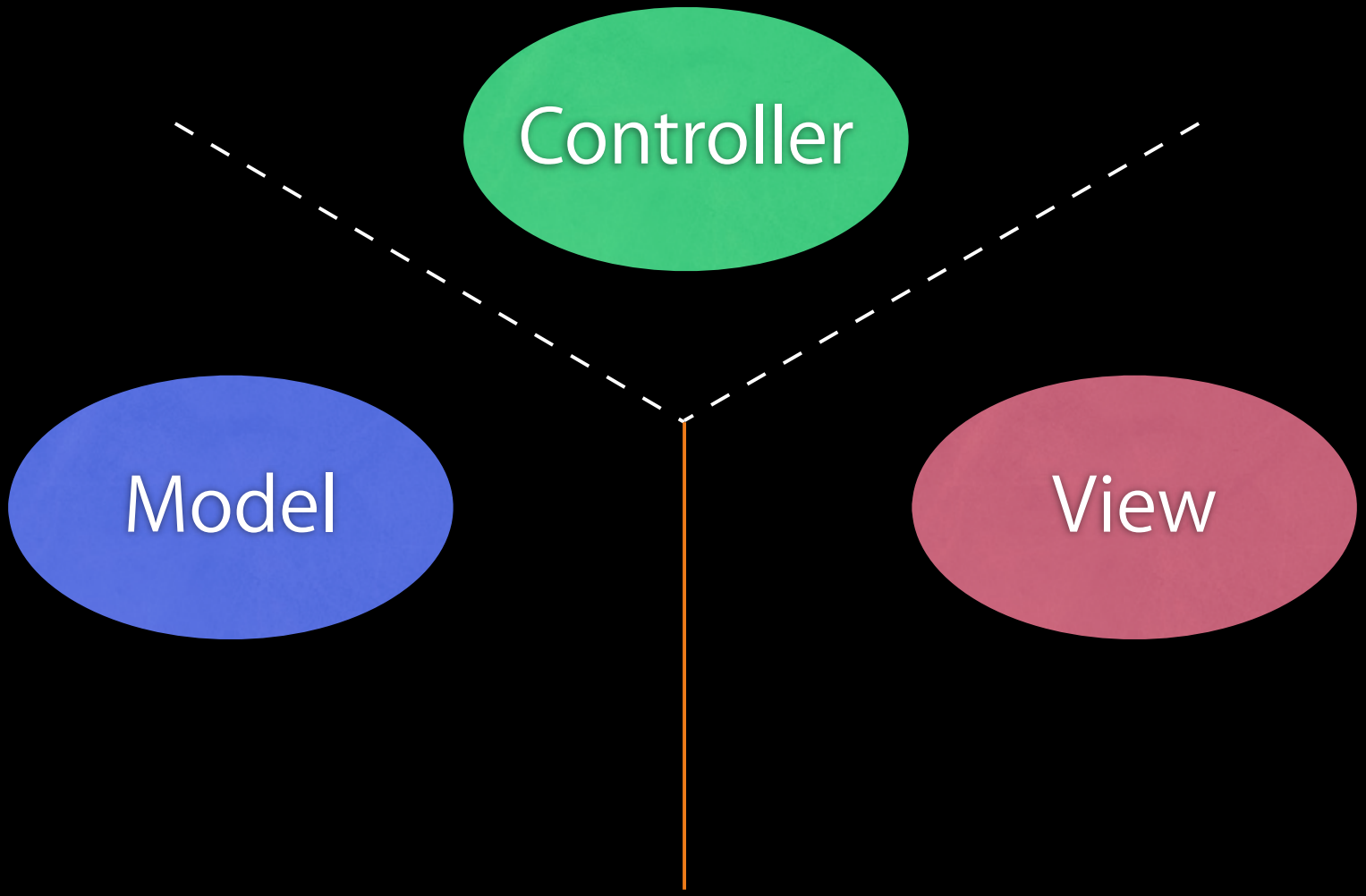
View

- Present the Model to the user in an appropriate interface
  （ビューは）モデルをユーザに提示する（それに適したインタフェースで）
- Allows user to manipulate data
  ユーザがデータを操作できるようにする
- Does not store any data データを保持しない
  - (except to cache state) （状態キャッシュを除く）
- Easily reusable & configurable to display different data
  再利用・調整が容易

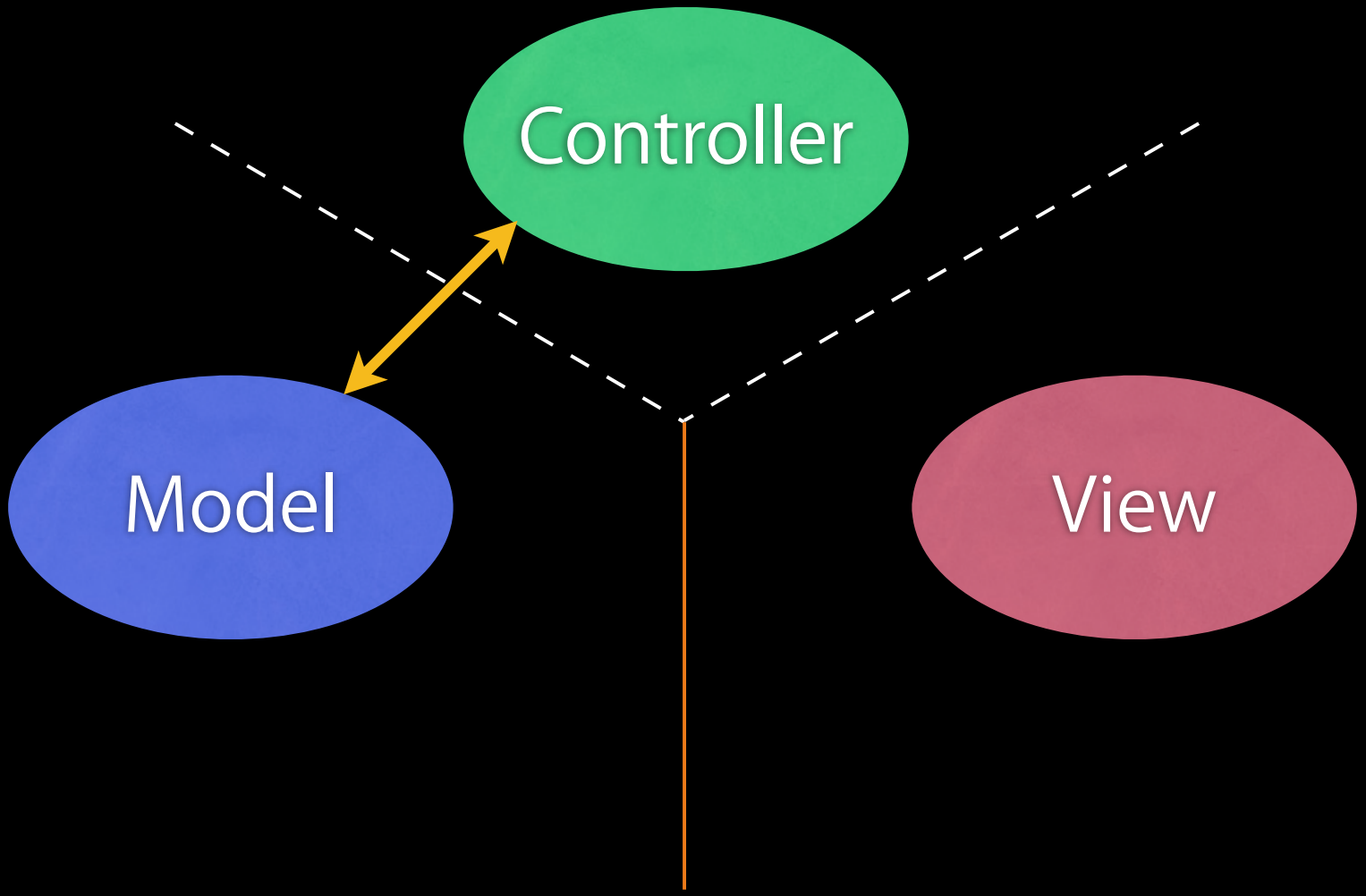
Controller

- Intermediary between Model & View
  モデルとビューの仲介役
- Updates the view when the model changes
  モデルが変化したらビューを更新する
- Updates the model when the user manipulates the view
  ユーザがビューを操作したらモデルを更新する
- Typically where the app logic lives.
  通常は，アプリの「動作ロジック」が宿るところ

