



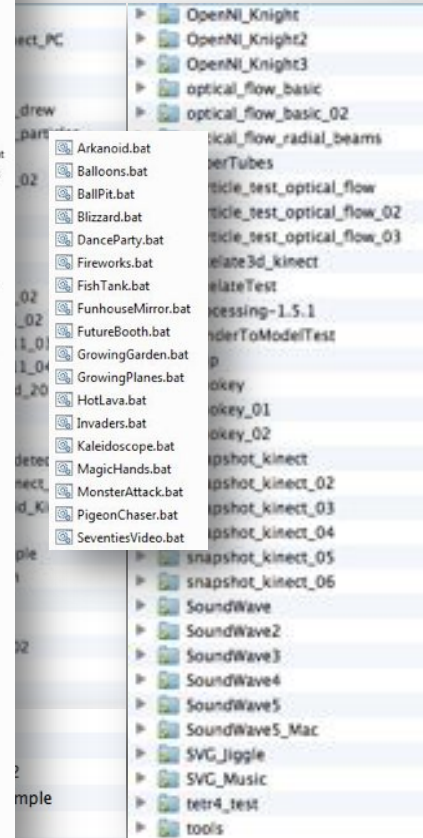
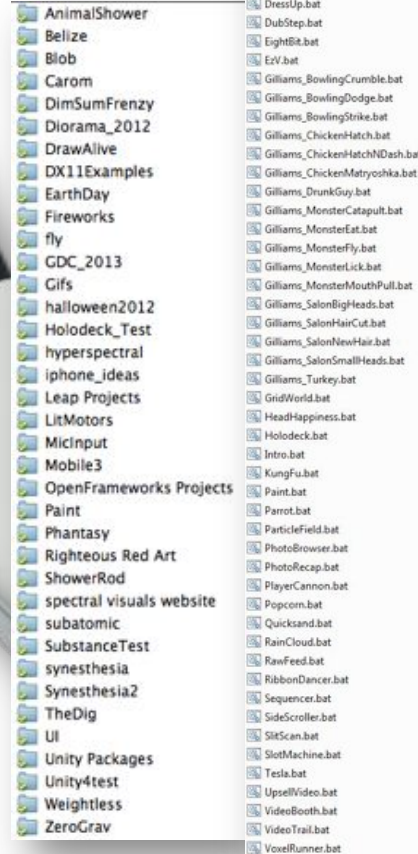
**Rapid Prototyping
at
Double Fine**

Introductions

- Hi, I'm Drew Skillman
- Project Lead: Kinect Party
- Background in Tech Art / VFX



My Motivation for the Talk



Rapid Prototyping Feels Different

- Don't worry about doing things "Right"
- No Expectations
- No Pressure
- *Cool things still happen!*





Prototyping Payoffs

- ✓ Build confidence in new ideas, and share them
- ✓ Learn engine strengths and limitations
- ✓ Learn input strengths and limitations
- ✓ Fail fast
- ✓ Sign projects
- ✓ Happy accidents

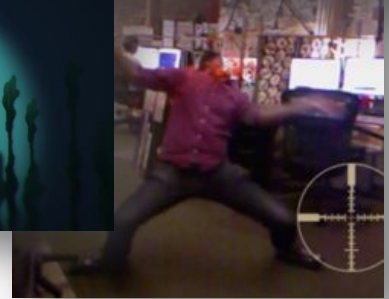
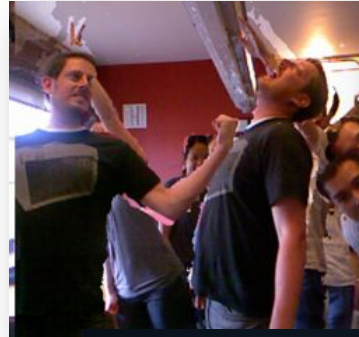


Technical Artists

- Fantastic Background for Rapid Prototyping
- Fast iteration on art, code, and design



Prototyping **VS** Rapid Prototyping



Talk Overview

Prototyping Platforms

```
RCBDepthTest | Processing 1.5.1
RCBDepthTest $
import org.openkinect.*;
import org.openkinect.processing.*;

Kinect kinect;
boolean depth = true;
boolean rgb = false;
boolean ir = false;

float deg = 15; // Start at 15 degrees

void setup() {
  size(1280, 800);
  kinect = new Kinect(this);
  kinect.start();
  kinect.enableDepth(depth);
  kinect.enableRGB(rgb);
  kinect.enableIR(ir);
  kinect.setDeg(deg);
}

void draw() {
  background(0);

  image(kinect.getVideoImage(), 0, 0);
  image(kinect.getDepthImage(), 640, 0);
  fill(255);
  text("RGB/IR FPS: " + (int) kinect.getIRFPS() + "
  text("Depth FPS: " + (int) kinect.getDepthFPS(), 640, 495);

  Camera tilt = (int) deg * 'degrees';
}

void stop() {
  kinect.stop();
  super.stop();
}
36
```



```
c:\dfp-ez6\Ez6\Workflow>make_new_activity.py -n my_prototype --player
Syncing code directories...
Added Code\DFEz\Inc\EzActivities\my_prototype.h
Updated Code\DFEz\Src\Common.cpp
Added Code\DFEz\Src\EzActivities\my_prototype.cpp
Identical Data\Config\DebugMenuTable.lua
Added Data\Prototypes\Activities\my_prototype.proto
Added Data\Script\EzActivities\my_prototype.lua
Added Unmugged\Gameplay\EzActivities\my_prototype.EzAct
Identical Unmugged\Gameplay\EzRules.EzRIs
Added Workflow\Launchers\my_prototype.bat
```

Examples from Kinect Party



Prototyping for Next Gen



Prototyping for New Input Devices



Links!



Prototyping Platforms

```
RCBDepthTest | Processing 1.5.1
RCBDepthTest $
import org.openkinect.*;
import org.openkinect.processing.*;

Kinect kinect;
boolean depth = true;
boolean rgb = false;
boolean ir = false;

float deg = 15; // Start at 15 degrees

void setup() {
  size(1280, 800);
  kinect = new Kinect(this);
  kinect.start();
  kinect.enableDepth(depth);
  kinect.enableRGB(rgb);
  kinect.enableIR(ir);
  kinect.setDeg(deg);
}

void draw() {
  background(0);

  image(kinect.getVideoImage(), 0, 0);
  image(kinect.getDepthImage(), 640, 0);
  fill(255);
  text("RGB/IR FPS: " + (int) kinect.getIRFPS() + "
  text("Depth FPS: " + (int) kinect.getDepthFPS(), 640, 495);

  Camera tilt = (int) deg * 'degrees';
}

void stop() {
  kinect.stop();
  super.stop();
}

```



```
c:\dfp-ez6\Ez6\Workflow>make_new_activity.py -n my_prototype --player
Syncing code directories...
Added Code\DFEz\Inc\EzActivities\my_prototype.h
Updated Code\DFEz\Src\Common.cpp
Added Code\DFEz\Src\EzActivities\my_prototype.cpp
Identical Data\Config\DebugMenuTable.lua
Added Data\Prototypes\Activities\my_prototype.proto
Added Data\Script\EzActivities\my_prototype.lua
Added Unmugged\Gameplay\EzActivities\my_prototype.EzAct
Identical Unmugged\Gameplay\EzRules.EzRIs
Added Workflow\Launchers\my_prototype.bat

```

Processing



- Java Programming Environment
- Cross platform (Mac/PC/Linux)
- (Also Android)
- Great for 2D!

A screenshot of a Processing IDE window titled "RGBDepthTest | Processing 1.5.1". The window shows a code editor with the following Java code:

```
import org.openkinect.*;
import org.openkinect.processing.*;

Kinect kinect;
boolean depth = true;
boolean rgb = false;
boolean ir = false;

float deg = 15; // Start at 15 degrees

void setup() {
  size(1200,600);
  kinect = new Kinect(this);
  kinect.start();
  kinect.enableDepth(depth);
  kinect.enableRGB(rgb);
  kinect.enableIR(ir);
  kinect.tilt(deg);
}

void draw() {
  background(0);

  image(kinect.getVideoImage(),0,0);
  image(kinect.getDepthImage(),640,0);
  fill(255);
  text("Tilt: IR FPS: " + (int) kinect.getVideoFPS() + " Camera tilt: " + (int) deg + " degrees",
  text("DEPTH FPS: " + (int) kinect.getDepthFPS(),640,496);
}

void stop() {
  kinect.quit();
  super.stop();
}
```


Processing

```
RGBDepthTest | Processing 1.5.1

import org.openkinect.*;
import org.openkinect.processing.*;

Kinect kinect;
boolean depth = true;
boolean rgb = false;
boolean ir = false;

float deg = 15; // Start at 15 degrees

void setup() {
  size(1200,520);
  kinect = new Kinect(this);
  kinect.start();
  kinect.enableDepth(depth);
  kinect.enableRGB(rgb);
  kinect.enableIR(ir);
  kinect.tilt(deg);
}

void draw() {
  background(0);

  image(kinect.getVideoImage(),0,0);
  image(kinect.getDepthImage(),640,0);
  fill(255);
  text("RGB/IR FPS: " + (int) kinect.getVideoFPS() + " Camera tilt: " + (int)deg + " degrees");
  text("DEPTH FPS: " + (int) kinect.getDepthFPS(),640,495);
}

void stop() {
  kinect.quit();
  super.stop();
}
```



Processing

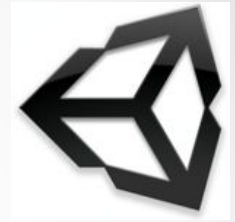


Unity



UNITYGUI2.0 SPACE COMMANDER DEMO - MADE IN A DAY FROM SCRATCH

Unity



- C# and Javascript
- High level 3D support including OpenGL ES 2.0, DirectX 11, etc...
- Massive Community
- Great for 3D!



Double Fine's Buddha Engine

- Proprietary C++/Lua based engine
- Developed for Brutal Legend
- Used for Kinect Party and Happy Action Theater



Rapid Prototyping

```
RGBDepthTest | Processing 1.5.1
RGBDepthTest $
import org.openkinect.*;
import org.openkinect.processing.*;

Kinect kinect;
boolean depth = true;
boolean rgb = false;
boolean ir = false;

float deg = 15; // Start at 15 degrees

void setup() {
  size(1280, 800);
  kinect = new Kinect(this);
  kinect.start();
  kinect.enableDepth(depth);
  kinect.enableRGB(rgb);
  kinect.enableIR(ir);
  kinect.setDeg(deg);
}

void draw() {
  background(0);

  image(kinect.getVideoImage(), 0, 0);
  image(kinect.getDepthImage(), 640, 0);
  fill(255);
  text("RGB/IR FPS: " + (int) kinect.getIrFps() + "
  text("Depth FPS: " + (int) kinect.getDepthFps(), 640, 495);

  Camera tilt = (int) deg * 'degrees';
}

void stop() {
  kinect.stop();
  super.stop();
}
```



```
c:\dfp-ez6\Ez6\Workflow>make_new_activity.py -n my_prototype --player
Syncing code directories...
Added Code\DFEz\Inc\EzActivities\my_prototype.h
Updated Code\DFEz\Src\Common.cpp
Added Code\DFEz\Src\EzActivities\my_prototype.cpp
Identical Data\Config\DebugMenuTable.lua
Added Data\Prototypes\Activities\my_prototype.proto
Added Data\Script\EzActivities\my_prototype.lua
Added Unmugged\Gameplay\EzActivities\my_prototype.EzAct
Identical Unmugged\Gameplay\EzRules.EzRIs
Added Workflow\Launchers\my_prototype.bat
```

