Implementing Visualization Apps on iOS Devices

Douglass Turner
Elastic Image Software
tweets: @dugla
email: douglass.turner@gmail.com

iOS software development is done using Objective-C an object-oriented superset of C.

It was developed in the spirit of Smalltalk.

- Objective-C is simple, approachable, lightweight, and pragmatic. No frills.
- Objective-C and C can be intermingled freely.
- Think OOP for C hackers and Unix heads.

Objective-C Supports

- **Class**: Grouping of data + code. The type of an object.
- **Instance**: A specific copy of a class.
- **Method**: A message that an object can respond to.
- **Instance variable (ivar)**: A piece of data belonging to an object

Interface - .h file

```
@interface className : superclassName

@property(nonatomic, retain) Type *propertyForType;

+(return_type)classMethod;

+(return_type)classMethodWithParam:(paramType) paramName;

-(return_type)instanceMethodWithParam1:(param1Type)param1Name
                           andParam2:(param2Type) param2Name;

@end
```

Implementation - .m file

```
@implementation classname

@synthesize propertyForType;

+(return_type)classMethod {
    // do stuff
}

+(return_type)classMethodWithParam:(paramType) paramName {
    // do stuff
}

-(return_type)instanceMethodWithParam1:(param1Type)param1Name
                             andParam2:(param2Type) param2Name {
    // do stuff
}

@end
```

Apple style tends towards `longSelfDocumentingMethodNames`.

Instantiation

```
self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
```

Property setting

```
self.window.backgroundColor = [UIColor whiteColor];
```

Message passing

```
[self.window makeKeyAndVisible];
```

Objective-C Types

Dynamically-typed: `id whoCaresWhatThisIs;`

Statically-typed: `Thang* aThang;`

Selectors identify methods by name

```
@interface Observer : NSObject

- (id)initWithTarget:(id)object action:(SEL)action;
@property(nonatomic, strong) id targetObject;
@property(nonatomic) SEL targetAction;

@end
```

Selectors identify methods by name

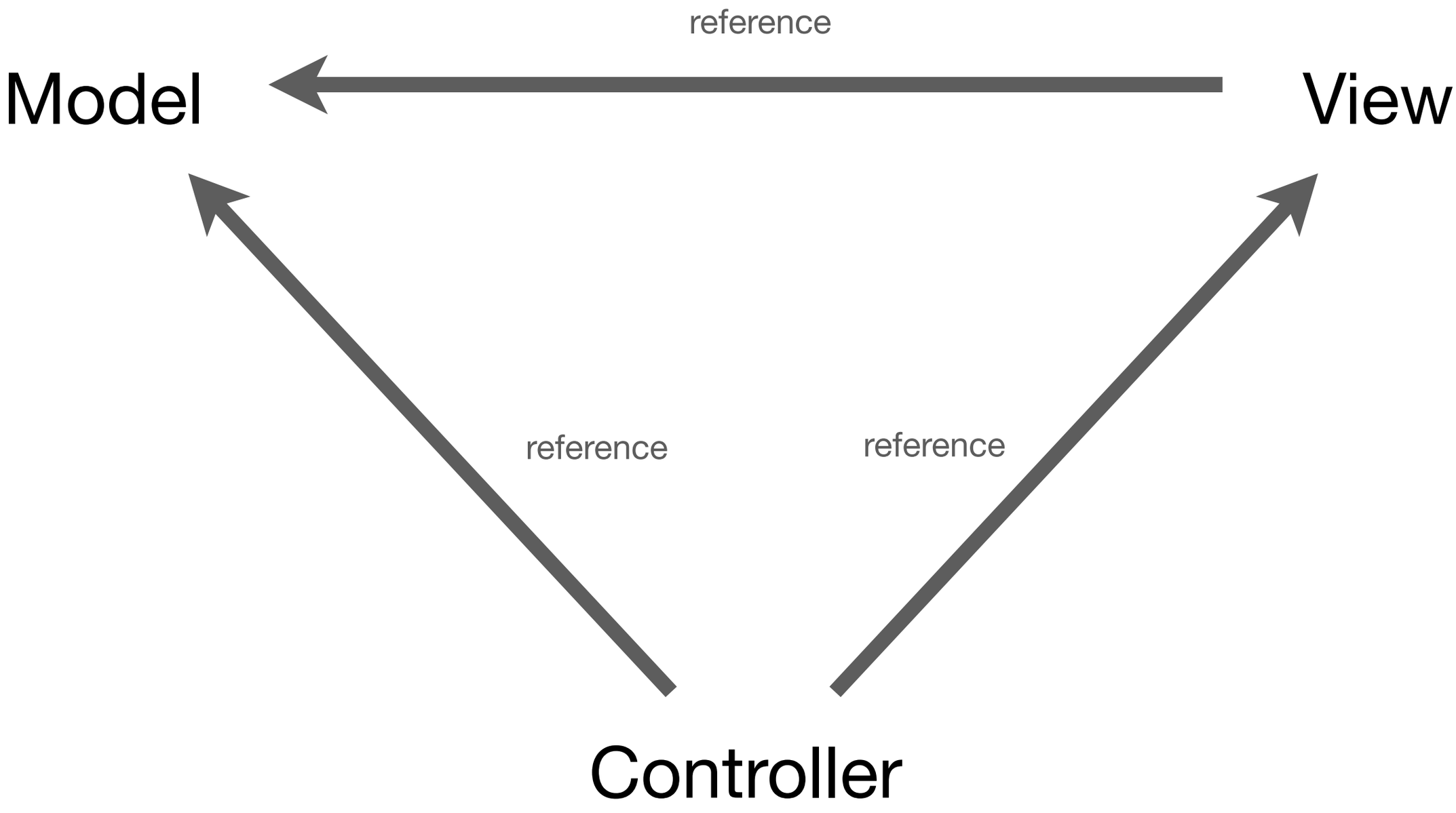
```
observer = [[Observer alloc] initWithTarget:self
                                     action:@selector(updateDisplay:)];
```

Objective Lifecycle

- Create an instance.
- Futz with it (Send messages. Pass it around.)
- Discard it.

iOS is designed around one foundational pattern: Model View Controller.

Much of iOS development - excluding Model development - involves customization and extension of this pattern.


reference
Model
View
reference
reference
Controller

Model

View

Key-value **O**bserving
or
Notification

Key-value **O**bserving
or
Notification

Target - Action

Controller

iOS has rich support for a loose, flat, decoupled style of programming

- Notification
- Target - Action
- Key-value Observing (KVO)
- Block & Dispatch Queue
- Delegation (Protocol)

Notifications

Notification

Notifier

```
[[NSNotificationCenter defaultCenter] postNotificationName:MyNotification object:self];
```

Notification

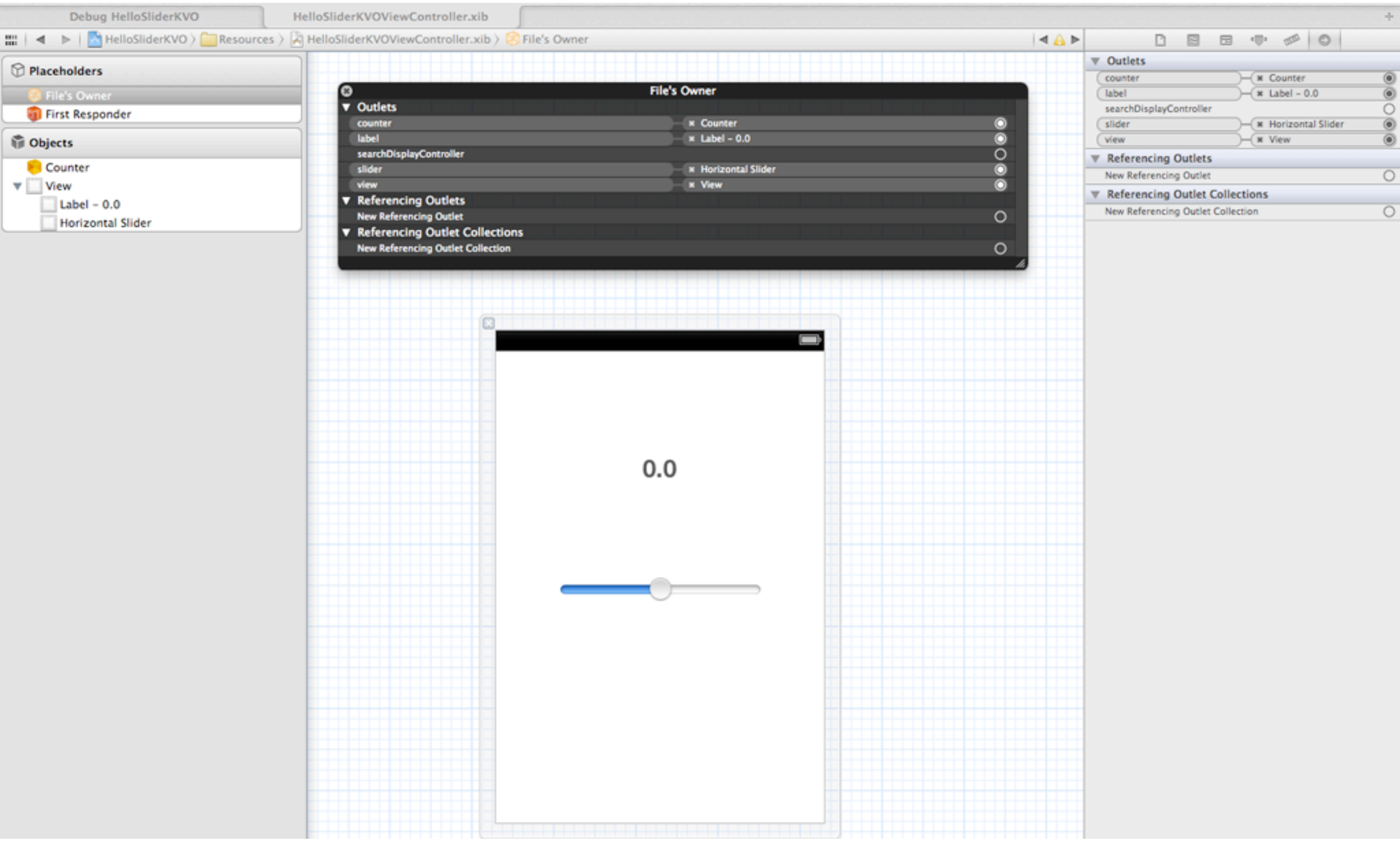
Notification respondent

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                         selector:@selector(respondToMyNotification:)
                                             name:MyNotification
                                           object:nil];
```

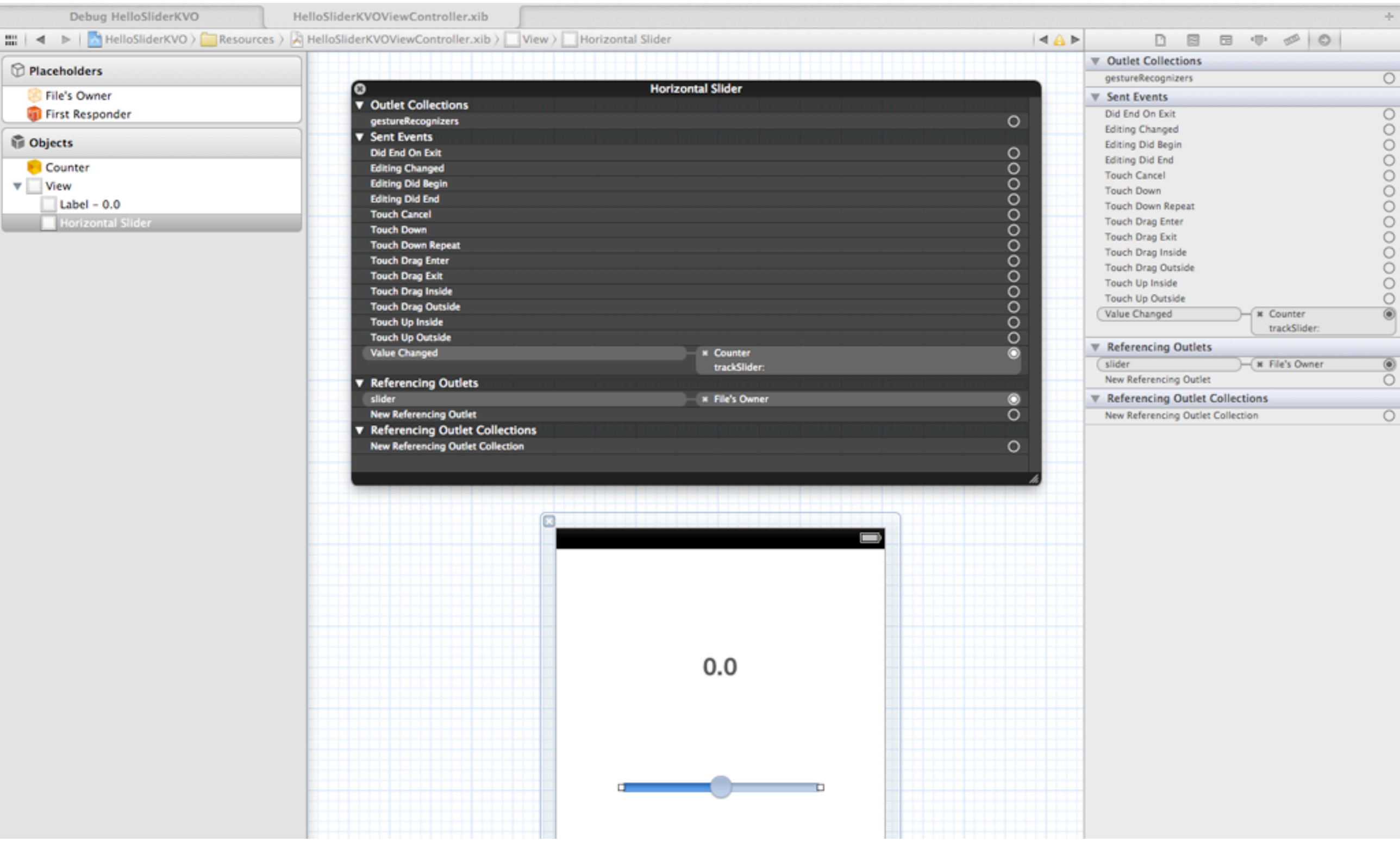
```
- (void) respondToMyNotification:(NSNotification *)notification {
    // do stuff
}
```

Target - Action

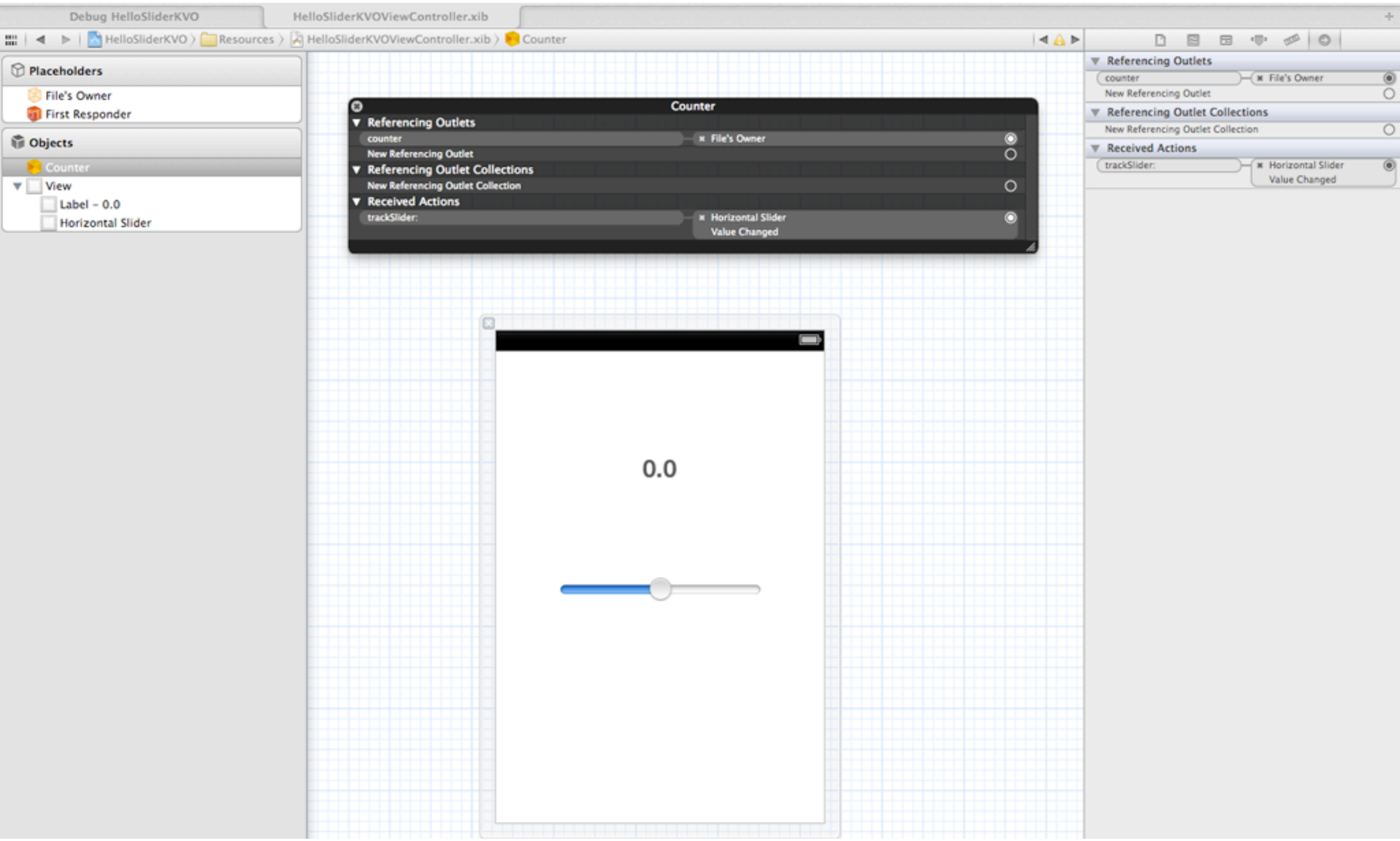
Target - Action



Target - Action



Target - Action



Target - Action

