Universidad de Chile
Facultad de Arquitectura y Urbanismo
Escuela de Pregrado
Carrera de Diseño

“La Anatomía De Las Emociones”

Ensayos para la visualización de gráfica digital con herramientas de desarrollo e interfaz gestual.

MEMORIA PARA OPTAR AL TÍTULO DE DISEÑADORA GRÁFICA

Autora:
Constanza Daniela Prado Durán

Profesor guía:
Mauricio Vico Sánchez

Santiago de Chile, Enero de 2013.

Santiago de Chile, Enero de 2013.

La anatomía de las emociones : Ensayos para la visualización de gráfica digital con el uso de la interfaz gestual

Introducción

Este documento corresponde a una documentación de ensayos experimentados con el dispositivo Kinect durante el segundo semestre del año 2012. La experimentación consistió en el proceso de desarrollo de interfaces gráficas naturales (NUI) para el uso de este dispositivo. Es así como este proyecto se manifiesta en dos formas: una, en la actuación (performance visual) que exhibe un ejercicio con el dispositivo; y la segunda, en ésta documentación del proceso de trabajo.
En este documento, los ejercicios se encuentran diferenciados por dos etapas de proceso : "Experimento parte 1" y "Experimento parte 2". Se diferencian así pues cada una responde a distintos objetivos.
La primera parte está enfocada a familiarizarse con las herramientas de desarrollo (IDES o entorno de programación) y la segunda, en el desarrollo de la performance y todas las variables de montaje que implica su producción.
Para cada etapa de los ejercicios se realiza una sistematización de la información con el fin de visualizar la evolución del trabajo. Todo esto realizado bajo mi punto de vista. Esto fue parte de un desarrollo personal de trabajo.

El desarrollo de esta documentación va dirigida hacia las personas que se quieren familiarizar con el uso del dispositivo Kinect, o que tengan algún tipo de interés en introducirse a las IDES. La idea es ir reformulando, corrigiendo y mejorando el documento.

Recursos utilizados

Los recursos utilizados para la experimentación fueron los siguientes:

-Computador Mac Os X (se ocuparon dos computadores con el mismo sistema operativo pero dos versiones Snow Leopard 10.6 y Mountain Lion 10.7)
-Procesador: 2.4 GHz Intel Core i5
-Tarjeta de video integrada y aparte.
-Dispositivo Microsoft Kinect Xbox 360
-Parlante activo marca Alto de 250watt MRS, el sistema de audio de la tienda es un line array marca Lexsen y consola digital Presonus, mas microfonía completa, monitoreo y accesorios.

- Proyector 3500 ANSI lúmenes marca DELL
- Memoria extraíble
- Telón de 2*3 metros.

Bibliografía Recomendada:

- "The Nature of Code" , Daniel Shiffman
- "Kinect Hacks Tips & Tools for motion and pattern detection", Jared St.Jean, Editorial O`Reilly
- "Programming Interactivity", Joshua Noble, Editorial O`Reilly
- "Processing", Casey Reas y Ben Fry, The MIT Press
- "Learning Processing", Daniel Shiffman
- "Meet the Kinect"; Sean Kean, Jonathan Hall, Phoenix Perry
- "Making Thigs See" Greg Borenstein, Editorial O`Reilly

Experimento parte 1

1.1 Sistematización de ensayos

a) Relación objetivos, requerimientos y etiquetas

Durante esta primera etapa el fin fue aprender a manejar a un nivel básico las herramientas de desarrollo (IDES) que lograrán conectarse con la Kinect.
Como esta etapa fue diversa y amplia en información, se realizó una sistematización de ensayos.
Este sistema funciona mediante tres elementos: Objetivos, Requerimientos y Etiquetas. (ver Tabla 1)

Los objetivos son los propósitos técnicos a desarrollar en esta etapa, qué es lo que quiero lograr. Los requerimientos es cómo llevaba a cabo cada objetivo técnico.
La etiqueta sirve para nombrar cada objetivo.
Por ejemplo, el objetivo número uno (ver Tabla 1) fue definir cuáles son las formas para reconocer el cuerpo con el dispositivo Kinect, eligiendo así el mejor método. Para llevar esto a cabo (requerimientos) tuve que conectar el dispositivo Kinect con cada herramienta. El nombre de la etiqueta que le puse a esta etapa se llama "Reconocimiento corporal con dispositivo Kinect". Y así va con el resto de los objetivos y etiquetas.

b) Registro de ensayos

Cada ensayo contiene siete piezas que lo registra. En este proyecto, el registro sirve como bitácora. Permite retomar, recordar y validar cada ensayo como potencial para futuros proyectos.

1. Etiqueta
Como se menciona anteriormente, las etiquetas se hacen a modo de tabulación de datos y para la facilitación del acceso a la información. Cumplen también con responder a los requerimientos, y estos requerimientos a su vez, son respuesta a los objetivos de esta etapa.

2. Definición
Todos los ensayos realizados deben tener una definición, pues los contextualiza, diferenciándolos del resto.

3. Fecha de realización
Forma parte de la ficha técnica con el fin de observar la evolución del proyecto en base al factor tiempo.

4. Software
Este dato se refiere a las interfaces de programación de aplicaciones (API), las aplicaciones (APP) y los entornos de desarrollo (IDE) que se utilizaron. Las APP corresponden a la colección Adobe y los de modelado 3D: Cinema 4d (C4D) y 3D Max, de la empresa Maxon y Autodesk respectivamente. Su rol corresponde a crear la gráfica digital que se importa a las IDE. En esta instancia se conectó el dispositivo Kinect con tres software: 3dUnity, Quartz Composer (QC) y Processing.
Processing es un entorno de programación (IDE) que ha sido empaquetado como un programa de aplicación, consiste en un editor de código, un depurador, un constructor de interfaz gráfica (GUI) y un compilador. Funciona con el lenguaje de programación JAVA.
Mac OS X incluye un conjunto de tecnologías de gráficos que otorgan las herramientas para construir experiencias visuales, (herramientas de desarrollador, developer tools). Para esto Quartz Composer es un ambiente de programación visual, incluido en Xcode, que permite crear composiciones de gráficos, sin la necesidad de escribir códigos. Simplemente se conectan bloques de funciones visuales programadas se puede diseñar visualizaciones que por ejemplo, combinan imágenes y en tiempo real información sobre los canales de video. Después de armar la composición, se puede utilizar el enlace hacia Cocoa para hacerla aplicación.

5. Librerías Ocupadas [1]
Para los ensayos que las tengan, tiene la finalidad de poder registrar que paquete de librería ocupados. Esto permite identificar las actualizaciones necesarias que hay que hacer en cada ejercicio. Las librerías van cambiando con el tiempo, es necesario ir actualizándolo.

1 No todos los ensayos ocupan librerías. Processing puede ocupar librerías, como no. Quartz Composer utiliza patches y plug ins.

Objetivos	Requerimientos	Etiquetas
1. Definir cuáles son las formas para reconocer el cuerpo con el dispositivo Kinect, escogiendo el mejor método.	1. Conexión dispositivo Kinect con cada herramienta de desarrollo, con distintos métodos.	**1.Reconocimiento corporal con dispositivo Kinect**
2. Experimentar cuál es el mejor formato para visualizar elementos gráficos no animados y animados externos a la herramienta de desarrollo.	2.1 Importación de elementos gráficos en 2d animados y no animados 2.2 Importación de elementos gráficos en 3d de distintos formatos y usos.	**2.1 Elementos gráficos 2d** **2.3 Elementos en formato 3d**
3. Comprender en un nivel básico la construcción de elementos gráficos parametrizables creados en la herramienta de desarrollo	3. Creación de ejercicios en base a proyectos ya alojados en la web y la comprensión de éstos para su manejo	**3.Elementos gráficos parametrizables.**
4. Explorar la capacidad de interactividad en los siguientes dispositivos como estímulos: el mouse, el micrófono y cámara web integrada al computador portátil,; y su respuesta expresada en una representación visual.	4. Realización de ejercicios que reaccionen visualmente a sonido, movimiento del mouse en plano X e Y, y a elementos capturados por la cámara web.	**4.Elementos interactivos**

Tabla 1: Definición de objetivos, requerimientos y etiquetas.

6. Observaciones[2]

Las observaciones son claves en el proyecto, permiten concluir los ejercicios, evaluando sus características técnicas. Estas características son resultado de textos consultados, conversaciones con colaboradores y/o la propia experimentación.

7. Códigos ocupados

El ensayo puede no tener códigos. Cada ejercicio de la primera etapa incluirá los códigos y el lenguaje utilizado en cada ejercicio, el fin es registrar y compartir lo aprendido. Cada código tiene una numeración la cual sirve para buscar en las referencias del código que están en los anexos. Si utiliza Quartz Composer se dirá “no tiene códigos, si nodos” , se dice esto porque el nivel que se maneja este programa es en nodos, por lo cual se adjuntarán todos los archivos.

Ejercicios	Varibles o datos
1. Ejercicio 1	1. Etiquetas
	2. Definición
	3. Fecha de realización
	4. Software Ocupados
	5. Librerías ocupadas
	6. Observaciones
	7. Códigos ocupados

Tabla 2: Elementos de registro

2 Las observaciones se encuentran debajo de todas las tablas. En algunos ensayos se puede observar que hay secciones muy “técnicas” las cuales poseen conceptos duros y propios del lenguaje informático.

El orden de lectura de cada ensayo es el siguiente: primero la tabla que nos da los datos. Seguido a eso y de manera separada a la tabla, se describe el punto 6, el de las observaciones.

1.2 Muestra de ensayos

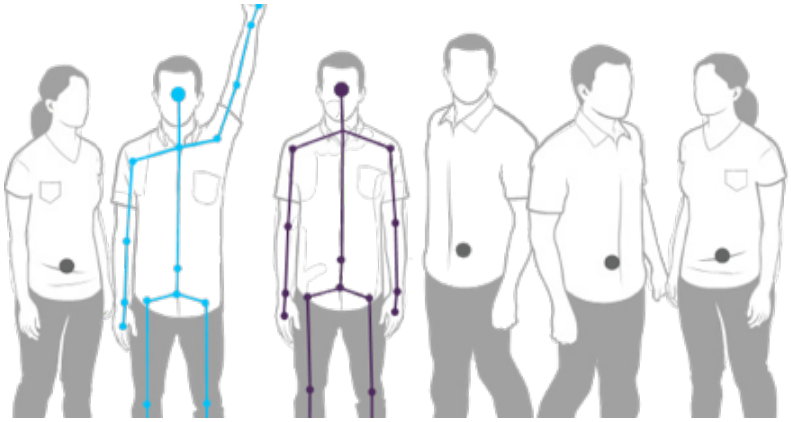
1.2.1 Ensayos con Etiqueta: Reconocimiento corporal con dispositivo Kinect.

