

Rapid Prototyping at Double Fine



Introductions

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



My Motivation for Talk



ActivityHub.bat	OpenNI_Knight
Ad_Blizzard.bat	OpenNI_Knight2
Ad_KungFu.bat	OpenNI_Knight3
Ad_Parrot.bat	optical_flow_basic
BathTime.bat	optical_flow_basic_02
BuildingBlocks.bat	optical_flow_radial_beams
Constellation.bat	particleTest
Credits.bat	particleTest_optical_flow
DressUp.bat	particleTest_optical_flow_02
DubStep.bat	particleTest_optical_flow_03
EightBit.bat	particle3d_kinect
EzV.bat	particleTest
Giliams_BowlingCrumble.bat	processing-1.5.1
Giliams_BowlingDodge.bat	renderToModelTest
Giliams_BowlingStrike.bat	rk
Giliams_ChickenHatch.bat	rk_01
Giliams_ChickenHatchNDash.bat	rk_02
Giliams_ChickenMatryoshka.bat	rk_03
Giliams_DrunkGuy.bat	rk_04
Giliams_MonsterCatapult.bat	rk_05
Giliams_MonsterEat.bat	rk_06
Giliams_MonsterFly.bat	rk_07
Giliams_MonsterLick.bat	rk_08
Giliams_MonsterMouthPull.bat	rk_09
Giliams_SalonBigHeads.bat	rk_10
Giliams_SalonHairCut.bat	rk_11
Giliams_SalonNewHair.bat	rk_12
Giliams_SalonSmallHeads.bat	rk_13
Giliams_Turkey.bat	rk_14
GridWorld.bat	rk_15
Headappiness.bat	rk_16
Holodeck.bat	rk_17
Intro.bat	rk_18
KungFu.bat	rk_19
Paint.bat	rk_20
Parrot.bat	rk_21
ParticleField.bat	rk_22
PhotoBrowser.bat	rk_23
PhotoRecap.bat	rk_24
PlayerCannon.bat	rk_25
Popcorn.bat	rk_26
Quicksand.bat	rk_27
RainCloud.bat	rk_28
RawFeed.bat	rk_29
RibbonDancer.bat	rk_30
Sequencer.bat	rk_31
SideScroller.bat	rk_32
SitScan.bat	rk_33
SlotMachine.bat	rk_34
Tesla.bat	rk_35
UpsellVideo.bat	rk_36
VideoBooth.bat	rk_37
VideoTrail.bat	rk_38
VoxelRunner.bat	rk_39
	snapshot_kinect
	snapshot_kinect_05
	snapshot_kinect_06
	SoundWave
	SoundWave2
	SoundWave3
	SoundWave4
	SoundWave5
	SoundWave5_Mac
	SVG_Jiggle
	SVG_Music
	tetris_test
	tools

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

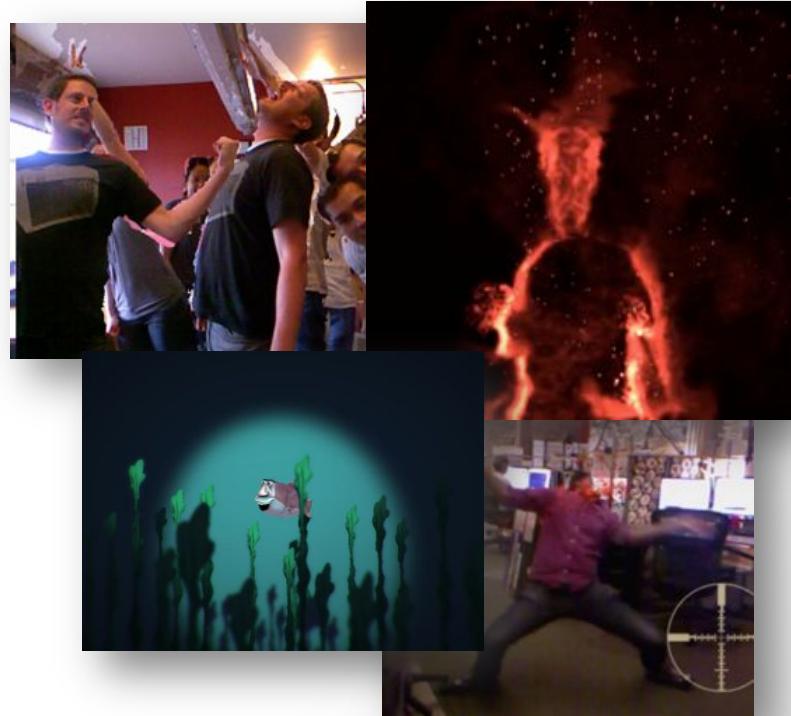


Technical Artists

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

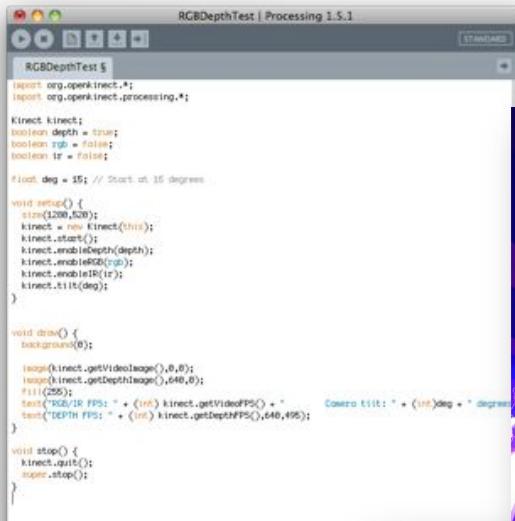


Prototyping **VS** Rapid Prototyping



Talk Overview

Prototyping Platforms



```
RGBDepthTest | Processing 1.5.1
STREAMS
RGBDepthTest $
```

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

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

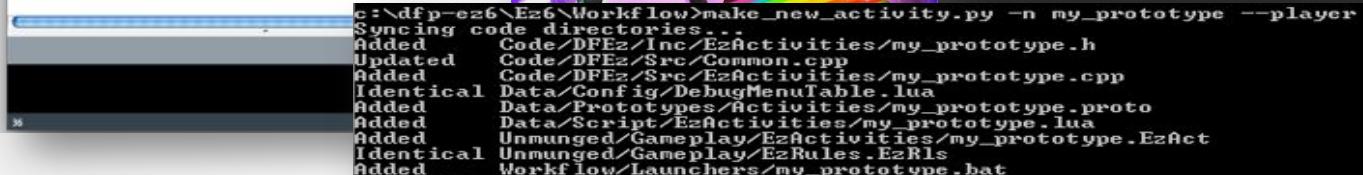
float deg = 165 // Start at 16 degrees

void setup() {
    size(1280,630);
    kinect = new Kinect(this);
    kinect.start();
    kinect.enableDepth(depth);
    kinect.enableRGB(rgb);
    kinect.enableIR(ir);
    kinect.enableDepth();
}

void draw() {
    background(0);

    image(kinect.getVideoImage(),0,0);
    image(kinect.getDepthImage(),640,0);
    fill(255);
    text("RGB/IR FPS: " + (int)kinect.getVideoFPS() + "\n", 10, 10);
    text("DEPTH FPS: " + (int)kinect.getDepthFPS(),640,10);
}

void stop() {
    kinect.quit();
    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     Unmunged/Gameplay/EzActivities/my_prototype.EzAct
Identical Unmunged/Gameplay/EzRules.EzRls
Added     Workflow/Launchers/my_prototype.bat
```

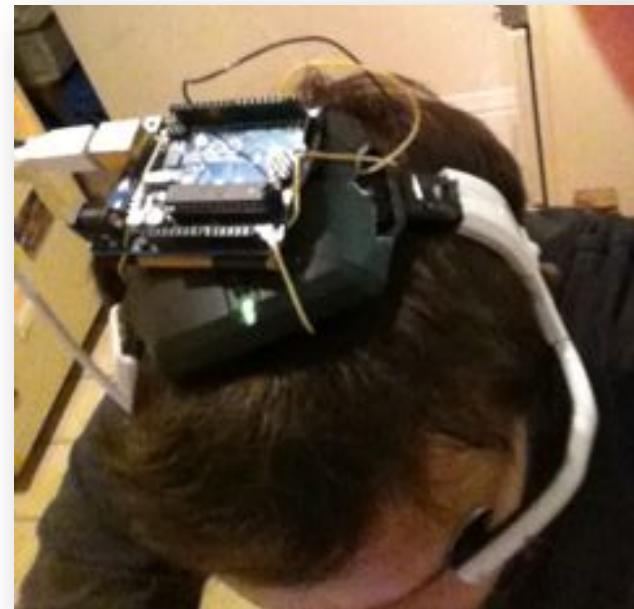
Examples from Kinect Party



Prototyping for Next Gen



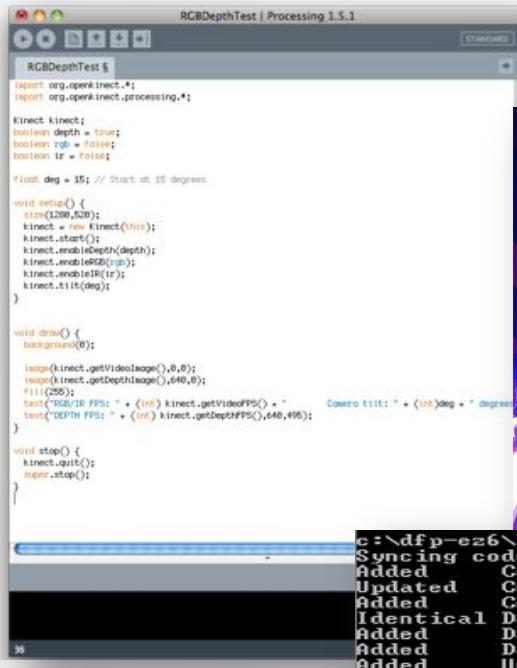
Prototyping for New Input Devices



Links!



Prototyping Platforms



```
RGBDepthTest | Processing 1.5.1
STREAMS
```

```
RGBDepthTest $
```

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

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

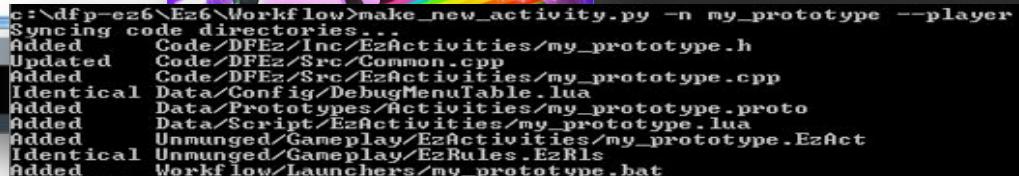
float deg = 165 // Start at 16 degrees

void setup() {
    size(1280,630);
    kinect = new Kinect(this);
    kinect.start();
    kinect.enableDepth(depth);
    kinect.enableRGB(rgb);
    kinect.enableIR(ir);
    kinect.enableDepth();
}

void draw() {
    background(0);

    image(kinect.getVideoImage(),0,0);
    image(kinect.getDepthImage(),640,0);
    fill(255);
    text("RGB/IR FPS: " + (int)kinect.getVideoFPS() + "\n", 10, 10);
    text("DEPTH FPS: " + (int)kinect.getDepthFPS(),640,10);
}

void stop() {
    kinect.quit();
    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     Unmunged/Gameplay/EzActivities/my_prototype.EzAct
Identical Unmunged/Gameplay/EzRules.EzRls
Added     Workflow/Launchers/my_prototype.bat
```