```
@interface Counter : NSObject

@property(nonatomic, strong) NSNumber *count;
- (IBAction)trackSlider:(UISlider *)slider;

@end
```

Target - Action

```
@implementation Counter

-(IBAction) trackSlider:(UISlider *)slider; {
    self.count = [NSNumber numberWithFloat:slider.value];
}

@end
```

Target - Action

```
@class Observer;
@class Counter;

@interface EIViewController : UIViewController
@property(nonatomic, strong) IBOutlet UILabel *label;
@property(nonatomic, strong)          Observer *observer;
@property(nonatomic, strong) IBOutlet Counter *counter;
- (void)updateLabel:(NSNumber *)newValue;
@end
```

Target - Action

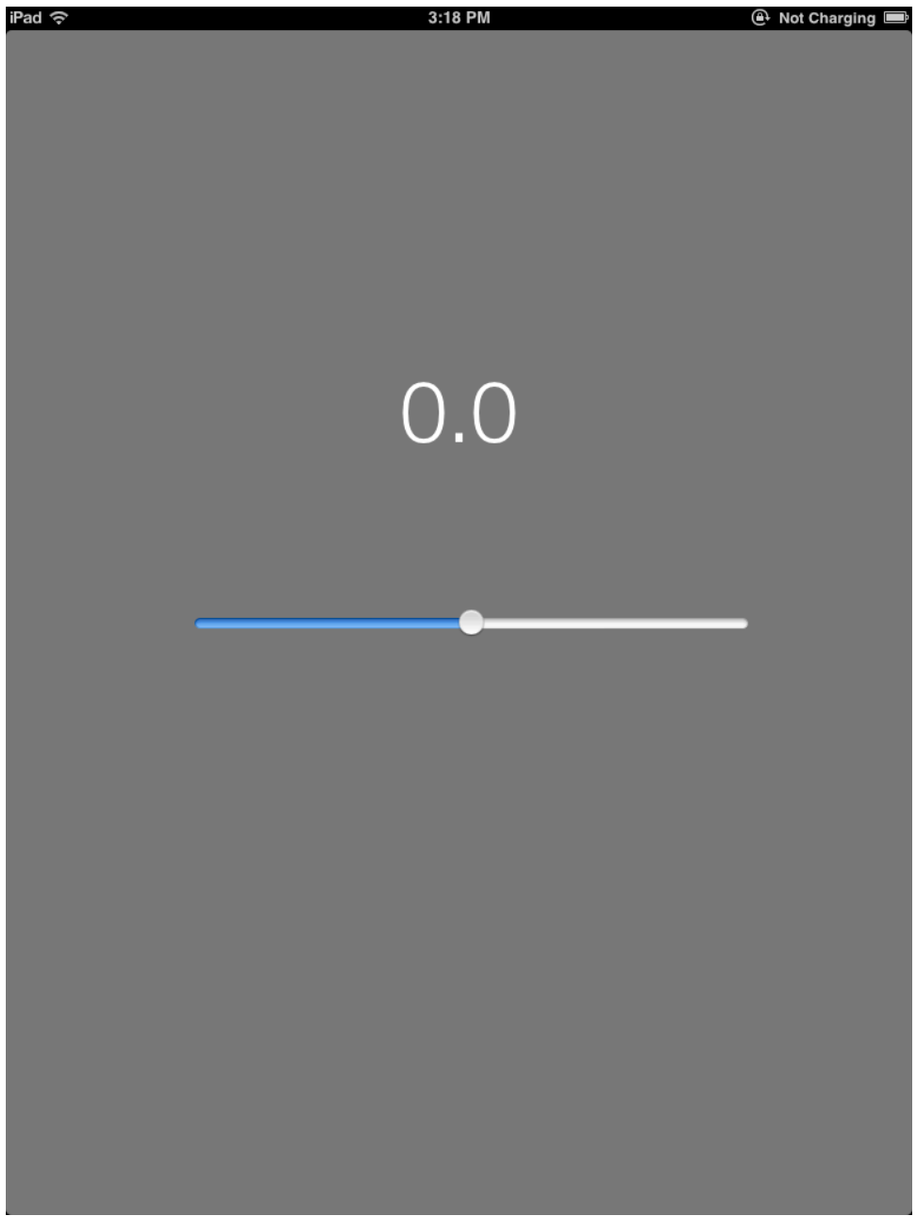
```
@implementation EIViewController

- (void)viewDidLoad {

    self.observer = [[Observer alloc] initWithTarget:self
                                              action:@selector(updateLabel:)];

    [self.counter addObserver:self.observer
                   forKeyPath:@"count"
                      options:NSKeyValueObservingOptionNew
                      context:NULL];
}

-(void) updateLabel:(NSNumber *)newValue {

    self.label.text = [NSString stringWithFormat:@"%.2f", [newValue floatValue]];
}

@end
```

Key-value Observing (KVO)

Any **property** is by default “Key-value Compliant” allowing it to be observed.

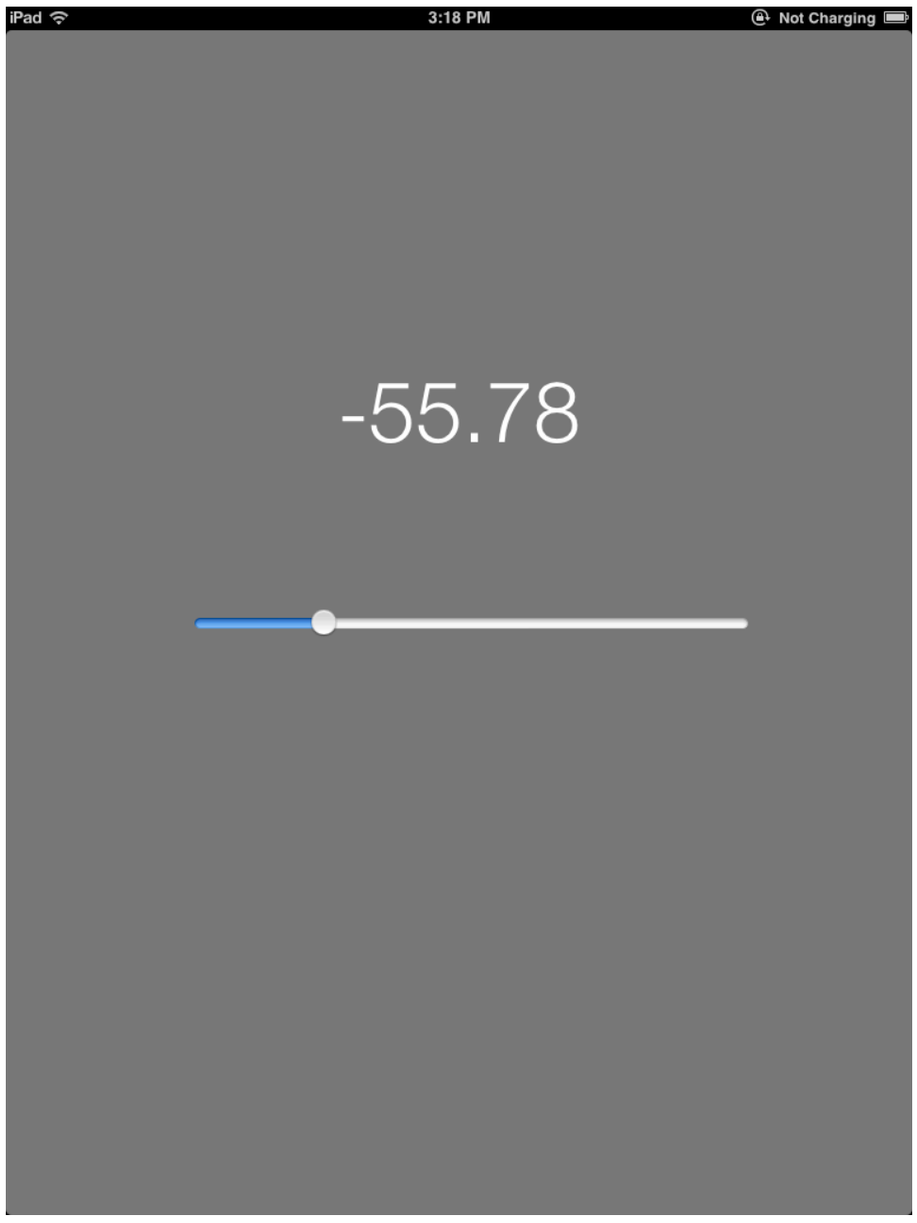
Example: HelloSlider

GitHub: http://bit.ly/XDxIvp



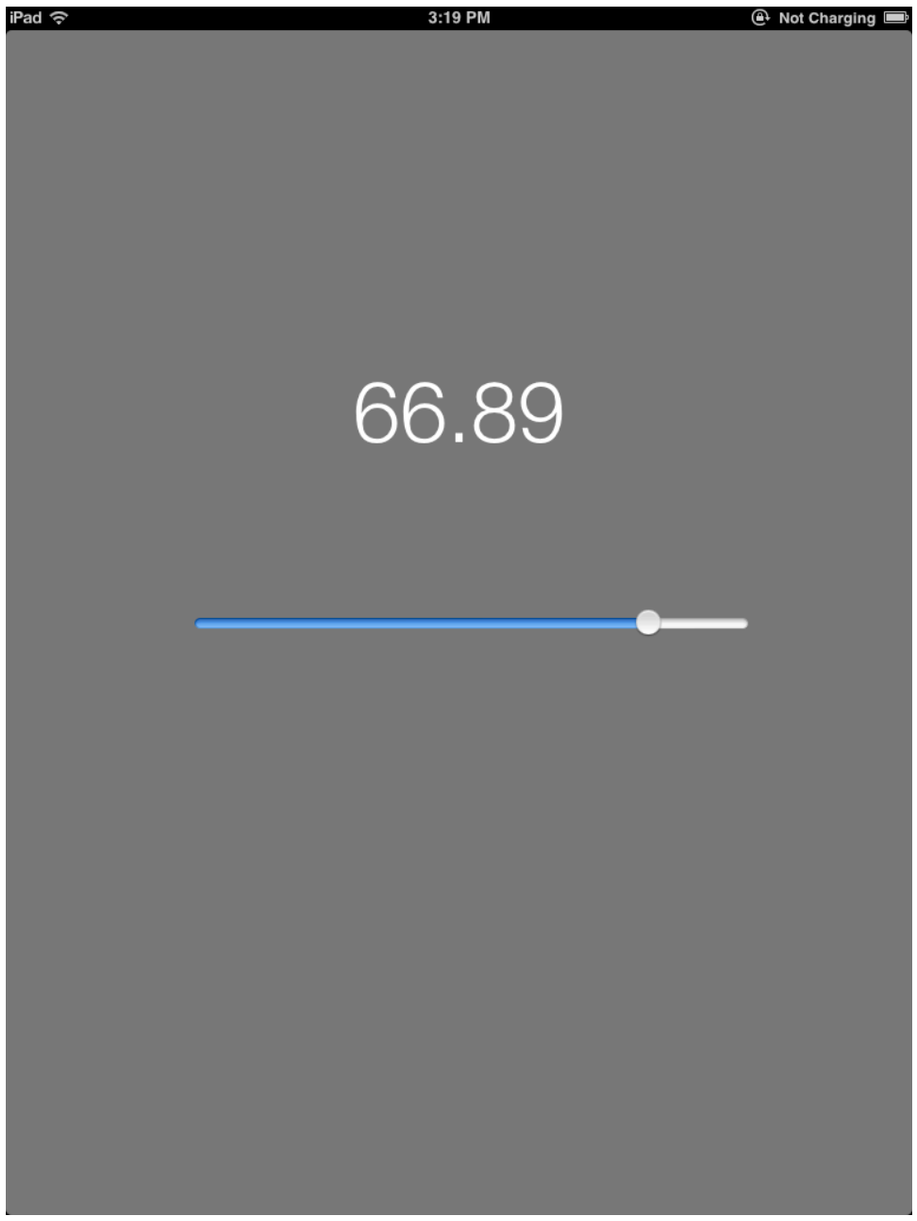
Example: HelloSlider

GitHub: http://bit.ly/XDxIvp



Example: HelloSlider

GitHub: http://bit.ly/XDxIvp



Example: HelloSlider

GitHub: http://bit.ly/XDxIvp

```
@interface Counter : NSObject

@property(nonatomic, strong) NSNumber *count;
- (IBAction)trackSlider:(UISlider *)slider;

@end
```

Example: HelloSlider

GitHub: http://bit.ly/XDxIvp

```
@implementation Counter

-(IBAction) trackSlider:(UISlider *)slider; {
    self.count = [NSNumber numberWithFloat:slider.value];
}