Rapid Prototyping



```
RGBDepthTest $
import org.opencv.inert.*;
import org.opencv.inert.processing.*;

Kinect kinect;
boolean depth = true;
boolean rgb = false;
boolean ir = false;

float deg = 35; // Start

void setup() {
  size(1280,520);
  kinect = new Kinect();
  kinect.start(1);
  kinect.enableDepth();
  kinect.enableRGB();
  kinect.enableIR();
  kinect.start();
}

void draw() {
  backg...

  Camera tilt: * = (int)deg * degree
  ct.getDepthFPS(),440,495);
}
```



```
c:\dfp-ez6\Ez6\Workflow>make_new_activity.py -n my_prototype --player
Syncing code directories...
Added Code/DFEz/Inc/EzActivities/my_prototype.h
Updated Code/DFEz/Src/Common.cpp
Added Code/DFEz/Src/EzActivities/my_prototype.cpp
Identical Data/Config/DebugMenuTable.lua
Added Data/Prototypes/Activities/my_prototype.proto
Added Data/Script/EzActivities/my_prototype.lua
Added Unmunged/Gameplay/EzActivities/my_prototype.EzAct
Identical Unmunged/Gameplay/EzRules.EzR1s
Added Workflow/Launchers/my_prototype.bat
```

Rapid Prototyping

- Minimize Compile Times
 - Maximize iteration

```
RGBDepthTest $
import org.open.jreact.*
import org.open.jreact_processing.*

Kinect kinect;
float kinect_depth = 0;
boolean rgb = false;
boolean ir = false;

float deg = 15; // Start at 35 degrees

void setup() {
  size(1200,600);
  kinect = new Kinect(0,0,0);
  kinect_start();
  kinect_enableDepth(depth);
  kinect_enableRGB(rgb);
  kinect_enableIR(ir);
  kinect_start(deg);
}

void draw() {
  background(0);

  image(kinect.getVideoImage(),0,0);
  image(kinect.getDepthImage(),640,0);
  fill(255);
  text("VIDEO FPS: " + (int) kinect.getVideoFPS() + " Camera tilt: " + (int)deg + " degrees",
  text("DEPTH FPS: " + (int) kinect.getDepthFPS(),640,495);
}

void stop() {
  kinect_stop();
  image_stop();
}
```



Rapid Prototyping

- **Minimize Compile Times**
 - Maximize iteration
- **Use Building Blocks**
 - Don't ever start from scratch if you can help it

Building Blocks

- Processing: Libraries
 - <http://processing.org/reference/libraries/>

Video
Read images from a camera, play movie files, and create movies.

Networks
Send and receive data over the Internet through simple clients and servers.

Serial
Send data between Processing and external hardware through serial communication (RS-232).

Simulation / Math

- **Physics**
by Jeffrey Traer Bernstein
Simple particle system physics engine. No collisions, just particles, springs, gravity & drag.
- **MSAFluid**
by Hiroshi Akiyoshi
A library for solving real-time fluid dynamics simulations.
- **AI Libraries**
by Aaron Stead
A set of libraries to assist with artificial programming tasks such as genetic algorithms and the AStar algorithm.
- **proovv**
by Adria Falcas
A collection of utilities for performing some statistical and matrix-related manipulations.
- **PDF Export**
Create PDF files. These vector graphics files can be scaled to any size and printed at high resolutions.
- **DXF Export**
Create DXF files to save geometry for loading into other programs. It works with triangle-based graphics including polygons, boxes, and spheres.
- **Minim**
Uses JavaSound to provide an easy-to-use audio library while still providing flexibility for more advanced users.
- **Arduino**
Directly control an Arduino board through Processing.
- **Netscape JavaScript**
Methods for interfacing between Javascript and programs exported from Processing.
- **MatrixMath**
by Francis Bondi
Helpful code for matrix operations.
- **Cellular Automata**
by Francis Bondi
Simplifies making cellular automata calculations.
- **BoxWrap2D**
by esjordan
BoxWrap2D runs on top of JBox2D enabling simple integration of JBox2D with Processing.
- **Combinatorics**
by Florian Jenett
Generate combinations, variations and permutations.
- **Cell Noise**
by Carl-Johan Rosén
Explores cell noise (Worley noise), a pattern generation algorithms useful for animation.
- **ELite**
by Andrea Coladri
Implementation of the classes A.I. bot, Eliza.
- **LSystem Utilities**
by Martin Froud
A library for exploring and creating Lindenmayer Systems in 2D and 3D.
- **Flake**
by Edward Hacker Flood
A wrapper for JBox2D, a 2D physics engine.

Computer Vision / Video

- **GSVideo**
by Andrea Coladri
Uses GStreamer as an alternative to QuickTime for movie playback and camera capture.
- **BlobDetection**
by x3ss
Performs the computer vision technique of finding "blobs" in an image.
- **OpenCV**
by Stéphane Casseot and Douglas Edic Stanley
An OpenCV implementation for processing including blob detection, face recognition and more. This library is highly recommended.
- **IntegralHistogram**
by Giovanni Tarabucci and Alessandro Badocci
The integral histogram method allows to obtain the color or intensity histogram of all possible target regions in a Cartesian data space.
- **Fish**
by André Sier
A fast multi-blob detector and tracker using flood-fill algorithms
- **blobscanner**
by Antonio Baldoni
A library for blob detection and analysis in image and video streams.
- **imovideo**
by Andrea Farber
A simple wrapper for playing videos and grabbing video data for any of the formats that the JMC library supports.
- **Face Detect (PC)**
by Bryan Chubb
Face detection library made with WebCamXtra and the openCV framework. PC only.
- **TUIO Client**
by Marco Kubitzer
Client library for the simple creation of tangible interactive surfaces, receiving TUIO data from object and multi-touch trackers such as [reactiVision](#).
- **P-SURF**
by Claudio Parraqui and Alessandro Martini
An implementation of the SURF (Speeded Up Robust Features) feature detector to search for discrete image correspondences.
- **openkinect**
by Daniel Shiffman
A kinect implementation for Processing.
- **simale-openssl**
by Max Khoner
A simple OpenNI and NITE wrapper for Processing.
- **IMaran (WebCamXtra)**
by Josh Nimetz et al.
Camera library for motion detection, color tracking, blob distinction, and pixel addressing. Does not require QuickTime or WinVidG for Windows machines.
- **libCV**
by toxi
Grabs video frames from a camera using the Java Media Framework (JMF). Does not require QuickTime or WinVidG for Windows machines.
- **tuoZones**
by jLust
tuoZones provides a way to set zones within a multi-touch screen to respond to TUIO messages sent from a tracking application.
- **CoModel**
by Federico Bartoli
CoModel is a library for motion detection based on background modeling and subtraction.
- **libjs**
by Thomas Diebold
This kinect library is based on the [libfreenect-software](#), it currently only works on windows.

Building Blocks

- **Unity:** Unity Packages
 - Asset Store



Building Blocks

- **Buddha Engine: Perforce!**
 - Every Double Fine game is a possible building block for art assets or code



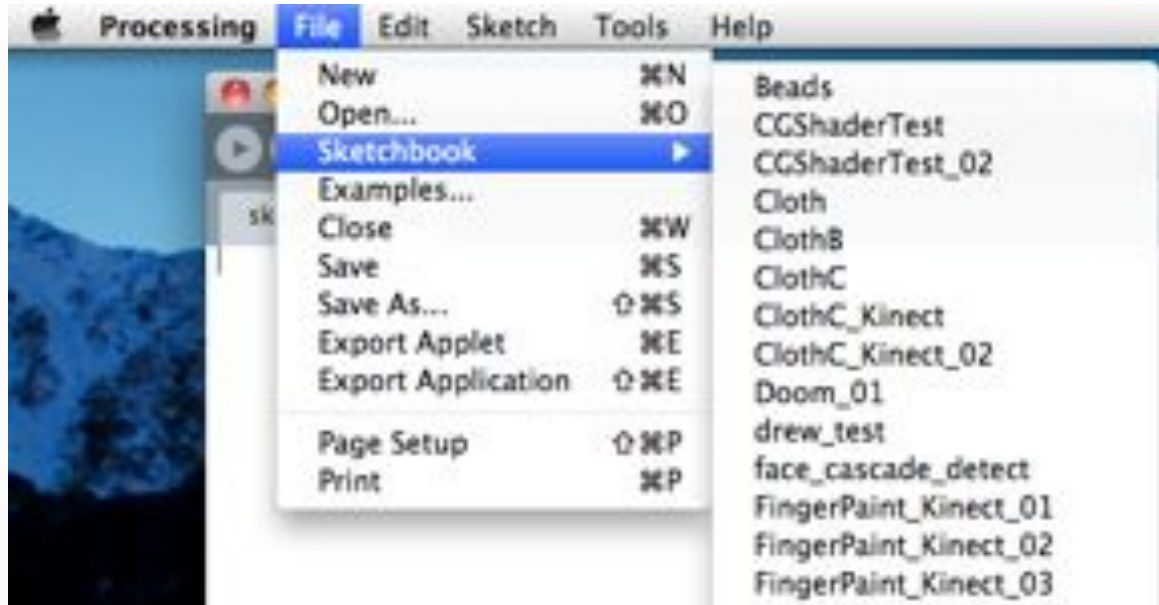
Rapid Prototyping

- **Minimize Compile Times**
 - Maximize iteration
- **Use Building Blocks**
 - Don't ever start from scratch if you can help it
- **Sketchbook Approach**
 - Make it easy to start, easy to branch, easy to experiment

Sketchbooks



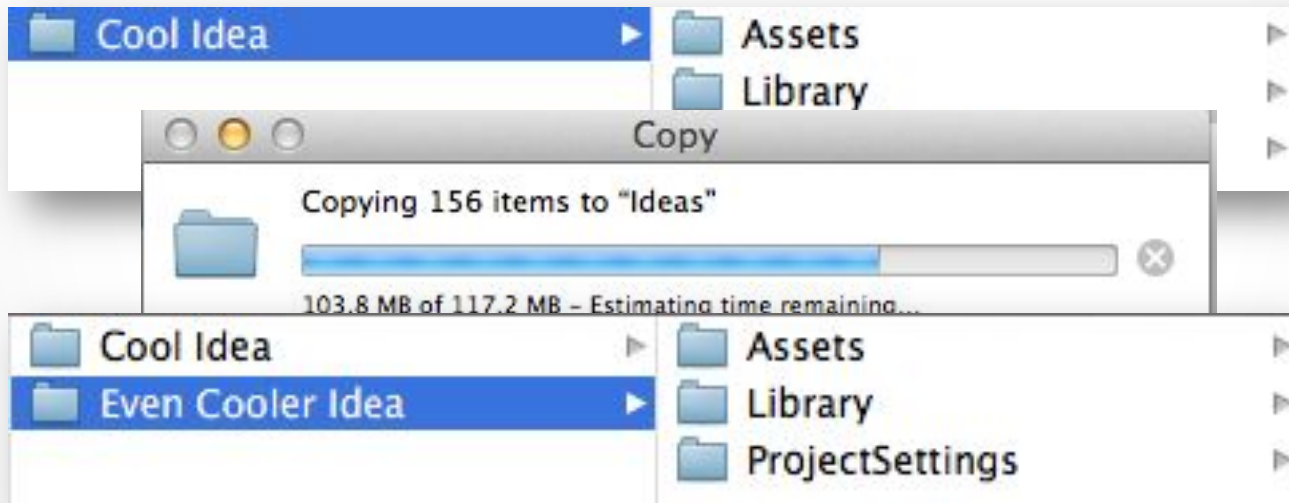
- Processing:



Sketchbooks



- **Unity:** Branching an Idea
 - Every asset path is relative, just duplicate project folders



Sketchbooks

- Buddha Engine:



Sketchbooks

- Buddha Engine: [make_new_activity.py](#)



```
c:\dfp-ez6\Ez6\Workflow>make_new_activity.py -n my_prototype --player
Syncing code directories...
Added      Code/DFEz/Inc/EzActivities/my_prototype.h
Updated    Code/DFEz/Src/Common.cpp
Added      Code/DFEz/Src/EzActivities/my_prototype.cpp
Identical  Data/Config/DebugMenuTable.lua
Added      Data/Prototypes/Activities/my_prototype.proto
Added      Data/Script/EzActivities/my_prototype.lua
Added      Unmunged/Gameplay/EzActivities/my_prototype.EzAct
Identical  Unmunged/Gameplay/EzRules.EzRls
Added      Workflow/Launchers/my_prototype.bat
```