Processing



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



A screenshot of the Processing IDE showing the code for an RGBDepthTest sketch. The code initializes a Kinect sensor and sets up the camera's tilt. It then loops, displaying the video feed and depth data, and prints the FPS of both cameras to the console.

```
RGBDepthTest | Processing 1.5.1
STANDARD

RGBDepthTest {
    import org.openkinect.*;
    import org.openkinect.processing.*;

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

    #loop deg = 15; // Start at 15 degrees

    void setup() {
        size(1200,500);
        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() + " degrees", 100, 50);
        text("DEPTH FPS: " + (int)kinect.getDepthFPS(), 640, 495);
    }

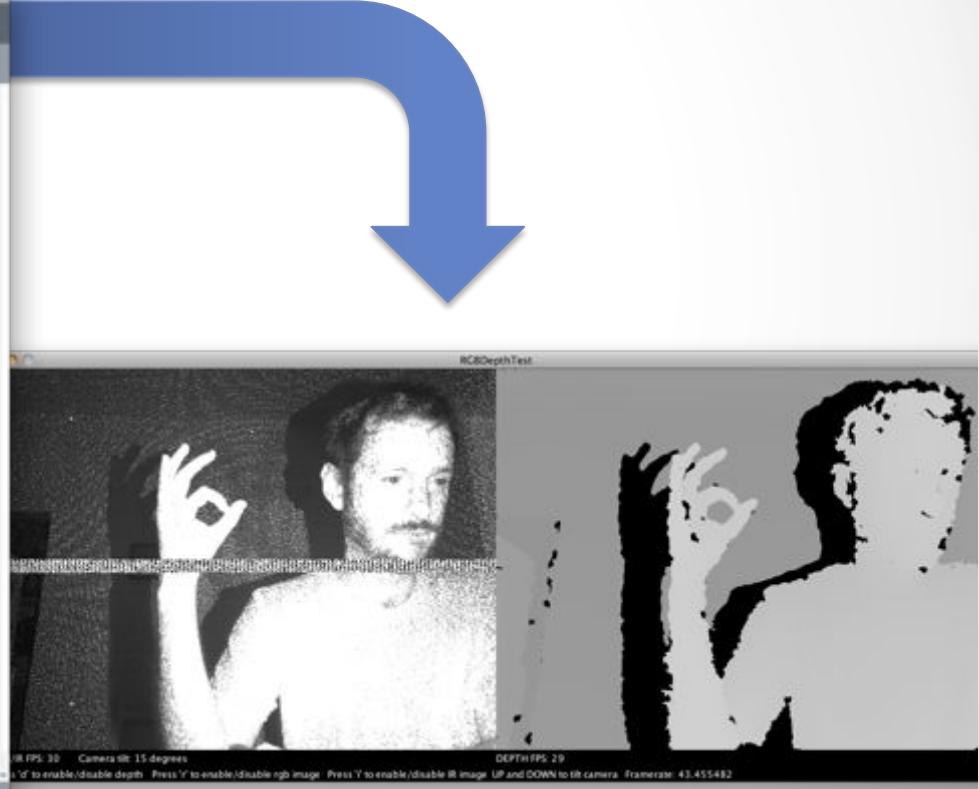
    void stop() {
        kinect.stop();
        stop();
    }
}
```

Processing

RGBDepthTest | Processing 1.5.1

```
STANDARDS
```

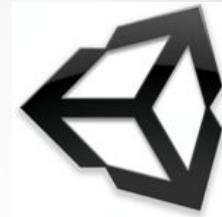
```
RGBDepthTest $  
import org.openkinect.*;  
import org.openkinect.processing.*;  
  
Kinect kinect;  
boolean depth = true;  
boolean rgb = false;  
boolean ir = false;  
  
float deg = 15; // Start at 15 degrees  
  
void setup() {  
    size(1280,530);  
    kinect = new Kinect(this);  
    kinect.start();  
    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", 10, 10);  
    text("DEPTH FPS: " + (int)kinect.getDepthFPS(),640,495);  
}  
  
void stop() {  
    kinect.quit();  
    super.stop();  
}
```



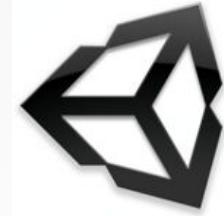
Processing



Unity



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
STREAMS

RGBDepthTest $ 
import org.openkinect.*;
import org.openkinect.processing.*;

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

float deg = 10; // Start at 10 degrees

void setup() {
    size(1280,630);
    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() + "\n", 10, 10);
    text("DEPTH FPS: " + (int)kinect.getDepthFPS(),640,10);
}

void stop() {
    kinect.quit();
    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     Unmunged/Gameplay/EzActivities/my_prototype.EzAct
Identical Unmunged/Gameplay/EzRules.EzRls
Added     Workflow/Launchers/my_prototype.bat
```

Rapid Prototyping

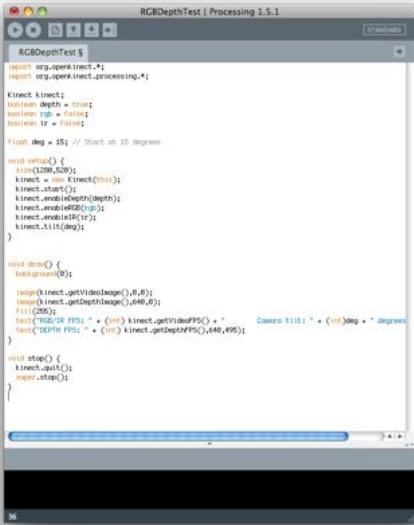
The collage consists of four panels:

- Top Left:** A screenshot of a Processing IDE showing code for a Kinect depth test. An orange arrow points from this panel towards the center of the collage.
- Top Right:** A screenshot of the Unity Editor showing a scene titled "UNINITIALIZED SPACE: DANCE TO A RAY FROM SCRATCH". It displays a 3D model of a glowing blue ring and its properties in the Inspector and Timeline panels.
- Bottom Left:** A screenshot of a terminal window displaying the command: `c:\dfp-ez6\Workflow>make_new_activity.py -n my_prototype --player`. The output shows files being added, updated, and identified.
- Bottom Center:** A screenshot of a game's main menu titled "KINECT Party". The menu features a smiling man, a DJ booth, and a girl with butterfly wings, all set against a vibrant, colorful background.

```
c:\dfp-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
 - Maximize iteration



