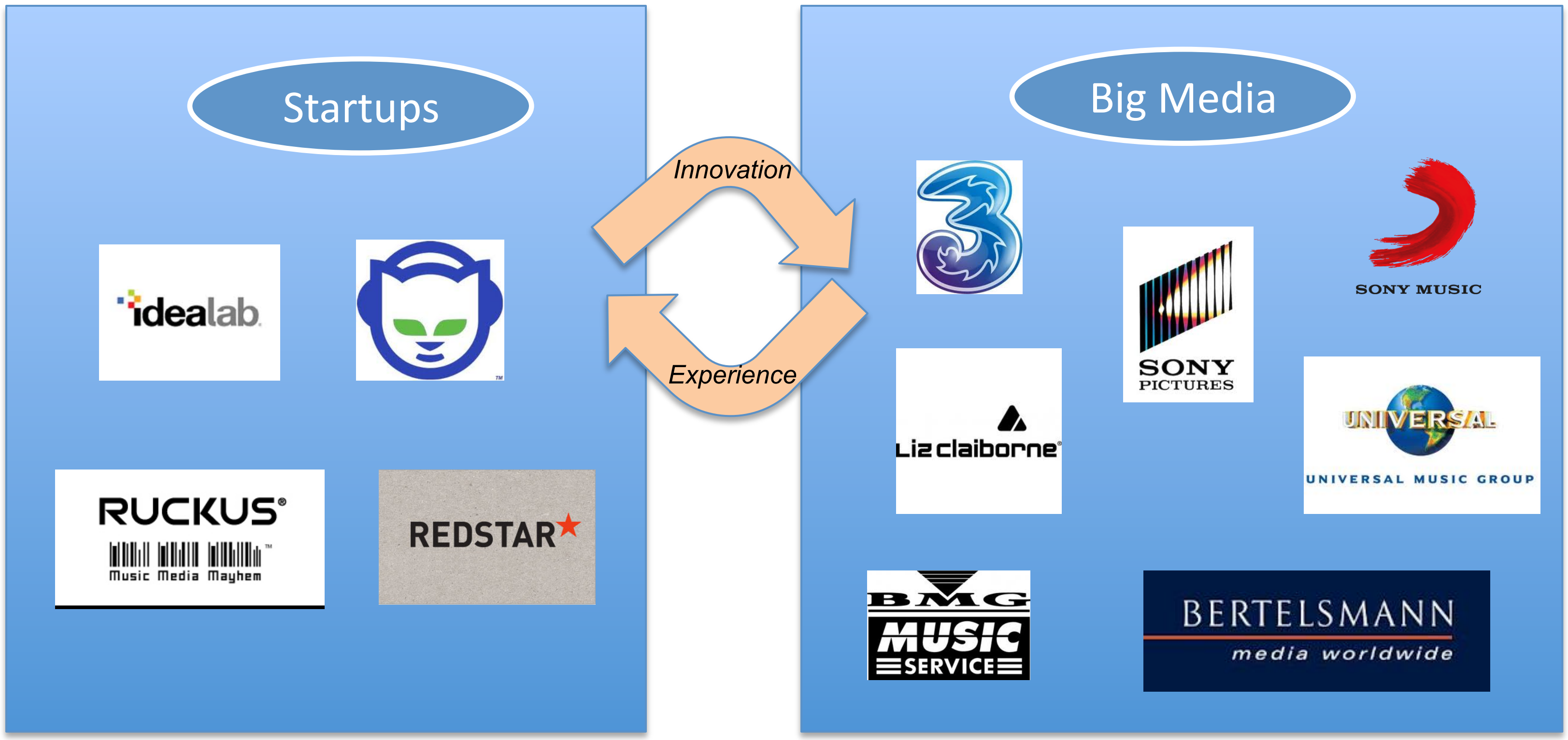
Introducing Objectlab

- Boutique technology consulting practice
- Research, advise, manage, design, architect, implement
- Technology Focus: the Intersection of ***Important*** + ***Fun***
- Open Source Contributors: OpenIPMP … and now our Party App!

"We help startups get started …

… and help large companies adapt to a changing landscape"

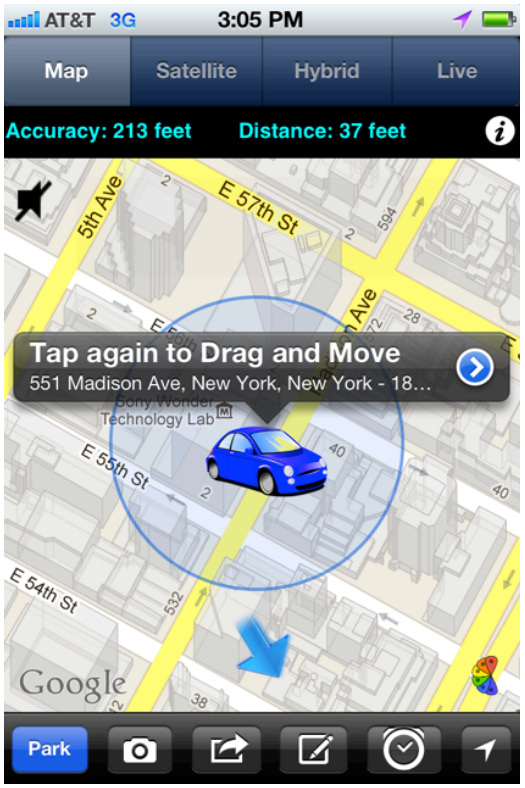
OpenIPMP DIGITAL RIGHTS MANAGEMENT

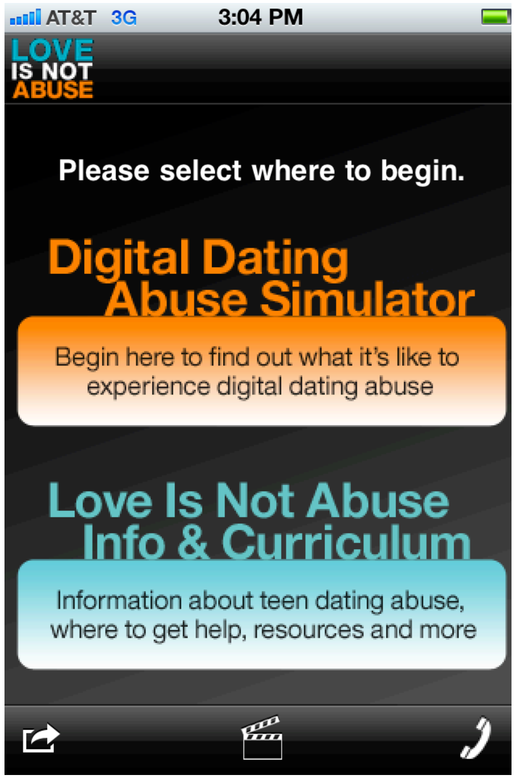


Apps on the Side

appsontheside

- A new fun brand for our mobile pursuits
- Learning: Location awareness (Back to My Car)
- Client Work: simulations using computer generated phone calls and text messages





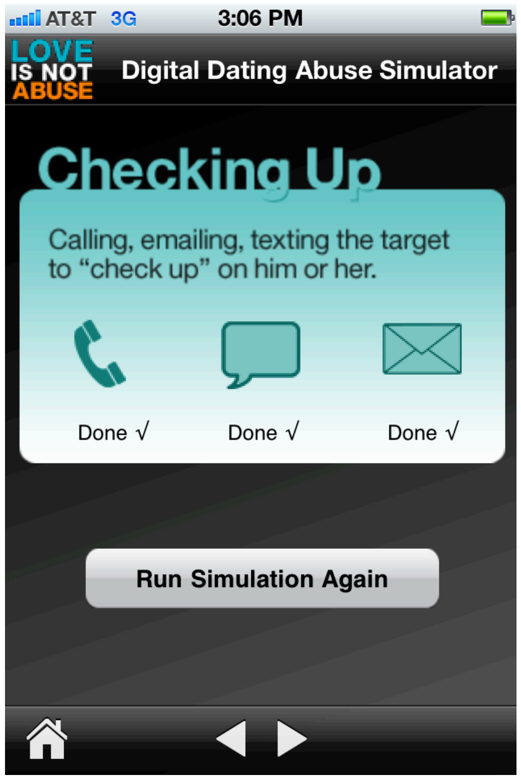
Please select where to begin.

Digital Dating Abuse Simulator

Begin here to find out what it's like to experience digital dating abuse

Love Is Not Abuse Info & Curriculum

Information about teen dating abuse, where to get help, resources and more



Checking Up

Calling, emailing, texting the target to "check up" on him or her.

Objectlab's Holiday Party

- Always trying to impress our entertainment-minded clients
- Over a decade of trying to out-do ourselves with wacky themes and entertainment
- For our 2013 party, it occurred to us to use our *technology* strengths for a change …

Objectlab's Holiday Party

- Always trying to impress our entertainment-minded clients
- Over a decade of trying to out-do ourselves with wacky themes and entertainment
- For our 2013 party, it occurred to us to use our *technology* strengths for a change …