El reconocimiento corporal es realizado en base a los siguientes programas: 3d Unity 3.5.6f4 (puede ser la versión pro y la normal, acá se ocuparon ambos en distintos tiempos), Quartz Composer Version 4.0 (103.1) y Processing 1.5.1.

Observaciones Preliminares

Una vez instaladas estas librerías se puede comenzar a experimentar. Sin embargo, inicialmente es recomienda tener en consideración las condiciones técnicas propias del dispositivo no logra captar mas de seis personas que estén en su área de visión. De estas seis personas, logra trackear (es decir reconocer sus articulaciones) en solamente dos. También existe la condición de que al captar la profundidad por defecto, la distancia varíe desde 0.8 metros hasta los cuatro metros. Esto quiere decir que manteniendo un límite físico, la imagen en profundidad se puede ver entre ese rango, a menos que se utilice la versión virtual extendida que va más allá de los límites físicos, pero eso considera que se reduce su resolución y su precisión.

Solo dos personas pueden ser trackeradas y calibradas. Fuente: http://msdn.microsoft.com/en-us/library/jj663791.aspx



Otra consideración que hay que especificar es que los ensayos o ejercicios expuestos en este documento no son la totalidad que se ejecutaron. En esta sección seleccionamos los que pueden de alguna manera representar los ejercicios cruciales de cada etapa que determinaron la metodología y software ocupados para la segunda parte del experimento.

1.2.1.1 Lo Primero es lo primero: instalaciones

Para iniciar la experimentación con este dispositivo se deben tener las plataformas, plugins, patches, y librerías instaladas en tu computador. Para esto, es necesario entender qué es lo que realmente sucede en el computador cuando se instalan estas dependencias. En este caso, se instaló la plataforma Open NI.

1.2.1.1.1 Plataforma OpenNI

Interacción natural: El término interacción natural (NI) se refiere al concepto donde la interacción del humano con el dispositivo está basado en sentidos humanos, mas que nada focalizado en audio y visión, dejando atrás los dispositivos como controles remotos, teclados y mouse. Un ejemplo puede ser cuando los dispositivos reciben instrucciones vía comandos vocales. También puede ser mediante gestos de manos, los cuales pueden estar programados para activar diversas funciones, y por último el utilizado por nosotros : seguimiento (tracking) de movimiento del cuerpo. Cuando un cuerpo es trackeado completamente por cualquier finalidad que este tenga.

OpenNI permite conectar los sensores que envían datos en bruto desde el dispositivo Kinect hacia un middleware que los analiza. Seguido a esto, los datos analizados son enviados hacia otro middleware que se ocupará de ellos para finalmente enviarlos al nivel más alto de datos de la aplicación. Esta arquitectura modular le permite interconectar diferentes middleware o dispositivos que cumplen con la API definida por Open NI.
OpenNI significa literalmente “Interacción abierta natural”. La interacción natural se refiere a la interacción con un dispositivo tecnológico que no depende de por ejemplo un mouse o un teclado. El objetivo es interactuar con los aparatos tecnológicos de

una manera similar a lo que lo hacemos los humanos. Así es una plataforma o sistema (framework) multilenguaje, multiplataforma que define APIS para escribir APP usando la Interacción natural. Una plataforma o framework es la que representa una arquitectura de estos software.
OpenNI permite la comunicación con el dispositivo Kinect, permite reconocer el cuerpo y calibrar el cuerpo para poder hacer el tracking. Lo hace de manera automática, una vez que "instancias" a la plataforma.
Open NI (Open Natural Interaction) Las APIS que utilizan OpenNI están compuestas por un set de interfaces para escribir APP de interacción natural. El principal propósito de OpenNI es formar una API standard que permita la comunicación entre ambos:

1)sensores de visión y audio (dispositivos que escuchan y ven las figuras y su alrededor)
2)el middleware de la visión y el audio: los componentes del software que analiza los datos del audio y la visualidad que son grabados de la escena y los comprende. Por ejemplo, el software que recibe datos visuales, como una imagen, devuelve la locación de la palma de la mano detectada sin la imagen.

La API estándar de OpenNI permite que los desarrolladores de aplicaciones de interacción natural realicen el seguimiento de escenas 3d de la vida real al utilizar tipos de datos que son calculados desde el sensor, por ejemplo, la representación de un cuerpo entero, representación de la locación de una mano, un conjunto de pixeles en un mapa de profundidad, etc. Las aplicaciones pueden ser escritas sin importar el proveedor del sensor o de los middleware.

Instalación de Plataforma Open NI

El método que funcionó en ambas versiones del Mac OSX serán explicadas a continuación, independiente de eso, existen distintos métodos de instalarlo, incluso uno de una manera más fácil y automática pero no resultó en ambas versiones.

"Fase uno: Cinco pasos" (tabla 3)

Los pasos son sencillos si existe conocimiento previo del Terminal. Seguido a eso viene la Fase Dos, la cual es la de instalación tres actores, la primera es la plataforma Open NI (Open Natural Interface), el segundo es el Sensor Kinect y tercero Middleware NITE (en ese orden).En el paso uno y en el tres, lo que se hace es llamar a dos librerías. La librería (libusb) es la que permite conectar aplicaciones de interfaz gestual con los dispositivos USB (en este caso el dispositivo Kinect). Por su parte, Libtool proporciona servicios de soporte generalizados para la compilación de librerías.

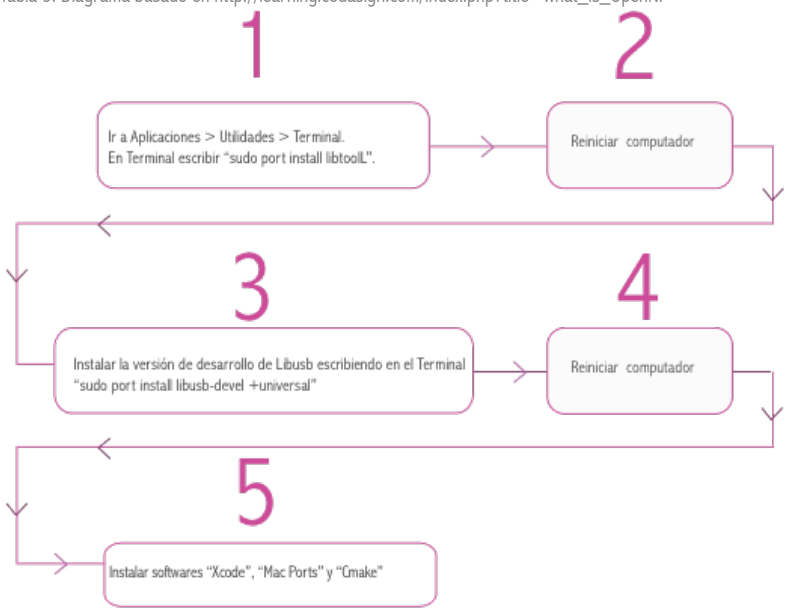
1.2.1.1.2 Instalación de dependencias

Se necesita crear un directorio para colocar todas las aplicaciones o dependencias indispensable para la utilización del dispositivo Kinect en el Mac Osx. En este proyecto se ocupó la carpeta "Kinect" dentro del Home.
Seguido a eso abre el terminal y escribe la siguiente dirección:

MKDIR ~/KINECT
CD ~/KINECT

Seguido a eso, se realizan los pasos del diagrama (ver tabla 4)

Tabla 3: Diagrama basado en http://learning.codasign.com/index.php?title=What_is_OpenNI



