

Translating Biosignals



You will not be able to read someone's mind.

You will not be able to write onto a mind.

You will not be able to predict the future.

Directly replace buttons.

What we cannot do



Add an element of creative intrigue to your game/experience.

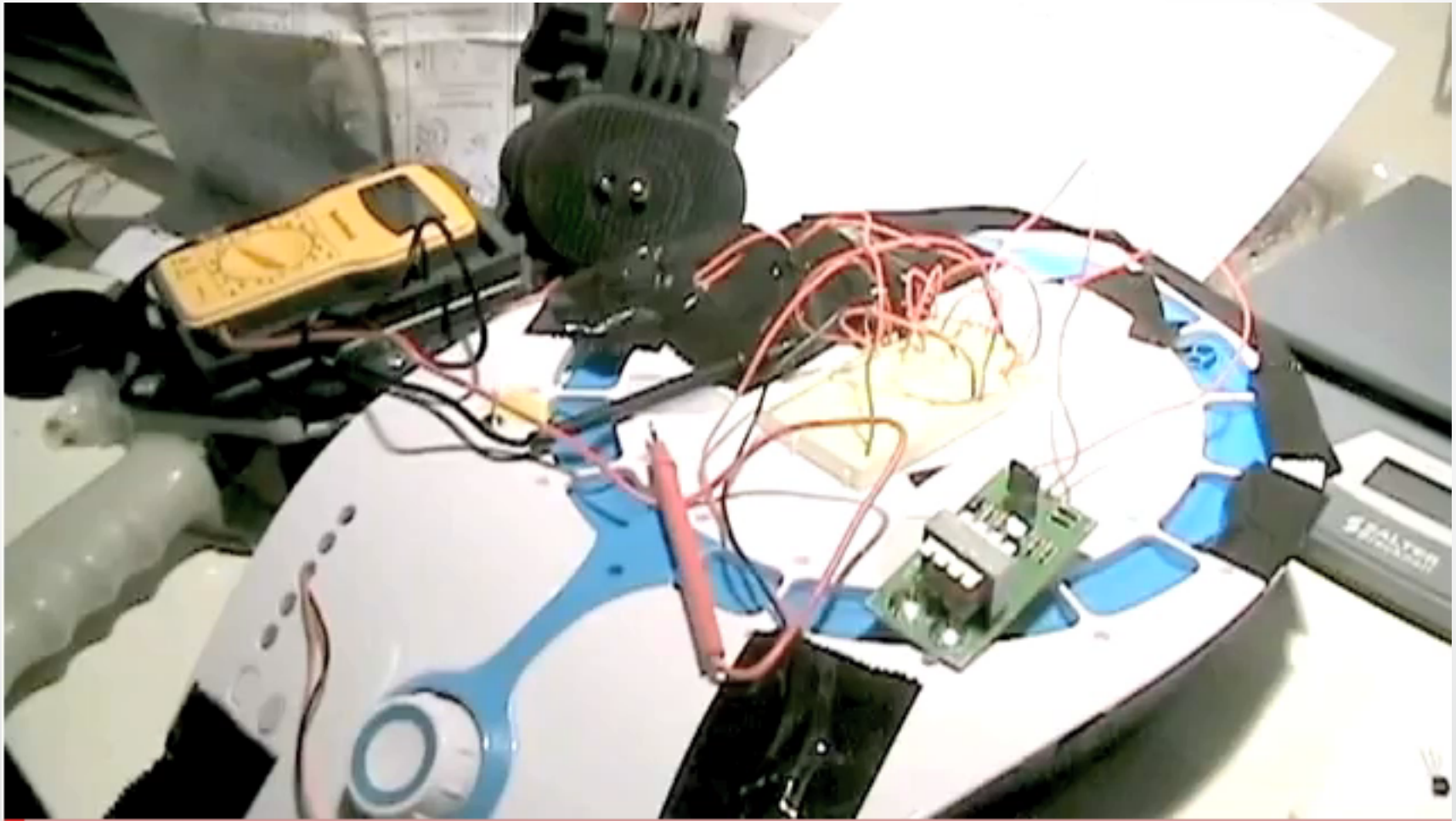
Monitor the body's biorhythms to create health / wellness apps.

