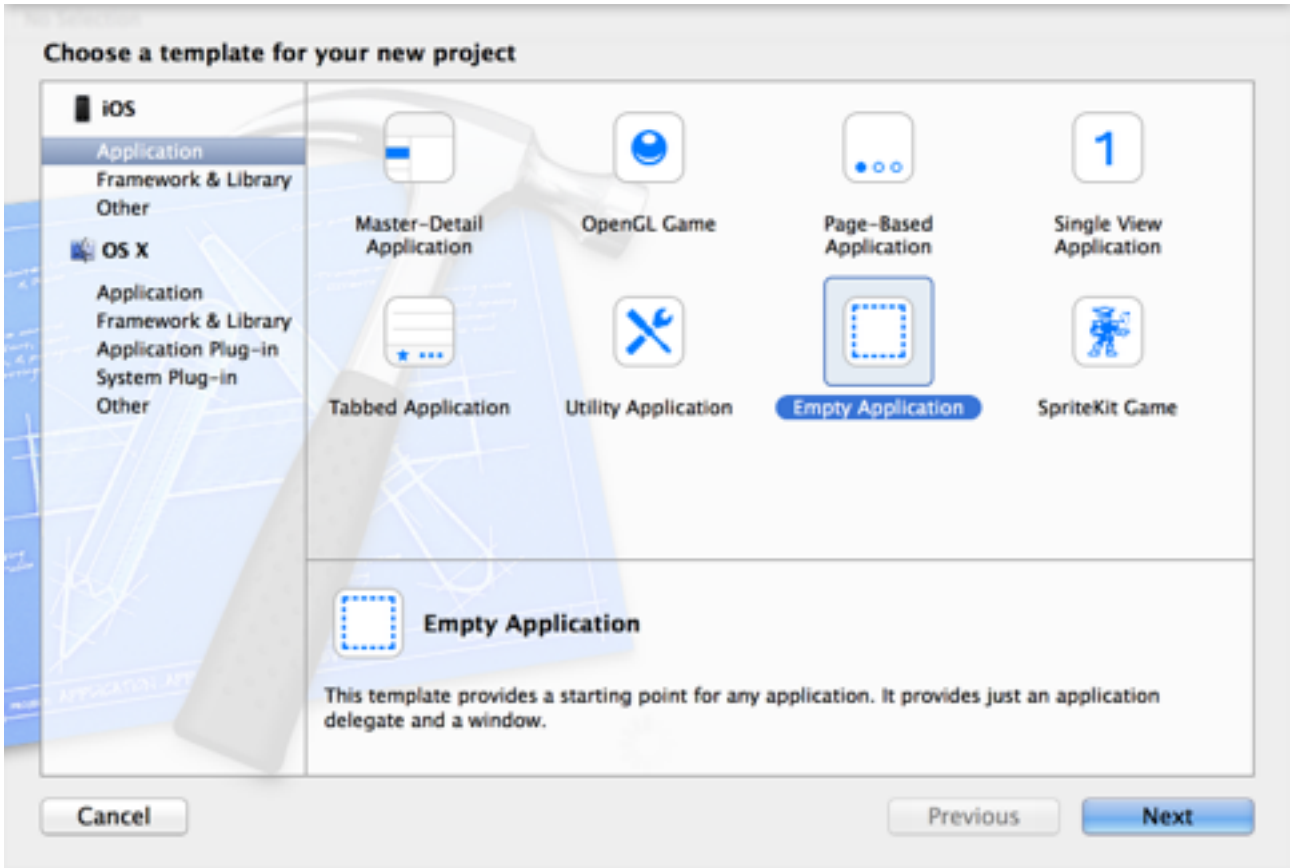
Eventos de toque

Nessa aula criaremos um projeto chamado Paint, este projeto será um app de desenhar linhas baseado nos eventos básicos de toque:

- touchesBegan:withEvent:
- touchesMoved:withEvent:
- touchesEnded:withEvent:
- touchesCancelled:withEvent:

