Beginner

OpenGL ES & GLKit

Hands-On Challenges

Beginner OpenGL ES & GLKit Hands-On Challenges

Challenge #1: Flash Start

Let's get started with a flash – quite literally!

In this short challenge, you will make a new project that uses OpenGL to make the screen flash in an animated manner.

Part 1: Red Alert

Create a new project with the **iOS\Application\Single View Application** template. Name it **RedAlert**, set the class prefix to **RWT**, and save the project.

Open **Main.storyboard** and delete the view controller inside. Drag a new **GLKit View Controller** from the object library in its place.

Next you need to subclass `GLKViewController` and set this new view controller to use your subclass. To do this, open **RWTViewController.h** (the template made this for you) and modify the file to look like the following:

```
#import <UIKit/UIKit.h>

@import GLKit;

@interface RWTViewController : GLKViewController

@end
```

Next open **Main.storyboard**, select the view controller, and in the Identity Inspector (3rd tab) set the **Class** to **RWTViewController**.

Now your project is set up to use `RWTViewController` to use OpenGL to render its view. You just need to do three things:

1. Create and set the OpenGL context

2. Override glkView:drawInRect: to render

3. (Optionally) override update to update the view

Let's start with the first step. Open **RWTViewController.m** and replace `viewDidLoad` with the following:

```
- (void)viewDidLoad {
  [super viewDidLoad];
  GLKView *view = (GLKView *)self.view;
  view.context = [[EAGLContext alloc]
```

```
    initWithAPI:kEAGLRenderingAPIOpenGLES2];
  [EAGLContext setCurrentContext:view.context];
}
```

This creates and sets an OpenGL context.

Next, you're going to make your screen flash between red and black. To do this, add the following private instance variable to the top of the file:

```
@implementation RWTViewController {
  float _curRed;
}
```

This will keep track of the current "red" value (between 0 and 1); you will update this each frame.

Add this new method to render the scene:

```
- (void)glkView:(GLKView *)view drawInRect:(CGRect)rect {
  glClearColor(_curRed, 0, 0, 1.0);
  glClear(GL_COLOR_BUFFER_BIT);
}
```

This clears the screen to be the color according to `_curRed`.

Finally, add this new method to update the _curRed value each frame:

```
- (void)update {
  float secsPerFlash = 2;
  _curRed = (sinf(self.timeSinceFirstResume * 2*M_PI / secsPerFlash) *
    0.5) + 0.5;
}
```

This makes `_curRed` alternate between 0 and 1 over a period of 2 seconds.

Note: Not sure how this works? Well, sinf() is the sin function, which by default alternates between y = -1 and 1 for x = 0 -> M_PI. Here you are substituting time for x, so by default it will go between -1 and 1 every ~3.14 seconds.

You want it to flash more frequently than this. To do this, you want to modify the period of the sin function. You can do this by multiplying your x value by 2*M_PI/[desired period].

Similarly, you don't want the values to go from -1 to 1, since negative colors don't make any sense. You want it to go from 0 to 1 instead. To fix this, you

multiply the result by 0.5 (so the range is now -0.5 to 0.5) and then add 0.5 (for a final 0 to 1).

As you can see, sin functions are very handy for periodic value changes like this. To learn more, check out this video:

* https://www.khanacademy.org/math/trigonometry/basic-trigonometry/trig_graphs_tutorial/v/amplitude-and-period-cosine-transformations

Build and run, and watch out – your screen is flashing, alerting you that your first uber haxx0r challenge is on the way! :]

Uber Haxx0r Challenge: Flashing Flag

As you can see a `GLKViewController` is a view controller, just like any other view controller. This means two things:

1. You can embed multiple GLKViewControllers inside a single view controller, if you have different things you want to render.
2. You can mix GLKViewControllers with other kinds of UIKit controls and view controllers, such as sliders.

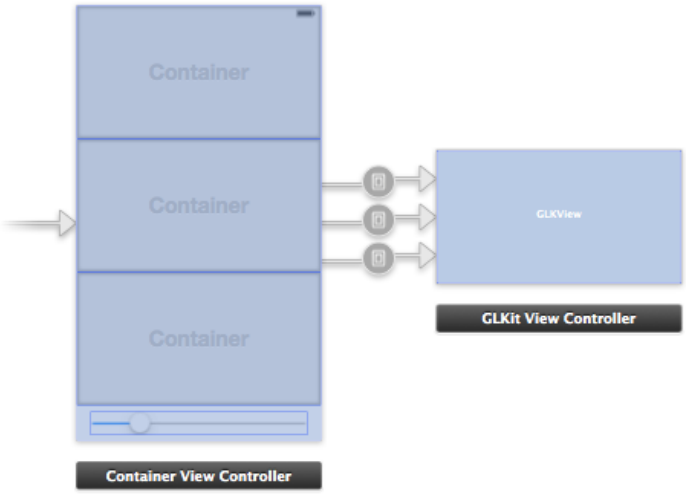
To experiment with this, you should modify your `RWTViewController` so it contains the following properties:

```
@property (assign) float rMult;
@property (assign) float gMult;
@property (assign) float bMult;
@property (assign) float secsPerFlash;
```

And modify your draw and update methods as follows:

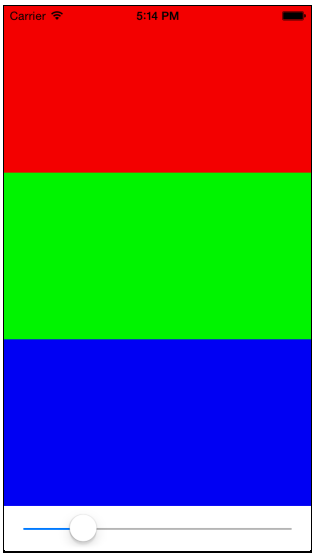
```
- (void)glkView:(GLKView *)view drawInRect:(CGRect)rect {
  glClearColor(_curVal * self.rMult, _curVal * self.gMult, _curVal *
    self.bMult, 1.0);
  glClear(GL_COLOR_BUFFER_BIT);
}

- (void)update {
  _curVal = (sinf(self.timeSinceFirstResume * 2*M_PI / _secsPerFlash) *
    0.5) + 0.5;
}
```

Now, open your storyboard and use container view controllers to create a layout that looks something like this:



You should create a class for this container view controller, and configure it so that the top-most view controller flashes red, the middle green, and the bottom blue. The slider (range 0.25-10) should allow you to configure the secsPerFlash for each child view controller (so the user can toggle how quickly/slowly each view controller flashes).

If you get it working, you should see something like this (except it flashes):



As you can see, you can render to just one portion of the screen, or even multiple places, and use all the UIKit controls you know and love with OpenGL!