CS193p

Spring 2010

Today's Topics

- More on Touch Events
- View Animation
- Accelerometer
- GameKit

Animating Views

- Some properties on a `UIView` can be animated

  You set them normally, but the change does not show on screen immediately.
  It shows up over a period of time (in a manner you can control).

- Animatable properties

  You can animate changes to `frame` (and `center`), `alpha` and `transform`.
  This last one is pretty powerful (rotation, scaling).

- Steps

```
[UIView beginAnimations:(NSString *) context:(void *)];
[UIView setAnimationDuration/Curve/Delay/RepeatCount:...];
// change the view properties
[UIView commitAnimations];
```

  Now the animation will start and run until done or new animation starts.
  The changes actually happen immediately (i.e. if you ask the view).
  Multiple animations just fine. They'll happen simultaneously.

Animating Views

- Multistep animations

  Sometimes you need one step of an animation to finish before another.

  For example, if you want a view to scale up, then back down (e.g. pulse).

- Animation Delegate

  Get notified when first step finishes, then start another animation.

```
[UIView setAnimationDelegate:self];
[UIView setAnimationDidStopSelector:@selector(methodName:finished:context:)];

- (void)methodName:(NSString *)animationID
          finished:(NSNumber *)finished
           context:(void *)context
{
    if ([animationID isEqualToString:@"MyID"] && [finished boolValue]) {
        [UIView beginAnimations:@"NextStepInMultiStep" context:(void *)];
        // do another animation, might use the context in here
        [UIView commitAnimations];
    }
}
```

Animating Views

- Demo

Accelerometer

- Accelerometer can be an input device like touches
- Get shared accelerometer object

```
UIAccelerometer *accelerometer = [UIAccelerometer sharedAccelerometer];
```

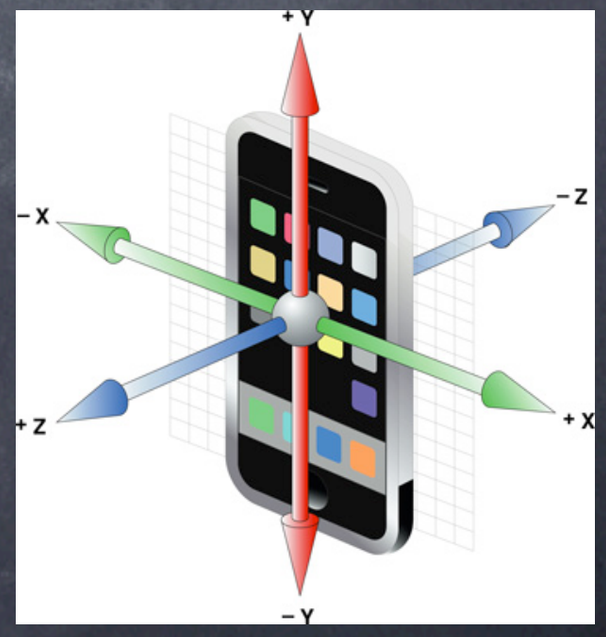
- Set its delegate and update rate

```
accelerometer.delegate = self;
accelerometer.updateInterval = 0.2; // 5 times per second
```

- Then react to changes

```
- (void)accelerometer:(UIAccelerometer *)accelerometer
        didAccelerate:(UIAcceleration *)acceleration;
```

`UIAcceleration` class has three properties: `x`, `y`, and `z`.
Each is a `UIAccelerationValue` (a double)
measuring acceleration (in g) in three directions.
Each `UIAcceleration` also has a `timestamp`.



Accelerometer

- Demo

Game Kit

- Manages a connection between multiple devices
- Provides user-interface for setting up connection
- Can work over Bluetooth or via WiFi
- WiFi requires you to manage the connection
  Bluetooth gets set up for you.
- After connection, send `NSData`'s back and forth
- This is networking, not hard-wired connections
  Expect the unexpected.
  Be ready to have a client disconnect, then try to reconnect.

Game Kit

- Lifecycle (using `GKPeerPickerController`)

  Put up picker.

  Create a `GKSession` via `GKPeerPickerControllerDelegate` callback.

  Set `GKSession` delegate (to watch for connections/disconnections).