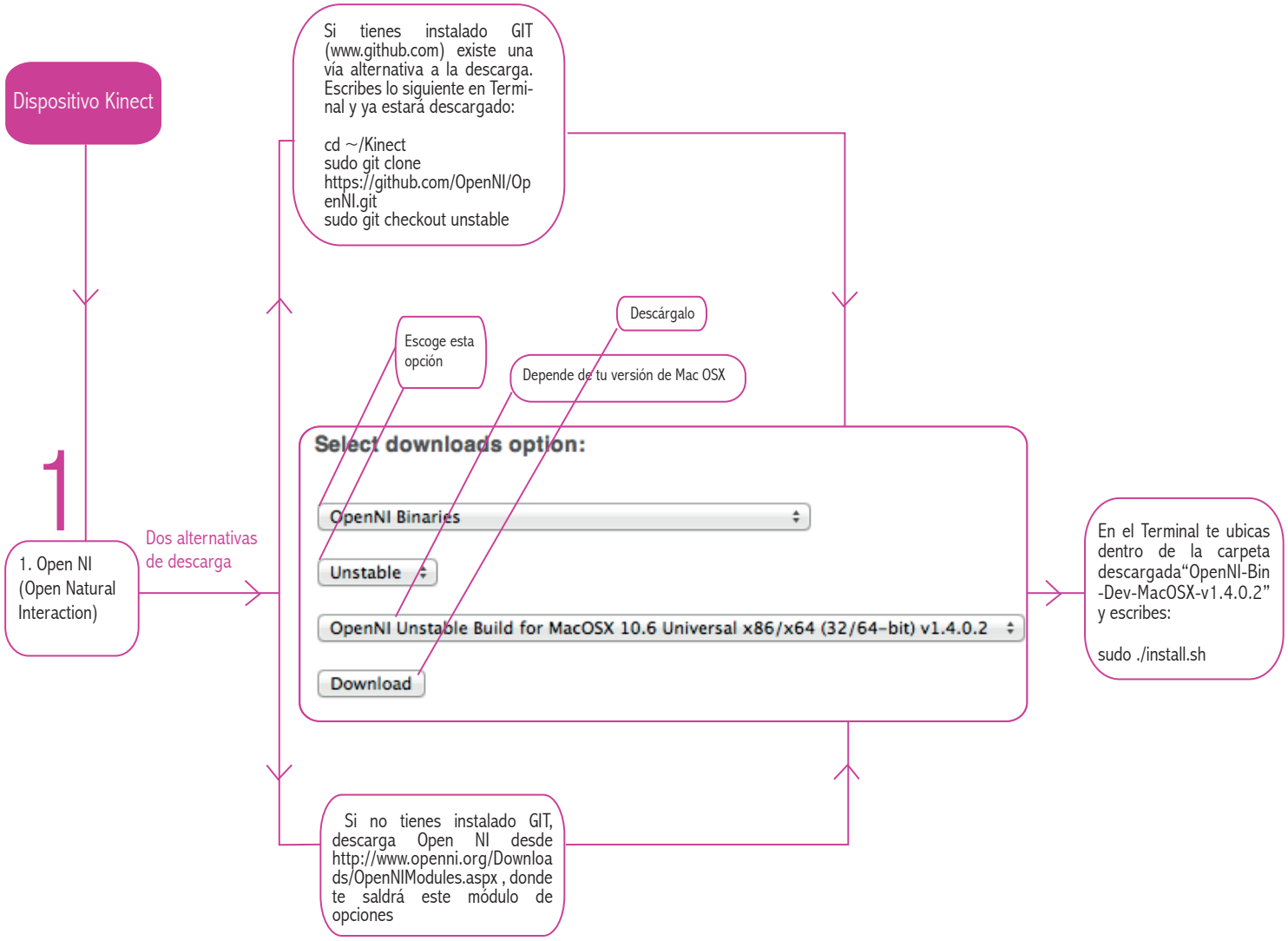


Diagrama instalación. Tabla 4

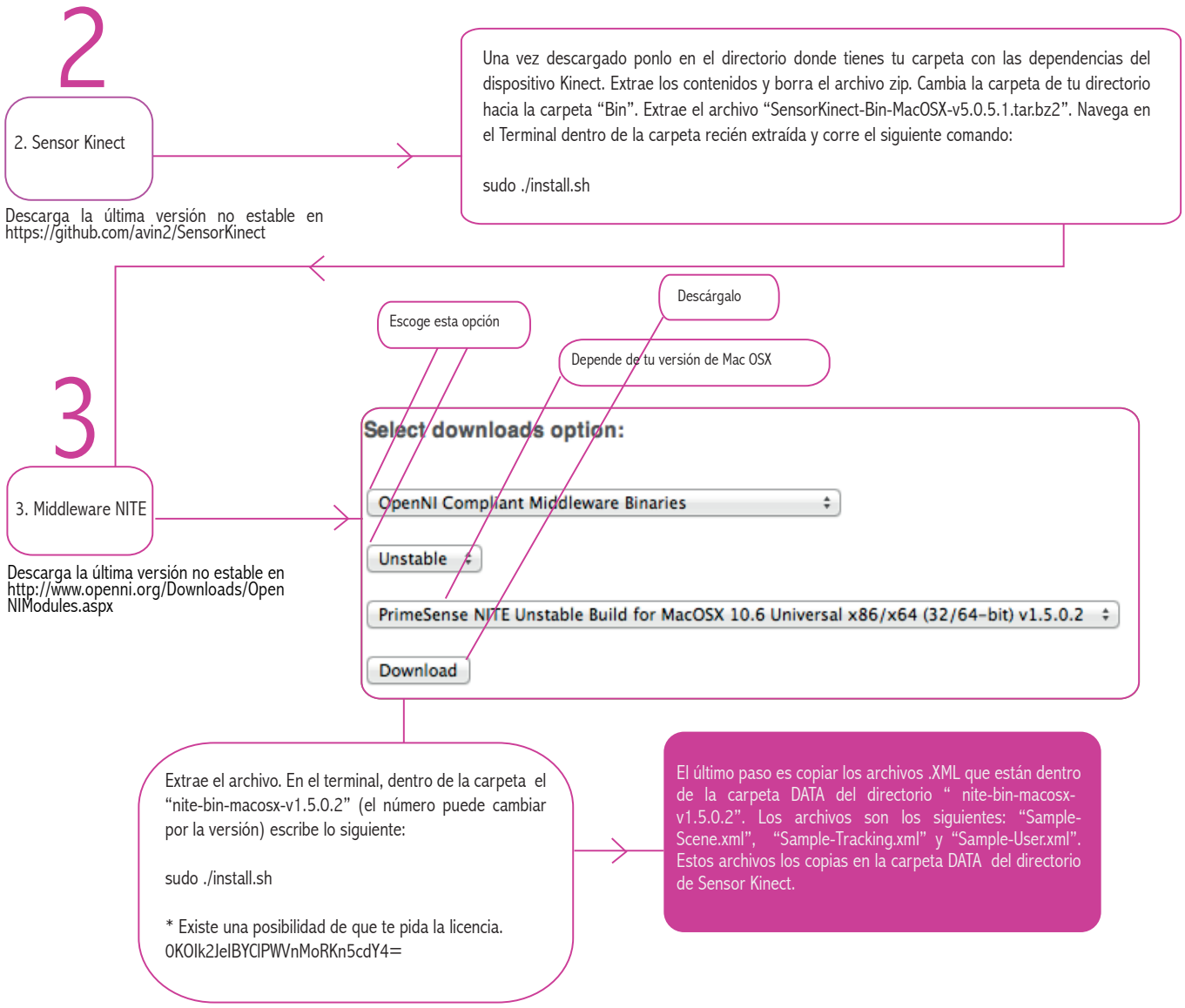
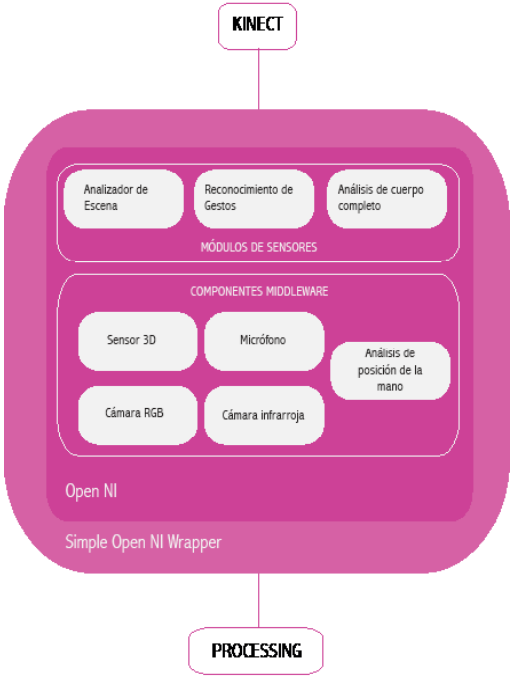


Diagrama instalación. Tabla 4

La librería Simple OpenNI define los distintos métodos para poder acceder a los datos que nos entrega en Middleware la plataforma OpenNI. Es un programa que controla el acceso a un segundo programa. En este caso, cada API que utilice para conectar el dispositivo Kinect es el que posee un envoltorio o wrapper.

Middleware NITE es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos. Fue creado por el desarrollador Prime Sense.

Diagrama basado en http://learning.codasign.com/index.php?title=What_is_OpenNI



Ejercicios

Observación ejercicio 1

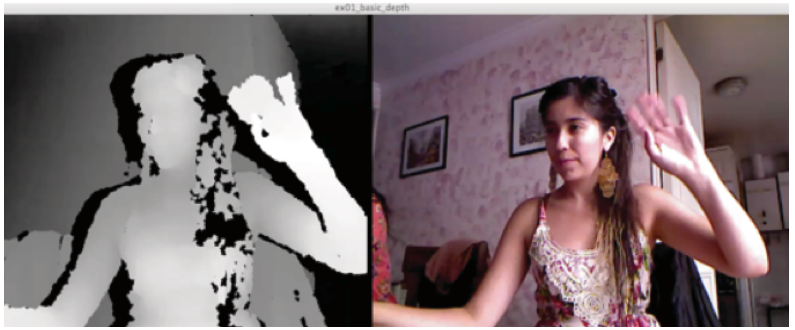
En este ensayo lo básico es lograr conectarse con el dispositivo Kinect: la profundidad de imagen (DEPH IMAGE). Este ejercicio se hizo en base a ejercicios realizados por Greg Borenstein (Making Things see, 2012). Se tiene acceso por primera vez a los datos "puros" enviados desde el dispositivo Kinect. Este ensayo fue pertinente pues explica cómo captura la volumetría de los objetos. Este código sirve como introducción a los elementos de conexión con el dispositivo. Se llama a la librería Simple OpenNI.

Los siguientes códigos son las variables que te permiten declarar que existe una variable llamada KINECT para los datos de la SIMPLEOPENNI y una variable que de la imagen de profundidad (DEPTH IMAGE), señalando que salga mediante una PIMAGE. En el SETUP defines que cuando declaras KINECT te refieres a los datos que te da la SIMPLE OPENNI (THIS) y después le pides a esta variable que permita el acceso a la profundidad de datos.

La siguiente línea es un arreglo de enteros, donde se guardan puntos los cuales determinan la profundidad de la imagen :

```
INT[] DEPTHVALUES = KINECT.DEPTHMAP();
```

La función IMAGE(DEPTHIMAGE, 0, 0); dibuja una imagen de los datos que uno recibe, declarando que está en la esquina de x=0 e y =0.



Ejercicio 1	Variables o datos
Control de Profundidad	1. Reconocimiento corporal dispositivo Kinect
	2. Reconocimiento de profundidad del dispositivo Kinect . El código ocupado está basado en ejercicios realizados dentro del texto "Making this see".
	3. Septiembre 2012
	4. Processing 1.5.1
	5. Simple OpenNi
	7.Código 1

Ejercicio 2	Variables o datos
Identificador de objetos en scene map	1. Reconocimiento corporal dispositivo Kinect
	2. Este código te permite identificar los objetos que están siendo mapeados por el dispositivo Kinect.
	3. Octubre 2012
	4. Processing 1.5.1
	5. Simple OpenNi
	7.Código 2

Observación ejercicio 2

En este ejercicio se logró conectar con la profundidad del dispositivo y asignarle color a los objetos que reconoce el dispositivo. Esto sirvió para observar como funciona el sensor de la profundidad, al ver como separa los objetos.
Por ejemplo, cuando el usuario se acerca a un objeto a una pequeña distancia, el dispositivo lo reconocerá como un solo ser, asignándoles el mismo color con al objeto con el usuario. De esta manera, si el usuario se alejaba, se creaba otro color, asignándoles colores para cada uno. Esto ayuda a lograr entender como podrían ser la distribución de elementos (la distancia entre ellos, del dispositivo y su posición) en una escena que esté siendo captada por el dispositivo.
La línea `sceneMap = kinect.sceneMap();` determina que `scene map` siendo un array de enteros, es la cantidad de objetos que puede llegar a detectar el dispositivo kinect.
El reconocimiento del cuerpo funciona, hay que tener en consideración que me reconoce como nube de puntos, reconociendo mi cuerpo pero no las partes. Una diferencia sustancial que se encuentra en este ejercicio comparado con el anterior, es que acá se llama a Open GL (Open Graphics Library)

```
import processing.opengl.*
```

Es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D.
Fundamentalmente OpenGL es una especificación, es decir, un documento que describe un conjunto de funciones y el comportamiento exacto que deben tener. Este ejercicio se hizo en base a ejercicios realizados por Greg Borenstein (Making Things see, 2012)

Ejercicios 3	Variables o datos
Píxel más cercano	1. Reconocimiento corporal dispositivo Kinect
	2. Reconocimiento de profundidad del dispositivo Kinect . El código ocupado está basado en ejercicios realizados dentro del texto “Making this see”.
	3. Septiembre 2012
	4. Processing 1.5.1
	5. Simple OpenNi
	7.Código 3

Observación ejercicio 3

El Píxel más cercano es el nombre asignado para este ejercicio porque precisamente lo que hace es eso, reconocer cuál es el píxel mas cercano al dispositivo Kinect.
A éste píxel le puedes asignarle cosas. En este ejercicio solamente le fue asignado un elemento básico: una imagen jpg. ¿Pero qué pasaría si fuese una acción? Una posibilidad podría ser hacer que se active un elemento interactivo que se active solamente en el pixel mas cercano, como un campo de fuerza que atrae al usuario dentro del dispositivo.

