TIMERS

Target = the view controller that you are in (you can use word “self”)

Selector = what do we want to happen every second, the function the timer uses every time it activates

* selector: #selector(ViewController.decreaseTimer)

timer.invalidate() = to stop a timer

TABLE VIEWS

Ctrl drag table view to yellow circle on top (view controller)

* data source = view controller can supply table view with data
* delegate = view controller controls the table view

you need to add types to the ViewController class in the ViewController.swift file

* particularly the UITableViewDataSource and UITableViewDelegate

you need to add two methods to be able to use the UITableViewDataSource

* the numberOfRowsInSection returns how many rows are in the table
* the cellForRowAt defines the content of row using the indexPath(index number of row) and identifier (you set it up)

table.reloadData()

STORING PERMANENT DATA

Use UserDefaults

When retrieving info that was stored, you are retrieving objects

CONTROLLING THE KEYBOARD

To make the keyboard go away when the user presses the app

* use the touchesBegan function
* self.view.endEditing(true)

To make the keyboard go away when the user presses the return button

* add a new data type to the ViewController class: UITextFieldDelegate
* use ctrl + drag of your textField in storyboard to the viewController yellow button, click delegate
* use the textFieldShouldReturn function
* textField.resignFirstResponder()
* return true

TO DO LIST APP

We used a different template: tabbed application

* creates by default the navigation of two view controllers

MULTIPLE VIEW CONTROLLERS

You need to make a new cocoa touch class

* is going to be a subclass of UIViewController

you need to link your second view controller to the new viewController.swift file

* go to identity button on top, select class and choose the file to link it to

to switch from viewControllers

* put a button on the first one, use ctrl + drag to the second view controller and click show

DOWNLOADING WEB CONTENT

Web view

* Embedd web browser

Info.plist file

* Shows info about your app
* To load content from http or insecure URLs: click add row, then write NSAppTransportSecurity….click the + button; write the exception domain, set it up as a dictionary….add an ítem of type boolean that return Yes, write NSTemporaryExceptionAllowsInsecureHTTPLoads….add another ítem and write NSIncludesSubdomains, set it as boolean that returns true

WEATHER APP

When working inside the function or thread task(), known as a closure, and you want to reference something outside the task() function, you need to use the self keyword at the beginning

* When the varo r outlet or action that you are trying to use is not declared inside the task function

TIC TAC TOE

You can associate multiple buttons with a single action

Any user interface element can have a tag, which is a number to identify it

You can have arrays inside arrays

INTEGRATING MAPS

You need to use map views

Need to import library

* Import MapKit

Need to include another type to the main class

* MKMapViewDelegate

FINDING A USER LOCATION

You need a framework from the button “build phases”

* Link binary with libraries; press + button
* Add framework: CoreLocation.framework

Need to edit the info.plist file

* Add: NSLocationWhileInUseUsageDescription; add a description of why you want to know the user’s location
* Add: NSLocationAlwaysUsageDescription (this one pop ups if you want to know their location all the time)

You need to add another type to the main class

* CLLocationManagerDelegate

You need to import the framework

* Import CoreLocation

ADVANCED SEGUES

To make segueways just ctrl + drag an item on one view controller to another one

How to pass variables across different view controllers

1. Use a global variable (can use it anywhere in the app)
2. Pass a value from one view controller to another

You need to create an identifier for the segueway

MEMORABLE PLACES

Navigation controller

* Has a table view in the root view controller
* For apps that need a table in root view controller

When adding a bar button item to root view controller with table view

* Drag the button to the files, directly under root view controller

The navigation controller is not a view controller, you don’y see it…it just controls the navigation

View did load is not actually run everytime the screen appears

* For this you use view did appear

AUDIO

Need to import library

* AVFoundation

CORE DATA

Built-in database storage coming with ios apps (it is permanent storage)

* Works for more complex info than using userDefaults

UserDefaults is not good for storing large amounts of data, or making it searchable, or store it in a DB like format