Rapid Prototyping

- **Minimize Compile Times**
 - They kill creativity dead
- **Use Building Blocks**
 - Don't ever start from scratch if you can help it
- **Sketchbook Approach**
 - Make it easy to start, easy to branch, easy to experiment

Examples from Kinect Party



Processing Prototype: Depth Freeze



Processing Prototype: Depth Freeze

- We had fun with this



Shipping Version: Future Booth



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents

Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations**
- Fail fast
- Sign projects
- Happy accidents



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations**
- Fail fast
- Sign projects**
- Happy accidents



Processing Prototype: Fire

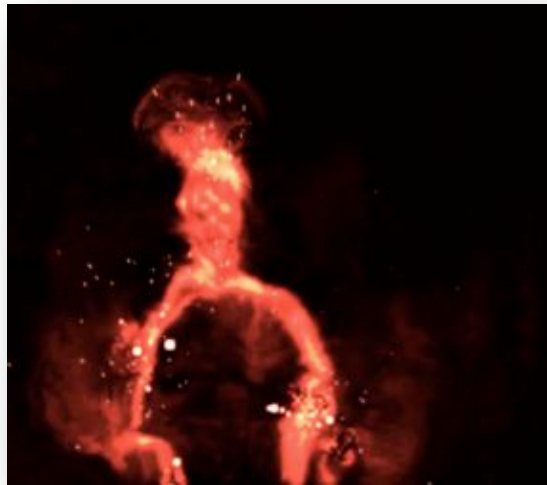


Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents

Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations**
- Fail fast
- Sign projects
- Happy accidents



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations**
- Fail fast
- Sign projects**
- Happy accidents



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations**
- Fail fast
- Sign projects**
- Happy accidents**



GLGraphics Library

- Revealed the potential for full screen shaders

```
vec2 center = vec2(0.5, 0.5);  
vec2 rot_coord = center + rotation * (brush_coord - center);  
  
float brush_alpha = texture2D(tex_unit brush, rot_coord).a;  
vec3 image_color = texture2DRect(tex_unit color, color_coord).rgb;
```

GLGraphics

[About](#) \ [Download](#) \ [Installation](#) \ [Examples](#) \ [Reference](#)

GLGraphics Library

- Define some textures, render targets, and full screen shaders

```
// Textures
GLTexture gradientTex;
GLTexture videoTex;
GLTexture fluidTex;

// Render Targets
GLGraphicsOffScreen gl_buffer_scratch;
GLGraphicsOffScreen gl_buffer_occul;
GLGraphicsOffScreen gl_buffer_finalcolor;

// Full Screen Shaders
GLTextureFilter blurFilter;
GLTextureFilter rewrapFilter;
GLTextureFilter combineFilter;
GLTextureFilter compFilter;
GLTextureFilter copyFilter;
GLTextureFilter diaFilter;
```


GLGraphics Library

- Initialize Them

```
void initTextures()
{
    gradientTex = new GLTexture(this, "grad.png");
    gl_buffer_scratch = new GLGraphicsOffScreen(this, width, height);
    gl_buffer_accum = new GLGraphicsOffScreen(this, width, height);
    gl_buffer_finalcolor = new GLGraphicsOffScreen(this, width, height);
    videoTex = new GLTexture(this, width, height);
    fluidTex = new GLTexture(this, width, height);
    videoTex.loadPixels();
    fluidTex.loadPixels();
}
```

GLGraphics Library

- Initialize Them

```
void initFilters()
{
    blurFilter = new GLTextureFilter(Texture, "Blur.xml");
    blurFilter.setBlendMode(ADD);

    copyFilter = new GLTextureFilter(Texture, "CopyImage.xml");
    copyFilter.setBlendMode(REPLACE);

    disFilter = new GLTextureFilter(Texture, "DisImage.xml");
    disFilter.setBlendMode(REPLACE);

    rewapFilter = new GLTextureFilter(Texture, "RwapImage.xml");
    rewapFilter.setBlendMode(ADD);

    combineFilter = new GLTextureFilter(Texture, "CombineImage.xml");
    combineFilter.setBlendMode(ADD);

    compFilter = new GLTextureFilter(Texture, "CompImage.xml");
    compFilter.setBlendMode(SCREEN);
}
```

GLGraphics Library

- Apply Shaders to Textures (like a video feed)

```
// add fluid to the accumulation buffer
copyFilter.setBlendMode(ADD);
copyFilter.apply(fluidTex, gl_buffer_accum.getTextTexture());

// Blur the output of the fluid simulation
copyFilter.setBlendMode(REPLACE);
copyFilter.apply(gl_buffer_accum.getTextTexture(), gl_buffer_scratch.getTextTexture());
gl_buffer_accum.getTextTexture().clear(0, 0, 0, 255);
blurFilter.setParameterValue("width", 3.2);
blurFilter.apply(gl_buffer_scratch.getTextTexture(), gl_buffer_accum.getTextTexture());

// Apply gradient reap effect
GLTexture[] inputTex = { gl_buffer_accum.getTextTexture(), gradientTex};
reapFilter.apply(inputTex, gl_buffer_finalcolor.getTextTexture());

// Draw the render target to the frame buffer
reap(gl_buffer_finalcolor.getTextTexture(), 0,0,width,height);
```

```
make_new_activity.py -n Blizzard
```



make_new_activity.py -n Blizzard



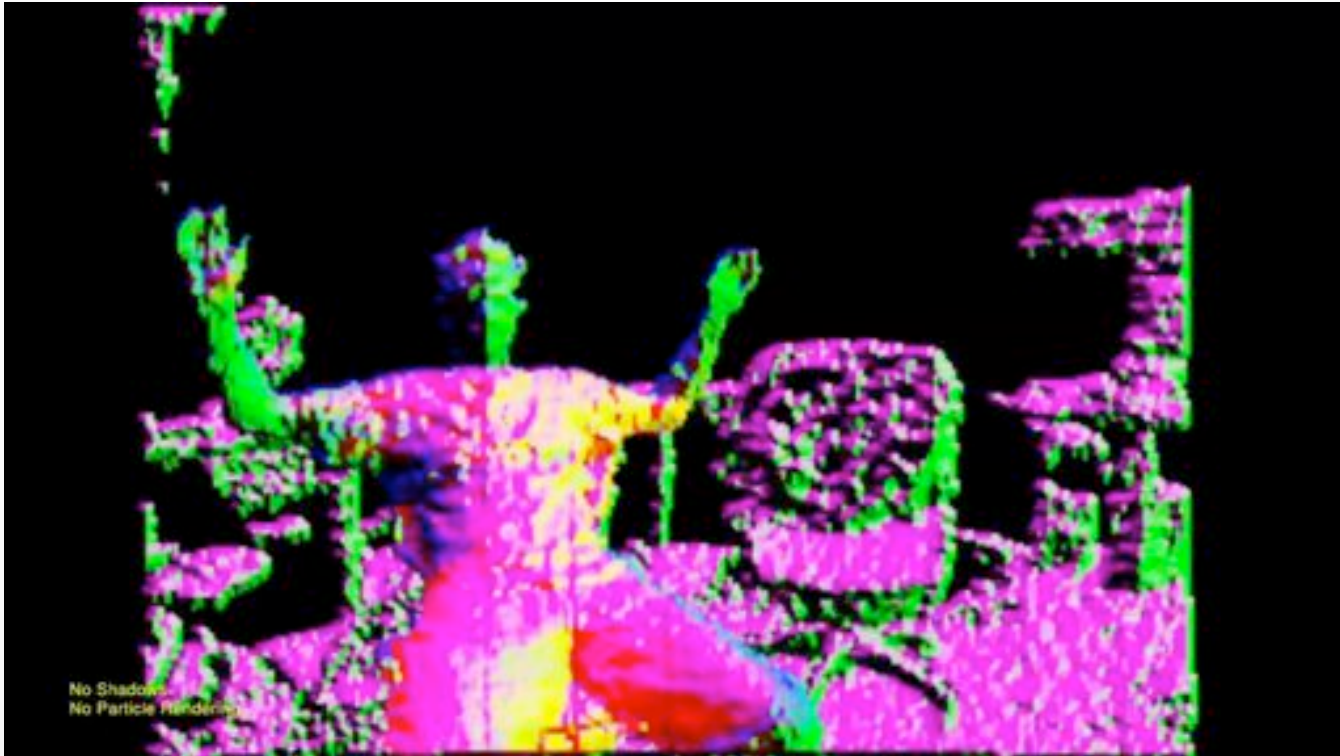
```
make_new_activity.py -n Blizzard
```

- Raw Depth



```
make_new_activity.py -n Blizzard
```

- Sobel filtered depth



```
make_new_activity.py -n Blizzard
```

- Isolating upward facing normals



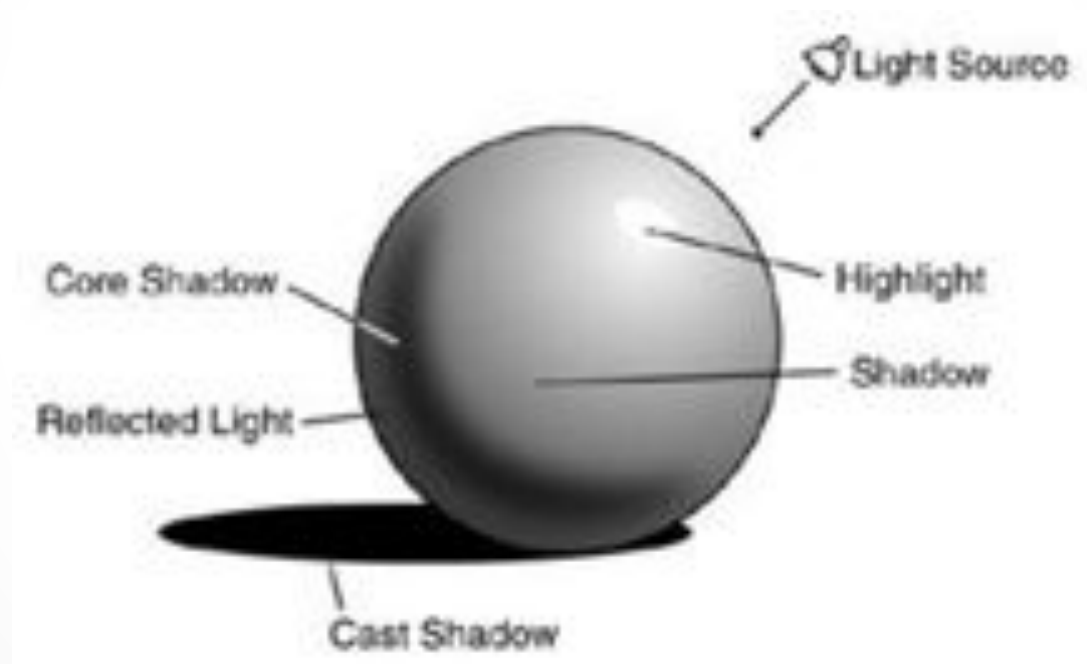

```
make_new_activity.py -n Blizzard
```

- Accumulation



Shipping Version: Blizzard

- It's cool – let's polish it up!