  Set `GKSession` data handler (to handle incoming data).

  Now that connection is established, dismiss the picker (and release it).

  Clean up when a connection goes away.

  Welcome a new peer (usually by sending it some data) when it joins.

  Send data to peers whenever necessary.

  Handle incoming data from peers appropriately.

- Putting up the picker

```
GKPeerPickerController *picker = [[GKPeerPickerController alloc] init];
picker.delegate = self;
[picker show];
```

GKPeerPickerControllerDelegate

- Create a session (at the behest of the picker)

```
- (GKSession *)peerPickerController:(GKPeerPickerController *)picker
            sessionForConnectionType:(GKPeerPickerConnectionType)type
{
    if (!mySession) {
       mySession = [[GKSession alloc] initWithSessionID:(NSString *)
                                                displayName:(NSString *)
                                                sessionMode:(GKSessionMode)];
    }
    return mySession;
}
```

- `GKPeerPickerConnectionType` is `Nearby` or `Online`
  Nearby means Bluetooth, Online means WiFi.
  If Online, you need to set up the connection yourself.
- `SessionID:` can be `nil` (will be app bundle id)
- `displayName:` can be `nil` (will be device name)
- `GKSessionMode` is `Peer`, `Client` or `Server`.
- `mySession` is an instance variable (need for sending)

GKPeerPickerControllerDelegate

- Get notified when picker connects a peer to you

```
- (void)peerPickerController:(GKPeerPickerController *)picker
                 didConnectPeer:(NSString *)peerID
                      toSession:(GKSession *)session
{
   [session setDataReceivedHandler:self withContext:nil];
   session.delegate = self;  // GKSessionDelegate protocol
   [picker dismiss];
   [picker release];
}
```

The delegate and data receive handler could be set in previous slide.

- Clean up if user cancels the picker

```
- (void)peerPickerControllerDidCancel:(GKPeerPickerController *)picker
{
   [picker release];
}
```

Note that it dismisses itself on cancel (but not on accepting a connection).

GKSession Data Handler

- Receiving data

```
- (void)receiveData:(NSData *)data
           fromPeer:(NSString *)peerID
          inSession:(GKSession *)session
            context:(void *)context
{
  // convert NSData into your local data structures and handle
}
```

- Sending data

```
[mySession sendDataToAllPeers:(NSData *)
                 withDataMode:(GKSendDataMode)
                        error:(NSError **)];

[mySession sendData:(NSData *)
            toPeers:(NSArray *)
       withDataMode:(GKSendDataMode)
              error:(NSError **)];
```

GKSessionDelegate

- Respond to state changes

```
- (void)session:(GKSession *)session
           peer:(NSString *)peerID
 didChangeState:(GKPeerConnectionState)state
{
    // state could be Available, Unavailable, Connected, Disconnected, Connecting
    // usually only interested in Connected, Disconnected if using picker to connect
}
```

Non-Picker Connect

- How to connect without the Picker?

  Create GKSession, set `available` property to YES, and watch state changes.

```
- (void)session:(GKSession *)session
           peer:(NSString *)peerID
 didChangeState:(GKPeerConnectionState)state
{
   if (state == GKPeerConnectionStateAvailable) {
      [session connectToPeer:peerId withTimeout:(NSTimeInterval)timeout];
   }
}
```

- On the other side, accept or deny the request

```
- (void)session:(GKSession *) didReceiveConnectionRequestFromPeer:(NSString *)peerID
{
   if (iLikeThisPeer) {
      NSError *error = nil;
      [session acceptConnectionFromPeer:peerID error:&error];
   } else {
      [session denyConnectionFromPeer:peerID];
   }
}
```

GKSession Errors

- Handle failure

```
- (void)session:(GKSession *)session
     connectionWithPeerFailed:(NSString *)peerID
                    withError:(NSError *)error
{
    // failure during connection process
    // not necessary to handle this if you are using picker to establish connections
}
```

```
- (void)session:(GKSession *)session didFailWithError:(NSError *)error
{
    // catastrophic, entire session failure
    [session disconnectFromAllPeers];
    // clean up instance variables that point to this session
}
```

Game Kit

- Demo