@end
```

Example: HelloSlider

GitHub: http://bit.ly/XDxIvp

```
@interface Observer : NSObject
- (id)initWithTarget:(id)object action:(SEL)action;
@property(nonatomic, strong) id targetObject;
@property(nonatomic) SEL targetAction;
@end
```

Example: HelloSlider

GitHub: http://bit.ly/XDxIvp

```
@implementation Observer

- (void)observeValueForKeyPath:(NSString *)keyPath
                      ofObject:(id)object
                        change:(NSDictionary *)change
                       context:(void *)context {

    [self.targetObject performSelectorOnMainThread:self.targetAction
                                        withObject:[object valueForKeyPath:keyPath]
                                     waitUntilDone:NO];

}

@end
```

Example: HelloSlider

GitHub: http://bit.ly/XDxIvp

```
@implementation EIViewController

- (void)viewDidLoad {

    self.observer = [[Observer alloc] initWithTarget:self
                                              action:@selector(updateLabel:)];

    [self.counter addObserver:self.observer
                   forKeyPath:@"count"
                      options:NSKeyValueObservingOptionNew
                      context:NULL];
}

-(void) updateLabel:(NSNumber *)newValue {

    self.label.text = [NSString stringWithFormat:@"%.2f", [newValue floatValue]];
}

@end
```

Blocks & Dispatch Queues

Blocks & Dispatch Queues

Block

```
^{ NSLog(@"Doing something"); }
```

Dispatch Queue

```
dispatch_queue_t queue = dispatch_get_global_queue(0,0);

dispatch_async(queue, ^{
 NSLog(@"Doing something");
});
```

Blocks & Dispatch Queues

```
- (void)updateFeaturesWithNotification:(NSNotification *)notification {

    dispatch_async([UIApplication sharedIGVAppDelegate].trackControllerQueue, ^{

        [self updateFeatures];

        dispatch_async(dispatch_get_main_queue(), ^{

            [self.coverageTrack setNeedsDisplay];
            [self.track setNeedsDisplay];

        });
    });

}
```

Delegation (Protocol)

Delegation (Protocol)

A protocol is identical to an interface in Java. A collection of method signatures implemented by the object that “conforms” to the protocol.

The delegate/protocol pattern is ubiquitous throughout iOS.

Delegation (Protocol)

```
@interface UITableViewController:UIViewController <UITableViewDelegate, UITableViewDataSource>
```

UITableViewController inherits from UIViewController and conforms to the UITableViewDelegate and UITableViewDataSource protocols

UITableViewDelegate

```
@protocol UITableViewDelegate<NSObject, UIScrollViewDelegate>

@optional
    - (NSIndexPath *)tableView:willSelectRowAtIndexPath:;
    - (NSIndexPath *)tableView:willDeselectRowAtIndexPath:;

@end
```

UITableViewDataSource

```
@protocol UITableViewDataSource<NSObject>

@required
- (NSInteger)tableView:numberOfRowsInSection:;

@optional
- (NSInteger)numberOfSectionsInTableView:;
- (NSArray *)sectionIndexTitlesForTableView:;