La siguiente línea define que los valores cercanos en el eje x e y son enteros

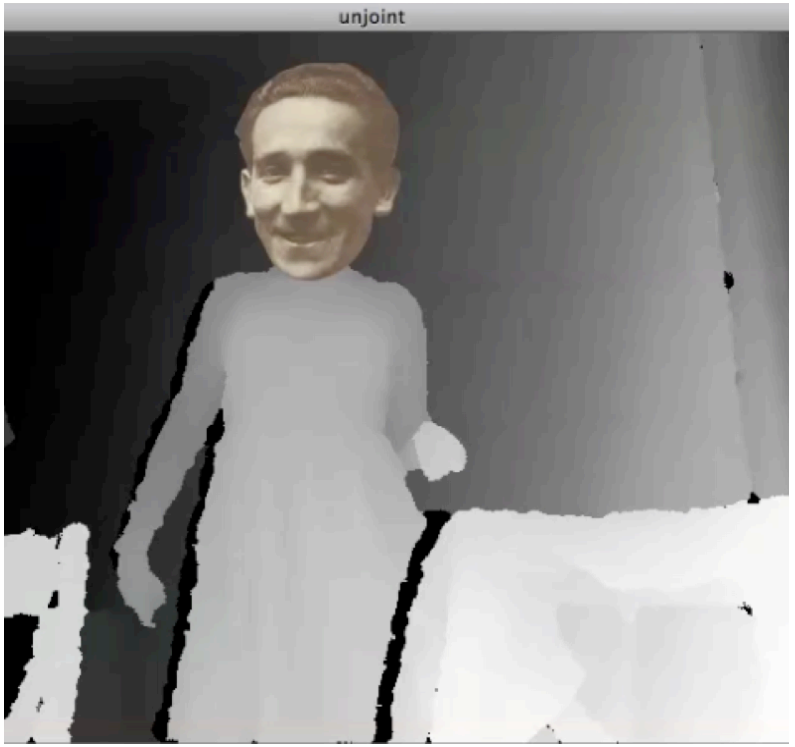
```
int closestValue;
Int closestX;
Int closestY;
```

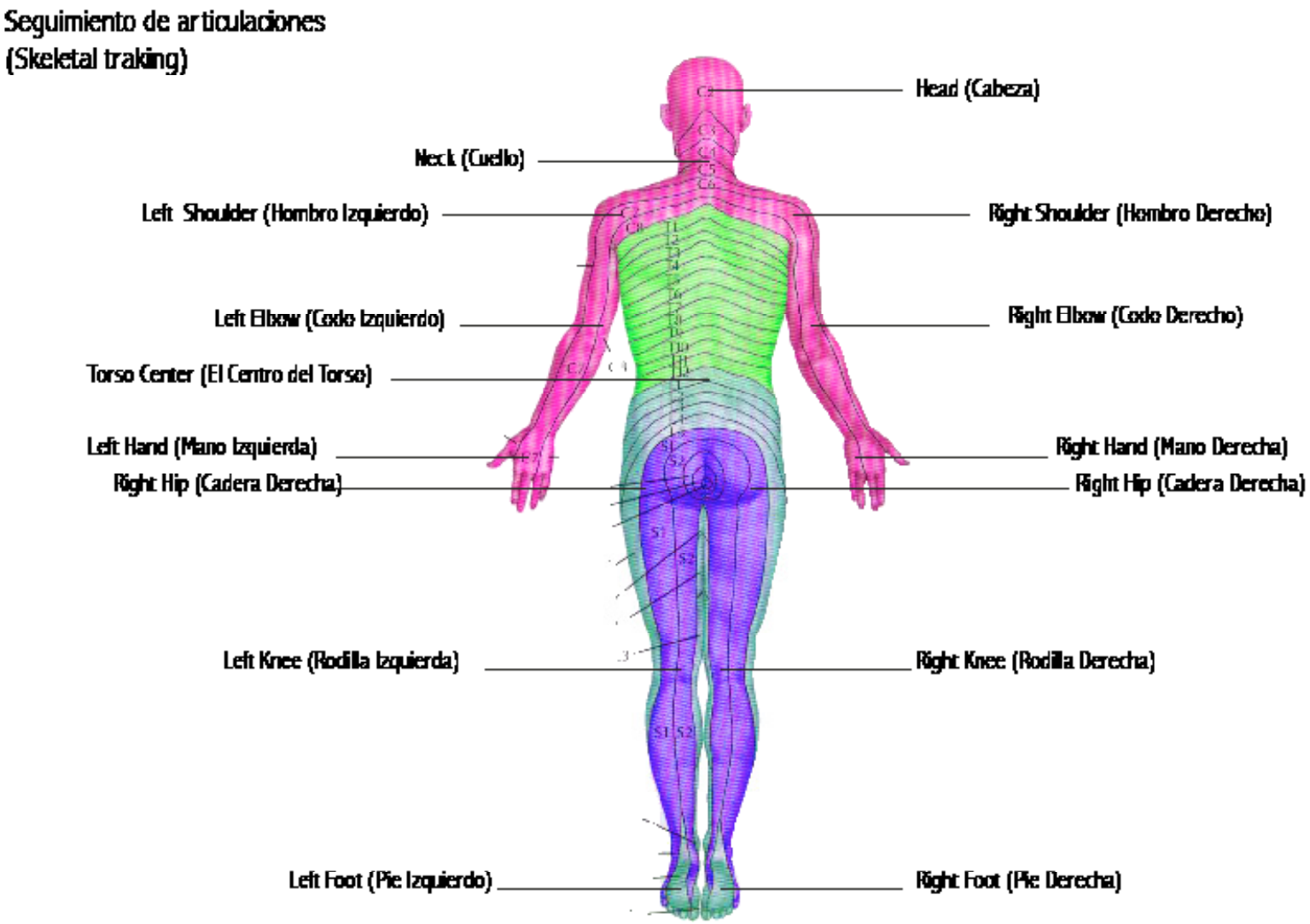
Este ejercicio se hizo en base a ejercicios realizados por Greg Borenstein (Making Things see, 2012)

Ejercicios 4	Variables o datos
Joints: Ensayo y error	1. Reconocimiento corporal dispositivo Kinect
	2. Se define lo que es un joint y como funciona.
	3. Octubre 2012
	4. Processing 1.5.1
	5. Simple OpenNi
	7.Código 4

Observación ejercicio 4

Este ejercicio es crucial para el desarrollo del ejercicio de la Performance. En los ejercicios anteriores se observó el acceso para el control de profundidad, pero sin poder reconocer el cuerpo y sus partes. En este ensayo se puede acceder de manera directa a cada parte del cuerpo que se reconoce. Una vez queOpen NI ha detectado al usuario, nos dice la posición de cada "Joint".
El joint es una articulación, y se denomina como: cabeza, cuello, hombros, codos, manos, torso, cadera, rodilla y piernas. No todo lo que se ve es literalmente una articulación (traducción en inglés del término joint) pero Open NI usa el término para referirse a todos los puntos que es capaz de trackear la librería. Este ejercicio cumple con poder reconocer un joint (cabeza), asignándole una imagen en formato jpg (en este caso le asignamos la imagen de David Arellano).
Reconocer un joint nos abre la posibilidad de asignarle elementos interactivos a cada parte del cuerpo. Es por eso que, para poder entender realmente como manejar los joints hay que tener una idea básica de lo que es un vector (en dos y tres dimensiones) y términos matemáticos relacionados a comportamientos físicos. Estos conceptos son también fundamentales para desarrollar ejercicios que involucren para el diseño de ejercicios parametrizables, es decir sirven para emular situaciones físicas que se ven en la naturaleza.





Seguimiento de articulaciones: Skeletal tracking Término asociado a la visualización de todos los puntos posibles que una API puede detectar.

En el código, se define que existe un vector para poder cargar el punto de la cabeza. De esta manera:

```
PVector head = new PVector();
```

Este guarda la magnitud y la dirección del joint. En la imagen de la derecha se puede observar como se puede trackear los joints.

Con la función `kinect.getJointPositionSkeleton(userId, SimpleOpenNI.SKEL_HEAD, head);` pones la posición de la cabeza en ese vector creado. Para llamar a las distintas partes del cuerpo, se identifican con la línea SKEL.
Por ejemplo, si quiero llamar al codo derecho, llamas de esta manera a los datos de la posición

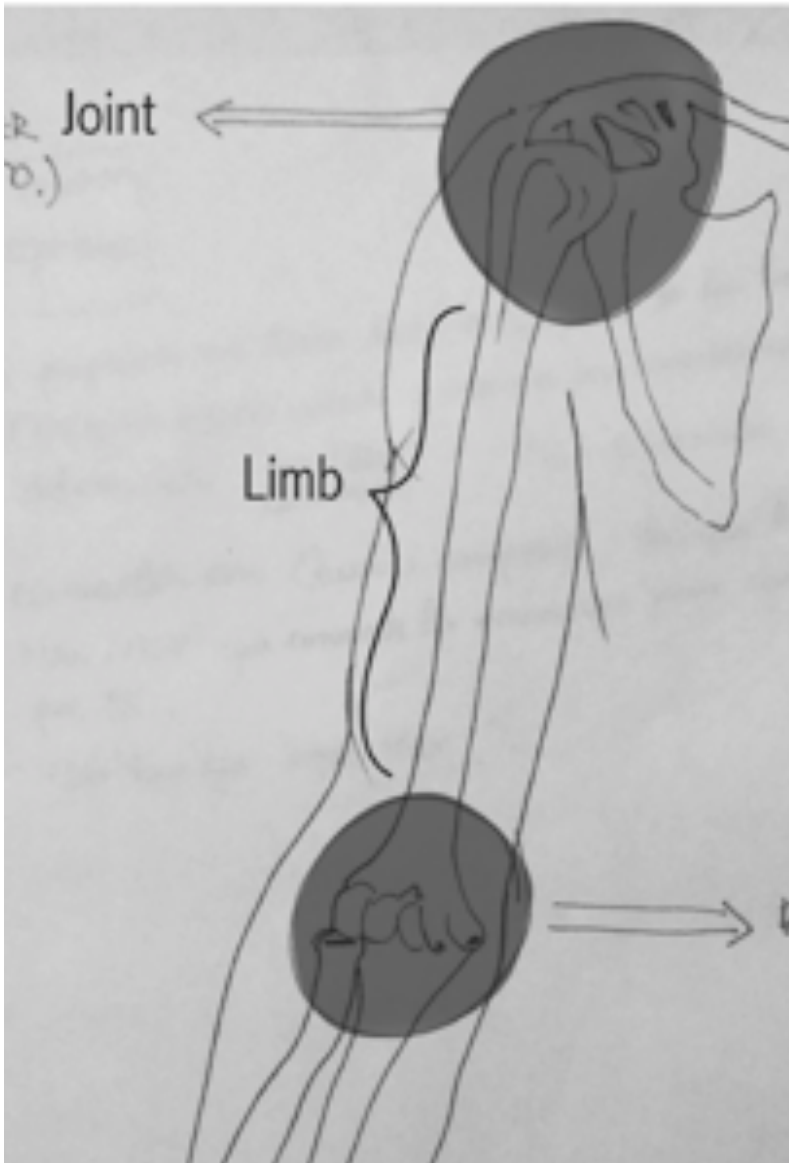
```
SimpleOpenNI.SKEL_RIGHT_ELBOW
```

Para hacer símil las coordenadas del espacio creado en processing con la profundidad de imagen real (mundo real) se utiliza la siguiente línea.

```
kinect.convertRealWorldToProjective(head, head);
    head.y = head.y ;
    head.x = head.x ;
```

En este caso, la cabeza de David Arellano no se veía 100% en la posición de mi cabeza, por lo cual le hice una pequeña modificación de coordenadas.