Shipping Version: Blizzard

- Approximate illumination from normals



Shipping Version: Blizzard

- Shift normals up to round out the snow



Shipping Version: Blizzard

- Shift entire snow texture upward



Shipping Version: Blizzard

- Apply post processing



Shipping Version: Blizzard

- Particle Effects



Shipping Version: Blizzard



No Shadows

Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents

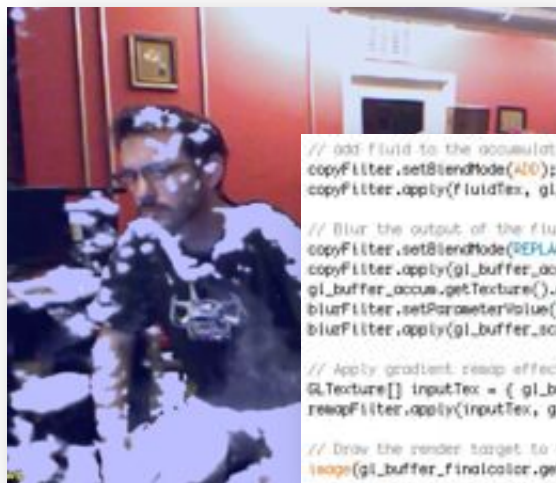
Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them**
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents



Rapid Prototyping Payoffs

- ✓ **Build confidence in new ideas, and share them**
- ✓ **Learn engine strengths and limitations**
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents



```
// add fluid to the accumulation buffer
copyFilter.setBlendMode(ADD);
copyFilter.apply(fluidTex, gl_buffer_accum.getTexture());

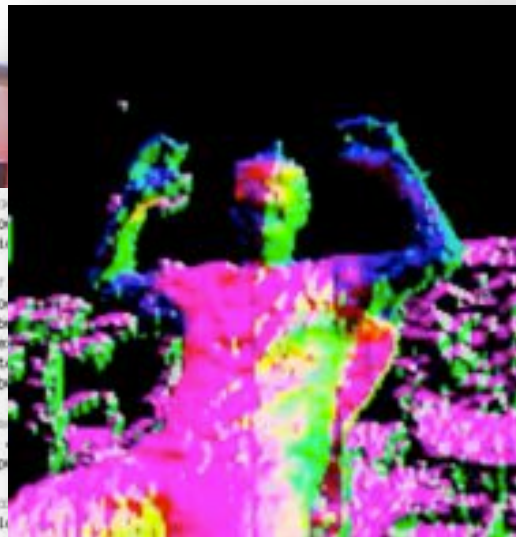
// Blur the output of the fluid simulation
copyFilter.setBlendMode(REPLACE);
copyFilter.apply(gl_buffer_accum.getTexture(), gl_buffer_scratch.getTexture());
gl_buffer_accum.getTexture().clear(0, 0, 0, 255);
blurFilter.setParameterValue("width", 3.2);
blurFilter.apply(gl_buffer_scratch.getTexture(), gl_buffer_accum.getTexture());

// Apply gradient reap effect
[Texture[] inputTex = { gl_buffer_accum.getTexture(), gradientTex};
reapFilter.apply(inputTex, gl_buffer_finalcolor.getTexture());

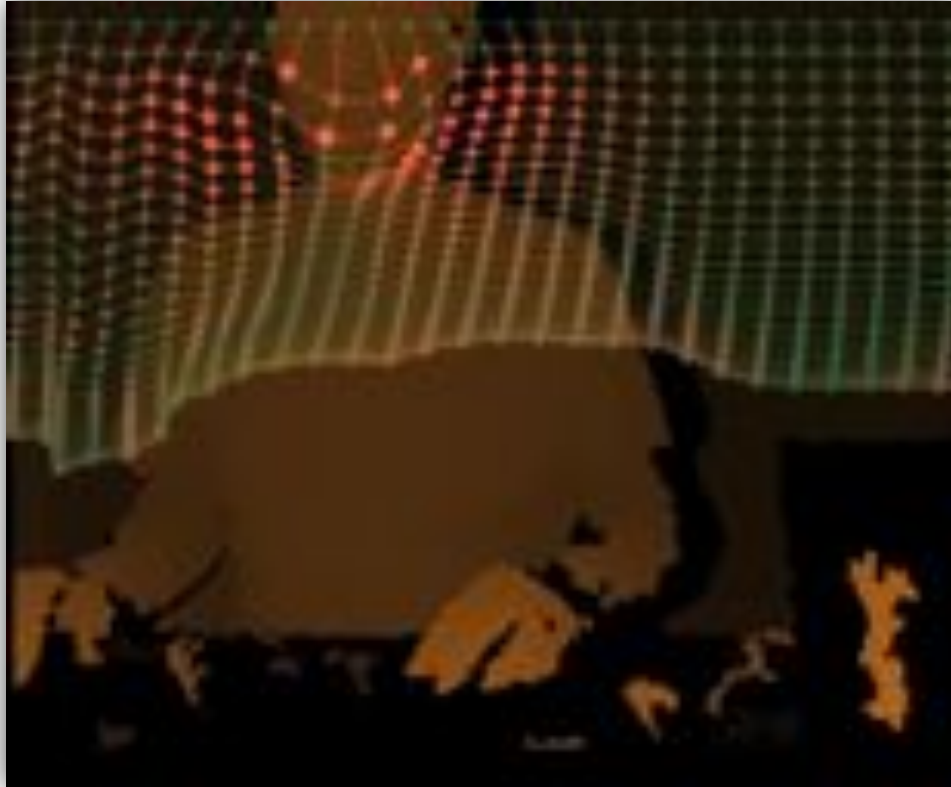
// Draw the render target to the frame buffer]
image(gl_buffer_finalcolor.getTexture(), 0,0,width,height);
```

Rapid Prototyping Payoffs

- ✓ Build confidence in new ideas, and share them
- ✓ Learn engine strengths and limitations
- ✓ Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents



Processing Prototype: Cloth



Motion Blobs



Motion Blobs Example

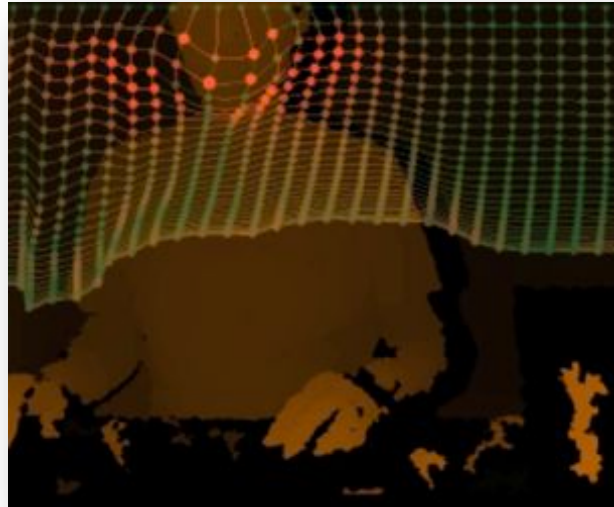


Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents

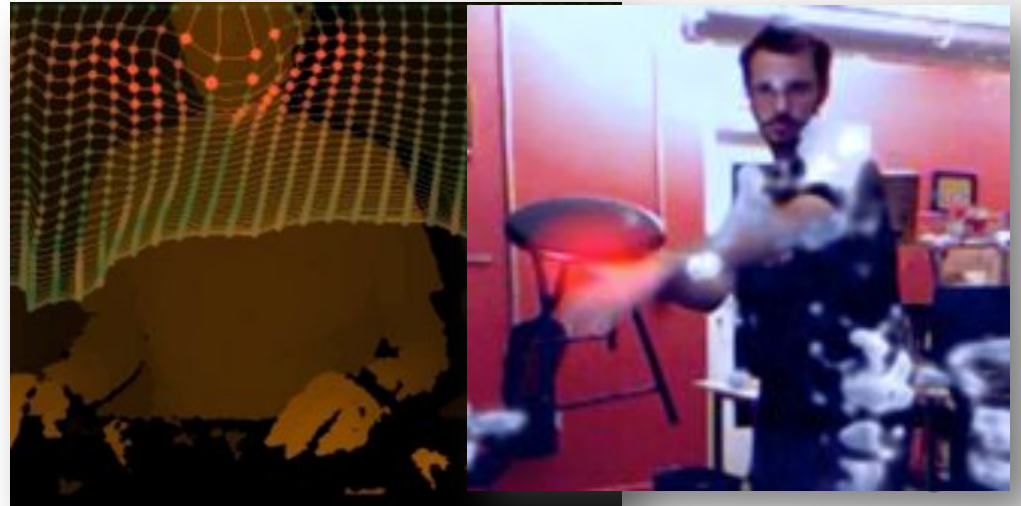
Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast**
- Sign projects
- Happy accidents



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them**
 - Learn engine strengths and limitations
 - Learn input strengths and limitations
- Fail fast**
 - Sign projects
 - Happy accidents

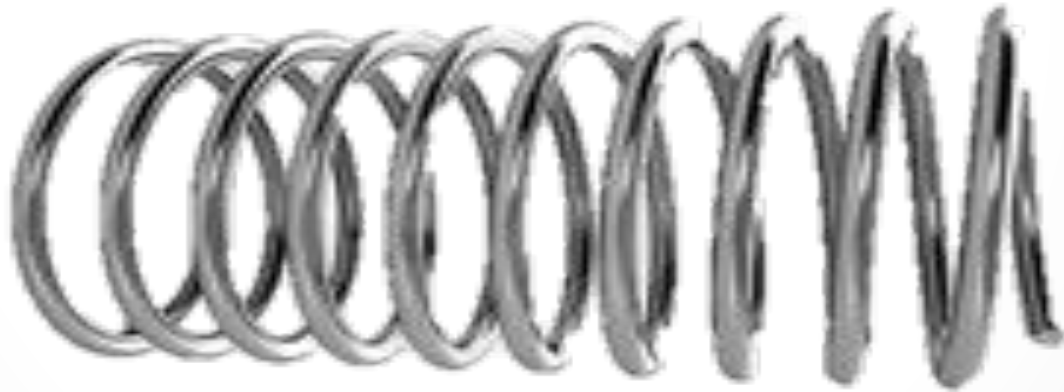


make_new_activity.py -n Funhouse



Physics + Lots of Full Screen Shaders

- Force on a spring: $F = -K \times x$
 - K is "Hookes Constant"
 - x is spring length



Physics + Lots of Full Screen Shaders

- Force on a spring: $F = -K * x$
- In HLSL:

```
// retrieve this frames simulation texture
float4 CurrentFrame =ToWorldUnits(tex2D(g_samSourceA, Tex));
float2 dist = CurrentFrame.xy;
float2 velocity = CurrentFrame.zw;

// hookes law
velocity -= 1 * normalize(dist) * pow(length(dist),2);

// damping
dist = dist * .95;
velocity = velocity * .95;

// Step simulation
dist += velocity * dt;

// output
vCurrentFrame = ToTextureUnits(float4(dist.xy, velocity.xy));
return vCurrentFrame;
```

Displacement Texture



Weak Springs



No Shadows

Strong Springs



Shipping Version: Jello



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents

Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them**
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents



make_new_activity.py -n Forbidden



`make_new_activity.py -n Forbidden`

- Sketch put together entirely by our Character TD...