Objectlab's Holiday Party

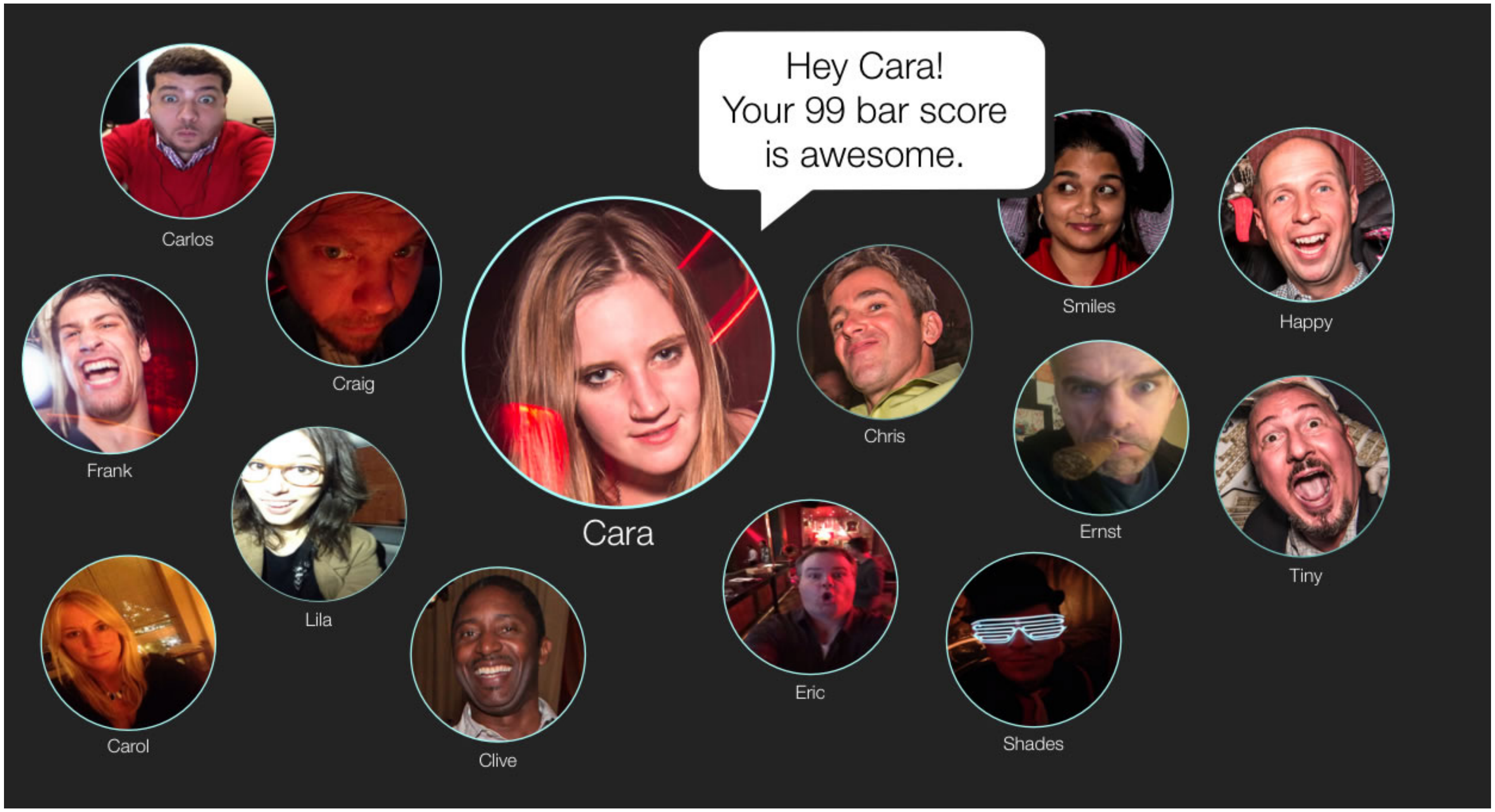
- Always trying to impress our entertainment-minded clients
- Over a decade of trying to out-do ourselves with wacky themes and entertainment
- For our 2013 party, it occurred to us to use our *technology* strengths for a change …

Objectlab’s Holiday Party

- Always trying to impress our entertainment-minded clients
- Over a decade of trying to out-do ourselves with wacky themes and entertainment
- For our 2013 party, it occurred to us to use our *technology* strengths for a change …

What Kind of Game?

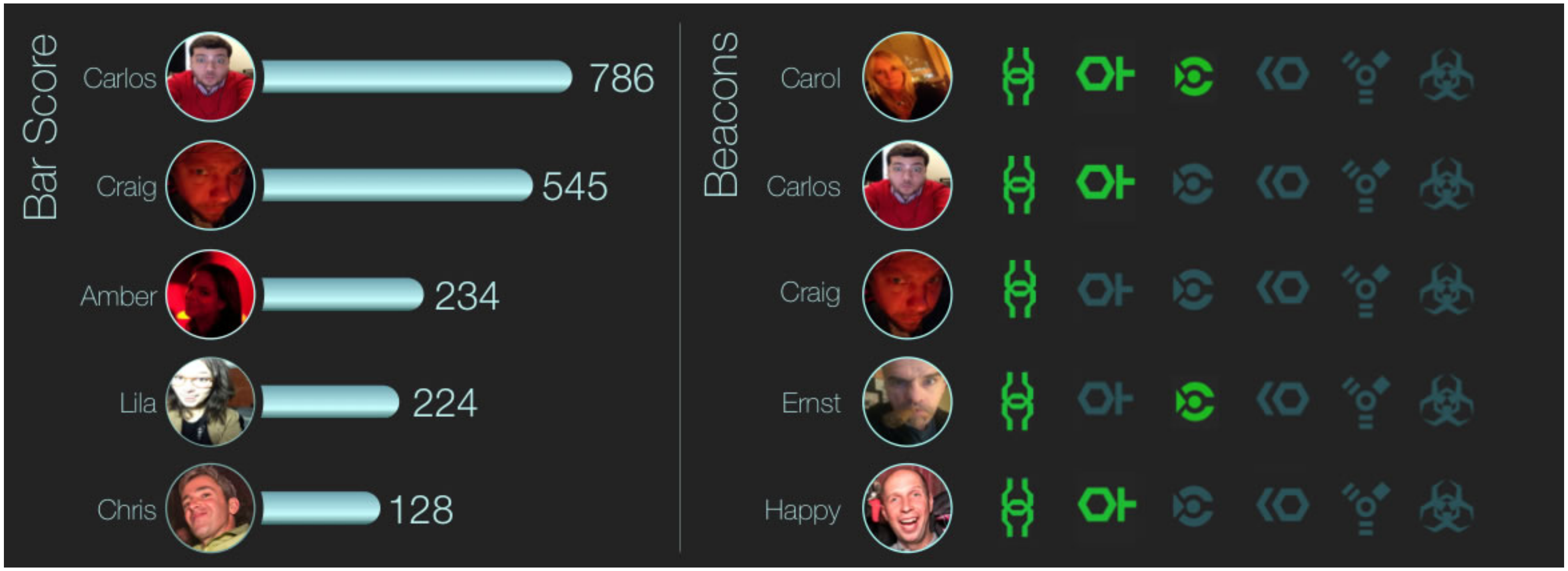
- Fun and interactive; not a "heads down" game that folks play on the subway
- Foster Community; use TVs to provide real-time and public achievement recognition
- Use the TVs + App to "teach" folks how to play; …. but leave some mystery



http://tinyurl.com/mcvz4mf

Game Play

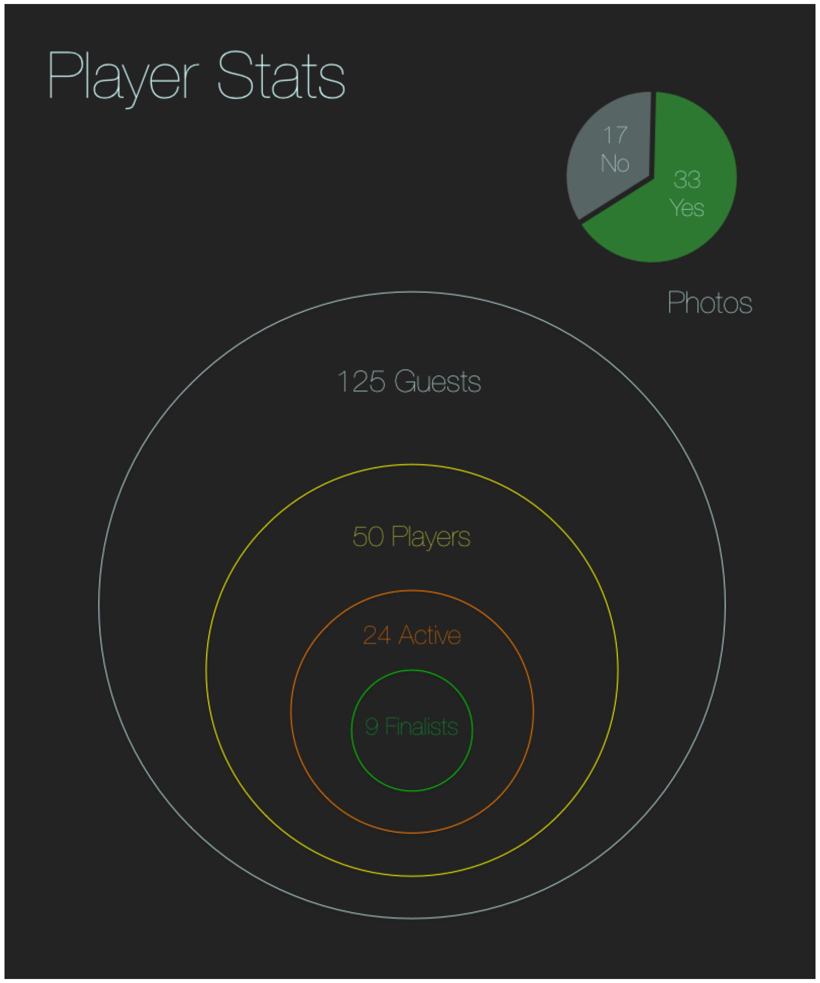
- Points for being at the BAR
- Points for leaving the bar to find hidden iBeacons (… be the 1st to find all 6!)
- Secret formula for scoring ….