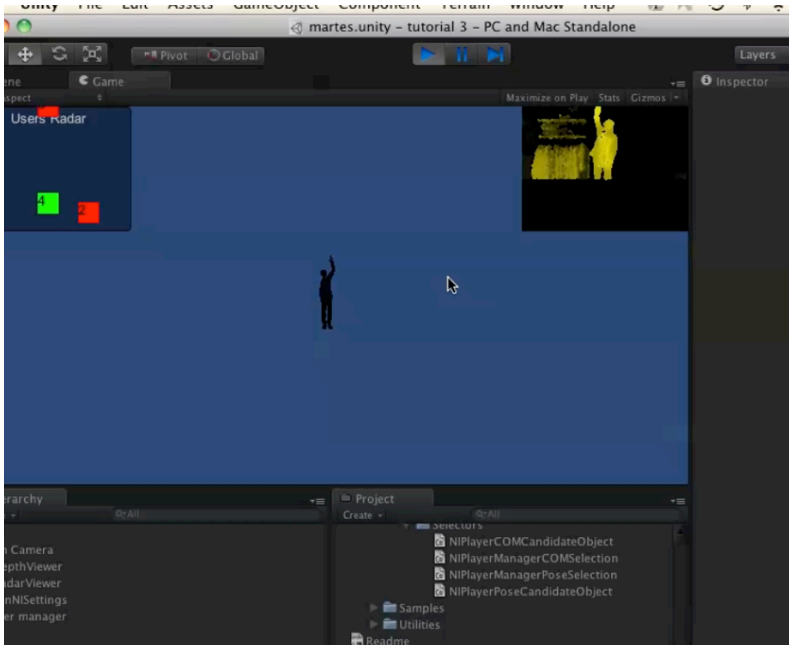
```
    head.y = head.y+30 ;
    head.x = head.x+30 ;
```



Ejercicio 5	Variables o datos
Reconocimiento de huesos y asignación de modelo	1. Reconocimiento corporal dispositivo Kinect
	2. Reconocimiento del cuerpo 3d
	3. Octubre 2012
	4. Unity, Unity Pro
	5. OpenNi, nite 5
	7.Nite 5, se modifica en mono develop (API)

Observación ejercicio 5

Estas instrucciones son las predeterminadas que se siguieron en un principio. Acá se importó un modelo riggeado desde Cinema 4d, el cual tiene compatibilidad con Unity.
Al riggear en Cinema es conveniente guardar cada parte de la articulación creada con los mismos nombres de como se nombran los joints en OpenNI. (ver imagen del seguimiento de articulaciones) y así hacerlas coincidentes con los datos que te pide en Unity.
Instrucciones básicas:
Primero se debe descargar Unity desde HTTP://UNITY3D.COM/ e instalar como cualquier otra aplicación. Luego instalar el plugin de Zigfu desde www.zigfu.com , el cual instala Open NI y NITE, lo cual hace posible el uso del dispositivo Kinect dentro de Unity. Si se instalaron estas librerías con anterioridad para Processing u otro Software, no es necesario volver a instalarlo. Una vez realizada la instalación, crearemos un proyecto siguiendo los pasos:
1. Selecciona File > New Project desde el menú.
2. En el popup que se abre seleccionar Standard Assets para importar.
3. Importar el Zigfu Unity Package. En el menu selecciona Assets > Import Package > Custom Package (el paquete se debe encontrar donde realizaste la descarga de Zigfu)
4. Navegar hacia la ruta de descarga del paguete Zigfu e importar el siguiente archivo -UnityOpenNIBindings-v.1.1.unitypackage
5. Unity importará todas las librerias necesarias para el proyecto.
6. Para probar un ejemplo dentro de Unity ir a la pestaña Project y abrir la carpeta llamada _Scenes donde encontraras ejemplos de escenas creadas, existen 2 categorías de ejemplos: interface y game.
7. Se debe hacer doble click en cualquiera de los ejemplos de la carpeta, luego presionar el botón “play” para ejecutarlos, no olvidar conectar antes el dispositivo Kinect



Ejercicios 6	Variables o datos
Reconocimiento de articulaciones	1. Reconocimiento corporal dispositivo Kinect, elementos gráficos parametrizables
	2. Reconocimiento de articulaciones y dibujo de una lÍnea de unión entre cada una de estas pareciendo extremidades del cuerpo.
	3. Octubre 2012
	4. Processing
	5. Simple OpenNi
	7.Código 6

Observación ejercicio 6

Como mencionamos anteriormente, el algoritmo rastrea a personas frente al dispositivo Kinect. Al momento de encontrar un cuerpo, realiza el seguimiento y reconoce las articulaciones del cuerpo (joints) dibujando un círculo rojo para cada una :

```
DRAWJOINT(USERID,SIMPLEOPENNI.SKEL_HEAD); .
```

Luego tomando 2 articulaciones dibuja una línea entre ellas (KINECT.DRAWLIMB(USERID, SIMPLEOPENNI.SKEL_HEAD, SIMPLEOPENNI.SKEL_NECK);)

Pertenece a los elementos gráficos parametrizables pues se utiliza una elipse para demarcar donde está el joint de cada parte del cuerpo, como veremos en la siguiente línea:

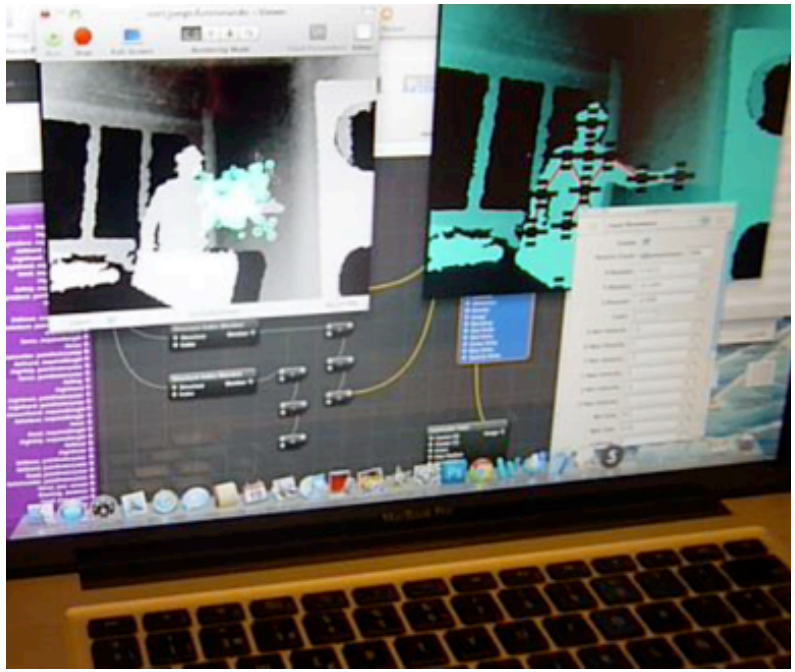
```
PVECTOR CONVERTEDJOINT = NEW PVECTOR();
  KINECT.CONVERTREALWORLDTOPROJECTIVE(JOINT, CONVERTEDJOINT);
  ELLIPSE(CONVERTEDJOINT.X, CONVERTEDJOINT.Y, 5, 5);
  TEXT("X:"+CONVERTEDJOINT.X);
```

Este ejercicio me puede permitir medir las distancias de las extremidades del cuerpo, y generar gráfica entre los joints.

Ejercicios 7	Variables o datos
Synapse	1. Reconocimiento corporal dispositivo Kinect y elementos gráficos parametrizables
	2. Ejercicio basado en API Synapse.
	3. Septiembre 2012
	4. Quartz Composer
	5. OpenNi
	7. Sin código, con nodos

Observación ejercicio 7

Synapse, esta API puede ser el que recibe la aplicación para poder conectarse con el dispositivo Kinect. Puede comunicarse con Ableton Live, Max /MSP /Jitter (o aplicaciones que pueden recibir mensajes OSC "Open Sound Control") . Su propiedad principal es que puede acceder a los joints, es decir trackear y calibrar el cuerpo. En este caso solamente se utilizó la IDE de Quartz Composer (QC). Este ensayo es típico en el desarrollo de la unión entre el dispositivo y QC.
Al realizarlo hubo que tener cuidado con el orden de pasos a seguir(todos los elementos descargables en http://synapsekinect.tumblr.com/).
Primero se recibe la señal de Synapse, luego se abre el "quart_passtrought" y después llamas a la señal OSC. Te irá dando los datos de salida de las partes del cuerpo. Esto se puede unir a las funciones matemáticas realizadas en el nodo Structure Index Member con cada uno de los ejes. El plug in "synapse kinect" unido por nodos a un sprite o billboard hace conectarnos con la profunidad de imagen que recibe desde la señal de synapse.