@end
```

iOS devices combine mobility, gesture, and a powerful GPU. This makes iOS App development an entirely new form of software development.

Mobility introduces context as a driver for usage.
New app types appear to meet user needs driven by context.

Gesture forces a “no interface”, approach to app design and U/X mindset.

Data representation drives interaction and associated affordances.

GPU. That cute little iPhone in your hand is a graphics processing beast.

It is a GPU device tamed for domestic use.
The entire interface is GPU driven.

That is why iOS apps feel the way they do.
Light. Effortless. Friction free. Like butter.

Developers must discard many of their desktop assumptions when developing for iOS

- No mouse
- No interface
- Minimal keyboard
- Arms length interaction
- One handed Interaction
- Two handed Interaction
- Untethered resources

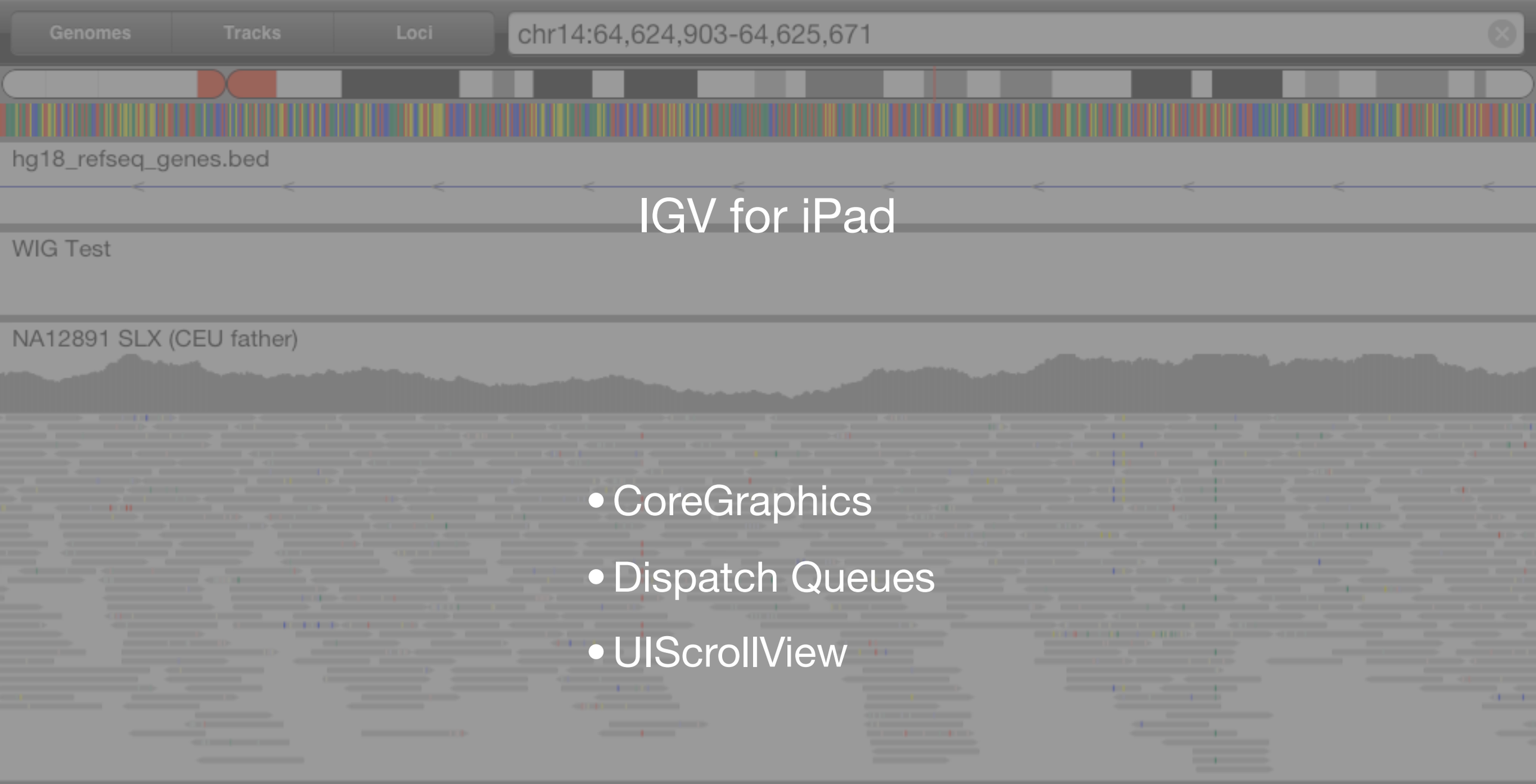
Demos & Code

Multi-resolution Image

- CATiledLayer
- UIScrollView
- Amazon Web Services (S3)

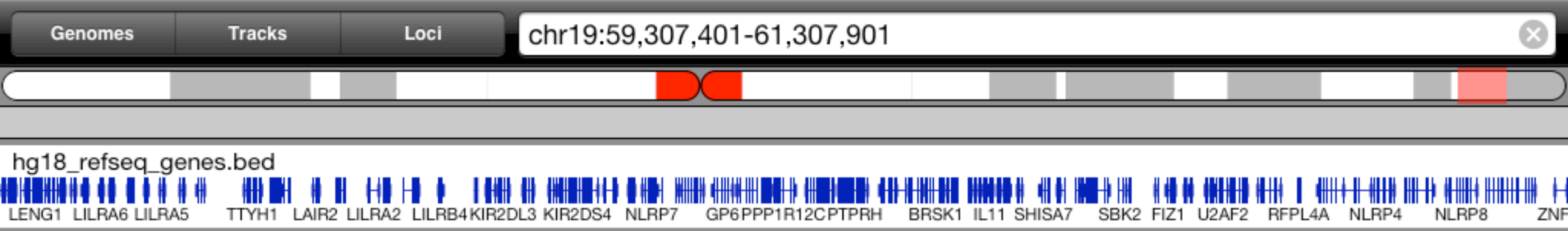
Multi-resolution Image

- 8k x 8k image resolution. 101MB on disk.
- Subsampled 4 successive times
