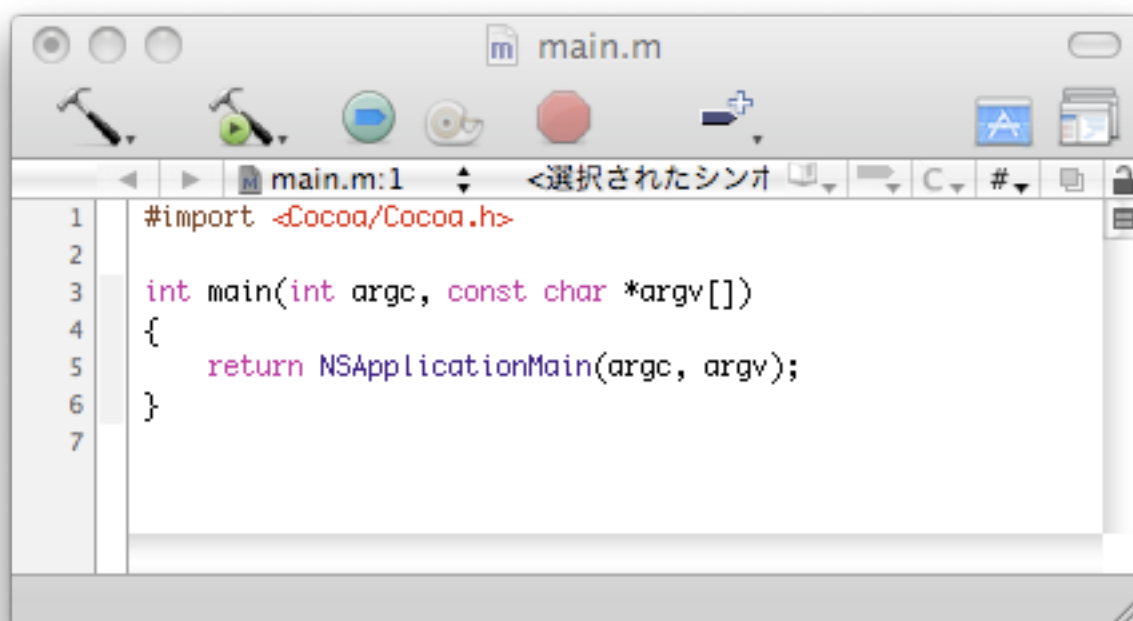


Cocoaアプリケーションの起動手順と処理形式

Cocoaアプリケーションが作動する仕組みを知ろう

Cocoaアプリケーションの起動手順

xcodeでCocoaアプリケーションを作った場合、どのプロジェクトにも「main.m」というソースが自動生成されている。実は、Cocoaアプリケーションを起動する際に一番最初に呼ばれるのはこの「main.m」である。だけど、「main.m」の内容は下図の通り至って簡単。NSApplicationMainと言うファンクションを呼んでいるだけである。が、この簡単なファンクション呼び出しはCocoaアプリケーションが動くために必要な様々な作業を行うトリガーである。

A screenshot of the Xcode IDE showing the main.m file. The window title is 'main.m'. The code editor displays the following code:

```
1  #import <Cocoa/Cocoa.h>
2
3  int main(int argc, const char *argv[])
4  {
5      return NSApplicationMain(argc, argv);
6  }
7
```

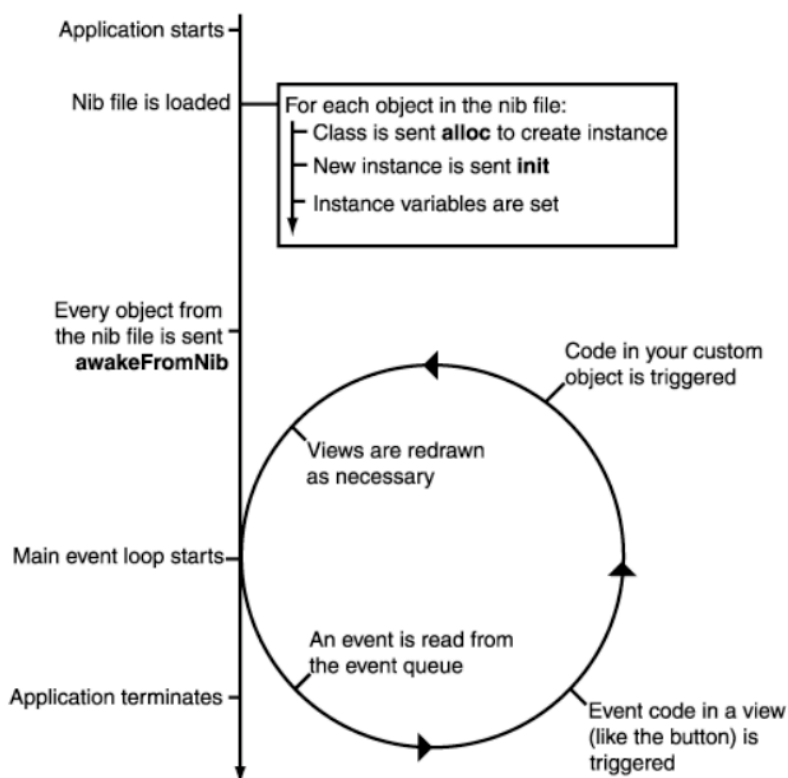
以下にNSApplicationMainファンクションが行う作業を順番に記述する。