RGBDepthTest | Processing 1.5.1

```
RGBDepthTest $
```

```
import org.openintents.libprocessing.*;
import org.openintents.libprocessing.*;

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

float deg = 15; // Start at 15 degrees

void setup() {
    size(1280,500);
    kinect.start();
    kinect.enableDepth(depth);
    kinect.enableRGB(rbg);
    kinect.enableIR(ir);
    kinect.setFps(100);
}

void draw() {
    background(0);

    image(kinect.getVideoImage(),0,0);
    image(kinect.getDepthImage(),480,0);
    #111(255);
    text("RGB/IR FPS: " + (int)kinect.getVideoFPS() + " Camera tilt: " + (int)deg + " degrees");
    text("DEPTH FPS: " + (int)kinect.getDepthFPS(),480,495);
}

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

36



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.	PDF Export Create PDF files. These vector graphics files can be scaled to any size and printed at high resolutions.	Minim Uses JavaSound to provide an easy-to-use audio library while still providing flexibility for more advanced users.	Computer Vision / Video GStreamer by Andreas Colodri Uses GStreamer as an alternative to QuickTime for movie playback and camera capture.	imoviekit by Agnus Forbes A simple wrapper for playing videos and grabbing video data for any of the formats that the JMC library supports.	JMxram (WebCamXtra) by Josh nimoy et al. Camera library for motion detection, color tracking, blob distinction, and pixel addressing. Does not require QuickTime or WinAVG for Windows machines.
Network Send and receive data over the Internet through simple clients and servers.	DXF Export Create DXF files to save geometry for loading into other programs. It works with triangle-based graphics including polygons, boxes, and spheres.	Ardublock Directly control an Arduino board through Processing.	BlobDetection by YiZhe Performs the computer vision technique of finding "blobs" in an image.	Face Detect (PJC) by Brooks Cagle Face detection library made with WebCamXtra and the openCV framework. PC only.	libCV by todd Grabs video frames from a camera using the Java Media Framework (JMF). Does not require QuickTime or WinAVG for Windows machines.
Serial Send data between Processing and external hardware through serial communication (RS-232).	Netscape JavaScript Methods for interfacing between Javascript and programs exported from Processing.	OpenCV by Sébastien Cousty and Douglas Ertl An OpenCV implementation for processing including blob detection, face recognition and more. This library is highly recommended.	TUIO Client by Hanno Kalmijnpoort Client library for the simple creation of tangible interactive surfaces, receiving TUIO data from object and multi-touch trackers such as reactables .	TUIO Zones by jLXST tuiZones provides a way to set zones within a multi-touch screen to respond to TUIO messages sent from a tracking application.	
Simulation / Math Physics by Jeffrey Traer Bernstein Simple particle system physics engine. No collisions, just particles, springs, gravity & drag.	Cell Noise by Curtis-Jordan Rosen Explores cell noise (Worley noise), a pattern generation algorithm useful for animation.	MatrixMath by Frances Blitzen Helpful code for matrix operations.	IntegralHistogram by Giovanni Tamburri and Alessandro Bartolucci The integral histogram method allows to obtain the color or intensity histogram of all possible target regions in a Cartesian data space.	P-SURE by Claudio Pantacci and Alessandro Marzocchini An implementation of the SURF (Speeded Up Robust Features) feature detector to search for discrete image correspondences.	CBModel by Federico Bartoli CBModel is a library for motion detection based on background modeling and subtraction.
MSAFluid by Hans Aken A library for solving real-time fluid dynamics simulations.	Eliza by Andreas Colodri Implementation of the classes A.I., bot, Eliza...	Cellular Automata by Frances Blitzen Simplifies making cellular automata calculations.	BoxWrap2D by ewjordan BoxWrap2D runs on top of JBox2D, enabling simple integration of JBox2D with Processing.	blob by André Sier A fast multi-blob detector and tracker using flood-fil algorithms.	dklib by Thomas Diewald This kinect library is based on the libfreenect-software , it currently only works on windows.
AI Libraries by Aaron Steed A set of libraries to assist with artificial programming tasks such as genetic algorithms and the AStar algorithm.	LSystem Utilities by Martin Proulx A library for exploring and creating Lindenmayer Systems in 2D and 3D.	Combinatorics by Ricard Marquer Pinon A wrapper for JBox2D, a 2D physics engine.	blobscanner by Antonio Holnagl A library for blob detection and analysis in image and video streams.	simple-openni by Max Rehner A simple OpenNI and NITE wrapper for Processing.	
Probabilities by Yannick Faust A collection of utilities for performing some statistical and matrix-related manipulations.					

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



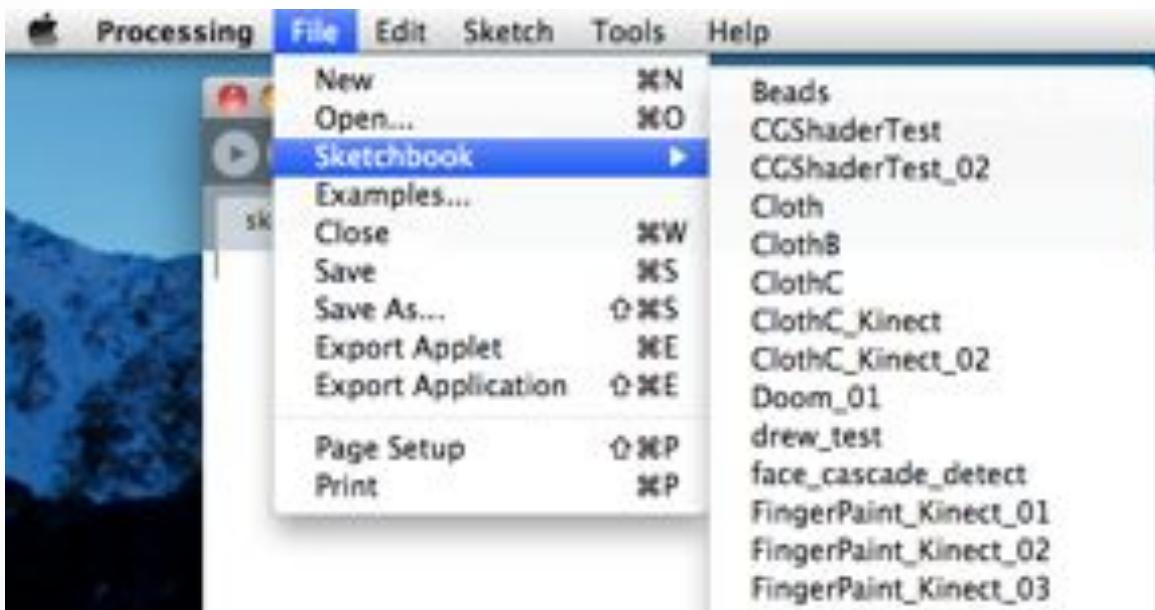
Rapid Prototyping

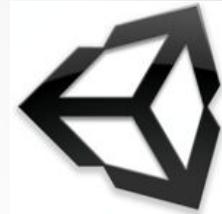
- **Minimize Compile Times**
 - Maximize iteration
- **Use Building Blocks**
 - Don't ever start from scratch if you can help it
- **Sketchbook Approach**
 - Make it easy to start, easy to branch, easy to experiment

Sketchbooks



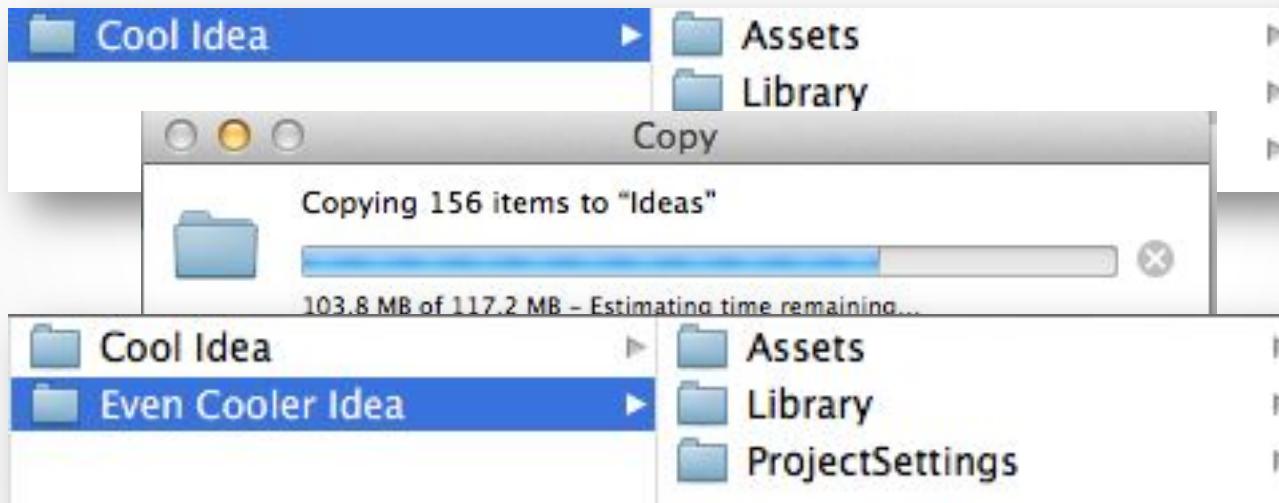
- Processing:





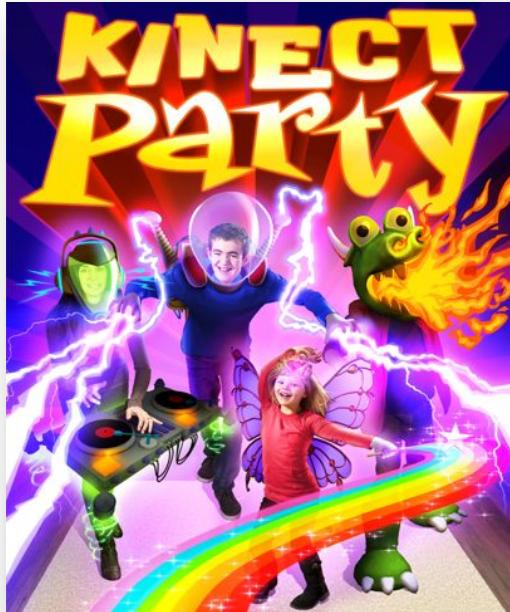
Sketchbooks

- **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
```



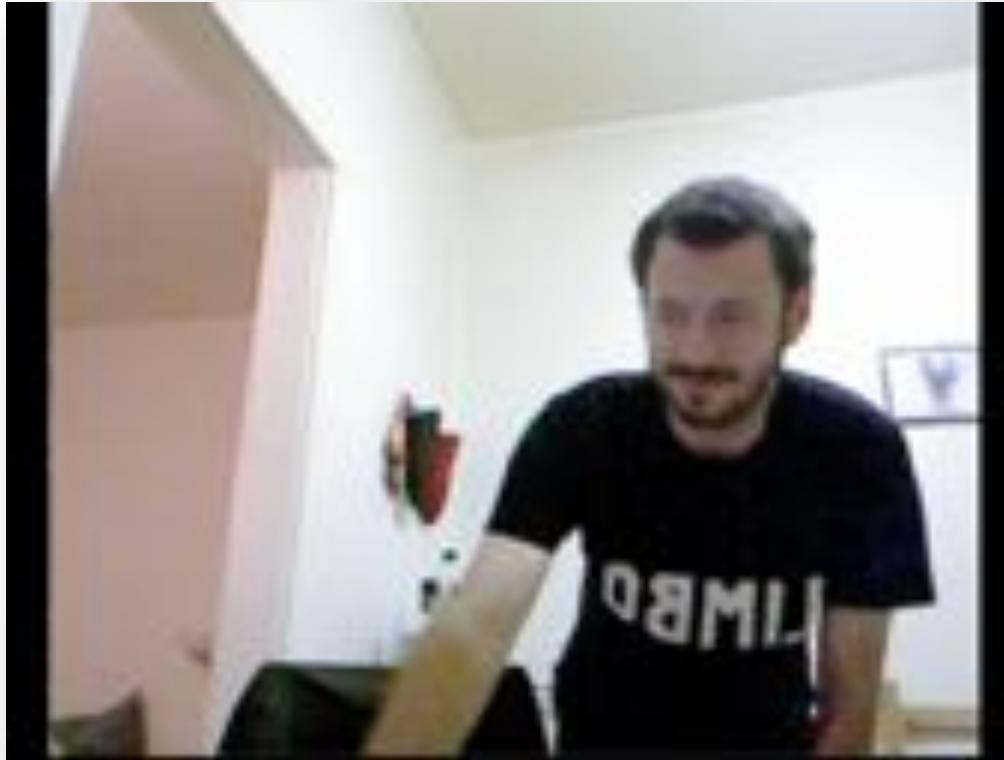
Rapid Prototyping

- **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**



```
vec2 center = vec2(0.5, 0.5);
vec2 rot_coord = center + rotation * (brush_coord - center);

float brush_alpha = texture2D(tex_unit_brush, rot_coord).r;
vec3 image_color = texture2DRect(tex_unit_color, color_coord).rgb;
```

GLGraphics

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).r;
vec4 image_color = texture2DRect(tex_unit_color, color_coord);
```

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_occlusion;  
GLGraphicsOffScreen gl_buffer_finalcolor;  
  
// Full Screen Shaders:  
GLTextureFilter blueFilter;  
GLTextureFilter greenFilter;  
GLTextureFilter combineFilter;  
GLTextureFilter compFilter;  
GLTextureFilter copyFilter;  
GLTextureFilter disFilter;
```

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()
{
    blueFilter = new GLTextureFilter(this, "Blur.vml");
    blueFilter.setBlendMode(ADD);

    copyFilter = new GLTextureFilter(this, "CopyImage.vml");
    copyFilter.setBlendMode(REPLACE);

    dispFilter = new GLTextureFilter(this, "DisplayImage.vml");
    dispFilter.setBlendMode(REPLACE);

    resopFilter = new GLTextureFilter(this, "ResampleImage.vml");
    resopFilter.setBlendMode(ADD);

    combineFilter = new GLTextureFilter(this, "CombineImage.vml");
    combineFilter.setBlendMode(ADD);

    coopFilter = new GLTextureFilter(this, "CoopImage.vml");
    coopFilter.setBlendMode(SCREWDIVIDE);
}
```

GLGraphics Library

- Apply Shaders to Textures (like a video feed)

```
// add fluid to the accumulation buffer
copyFilter.setBlendMode(ALD);
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 resep effect
GLTexture[] inputTex = { gl_buffer_accum.getTexture(), gradientTex };
resepFilter.apply(inputTex, gl_buffer_finalcolor.getTexture());