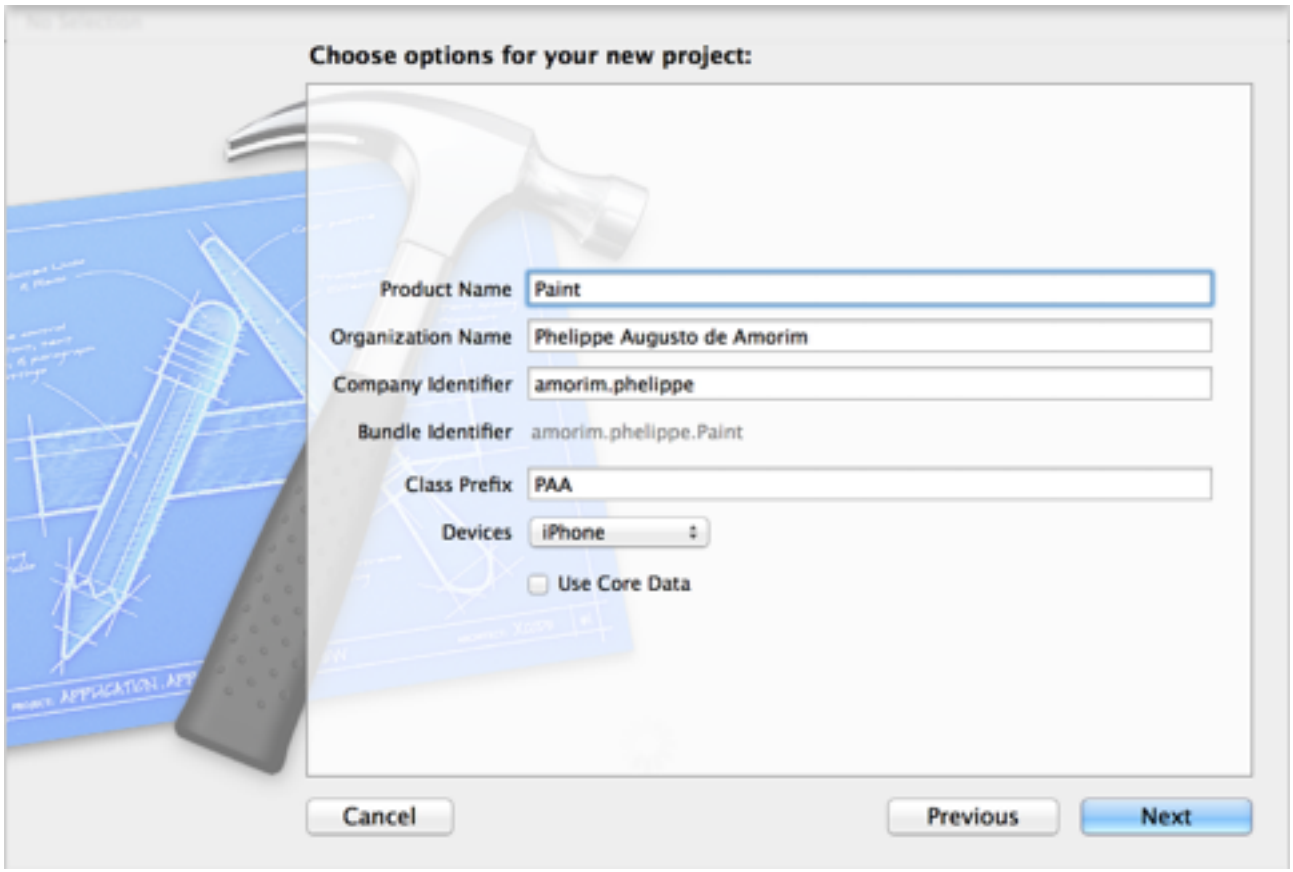
Preparação

Crie um projeto vazio.



Nomeie o projeto como Paint, lembrando que o target do projeto é o iPhone.

Não esqueça de colocar o seu prefixo de classe.