Shipping Version: Costume Party



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents

Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents**



Kinect Party: Unused Prototypes



Unused for *various* reasons

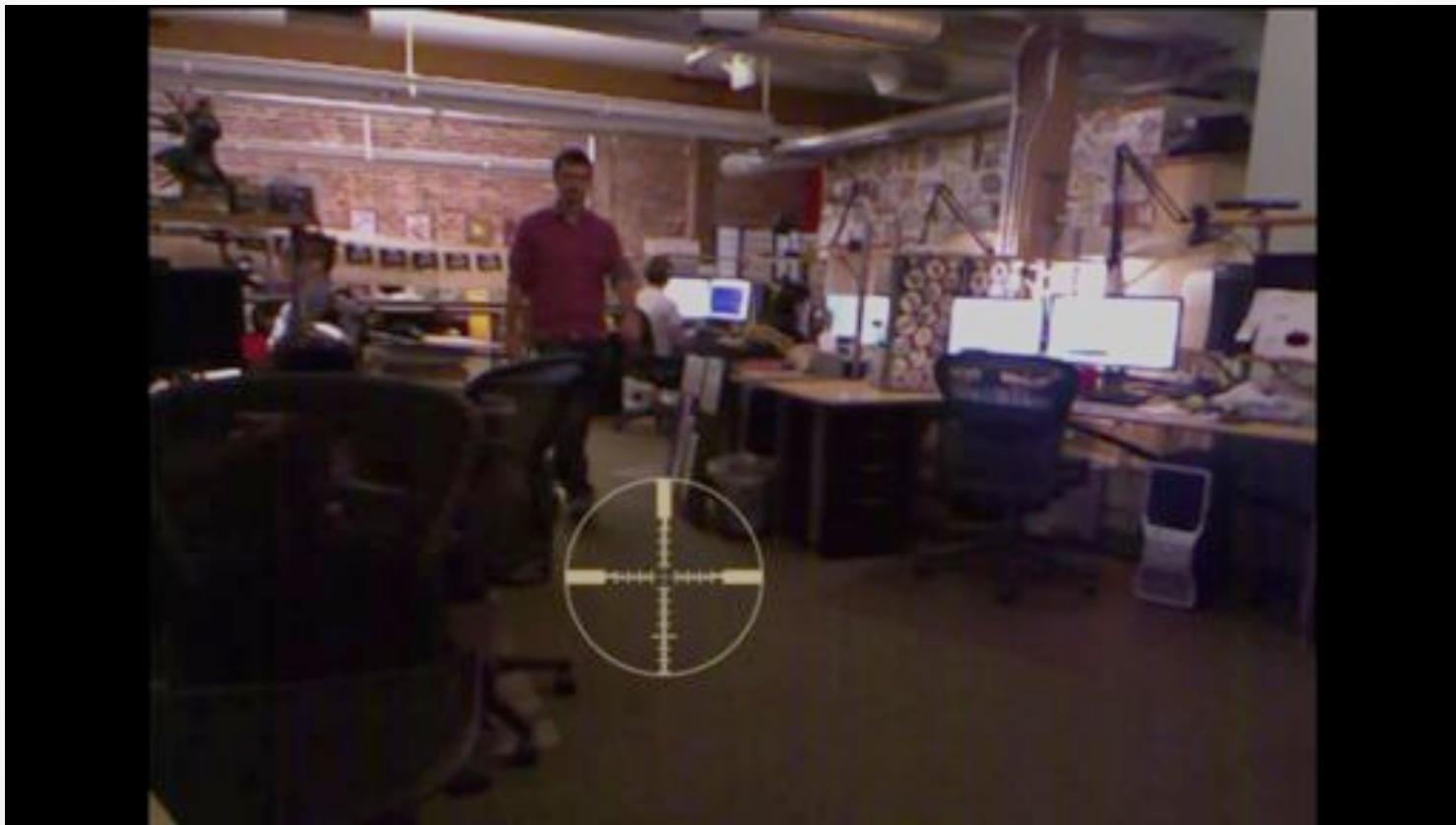


HATERS GONNA HATE

```
make_new_activity.py -n EnergySwords
```



```
make_new_activity.py -n Splosion
```



make_new_activity.py -n Colbert



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents

Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them**
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them**
 - Learn engine strengths and limitations
- Learn input strengths and limitations**
 - Fail fast
 - Sign projects
 - Happy accidents



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them**
 - Learn engine strengths and limitations
- Learn input strengths and limitations**
 - Fail fast
- Sign projects**
 - Happy accidents



Prototyping for Next Gen



Kinect Party 2.0?

- What would we do with the fancy hardware?
- Let's experiment in Unity!



Dub Step



Unity Prototype: Audio Driven Dub Step

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class DubStep : MonoBehaviour {
5
6     public Texture2D[] textureStack;
7     public int writeID = 0;
8     public int readID = 0;
9     public int frameDelay = 0;
10    public Texture2D movieRenderTexture;
11
12    int textureCount = 30;
13    WebCamTexture webcamTexture;
14
15    Color32[] data;
16    bool texturesInitialized = false;
17    int width = 640;
18    int height = 480;
19
20
```

```
21 IEnumerator Start ()
22 {
23     yield return Application.RequestUserAuthorization(UserAuthorization.WebCam | UserAuthorization.Microphone);
24
25     if (Application.HasUserAuthorization(UserAuthorization.WebCam | UserAuthorization.Microphone))
26     {
27         webcamTexture = new WebCamTexture(width,height,30);
28         data = new Color32[width * height];
29         webcamTexture.Play();
30     }
31     textureStack = new Texture2D[textureCount];
32     for (int i = 0; i < textureCount; i++)
33     {
34         textureStack[i] = new Texture2D(width, height);
35     }
36
37     // Update is called once per frame
38     void Update () {
39
40         if (!webcamTexture)
41         {
42             Debug.Log ("Waiting for webcam...");
43             return;
44         }
45
46         UpdateFrameDelay();
47
48         webcamTexture.GetPixels32 (data);
49
50         textureStack[writeID].SetPixels32(data);
51         textureStack[writeID].Apply();
52         readID = (writeID + (textureCount - frameDelay)) % textureCount;
53         renderer.material.mainTexture = textureStack[readID];
54
55         writeID++;
56         writeID = writeID % textureCount;
57
58     }
59
60
61     // The frame delay is currently being driven by the visualization studio
62     void UpdateFrameDelay()
63     {
64         frameDelay = (int)(transform.localScale.y * textureCount);
65     }
66
67 }
```

Unity Prototype: Audio Driven Dub Step



Unity Prototype: Audio Driven Dub Step

- Web Browser can be a great way to share ideas



Unity Prototype: Audio Driven Dub Step

- Prototyping tools make it easy to manipulate data flow



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents

Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them**
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them**
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents**





ST. JOHN PHILLIPS
glitch

*Live in' la Video Loca con
Puerta Para Gráficos
Accelerados Gigante!*

Whin'fast! 3D²⁰⁰⁰

How emulates your surging electronic man-boo? If you want
to buy a videocard that just plain does it make sense
to connect with not one, not two, but ALL FIVE of today's
big boys in all jammed together in some kind of holy,
holy trinity... it shouldn't work... but it does.

Standwith UNIFORM
Like X2 Murder Simulator! Duke X2 The
X2 Duke X2 It's Hammer Time,
simple, light aircraft, & MOA-certified
wing systems.

Compare with all the other leading
cards of the world actually moved
we're confident that if they
would have looked at the
performance should this
card not.

is Don't
is the

Wow!

BungolioMarks

Card	Score
ATI Radeon 7500	~100
NVIDIA GeForce 256	~150
ATI Radeon 7600	~200
NVIDIA GeForce 256 Ultra	~250
ATI Radeon 7700	~300
NVIDIA GeForce 256 Ultra II	~350
ATI Radeon 7800	~400
NVIDIA GeForce 256 Ultra III	~450
ATI Radeon 7900	~500
NVIDIA GeForce 256 Ultra IV	~550
ATI Radeon 8000	~600
NVIDIA GeForce 256 Ultra V	~650
ATI Radeon 8100	~700
NVIDIA GeForce 256 Ultra VI	~750
ATI Radeon 8200	~800
NVIDIA GeForce 256 Ultra VII	~850
ATI Radeon 8300	~900
NVIDIA GeForce 256 Ultra VIII	~950
ATI Radeon 8400	~1000
NVIDIA GeForce 256 Ultra IX	~1050
ATI Radeon 8500	~1100
NVIDIA GeForce 256 Ultra X	~1150
ATI Radeon 8600	~1200
NVIDIA GeForce 256 Ultra XI	~1250
ATI Radeon 8700	~1300
NVIDIA GeForce 256 Ultra XII	~1350
ATI Radeon 8800	~1400
NVIDIA GeForce 256 Ultra XIII	~1450
ATI Radeon 8900	~1500
NVIDIA GeForce 256 Ultra XIV	~1550
ATI Radeon 9000	~1600
NVIDIA GeForce 256 Ultra XV	~1650
ATI Radeon 9100	~1700
NVIDIA GeForce 256 Ultra XVI	~1750
ATI Radeon 9200	~1800
NVIDIA GeForce 256 Ultra XVII	~1850
ATI Radeon 9300	~1900
NVIDIA GeForce 256 Ultra XVIII	~1950
ATI Radeon 9400	~2000
NVIDIA GeForce 256 Ultra XIX	~2050
ATI Radeon 9500	~2100
NVIDIA GeForce 256 Ultra XX	~2150
ATI Radeon 9600	~2200
NVIDIA GeForce 256 Ultra XXI	~2250
ATI Radeon 9700	~2300
NVIDIA GeForce 256 Ultra XXII	~2350
ATI Radeon 9800	~2400
NVIDIA GeForce 256 Ultra XXIII	~2450
ATI Radeon 9900	~2500
NVIDIA GeForce 256 Ultra XXIV	~2550
ATI Radeon 10000	~2600
NVIDIA GeForce 256 Ultra XXV	~2650
ATI Radeon 10100	~2700
NVIDIA GeForce 256 Ultra XXVI	~2750
ATI Radeon 10200	~2800
NVIDIA GeForce 256 Ultra XXVII	~2850
ATI Radeon 10300	~2900
NVIDIA GeForce 256 Ultra XXVIII	~2950
ATI Radeon 10400	~3000
NVIDIA GeForce 256 Ultra XXIX	~3050
ATI Radeon 10500	~3100
NVIDIA GeForce 256 Ultra XXX	~3150
ATI Radeon 10600	~3200
NVIDIA GeForce 256 Ultra XXXI	~3250
ATI Radeon 10700	~3300
NVIDIA GeForce 256 Ultra XXXII	~3350
ATI Radeon 10800	~3400
NVIDIA GeForce 256 Ultra XXXIII	~3450
ATI Radeon 10900	~3500
NVIDIA GeForce 256 Ultra XXXIV	~3550
ATI Radeon 11000	~3600
NVIDIA GeForce 256 Ultra XXXV	~3650
ATI Radeon 11100	~3700
NVIDIA GeForce 256 Ultra XXXVI	~3750
ATI Radeon 11200	~3800
NVIDIA GeForce 256 Ultra XXXVII	~3850
ATI Radeon 11300	~3900
NVIDIA GeForce 256 Ultra XXXVIII	~3950
ATI Radeon 11400	~4000
NVIDIA GeForce 256 Ultra XXXIX	~4050
ATI Radeon 11500	~4100
NVIDIA GeForce 256 Ultra XL	~4150
ATI Radeon 11600	~4200
NVIDIA GeForce 256 Ultra XLI	~4250
ATI Radeon 11700	~4300
NVIDIA GeForce 256 Ultra XLII	~4350
ATI Radeon 11800	~4400
NVIDIA GeForce 256 Ultra XLIII	~4450
ATI Radeon 11900	~4500
NVIDIA GeForce 256 Ultra XLIV	~4550
ATI Radeon 12000	~4600
NVIDIA GeForce 256 Ultra XLV	~4650
ATI Radeon 12100	~4700
NVIDIA GeForce 256 Ultra XLVI	~4750
ATI Radeon 12200	~4800
NVIDIA GeForce 256 Ultra XLVII	~4850
ATI Radeon 12300	~4900
NVIDIA GeForce 256 Ultra XLVIII	~4950
ATI Radeon 12400	~5000
NVIDIA GeForce 256 Ultra XLIX	~5050
ATI Radeon 12500	~5100
NVIDIA GeForce 256 Ultra L	~5150

K1000!

When LUCK is a fully loaded subsidiary of Right Computer,
Market Leader™ supports another leader! Speedy and Smooth,
our video is fit to any video format to show.

- (Of course we have to make a compute shader based particle system!)
-

Define Buffers and the Shader

```
private ComputeBuffer bufferPoints;  
private ComputeBuffer bufferPos;  
private ComputeBuffer bufferPreviousPoints;  
private ComputeBuffer bufferColors;  
private ComputeBuffer bufferAccelerations;  
public ComputeShader cs;
```

Initialize Buffers

```
void Start () {
    last_delta_time = Time.deltaTime * timestep;

    var verts = new Vector3[vertexCount];
    for (var i = 0; i < vertexCount; ++i)
    {
        verts[i] = Random.insideUnitSphere*10.0f;
    }