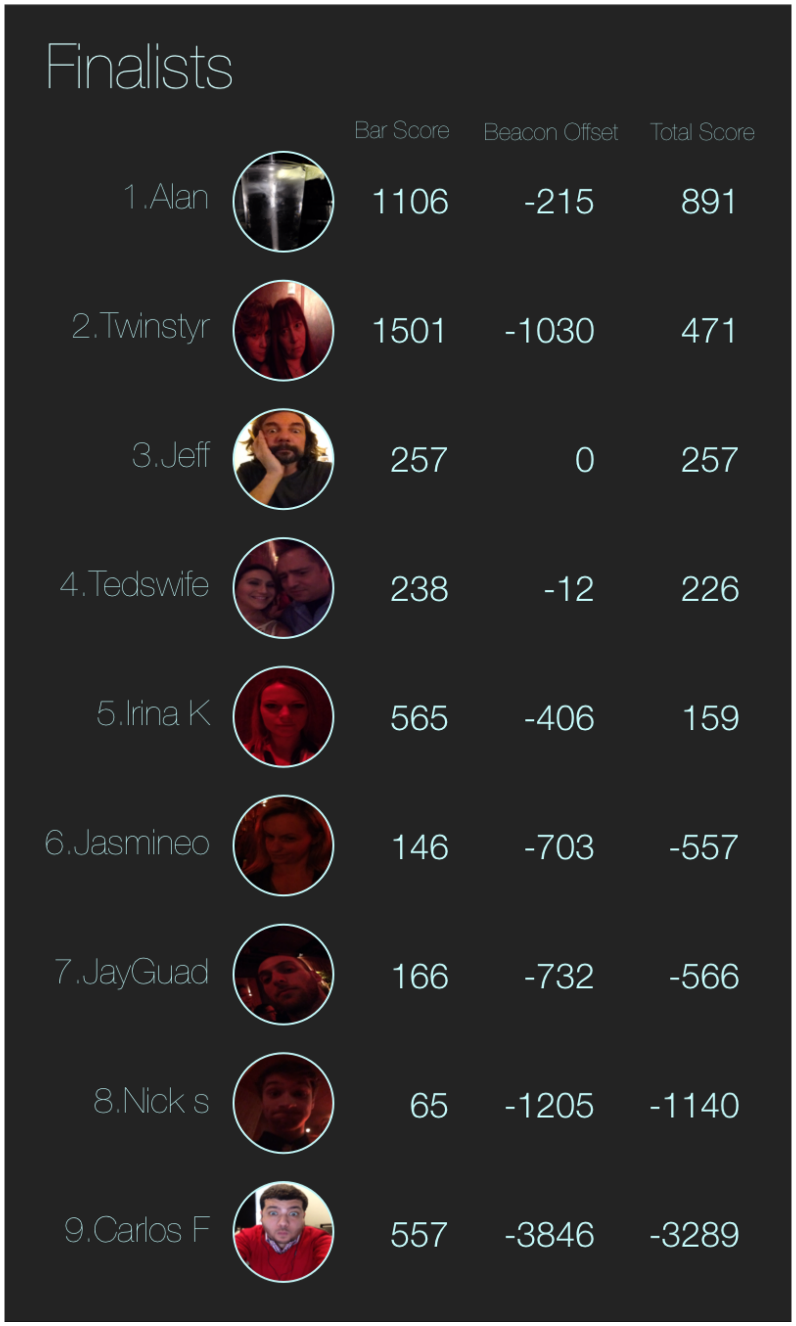


http://tinyurl.com/mcdw7eo

How'd it Go?

- Better than we expected
- Folks engaged with varying strategies
- Could have engaged more with Android



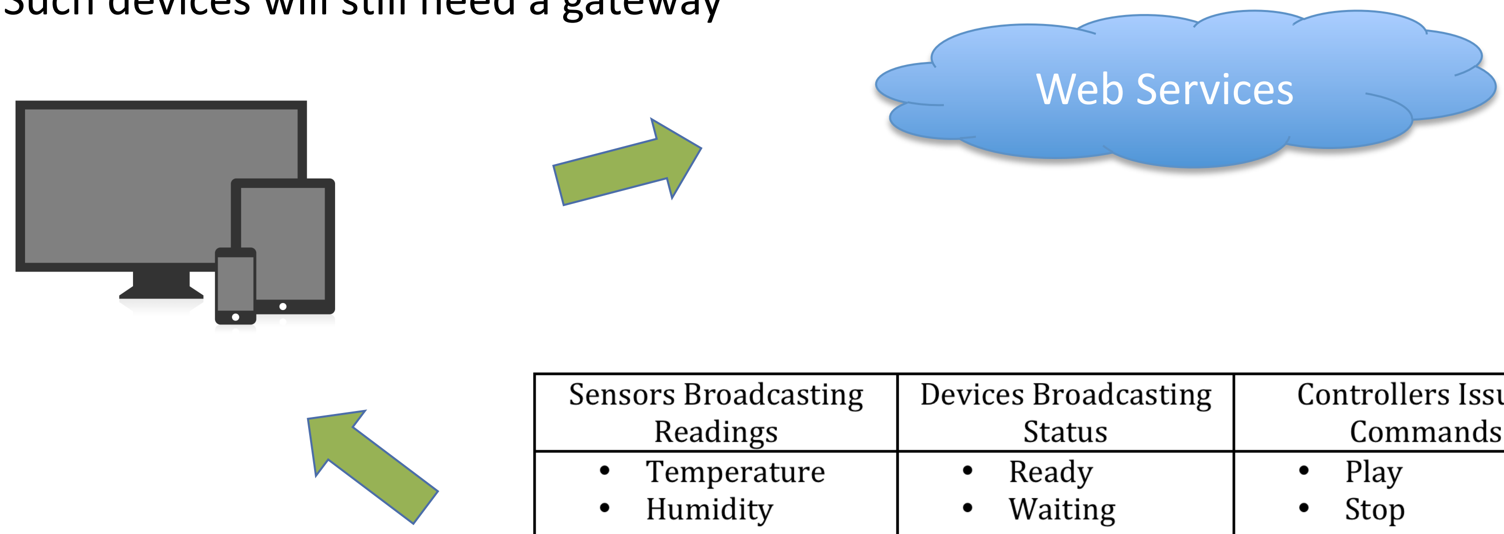


Finalists

		Bar Score	Beacon Offset	Total Score
1.Alan		1106	-215	891
2.Twinstyr		1501	-1030	471
3.Jeff		257	0	257
4.Tedswife		238	-12	226
5.Irina K		565	-406	159
6.Jasmineo		146	-703	-557
7.JayGuad		166	-732	-566
8.Nick s		65	-1205	-1140
9.Carlos F		557	-3846	-3289

Bluetooth LE

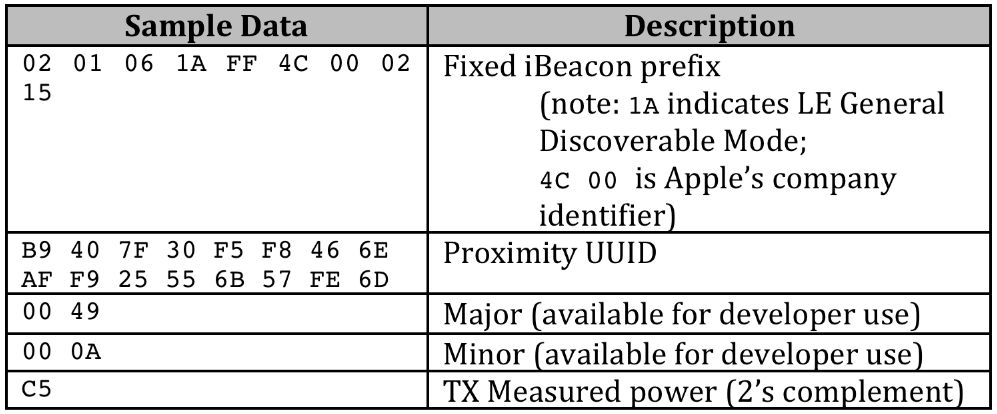
- LE = Low Energy
- Quick delivery of small payloads
- Similar to older ANT+ but uses Bluetooth radio + antenna
- Important for the "Internet of Things"
- Such devices will still need a gateway



Sensors Broadcasting Readings	Devices Broadcasting Status	Controllers Issuing Commands
• Temperature • Humidity • Tire Pressure • Blood Pressure • Barometric Pressure • Altitude • Weight	• Ready • Waiting • Locked • Processing • In Error	• Play • Stop • Unlock • Change Channel • Dehumidify • Dim lights

iBeacon

- Apple implementation of Bluetooth LE for Proximity (… although not closed)
- Extension of *Location Services*
- Device that broadcasts an advertising packet with a specific packet
- Advertising it's identity and info about signal strength



Sample Data	Description
`02 01 06 1A FF 4C 00 02 15`	Fixed iBeacon prefix (note: `1A` indicates LE General Discoverable Mode; `4C 00` is Apple's company identifier)
`B9 40 7F 30 F5 F8 46 6E AF F9 25 55 6B 57 FE 6D`	Proximity UUID
`00 49`	Major (available for developer use)
`00 0A`	Minor (available for developer use)
`C5`	TX Measured power (2's complement)

- TX Measured Power – strength at 1 meter
- Receiver gets this packet, plus the measured strength of the received signal
- Can calculate distance by comparing received signal strength and measured TX
- Apple abstracts this to a measurement of meters – or "Immediate", "Near", "Far"

Our Implementation

- We needed 8 iBeacons; but only the Bar Beacons needed background monitoring

Type	Count	UUID	Major/Minor
Claim	6	E2C56DB5-DFFB-48D2-B060-D0F5A71096E0	1-6
Bar	2	5A4BCFCE-174E-4BAC-A814-092E77F6B7E5	Not applicable

- Used 2 generic “developer” UUIDs
- Able to use Air Locate for Testing
- Made approval with Apple easier
- Can use these, but normally use your own

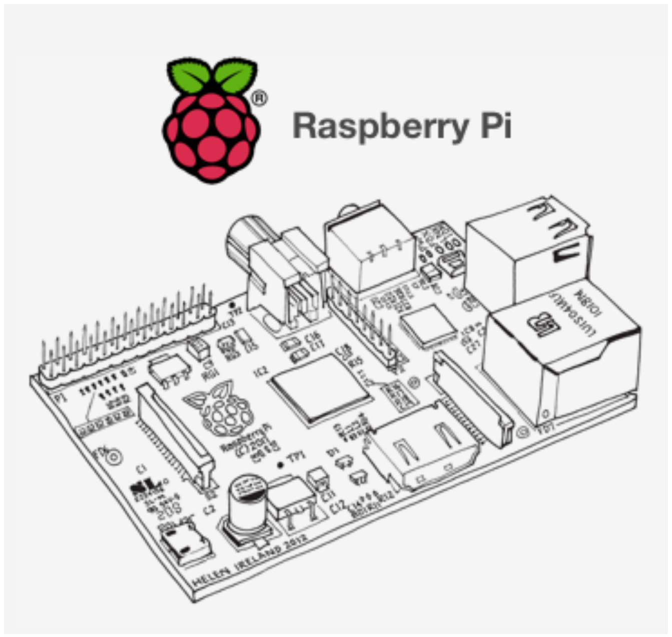
```
E2C56DB5-DFFB-48D2-B060-D0F5A71096E0
5A4BCFCE-174E-4BAC-A814-092E77F6B7E5
74278BDA-B644-4520-8F0C-720EAF059935
112EBB9D-B8C9-4ABD-9EB3-43578BF86A41
22A17B43-552A-4482-865F-597D4C10BACC
33D8E127-4E58-485B-BEE7-266526D8ECB2
44F506A4-B778-4C4E-8522-157AAC0EFABD
552452FE-F374-47C4-BFAD-9EA4165E1BD9
```