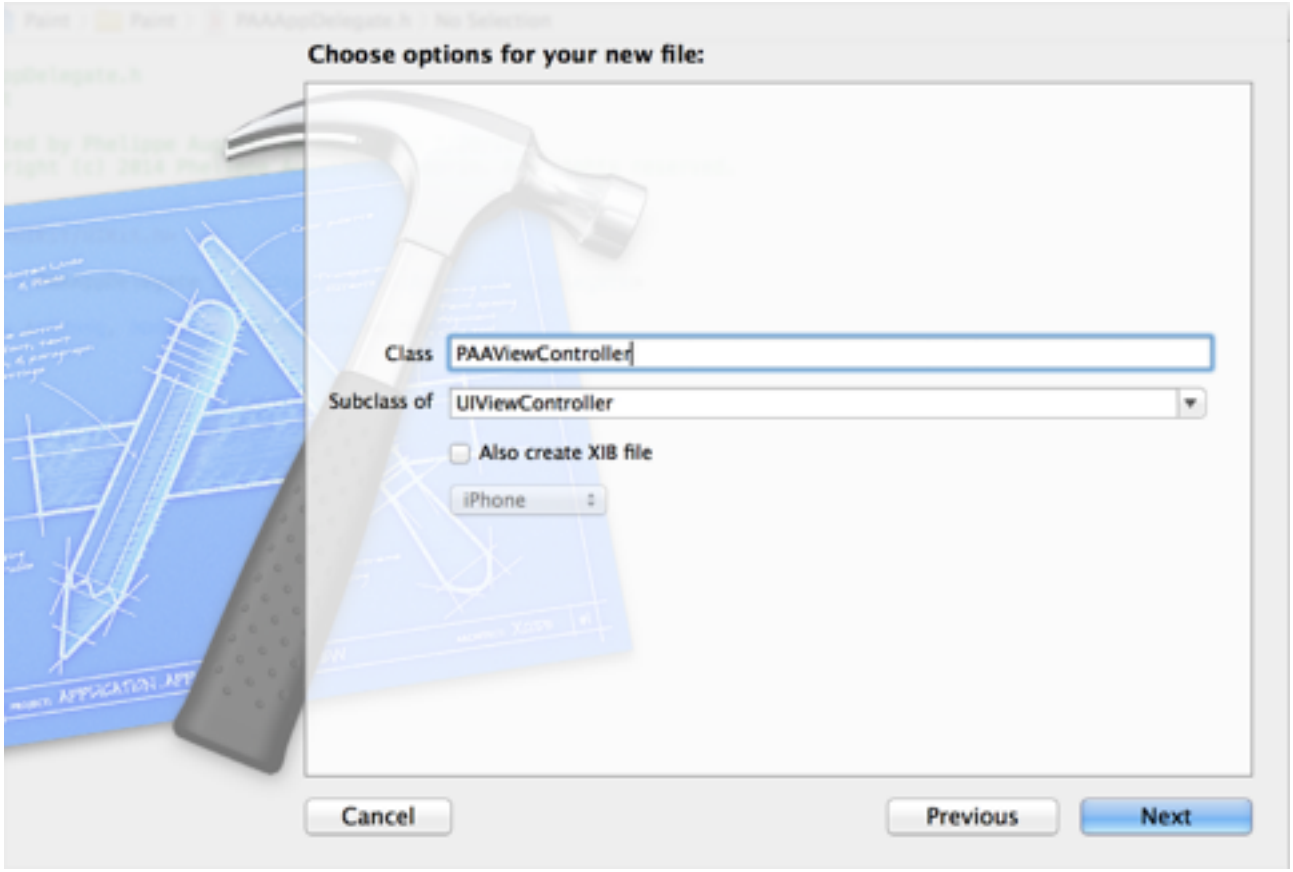


Execução

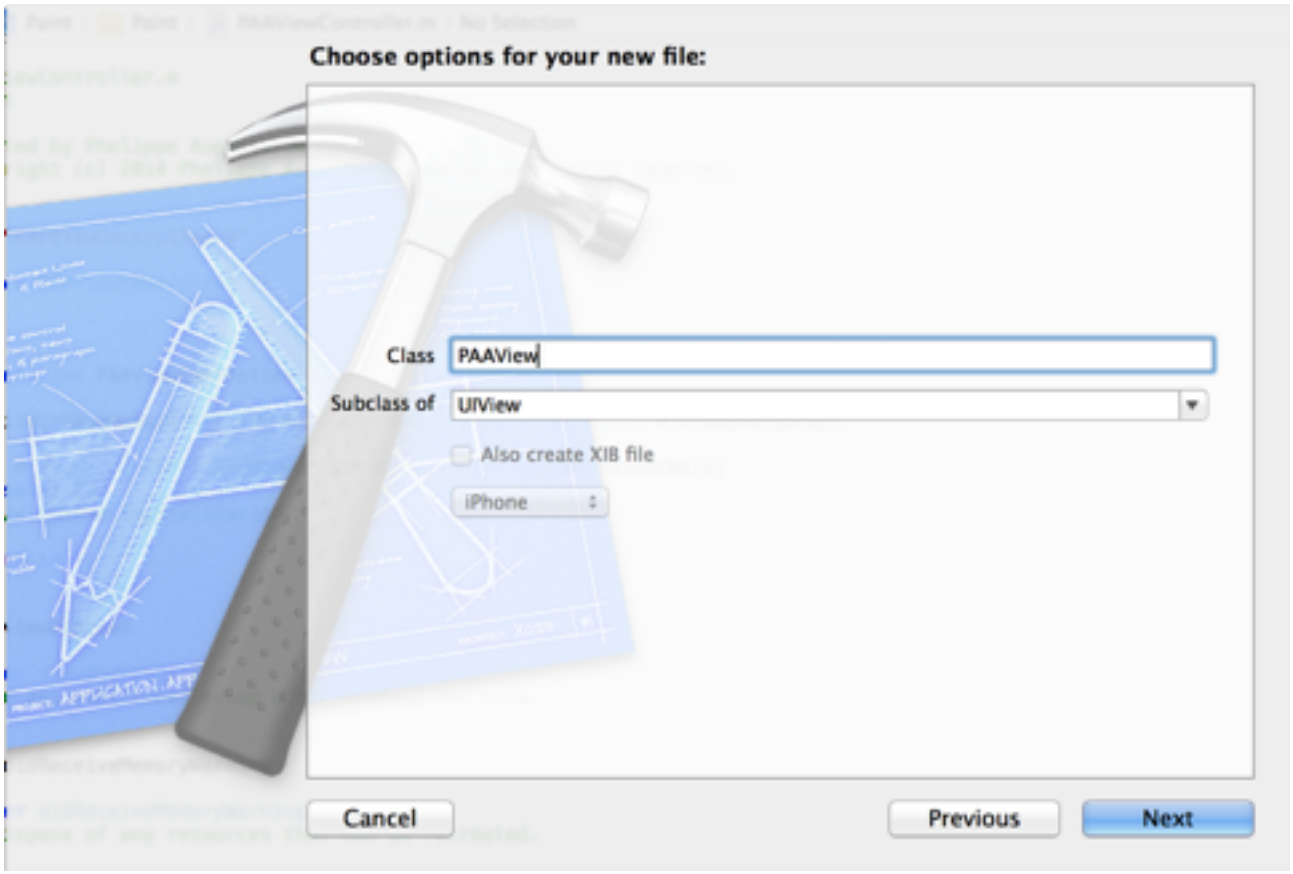
Para o nosso projeto devemos criar três classes, a ViewController, a View e a Line.

Então, mãos à obra.