// Draw the render target to the final buffer
swap(gl_buffer_finalcolor.getTexture(), 0,0,width,height);
```

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



make_new_activity.py -n Blizzard



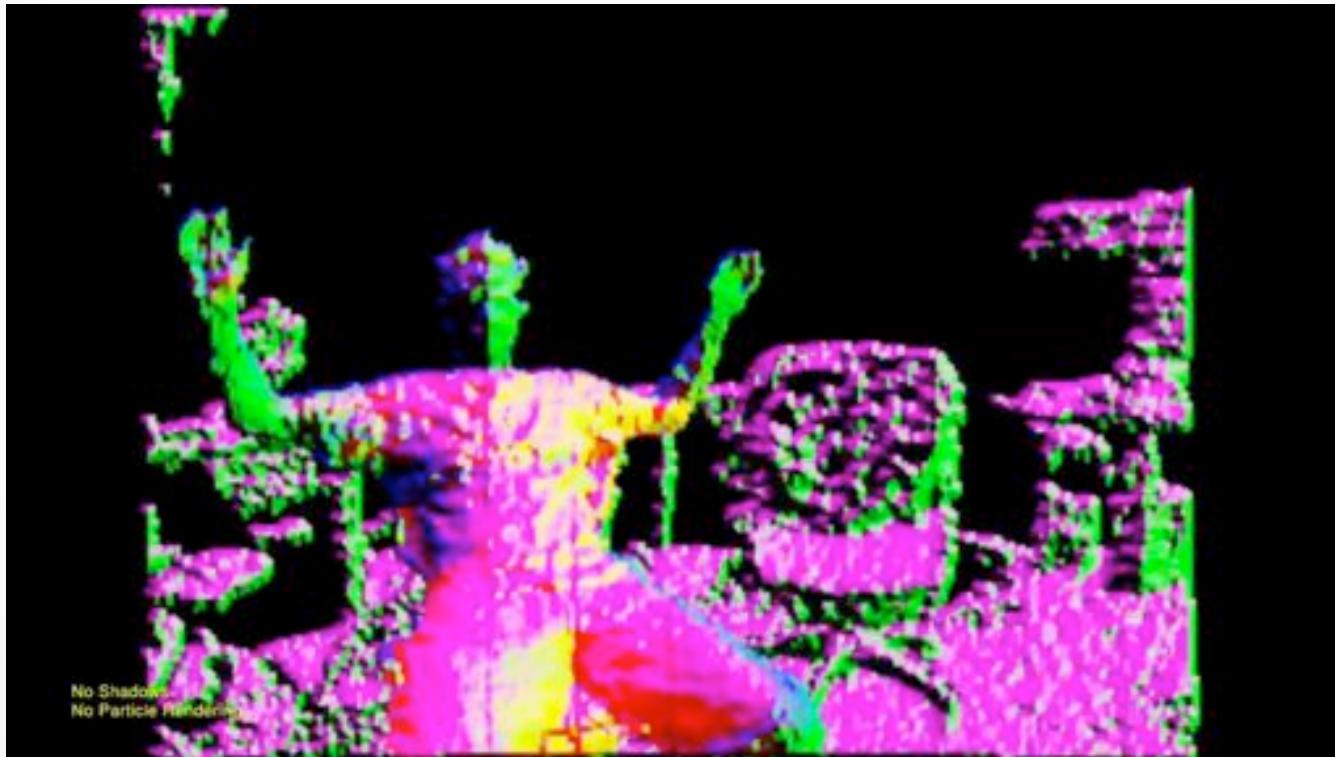
```
make_new_activity.py -n Blizzard
```

- Raw Depth



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

- Sobel filtered depth



`make_new_activity.py -n Blizzard`

- Isolating upward facing normals



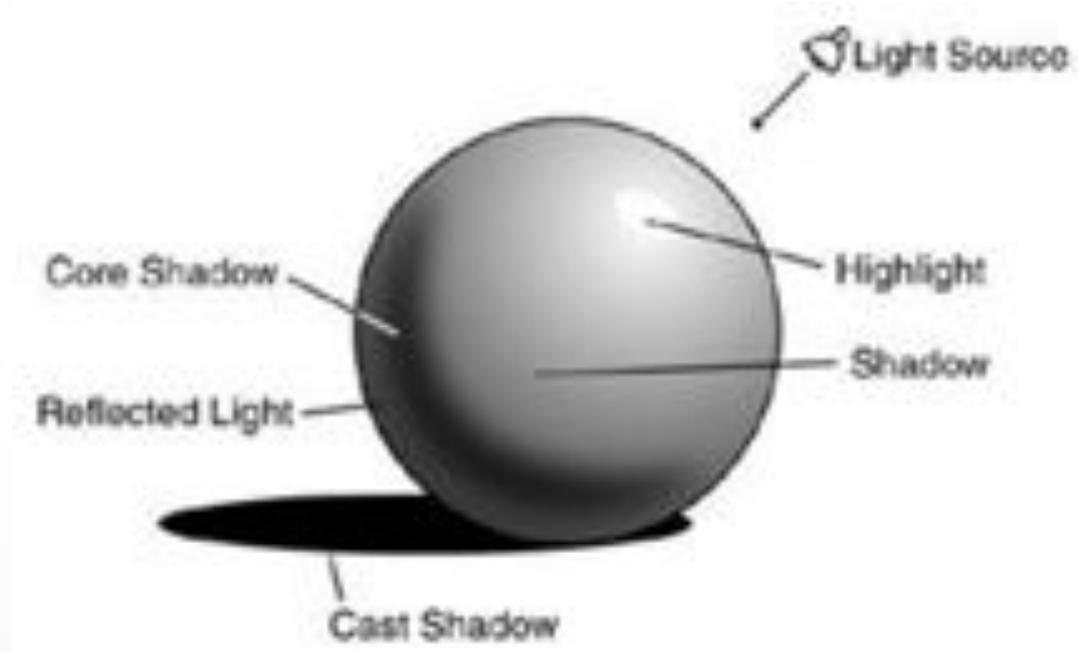
`make_new_activity.py -n Blizzard`

- 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



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(400);
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 cleanup effect
GLTexture[] inputTex = { gl_buffer_accum.getTexture(), gradientTex };
cleanupFilter.apply(inputTex, gl_buffer_finalcolor.getTexture());

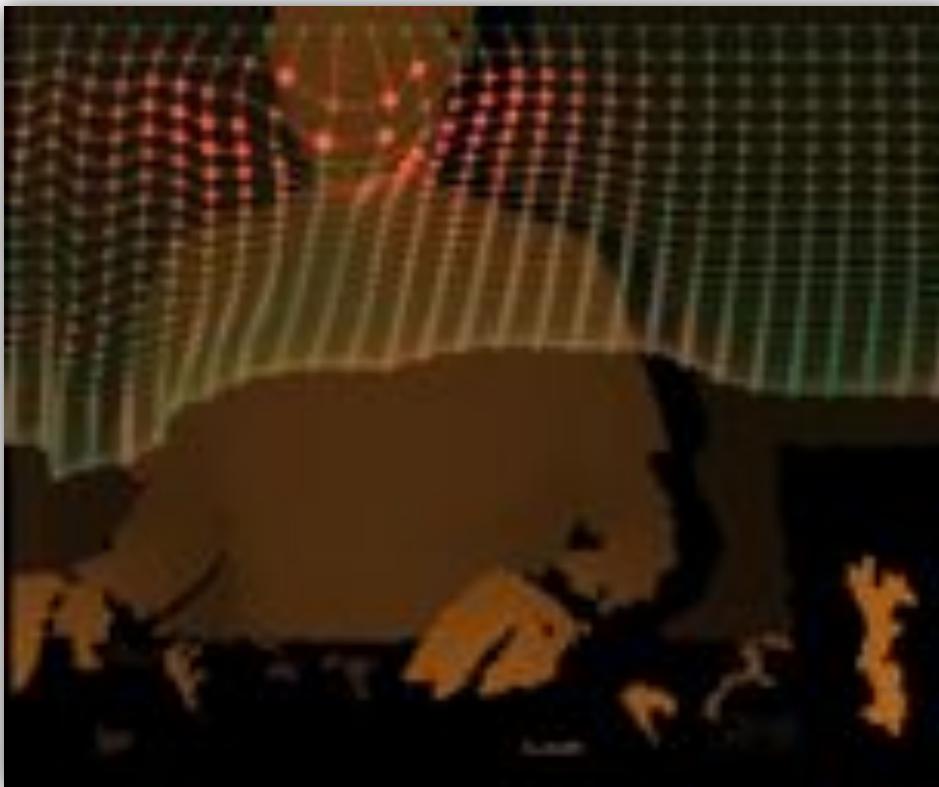
// Draw the render target to the final buffer
image(gl_buffer_finalcolor.getTexture(), 0,0,width,height);
```

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

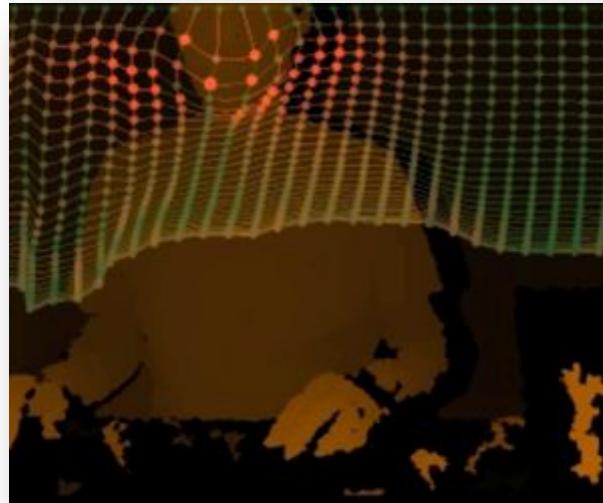
- Build confidence in new ideas, and share them
- Learn engine strengths and limitations
- Learn input strengths and limitations
- Fail fast
- Sign projects
- Happy accidents