Involve the player in a new level of interactivity.

What we can do



MindFlex



MindFlex Hacked



Orbit on Kickstarter

What kind of information is available from our headset / drivers?

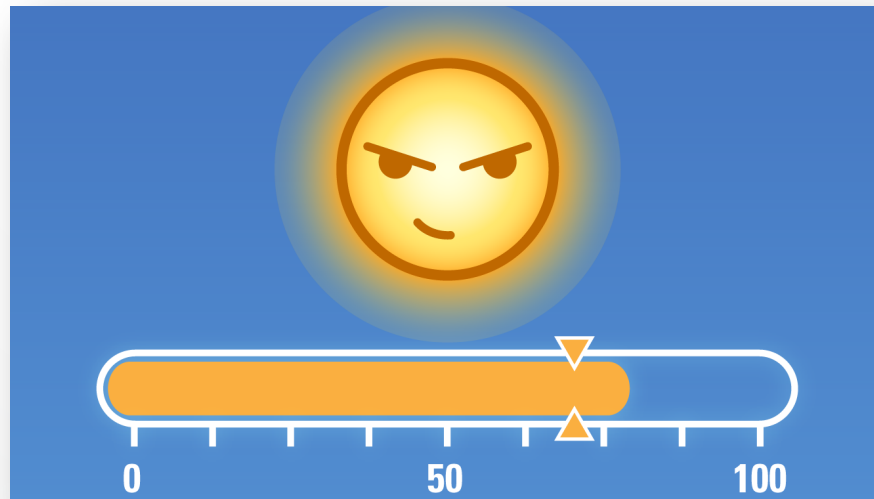
- **Raw Waves**
 - 512 Hz
- **Brainwave Bands**
 - 8 Bands
 - Delta, Theta, High / Low Alpha, High / Low Beta, High / Low Gamma
- **Attention / Meditation eSense**
 - eSense is a relative measurement from 0 – 100 at 1 Hz
- **Eyeblinks**
- **Signal Status**



Headset Output

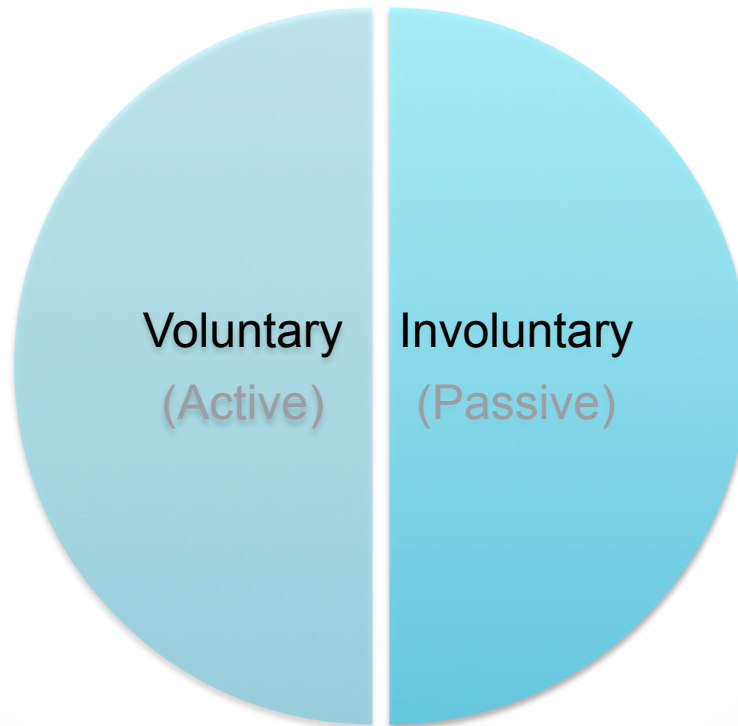
Attention and Meditation are two systems of controls created by NeuroSky.