- Diced into 256 x 256 tiles
- Stored on Amazon S3



IGV for iPad

- CoreGraphics
- Dispatch Queues
- UIScrollView


Genomes
Tracks
Loci
chr19:59,307,401-61,307,901
hg18_refseq_genes.bed
LENG1 LILRA6 LILRA5 TTYH1 LAIR2 LILRA2 LILRB4 KIR2DL3 KIR2DS4 NLRP7 GP6 PPP1R12C PTPRH BRSK1 IL11 SHISA7 SBK2 FIZ1 U2AF2 RFPL4A NLRP4 NLRP8 ZNF

Genomes | Tracks | Loci

chr14:64,624,903-64,625,671

hg18_refseq_genes.bed

WIG Test

NA12891 SLX (CEU father)

Genomes | Tracks | Loci

chr19:59,307,401-61,307,901

PTPRH BRSK1 IL11 SHISA7 SBK2 FIZ1 U2AF2 RFPL4A NLRP4 NLRP8 ZNF

My Tracks

Loaded Tracks

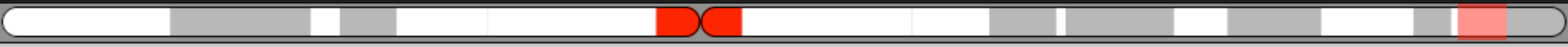
Annotations (hg18)

Breast Cancer (METABRIC)

The Cancer Genome Atlas

Genomes | Tracks | Loci

chr19:59,307,401-61,307,901



hg18_refseq_genes.bed

WIG Random Data Test

Genomes | Tracks | Loci

chr12:97,514,589-97,518,248

hg18_refseq_genes.bed

SLC25A3

NA12891 SLX (CEU father)

Genomes | Tracks | Loci

chr12:97,516,735-97,517,649

hg18_refseq_genes.bed

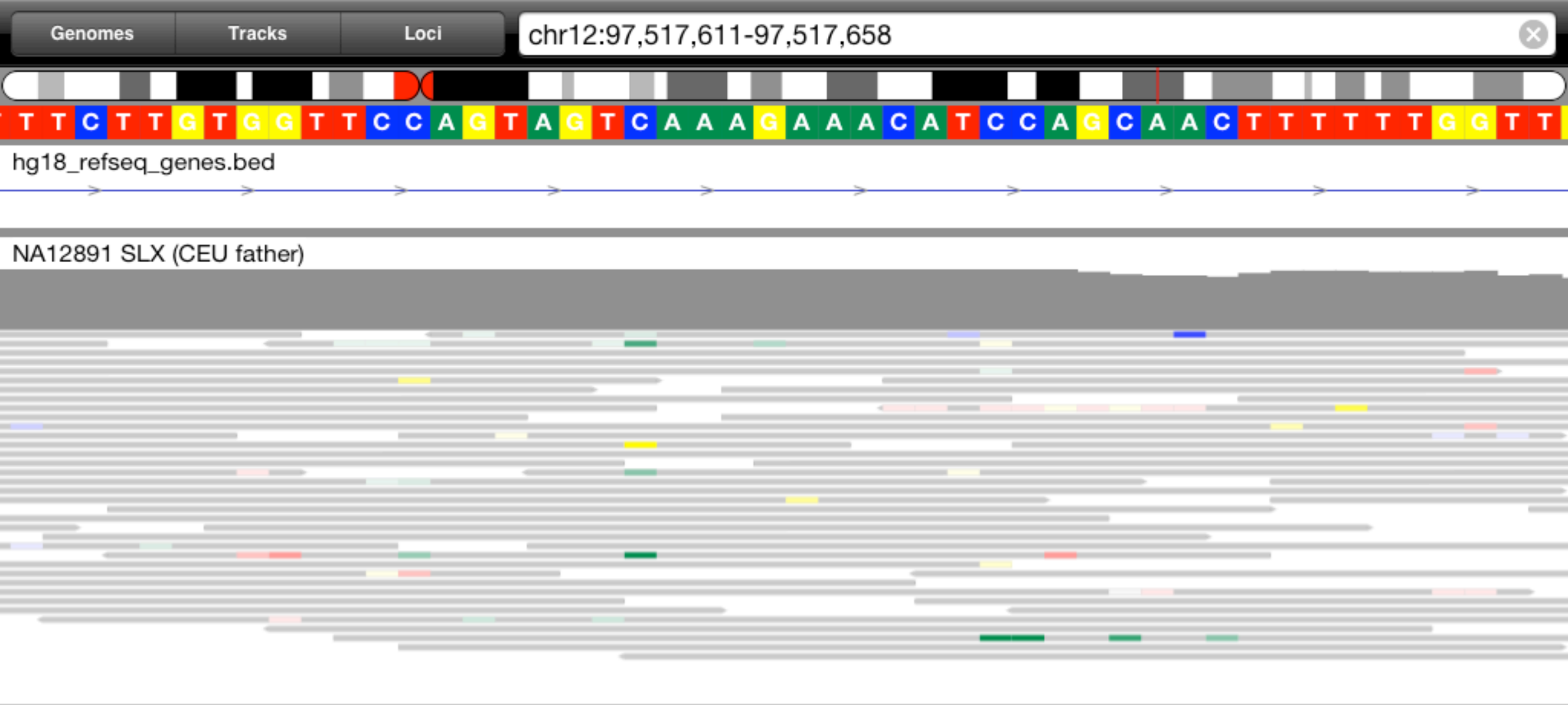
NA12891 SLX (CEU father)

Genomes | Tracks | Loci

chr12:97,517,559-97,517,672

hg18_refseq_genes.bed

NA12891 SLX (CEU father)


Genomes