    ReleaseBuffers ();

    bufferPoints = new ComputeBuffer (vertexCount, 12);
    bufferPreviousPoints = new ComputeBuffer (vertexCount, 12);
    bufferPoints.SetData (verts);
    bufferPreviousPoints.SetData (verts);
    material.SetBuffer ("buf_Points", bufferPoints);
    bufferColors = new ComputeBuffer (vertexCount, 12);
    material.SetBuffer ("buf_Colors", bufferColors);
    bufferAccelerations = new ComputeBuffer (vertexCount, 12);

    origPos = new Vector3[instanceCount];
    for (var i = 0; i < instanceCount; ++i)
        origPos[i] = Random.insideUnitSphere * 1.0f;
    pos = new Vector3[instanceCount];

    bufferPos = new ComputeBuffer (instanceCount, 12);

    for (var i = 0; i < instanceCount; ++i)
    {
        pos[i] = origPos[i] + Random.insideUnitSphere*.2f;
    }
    bufferPos.SetData (pos);

    material.SetBuffer ("buf_Positions", bufferPos);
}
```

Run the Compute Shader

```
void OnRenderImage (RenderTexture src, RenderTexture dst)
{
    cs.SetBuffer(0, "buf_Points", bufferPoints);
    cs.SetBuffer(0, "buf_PreviousPoints", bufferPreviousPoints);
    cs.SetBuffer(0, "buf_Colors", bufferColors);
    cs.SetBuffer(0, "buf_Accelerations", bufferAccelerations);
    cs.SetFloat ("_time", Time.timeSinceLevelLoad);

    cs.SetFloat ("_deltatime", Time.deltaTime * timestep);
    cs.SetFloat ("_lastdeltatime", last_delta_time);
    last_delta_time = Time.deltaTime * timestep;

    cs.SetFloat ("_drag", drag);
    cs.SetFloat ("_temperature", temperature);
    cs.SetFloat ("_mag", mag);
    cs.SetFloat ("_worldscale", worldscale);
    cs.SetVector("_origin", origin.position);
    cs.Dispatch (0, 128, 128, 1);

    Graphics.Blit (src, dst);
}
```

Blit the results to the screen

```
void OnPostRender () {  
    material.SetPass (0);  
    material.SetFloat("_time", Time.timeSinceLevelLoad);  
    material.SetFloat("_vertexCount", vertexCount);  
    material.SetBuffer ("buf_Points", bufferPoints);  
    material.SetBuffer ("buf_Colors", bufferColors);  
    Graphics.DrawProcedural (MeshTopology.Points, vertexCount, instanceCount);  
}
```


The Compute Shader

```
// each #kernel tells which function to compile; you can have many kernels if you wish
#pragma kernel CSMain

RWStructuredBuffer<float3> buf_Points;
RWStructuredBuffer<float3> buf_PreviousPoints;
RWStructuredBuffer<float3> buf_Accelerations;
RWStructuredBuffer<float3> buf_Colors;

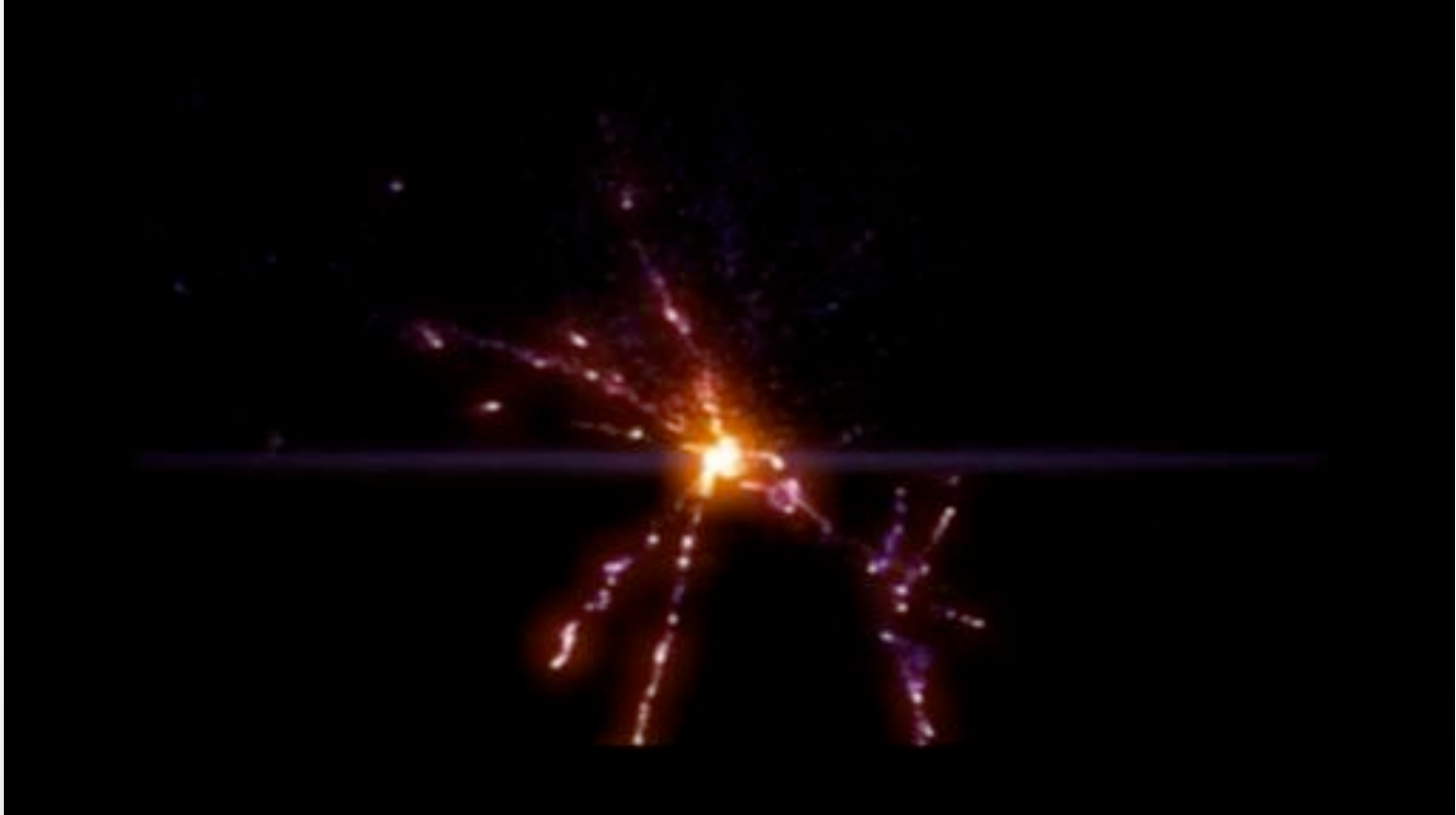
float _time;
float _deltatime;
float _lastdeltatime;
float3 _origin;
float _vertexCount;
float _drag;
float _temperature;
float _mag;
float _worldscale;

[numthreads(32,32,1)]
void CSMain( uint3 Gid : SV_GroupID, uint3 DTid : SV_DispatchThreadID, uint3 GTid : SV_GroupThreadID, uint GI : SV_GroupIndex
{
    uint uniqueID = DTid.x + DTid.y*(128*32) + DTid.z * (8);
    float3 pos = buf_Points[uniqueID];
    float3 previous_pos = buf_PreviousPoints[uniqueID];
    buf_PreviousPoints[uniqueID] = pos;

    float3 orig_accel = buf_Accelerations[uniqueID];
    float3 accel = 0;

    if (true)
    {
        if (uniqueID < 20000)
        {
            for (uint i = 0; i < 20000; i++)
            {
                if (i == uniqueID) continue;
                float3 pos2 = buf_Points[i];
            }
        }
    }
}
```

Unity Prototype: Galaxy



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents

Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations**
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents

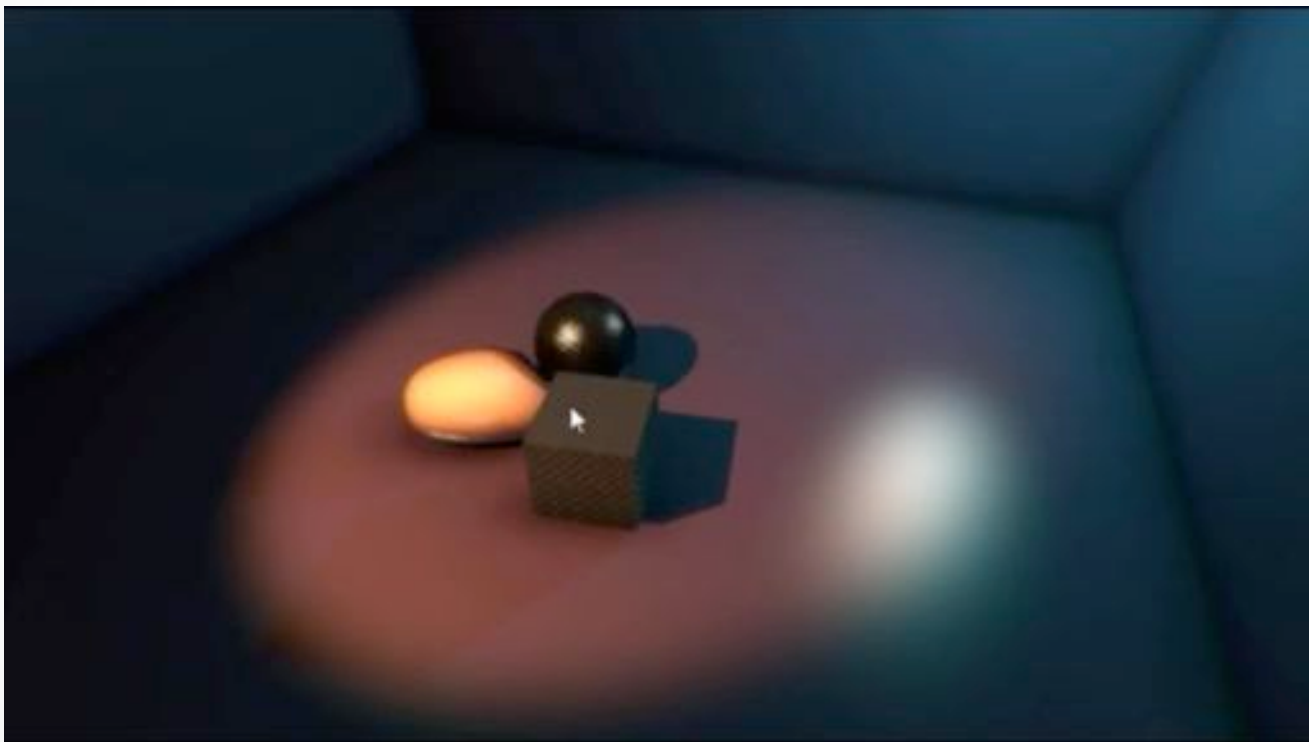
```
[numthreads(32,32,1)]
void CSMain( uint3 Gid : SV_GroupID, uint3 DTid : SV_DispatchT
{
    uint uniqueID = DTid.x + DTid.y*(128*32) + DTid.z * (8);
    float3 pos = buf_Points[uniqueID];
    float3 previous_pos = buf_PreviousPoints[uniqueID];
    buf_PreviousPoints[uniqueID] = pos;

    float3 orig_accel = buf_Accelerations[uniqueID];
    float3 accel = 0;