➢ *iBeacon Registry Services?*
  - DNS-like registry of UUIDs could be helpful
  - Make iBeacon deployment public; advertise capabilities
  - Might see vertical implementations

Raspberry Pi as iBeacon

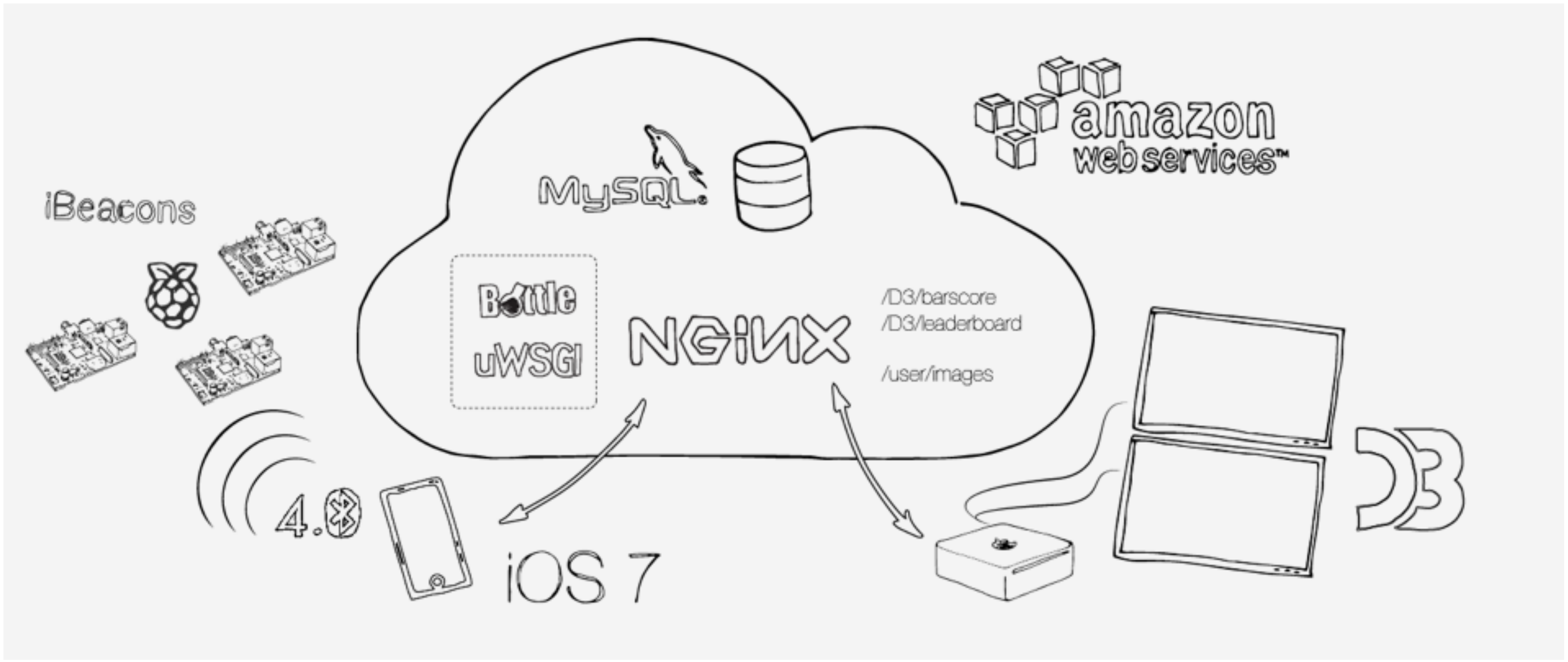
- Companies manufacture iBeacons for as little as $10 (*Estimote*)
- Lead time was too long; we made our own
- Raspberry Pi ($40) + Bluetooth dongle ($10)

- Raspberry PI: Low cost educational platform
- Also great for prototypical development
- Valuable tool for pursuing "Internet of Things"



Step	UUID
1	Flash a SD Card with Rasbian
2	Install BlueZ drivers (to support Bluetooth)
3	Start BlueZ Bluetooth
4	Establish a config file with the UUID, Major/Minor numbers (and a default TX value)
5	Instruct the BlueZ drivers to begin advertising the iBeacon packet
6	Use Air Locate (on IOS device) to measure signal strength at 1 meter
7	Modify the config file with the new Tx value)
8	Add the iBeacon advertising startup command to the boot script

Server Overview



Server

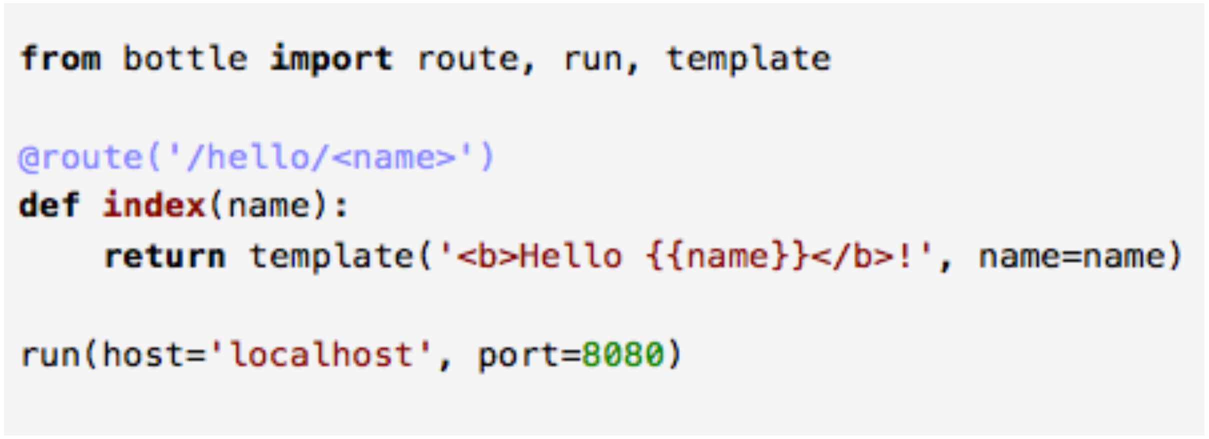
- Primary Functions:
  - Central networked data service
    - Serve and store images, html, javascript
  - Use-case coordinator:
    - User registration
    - User image upload and storage
    - Bar score update
    - iBeacon ‘claiming’
    - Update leaderboards

Server

- Hosting
  - Ubuntu 12.04.3 on AWS
- Software
  - MySQL 5.5.34
  - Bottle (python)

App Server

- Python Micro-framework
  - Easy to use, easy to deploy



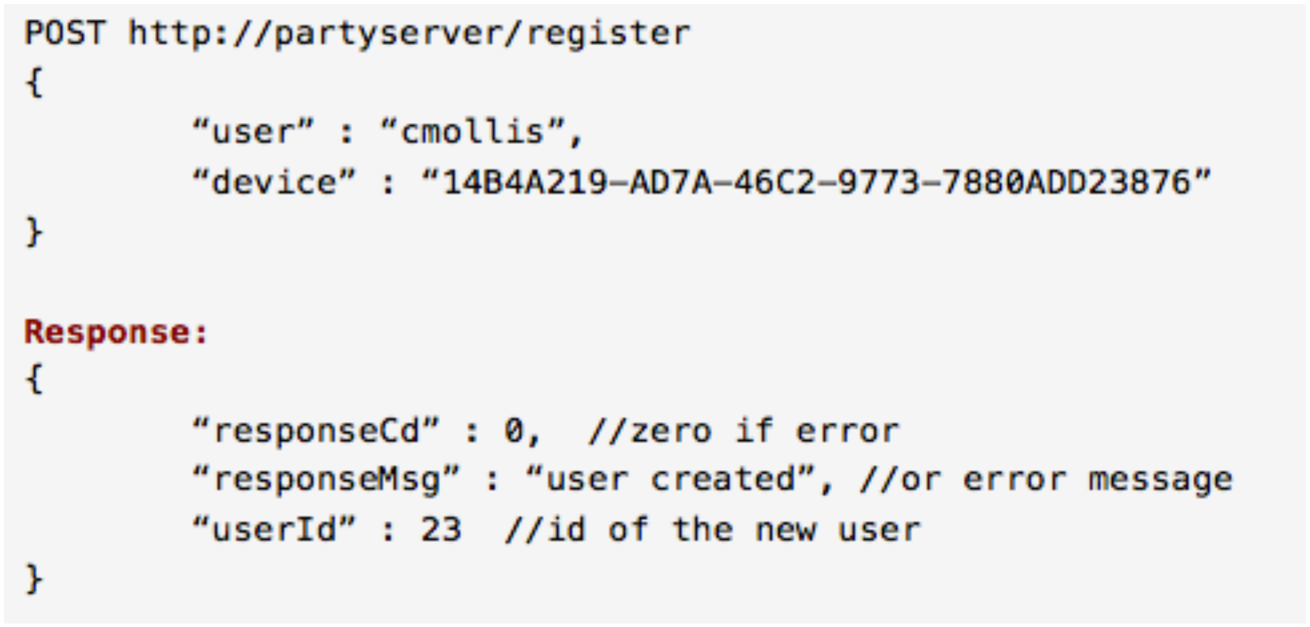
```python
from bottle import route, run, template

@route('/hello/<name>')
def index(name):
    return template('<b>Hello {{name}}</b>!', name=name)

run(host='localhost', port=8080)
```