Tracks
Loci
chr12:97,517,611-97,517,658
T T C T T G T G G T T C C A G T A G T C A A A G A A A C A T C C A G C A A C T T T T T T G G T T
hg18_refseq_genes.bed
NA12891 SLX (CEU father)

Genomes | Tracks | Loci

chr12:97,517,624-97,517,672

A G T A G T C A A A G A A A C A T C C A G C A A C T T T T T T G G T T G T A T A G T C A A A G G

hg18_refseq_genes.bed

NA12891 SLX (CEU father)

Genomes | Tracks | Loci

ad

hg18_refseq_genes.bed

NA12891 SLX (CEU father)

- **ADD1** chr4:2,815,381-2,901,600
- **ADAMTS3** chr4:73,365,549-73,653,380
- **ADAR** chr1:152,821,157-152,867,080
- **ADCY7** chr16:48,879,323-48,909,544
- **ADAMTS5** chr21:27,212,101-27,261,310
- **ADH7** chr4:100,552,440-100,575,690
- **ADI1** chr2:3,480,696-3,502,354
- **ADAM10** chr15:56,675,801-56,829,469
- **ADAMTS7**

Genomes | Tracks | Loci

chr12:97,517,624-97,517,672

- hg18
- hg19
- mm9

Genomes | Tracks | Loci

ad

hg18_refseq_genes.bed

NA12891 SLX (CEU father)

Delete Track: NA12891 SLX (CEU father)

Hide Track Labels

Squish

Genomes | Tracks | Loci

ad

hg18_refseq_genes.bed

The Elastic Image

- OpenGL
- GLSL - OpenGL Shading Language
- Shader/Gesture plug-ability via plist