- Relative, real-time controls ideal for immediate feedback
- 40-60 = Neutral Center



Attention and Meditation eSenses

Two Ways of Applying Brainwaves



Utilization

Qualified Applicants Receive:

- Hardware Resources
 - Loaner Prototypes
 - Discounts
- Alpha APIs
 - Project Examples
- Development Support / Feedback
- Referral Sales: Sell hardware through your applications
- Sell PC/Mac Apps through NeuroSky's Store
 - Mobile App directly through corresponding Mobile Stores



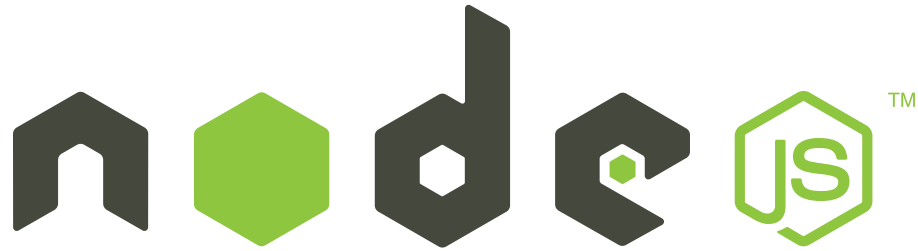
NeuroSky Developer Program

Set your *apps* and project above the
rest with our biosensor technologies.
Accelerate your project towards
innovation and *monetization*.



ThinkGear Connector is a light application that passes bio-sensor information from NeuroSky's headsets to a local server for easy access.

Sockets!



A platform for built from Chrome's JavaScript runtime for building fast, scalable network applications.

node.js

<https://github.com/dluxemburg/node-neurosky>

The newest versions of Node.js will allow for a node packaged module install of the Client library.

\$ npm install node-thinkgear

```
Johnny — bash — 80x24
Last login: Fri Aug 2 10:43:09 on console
Johnnys-MacBook-Air:~ Johnny$ node -v
v0.10.20
Johnnys-MacBook-Air:~ Johnny$ node example.js
Server running at http://127.0.0.1:1337/
^CJohnnys-MacBook-Air:~ Johnny$ npm install node-thinkgear
npm http GET https://registry.npmjs.org/node-thinkgear
npm http 200 https://registry.npmjs.org/node-thinkgear
npm WARN deprecated node-thinkgear@0.0.1: Replaced by node-neurosky
npm http GET https://registry.npmjs.org/node-thinkgear/-/node-thinkgear-0.0.1.tgz
Z
npm http 200 https://registry.npmjs.org/node-thinkgear/-/node-thinkgear-0.0.1.tgz
Z
node-thinkgear@0.0.1 node_modules/node-thinkgear
Johnnys-MacBook-Air:~ Johnny$
```

Node.js client library

app.js

The javascript file does three key steps.

1. Opens up the socket
2. Relays the App Name / App Key
3. Receives the JSON datafeed

Step 1.

Host address: 127.0.0.1

Port: 13854

Protocol: TCP

You will see the ThinkGear Connector (TGC) icon turn into an hourglass as it attempts to connect. The TGC is scanning different COM ports.



Accessing Local Host

Step 2.

An appName and appKey are necessary for the client to handshake with the server. In the app.js example:

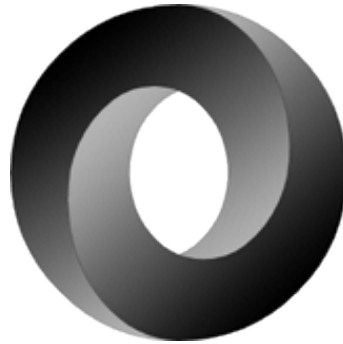
```
var tgClient = nodeThinkGear.createClient({  
  appName:'NodeThinkGear',  
  appKey:'0fc4141b4b45c675cc8d3a765b8d71c5bde9390'  
});
```

The appKey is 40 hexadecimal characters, ideally generated using an SHA-1 digest.

When the headset properly pairs, the icon will turn into a blue brain.



Accessing Local Host



Step 3.

Javascript Object Notation: a lightweight data interchange format that is easy for humans to read and write, and easy for machines to parse and generate.