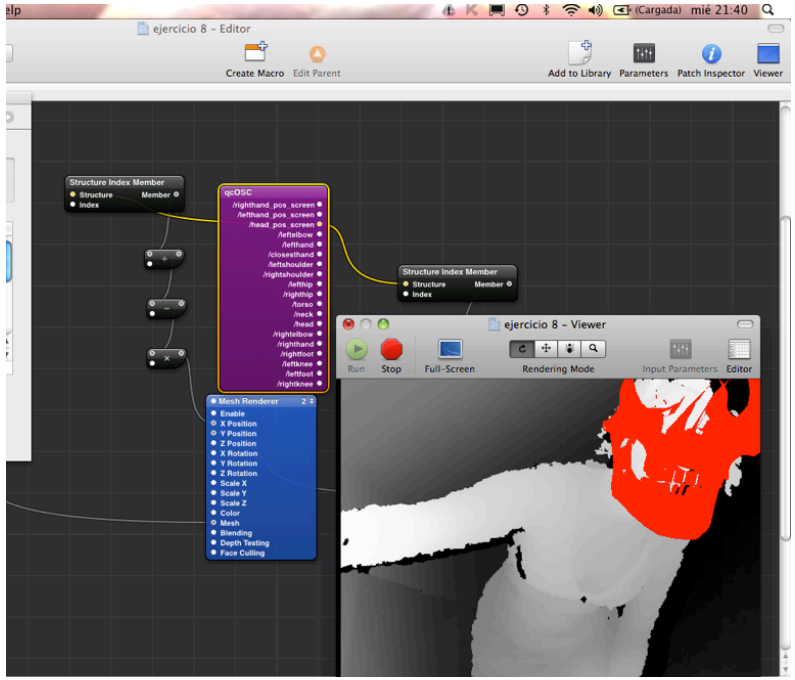


Ejercicios 8	Variables o datos
Reconocimiento de la cabeza (calavera)	1. Reconocimiento corporal dispositivo Kinect y elementos gráficos 3d
	2. Importar modelo 3d, y conectarlo a la Kinect.
	3. Octubre 2012
	4. Quartz Composer
	5. OpenNi
	7. Sin código, con nodos

Observación ejercicio 8

La importación de objetos 3d hacia Quartz Composer fue realizada desde Cinema 4d en formato .DAE. Este formato permite ser importado por el plug in MESH IMPORTER (es acá donde se pone en settings la ruta del archivo 3d)el cual se une con el plug in MESH RENDERER. Éste plug in se une con los STRUCTURE INDEX MEMBER el cual realiza funciones matemáticas hacia QCOSC (LOS JOINTS).

En este ejercicio se realizó un cambio de ejes, jugando con la relación invertida del movimiento del cuerpo con el del modelo 3d (calavera). Esto permite concluir que uno puede controlar el seguimiento del cuerpo, el cual no tiene que ser necesariamente sincrónico con la interfaz. Esto se puede controlar con las operaciones matemáticas del STRUCTURE INDEX MEMBER.
Este ejercicio no considera el plano Z, pero si se podría llegar a implementar para considerar el espacio, haciendo que el modelo se "aleje" o "acerque" de acuerdo al movimiento del cuerpo.



Ejercicios 9	Variables o datos
John Lennon Puppet	1. Reconocimiento corporal dispositivo Kinect y Elementos Gráficos 2D
	2. Ejercicios basados en el proyecto Processing llamado “MantisShootingExample”.
	3. Septiembre 2012
	4. Processing 1.5.1
	5. Processing Open GL, Simple Open NI, CodeAnticode .glgraphics, ddf.minim (de sonido), javax.media. opengl
	7.Código 9

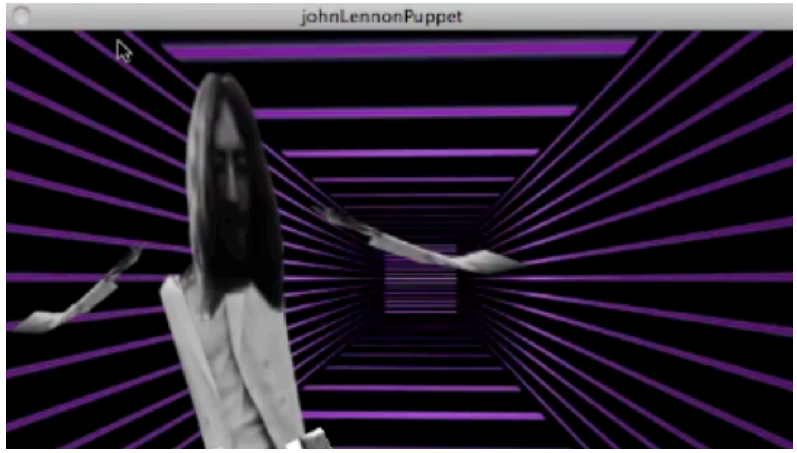
Observación ejercicio 9

Ejercicio en el cual se reunieron varias etiquetas en su funcionamiento. Incluye el reconocimiento de los joints, su calibración y el acceso a sus datos. Es el ejercicio más completo que encontré y de mayor análisis. Tiene variables de sonido que reacciona a movimiento, cambio de fondo con el movimiento de los brazos y permite cargar archivos livianos para poder visualizar el cuerpo. Esto hace que el movimiento sea más acorde con el movimiento del cuerpo, reaccionando de manera casi instantánea. Se consideró este ejercicio para el desarrollo de un proyecto. La falla de este ejercicio es que no tiene considerado el codo ni la rodilla, lo que hace menos fluido o creíble el movimiento del cuerpo. Este proyecto fue hecho en base al proyecto “Mantis Shooting Example”.
Está programado para que desde el joint “hand” o mano pueda liberar algún objeto tipo de partícula. En este caso libera signos de la paz, pero puede ser algo más elaborado.
El cuerpo del personaje se realizó mediante cinco composiciones en Photoshop. En este programa se crea la cabeza, los brazos (en realidad es un brazo que es puesto como espejo en el código), las piernas (lo mismo que en los brazos), y el cuerpo o body.
La proporción del cuerpo de John Lennon realizada en Photoshop no es concordante con lo que carga el Processing, esto es porque hay una proporción determinada dentro de los canales de la imagen PImage , el cual es definida por el padding que se encuentra en este tipo de líneas (en el main)

```
renderRectFromVectors(skel.leftShoulderPos, skel.rightShoulderPos, skel.rightHipPos, skel.leftHipPos, 5, 10, bodyTex);
```

Las variables numéricas 5,10 son las proporciones del padding en X e Y. El concepto de padding es como el espacio que tiene de borde para que la imagen se “acomode”, como en el concepto de las páginas web. En esta misma línea se puede observar que es la textura del cuerpo, al armar un “cuadrado” que es un espacio correspondiente a la imagen. El cuadrado se hace con la relación entre el codo izquierdo (leftShoulderPos), el derecho (rightShoulderPos), la cadera derecha y la cadera izquierda. La última variable corresponde al body text que permite llamar a la textura definida con ese nombre al principio, en la declaracion de variables generales.

```
PImage bodyTex, headTex, armTex, legTex, bg, firingBg;
String bodyTexFile = "torso.png";
```



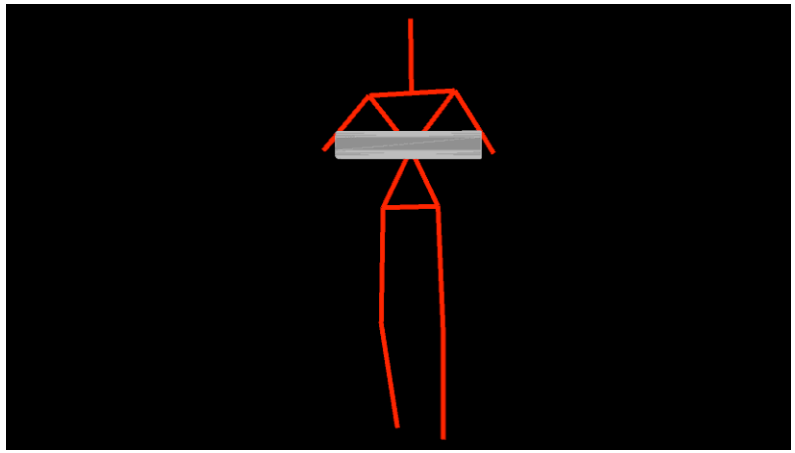
Ejercicios 10	Variables o datos
Carga de objeto 3D y posicionamiento	1. Reconocimiento corporal dispositivo Kinect
	2. Carga de modelo de animación 3d del dispositivo Kinect entre las manos de un esqueleto.
	3. Octubre 2012
	4. Processing 1.5.1
	5. Simple OpenNI
	7.Código 10

Observación ejercicio 10

El algoritmo rastrea a personas frente a al dispositivo, al momento de encontrar un cuerpo, realiza el seguimiento, reconoce las articulaciones del cuerpo (joints) DrawJoint(UserID,SimpleOpenNI.SKEL_HEAD);.
Luego, tomando 2 joints dibuja una línea entre ellas (Kinect.DrawLimb(UserID, SimpleOpenNI.SKEL_HEAD, SimpleOpenNI.SKEL_NECK);).
Además realiza la carga de un modelo 3D que se moverá y adaptará a las posiciones desde la mano izquierda a la mano derecha.
También es acá donde se comprendió el concepto de Push y Pop- Matrix, el cual guarda lo que hay hasta ese punto en las transformaciones, se hace la transformación y después el pop matrix vuelve a cargar lo que guardo anteriormente. El pop matrix va al final de lo que quieres que esté aislado . El orden en que uno hace las transformaciones y las rotaciones determina que es lo que pasa con el comportamiento del objeto. Por ejemplo, un objeto puede hacer una rotación de 30 grados , cambiando sus ejes, y después una traslación de x+2. Eso no es lo mismo que primero hacer la misma traslación y después la rotación. El orden en que uno hace las modificaciones altera el resultado.

```
pushMatrix();
    lights();
    stroke(175);
    strokeWeight(1);
    fill(250);
    translate(rightHand.x, rightHand.y, rightHand.z);
    rotate(angle, axis.x, axis.y, axis.z);
    model.draw();
    popMatrix();
```

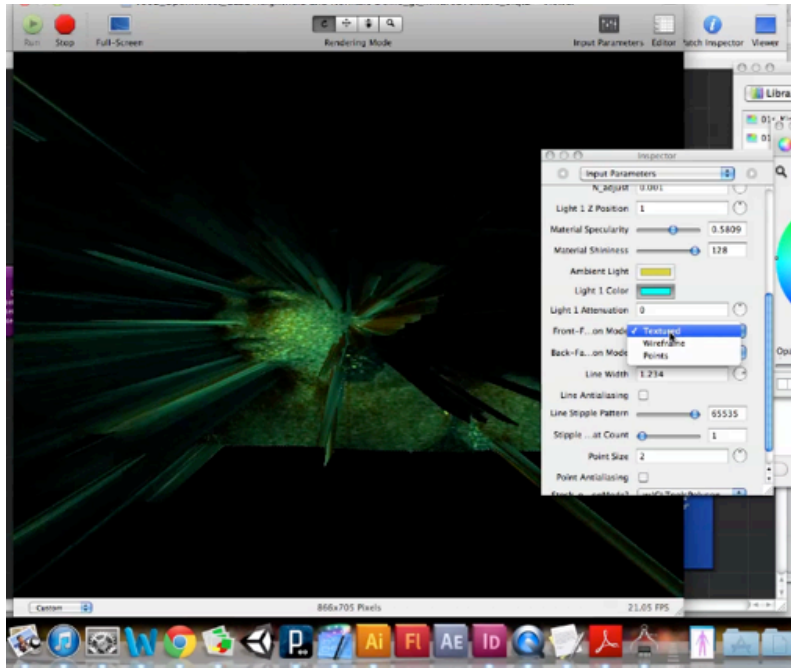
Este ejercicio realiza algo similar al ejercicio 6, con el agregado de importar un objeto 3d dentro de la escena. Con los recursos utilizados, la fluidez del movimiento se ve un poco alterada, comparado con el movimiento del JPG. La orientación del objeto es una variable que tiene potenciales para definir objetos que interactúen con el cuerpo y que emulan el espacio Z , haciendolos coincidir con el espacio real. Esto hace que no solo el cuerpo se pueda ver de distintos ángulos, si no que también objetos creados digitalmente pueden tener distintos puntos de vista.