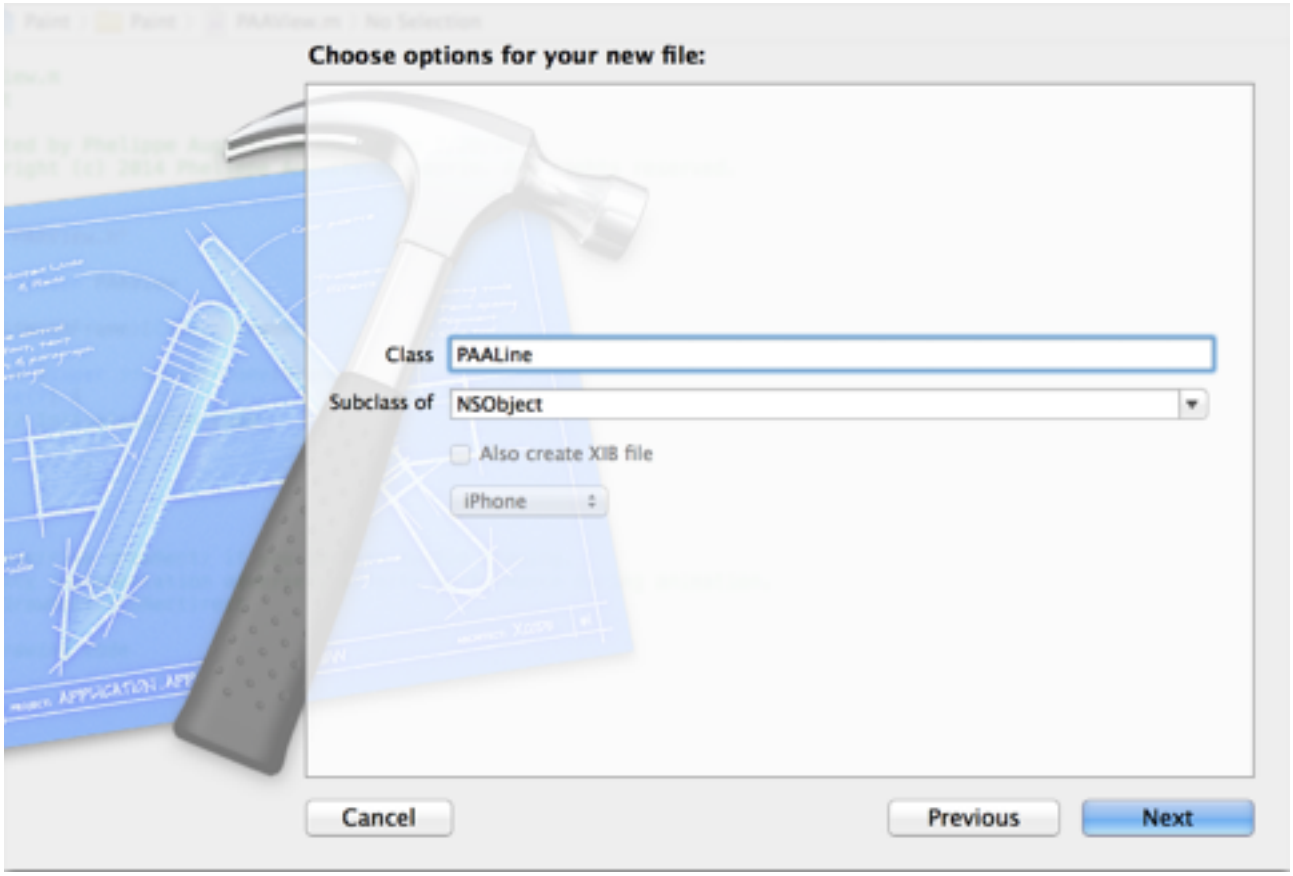
Crie a ViewController, lembrando de não esquecer o prefixo antes do nome da classe.



Crie a View, não esquecendo também do prefixo antes do nome da classe.



E por ultimo, mas não menos importante, crie a Line, não esquecendo também do prefixo antes do nome da classe.



Com todas as classes criadas vamos colocar a nossa View como a view da ViewController.

Na ViewController.m remova todos os métodos criados automaticamente e deixe apenas o viewDidLoad.

```
#import "PAAViewController.h"

@interface PAAViewController ()

@end

@implementation PAAViewController
- (void)viewDidLoad
{
    [super viewDidLoad];
}

@end
```

Ainda na ViewController.m importe a View e coloque-a como a view da ViewController

```
#import "PAAViewController.h"
#import "PAAView.h"

@interface PAAViewController ()

@end

@implementation PAAViewController
- (void)viewDidLoad
{
    [super viewDidLoad];
    self.view = [[PAAView alloc] initWithFrame:CGRectZero];
}

@end
```

No AppDelegate.m importe a ViewController, no application:didFinishLaunchingWithOptions crie uma instancia da ViewController e coloque-a como rootViewController

```
#import "PAAAppDelegate.h"
#import "PAAViewController.h"

@implementation PAAAppDelegate

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[UIWindow alloc]
                   initWithFrame:[[UIScreen mainScreen] bounds]];
    // Override point for customization after application launch.

    PAAViewController *vc = [[PAAViewController alloc] init];
    self.window.rootViewController = vc;

    self.window.backgroundColor = [UIColor whiteColor];
    [self.window makeKeyAndVisible];
    return YES;
}
```

Vamos fazer um teste para garantir que está tudo correto até agora.

Na View.m no initWithFrame coloque o backgroundColor como lightGray.

```
- (id)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if (self) {
        self.backgroundColor = [UIColor lightGrayColor];
    }
    return self;
}
```

Execute o app. Ele deve estar assim:

Com essa parte inicial funcionando, vamos alterar a Line de modo que esta tenha como properties os pontos que a representam.