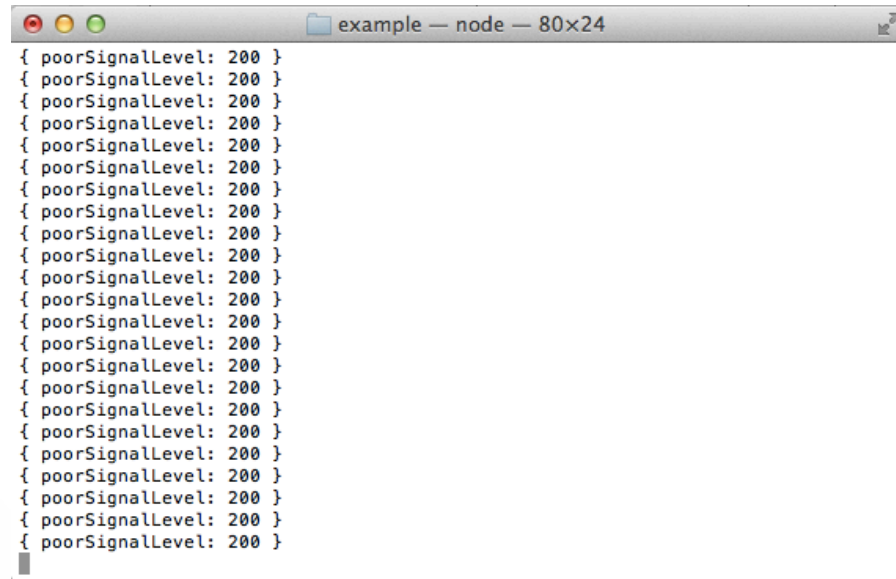
In the app.js example, Node.js is prepared to handle the incoming JSON data feed.

```
client.write(JSON.stringify(self.config));
```

JSON: Javascript

Step 3 (continued):

If you are running the app.js and have made a successful connection, the output of the headset will be “poorSignalLevel: 200”

A screenshot of a terminal window titled "example — node — 80x24". The window displays a list of 20 identical JSON objects, each containing the key "poorSignalLevel" with the value 200. The objects are enclosed in curly braces and separated by newlines.

```
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }  
{ poorSignalLevel: 200 }
```

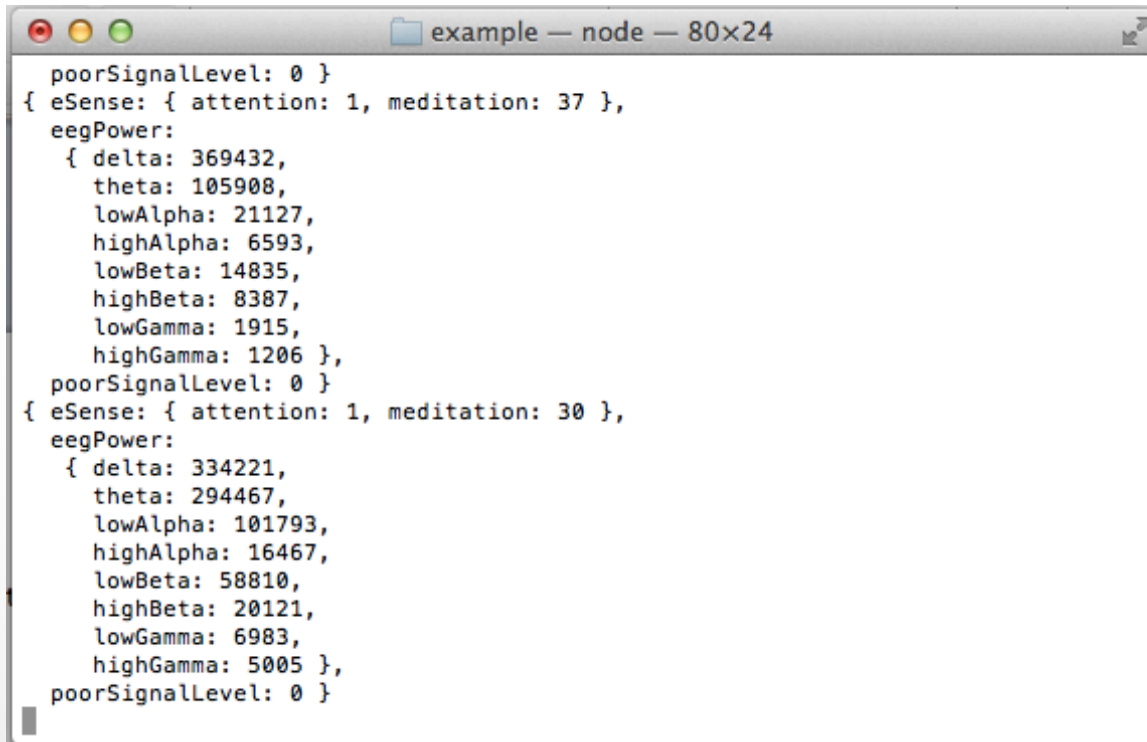
PoorSignalLevel is a measurement that we use to determine if someone is properly wearing a headset and brainwave data is being fed through.

