```objc
//
//  HappinessViewController.h
//  Happiness
//
//  Created by Gabriel Parriaux on 11.10.12.
//  Copyright (c) 2012 gymo. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface HappinessViewController : UIViewController

@property (nonatomic) int happiness; // 0 est triste, 100 est très heureux

@end
```

```objc
//
//  HappinessViewController.m
//  Happiness
//
//  Created by Gabriel Parriaux on 11.10.12.
//  Copyright (c) 2012 gymo. All rights reserved.
//

#import "HappinessViewController.h"
#import "FaceView.h"

@interface HappinessViewController ()

@property (nonatomic, weak) IBOutlet FaceView *faceView;

@end

@implementation HappinessViewController

@synthesize happiness = _happiness;
@synthesize faceView = _faceView;

- (void)setFaceView:(FaceView *)faceView
{
    _faceView = faceView;
    [self.faceView addGestureRecognizer:[[UIPinchGestureRecognizer alloc]
initWithTarget:self.faceView action:@selector(pinch:)]];
}

- (void)setHappiness:(int)happiness
{
    _happiness = happiness;
    [self.faceView setNeedsDisplay];
}

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation
{
    return YES;
}

@end
```

```
//
//  FaceView.h
//  Happiness
//
//  Created by Gabriel Parriaux on 11.10.12.
//  Copyright (c) 2012 gymo. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface FaceView : UIView

@property (nonatomic) CGFloat scale;

- (void)pinch:(UIPinchGestureRecognizer *)gesture;

@end
```

```
//
//  FaceView.m
//  Happiness
//
//  Created by Gabriel Parriaux on 11.10.12.
//  Copyright (c) 2012 gymo. All rights reserved.
//

#import "FaceView.h"

@implementation FaceView

@synthesize scale = _scale;

#define DEFAULT_SCALE 0.90

- (CGFloat)scale
{
    if (!_scale) {
        return DEFAULT_SCALE;
    } else return _scale;
}

- (void)setScale:(CGFloat)scale
{
    if (_scale != scale) {
        _scale = scale;
        [self setNeedsDisplay];
    }
}

- (void)pinch:(UIPinchGestureRecognizer *)gesture
{
    if ((gesture.state == UIGestureRecognizerStateChanged) || (gesture.state ==
UIGestureRecognizerStateEnded)) {
        self.scale *= gesture.scale;
        gesture.scale = 1;
    }
}

- (void)drawCircleAtPoint:(CGPoint)p
              withRadius:(CGFloat)radius
               inContext:(CGContextRef)context
{
    UIGraphicsPushContext(context);

    CGContextBeginPath(context);
    CGContextAddArc(context, p.x, p.y, radius, 0, 2*M_PI, YES);
    CGContextStrokePath(context);

    UIGraphicsPopContext();
}


- (void)drawRect:(CGRect)rect
{
    CGContextRef context = UIGraphicsGetCurrentContext();

        // dessiner la figure (un cercle)
    CGPoint midPoint;
    midPoint.x = self.bounds.origin.x + self.bounds.size.width / 2;
    midPoint.y = self.bounds.origin.y + self.bounds.size.height / 2;

    CGFloat size = self.bounds.size.width / 2;

    if (self.bounds.size.height < self.bounds.size.width) size = self.bounds.size.height / 2;

    size *= self.scale;

    CGContextSetLineWidth(context, 5.0);
    [[UIColor greenColor] setStroke];

    [self drawCircleAtPoint:midPoint withRadius:size inContext:context];

```

```
        // dessiner deux yeux (deux cercles)

#define EYE_H 0.35
#define EYE_V 0.35
#define EYE_RADIUS 0.10

    CGPoint eyePoint;
    eyePoint.x = midPoint.x - size * EYE_H;
    eyePoint.y = midPoint.y - size * EYE_V;

    [self drawCircleAtPoint:eyePoint withRadius:size * EYE_RADIUS inContext:context];

    eyePoint.x += size * EYE_H * 2;

    [self drawCircleAtPoint:eyePoint withRadius:size * EYE_RADIUS inContext:context];


        // pas de nez

        // dessiner la bouche (courbe de Bézier)
#define MOUTH_H 0.45
#define MOUTH_V 0.40
#define MOUTH_SMILE 0.25

    CGPoint mouthStart;
    mouthStart.x = midPoint.x - MOUTH_H * size;
    mouthStart.y = midPoint.y + MOUTH_V * size;

    CGPoint mouthEnd = mouthStart;
    mouthEnd.x += MOUTH_H * size * 2;

    CGPoint mouthCP1 = mouthStart;
    mouthCP1.x += MOUTH_H * size * 2/3;
    CGPoint mouthCP2 = mouthEnd;
    mouthCP2.x -= MOUTH_H * size * 2/3;

    float smile = 1;

    CGFloat smileOffset = MOUTH_SMILE * size * smile;

    mouthCP1.y += smileOffset;
    mouthCP2.y += smileOffset;

    CGContextBeginPath(context);
    CGContextMoveToPoint(context, mouthStart.x, mouthStart.y);
    CGContextAddCurveToPoint(context, mouthCP1.x, mouthCP1.y, mouthCP2.x, mouthCP2.y,
mouthEnd.x, mouthEnd.y);
    CGContextStrokePath(context);

}




@end
```