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



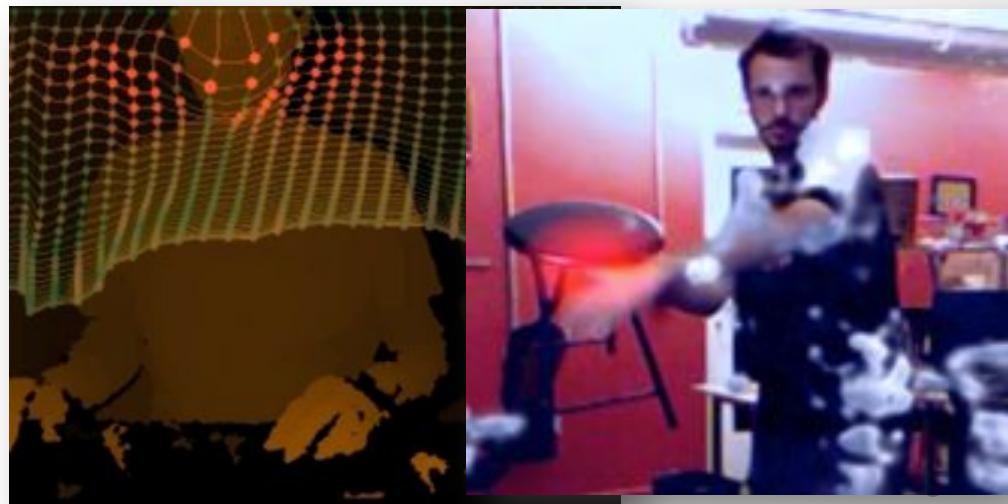
Rapid Prototyping Payoffs

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
```

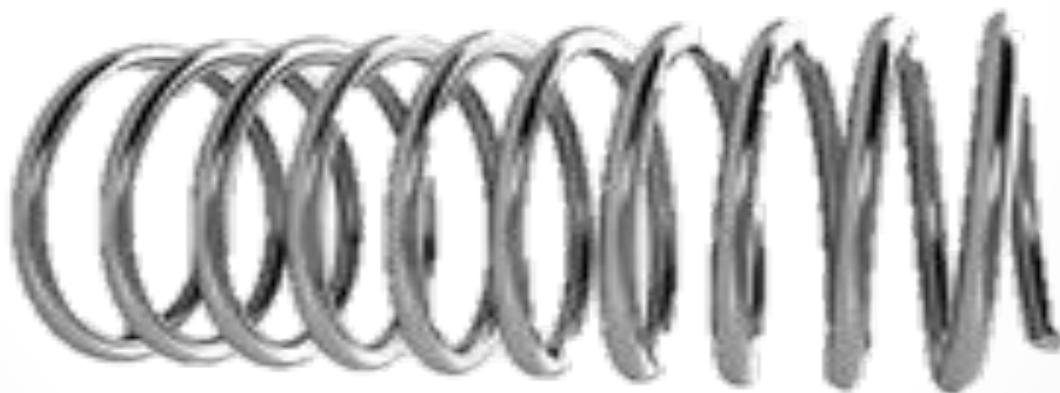


Physics + Lots of Full Screen Shaders

- Force on a spring: $F = -K \times x$

K is “Hooke’s Constant”

x is spring length



Physics + Lots of Full Screen Shaders

- Force on a spring: $F = -K * \mathbf{x}$
- 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



Strong Springs



No Shadows

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

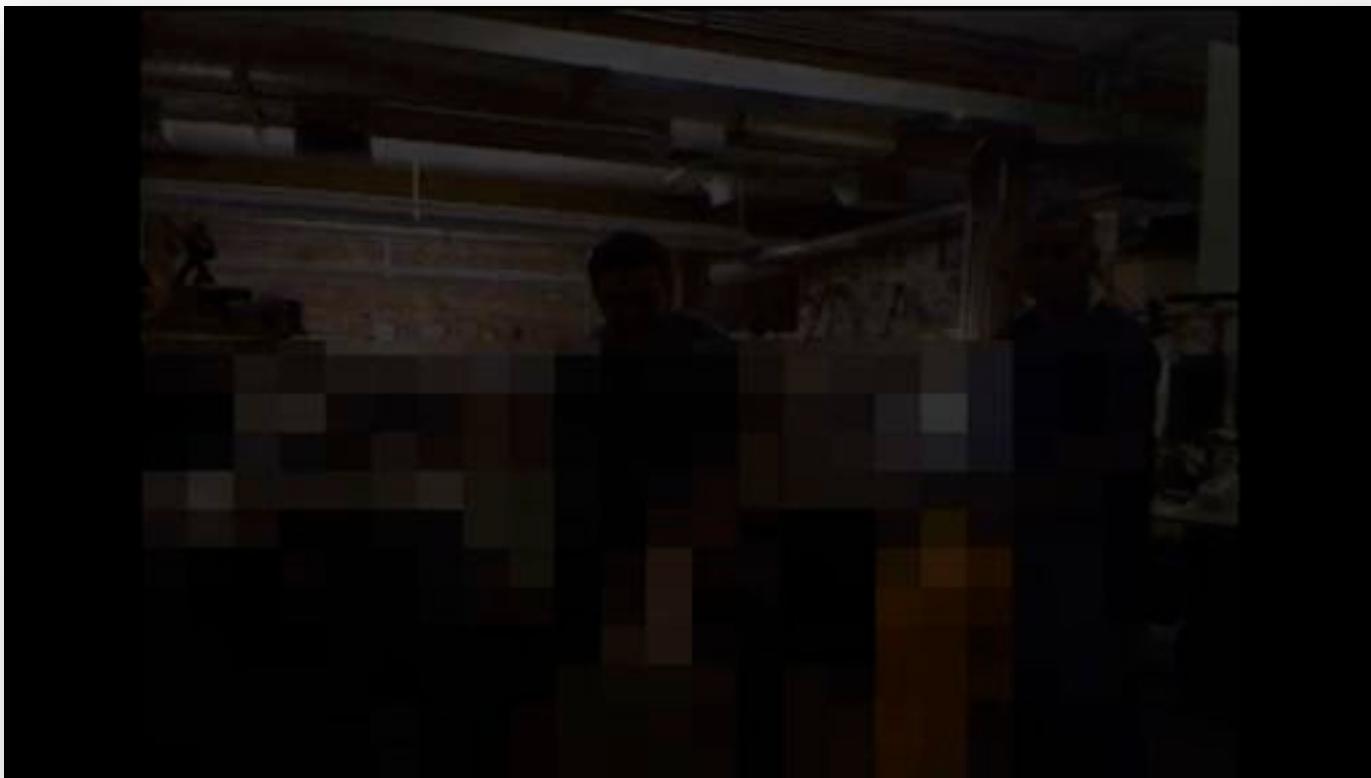


make_new_activity.py -n Forbidden



`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

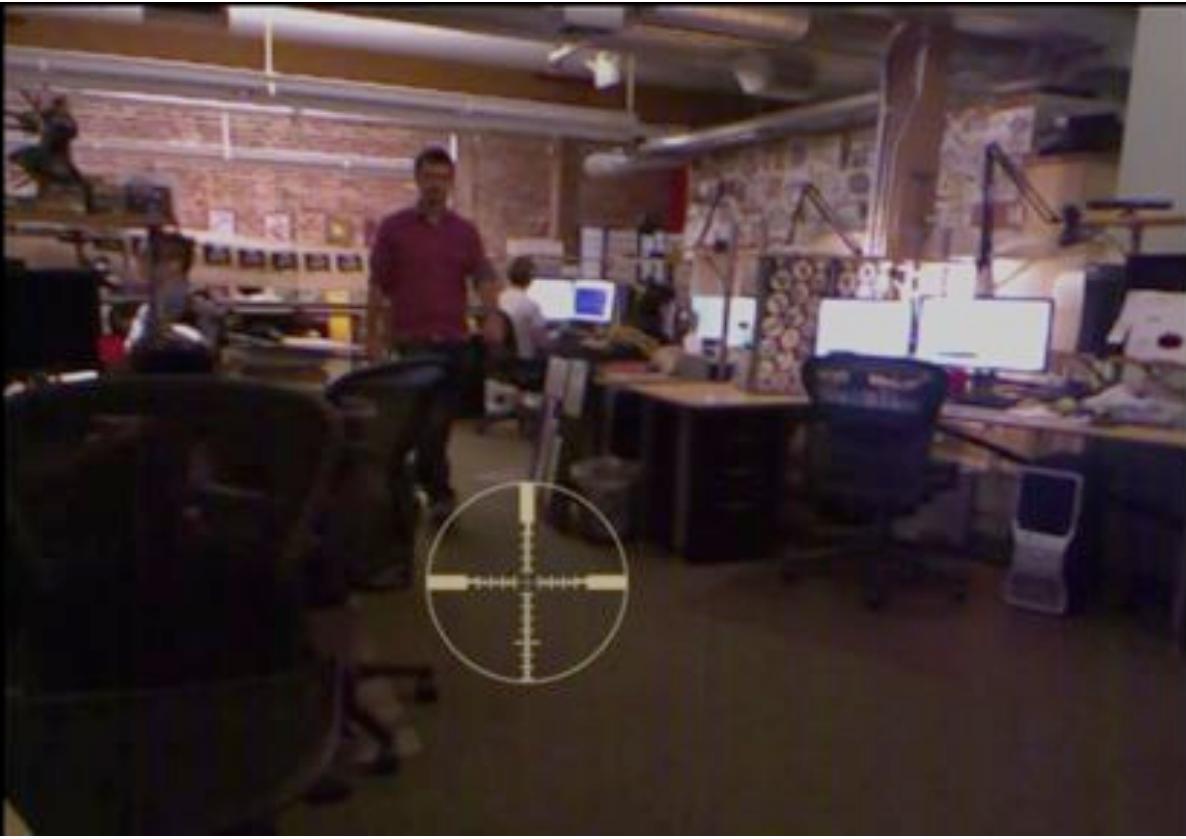


HATERS GONNA HATE

```
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
22
23     IEnumerator Start () {
24
25         yield return Application.RequestUserAuthorization(UserAuthorization.WebCam | UserAuthorization.Microphone);
26
27         if (Application.HasUserAuthorization(UserAuthorization.WebCam | UserAuthorization.Microphone))
28         {
29             webcamTexture = new WebCamTexture(width,height,30);
30             data = new Color32[width * height];
31             webcamTexture.Play();
32
33             textureStack = new Texture2D[textureCount];
34             for (int i = 0; i < textureCount; i++)
35             {
36                 textureStack[i] = new Texture2D(width, height);
37             }
38
39             // Update is called once per frame
40             void Update () {
41
42                 if (!webcamTexture)
43                 {
44                     Debug.Log ("Waiting for Webcam...");
45                     return;
46                 }
47
48                 UpdateFrameDelay();
49
50                 webcamTexture.GetPixels32 (data);
51
52                 textureStack[writeID].SetPixels32 (data);
53                 textureStack[writeID].Apply();
54                 readID = (writeID + (textureCount - frameDelay)) % textureCount;
55                 renderer.material.mainTexture = textureStack[readID];
56
57                 writeID++;
58                 writeID = writeID % textureCount;
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     }
68 }
```

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**





- (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 () {
    lastDeltaTime = 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

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

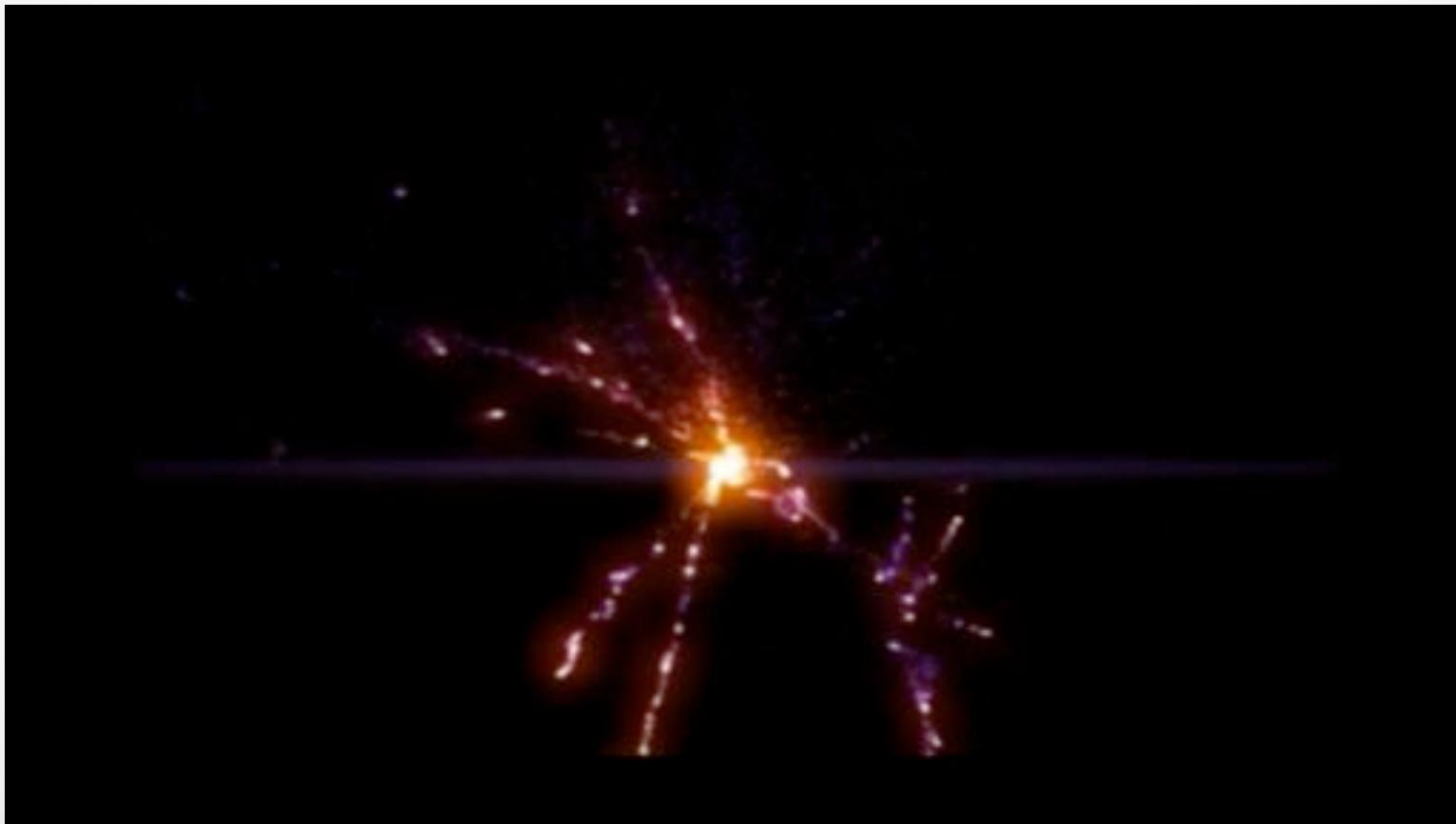
float _time;
float _deltatime;
float _lastdeltatime;
float3 _origin;
float _vertexCount;
float _drag;
float _temperature;
float _mag;
float _worldscale;

[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];
                float3 diff = pos2 - pos;
                float3 dir = normalize(diff);
                float3 cross = cross(dir, orig_accel);
                float3 temp = cross * 0.001f;
                accel += temp;
            }
        }
    }
}
```