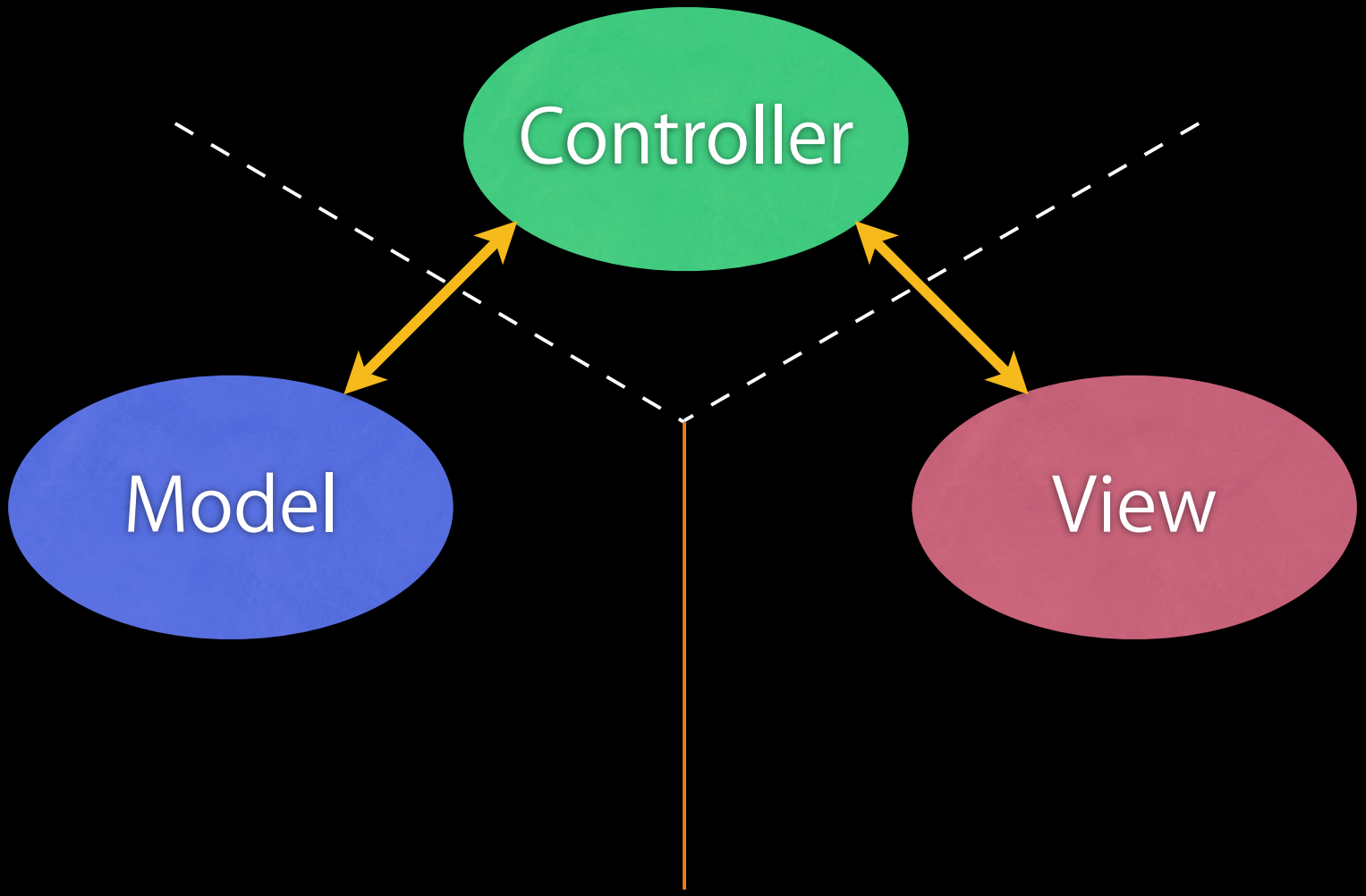
Model, View, Controller



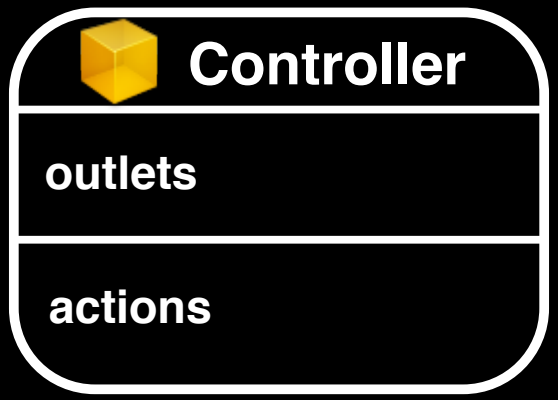
Model, View, Controller



Model, View, Controller

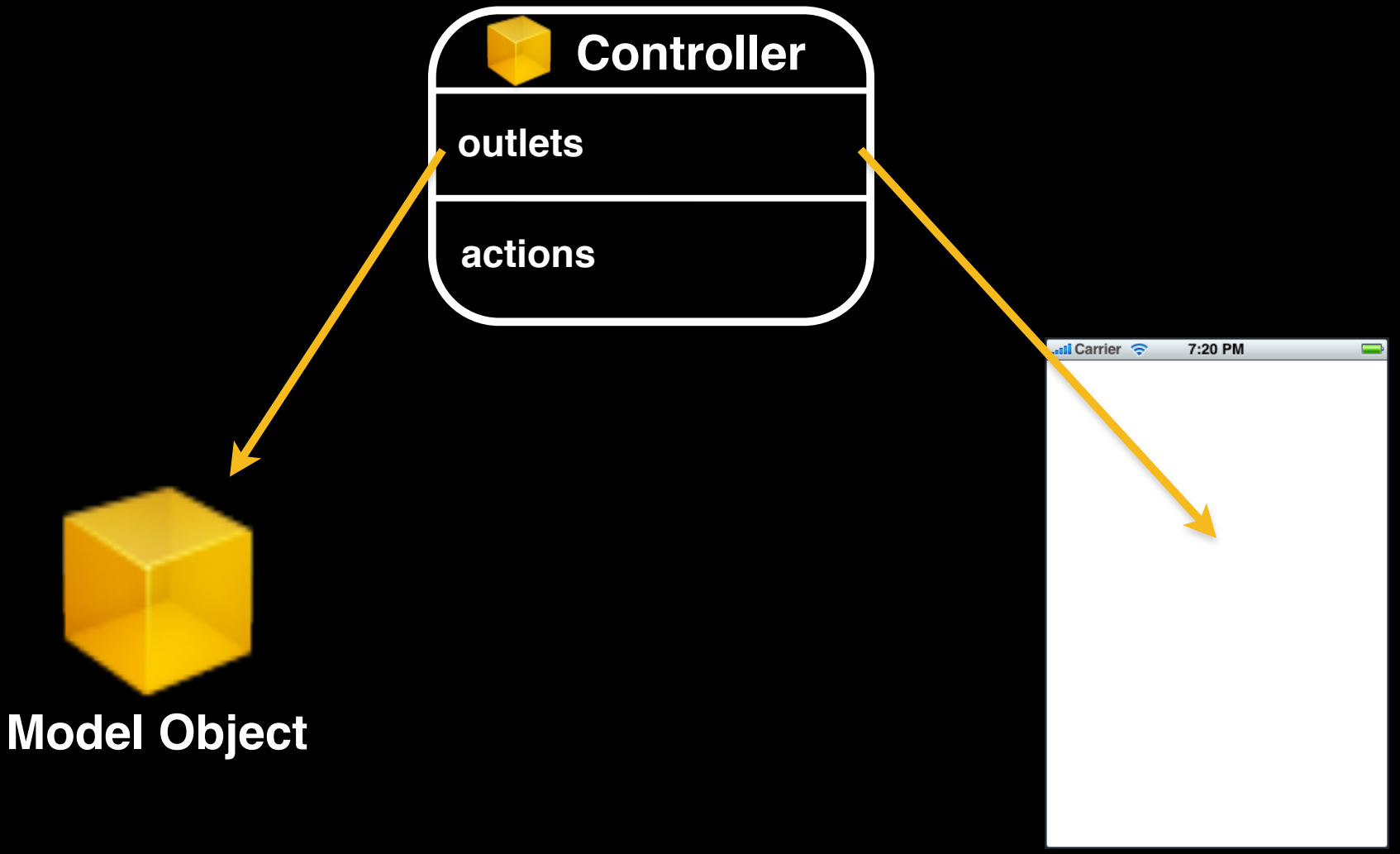


Model, View, Controller

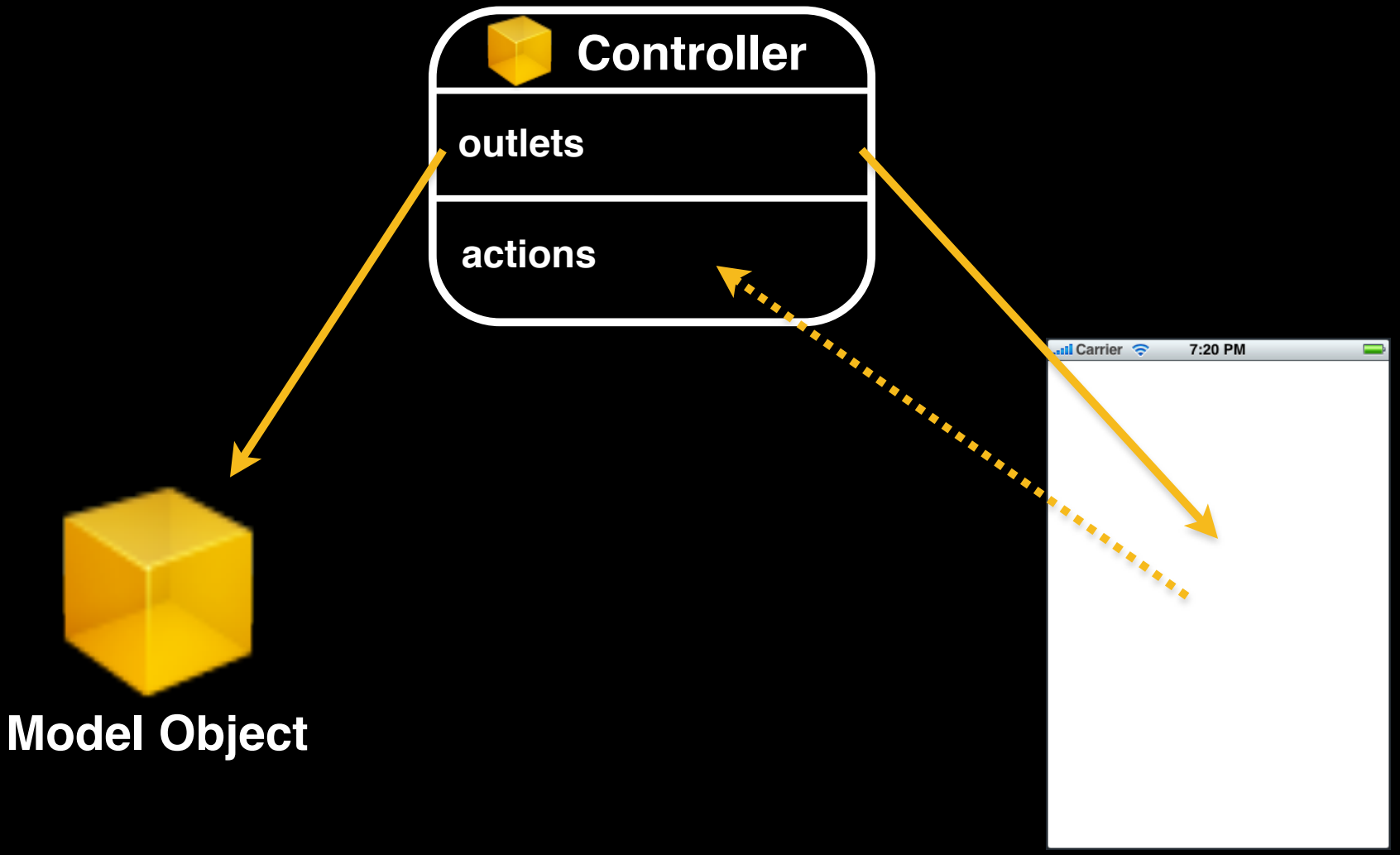


Model Object

Model, View, Controller

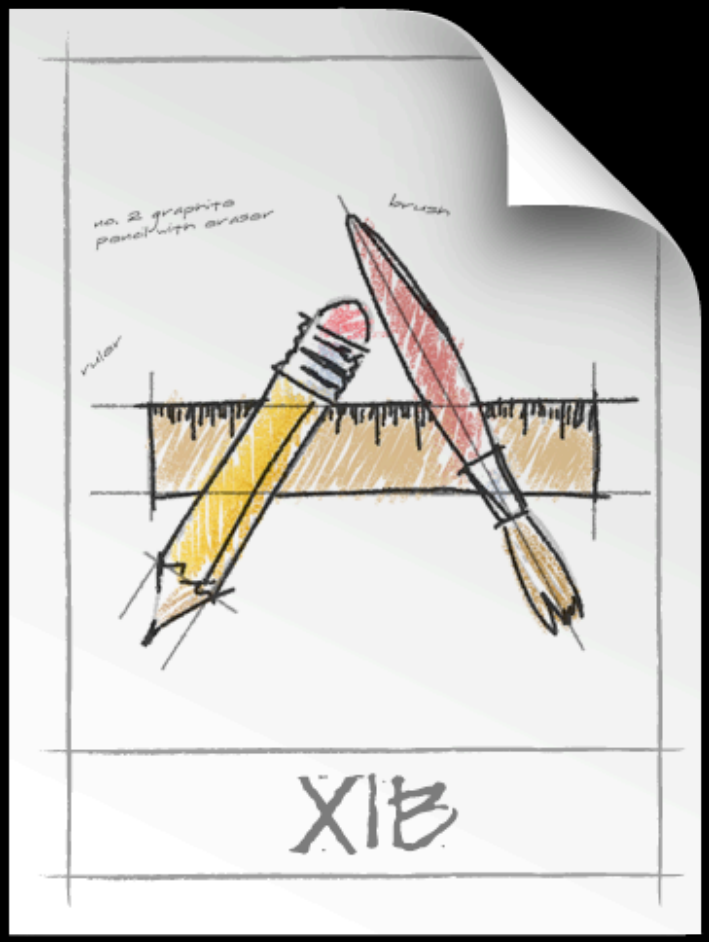


Model, View, Controller



Interface Builder and Nibs

Nib files nibファイル（xibファイル）



Nib Files - Design time

- Helps you design the 'V' in MVC:
  - layout user interface elements　UI部品を配置する
  - add controller objects　コントローラを加える
  - Connect the controller and UI　コントローラとUI部品をつなげる

Nib Loading nib/xibファイルのロード

- At runtime, objects are unarchived 実行時に展開される
  - Values/settings in Interface Builder are restored IBでの設定値が復元
  - Ensures all outlets and actions are connected outlet/action 接続確認
  - Order of unarchiving is not defined 展開される順序は不定

- If loading the nib automatically creates objects and order is undefined, how do I customize? 不定順序で展開されるオブジェクトをカスタマイズするには…
  - For example, to displaying initial state

