```
//
//  moon_diplay_view.m
//  Moon Calender 2
//
//  Created by Nicolas Guyader on 1/7/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//
#import <Cocoa/Cocoa.h>
#import "moon_diplay_view.h"


@implementation moon_diplay_view

- (id)initWithFrame:(NSRect)frame {


    if (k_imput==0) {
        k_imput=0.1;
    }


    [super initWithFrame:frame];
    // First, we set default values for the various parameters.
    jojo = NSMakeRect(23, 45, 56, 23);


    return self;
}


- (void) set_k:(double)k_update;
{
    k_imput = k_update;

    printf(" K update %f\n", k_update);


    [self setNeedsDisplay:YES];


}


- (void)drawRect:(NSRect)dirtyrect {







    NSImage *imageFromBundle = [NSImage imageNamed:@"Full Moon.jpg"];

    [[NSGraphicsContext currentContext]
     setImageInterpolation: NSImageInterpolationHigh];
    NSSize viewSize = [self bounds].size;
    NSPoint viewCenter;
    viewCenter.x = viewSize.width  * 0.50;
```

```
viewCenter.y = viewSize.height * 0.50;

// calculate image rect to draw the image to
NSSize size = [imageFromBundle size];
NSRect image_rect;
image_rect.origin.x = viewCenter.x - (size.width  * 0.50);
image_rect.origin.y = viewCenter.y - (size.height * 0.50);
image_rect.size = size;

printf("\n fuck that XXXXXXXXXX \n");

// draw composite image
[imageFromBundle drawInRect: image_rect
                   fromRect: NSZeroRect
                  operation: NSCompositeSourceOver
                   fraction: 1.0];

NSBezierPath * path = [NSBezierPath bezierPathWithRect:image_rect];
[path setLineWidth:2.5];
[[NSColor whiteColor] set];
[path stroke];

// setup basic size and color properties
float dm = 110 * 2;
float R = dm * 0.50;



NSColor * black = [NSColor blackColor];
[black colorWithAlphaComponent:0.9];

NSBezierPath *path1;

// find the center of the view
float center = [self bounds].size.width  * 0.50;
float middle = [self bounds].size.height * 0.50;

// create a rect in the center
NSPoint origin  = { center - R, middle - R };
NSRect mainOval = { origin.x, origin.y, dm, dm };

// create a oval bezier path using the rect
path1 = [NSBezierPath bezierPathWithOvalInRect:mainOval];
[path1 setLineWidth:0];

// draw the path

[black set];[path1 stroke];

double k,a,evolution,E,f,g;


    printf(" Kimput %f\n", k_imput);
k= k_imput * R;
printf(" K : %f\n", k);
a = R-k;
evolution = 0;
```

```
E = evolution * k;
f = (a * E)/k;
g = sqrt( R*R - (((a*E)/k)+E)* (((a*E)/k)+E));


NSBezierPath* croissant = [NSBezierPath bezierPath];
[croissant moveToPoint:NSMakePoint(viewCenter.x,viewCenter.y + R)];
[croissant lineToPoint:NSMakePoint(viewCenter.x + f,viewCenter.y +g )
    ];

int i;
int count = 200;
for ( i = 1; i < count; i++ )
{
    evolution = evolution + 0.005;
    E = evolution * k;
    f = (a * E)/k;
    g = sqrt( R*R - (((a*E)/k)+E)* (((a*E)/k)+E));
    if (E == 0) {
        f = a;
        g = 0;
    }

    [croissant lineToPoint:NSMakePoint(viewCenter.x + f,viewCenter.y
        +g )];
}

evolution = 1;
for ( i = 1; i < count; i++ )
{
    evolution = evolution - 0.005;
    E = evolution * k;
    f = (a * E)/k;
    g = sqrt( R*R - (((a*E)/k)+E)* (((a*E)/k)+E));
    if (E == 0) {
        f = a;
        g = 0;
    }

    [croissant lineToPoint:NSMakePoint(viewCenter.x + f,viewCenter.y
        -g )];
}

// arc from the center to construct right side
NSPoint mainOvalCenter = { center, middle };
[croissant appendBezierPathWithArcWithCenter: mainOvalCenter
                                      radius: R
                                  startAngle: 270
                                    endAngle: 90
                                   clockwise: NO];

[black set];[croissant stroke];[croissant fill];
```

```
    // Drawing code here.
}
- (BOOL)isOpaque {
    return YES;
}
- (void)dealloc {

    [super dealloc];
}
@end
```