1. Applicationオブジェクトの作成
2. Autorelease poolのセットアップ
3. main.nibファイルから初期ユーザインタフェースをロード
4. Applicationを稼働し、イベントを受け付けることができる状態にする

3.の段階でユーザインタフェースの初期化は「`awakeFromNib`」メソッドを呼び出すことで実現される。main.nibファイルに格納されているオブジェクトはアンアーカイブされた直後に「`awakwFromNib`」が呼び出される。

4.では「responder chain」が作成され、イベントを受け付ける状態になる。

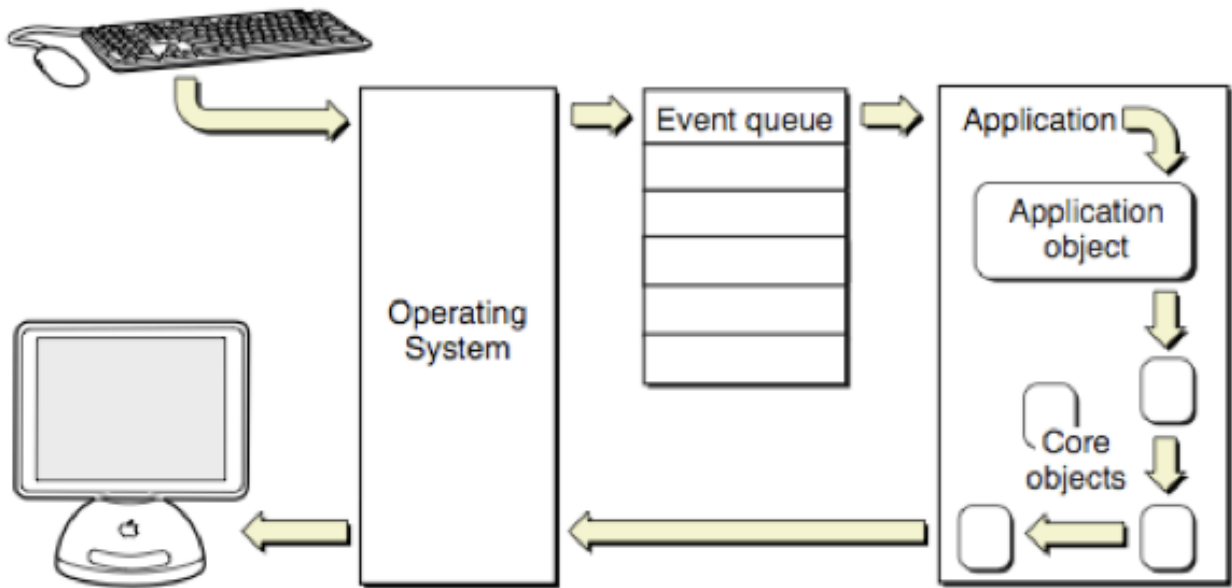
一旦、これでCocoaアプリケーションの初期化は完了し、アプリケーションは正常稼働の状態に入る。ここからアプリケーションはマウスの操作やキーボードからの入力等、ユーザからのイベントに応じて処理を行うこととなる。以下に、Cocoaアプリケーションの起動から稼働、停止までの手順を示す。



