**DATE: Tuesday November 11 2014**

**WHAT:**

**#2 Programmer meeting:**

**Using CIDetector to detect rectangles with the camera.**

**Then drawing a background over the rectangle.**

**Making ducks fly out from the background and make it look 3D.**

**3 PARTS: Using CIDetector, bg + duck over rectangle, challenge**

** **

**BEFORE AFTER RECTANGLE DETECT + BACKGROUND + DUCK**

**PART 1) Using CIDetector**

**//code from rectangleTest**

****

**//takes in an image**

**//The following pages are**

**//inside ViewController.m**

**//the code takes an image**

**//then detects rectangle**

**//then outputs the 4 points of it**

**//then draws a red line around the rectangle on the image**

#import "ViewController.h"

#import <CoreImage/CoreImage.h>

#import <QuartzCore/QuartzCore.h>

//declare variables here

@interface ViewController ()

{

//make a UIImageView which is a view

//that displays UIImages onscreen

UIImageView \*imageView;

}

@end

//implement variables here

@implementation ViewController

- (void)viewDidLoad {

[super viewDidLoad];

//make a UIImage with the image named socket.jpeg

UIImage \*image = [UIImage imageNamed:@"socket.jpeg"];

//make an UIImageView to display the UIImage onscreen

imageView = [[UIImageView alloc] initWithImage:image];

//add the imageView onto the screen

[self.view addSubview:imageView];

//detect for rectangles in that UIImage

[self detectForRectInUIImage:image];

}

//this function detects for rectangles in a UIImage

//a UIImage is an image

//the image we passed it is socket.jpeg

-(void)detectForRectInUIImage:(UIImage \*)pic

{

//Make a CIImage, which holds data from the UIImage

CIImage\* image = [CIImage imageWithCGImage:pic.CGImage];

//Make a CIDetector (Core Image Detector) which can detect 1 of 3 things

//1) rectangles

//2) faces

//3) QR codes

//CIDetectorTypeRectangle means we want to detect rectangles

//There’s also CIDetectorTypeFace, CIDetectorTypeQRCode

//CIDetectorAccuracyLow means we want low accuracy in detecting rectangles

//if we want better detection of them, we should use CIDetectorAccuracyHigh

CIDetector\* detector = [CIDetector detectorOfType:CIDetectorTypeRectangle

context:nil options:[NSDictionary dictionaryWithObject:CIDetectorAccuracyLow forKey:CIDetectorAccuracy]];

//[detector featuresInImage: image];

//this calls the featuresInImage function from the detector object

//with the parameter: image

//it returns an array containing many CIRectangleFeatures

NSArray\* features = [detector featuresInImage:image];

//Go through the array and get each individual CIRectangleFeature

//CIRectangleFeature is an object that contains information regarding

//the detected rectangle

//here I get the 4 points of the rectangle

//rectObject.topLeft

//rectObject.topRight

//rectObject.bottomRight

//rectObject.bottomLeft

//to get the x and y

//rectObject.bottomLeft.x

//rectObject.bottomLeft.y

for(CIRectangleFeature\* rectObject in features)

{

NSLog(@"Detected Rectangle's 4 pts (x,y):");

NSLog(@"TopLeft: (%i, %i)\n",(int)rectObject.topLeft.x,(int)rectObject.topLeft.y);

NSLog(@"TopRight: (%i, %i)\n",(int)rectObject.topRight.x,(int)rectObject.topRight.y);

NSLog(@"BottomRight: (%i, %i)\n",(int)rectObject.bottomRight.x,(int)rectObject.bottomLeft.y);

NSLog(@"BottomLeft: (%i, %i)\n",(int)rectObject.bottomLeft.x,(int)rectObject.bottomLeft.y);

//draw lines to connect the 4 points

//this will in effect draw the 4 sides

//of the rectangle

[self drawLine:rectObject.topLeft.x :rectObject.topLeft.y :rectObject.topRight.x :rectObject.topRight.y];

[self drawLine:rectObject.topRight.x :rectObject.topRight.y :rectObject.bottomRight.x :rectObject.bottomRight.y];