    たとえばアプリ初期状態を表示するには…
    （例．現在時刻を表示するとか）

-awakeFromNib

- Control point to implement any additional logic after nib loading　nib/xibのロード（と展開）の後で追加処理を入れるところ
- Default empty implementation on NSObject　いつもは空っぽ
- You often implement it in your controller class　コントローラの中で実装（上書き）する
  - e.g. to restore previously saved application state
- Guaranteed everything has been unarchived from nib, and all connections are made before `-awakeFromNib` is called

```
- (void)awakeFromNib {
    // do customization here
}
```

nib/xib が展開され，すべてが接続されてから，
-awakeFromNib が呼ばれる

View-based アプリの場合は，
-viewDidLoad を使うとよい．

Controls and Target-Action

Controls - Events

コントロール＝ユーザから何らかのアクションを起動できるビュー

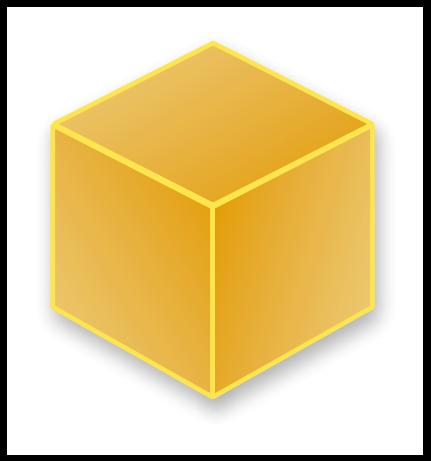
- View objects that allows users to initiate some type of action
- Respond to variety of events　さまざまなイベントに応答する
  - Touch events　タッチ イベント
    - touchDown
    - touchDragged (entered, exited, drag inside, drag outside)
    - touchUp (inside, outside)
  - Value changed　値が変化した（スライダやスイッチ等）
  - Editing events　編集イベント
    - editing began
    - editing changed
    - editing ended

Controls - Target/Action

イベントが発生したらターゲットオブジェクトでアクションを起動

- When event occurs, action is invoked on target object

Controller



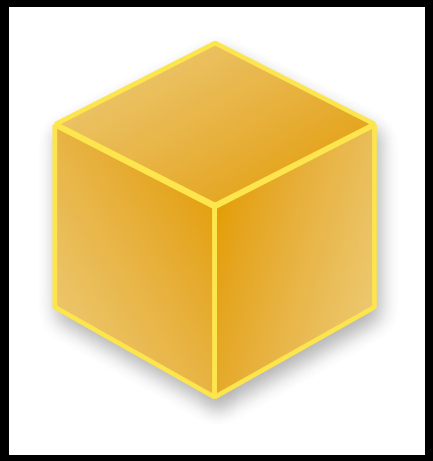
Controls - Target/Action

イベントが発生したらターゲットオブジェクトでアクションを起動

- When event occurs, action is invoked on target object


```
target:     myObject
action:     @selector(decrease)
event:      UIControlEventTouchUpInside
```

Controller



Controls - Target/Action

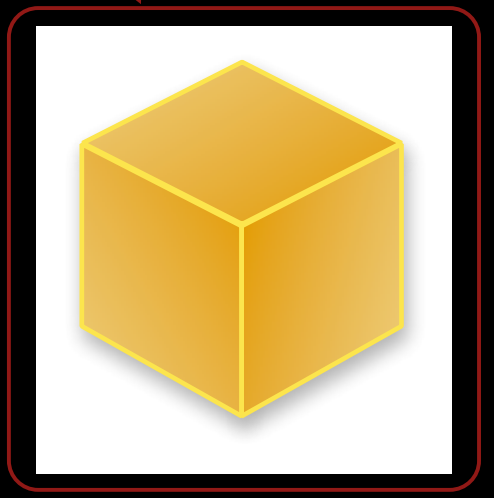
イベントが発生したらターゲットオブジェクトでアクションを起動

- When event occurs, action is invoked on target object


```
target:     myObject
action:     @selector(decrease)
event:      UIControlEventTouchUpInside
```

これが myObject
（あなたが書くプログラム）

Controller



Controls - Target/Action

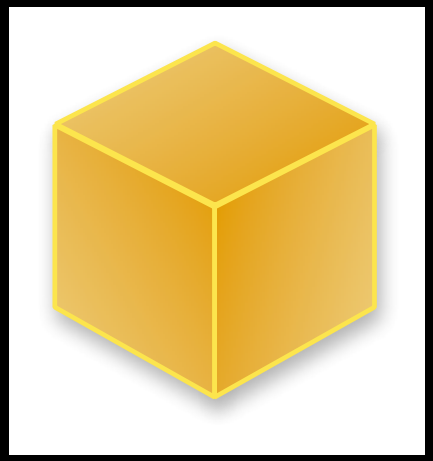
イベントが発生したらターゲットオブジェクトでアクションを起動

- When event occurs, action is invoked on target object

押す


```
target:     myObject
action:     @selector(decrease)
event:      UIControlEventTouchUpInside
```

Controller



Controls - Target/Action

イベントが発生したらターゲットオブジェクトでアクションを起動

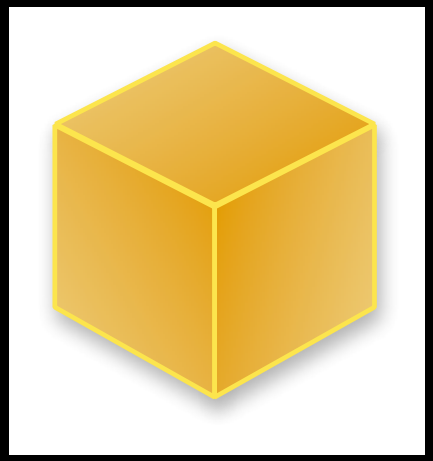
- When event occurs, action is invoked on target object


```
target:     myObject
action:     @selector(decrease)
event:      UIControlEventTouchUpInside
```