Na Line.h declare duas properties CGPoint.

```
@interface PAALine : NSObject

@property (nonatomic) CGPoint begin;
@property (nonatomic) CGPoint end;

@end
```

Agora vamos ao que realmente interessa, detectar os toques do usuário na tela e traçar retas com esses toques.

Na View.m importe a Line e crie uma property Line que representará a linha que está sendo desenhada.
Ps: Caso seja necessário crie o @interface antes do @implementation.

```
#import "PAAView.h"
#import "PAALine.h"

@interface PAAView ()

@property (nonatomic) PAALine *currentLine;

@end

@implementation PAAView
```

Na Line.h adicione a assinatura do método stroke.

```
@interface PAALine : NSObject

@property (nonatomic) CGPoint begin;
@property (nonatomic) CGPoint end;

- (void)stroke;

@end
```

Na Line.m crie a implementação do método stroke.

```
- (void)stroke
{
    UIBezierPath *bp = [UIBezierPath bezierPath];

    bp.lineWidth = 10;
    bp.lineCapStyle = kCGLineCapRound;

    [bp moveToPoint:self.begin];
    [bp addLineToPoint:self.end];
    [bp stroke];
}
```

Na View.m sobrescreva o método drawRect para desenhar a Line.

```
- (void)drawRect:(CGRect)rect
{
    if (self.currentLine) {
        [[UIColor blackColor] set];
        [self.currentLine stroke];
    }
}
```

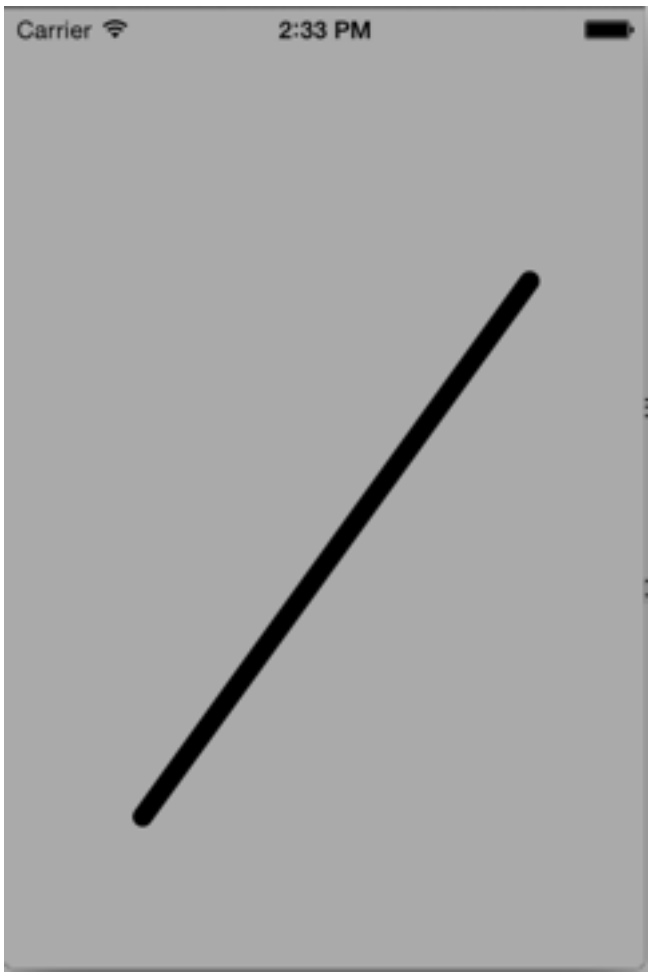
Com a View preparada para desenhar a Line, vamos transformar toques na tela em linhas. Para fazer isso, vamos reinstanciar a Line toda vez que o usuário iniciar um toque na tela, colocando o ponto inicial e ponto final no ponto em que o usuário tocou. Quando o toque se mover, o ponto final será atualizado. Quando o toque terminar a linha estará finalizada.

Para isso temos que sobrescrever os métodos touchesBegan:withEvent:, touchesMoved:withEvent: e touchesEnded:withEvent:.