Protocol

- JSON REST(ish)
  - Easily supported in Cocoa, Javascript, Python

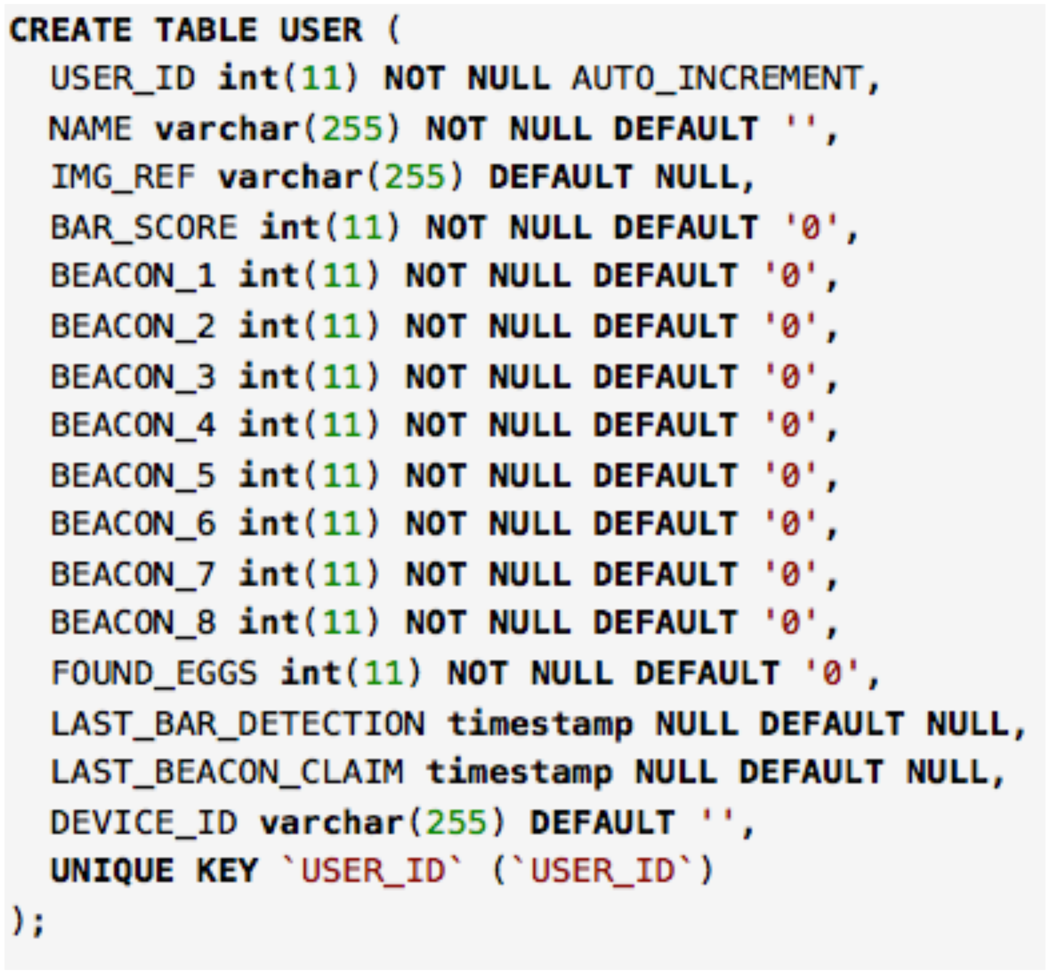


```
POST http://partyserver/register
{
        “user” : “cmollis”,
        “device” : “14B4A219-AD7A-46C2-9773-7880ADD23876”
}

Response:
{
        “responseCd” : 0,  //zero if error
        “responseMsg” : “user created”, //or error message
        “userId” : 23  //id of the new user
}
```

Schema

- Single table, highly de-normalized (not recommended for production use)

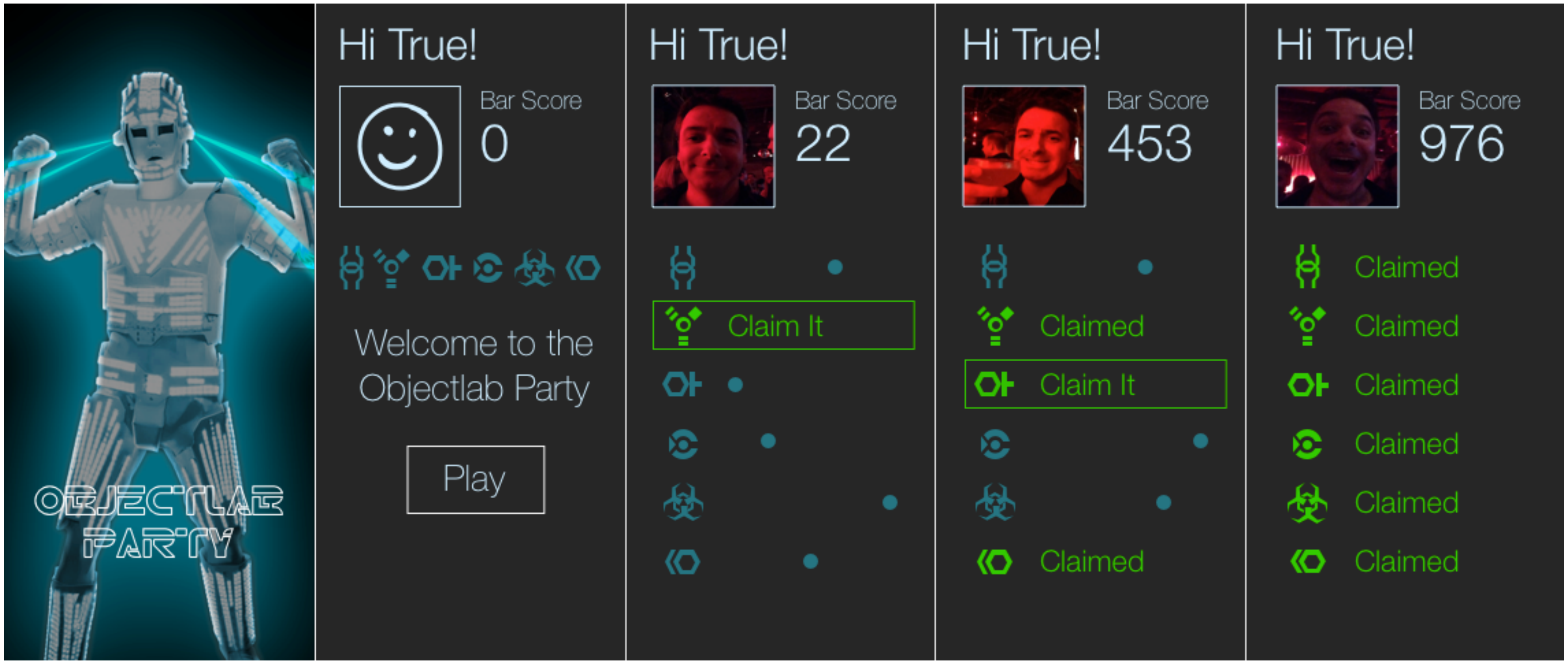


```
CREATE TABLE USER (
  USER_ID int(11) NOT NULL AUTO_INCREMENT,
  NAME varchar(255) NOT NULL DEFAULT '',
  IMG_REF varchar(255) DEFAULT NULL,
  BAR_SCORE int(11) NOT NULL DEFAULT '0',
  BEACON_1 int(11) NOT NULL DEFAULT '0',
  BEACON_2 int(11) NOT NULL DEFAULT '0',
  BEACON_3 int(11) NOT NULL DEFAULT '0',
  BEACON_4 int(11) NOT NULL DEFAULT '0',
  BEACON_5 int(11) NOT NULL DEFAULT '0',
  BEACON_6 int(11) NOT NULL DEFAULT '0',
  BEACON_7 int(11) NOT NULL DEFAULT '0',
  BEACON_8 int(11) NOT NULL DEFAULT '0',
  FOUND_EGGS int(11) NOT NULL DEFAULT '0',
  LAST_BAR_DETECTION timestamp NULL DEFAULT NULL,
  LAST_BEACON_CLAIM timestamp NULL DEFAULT NULL,
  DEVICE_ID varchar(255) DEFAULT '',
  UNIQUE KEY `USER_ID` (`USER_ID`)
);
```

Deployment

- NGINX
  - Serve static files (images, html, etc)
  - Reverse-proxy to bottle app
- uWSGI
  - High-performance web container for WSGI apps

IOS App Overview



iOS App

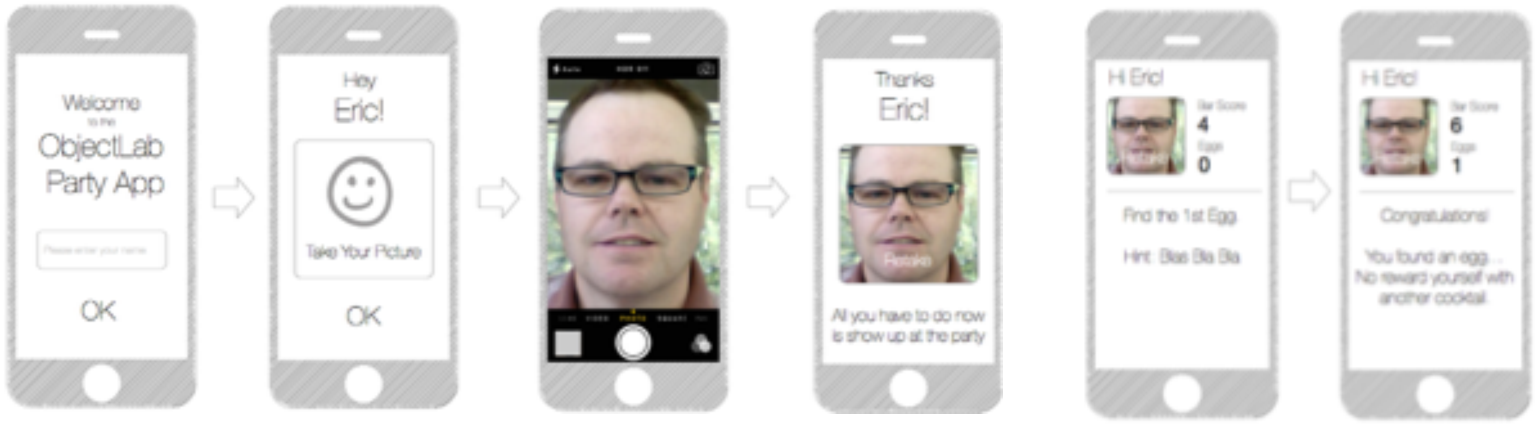
- At the outset of development, we had a few unknowns:
  - Accuracy of RSSI-to-distance calculations
  - APIs were confusing with few code examples
  - Raspberry Pi's problematic?
  - Would Apple accept an app like ours? Most iBeacon apps (at the time) were 'test' apps. It wasn't obvious what our app was doing (which tends to annoy Apple)