`UIControlEventTouchUpInside`

イベント発生

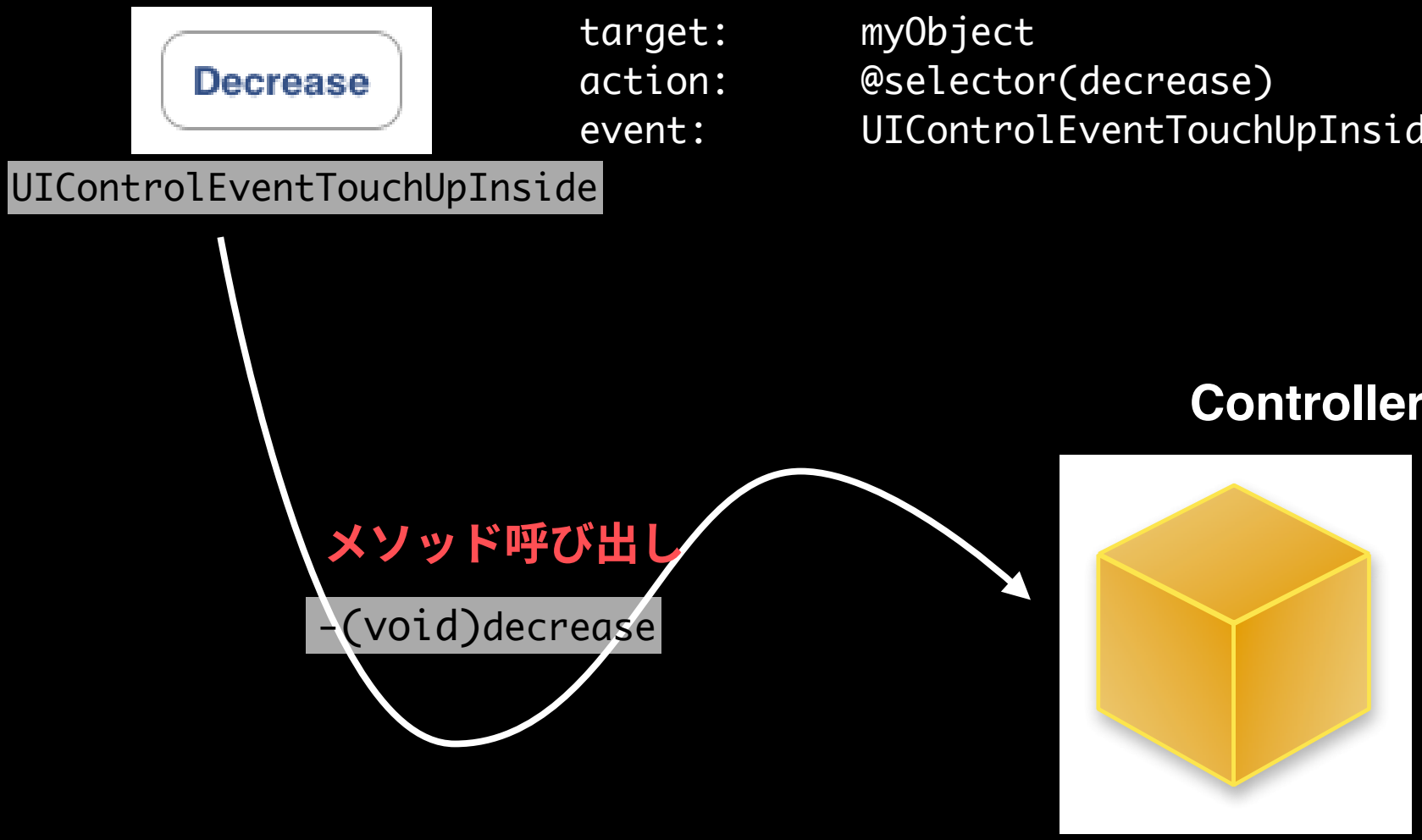
Controller



Controls - Target/Action

イベントが発生したらターゲットオブジェクトでアクションを起動

- When event occurs, action is invoked on target object



Action Methods

アクションメソッドのセレクタ型を３つのフレーバから選べる

- 3 different flavors of action method selector types

```
- (void)actionMethod;
- (void)actionMethod:(id)sender;
- (void)actionMethod:(id)sender withEvent:(UIEvent *)event;
```

sender はイベント発生源
（ボタンやスライダなど）

- UIEvent contains details about the event that took place

UIEvent には発生したイベントについての詳しい情報が入っている
（たとえばタッチ座標 (x, y) など）

Action Method Variations

- Simple no-argument selector 引数なしのセレクタの場合

```
- (void)increase {
    // bump the number of sides of the polygon up
    polygon.numberOfSides += 1;
}
```

ボタンが「押された」ことだけを捉える

- Single argument selector - control is 'sender' sender 引数ありの場合

```
// for example, if control is a slider...
- (void)adjustNumberOfSides:(id)sender {
    polygon.numberOfSides = [sender value];
}
```

その sender にメッセージを送り値をゲットしたりできる.

Action Method Variations

- Two-arguments in selector (sender & event)
  ２引数（sender & event）の場合

```
- (void)adjustNumberOfSides:(id)sender
        withEvent:(UIEvent *)event
{
    // could inspect event object if you needed to
}
```

タッチ座標，タッチした時刻など，
詳しい情報が取得できる．（詳細は UIEvent のヘルプを参照）

Multiple target-actions 複数のターゲット-アクション

- Controls can trigger multiple actions on different targets in response to the same event １つのイベントから，異なるターゲットに複数のアクションを起動させることが可能
- Different than Cocoa on the desktop where only one target-action is supported （Mac 上の）Cocoa は１組の target-action しかサポートしていない（iOS の方がスゴイ）
- Different events can be setup in IB
  IB で複数の異なるイベントを設定できる
  （もちろん，直接コードに書く方法でも可能）

Manual Target-Action

IB を使わずに，コードに書いて target-action を仕込む方法

- Same information IB would use
- API and UIControlEvents found in UIControl.h
- UIControlEvents is a bitmask

```
@interface UIControl

- (void)addTarget:(id)target action:(SEL)action
        forControlEvents:(UIControlEvents)controlEvents;

- (void)removeTarget:(id)target action:(SEL)action
        forControlEvents:(UIControlEvents)controlEvents;

@end
```

HelloPoly Demo

（宿題の）HelloPoly のデモ

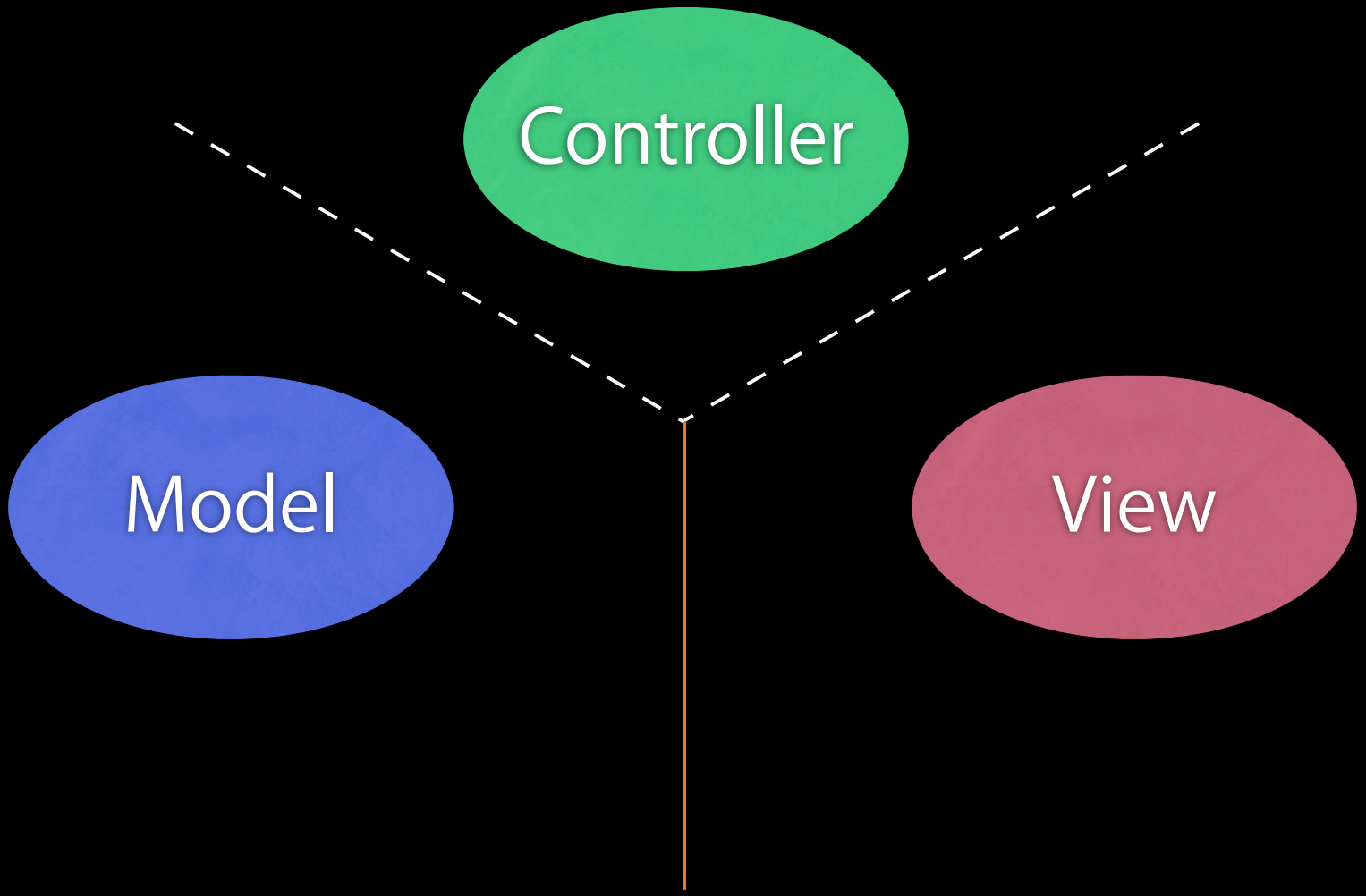
詳細については，
道場のページに「Stanford 宿題」へのリンク（赤字）があるので
そこから資料をダウンロードしてください．