Ejercicio 11	Variables o datos
Captura de nube de puntos (depth image)	1. Reconocimiento corporal dispositivo Kinect
	2. Permite captar la imagen rgb o depth desde la camara
	3. octubre 2012
	4.Quartz Composer
	5.Kineme Kinect Patch
	7. Este ejercicio esta limitado por el patch que se utiliza el cual no nos entrega información de reconocimiento de las partes del cuerpo, para integrar mayor interactividad.

Observación ejercicio 11

Este ejercicio es uno de los ejemplos demo del v002OpenKinect. El patch permite capturar la imagen RGB y la imagen de profundidad. En este ejemplo se manipula la imagen de profundidad para controlar el área de captura en el eje Z, el cual permite limitar lo que deseamos capturar. Además se agregan filtros para manipular los parámetros de estos en tiempo real e interacción de audio, obteniendo diferentes gráficas como nube de puntos, triángulos o cuadrados.
También se puede modificar el llamado TILT el cual permite mover el motor del eje del dispositivo Kinect para poder realizar el tracking adecuado del cuerpo. Esto fue de ayuda para el montaje en la siguiente etapa. El ángulo de visión de la Kinect depende de donde se encuentre con el cuerpo, el tamaño de esto, y la distancia entre ellos.
Con el movimiento del mouse se puede "navegar" dentro del espacio 3d simulado por la nube de puntos, lo que permite ver la imagen de distintos ángulos.



Ejercicio 12	Variables o datos
De Cinema 4d a Processing	1. Elementos en formato 3d
	2. Importar elementos 3d
	3.Diciembre 2012
	4. Cinema (plug in Riptide) y Processing
	5. saito.objloader
	7.Código 12

Observación ejercicio 12.1

Para importar los objetos en 3d, se utiliza la librería Saito Obj Loader, la cual importa objetos 3d en formato OBJ. En este ejercicio se realizaron dos pruebas: la importación de distintos objetos 3d y correcta su posición espacial, y los modos son las formas en que están constituidas las mesh de los objetos 3D.

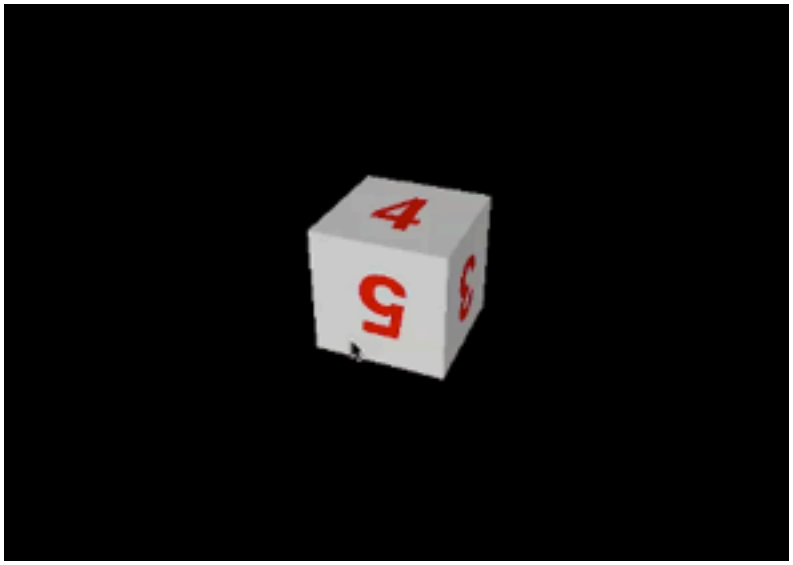
```
model = new OBJModel(this, "prototipoBoca.obj", "absolute", TRIANGLES);
model1 = new OBJModel(this, "ojoIzq.obj", "absolute", TRIANGLES);
model2 = new OBJModel(this, "prototipoPiel.obj", "absolute", TRIANGLES);
model3 = new OBJModel(this, "ojoDer.obj", "absolute", TRIANGLES);

model.shapeMode(POINTS);
model.shapeMode(LINES);
model.shapeMode(TRIANGLES);
model.shapeMode(QUADS);
```

En este ejercicio podemos ver al objeto 3D en cuatro estados , uno en POINTS, otro en LINES otro en TRIANGLES y por último QUADS, siendo este último el resultado gráfico esperado para la importación de un objeto 3D. (ver imágenes en la página siguiente)

Observación ejercicio 12.2 : Cargar el modelo 3d de manera correcta (de Cinema 4d a Processing)

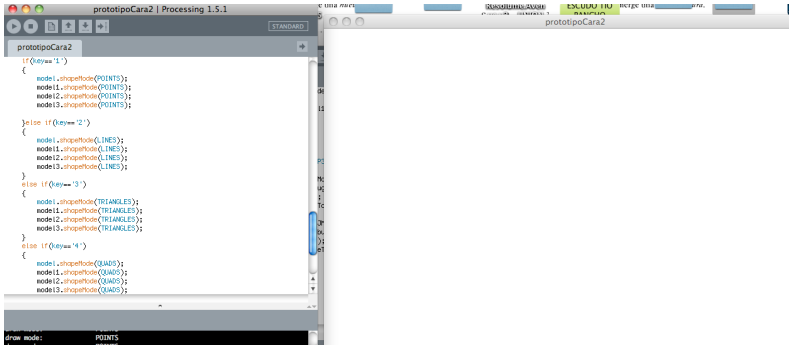
En el caso específico de esta experimentación, la mayoría de los objetos creados en 3D son en Cinema 4D , el cual no genera objetos en textura MTL. Para solucionar esto se debe descargar el Plug-in llamado RIPTIDE (riptide pro 2.2 SN=1630092515), ideal que se pueda descargar el RIPTIDE PRO. Riptide da una serie de alternativas de exportación del objeto en OBJ y su textura la exporta en MTL. Dentro de las opciones de exportación del archivo, el MESH orientation es el parámetro que afecta la textura. Se relizó una serie de pruebas con un cubo, el cual tenía en cada una de sus caras un número.
La orientación de este número y su posición determinan si la textura está bien cargada. El resultado de las pruebas fue que las opciones "voltear UV verticalmente", y con "Flip z axis" son las únicas opciones que hay que marcar en el editor de "mesh orientation" y de "texture orientation". Con esto probado, las cargas en Processing de la textura de los objetos3d creados en Cinema 4d ya no fueron ningún problema.
Al hacer las pruebas, se mantuvieron las siguientes propiedades: todos los archivos estaban siendo exportados en "simple file per scene", con ordenamiento c4d, sin animación, y sin "texture orientation".

Manera correcta de exportar la mesh

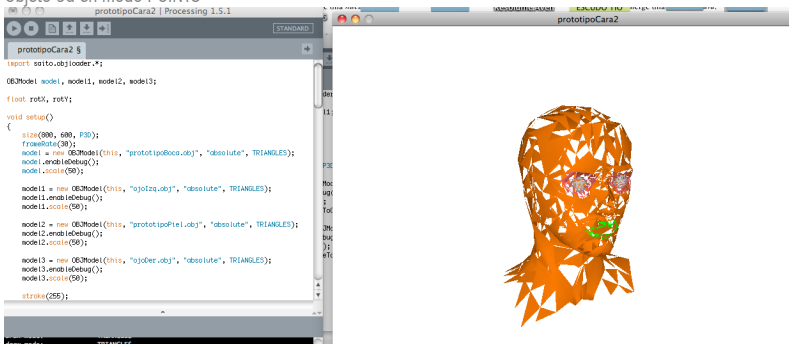


Construcción de cubo para probar textura. Arriba una prueba en processing,

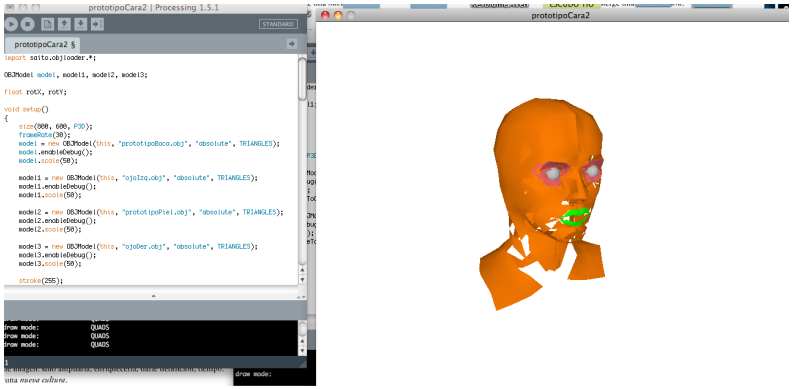
Como consejo, si se ocupa “Hyper Nurbs” hay que asegurarse de que el “subdivisión renderer” esté correcto. La mesh va a ser subdividida en el obj. file , ocupando el “renderer setting”, no el “editor setting”.
Es importante también, al momento de denominar los archivos exportados, no ocupar espacios, para que el processing no tenga problemas al importarlo.
Si bien puede considerarse un tema poco relevante, importación de la textura es más complejo de lo que puede parecer. La orientación, la posición, los colores, los relieves, el mapeo UV de la textura es un caso delicado si nos interesa mantener una buena calidad gráfica en el Processing. Más si es en algún momento controlado por algún joint.
Dentro de este proceso se abrió también la posibilidad de controlar los puntos de los mesh con el dispositivo Kinect. Se hicieron pruebas (fallidas) con el programa Rhino y el plug in Grasshopper, el cual permite exportar las coordenadas de mesh en un formato admisible para Processing. Esto fue un plan en algun minuto, llegar a poder controlar puntos de un mesh con el cuerpo. Este ejercicio ha sido realizado pero en OFX y aún no manejo nada de c++.