iOS App

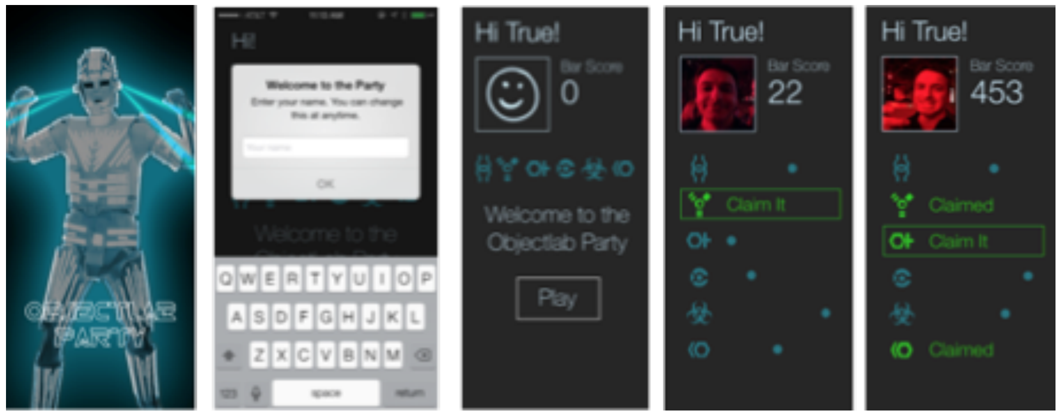
- Needed a good starting point to test required functionality : ***AirLocate***
- Demonstrates API usage:
  - iBeacon proximity detection
  - IOS device-based iBeacon (CBPeripheralManager)
  - RSSI calibration (range at 1m)
- Allowed us to test our major use-cases and technical assumptions

Design

- Simple single-view controller design



Became…

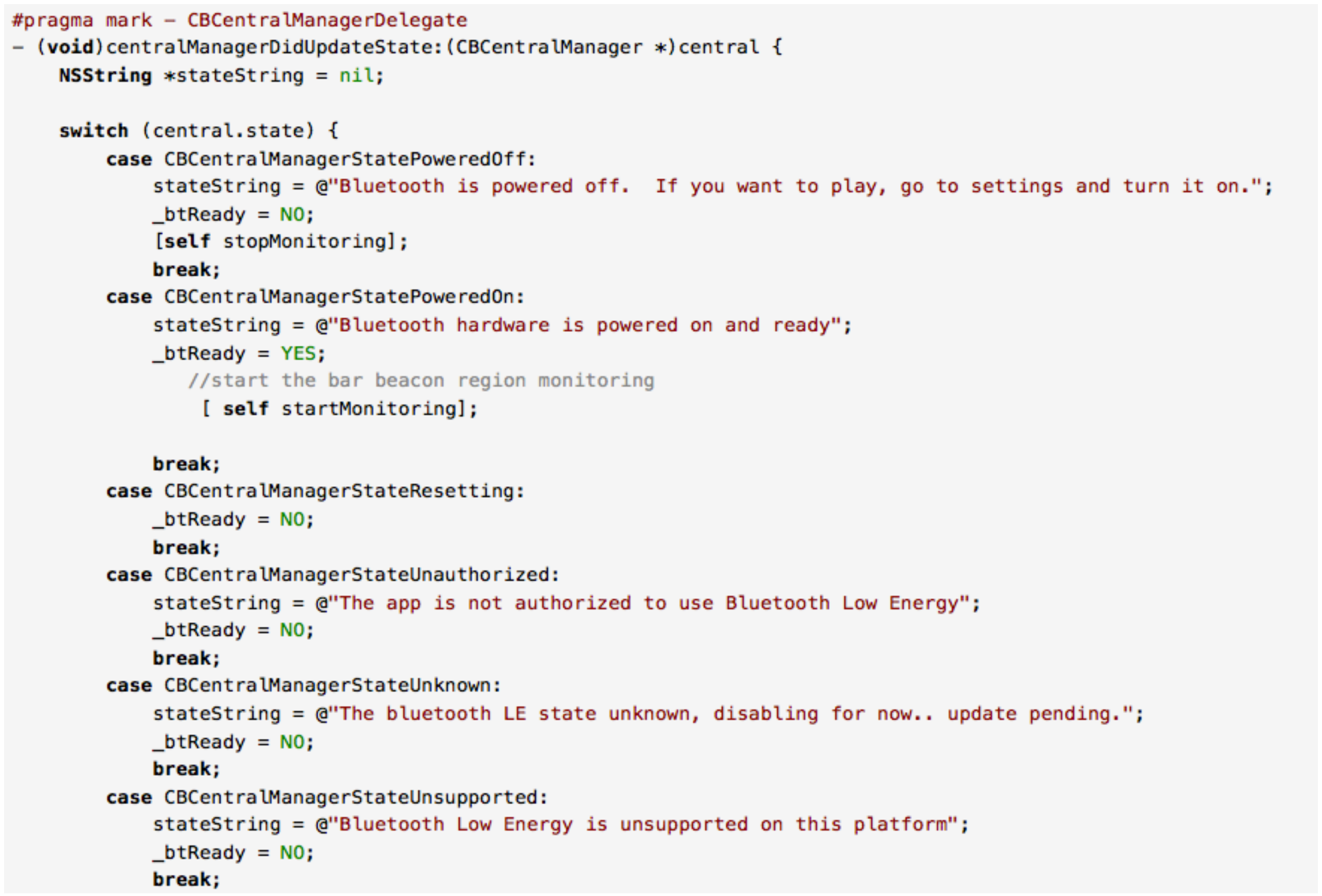


Third-party Components

- CocoaPods
  - AFNetworking 2.0
  - SGNavigationProgress
  - TSMessages
- UIImage Category
  - Handles image transformations from the camera (resize, rotate, orientation, etc)

Sanity Checks

- ‘Reachability’ (standard, used AFNetworking)
- Bluetooth checks (CBCentralManager)
  - Implement CBCentralManagerDelegate protocol to receive Bluetooth status at any time



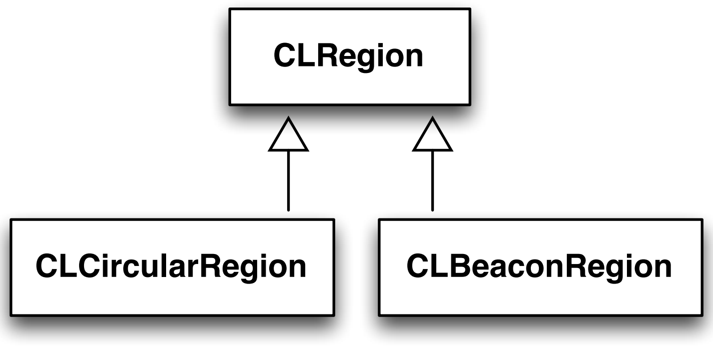
```objc
#pragma mark - CBCentralManagerDelegate
- (void)centralManagerDidUpdateState:(CBCentralManager *)central {
    NSString *stateString = nil;

    switch (central.state) {
        case CBCentralManagerStatePoweredOff:
            stateString = @"Bluetooth is powered off.  If you want to play, go to settings and turn it on.";
            _btReady = NO;
            [self stopMonitoring];
            break;
        case CBCentralManagerStatePoweredOn:
            stateString = @"Bluetooth hardware is powered on and ready";
            _btReady = YES;
               //start the bar beacon region monitoring
                [ self startMonitoring];

            break;
        case CBCentralManagerStateResetting:
            _btReady = NO;
            break;
        case CBCentralManagerStateUnauthorized:
            stateString = @"The app is not authorized to use Bluetooth Low Energy";
            _btReady = NO;
            break;
        case CBCentralManagerStateUnknown:
            stateString = @"The bluetooth LE state unknown, disabling for now.. update pending.";
            _btReady = NO;
            break;
        case CBCentralManagerStateUnsupported:
            stateString = @"Bluetooth Low Energy is unsupported on this platform";
            _btReady = NO;
            break;
```

CoreLocation

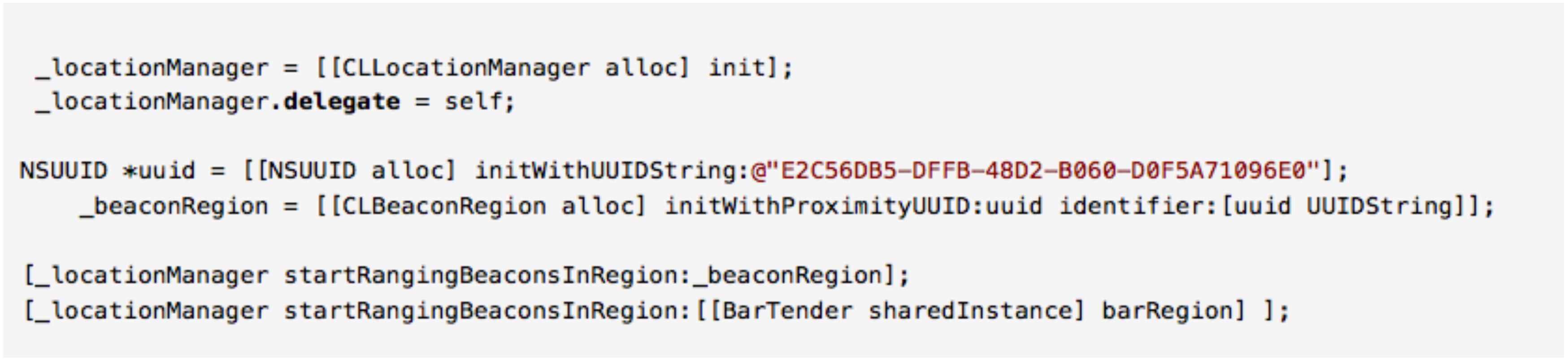
- Apple extends this API beyond GPS ‘proximity’ to include device proximity