HelloPoly

- This week's assignment is a full MVC application
- Next week's assignment will flesh it out further
- It is not designed to be a complex application
  - rather, provide a series of small studies of the fundamentals of a Cocoa Touch application

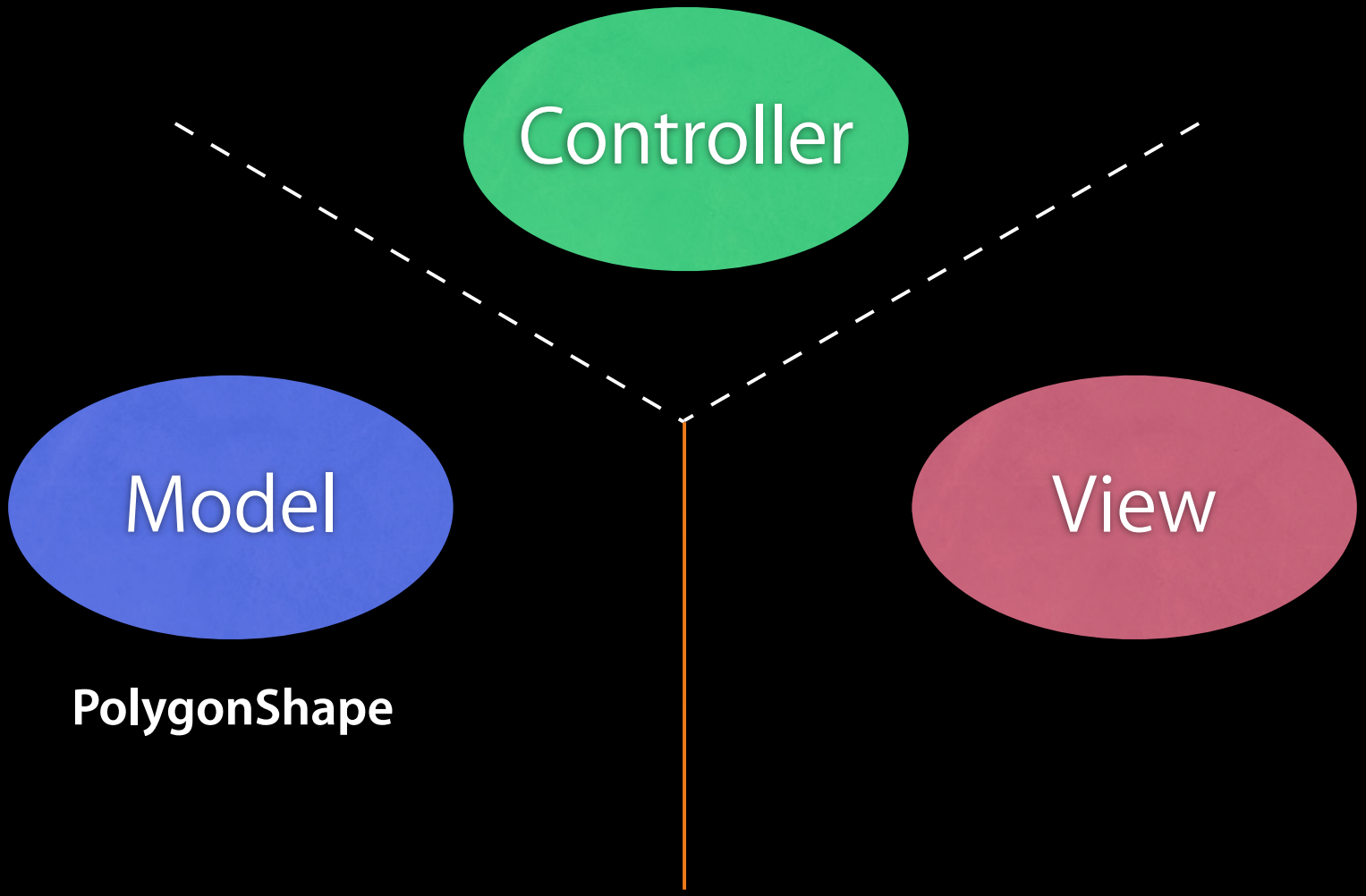
Model, View, Controller

HelloPoly



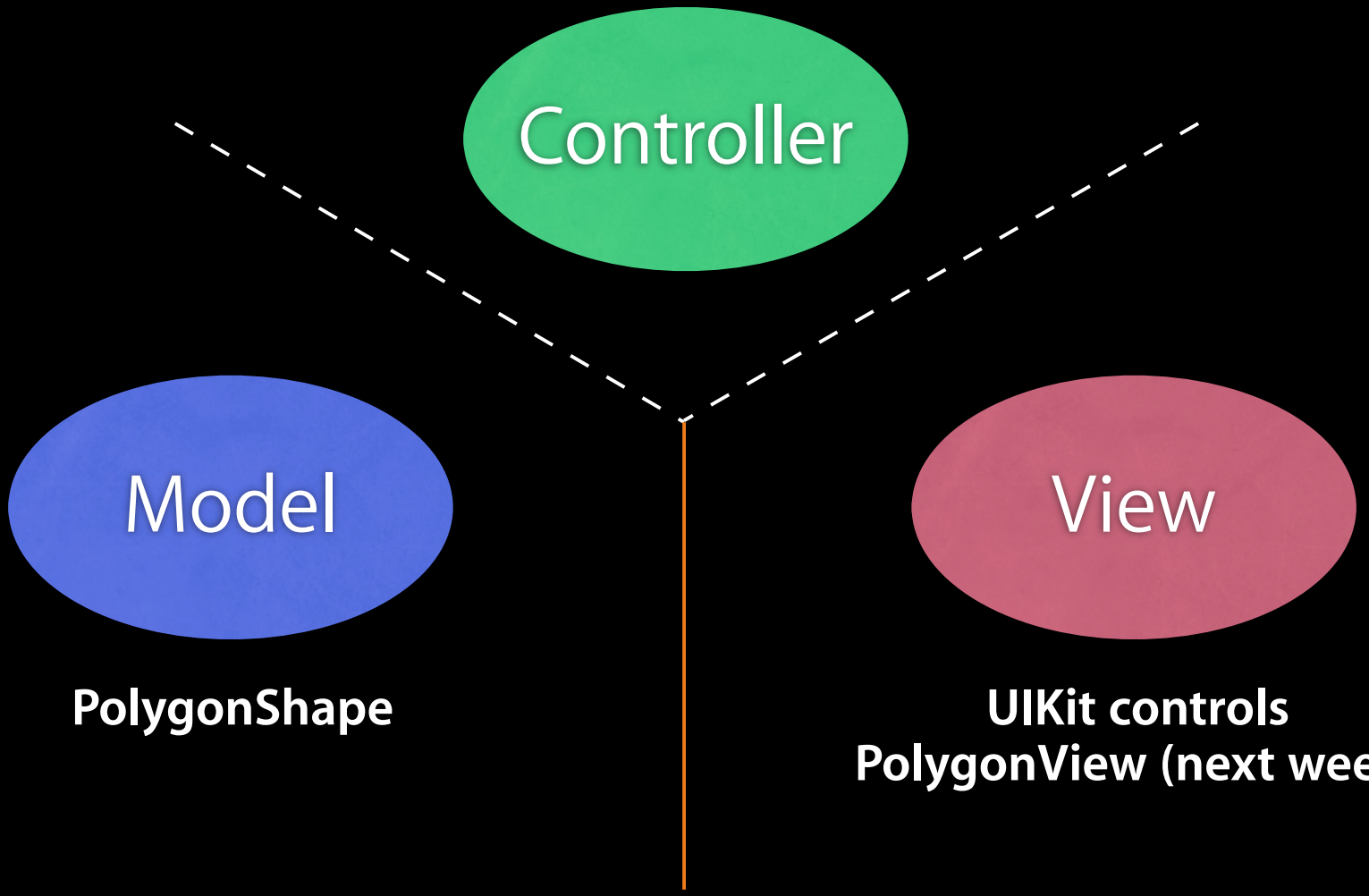
Model, View, Controller