Most commonly used when users log in to an app

You need to click on button “use core data” when creating the project

You will have a file specifically for core data

* Called a data model: a DB structure

The appDelegate.swift file contains a collection of methods that control the basic functionality of our apps

* Controls big events when app is loaded and removed from memory
* New methods are introduced to the file since we are working with coreData

You need to import framework: import CoreData

ADVANCED CORE DATA

In CoreData we saw how to set it up, how to store new items and how to retrieve them, in advanced CoreData we saw how to return specific values, how to update and delete values already stored

For request to return a specific value we need a predicate

* An instruction to only look for certain data with certain properties

API’S AND JSON DATA

API

* Is a programmatic way to access functions and data from other apps

JSON

* To store data in a plain text format

When creating a url in swift, you are going to include you API key, which lets you communicate with the API of the webpage

BLOG READER APP

We used the master detail template, enabling core data

If you click on product, then clean, all cached data on app gets erased

* Also you need to erase app from simulator

**INSTAGRAM CLONE**

Parse Server & AWS

is all about making an ec2 instance and launching parser on it, then connecting your app to the server by getting the appID, clientKey, and serverURL and writing them down on a swift file.

You also need to allow for the http domain of your server in the info.plst file

PFPbjects work like dictionaries

Retreiving & Updating Data

all about updating, saving, and retrieving info. from parse

* You can update info right in the parse webpage

ACL = access control lists

* Controls who can read and write to an object
* You can set up public access, so that everyone can read and write, not only specific users (manually in parse)

Accessing The Camera Roll

You need to add these types to the main class:

* UINavigationControllerDelegate, UIImagePickerControllerDelegate

You need to add a row in the info.plist file

* Privacy- Photo library usage (to give a reason of why you want to access the user’s photos)

SPINNERS AND ALERTS

Alert

* A simple pop up msg

Spinner

* Is used to indicate that the app is busy

LOGIN AND SIGNUP

Secure text entry

* So users don’t see the password they are typing

By using parse we don’t need to check if email addresses are valid, parse does it for us

THE USER TABLE

We use a nav. Controller

* So it brings a view with a table

We created a segueway from the initial viewcontroller to the navigation controller

We created a class in parse called Followers

POSTING IMAGES

We created another standard view controller

* With a segueway from the table view to the new view controller

We added a new row to the info.plist file

* So we have permission to enter photo library of the user

VIEWING USER’S FEED

In the app you can see a list of the users, follow and unfollow, post a picture directly to parse server

* Now we want the user to the the feeds of the users that they are following

We added a table view controller

* So the user can scroll down and view all the photos

We added a segueway from our feed button to our new table view controller

We stretched the row to add an image view and a couple of labels

We created a new file called feedTabeViewCell, subclass of UITableViewCell

We download all of the users, then we are downloading all of the user’s that are being followed by the logged in user, then we are downoaling all of the posts of these users

* Is not efficient if your app has thousands of users

**TINDER CLONE**

DRAGGING OBJECTS

We learn how to show elements in the view programmatically, which means that there is no need to add elements to the storyboard

LOG IN PAGE

We commented out the PFUser.enableAutomaticUser(); because we only want a user after they have signed up properly

* In the appDelegate.swift file

Cannot modify user ….. error in parse

* Solution: go to parse directly and delete all users (or the one being mentioned) and to be extra sure, delete the app from the iphone simulator

We didn’t put the condition that if the username was empty, then the user would receive an error

* So, if a user sign up with a blank username, parse creates one for them automatically

ADDING USER DETAILS

We added a new view controller and we use switches (another UIElement)

We added a new row in the info.plist file to get access to the photo library

We added segues for when the user either signs up or logs in in ViewController.swift file

We can download images directly to the simulator by surfing the web and making a long press on the image

To allow us to update the current user we need to set publicwriteaccess to true

Switch

* Their main control is either on or off (isOn= is a bool), they don’t have a value