- CLBeaconRegion governed by bluetooth range (~70m)
- Proximity accuracy to < *1m*
- iBeacon BLE packets differentiated by Apple GATT and passed through IOS stack to CoreLocation services
- Applications include:
  - *In-store advertisements*
  - *Indoor positioning*
  - *Point-of-sale*

CoreLocation : new APIs

- New CLLocationManager and CLLocationManagerDelegate functions



```
 _locationManager = [[CLLocationManager alloc] init];
 _locationManager.delegate = self;

NSUUID *uuid = [[NSUUID alloc] initWithUUIDString:@"E2C56DB5-DFFB-48D2-B060-D0F5A71096E0"];
    _beaconRegion = [[CLBeaconRegion alloc] initWithProximityUUID:uuid identifier:[uuid UUIDString]];

[_locationManager startRangingBeaconsInRegion:_beaconRegion];
[_locationManager startRangingBeaconsInRegion:[[BarTender sharedInstance] barRegion] ];
```

CoreLocation : new APIs

- Proximity detection : callbacks every second for each CLBeaconRegion instance

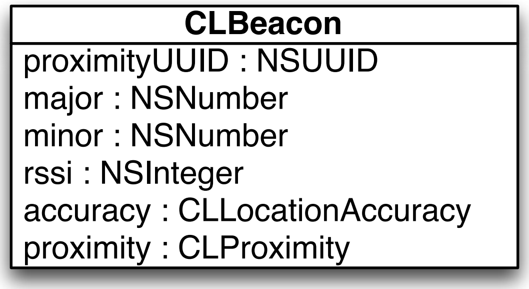
```
#pragma mark - CLLocationManagerDelegate
- (void)locationManager:(CLLocationManager *)manager didRangeBeacons:(NSArray *)beacons inRegion:(CLBeaconRegion *)region
{
    //CM should be two beacons if the region is the bar region
    if ([[region.proximityUUID UUIDString] isEqualToString:[[[BarTender sharedInstance] defaultProximityUUID] UUIDString]]) {

        [[BarTender sharedInstance] checkForBarProximity:beacons];
        //return out of this if it's the bar UUID
        return;
    }
    else {
        // claimable beacons
        _rangedBeacons.array = beacons;
        [self.tableView reloadData];
    }

}
```

CoreLocation : Distance

- CLLocationManagerDelegate::didRangeBeacons will return an array of **CLBeacon** instances, if it can find any.



CLBeacon

proximityUUID : NSUUID
major : NSNumber
minor : NSNumber
rssi : NSInteger
accuracy : CLLocationAccuracy
proximity : CLProximity

- CLLocationAccuracy (float) is the detected accuracy in meters
- CLProximity is determined by Apple (generalizes distance)
  - *CLProximityUnknown*
  - *CLProximityNear*
  - *CLProximityFar*
  - *CLProximityImmediate*

CoreLocation : Distance

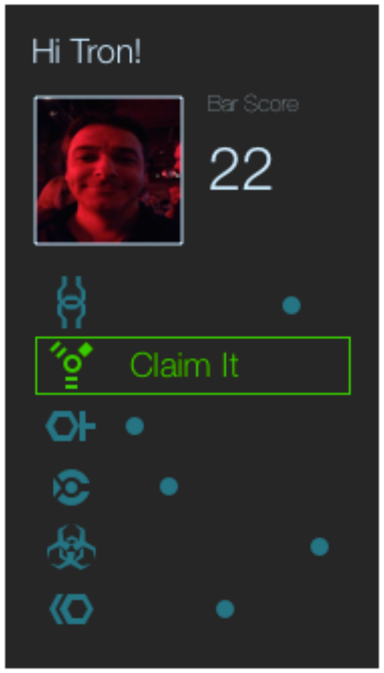
- Apple interpolates distance as a function of ***RSSI***
  - RSSI? Received Signal Strength in decibels (dbm).
  - One type of generalized function (there are many others, have no idea what Apple uses):
    - RSSI[dbm] = −(10n log10(d) − A) where:
      - d is the distance
      - A is the offset RSSI calibrated at 1m
      - N is a scalar used for different terrains
  - Apple samples the RSSI values in between ranging callbacks and computes the average.
- Depends on many factors:
  - Line of sight
  - Radio interference
  - Weather?
- May vary from location to location
- *May fluctuate wildly*

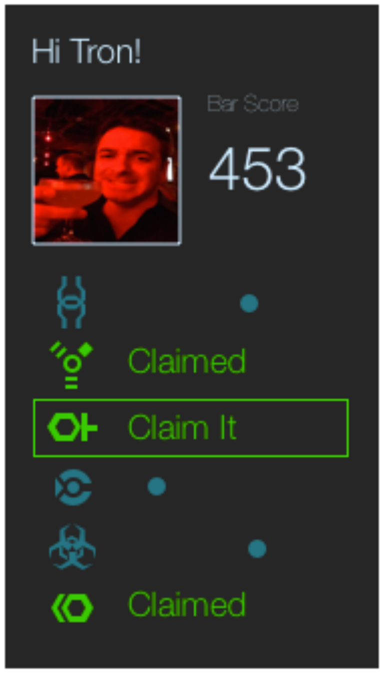
Game Play

- Two main iBeacon proximity use-cases:
  1. iBeacon ‘treasure hunt’ (users must hunt for the hidden beacons, more ‘active’)
  2. Bar Scoring (just being near the bar, more ‘passive’)

Game Play : Finding Hidden Beacons

- Seeded a local CoreData db with 6 beacon records distributed with app.
- Allowed us to:
  - Track and modify their state visually ('claimed', unclaimed, etc)
  - Track and update the user's position relative to the specific beacon





Game Play : Bar Score

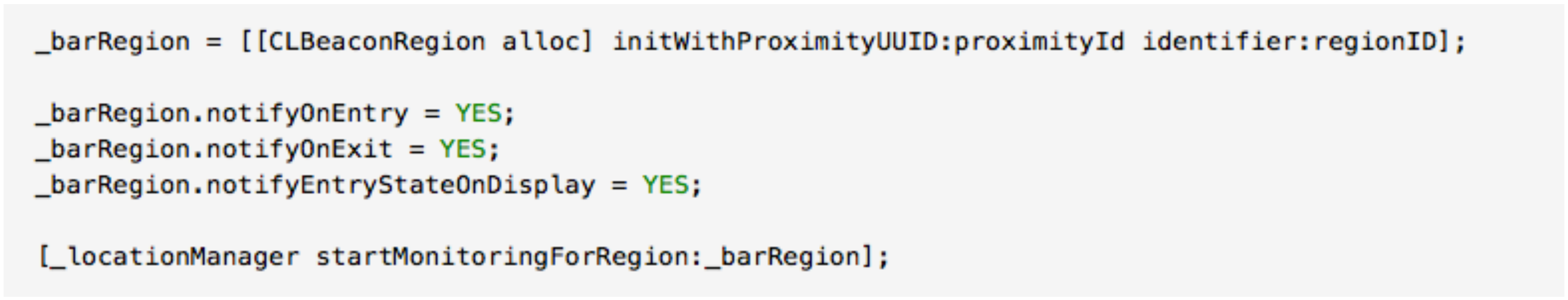
- We wanted the bar scoring to be more of a ‘passive’ experience in that:
  - We wanted it work while the app was in the background
  - We didn’t want the app to constantly update the score every second when proximity was detected
    - Smoother over time (detect every n seconds)
    - Slightly less obvious what we were doing

Game Play : Bar Score

- Background processing had to be handled differently
- CoreLocation allows you to monitor ***regions***
  - Regions defined by GPS (CLCircularRegion)
  - Regions defined by UUID (CLBeaconRegion)
- CoreLocation will notify you when you're inside or outside of a region, when the app is in the foreground and background (although not so much in the background)

Game Play : Bar Score

- You can tell CLLocationManager how you'd like beacon regions to be monitored



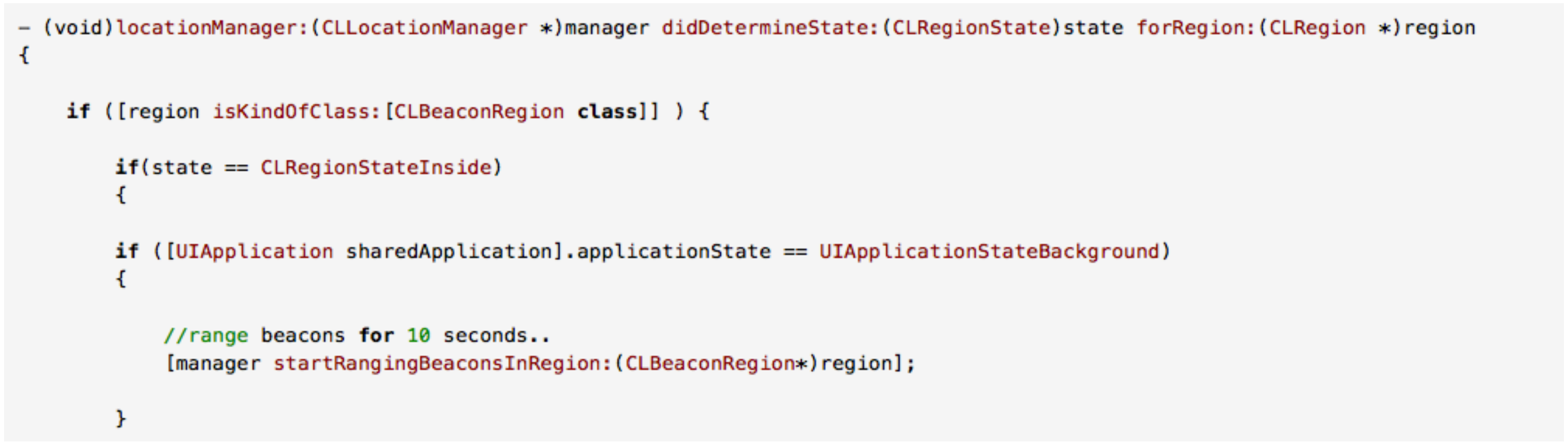
```
_barRegion = [[CLBeaconRegion alloc] initWithProximityUUID:proximityId identifier:regionID];

_barRegion.notifyOnEntry = YES;
_barRegion.notifyOnExit = YES;
_barRegion.notifyEntryStateOnDisplay = YES;

[_locationManager startMonitoringForRegion:_barRegion];
```