At this point, put on the headset.

poorSignalLevel

Step 3. (Continued)

Here is the outputted JSON data. You can focus in on the Attention, Meditation, and eyeBlink data to build your applications



```
poorSignalLevel: 0 }
{ eSense: { attention: 1, meditation: 37 },
  eegPower:
    { delta: 369432,
      theta: 105908,
      lowAlpha: 21127,
      highAlpha: 6593,
      lowBeta: 14835,
      highBeta: 8387,
      lowGamma: 1915,
      highGamma: 1206 },
  poorSignalLevel: 0 }
{ eSense: { attention: 1, meditation: 30 },
  eegPower:
    { delta: 334221,
      theta: 294467,
      lowAlpha: 101793,
      highAlpha: 16467,
      lowBeta: 58810,
      highBeta: 20121,
      lowGamma: 6983,
      highGamma: 5005 },
  poorSignalLevel: 0 }
```

Parsed JSON data

Johnny Liu
Director
Developer Program & eCommerce
johnny@neurosky.com
408-368-5596

<http://developer.neurosky.com>

Thank You



GameMaker: Studio

GameMaker: Studio

GameMaker

Inside of GameMaker, in order to access the ThinkGear Connector, you must:

1. Open up the socket
2. Relays the App Name / App Key
3. Receives the JSON datafeed

Step 1.

Host address: 127.0.0.1

Port: 13854

Protocol: TCP

You will see the ThinkGear Connector (TGC) icon turn into an hourglass when it attempts to connect. The TGC is scanning different COM ports.



Socket to me

Step 1. (continued)

http://docs.yoyogames.com/source/dadiospice/002_reference/networking/network_connect.html

In GameMaker, there are Rooms, Objects, Events, etc. One way to setup the initial ThinkGear Connector connection is to:

1. Create an object. (objConnect)
2. Add the following Event
 - Create
 - Add the following Action: “Execute a piece of code.”
3. Use the following code to create your socket.

```
client = network_create_socket(network_socket_tcp);  
network_connect(client, "127.0.0.1", 13854);
```

4. Take the newly created object and add it to your first "room". For example, your splash screen.

Setting up the Socket

Step 2.

Next, you need to handshake with the ThinkGear Connector in order to enable the feed of brainwave information.

ThinkGear Connector will receive an appName and an appKey.