    if (true)
    {
        if (uniqueID < 20000)
        {
            for (uint i = 0; i < 20000; i++)
            {
                if (i == uniqueID) continue;
                float3 pos2 = buf_Points[i];
```

Next Gen Physics

- What about physics performance on PS4 / XBOX3?



Soft Body Physics

- Soft body physics is cool!



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents

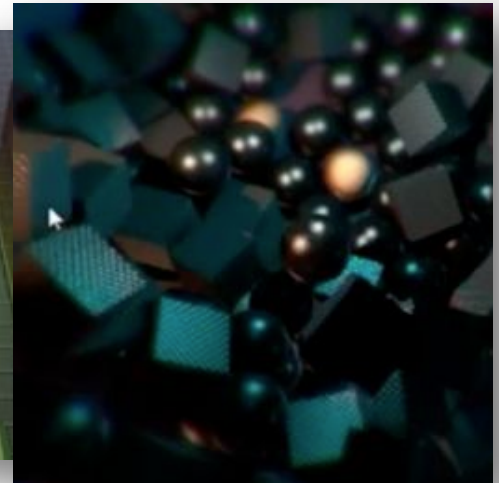
Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them**
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them**
- Learn engine strengths and limitations**
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents



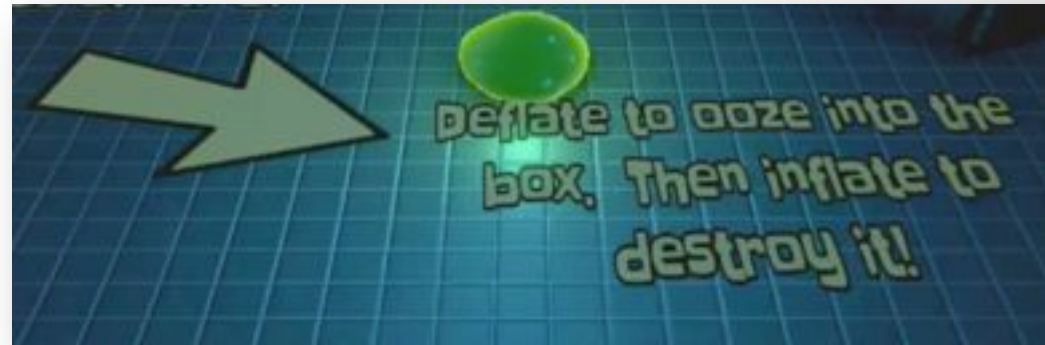
Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them**
- Learn engine strengths and limitations**
- Learn input strengths and limitations
- Fail fast**
- Sign projects
- Happy accidents



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them**
- Learn engine strengths and limitations**
- Learn input strengths and limitations
- Fail fast**
- Sign projects
- Happy accidents**



Prototyping for New Input Devices



New Input Devices

- Explosion of Inputs!
 - Motion Control (wii, kinect)
 - Inertial sensors (6 axis, iPhone accelerometer)
 - Touch, multi-touch
- A lot more is coming!
 - Leap Motion
 - Eye Tracking
 - Face Tracking
 - PS4 Controller
 - High precision body tracking
 - Oculus Rift
 - **BRAIN TRACKING**



Rapid Prototyping PS4



Rapid Prototyping PS4

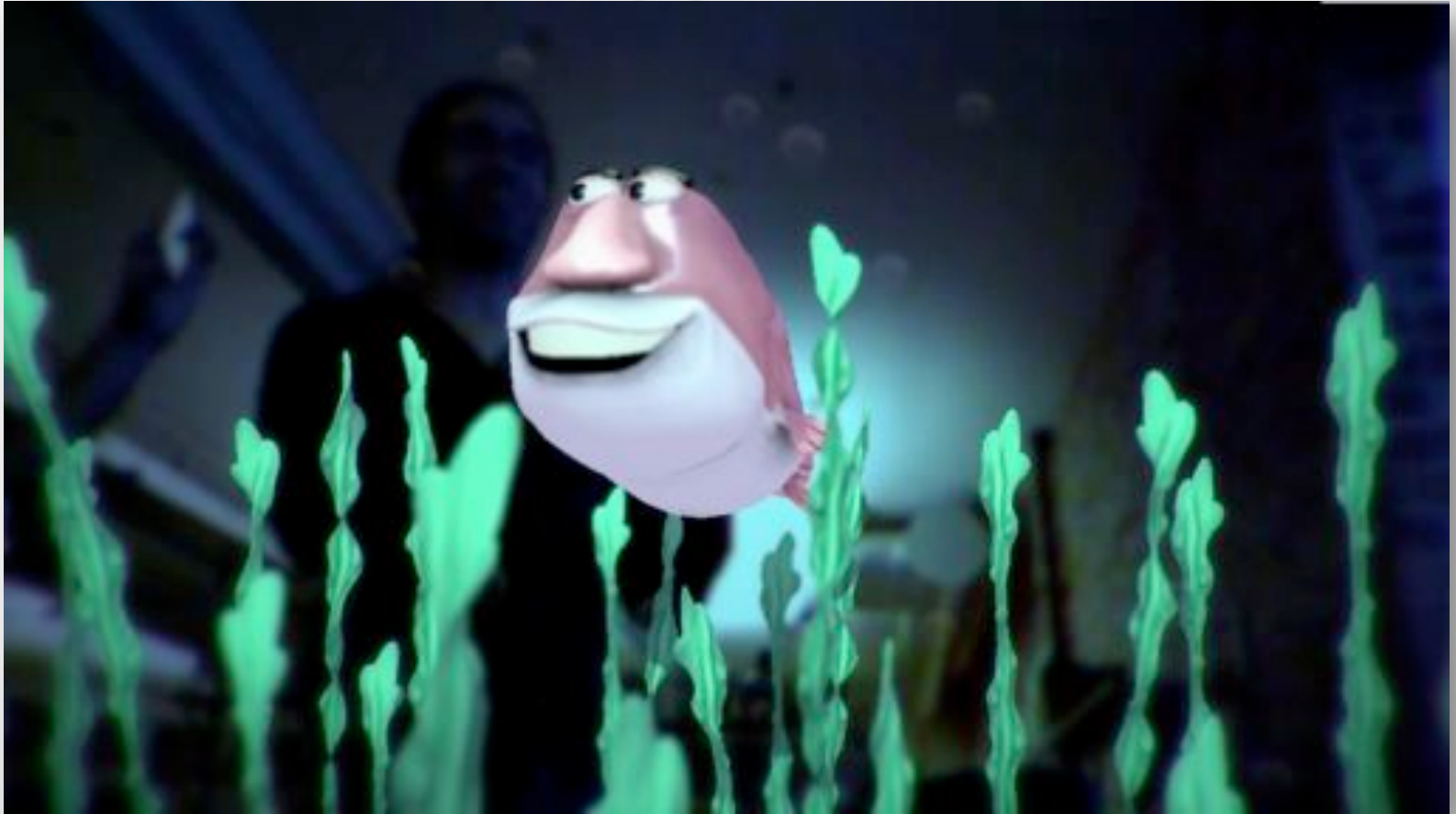
- Faking it!



Unity Prototype: Funny Fish



Unity Prototype: Funny Fish



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents

Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them**
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents



Kinect in Unity is Really Awesome



Rapid Prototyping Leap

- Works great with Unity and Processing



Leap Prototype





Leap Prototype: Lil Bunny





Leap Prototype: Dropchord



MOAI Version





Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents

Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them**
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them**
- Learn engine strengths and limitations
- Learn input strengths and limitations**
- Fail fast
- Sign projects
- Happy accidents



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them**
 - Learn engine strengths and limitations
- Learn input strengths and limitations**
 - Fail fast
- Sign projects**
 - Happy accidents



Prototyping for New Input Devices



Hackers Hack it First

- Experiment and Evaluate new tech *crazy early!*



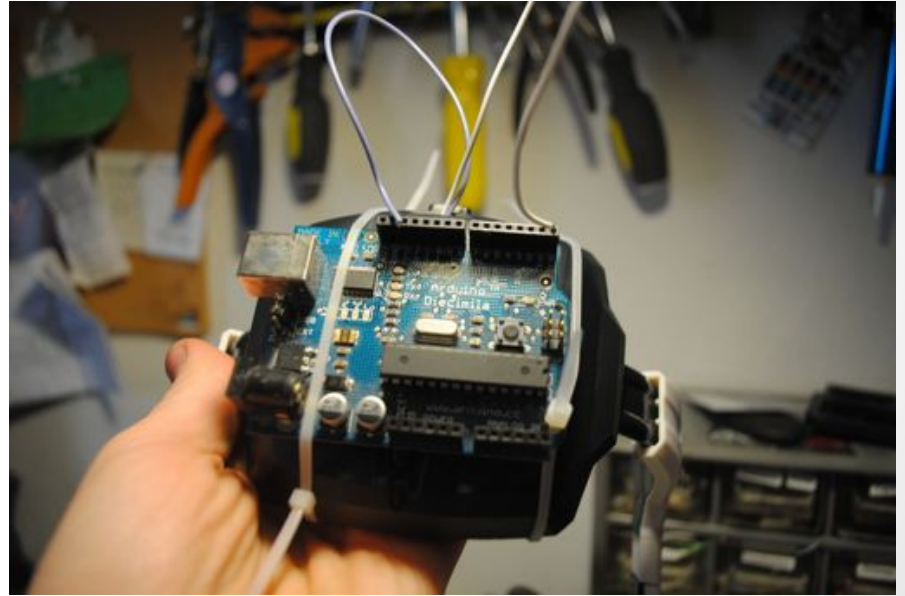
Arduino IDE

- Looks identical to Processing
- Integrates tightly with Processing
- Can communicate with Unity as well



BRAIN TRACKING!

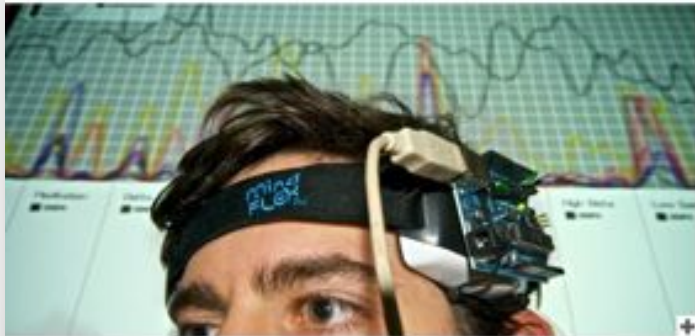
- Can you make fun, interactive stuff with it??



Arduino Tutorials are Great

- Force Trainer + Arduino Uno

How to Hack Toy EEGs



Intro to Physical Computing

arduino

brain

processing

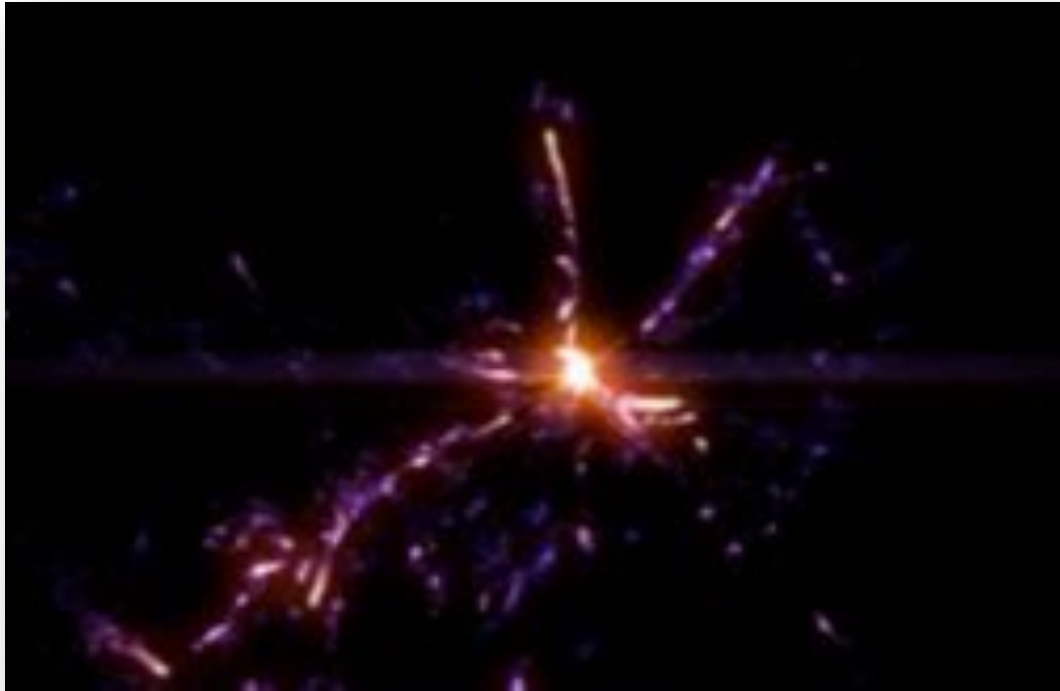
313 comments

Arturo Vidich, Sofy Yuditskaya, and I needed a way to read brains for our Mental Block project last fall. After looking at the options, we decided that hacking a toy EEG would be the cheapest / fastest way to get the data we wanted. Here's how we did it.



Hrmmm... What Next

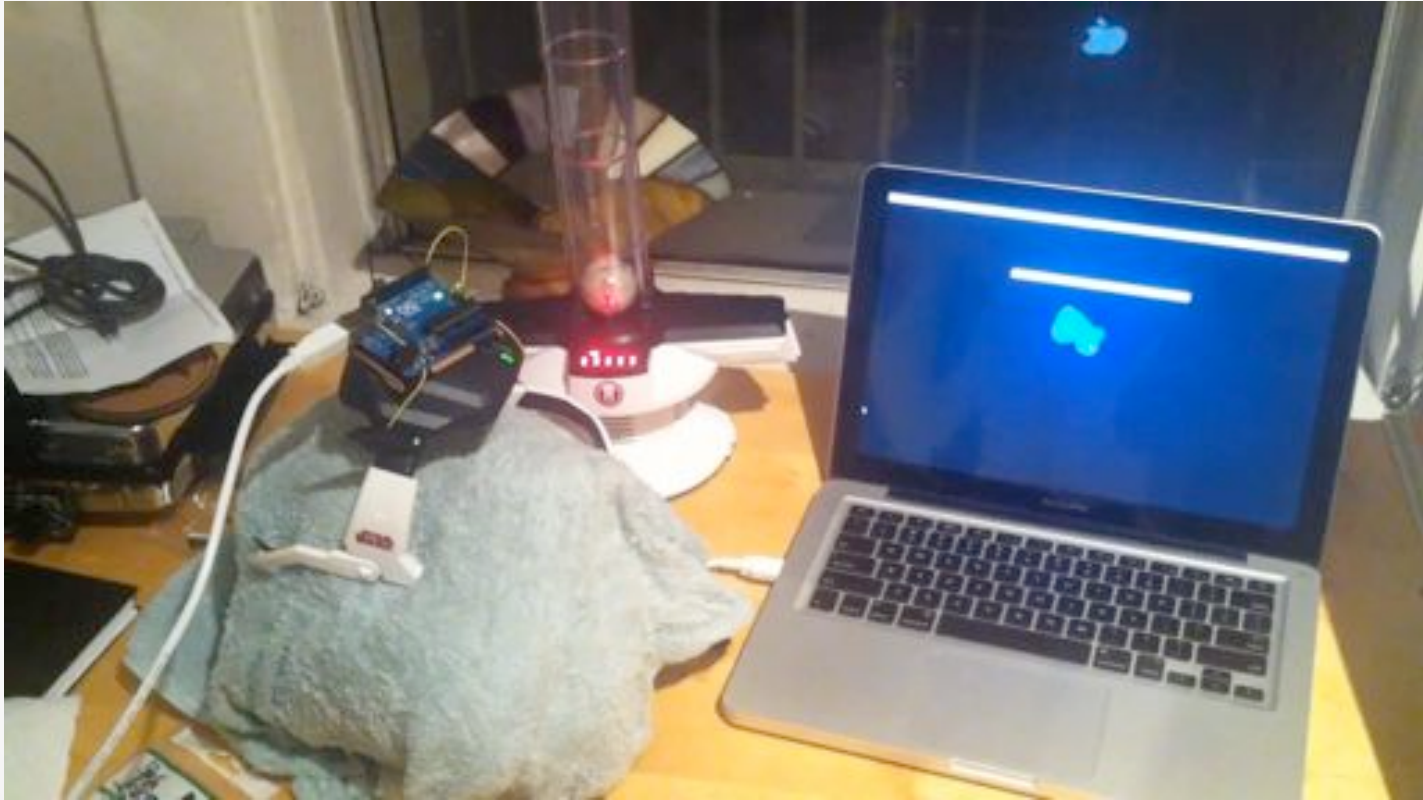
- Obviously we *MUST* to wire this up to a particle system...



Arduino Prototype: Brain Particles

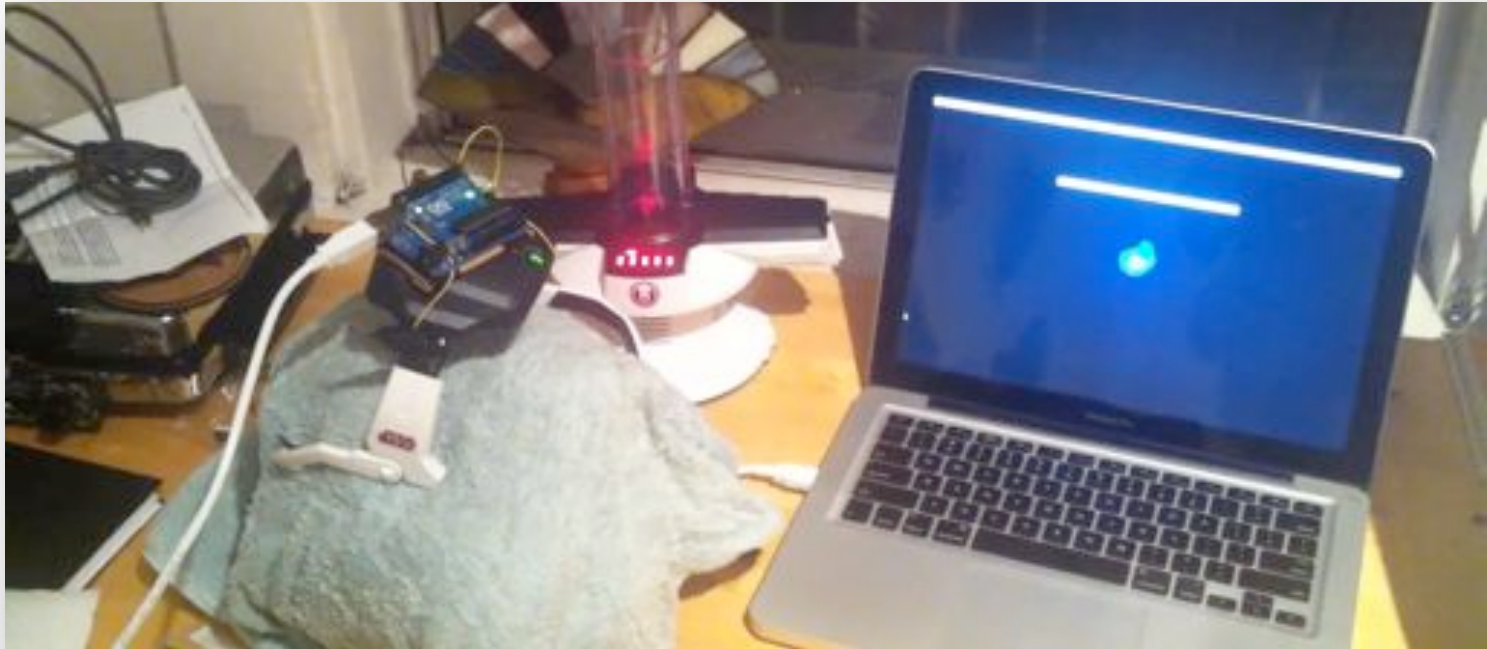


Wet Towel Midichlorians



Wet Towel Midichlorians

- \$50 mind control isn't there quite yet...

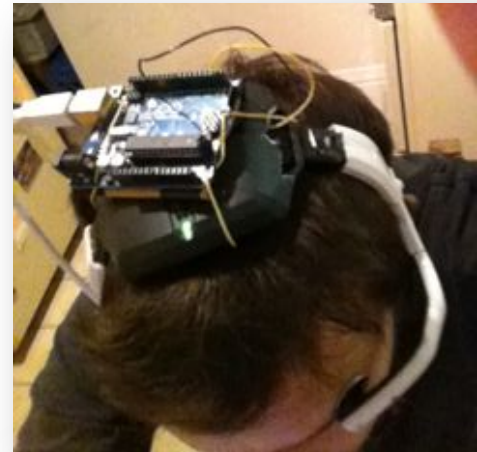


Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents

Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations**