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

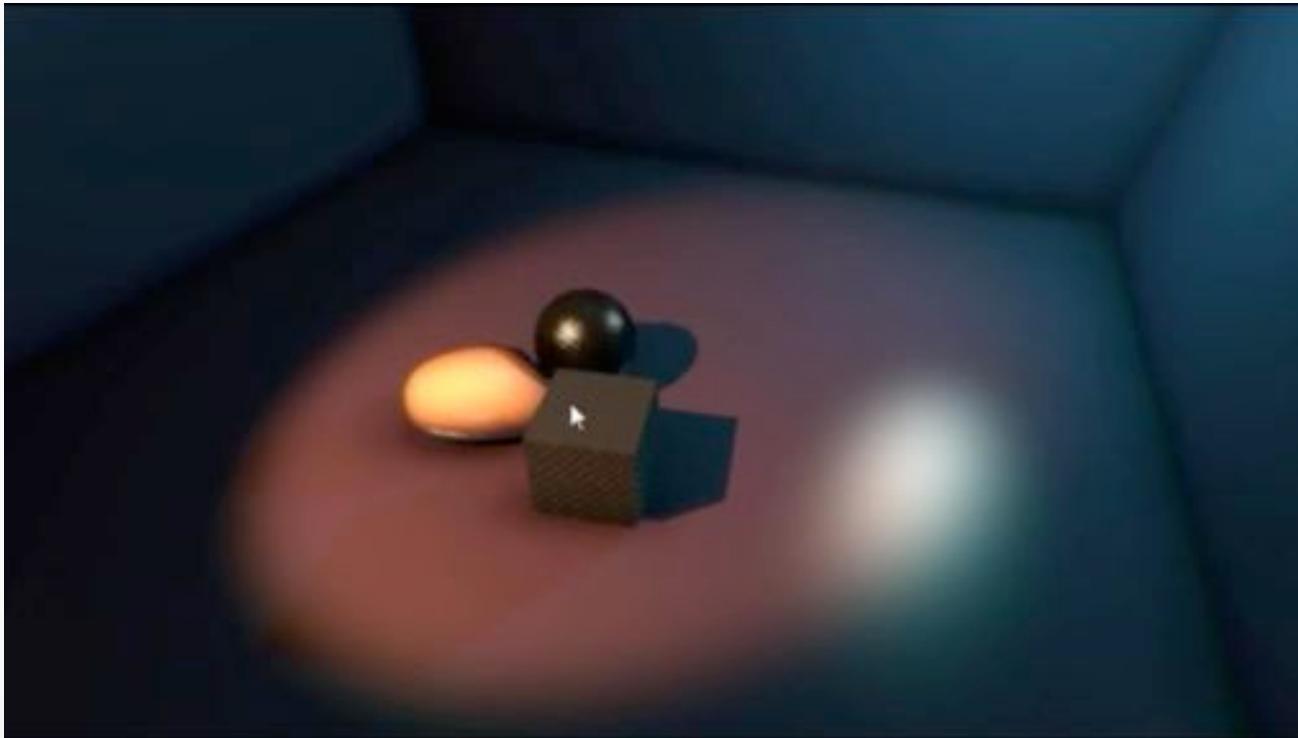
```
[numthreads(32,32,1)]
void CSMain( uint3 Gid : SV_GroupID, uint3 BTid : SV_Dispatcher
{
    uint uniqueID = BTid.x + BTid.y*(128*32) + BTid.z * (8);
    float3 pos = buf_Points[uniqueID];
    float3 previous_pos = buf_PreviousPoints[uniqueID];
    buf_PreviousPoints[uniqueID] = pos;

    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 nos2 = huf_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

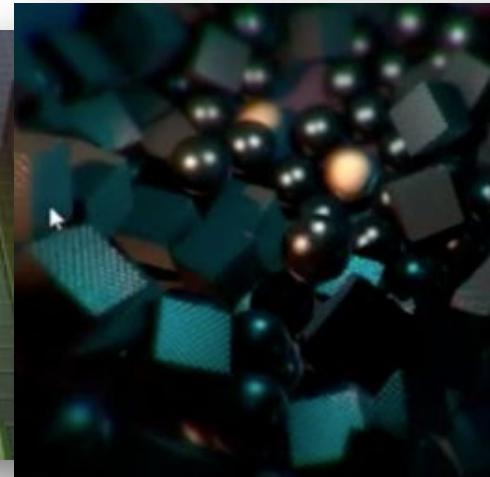
- Build confidence in new ideas, and share them**

- 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



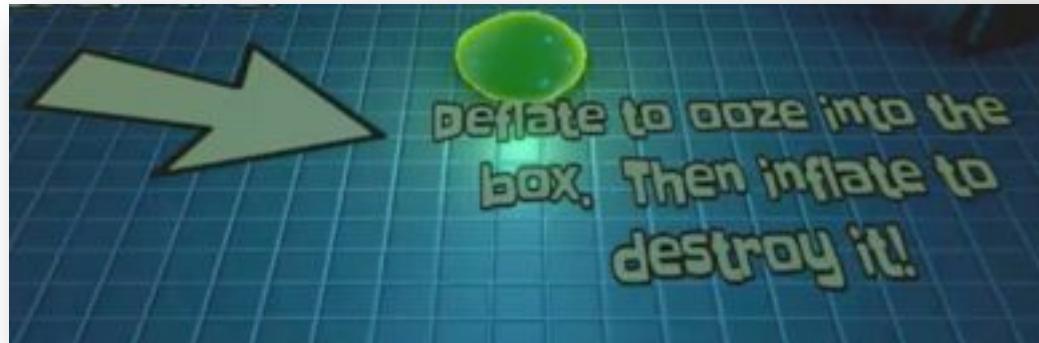
Rapid Prototyping Payoffs

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

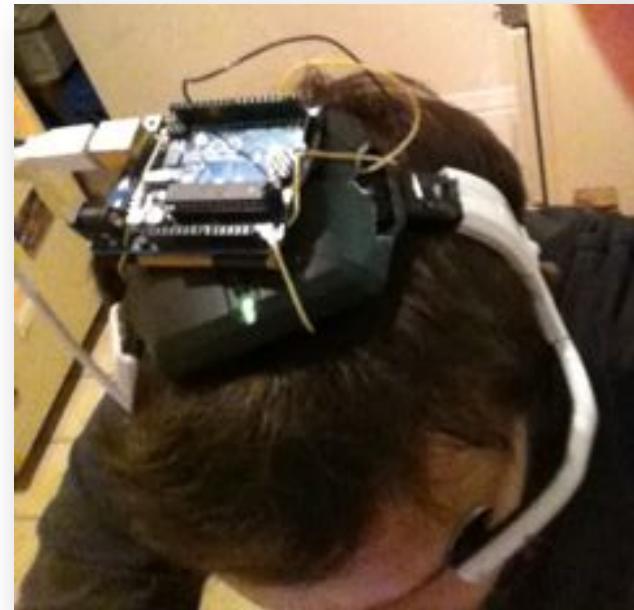


Rapid Prototyping Payoffs

- 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**

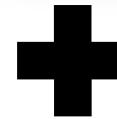


Rapid Prototyping PS4



Rapid Prototyping PS4

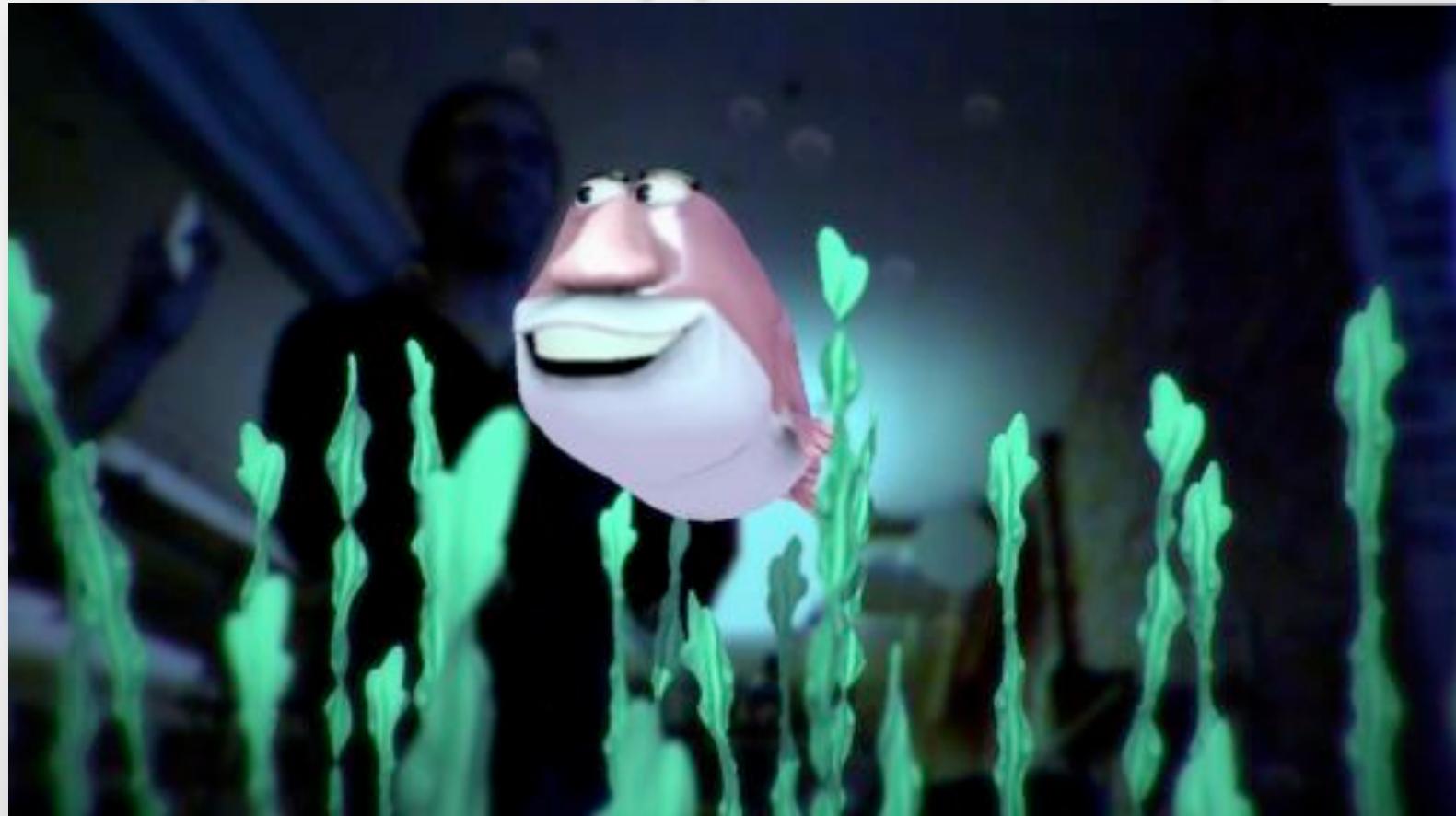
- Faking it!



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



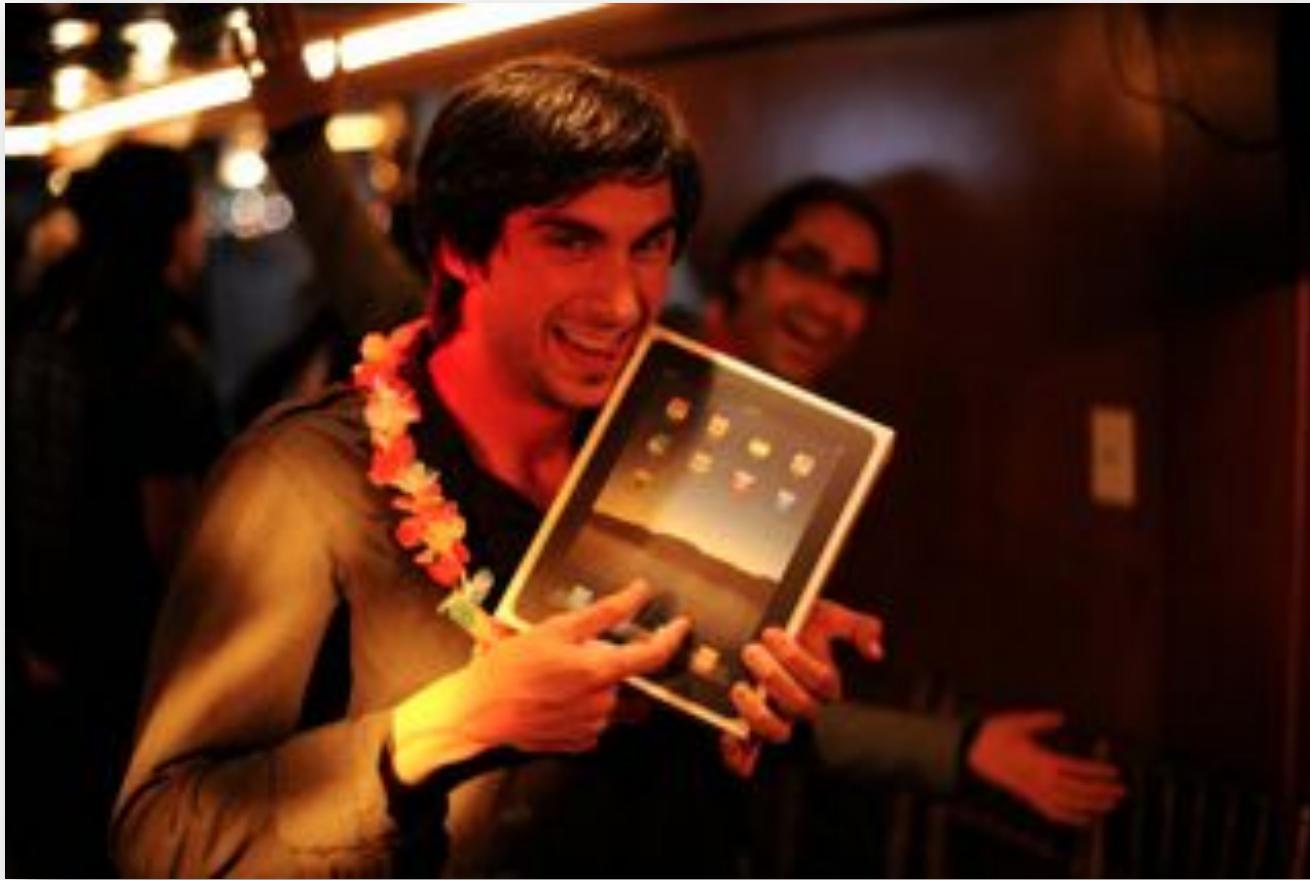
Leap Prototype





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



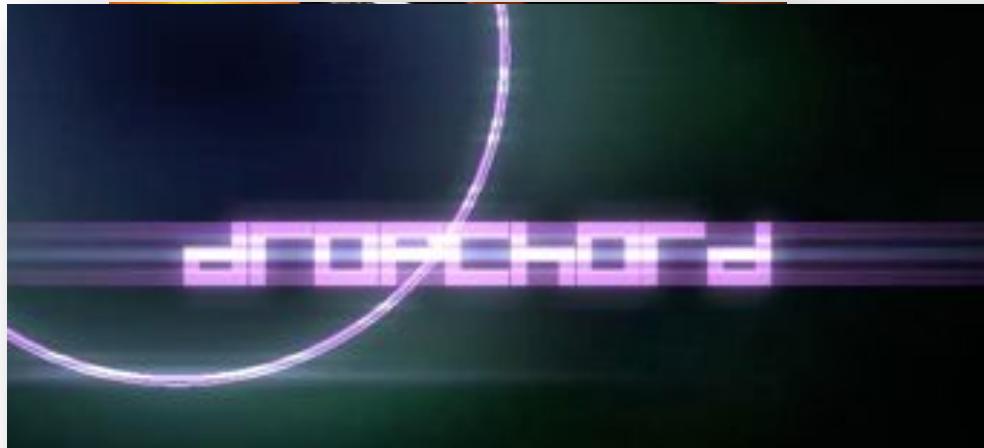
Rapid Prototyping Payoffs

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

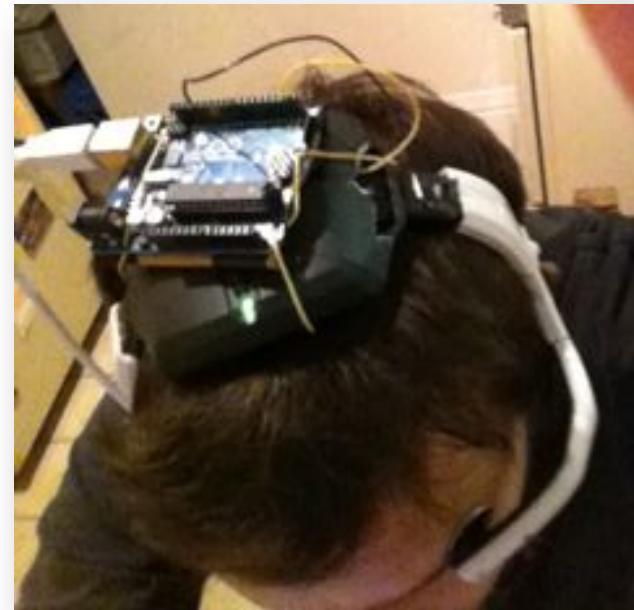


Rapid Prototyping Payoffs

- 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



A screenshot of the Arduino IDE interface. The title bar says "BrainGenOut | Arduino 1.0.3". The main window shows the following code:

