```
//
//  TriangleViewController.m
//  HelloOpenGLES
//
//  Created by Elhassan Ahmed on 4/25/15.
//  Copyright (c) 2015 Elhassan Ahmed. All rights reserved.
//

#import "TriangleViewController.h"
#import "Vertex.h"

@interface TriangleViewController ()
@end

@implementation TriangleViewController
{
    GLuint _vertexBuffer;
    RWTBaseEffect *_shader;
}

-(void) setupVertexBuffer{

    const static Vertex vertices[] = {
        {{-1.0,-1.0,0}},//A
        {{1.0,-1.0,0}}, //B
        {{0,0,0}}, //C
    };
    //Moving data to video card to set it up from cpu memory
    //Generate a empty buffer in gpu
    //parm 1 how many buffers we want to generate
    //parm 2 location with referance
    glGenBuffers(1, &_vertexBuffer);
    //set the buffer to be active
    //GL_array_buffer will point toward the empty buffer
    glBindBuffer(GL_ARRAY_BUFFER, _vertexBuffer);
    glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices,
GL_STATIC_DRAW);//Another option GL_DYNAMIC_DRAW
}
-(void) setupShader{
    _shader = [[RWTBaseEffect alloc]
initWithVertexShader:@"SimpleVertex.glsl"
fragmentShader:@"SimpleFragment.glsl"];
}

- (void)viewDidLoad {
    [super viewDidLoad];
    GLKView *view = (GLKView*)self.view;
    view.context = [[EAGLContext alloc]
initWithAPI:kEAGLRenderingAPIOpenGLES2];
```

```
    [EAGLContext setCurrentContext:view.context];
    [self setupShader];
    [self setupVertexBuffer];

}
- (void) glkView:(GLKView *)view drawInRect:(CGRect)rect
{
    //glClearColor(0, 0, 0, 1);
    glClearColor(0, 104.0/255, 55.0/255.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);

    //Using the loaded shader on to this model/primitive type
    [_shader prepareToDraw];

    //Enable Shader
    //telling the video card where the data is and binding them
togther to draw
    //Turn position attrib on
    glEnableVertexAttribArray(VertexAttribPosition);
    //Where is this data stored
    //param1: location
    //param2: how many; position is float[3] so it is 3,just
variable position
    //param3: type
    //param4: normalized, should be normalized
    //param5: how big is the structure where this information is
stored
    //param6: where is the offset inside the array; for position
it is 0 , but i will calc during run time
    glVertexAttribPointer(VertexAttribPosition, 3, GL_FLOAT,
GL_FALSE, sizeof(Vertex), (const GLvoid*)offsetof(Vertex,
Position));

    //Draw Buffer
    glBindBuffer(GL_ARRAY_BUFFER, _vertexBuffer);
    glDrawArrays(GL_TRIANGLES, 0, 3);

    glDisableVertexAttribArray(VertexAttribPosition);

}
- (void)update{

}
@end
```