- Fail fast
- Sign projects
- Happy accidents



Rapid Prototyping Payoffs

- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations**
- Fail fast**
- Sign projects
- Happy accidents



Rapid Prototyping

- Minimize Compile Times
- Use Building Blocks
- Sketchbook Approach
 - *Cool things will still happen!*



Rapid Prototyping Payoffs

- ✓ Build confidence in new ideas, and share them
- ✓ Learn engine strengths and limitations
- ✓ Learn input strengths and limitations
- ✓ Fail fast
- ✓ Sign projects
- ✓ Happy accidents



IT'S SUPER FUN

Getting Time at Work

- Get fast on your own time
- Use that speed to
 - Communicate ideas
 - Look dev
 - Profile hardware
 - Etc...
- Results pave the way for prototyping during work hours



We have the tools for **Interactive Sketchbooks**



We *finally* have the tools for Interactive Sketchbooks



Links!



Processing Versions

- Processing 1.5
 - I used this for all the prototypes in this presentation
- Processing 2.0 (beta)
 - Will be great soon (fully OpenGL)
 - GL Graphics library doesn't exist so I'm not switching yet
- The following libraries were all used with 1.5



Processing + Kinect Setup

- Processing
 - www.processing.org
- Simple OpenNI
 - code.google.com/p/simple-openni
- Drivers and Libraries
 - OpenNI, NITE, PrimeSensor, SensorKinect
 - All downloadable from:
 - code.google.com/p/simple-openni/wiki/Installation



Other Processing + Kinect Libraries

- OpenKinect (Mac OS X Only)
 - <http://www.shiffman.net/p5/kinect>
- dLibs_freeneect
 - http://thomasdiewald.at/processing/libraries/dLibs_freeneect



Processing Libraries

- Some of my favorite libraries:
 - Fluid Simulation: **msafluid**
 - http://www.memo.tv/msafluid_for_processing_v1_3
 - Blob Detection: **OpenCV**
 - <http://ubaa.net/shared/processing/opencv/>
 - Open GL: **GLGraphics** *Not compatible with processing 2.0
 - <http://glgraphics.sourceforge.net/>
 - Spring Physics: **traer.physics**
 - <http://murderandcreate.com/physics>
 - Box2D Physics: **fisica**
 - <http://www.ricardmarxer.com/fisica>



Arduino Libraries

- Brain Control
 - <http://frontiernerds.com/brain-hack>



Unity + Kinect Setup

- Unity
 - www.unity3d.com
- Zig Fu
 - Zigfu.com
 - Install
 - Zig Fu drivers
 - Zig Fu Unity example package



Some Cool Unity Packages

- Some of my favorite packages
 - Easy Touch
 - <https://www.assetstore.unity3d.com/#/content/3322>
 - Decal System
 - <https://www.assetstore.unity3d.com/#/content/3779>
 - Visualizer Studio
 - <https://www.assetstore.unity3d.com/#/content/1761>



Prototyping on Mobile Devices

- Processing deploys to Android
 - Almost instantaneous. Performance limited. No support for iOS
- Unity deploys to iOS and Android
 - Very performant and robust. ~2 minute compile time on my 2010 macbook.
 - Unity Remote helps mitigate some of that



A close-up photograph of an industrial robotic arm performing a welding task. The robot's gripper is positioned above a workpiece, and a bright, intense light from the welding process is visible at the point of contact. A massive spray of bright orange and yellow sparks is being ejected from the welding area, creating a dynamic and energetic scene. The background is dark, highlighting the industrial machinery and the welding process. Overlaid on the center of the image is the word "Thanks!" in a large, bold, white, sans-serif font with a slight drop shadow, making it stand out against the dark background and the bright sparks.

Thanks!

Session PDF

- www.drewskillman.com/gdc2013_rapid_prototyping.pdf

Visual Effects Artist Roundtable

Room 120, North Hall

11am – 12pm Wednesday

5:30pm – 6:30pm Thursday

2:30pm – 3:30pm Friday

Augmented Imagination: Exploiting Kinect for Happy Action Theater

Patrick Hackett | Weds 2pm-3pm | 303 South Hall