Na View.m sobrescreva esses métodos.

```
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    UITouch *t = [touches anyObject];
    CGPoint location = [t locationInView:self];

    self.currentLine = [[PAALine alloc] init];
    self.currentLine.begin = location;
    self.currentLine.end = location;

    [self setNeedsDisplay];
}

- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event
{
    UITouch *t = [touches anyObject];
    CGPoint location = [t locationInView:self];

    self.currentLine.end = location;

    [self setNeedsDisplay];
}

- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event
{
    UITouch *t = [touches anyObject];
    CGPoint location = [t locationInView:self];

    self.currentLine.end = location;

    [self setNeedsDisplay];
}
```

Com essa alteração já é possível desenhar uma linha na tela. Execute o app e faça um teste.



Salvando linhas antigas

Com essas implementações o app deve estar conseguindo desenhar uma linha por vez, porém toda vez que desenhamos uma linha nova a antiga apaga.

Vamos alterar o app de modo que ele salve as linhas antigas.

Na View.m adicione uma property MutableArray com o nome de finishedLines.

```
@interface PAAView ()

@property (nonatomic) PAALine *currentLine;
@property (nonatomic) NSMutableArray *finishedLines;

@end

@implementation PAAView
```

Ainda na View.m altere o initWithFrame para instanciar o finishedLines.

```
- (id)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if (self) {
        _finishedLines = [[NSMutableArray alloc] init];

        self.backgroundColor = [UIColor lightGrayColor];
    }
    return self;
}
```

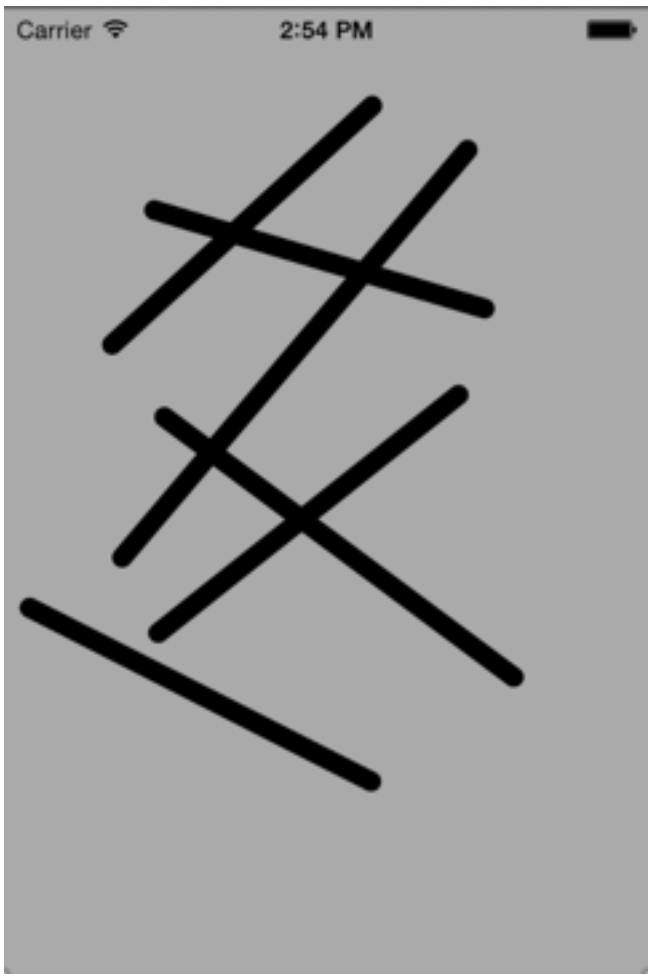
Vamos alterar o drawRect para desenhar também o finishedLines.

```
- (void)drawRect:(CGRect)rect
{
    [[UIColor blackColor] set];
    for (PAALine *line in self.finishedLines) {
        [line stroke];
    }

    if (self.currentLine) {
        [[UIColor blackColor] set];
        [self.currentLine stroke];
    }
}
```

Também alteraremos o touchesEnded:withEvent: para armazenar no MutableArray as linhas finalizadas.

```
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event
{
    [self.finishedLines addObject:self.currentLine];

    self.currentLine = nil;

    [self setNeedsDisplay];
}
```

Agora o app já armazena quantas linhas forem desenhadas. Execute o código e veja o funcionamento atual.



Com esse funcionamento não se tem diferença entre as linhas já desenhadas e a linha que está sendo desenhada. Vamos alterar a cor da linha que está sendo desenhada.