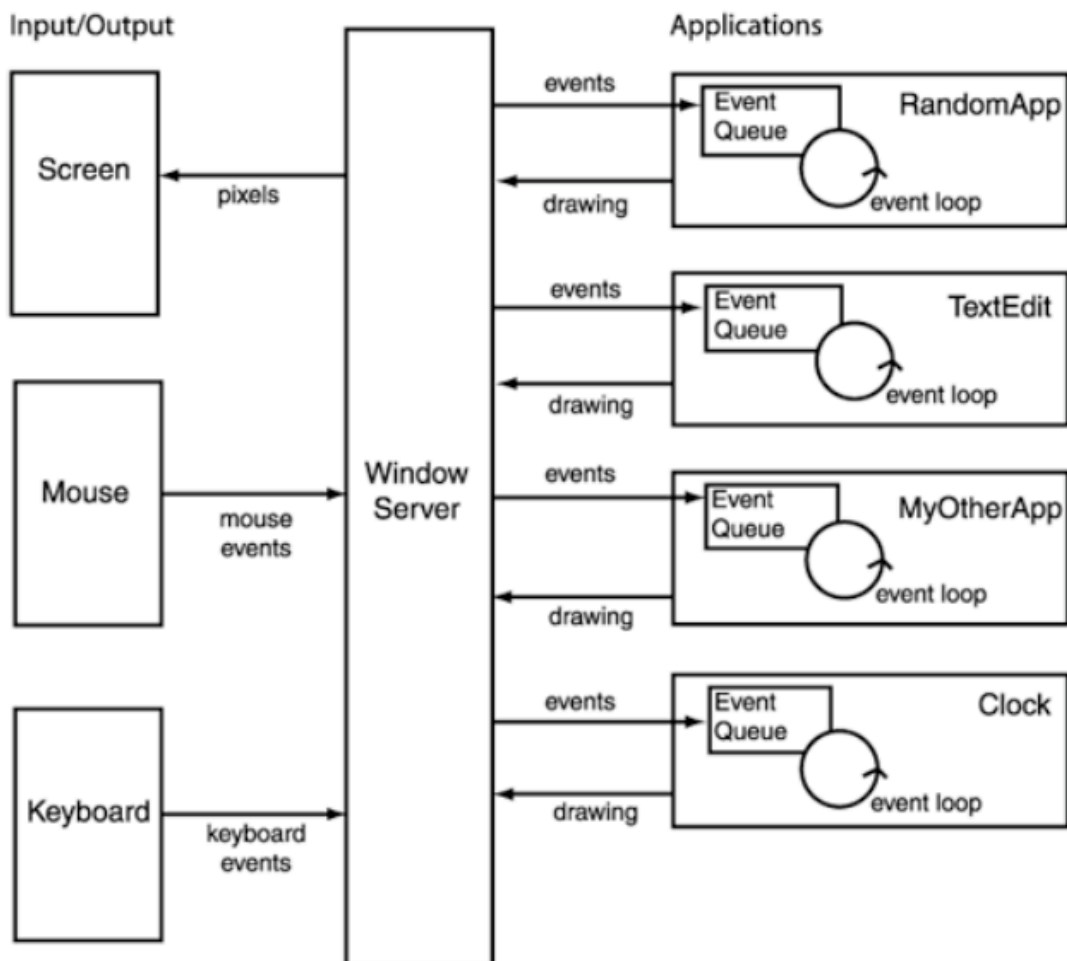
上図で「Main event loop starts」までが「`NSApplicationMain`」ファンクションの処理内容である。

Main event loopの動作イメージ

「Main event loop」の動作イメージを以下に示す。



また、上図でOperating SystemはWindow Serverであり、Window Serverと各アプリケーションとの関係は以下の通りである。CocoaのプログラミングでWindow Serverを直接意識することはあまりない。（と、今のところ思われる。）



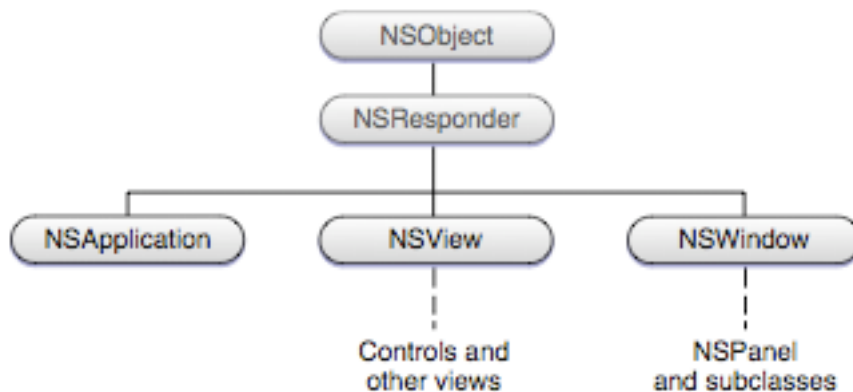
Responder Chainの登場

Event Queueに入って来たイベントは「Responder Chain」にそって処理される。「Responder Chain」はアプリケーションに対するイベントを処理するための「ApplicationKit」のアーキテクチャである。

