Programowanie aplikacji na iPhone.
Wstęp do platformy iOS.

Łukasz Zieliński

Plan Prezentacji.

- Programowanie iOS. Jak zacząć?
- Co warto wiedzieć o programowaniu na platformę iOS?
- Kilka słów na temat Obiective-C.
- Wstęp do modelu Model-View-Controller.
- Jak stworzyć pierwszą aplikacje?
- Zademonstrowanie działania.
- Tworzenie gier na iPhone:Obiectve-C Cocos2d.
- Tworzenie gier na Windows Phone: C# XNA .

Programowanie iOS. O App Store.

- 5 marca 2012 roku Apple ogłosiło, iż z App Store pobrana została 25-cio bilionowa aplikacja.
- Obecnie App Store oferuje ok. 550’000 aplikacji dla 315 milionów użytkowników iPhonów, iPodów i iPadów w 123 krajach.

Niezbędny sprzęt.

- Komputer Mac, oparty o platformę Intel i działający pod kontrolą systemu operacyjnego OSX w wersji 10.6.x (Snow Leopard) bądź nowszej.
- Xcode - jest to kompleksowe środowisko programowania dla systemów OSX oraz iOS. Aplikację można bezpłatnie pobrać z Mac-App Store.

Programowanie iOS. Jak zacząć?

- **Rejestracja w programie developerskim Apple - iOS Developer Program.**
- **https://developer.apple.com/**

Co warto wiedzieć o programowaniu na platformę iOS?

- Tylko jedna aktywna aplikacja.
- Tylko jedno okno.
  960x640 iPhone 4.
  640 x 1136 iPhone 5.
- Ograniczone zasoby systemowe. (Sym. Xcode)
  iPhone 5:
  RAM: 1 GB LPDDR2 eDRAM.
  Procesor CPU: 2 rdzenie, dynamiczny zegar 800-1300 MHz, architektura ARMv7

Kilka słów na temat Obiective-C.

- Język Objective-C rozszerza możliwości C poprzez dostarczenie funkcji zorientowanych obiektowo.
- Zastosowany tutaj model programowania zorientowanego obiektowo polega na wysyłaniu wiadomości obiektom.
- Możliwość wykorzystania bibliotek C (lub C++) w aplikacjach Objective-C.

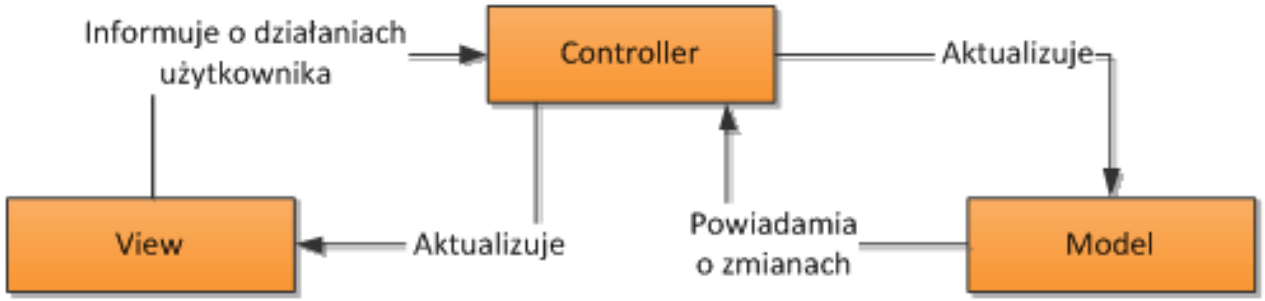
Obiective-C. Automatic Reference Counting i Garbage Collection.

- Używają metody: Zliczanie referencji (ang. reference counting) .
- ARC brak zabezpieczania: wzajemne (cykliczne) odwołania.
- ARC nie śledzi działania programu, w czasie kompilacji „dopisuje”, „uzupełnia” kod o komendy zwalniania pamięci.
- GC jest mniej wydajne niż ARC. Ale zapewnia niezawodne zwalnianie pamięci.

Wstęp do Model - View - Controller.

- - Model - wszelkie klasy odpowiedzialne za dane naszej aplikacji,
- - View - elementy interfejsu graficznego użytkownika, okna, widoki, przyciski, suwaki itd.,
- - Controller - kod wiążący interfejs aplikacji z jej danymi.

Wstęp do Model - View - Controller.



- **Głównym celem wzorca jest taka separacja kodu, aby wszystkie wspomniane elementy były od siebie jak najbardziej niezależne.**

Wstęp do Model - View - Controller.

- Znaczy to tyle, że nasz model nie wie nic o interfejsie aplikacji, podobnie GUI jest całkowicie niezależne od danych.
- Kontroler jest tym, co łączy interfejs graficzny z danymi.

Wstęp do Model - View - Controller.

- Komunikacja GUI z kontrolerem odbywa się za pomocą mechanizmu zwanego target - action.
- Element interfejsu w razie konieczności (np. gdy użytkownik wciśnie przycisk na ekranie) przekazuje odpowiedniemu kontrolerowi (target) akcję do wykonania (action).
- -(IBAction)Klik:(id)przycisk;

Jak stworzyć pierwszą aplikacje?

- Uruchommy Xcode i utwórzmy nowy projekt wybierając szablon.
- AppDelegate – klasa odpowiedzialna za działanie naszej aplikacji np. utworzenie okna.
- Nazwa-info.plist – plik z postawowymi danymi o naszej aplikacji.

Atrybuty.

- atomic - używając tego atrybutu, generowany jest setter i getter synchroniczny, chodzi o operacje wielowątkowe i zabezpieczeniem się przed dostępem do obiektu z dwóch wątków jednocześnie.
- nonatomic – przeciwieństwo atomic, setter i getter jest generowany bez synchronizacji.
- retain – oznacza, że jesteśmy odpowiedzialni za zwolnienie obiektu z pamięci.

Atrybuty.

- assign - jest to tzw. „słaba referencja”, nie zwiększa ona licznika referencji do obiektu, najczęściej używana przy definicji delegatów, oraz wskaźników na prymitywne typy języka C.
- copy – wykonuje kopie obiektu, za którą jesteśmy odpowiedzialni, musimy ja wyrzucić z pamięci. Tego atrybutu powinno się używać, gdy obiekt może się zmieniać (mutable object):

Atrybuty.

- strong – jest to coś na wzór retain, ustawiamy „silna referencja” do obiektu, przez co licznik referencji jest zwiększany o jeden, oczywiście kompilator sam martwi się o usuwanie obiektu.
- weak – coś podobnego do atrybutu assign „słaba referencja” z tą różnicą, że gdy obiekt na którego wskazujemy został usunięty a my chcemy się do niego odwołać, zwrócona wartość będzie nil ponieważ obiekt nie istnieje.

Zademonstrowanie działania. Przykładowa aplikacja.

- Zakupy:
- Model: Spis.plist.
- Controller: Dodaj(), ObliczCene().
- View: UIButton * Dodaj, UIButton * Oblicz, UITextView * Text;
- IBOutlet, IBAction, @synthesize.

Tworzenie prostych gier na iPhone:Cocos2d.

- Demonstracja.

Tworzenie gier na Windows Phone: C# XNA .

- Demonstracja.