ACL

* Our way of determining who can read or write to a particular object
* The default for parse users is that anyone can read them but only the users themselves can update their own data

ADDING USERS

We allowed arbitrary loads in info.plist file, under app transport security settings

SWIPING USERS

We added a new view controller

We made a segueway from the userDetails to the swiping view controller

In the view controller file we added code to make the user go directly to the swiping view controller if it has already submitted its info. In the userDetails view controller

We added new columns to parse, called accepted and rejected respectively; which is an array of userIDs

LOCATION AND MATCHES

Now we want for people to have accepted each other, to connect to each other

We have added:

1. Location awareness
2. Our user can see matches
3. Users can write messages to each other

Parse ha location built into it

We added a new row to info.plist file

* Privacy-location when in use

We added a new view controller, subclass of UIView controller

We added a hidden label in the matchestableviewCell view that contains the userID of the user being displayed in the picture

In the app we can log in/log out, swipe users that are near us, and look at our matches and chat with them

**FLAPPY BIRD CLONE**

Hello word

Gamescene.swift = a scene is a page of the app

Gamescene.sks = what mainstoeyboard was to our apss before

A node is anything that appears on the screen on the game

ADDING AND ANIMATING SPRITES

Func didMove

* Essentially like the viewDidLoad
* It means that a scene has appeared on screen

DETECTING COLLISIONS

Enum (data type)

* Used when we want to categorize certain groups of objects

We added a data type to the class

* SKPhysicsContactDelegate

**UBER CLONE**

SETTING UP PARSE

The public DNS of your ec2 instance goes in ParseMutableClientConfiguration.server space (in xcode, appDelegate); you need to add [http://](NULL) at beginning and add “/parse” at the end

To open up your parse server, copy your main URL, and add “/apps” in your browser

LOG IN PAGE

We commented out in AppDelegate.swift the line “”PFUser.enableautomaticuser()

We added a line of code to AppDelegate.swift

* defaultACL.getPublicWriteAccess = true

when getting following error message = “ this class is not key value coding-compliant for the key …..”

* check the names of your outlets, can’t have outlets with the same name

THE RIDER VIEW

We added a new view controller

We import mapKir and add mapkitdelegate to main class

Added row to info.plist file and added locationmanagerdelegate to main class

* privacy-locaiton when in use

THE DRIVER VIEW

We made a segue from the cell itself to the view controller

* so that when you click on a request, you can see the rider’s location in another view controller

SHARING DRIVER’S LOCATION TO RIDER

Problem with parse with locations

* you can’t have two or more PFGeopoints on a single object

in a PFObject you can only ever have 1 GeoPoint

**SNAPCHAT CLONE**

SIGNUP-USERLIST

Wickr

* handles security and encryption in a messaging system

RECEIVING AN IMAGE

**SUBMITTING YOUR APP TO THE APP STORE**

APP ICONS AND LAUNCH IMAGES

You can use the website <https://makeappicon.com> to create photos of different sizes

* you insert them in the assets folder in xcode

to show a photo while the app is first launching, go to the app general settings, then click on launch image source, then migrate (specifying an asset catalog)

check <https://developer.apple.com/ios/human-interface-guidelines/graphics/launch-screen/> for launch screen photos

check <https://developer.apple.com/ios/human-interface-guidelines/graphics/app-icon/> for the app icon

RUNNING YOUR APP ON YOUR OWN DEVICE

Go to xcode, preferences, accounts, add apple id, log in, click on view details, create a signing identity for IOS development (it creates a provisioning file)…then go to the app general settings, make sure that on the field “Team” is the one that you created earlier…plug in your device and select it from the app general settings (where all the possible phones appear)

You could get an error saying that your device is not trusted for a development account…go to your phone settings, click general, then profile an device management…click on the developer account, then click trust

SUBMITTING TO THE APP STORE

You need a paid developer account (100 US annually)

Check video again

label

text field

button

image view

navigation bar

toolbar

table view

sliders

multiple view controllers

map views

Navigation controller

Web views