[self drawLine:rectObject.bottomRight.x :rectObject.bottomRight.y :rectObject.bottomLeft.x :rectObject.bottomLeft.y];

[self drawLine:rectObject.bottomLeft.x :rectObject.bottomLeft.y :rectObject.topLeft.x :rectObject.topLeft.y];

}

}

//(x1,y1) is first point of the line

//(x2,y2) is the second point of the line

-(void)drawLine:(int)x1 :(int)y1 :(int)x2 :(int)y2

{

UIBezierPath \*path = [UIBezierPath bezierPath];

[path moveToPoint:CGPointMake(x1, y1)];

[path addLineToPoint:CGPointMake(x2, y2)];

CAShapeLayer \*shapeLayer = [CAShapeLayer layer];

[shapeLayer setPath:path.CGPath];

[shapeLayer setStrokeColor: [UIColor redColor].CGColor];

[shapeLayer setLineWidth:5];

[imageView.layer addSublayer:shapeLayer];

}

//end the class

@end

**PART 2)**

**Use points of detected rectangle to place background over the rectangle.**

**Then make ducks fly out in 3D using SKActions.**

**//code from duckRectDetect**

****

**//Open the DrawScene.m class inside the DuckRectDetect xcode project**

**//the entire DrawScene.m is on the following pages**

**//functions to look at**

**//init**

**//declare variables**

**//update**

**//update rectangle points**

**//call drawBg**

**//touchesBegan**

**//makeDuckFlyUpRight called here**

**//drawBg**

**//draw background**

**//makeDuckFlyUpRight**

**//SKSpriteNode make**

**//spritesheet make**

**//skactions**

**//sound**

**//makeDuckFlyUpLeft**

**//makeDuckFlyHorizontallyRight**

**//makeYourOwnDuckMotion**

#import "DrawScene.h"

//4 functions you should look at:

//drawBg - draw a background over the detected rectangle

//makeDuck - makes a duck

//update - called 1000 times a second to update detected rectangle coordinates and draw background

//touchesBegan - touch screen to make a duck (makeDuck)

//declare variables here!

@implementation DrawScene

{

//the 4 corner points of the rectangle detected

CGPoint topLeft;

CGPoint topRight;

CGPoint bottomRight;

CGPoint bottomLeft;

//the SKShapeNode which will overlay the rectangle

SKShapeNode \*fourSidedFigure;

Sound\* soundPlayer;

}

//this function is the update loop

//it is called nonstop 1000 times a second

//used for when you need something to be

//constantly updated

//like updating the detected rectangle's coordinates

//or drawing the background over the detected rectangle

int seconds = 0;

-(void)update:(NSTimeInterval)currentTime

{

seconds++;

if(seconds%30==0)

{

//update the detected rectangle's 4 points

[self requestRectangleObjectCoordinates];

//draw the background

[self drawBg];

//print the rectangle coordinates

NSLog(@"Detected Rectangle's 4 pts (x,y):");

NSLog(@"TopLeft: (%i, %i)\n",(int)topLeft.x,(int)topLeft.y);

NSLog(@"TopRight: (%i, %i)\n",(int)topRight.x,(int)topRight.y);

NSLog(@"BottomRight: (%i, %i)\n",(int)bottomRight.x,(int)bottomLeft.y);

NSLog(@"BottomLeft: (%i, %i)\n",(int)bottomLeft.x,(int)bottomLeft.y);

NSLog(@"\n");

}

}

//touched the screen

-(void)touchesBegan:(NSSet \*)touches withEvent:(UIEvent \*)event{

NSArray \*allTouches = [[event allTouches] allObjects];

[self touchesBeganSettingButtons :allTouches];

//make the SKSpritenode duck spawn when touch the screen

[self makeDuckFlyUpRight];

[self makeDuckFlyUpLeft];

[self makeDuckFlyHorizontallyRight];

[self makeYourOwnDuckMotion];

}

//DRAW THE BACKGROUND OVER THE DETECTED RECTANGLE

//THE RECTANGLE HAS 4 CGPoints

//to get the x or y:

//topLeft.x or topLeft.y

//4 pts

// topLeft ....... topRight

// .......

// .......

