Overview

Description

- Part 1: Introduction to Threading
- Part 2: Introduction to Animation (Using NSTimer)
- Part 3: Introduction to Animation (Using Animation Blocks / CoreAnimation)
- Part 4: Final Project Proposal (Online)

Submission

Lab evaluation form
Lab feedback form (Online)
Final Project Proposal (Online)

Note: If you are unable to complete the lab during this week's lab period, please be ready to demonstrate it at the beginning of the next lab.

Part 1. Introduction to Threading

In this part of the lab, you will download a large file from the internet on a separate thread. There is a convenience method that will create a NSThread object and run a selector in the background that you will find helpful:

```
- (void)viewDidLoad {
    [super viewDidLoad];
    NSString *myObject = @"Hello World";
    [self performSelectorInBackground:@selector(myFunc:) withObject:myObject];
}

- (void)myFunc:(NSString *)param {
    // The pool is necessary if you or the compiler autorelease any objects on this thread
    @autoreleasepool {
        NSLog(@"%@", param);
    }
}
```

Step 1) Create a new project called **iThread** using the single view application template.
(Alternatively, you may reuse the TableView lab that downloaded images from Flickr).
Step 2) Modify the sample code so that it downloads a large file in the background.
Example: http://media.vad1.com/temporary_url_20070929kldcg/tchaikovsky-1812-overture-op49-finale-oslo_philharmonic_orchestra.mp3
Step 3) In addition, your threaded function can call a function on the main thread (perhaps when it completes) by using the following function of the NSObject class:

performSelectorOnMainThread:withObject:waitUntilDone:

Step 4) Add a button to your application that logs a message. Click this button while the download is occurring to make sure your GUI remains responsive (ensure that the main thread is not tied up downloading the file).

Step 5) Demo your app to the TA

Part 2. Introduction to Animation (Using NSTimer)

To have the most control over your animation, you can setup a timer to call a function every “frame” of your animation.

Example:

```
NSTimer *timer = [NSTimer scheduledTimerWithTimeInterval:0.04 target:self selector:@selector(nextFrame) userInfo:nil repeats:YES];
```

Step 1) Create a new project called **Bounce** using the view based template.
Step 2) Find an image of a ball on Google, and add it as an UIImageView to your app.

In the View Controller
Step 3) Make the ball (UIImageView) to be a class variable; if you used Interface Builder, then add an IBOutlet.
Step 4) Add two class variables for the ball’s speed: xvel, yvel.
Step 5) Use the sample code to start a timer in the viewDidLoad method.
Step 6) Move the ball if your nextFrame function. One method is to use CGPointMake to create a new CGPoint and replace the center property of UIImageView every frame.

Step 7) Add logic to make the ball bounce around the screen. If the ball runs into the edge of the screen, you should negate the xvel or yvel value.

Step 8) Demo your app to the TA

Part 3. Introduction to Animation (Blocks)

In this part of the lab, you will animate the rotation/scaling/position of a UIView (of a subclass of UIView).

Step 1) Create a new project called **Transform** using the view based template.

Step 2) Find an image on Google that will fit on your view, and add it as an UIImageView to your app.

In the View Controller:

Step 3) Make the image (UIImageView) to be a class variable; if you used Interface Builder, then add an IBOutlet.

Step 4) Add a UIButton and an IBAction to start the animation.

In your IBAction, to trigger an animation**:**

Step 5) Move, rotate, and scale your image using an animation block. An example implementation is provided:

```
[UIView beginAnimations:@"example" context:NULL];
[UIView setAnimationDuration:0.5];

// Create transform matrix

CGAffineTransform transform;

transform = CGAffineTransformRotate(myimage.transform, 30);   //rotate 30 degrees
transform = CGAffineTransformScale(transform, 0.8, 0.8);   //shrink to 80%
transform = CGAffineTransformTranslate(transform, 0, 10);   //move right 10 pixels

// Change animateable properties

myimage.transform = transform;

// Done

[UIView commitAnimations];
```

The idea is to create a block around any change to an animateable property of UIView (frame, bounds, center, alpha, transform). These changes are then asynchronously and automatically animated.

Transform Matrices are an integral part of graphic rendering, games, Open GL ES, etc.

Step 8) Demo your app to the TA

Part 4: Final Project Proposal (Online)

Submit your final project proposal using the online Google Form. (Link)