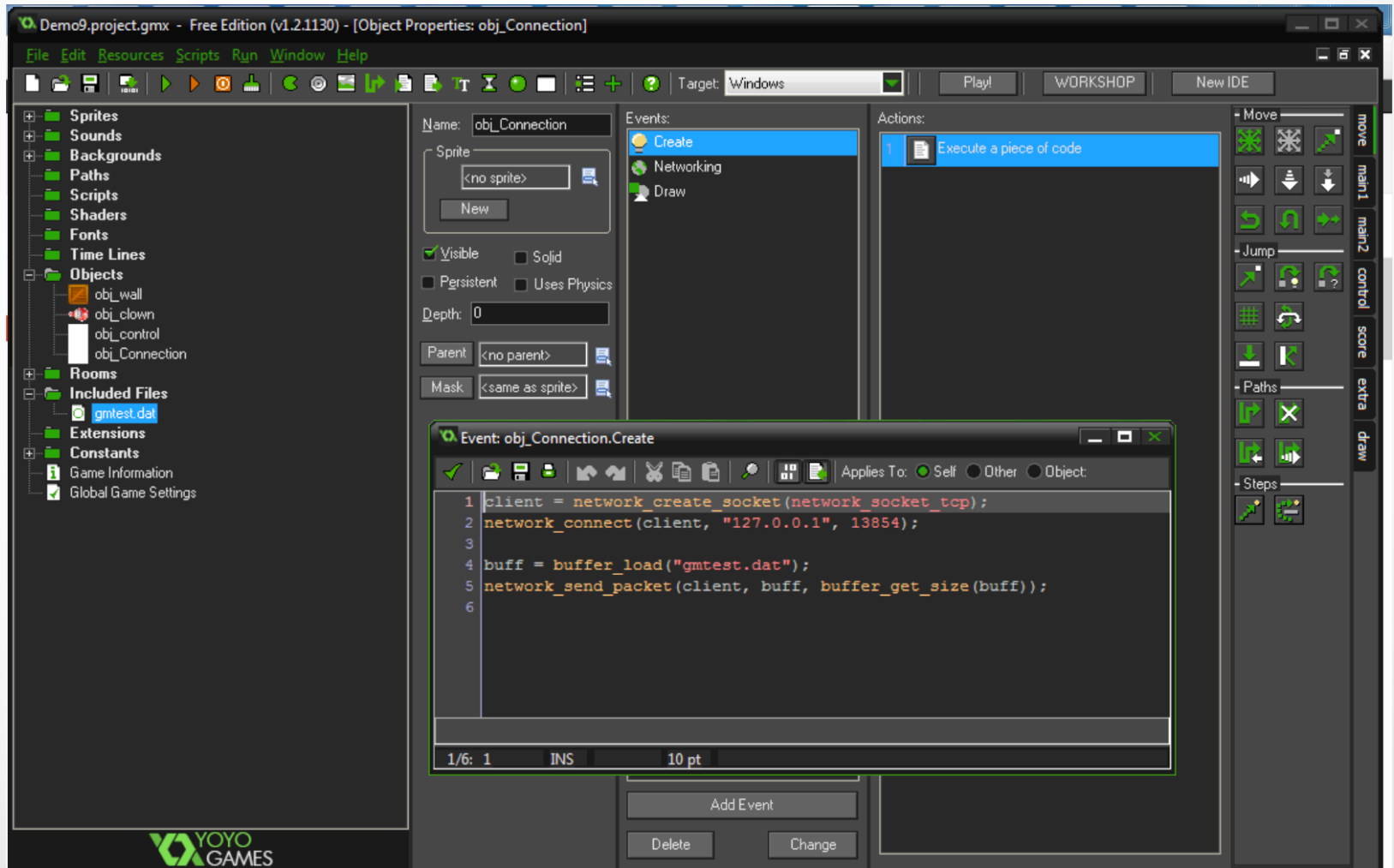
1. In the context of GameMaker, I created a small DAT file with the following information.

```
{"appName": "Test", "appKey": "9f54141b4b4c567c558d3a76cb8d715cbde03096"}
```

2. After the Socket connection in the objConnect code, that DAT file is then sent onto the same socket server.

```
buff = buffer_load("gmtest.DAT");  
network_send_packet(client, buff, buffer_get_size(buff));
```

appName and appKey





Step 3.

Javascript Object Notation: a lightweight data interchange format that is easy for humans to read and write, and easy for machines to parse and generate.

You will need to use JSON to parse the information from the socket. The GameMaker is supposed to support JSON through *json_decode* and *json_encode*.

The free version of GameMaker apparently has some limitations regarding script handling.

JSON and GameMaker

A.

<http://help.yoyogames.com/entries/25891363-Networking-Overview>

B.

<http://gmc.yoyogames.com/index.php?showtopic=565659>

C.

And a blog post about Game Maker's programming language and it's ability to handle JSON

<http://blog.tangrs.id.au/?p=492>

Hopefully, in time for the Hackathon, we can collaboratively figure out a solution for Step 3.

JSON: Links

Johnny Liu
Director
Developer Program & eCommerce
johnny@neurosky.com
408-368-5596

<http://developer.neurosky.com>

Thank You