//bottomLeft bottomRight

-(void)drawBg

{

//needed to make the SKShapeNode

CGPoint rect[] = {topLeft,topRight,bottomRight,bottomLeft,topLeft};

size\_t numPoints = 5;

//if SKShapeNode exists already, delete it

if(fourSidedFigure!=nil)

[fourSidedFigure removeFromParent];

//make SKShapeNode at the rectangle’s points and number of points (5)

fourSidedFigure = [SKShapeNode shapeNodeWithPoints:rect count:numPoints];

//make the rect green

[fourSidedFigure setFillColor:[UIColor greenColor]];

//give it an image for the background

SKTexture \*tex = [SKTexture textureWithImageNamed:@"duck\_hunt\_bg.png"];

//apply the image background to it

[fourSidedFigure setFillTexture:tex];

//put it behind the ducks

[fourSidedFigure setZPosition:-1];

//add it to the screen

[self addChild:fourSidedFigure];

}

//Make a duck, give it a spritesheet, give it animation, give it sound

-(void)makeDuckFlyUpRight

{

//make SKSpriteNode!

CGPoint center = CGPointMake((topLeft.x+topRight.x)/2, (topLeft.y + bottomLeft.y)/2);

SKSpriteNode \*duck = [SKSpriteNode spriteNodeWithImageNamed:@"duck"];

[duck setPosition:center];

[self addChild:duck];

//spritesheet!

//an atlas is a folder in the project

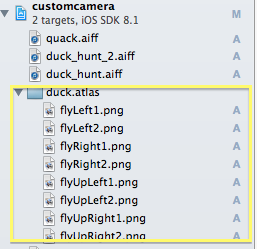
//here it is called duck.atlas

//it contains images for the animation spritesheets

//you use the images to make SKTexture

//then you make an SKAction called animateWithTextures

//and run it to apply the spritesheet



SKTextureAtlas \* atlas = [SKTextureAtlas atlasNamed:@"duck"];

SKTexture \* duckTex1 = [atlas textureNamed:@"flyUpRight1"];

SKTexture \* duckTex2 = [atlas textureNamed:@"flyUpRight2"];

NSArray \* runTexture = @[duckTex1,duckTex2];

SKAction\* runAnimation = [SKAction animateWithTextures:runTexture timePerFrame:0.075 resize:NO restore:NO];

[duck runAction:[SKAction repeatActionForever:runAnimation]];

//animations! SKAction!

//make duck gradually get larger by 500 and 500 in 2 sec

SKAction\* resizeOut = [SKAction resizeByWidth:500 height:500 duration:2];

//move the duck right by 700px and right by 700px in 2 sec

SKAction\* flyOut = [SKAction moveByX:700 y:700 duration:2];

//run the actions

[duck runAction:resizeOut];

[duck runAction:flyOut];

//sound!

soundPlayer = [[Sound alloc] init];

[soundPlayer playSound:@"quack"];}

-(void)makeDuckFlyUpLeft

{

}

-(void)makeDuckFlyHorizontallyRight

{

}

-(void)makeYourOwnDuckMotion

{

}

**PART 3) CHALLENGE:**

**Write the 3 other making flying duck functions, each doing different 3D-esque animations using the 4 rectangle points: topLeft, topRight, bottomRight, bottomLeft**

NOTE: inside touchesBegan is where the duck functions are called, so when you touch the screen, ducks are made

-(void)makeDuckFlyUpLeft

{

//like makeDuckFlyUpRight except fly left instead

//you will need to use the following textures “flyUpLeft1” and “flyUpLeft2”

//SKTexture \* duckTex1 = [atlas textureNamed:@"flyUpLeft1"];

//SKTexture \* duckTex2 = [atlas textureNamed:@"flyUpLeft2"];

}

-(void)makeDuckFlyHorizontallyRight

{

//make duck fly horizontally on the rectangle’s 2d plane

//will need to make duck move right

//will need to make duck get smaller or bigger

//depending on if that side is smaller or bigger than the opposite side

}

-(void)makeYourOwnDuckMotion

{

//make any duck motion you want

}

**DUE DATE: By next Tuesday Nov. 18 2014 at 11:59PM to calvin.tham@gmail.com**