- UISplitViewController

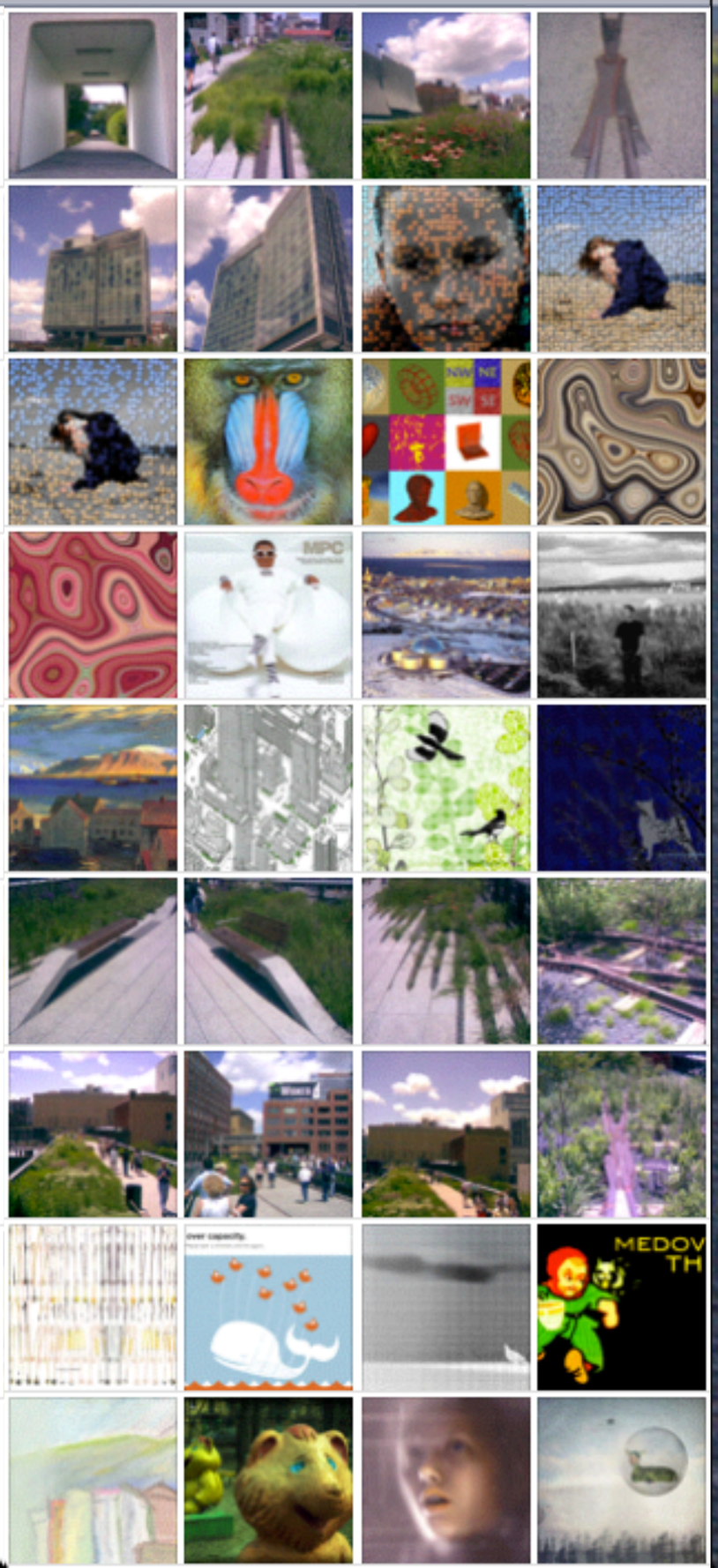
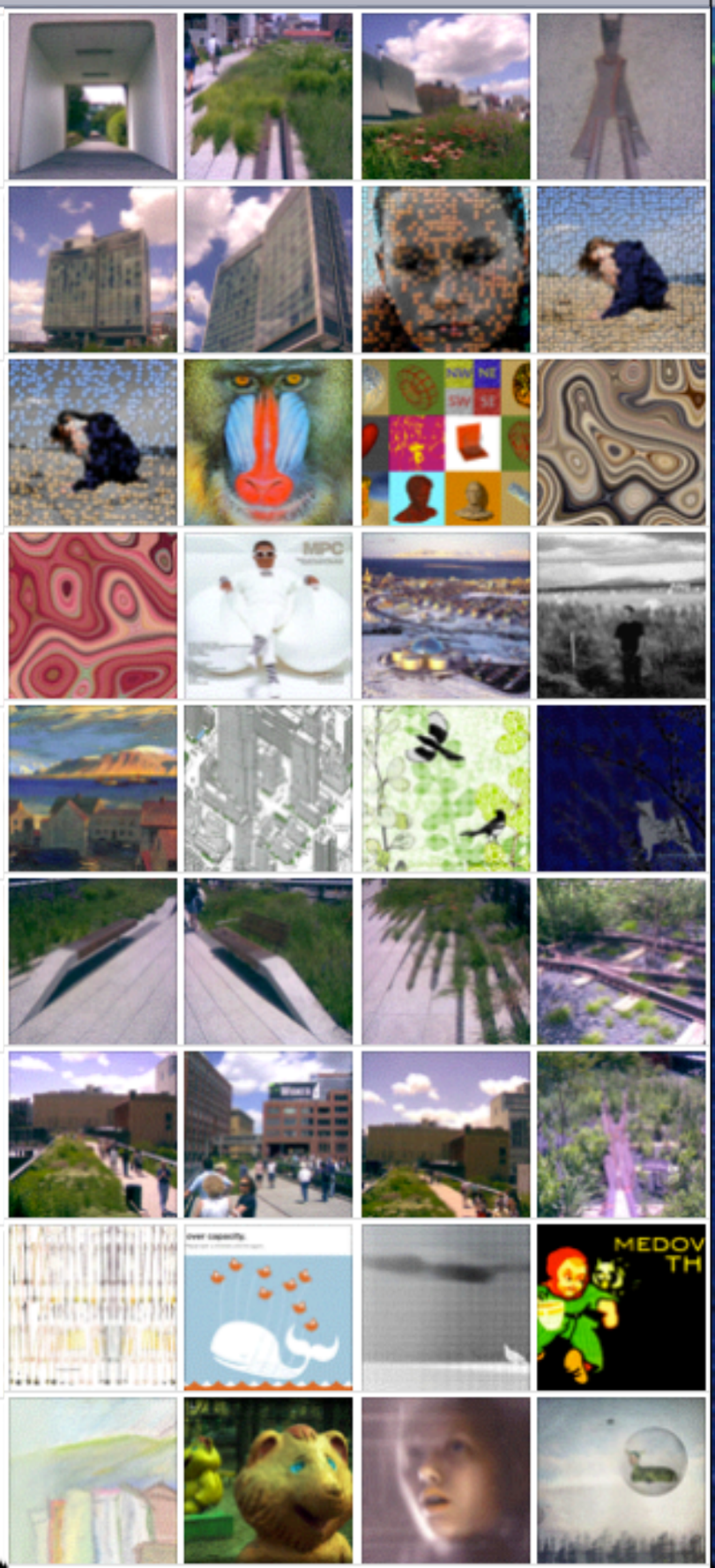
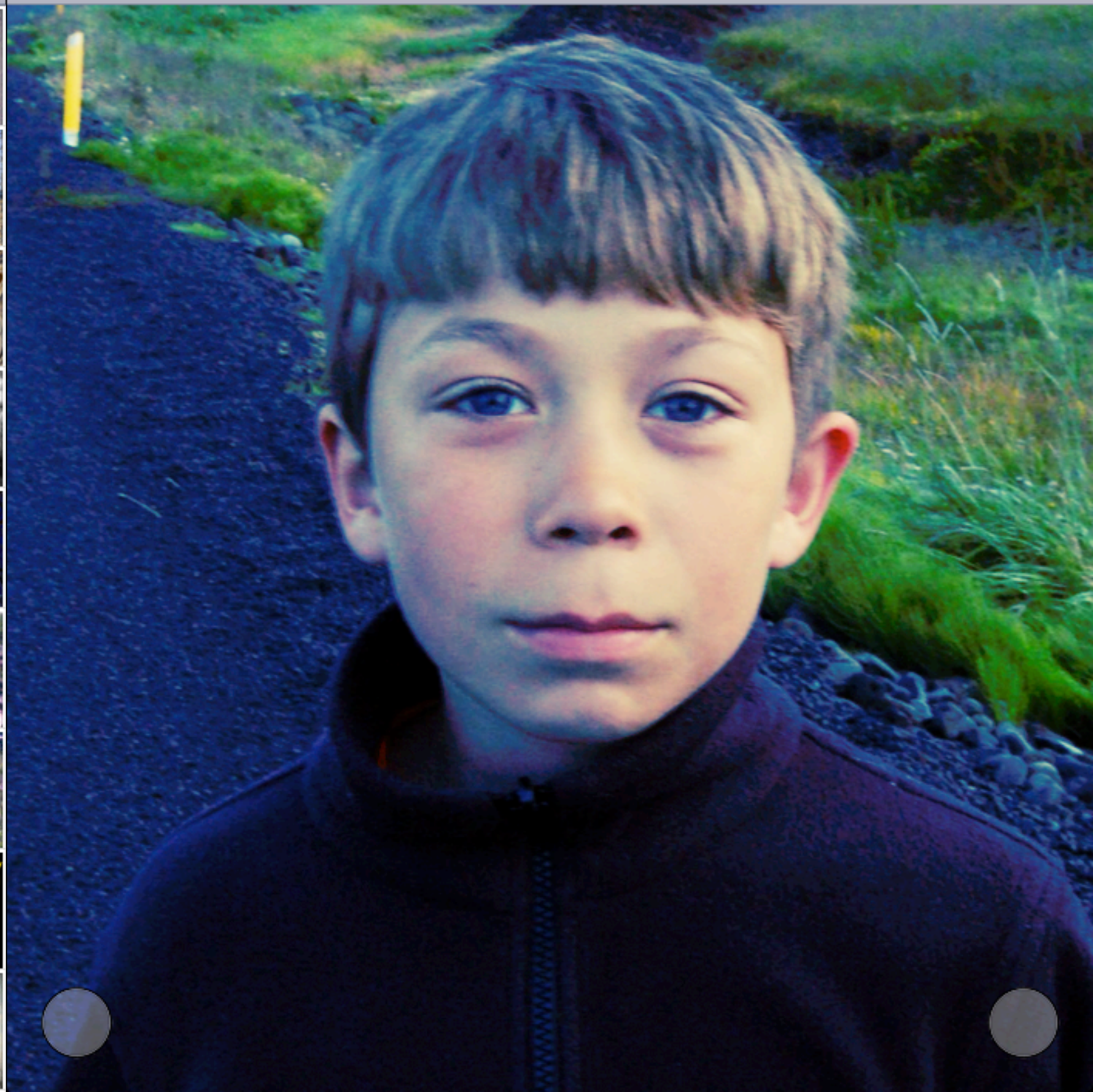


Photo Albums

Untitled

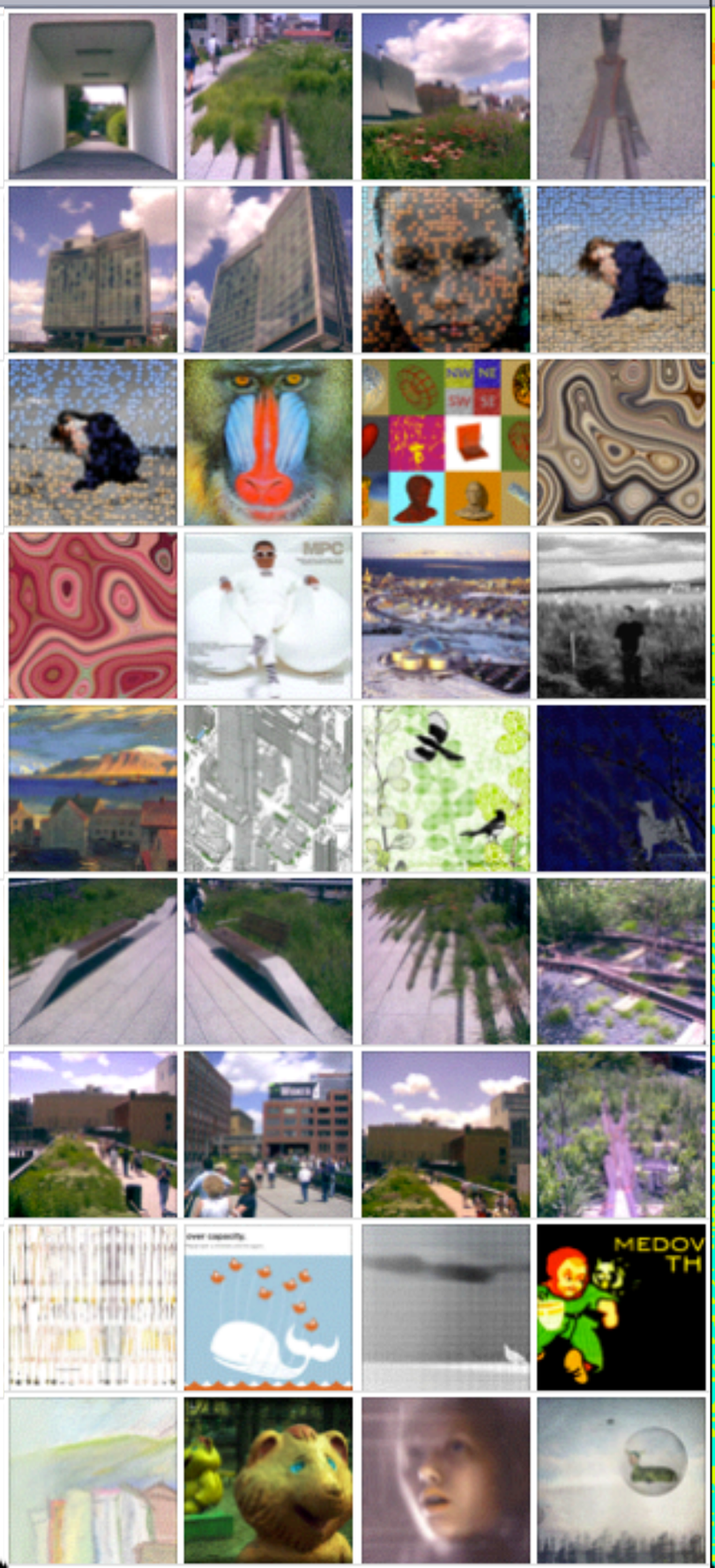
ColorGradingShader





Untitled

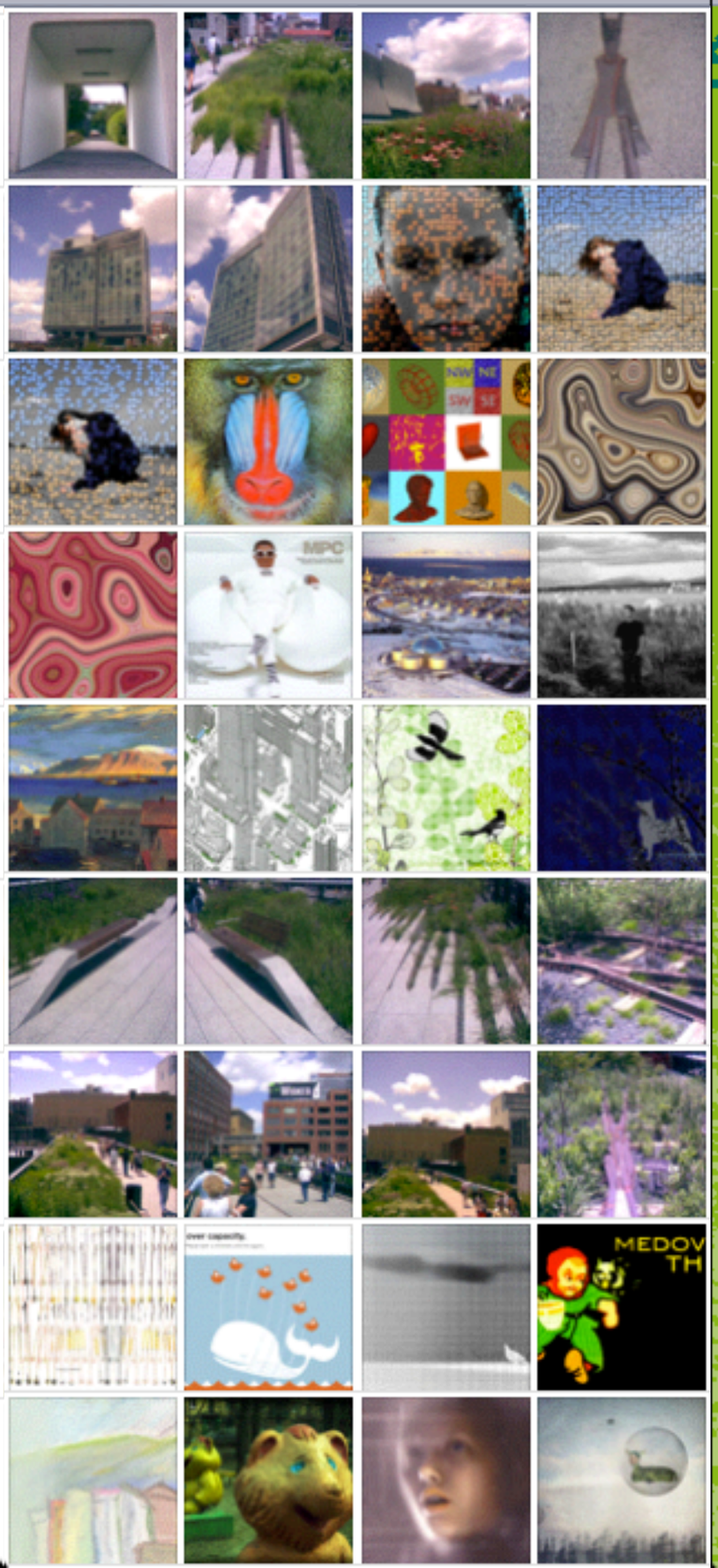
ColorRampShader



iPad 3:45 PM Not Charging

Photo Albums Untitled

ColorRampShader



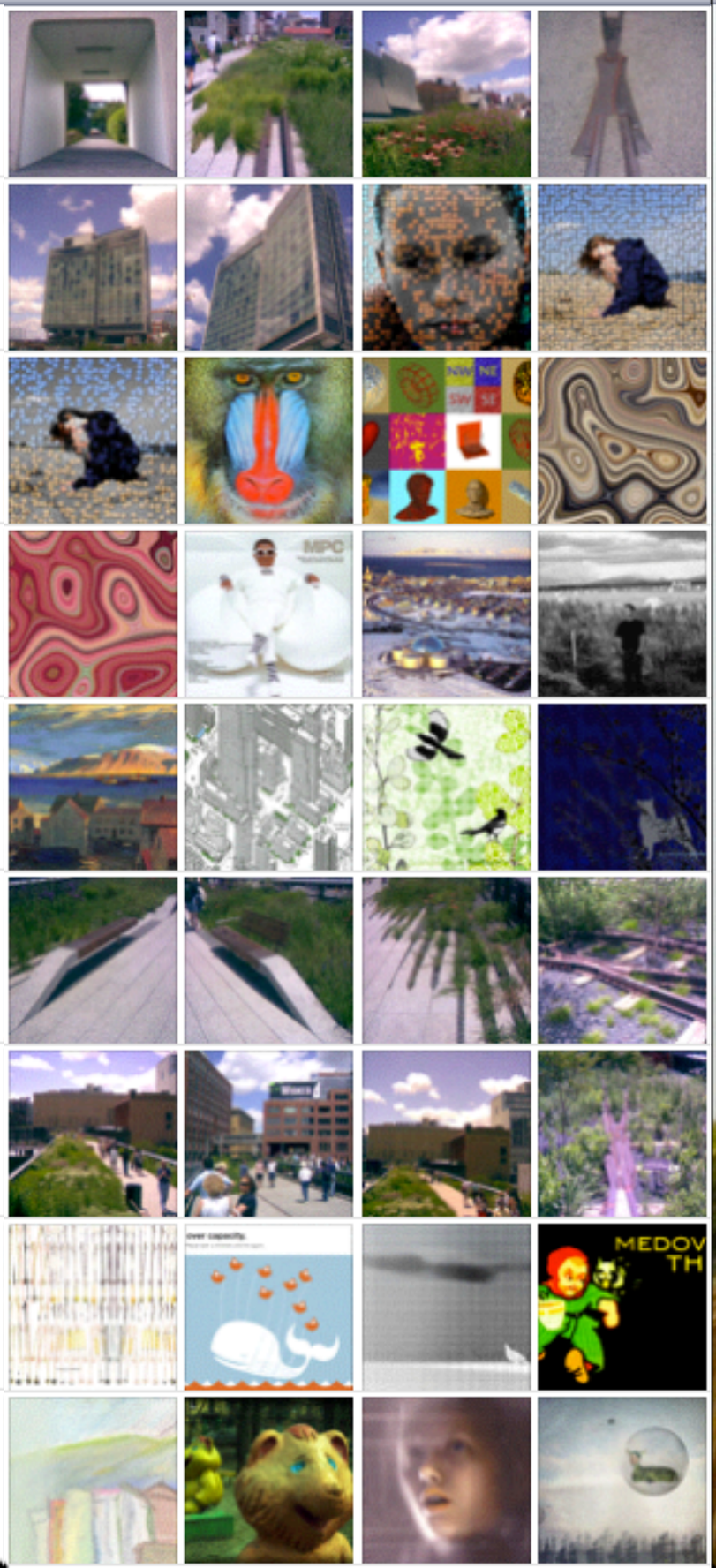
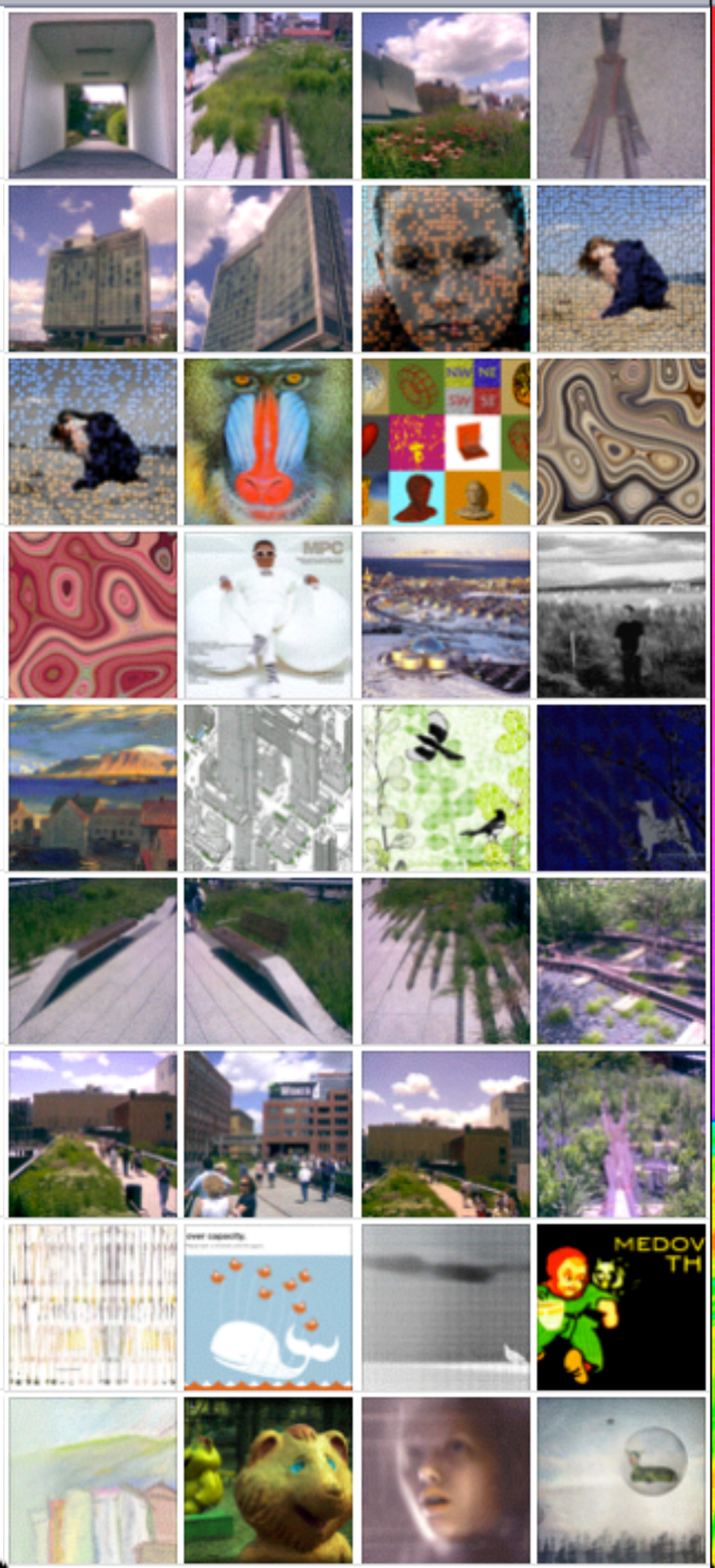
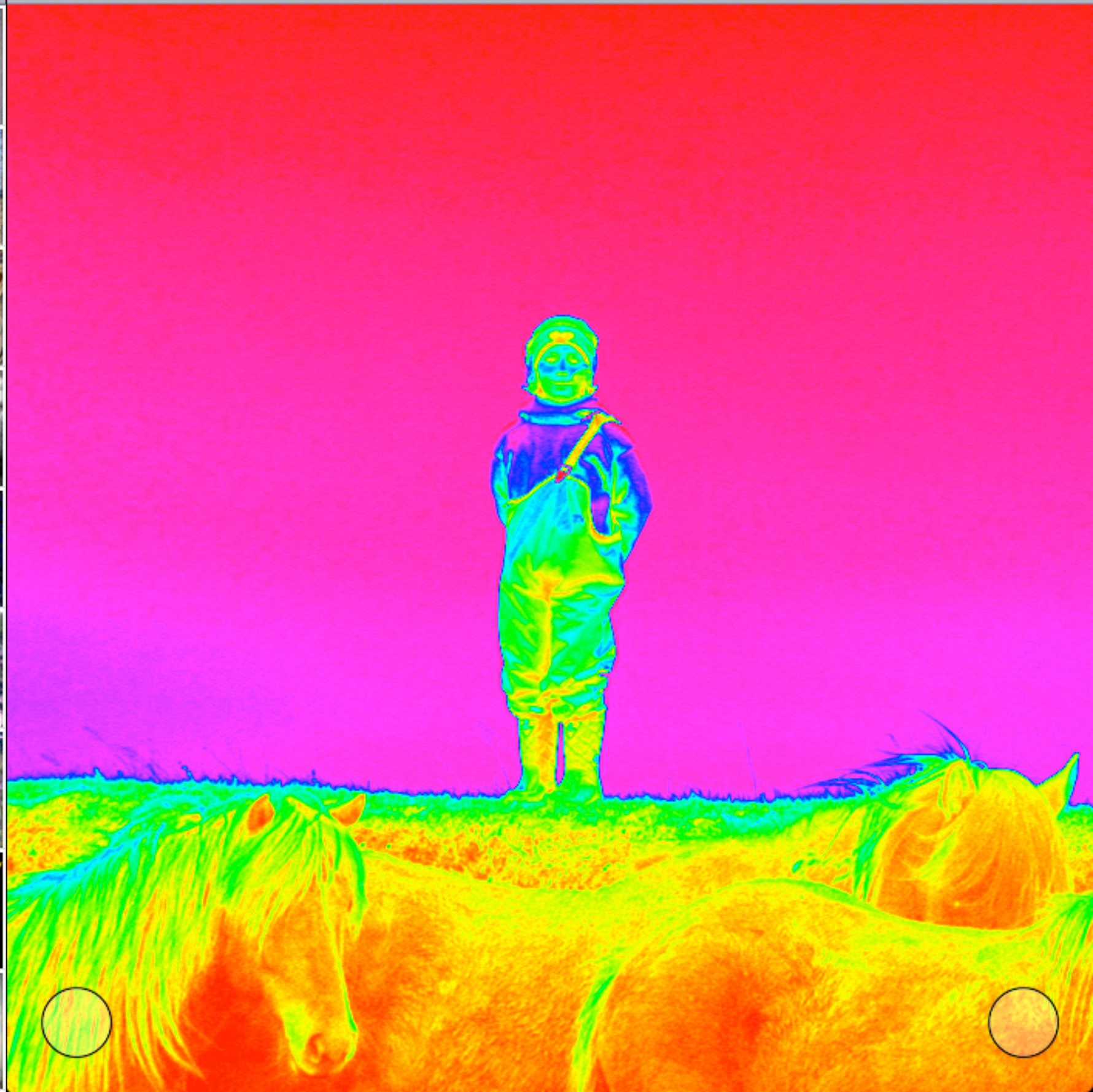


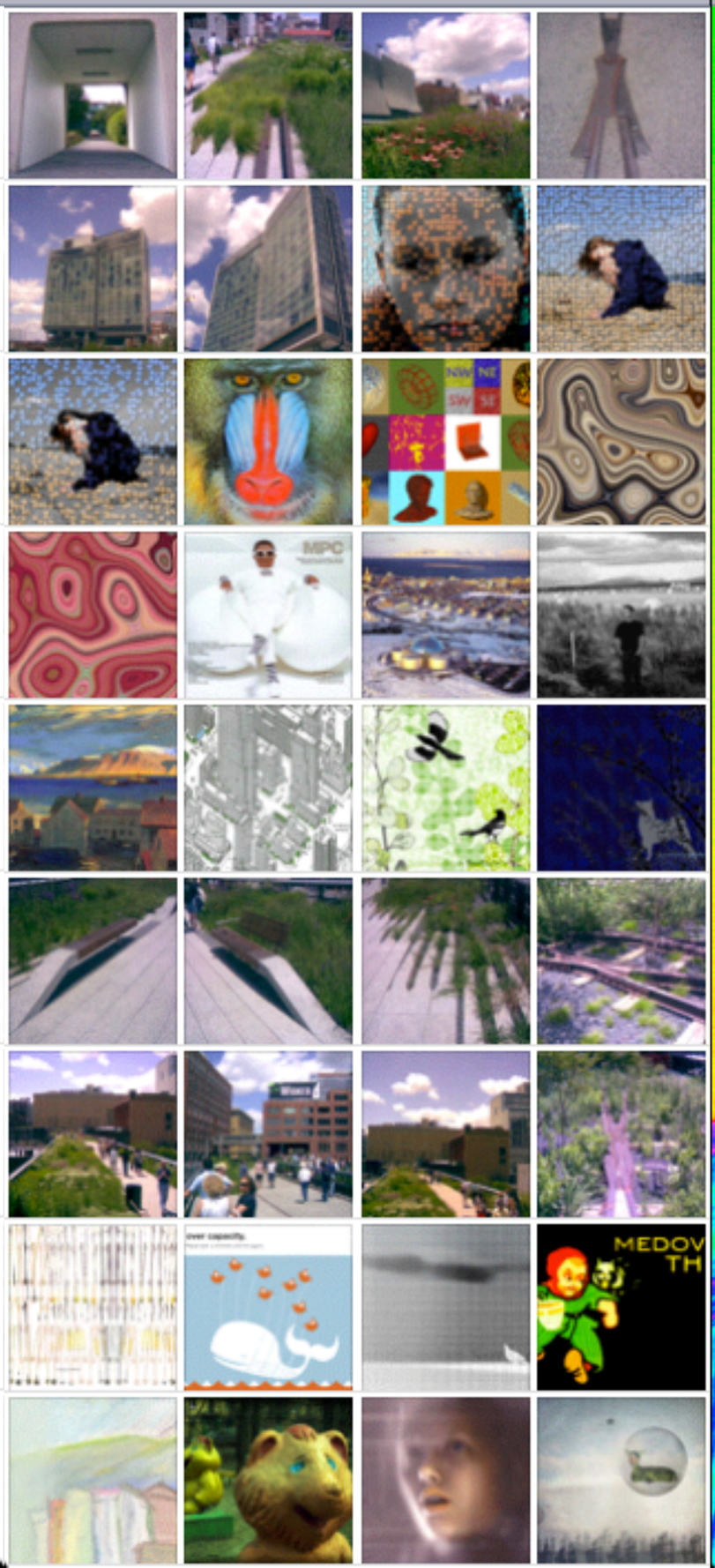
Photo Albums

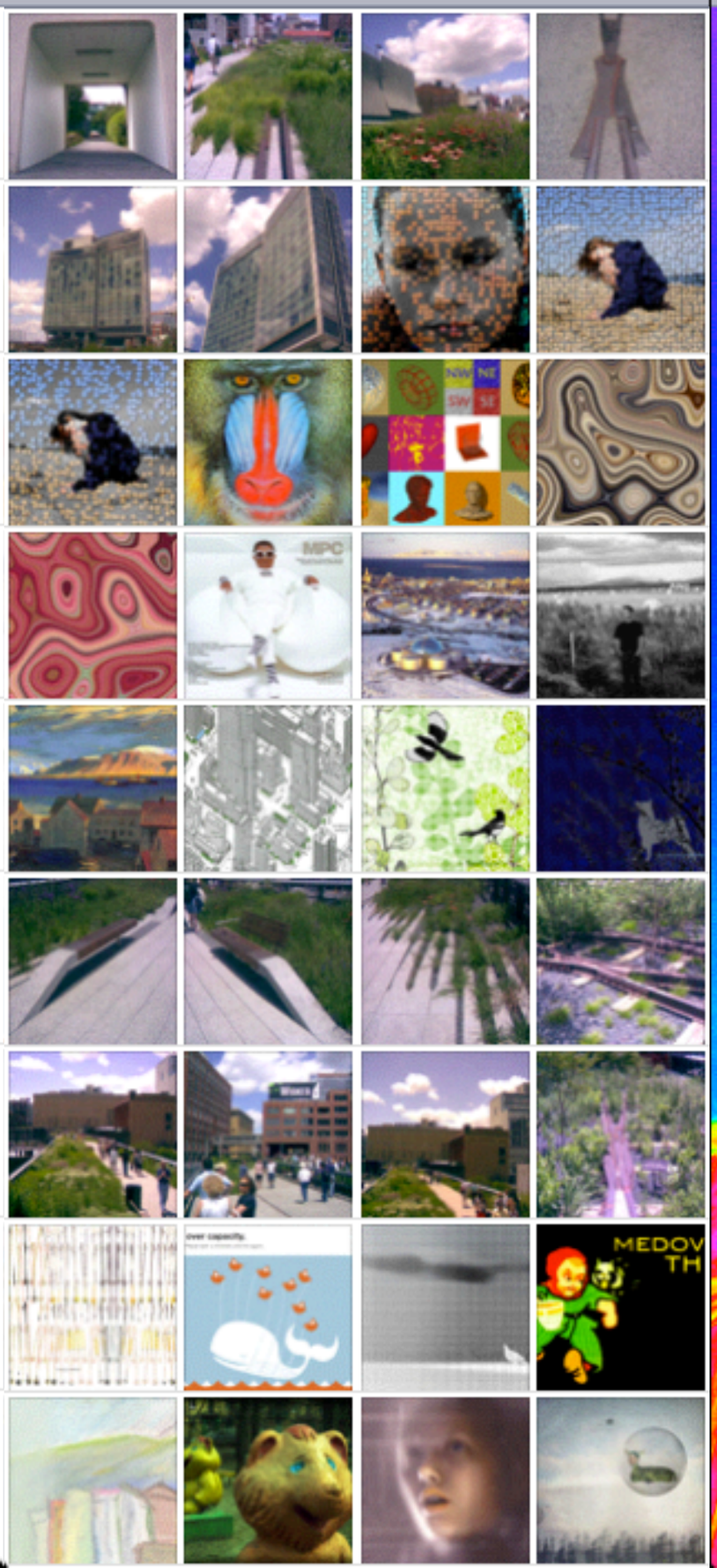
Untitled

HueShiftShader









HueShiftShader

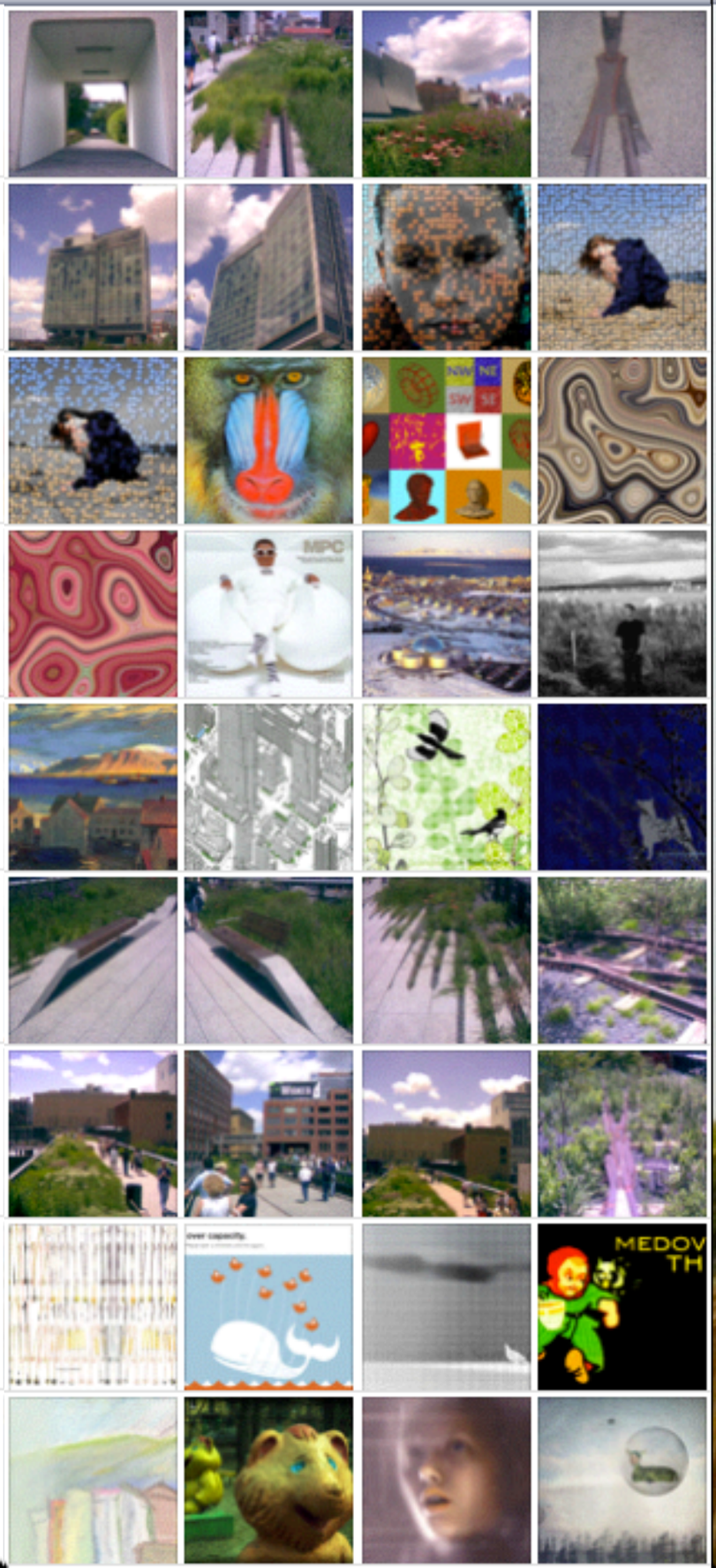
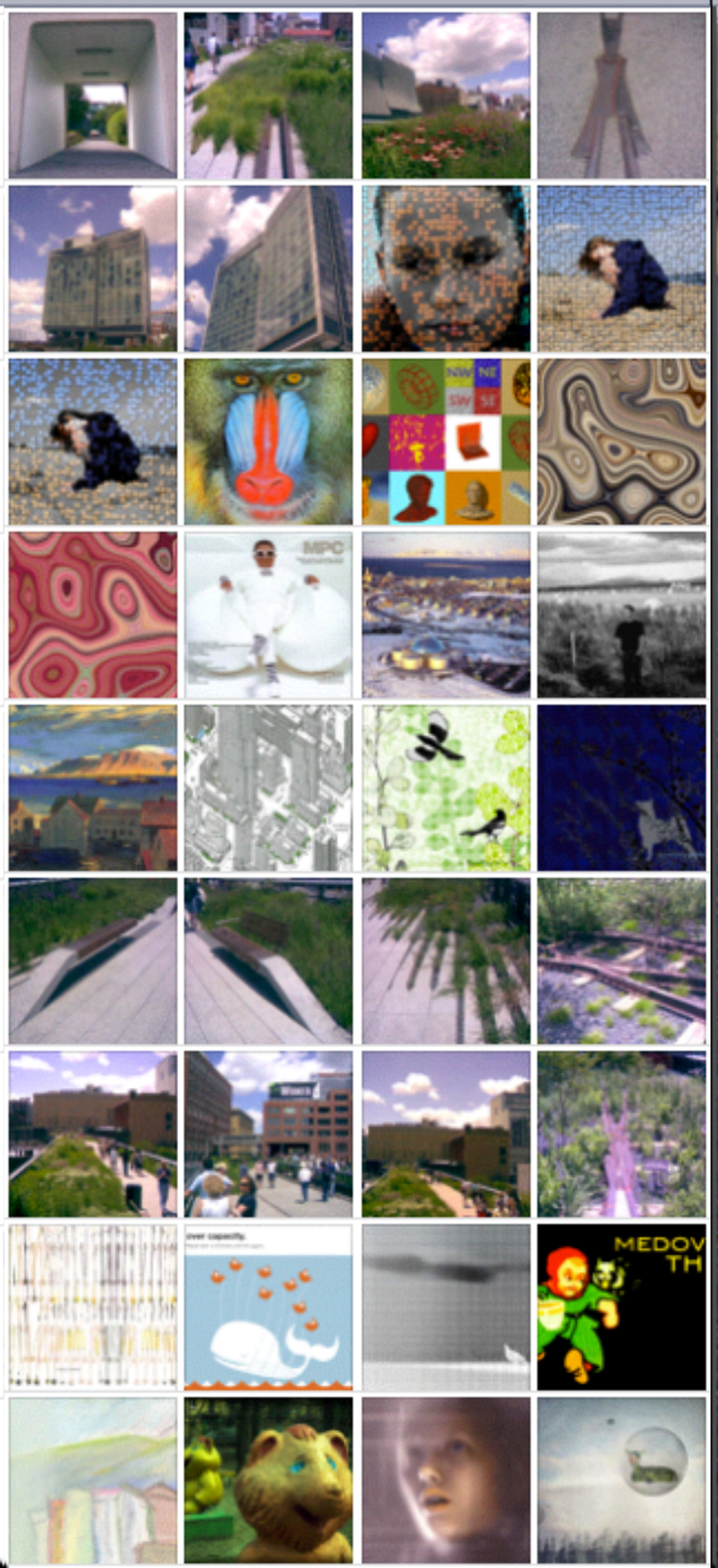


Photo Albums

Untitled

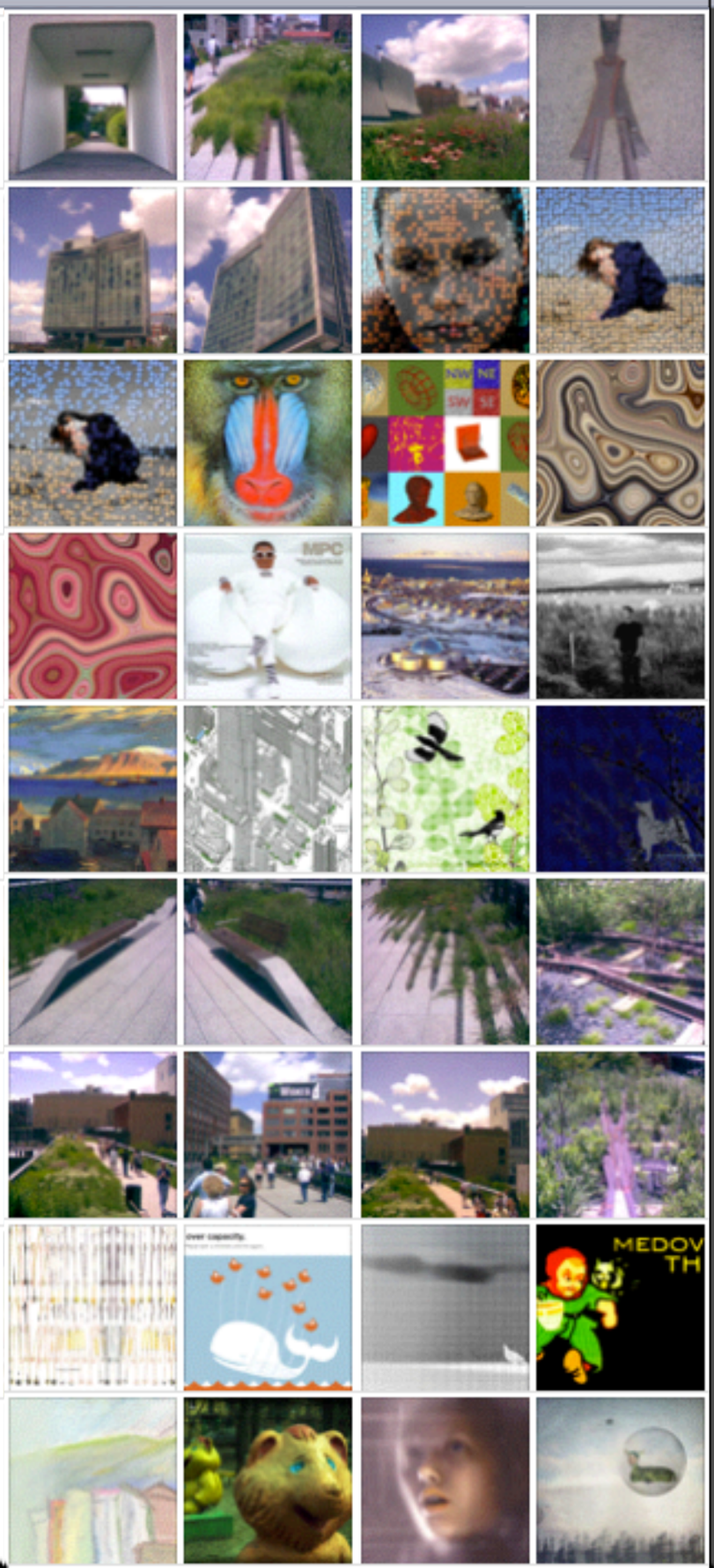
InvertColor

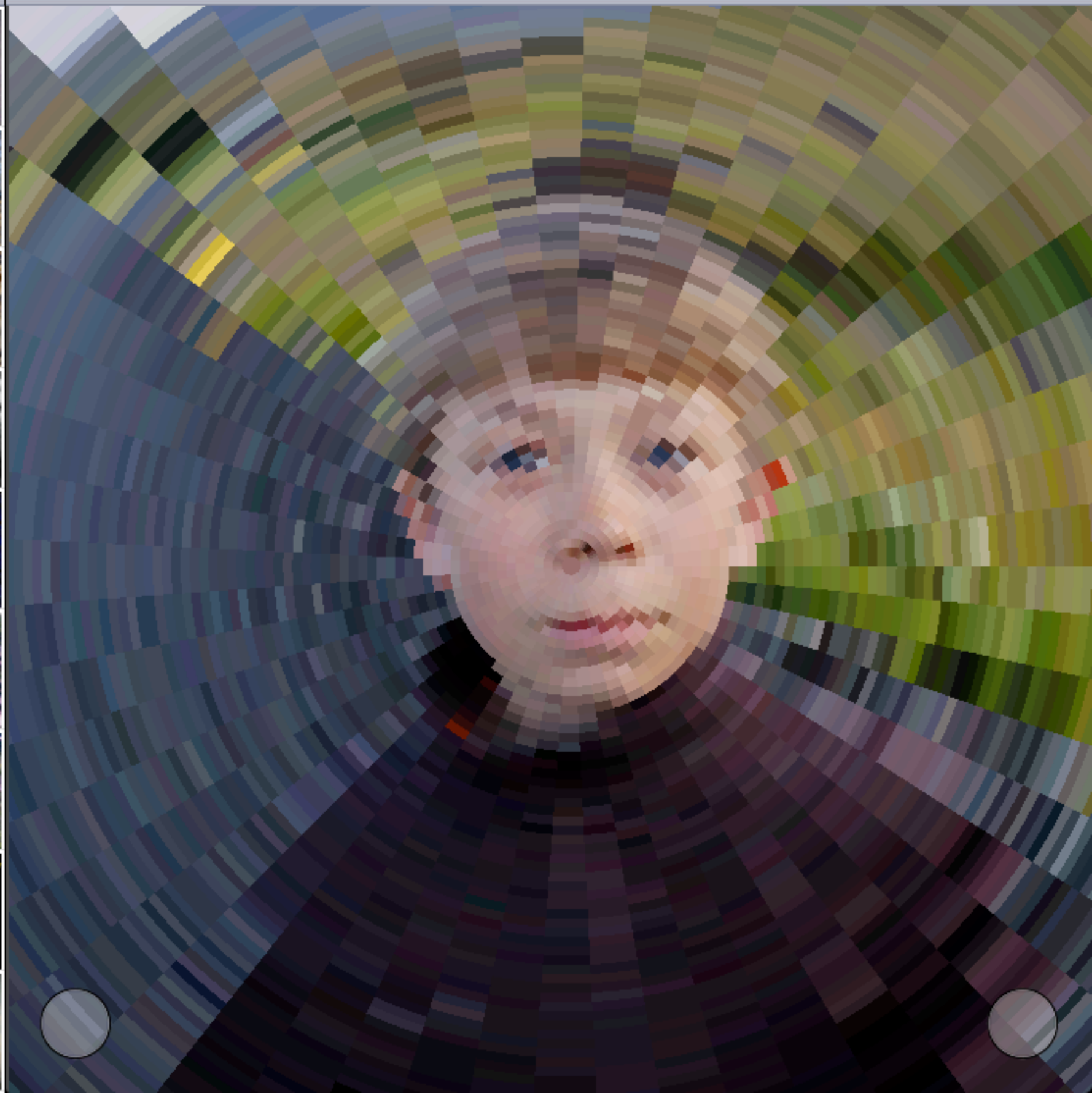


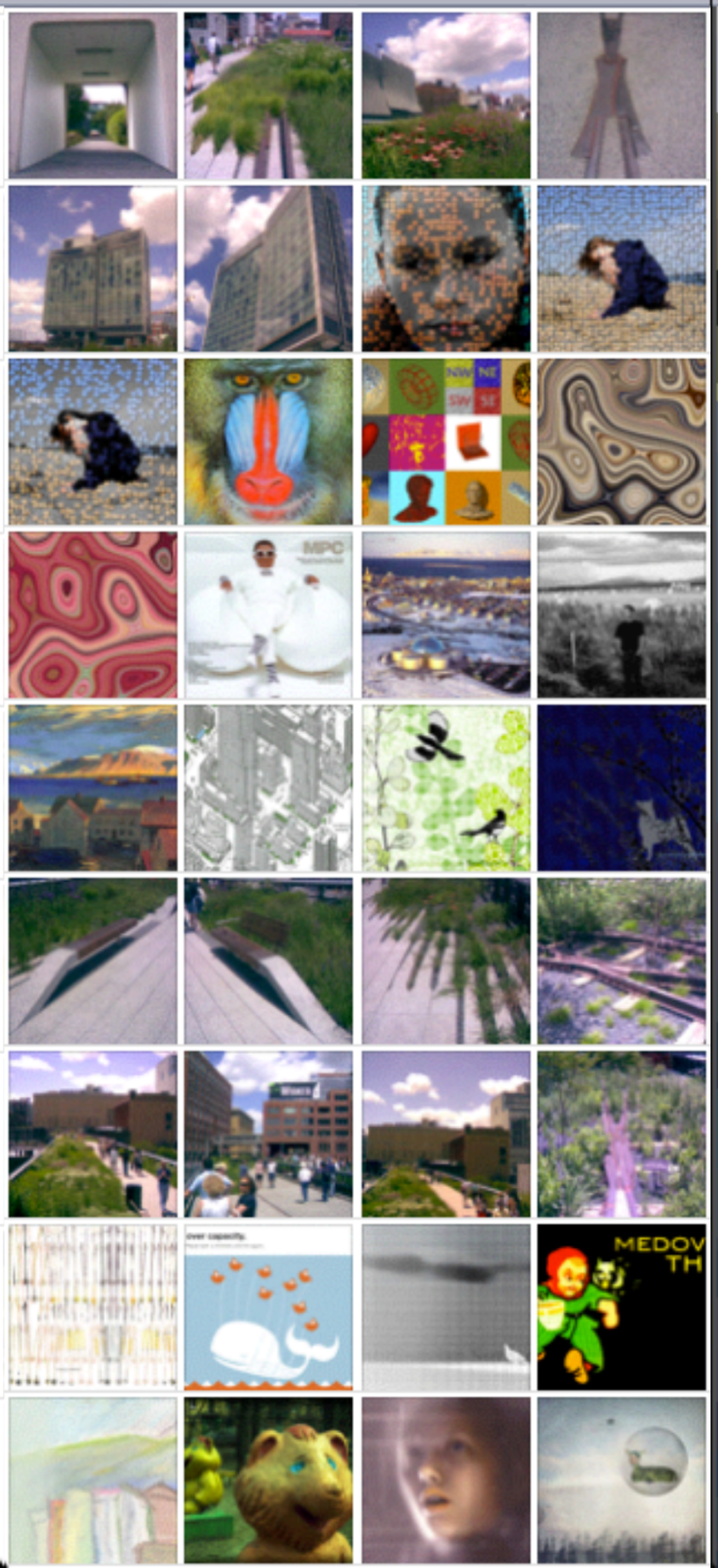
iPad 3:48 PM Not Charging

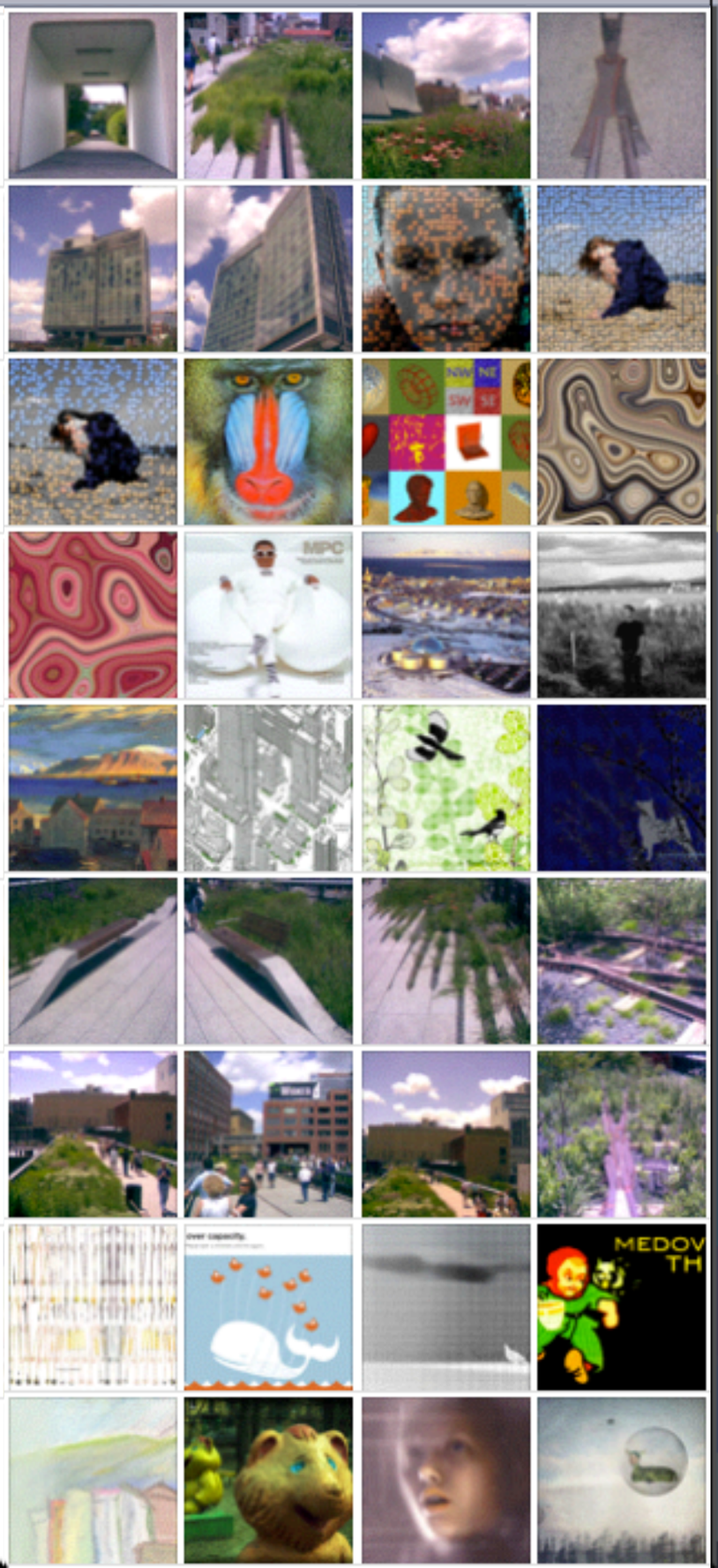
Photo Albums Untitled

QuantizeST









QuantizeST

The Elastic Image

Gestures are fundamental to iOS apps. A gesture is attached to a UIView. Gestures come in different flavors.

Pan