HelloPoly



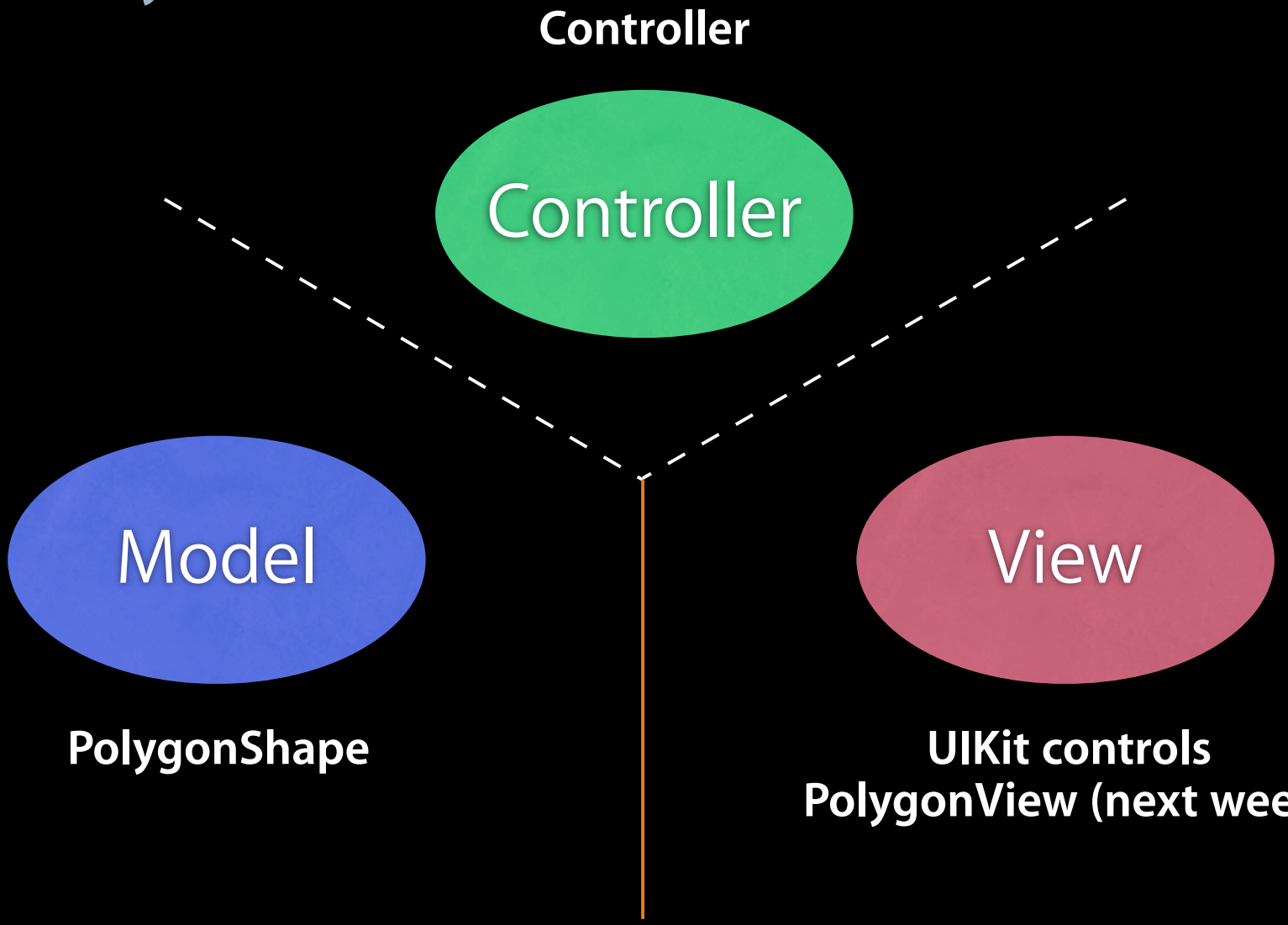
Model, View, Controller

HelloPoly



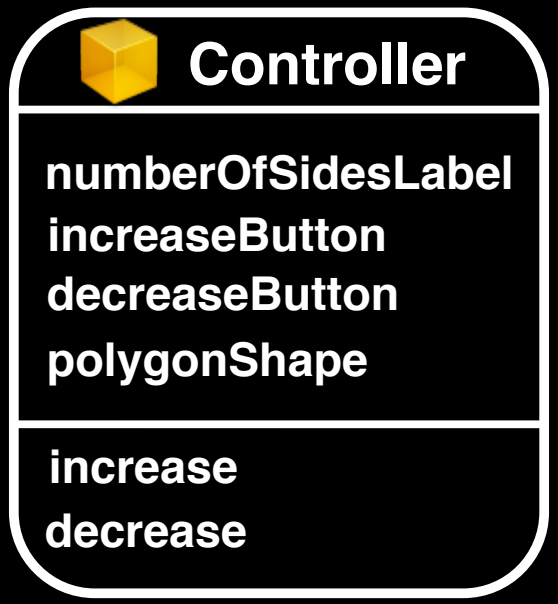
Model, View, Controller

HelloPoly



Model, View, Controller

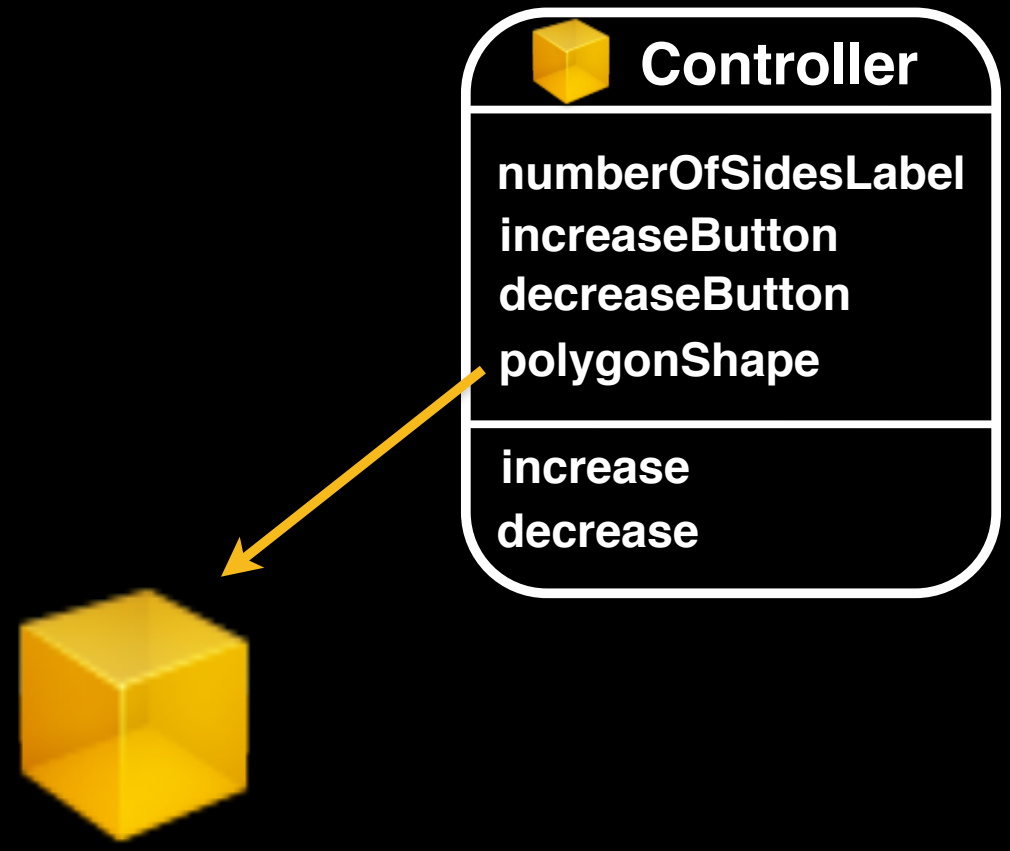
HelloPoly



PolygonShape

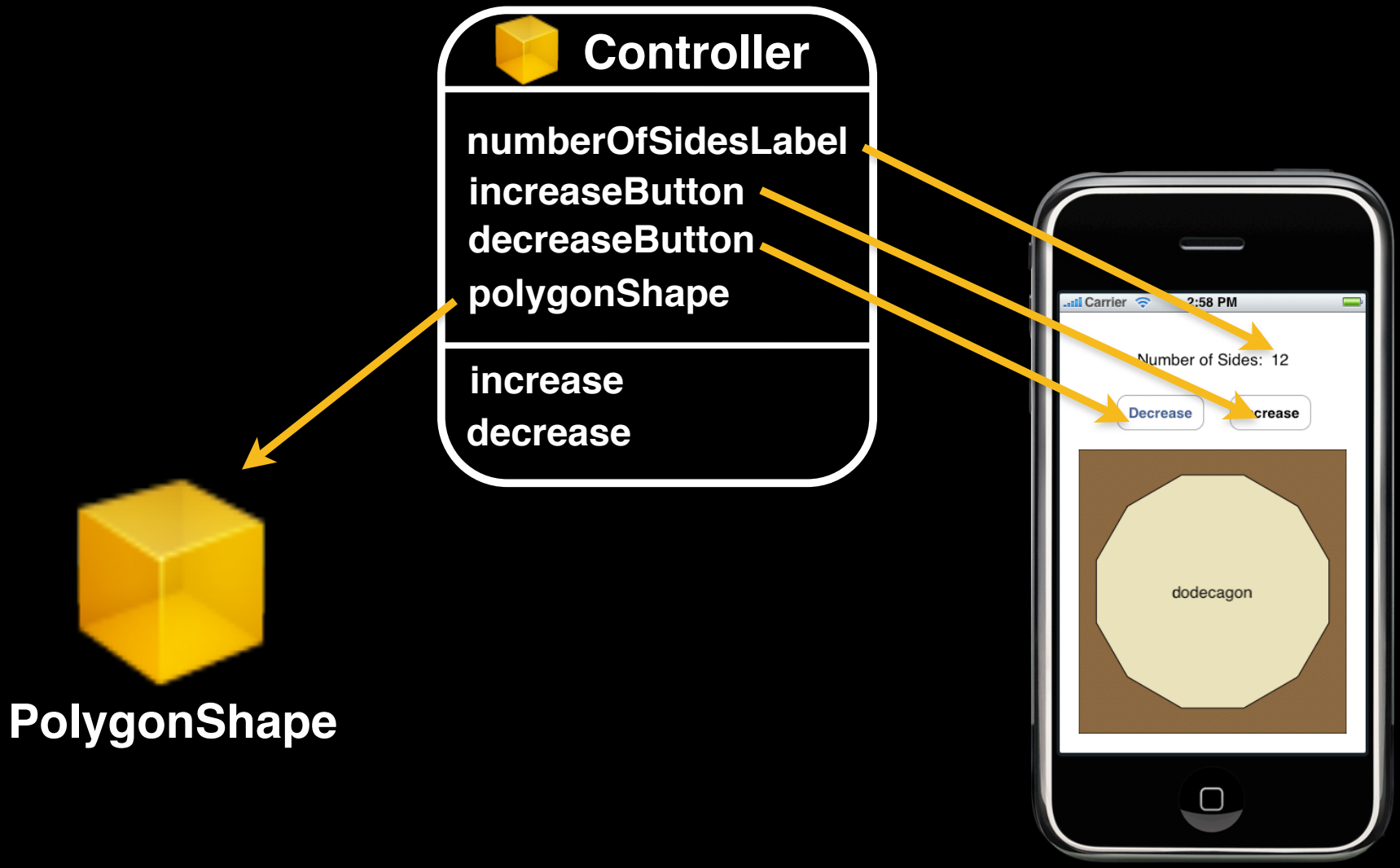
Model, View, Controller

HelloPoly



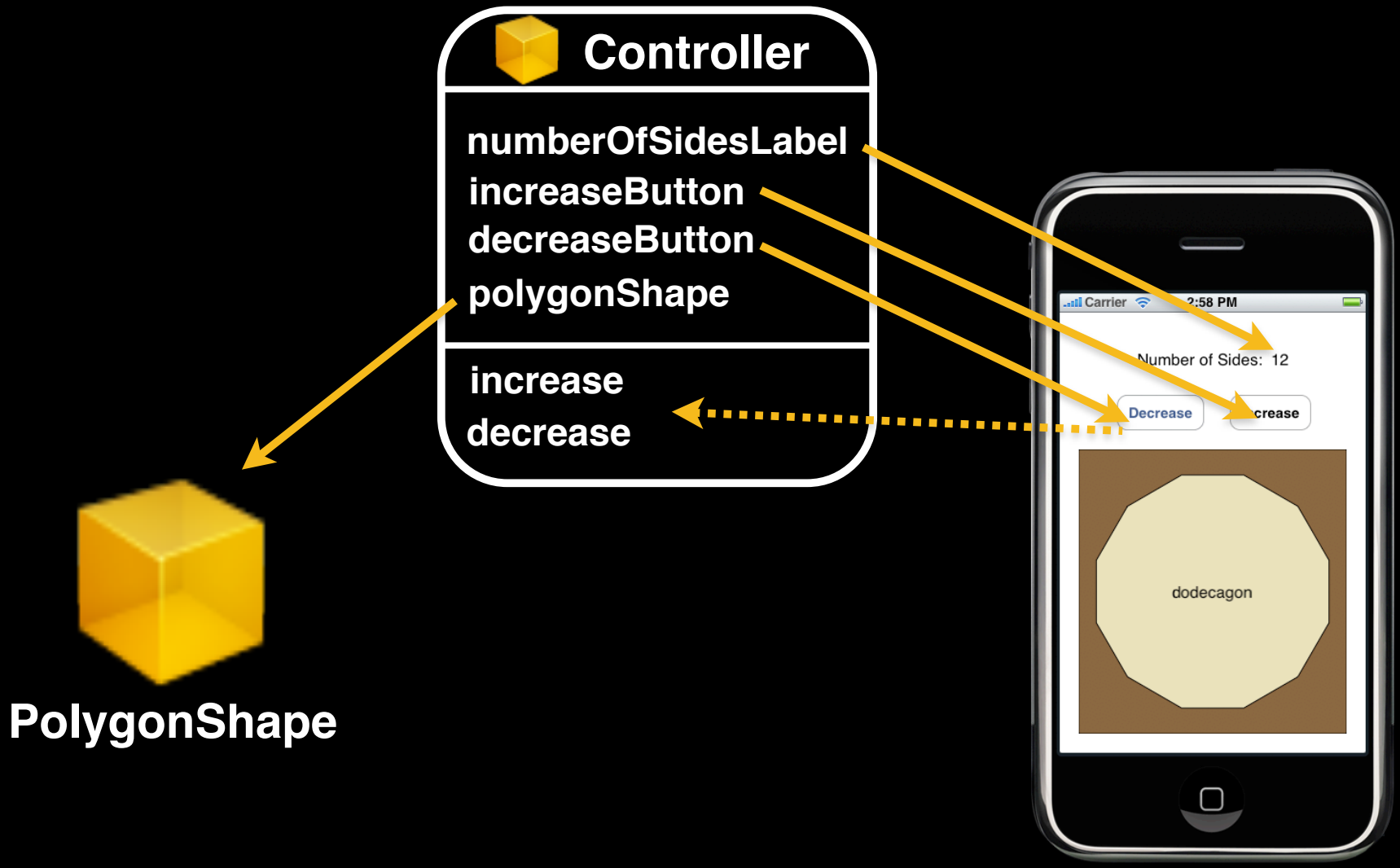
Model, View, Controller

HelloPoly



Model, View, Controller

HelloPoly



Nib Files - HelloPoly example

- HelloPoly has all objects (model, view and controller) contained in the same MainWindow.xib file
  - More common to have UI broken up into several nib files
- UIKit provides a variety of "View Controllers"
  - We will be introducing them with the Presence projects

Questions?