Na View.m no método drawRect altere para a linha que está sendo desenhada seja vermelha.

```
- (void)drawRect:(CGRect)rect
{
    [[UIColor blackColor] set];
    for (PAALine *line in self.finishedLines) {
        [line stroke];
    }

    if (self.currentLine) {
        [[UIColor redColor] set];
        [self.currentLine stroke];
    }
}
```

Execute o app e veja se a linha que está sendo desenhada está vermelha.

Múltiplos toques

Por padrão uma UIView aceita apenas um toque por vez, vamos atualiza-lo para aceitar múltiplos toques.

Va View.m altere o initWithFrame.

```
- (id)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if (self) {
        _finishedLines = [[NSMutableArray alloc] init];

        self.backgroundColor = [UIColor lightGrayColor];

        self.multipleTouchEnabled = YES;
    }
    return self;
}
```

Com essa alteração a View poderá receber vários toques simultaneamente. Mas para isso funcionar corretamente um refactory deverá ser feito no código, começando pela property currentLine que aceita apenas uma linha sendo desenhada por vez, passando pelo método drawRect que desenha apenas uma linha em progresso e terminando nos eventos de toque.

Na View.m remova a property currentLine e adicione uma nova property MutableDictionary linesInProgress.

```
@interface PAAView ()

@property (nonatomic) NSMutableDictionary *linesInProgress;
@property (nonatomic) NSMutableArray *finishedLines;

@end
```

Altere o initWithFrame para instanciar o linesInProgress.

```
- (id)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if (self) {
        _linesInProgress = [[NSMutableDictionary alloc] init];

        _finishedLines = [[NSMutableArray alloc] init];

        self.backgroundColor = [UIColor lightGrayColor];

        self.multipleTouchEnabled = YES;
    }
    return self;
}
```

Atualizaremos agora o drawRect.

```
- (void)drawRect:(CGRect)rect
{
    [[UIColor blackColor] set];
    for (PAALine *line in self.finishedLines) {
        [line stroke];
    }

    [[UIColor redColor] set];
    for (NSValue *key in self.linesInProgress) {
        PAALine *line = self.linesInProgress[key];
        [line stroke];
    }
}
```

Agora precisamos atualizar os métodos de toque para eles serem capazes de tratar vários toques.

Iniciaremos pelo touchBegan:withEvent:.

```
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    for (UITouch *t in touches) {
        CGPoint location = [t locationInView:self];

        PAALine *line = [[PAALine alloc] init];
        line.begin = location;
        line.end = location;

        NSValue *key = [NSValue valueWithNonretainedObject:t];

        [self.linesInProgress setObject:line forKey:key];
    }

    [self setNeedsDisplay];
}
```

Observe que o método valueWithNonretainedObject cria uma instancia de NSValue que contém o endereço de memória do UITouch associado à Line.

Vamos atualizar agora o touchesMoved:withEvent:.

```
- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event
{
    for (UITouch *t in touches) {
        NSValue *key = [NSValue valueWithNonretainedObject:t];
        CGPoint location = [t locationInView:self];

        PAALine *line = self.linesInProgress[key];
        line.end = location;
    }

    [self setNeedsDisplay];
}
```

Por ultimo atualizaremos o touchesEnded:withEvent:.

```
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event
{
    for (UITouch *t in touches) {
        NSValue *key = [NSValue valueWithNonretainedObject:t];
        PAALine *line = self.linesInProgress[key];

        [self.finishedLines addObject:line];
        [self.linesInProgress removeObjectForKey:key];
    }

    [self setNeedsDisplay];
}
```

Agora o app está preparado para receber vários toques simultaneamente. Execute e faça alguns testes.

Para finalizar precisamos implementar o touchesCancelled:withEvent:, ele é chamado quando o aplicativo é interrompido pelo SO, por exemplo, quando se recebe uma ligação.

```
- (void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event
{
    for (UITouch *t in touches) {
        NSValue *key = [NSValue valueWithNonretainedObject:t];
        [self.linesInProgress removeObjectForKey:key];
    }

    [self setNeedsDisplay];
}
```

Agora, a primeira versão do seu app está pronta.

Exercício 1

Quando o aplicativo for fechado, salve as linhas. Recarregue-as quando o aplicativo for reaberto.

Exercício 2

Faça com que a cor da linha traçada seja definida pelo ângulo dela.

Exercício 3

Use dois dedos para desenhar círculos.