```
self.panGesture = [[UIPanGestureRecognizer alloc] initWithTarget:self
                                                          action:@selector(handlePanGesture:)];
[self addGestureRecognizer:self.panGesture];
```

Pinch

```
self.scaleGesture = [[UIPinchGestureRecognizer alloc] initWithTarget:self
                                                              action:@selector(handleScaleGesture:)];
[self addGestureRecognizer:self. scaleGesture];
```

Tap

```
self.toggleGesture = [[UITapGestureRecognizer alloc] initWithTarget:self
                                                            action:@selector(handleToggleGesture:)];
self.toggleGesture.numberOfTapsRequired     = 1;
self.toggleGesture.numberOfTouchesRequired  = 1;
[self addGestureRecognizer:self.toggleGesture];
```

The Elastic Image

Photo f/x are implemented in GLSL. The OpenGL Shading Language. Shaders are written in a C-like language and evaluated in a SIMD manner on the entire image in realtime.

The Elastic Image

TextureMapShader - apply a texture (the photo) to a quad (the rendering surface).

```
varying vec2 v_st;

uniform sampler2D hero;

void main() {

    gl_FragColor =
(heroChannels == 1) ? vec4(texture2D(hero, v_st).a) : texture2D(hero, v_st);

}
```

```
varying vec2 v_st;

uniform sampler2D hero;

void main() {

    gl_FragColor =
    (heroChannels == 1) ? vec4(texture2D(hero, v_st).a) : texture2D(hero, v_st);

}
```

The Elastic Image

Property lists enable simple specification of a shader gesture and its handler.

The Objective-C runtimes enables easy conversion from string to class instance

Key	Class	Value
▼Root	Dictionary	⬍ 1 key/value pairs
▼shaderLibrary	Dictionary	⬍ 9 key/value pairs
▼ColorGradingShader	Dictionary	⬍ 4 key/value pairs
▼uniforms	Dictionary	⬍ 5 key/value pairs
▼gestures	Array	⬍ 1 ordered objects
▼0	Dictionary	⬍ 2 key/value pairs
class	String	⬍ UIPanGestureRecognizer
selector	String	⬍ colorGradingShaderGestureHandler:
▶hero	Dictionary	⬍ 2 key/value pairs
▶lut	Dictionary	⬍ 2 key/value pairs
▶mixer	Dictionary	⬍ 3 key/value pairs
▶projectionViewModelMatrix	Dictionary	⬍ 2 key/value pairs
fragment	String	⬍ EISColorGradingShader
vertex	String	⬍ EISColorGradingShader
▶vertexAttributes	Dictionary	⬍ 2 key/value pairs

The Elastic Image

Property lists enable simple specification of a shader gesture and its handler.

The Objective-C runtimes enables easy conversion from string to class instance

```
- (UIGestureRecognizer *)createGestureWithDictionary:(NSDictionary *)gestureDictionary {

    NSString *gestureClassName = [gestureDictionary objectForKey:@"class"];
    Class _gesture = NSClassFromString(gestureClassName);

    NSString *selectorName = [gestureDictionary objectForKey:@"selector"];
    SEL  _selector = NSSelectorFromString(selectorName);

    UIGestureRecognizer *shaderGesture =
    [[[_gesture alloc] initWithTarget:self action:_selector] autorelease];

    shaderGesture.delegate = self.detailController;

    return shaderGesture;
}
```

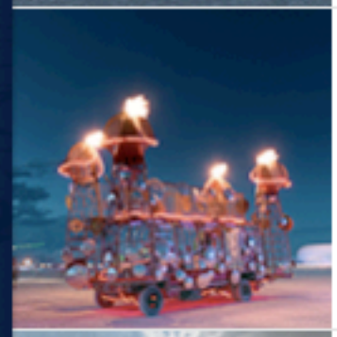
Beautiful Panoramas

- OpenGL
- GLSL - OpenGL Shading Language
- Proprietary Panorama Engine
- UIPopoverController

Sheikh Zayed Mosque Exterior, Abu Dhabi
Sheikh Zayed Mosque is one of the largest mosques in the world. The four corner minarets measure 377 feet tall and the mosque can house up to 40,000 worshippers.

Sheikh Zayed Mosque Interior, Abu Dhabi
Sheikh Zayed Mosque features the world's largest carpet made by more than 1,200 people and weighing over 51 tons. This main room alone accommodates 9,000 people.

Black Rock City, Nevada
Each year, people from around the world gather to "burn the man" deep in the Nevada Desert. Black Rock City is the temporary city that springs up to house these people who try to leave no trace on the desert landscape when they leave.



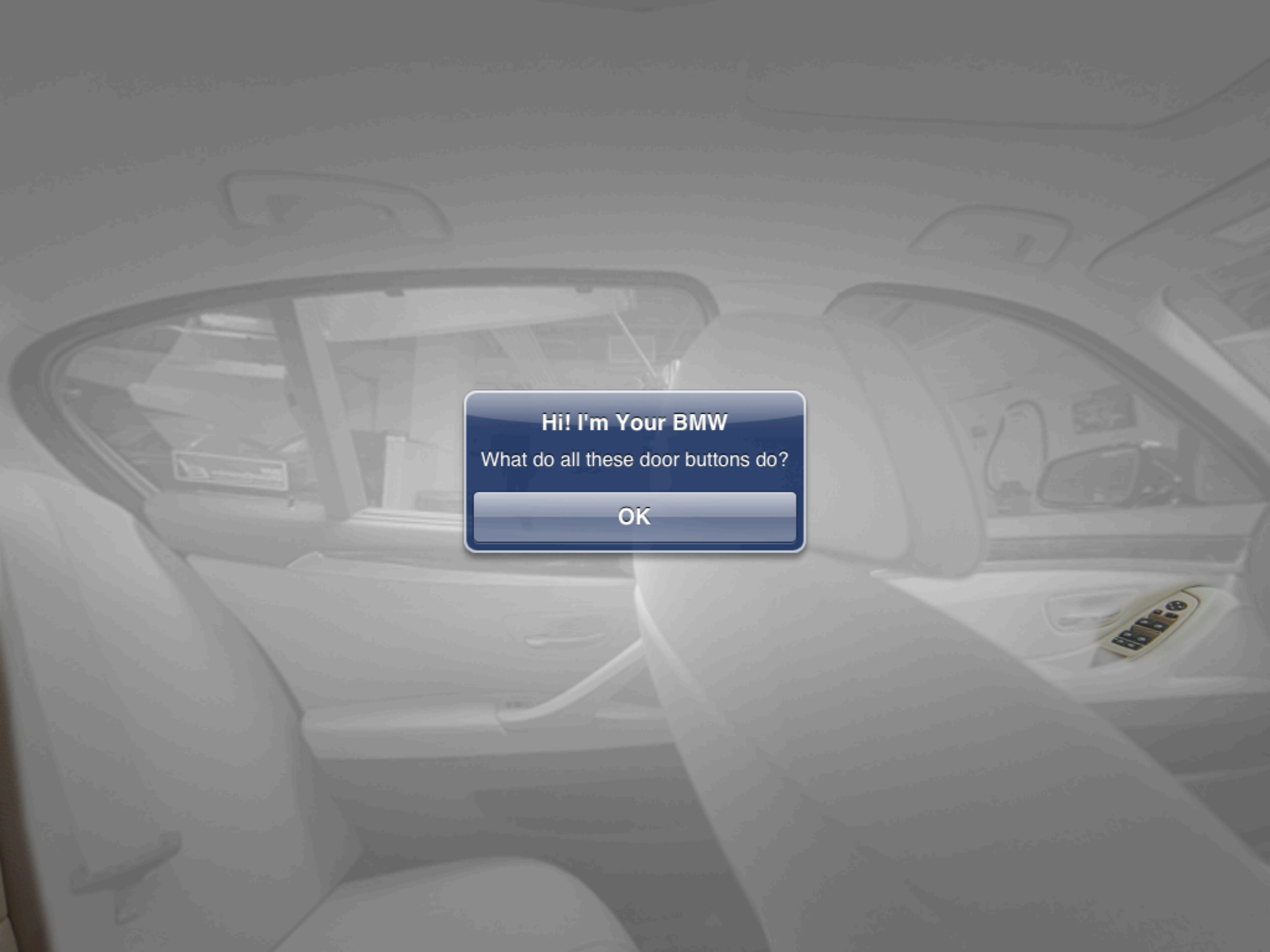
Burning Man Festival Vehicles
Some of the fantastical vehicles that made it to the Burning Man Festival in 2007.

Derelict Caravan, California
An abandoned caravan in the California desert that has been set on fire and used for target practice. Notice the bullet holes and gradual disintegration.

BMW Interior Panorama

- OpenGL
- GLSL - OpenGL Shading Language
- Proprietary Panorama Engine
- 3D Spatial Picking

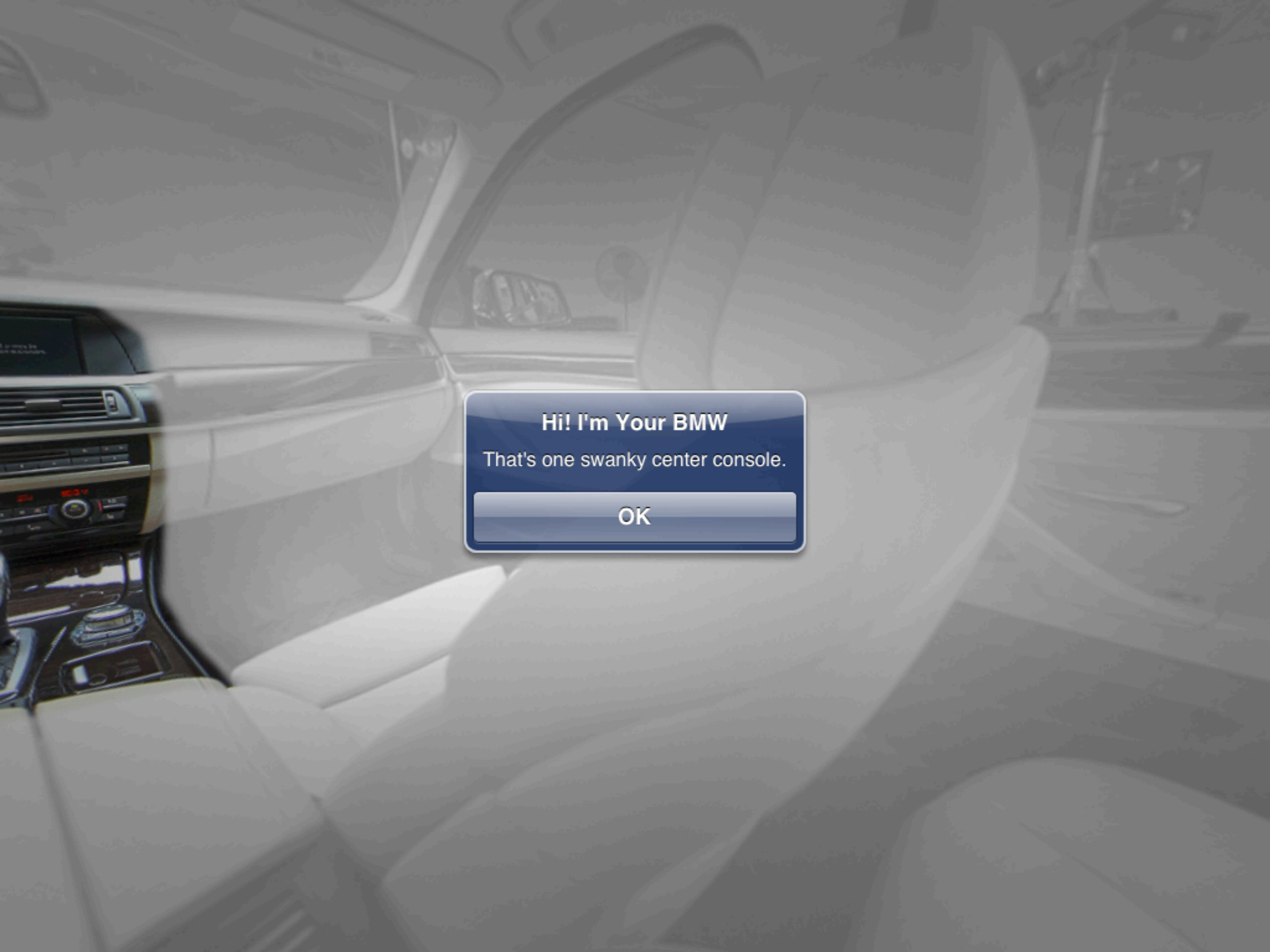

Hi! I'm Your BMW
What do all these door buttons do?
OK

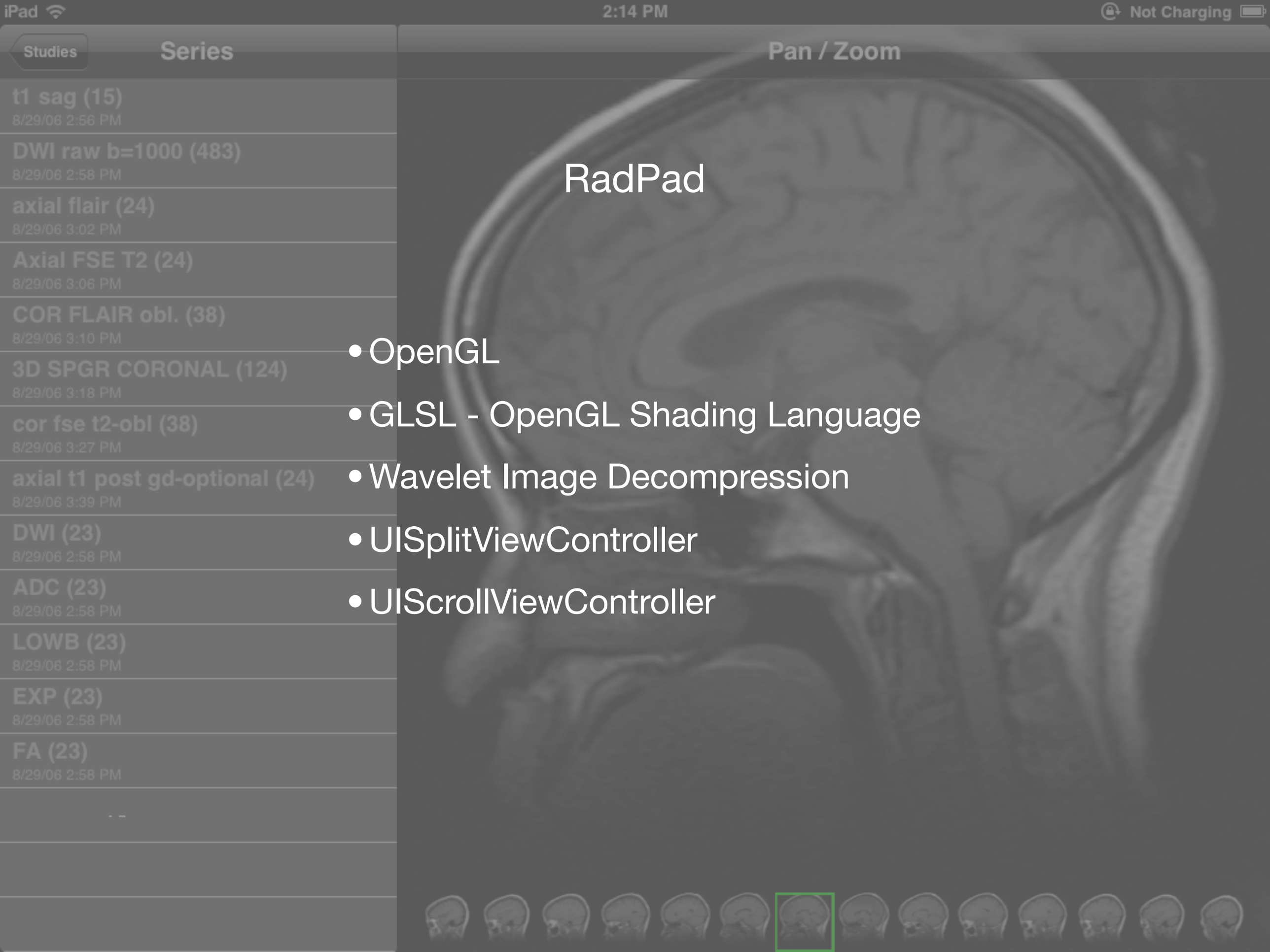

Hi! I'm Your BMW
What do all these door buttons do?
OK

Hi! I'm Your BMW

Mmmm, leather seats. Nice.

OK


Hi! I'm Your BMW
That's one swanky center console.
OK



RadPad

- OpenGL
- GLSL - OpenGL Shading Language
- Wavelet Image Decompression
- UISplitViewController
- UIScrollViewController

Studies

Series

- t1 sag (15) 8/29/06 2:56 PM
- DWI raw b=1000 (483) 8/29/06 2:58 PM
- axial flair (24) 8/29/06 3:02 PM
- Axial FSE T2 (24) 8/29/06 3:06 PM
- COR FLAIR obl. (38) 8/29/06 3:10 PM
- 3D SPGR CORONAL (124) 8/29/06 3:18 PM
- cor fse t2-obl (38) 8/29/06 3:27 PM
- axial t1 post gd-optional (24) 8/29/06 3:39 PM
- DWI (23) 8/29/06 2:58 PM
- ADC (23) 8/29/06 2:58 PM
- LOWB (23) 8/29/06 2:58 PM
- EXP (23) 8/29/06 2:58 PM
- FA (23) 8/29/06 2:58 PM

Pan / Zoom

Thank You

Douglass Turner
Elastic Image Software
tweets: @dugla
email: douglass.turner@gmail.com