```
BrainGenOut
// Arduino Brain Library
// Version 1.0.3, compatible with Arduino 1.0.3
// Eric Mao, 2008

BrainGenOut.h
// Set up the serial port, 38400, to the hardware serial object you want to listen
// to.
// Note: on the teensy power+, 38400 is the hardware serial object you want to listen
// to.

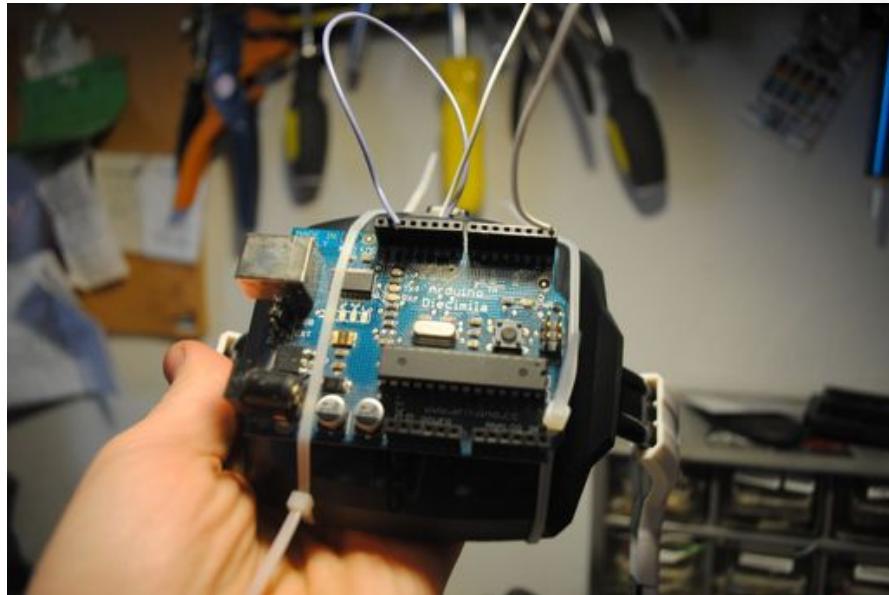
void setup() {
    // Set up the hardware serial.
    Serial.begin(38400);
}