Game Play : Bar Score

- Implement the CLLocationManagerDelegate protocol to receive region notifications



```
- (void)locationManager:(CLLocationManager *)manager didDetermineState:(CLRegionState)state forRegion:(CLRegion *)region
{

    if ([region isKindOfClass:[CLBeaconRegion class]] ) {

        if(state == CLRegionStateInside)
        {

        if ([UIApplication sharedApplication].applicationState == UIApplicationStateBackground)
        {

            //range beacons for 10 seconds..
            [manager startRangingBeaconsInRegion:(CLBeaconRegion*)region];

        }
```

Foreground and Background Updates

When the app is in the foreground…

Condition	Max time to detect a region change
App Ranging	1 second
App Not Ranging	Up to 15 mins

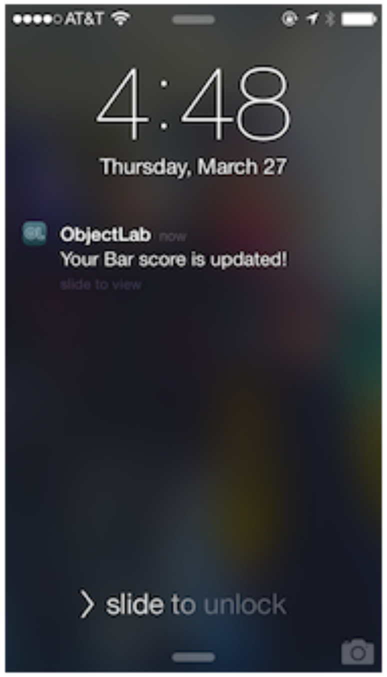
When the app is in the background…

Condition	Max time to detect a region change
Phone awakened, notifyEntryStateOnDisplay=YES	1 second
Phone awakened, notifyEntryStateOnDisplay=NO	Never
UIBackgroundModes=location ON	Up to 15 minutes
UIBackgroundModes=location OFF	Up to 15 minutes

*http://developer.radiusnetworks.com/2013/11/13/ibeacon-monitoring-in-the-background-and-foreground.html

Game Play : Bar Score

- Local notifications when app is ‘backgrounded’



Your Bar score is updated!

Lessons Learned

- iBeacons are closely coupled to specific apps.
  - Apple probably did this for a reason
  - Should be a registry of ibeacons that can be queried at run-time based on GPS.
  - Lots of players trying to create vertical registries
- Apple (or at least the iTunes approval team) wasn't prepared for an app like ours.
  - Took three tries to get it accepted
- Within an event space, real-time visual feedback of group activity was *the* compelling feature

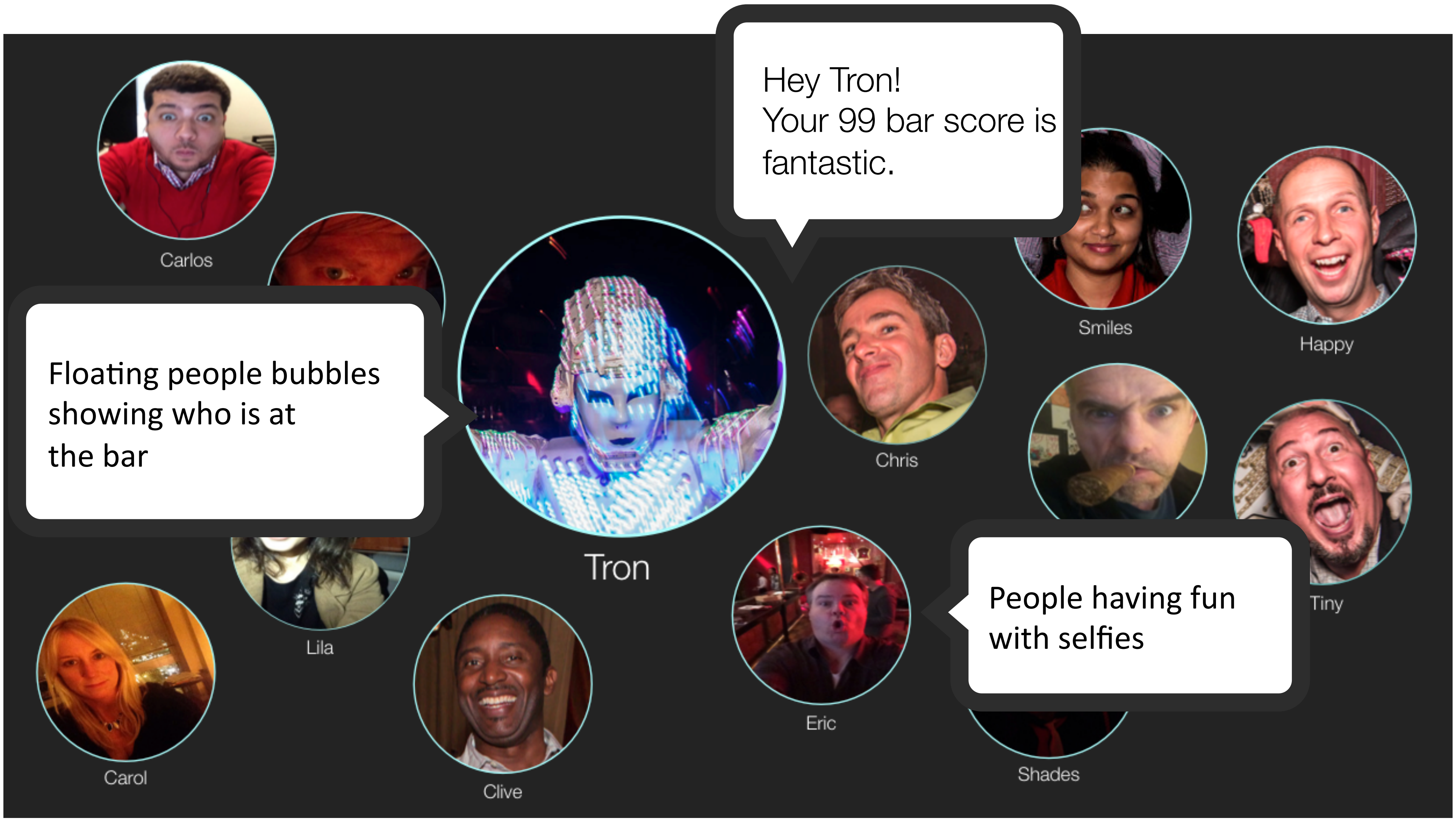
A no brainer for HTML 5 visualization

http://d3js.org/

Visualization Goals

- Draw people into the game
- Share player game status
- Real-time barscore and beacon claims
- Visualize micro location detection

At The Bar Visualization



Hey Tron!
Your 99 bar score is fantastic.

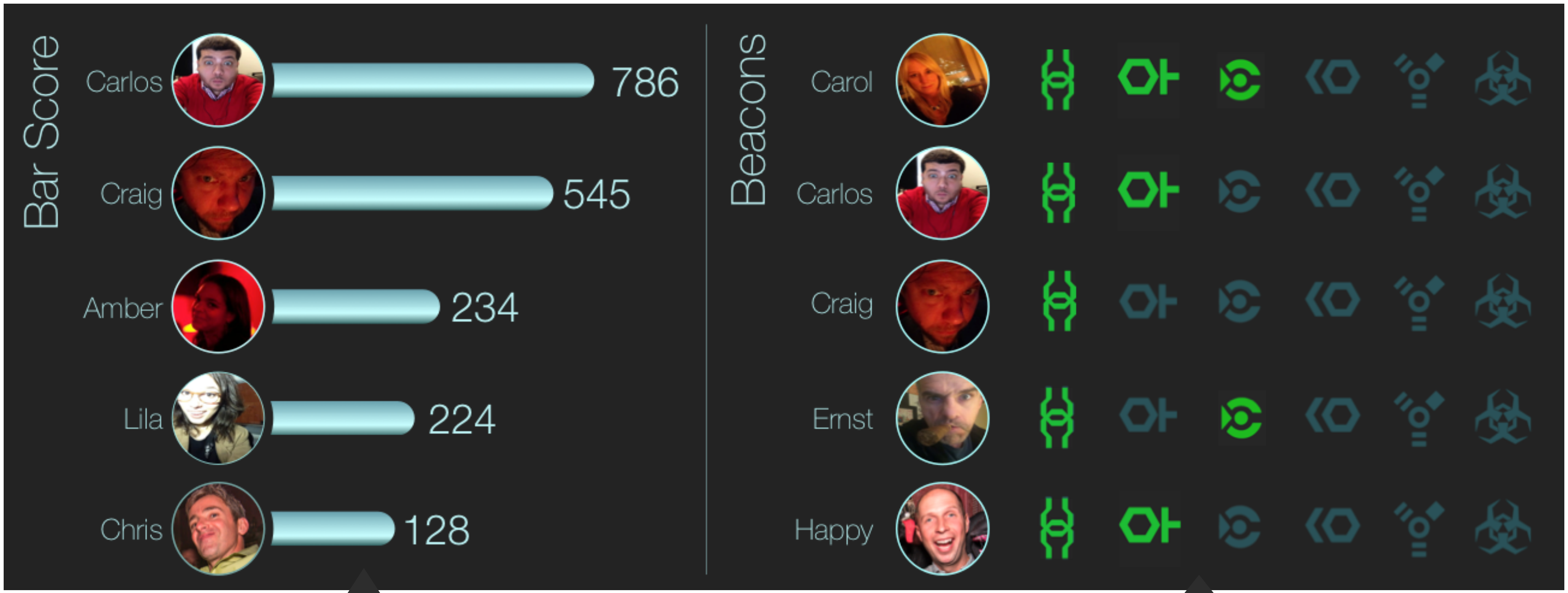
Floating people bubbles showing who is at the bar

People having fun with selfies

At The Bar Visualization

```
// configure the force layout
force = d3.layout.force()
        .nodes(peopleOnScreen)
        .size([width, height])
        .gravity(0.005)
        .charge(-radius*3.5)
        .on("tick", onForceTick)
        .start();
```

Leaderboard Visualization



Simple bar chart with some chrome

Claimed beacons are highlighted