「Responder Chain」は前述したNSApplicationMainファンクションによって作成される。「Responder Chain」はアプリケーションが持っているWindow毎に1つ作成される。よって、アプリケーションが複数のWindowを持っているならその数分の「Responder Chain」を持つこととなる。

Responder Chainの中身

「Responder Chain」には主に「NSWinodw」インスタンスとそのWindowが持つビューのツリーが格納される。「NSResponder」を継承し、「Responder Chain」に入れることができるクラスとしては「NSWindow」、「NSView」、「NSApplication」、「NSDrawer」、「NSWinodwController」、「NSViewController」などがある。勿論、実際にはもっと多くのクラスが事前に定義されており、また自作のクラスが「NSResponder」を継承することもできる。



First Responder

「Responder Chain」内で初めて処理のチャンスを与えられるクラスを「First Responder」と呼ぶ。nibファイルには「First Responder」を指すプロキシクラスが定義されている。「NSWinodw」は「First Responder」に対する参照を保持しており、イベントに対する最初の処理機会を「First Responder」に与える。常は「First Responder」はそのWndowの中の選択されているNSViewのインスタンスである。

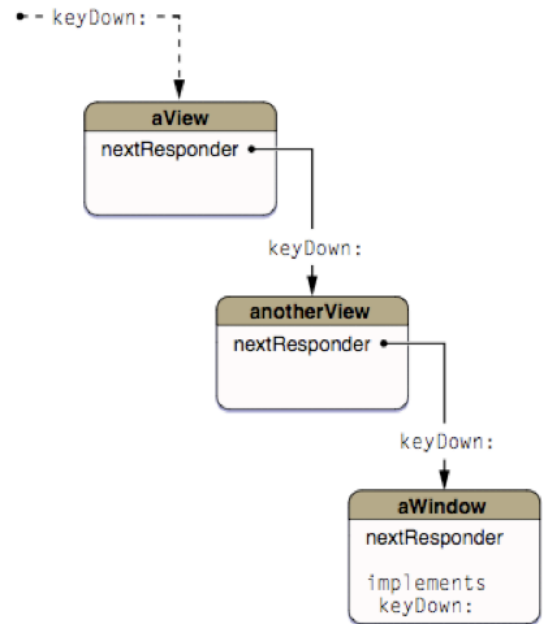
Responder Chainの処理方式

「Responder Chain」にはNSResponderクラスを継承するクラスが入れており、順番にイベントを処理するチャンスを与えられる。最初のNSResponderクラスのインスタンスによって処理されなかったイベントは次のインスタンスに渡され、再び処理が試される。

イベントメッセージとアクション メッセージの処理

「Responder Chain」で処理するメッセージはユーザからのイベントメッセージと特定のキー等にバイディングされたアクションメッセージがある。この二つのメッセージの処理方式は以下の通りである。

- ・ イベントメッセージの場合、「NSWindow」に到達したらそのメッセージの処理は終了する。「NSWindow」の次に「next responder」があってもそれは無視される。
- ・ アクションメッセージの場合、アプリケーションがキー・ウィンドウとメイン・ウィンドウを持っている場合は両方の「Responder Chain」が関与する。キー・ウィンドウの「Responder Chain」が先に実行される。「NSWindow」までメッセージが処理されなかったら「NSWindow」をデリゲートしてるクラス、「NSApplication」、更に「NSApplication」をデリゲートしているクラスがメッセージを処理する。その時まで処理されなかったメッセージは無視される。



キーイベントとマウスイベントの送り先

ウィンドウはキーイベントとマウスイベントを以下のように処理する。

- ・ キーイベントは「First Responder」に送付する。
- ・ マウスイベントはそのイベントが発生したビューに送付する。

参考：File's Owner

nibファイルを所有し、管理するオブジェクトのこと。nibファイル内のオブジェクトと外部のオブジェクト間の橋渡し役を担う。よって、File's Ownerとして指定されるオブジェクトは必ずnibファイルの外部に存在する必要がある。main.nibの「File's Owner」はNSAppである。アプリケーションが起動する時に、NSAppはmain.nibをロードし、その中のファイルを解凍し、メニューと初期ウィンドウを表示する。