void loop() {
    // Expect packets about once per second.
    // readString() function expects a string (well, char*) listing the words
    // received. It returns the number of characters, whitespace, delimiters, low bytes, high
    // bytes, etc.
    if (Serial.available() > 0) {
        String str = Serial.readString();
        Serial.print("Received: ");
        Serial.println(str);
    }
}
```

The status bar at the bottom right corner of the IDE window displays the text "Arduino Uno or Leonardo (atmega328)".

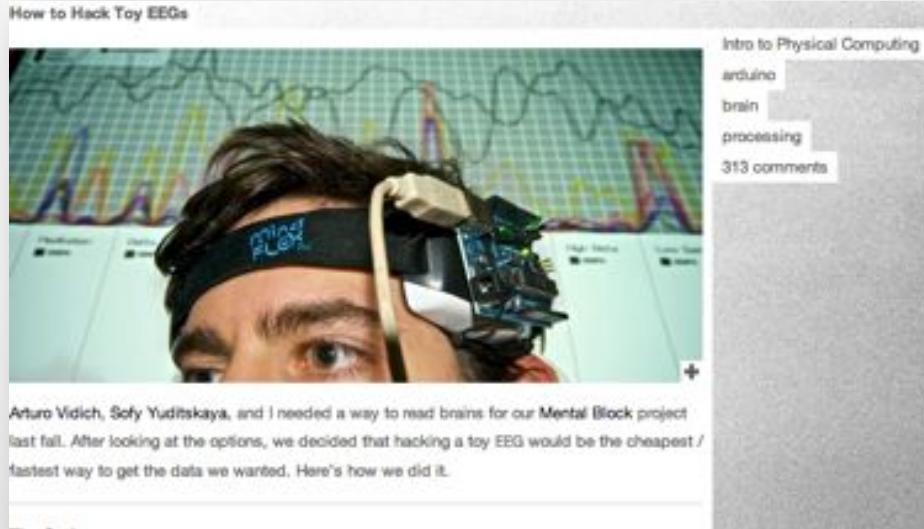
BRAIN TRACKING!

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



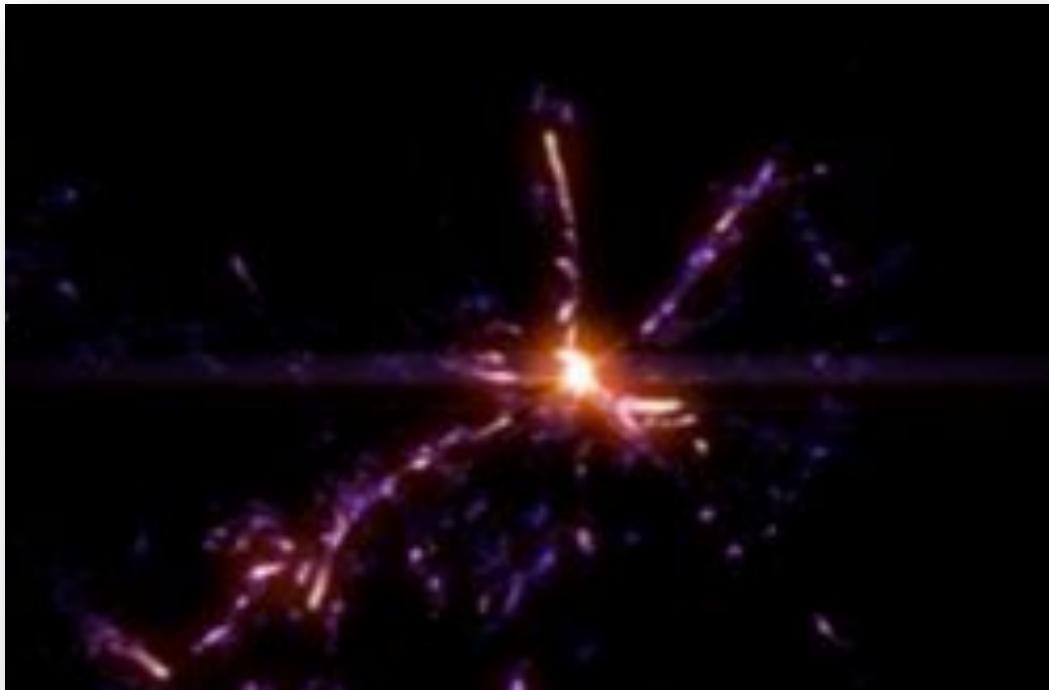
Arduino Tutorials are Great

- Force Trainer + Arduino Uno



Hrmmm.... What Next

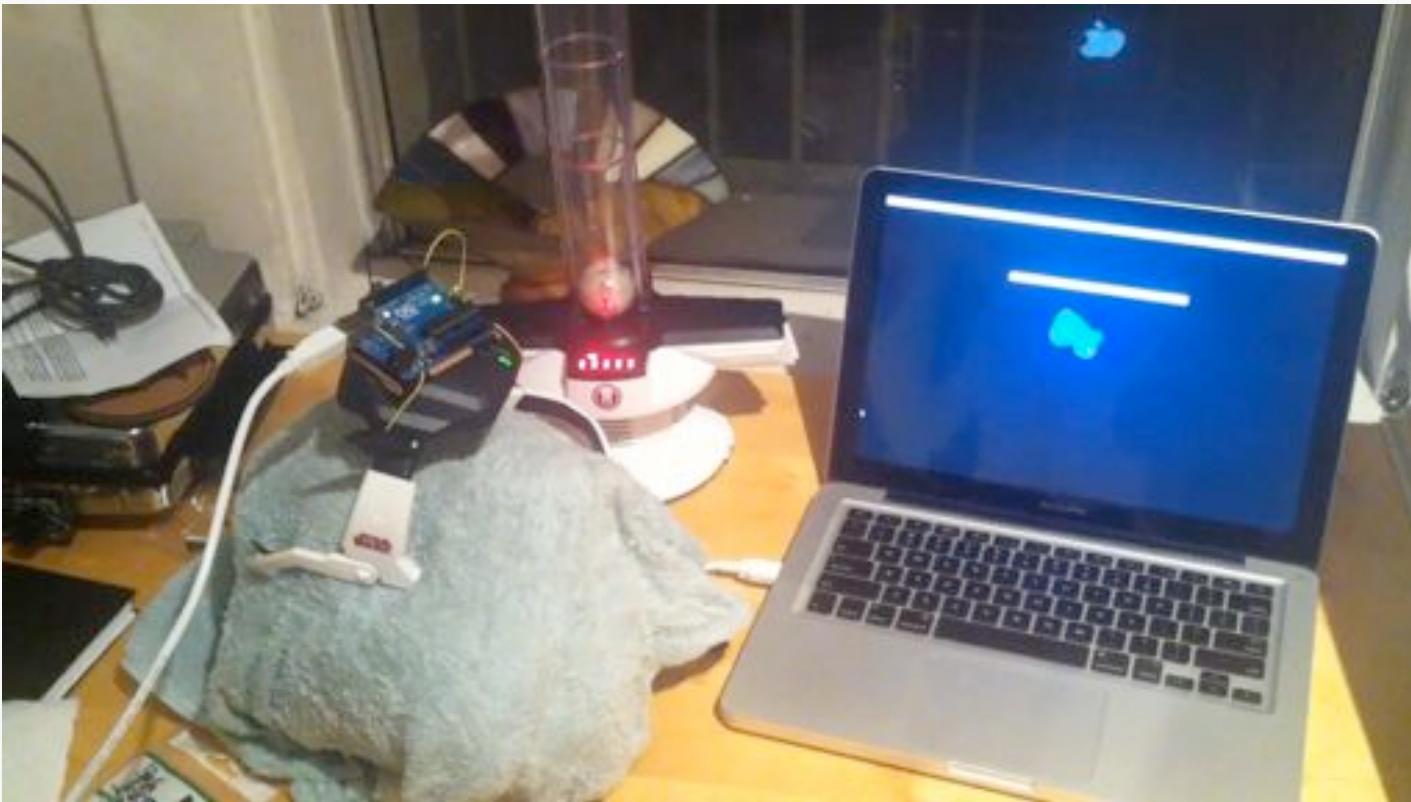
- Obviously we *MUST* to wire this up to a particle system...



Arduino Prototype: Brain Particles

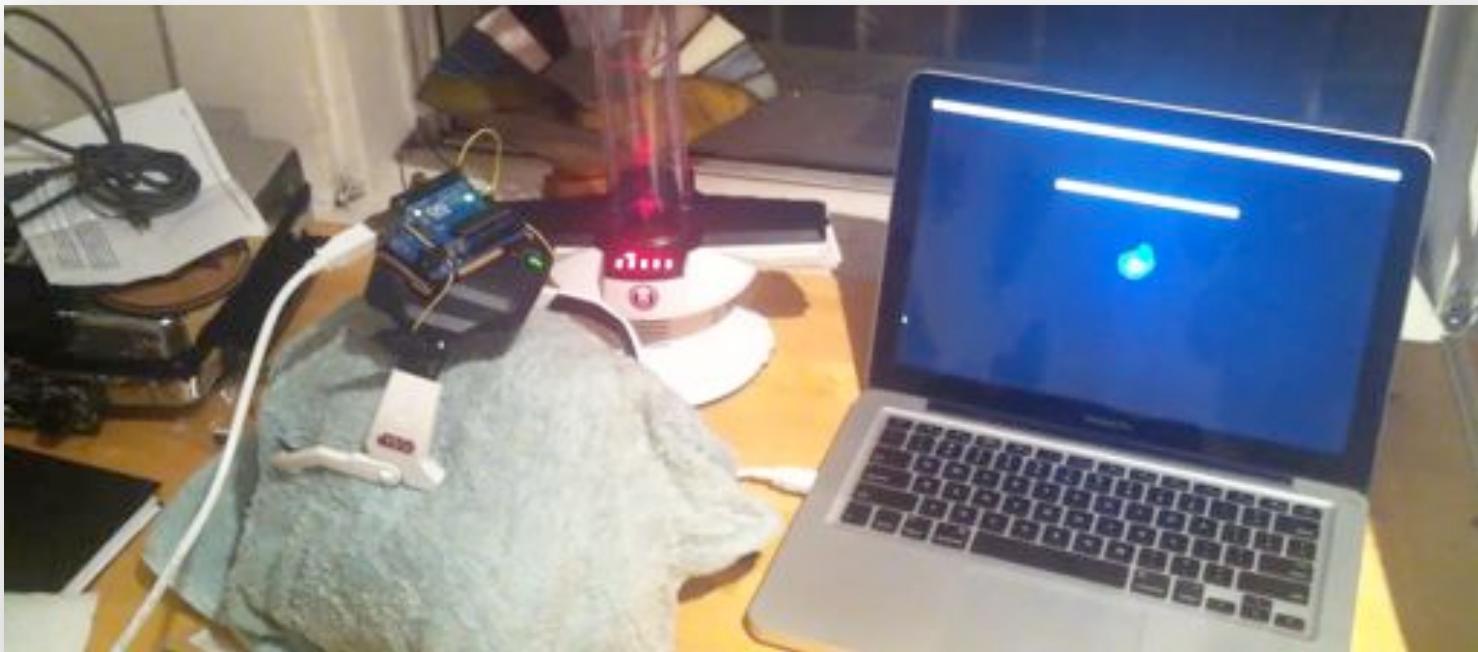


Wet Towel Midichlorians



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_freenect
 - http://thomasdiewald.at/processing/libraries/dLibs_freenect



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



Thanks!

A large, diverse crowd of people is gathered outdoors at night, filling the frame. They are looking upwards towards a massive digital screen that spans the width of the image. The screen displays the word "Thanks!" in a large, white, sans-serif font. The background of the screen is a dark orange gradient. The scene is filled with numerous small, glowing yellow and orange particles, resembling confetti or fireworks, which are falling from the top right and spreading across the scene. In the foreground, there are blurred lights and streaks of light, suggesting motion and a festive atmosphere.

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