Objeto 3d en modo POINTS



Objeto 3d en modo TRIANGULES



Objeto 3d en modo QUADS

Ejercicio 13	Variables o datos
Mandala	1.Elementos gráficos parametrizables
	2. Creación de “mandala” en base a ejercicio encontrado en la web.
	3. Octubre 2012
	4. Processing
	5. Sin librería
	7. Código 13

Observación ejercicio 13

Ejercicio que en el cual las gráficas funcionan en base al comportamiento de algoritmos matemáticos. En este caso son valores enteros de X e Y los cuales definen posiciones las cuales actúan de manera aleatoria. Como por ejemplo en esta línea:

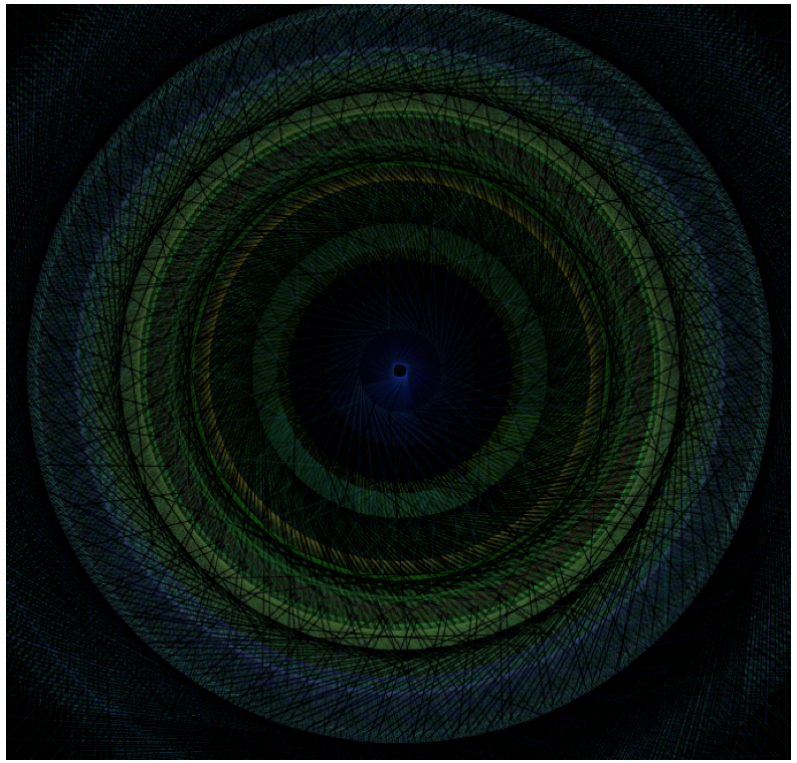
```
x1 = (int) random(width/2);
```

Los colores, que tiene valores de entero, también actúan de manera aleatoria (RGB Y ALPHA) , el cual tiene un valor de 0 a 1. La siguiente línea define las iteraciones que define el número máximo de veces que se va efectuar el loop.

```
int j = (int) random(10);
iterations = (int) pow(2, j);
```

Siendo el loop el siguiente código. Es el comportamiento de la línea formada por x e y y su rotación.

```
for( i=0 ; i < iterations; i++)
 {
  line(x1, y1, x2, y2);
  rotate(2.5f*PI/iterations);
 }
```



Ejercicio 14	Variables o datos
Pajaros	1.Elementos gráficos 2d y elementos interactivos
	2. Creación de array de movers que siguen el mouse
	3. Octubre 2012
	4. Processing
	5. Sin librería
	7. Código 14

Observación ejercicio 13

Ensayos de interactividad. Un sistema de partículas sigue al mouse. En este caso el sistema de partículas está basado en preceptos físicos tales como la denominación de un vector, la conceptualización de la fuerza, velocidad y aceleración . En este caso tenemos aplicada la velocidad a un conjunto de pájaros (sistema de partículas). Esta velocidad tiene un máximo del cual no puede traspasar (topspeed). Para adentrarse en el mundo del sistema de partículas, hay conceptos físicos que revisar, una buena bibliografía para este tema en específico se puede encontrar en "The Nature of Code" de Daniel Shiffman.

Mover un array el cual tiene definido 10 pájaros (movers)

```
Mover [] movers = new Mover [10] ;
```

La relación entre fuerzas es que cada una es dependiente de otra: la locación depende la velocidad y la velocidad depende de la locación.

```
velocity.add (acceleration);
velocity.limit (topspeed);
location.add (velocity);
```

Este es un ejercicio básico, pero es interesante incluir variables físicas para el desarrollo de gráfica parametrizable.

Ejercicio 15	Variables o datos
Batería virtual utilizando la nube de puntos	1.Reconocimiento corporal dispositivo Kinect y Elementos gráficos parametrizables
	2.Interacción con espacios virtuales (hot points) para generar sonido. Definición de áreas virtuales en nuestro espacio de desplazamiento real e interacción con éstas. Generando un sonido programado y "pintando" el área definido (un cubo) al momento de la interacción
	3. Noviembre 2012
	4.Quartz Composer
	5. Sin librería
	7. Sin código, con nodos

Observación ejercicio 15

Este ejemplo muestra como podemos definir espacios virtuales de interacción (hotpoint). La clase Hotpoint crea un espacio virtual de interacción en forma de cubo, la intensidad del color de relleno del cubo va a depender de grado de contacto de interacción con el área definida por el cubo. Los argumentos para la creación de los cubos son la posición de origen (x,y,z) y el tamaño del área. El algoritmo principal primero realiza la carga de archivos de formato wav que serán luego tocados según si se detecta que un cubo ha sido tocado por alguna parte de nuestro cuerpo.
Luego genera 2 cubos virtuales (hotpoints) en el espacio.

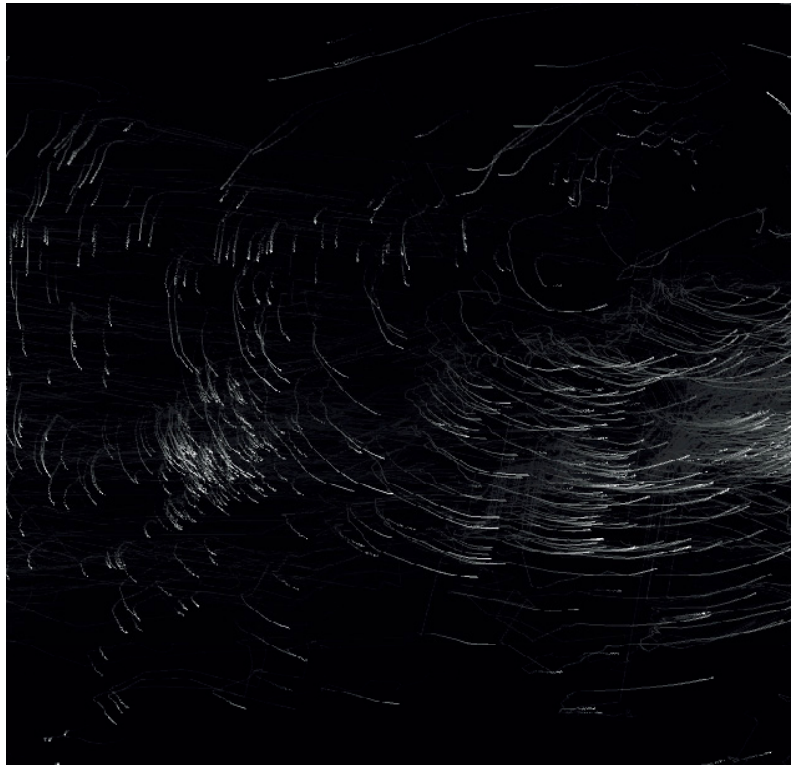
Para controlar la interacción se utiliza la nube de puntos de una persona

```
PVECTOR[] DEPTHPOINTS = KINECT.DEPTHMAPREALWORLD();
```

Al recorrer cada punto del arreglo que representa nuestro cuerpo, se evalúa si alguno de estos puntos coincide con cada una de los cubos definidos, comparando la posición del punto del arreglo con los puntos del área de cobertura del cubo en el espacio.
Si se genera alguna interacción, o sea una parte de nuestro cuerpo se encuentra en parte o en el total del área del cubo, se generará la reproducción de un archivo de audio que cargamos al inicio del algoritmo.

Ejercicio 16	Variables o datos
Sistema de Partículas	1.Elementos gráficos 2d y elementos interactivos
	2. Sistema de partículas complejo, MSA FLUID
	3. Diciembre 2012
	4. Processing
	5. msafluid.*; processing.opengl.*;javax.media.opengl.*; processing.opengl.*;
	7. Código 16

Observación ejercicio 16



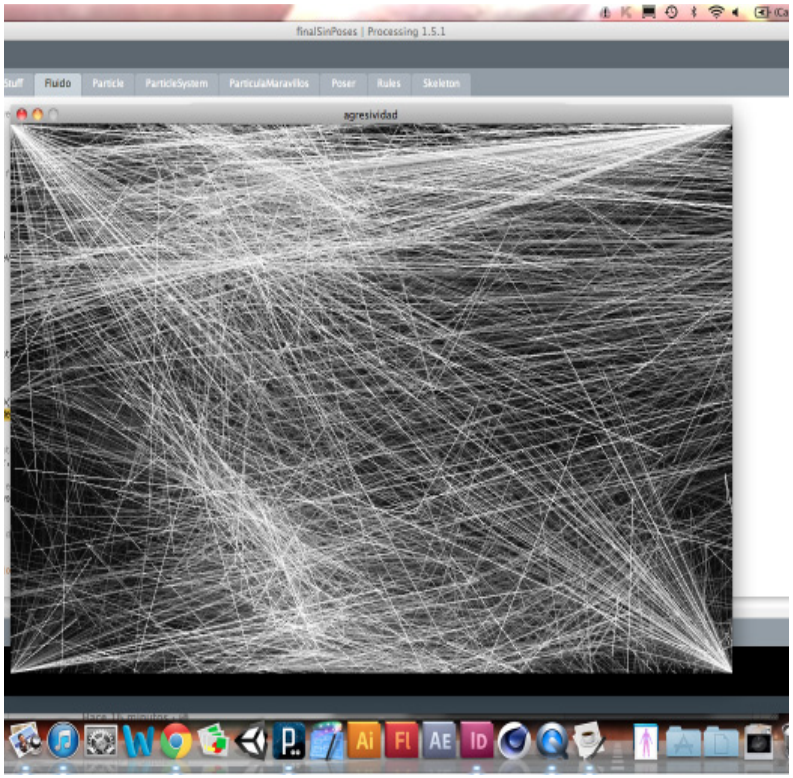
El sistema de partículas en este caso es complejo en el sentido de la cantidad de partículas y principalmente por el comportamiento. Gráficamente ofrece muchas alternativas de uso, emulando fluídos a un alto nivel de similiud con la realidad (en términos gráficos y de comportamiento). Si en este código (sketch) se desea realizar un sistema de partículas que emule un fluído lento y denso, en el código se deben configurar los parámetros que se refieran a estos conceptos. Este ejercicio fue realizado en base al ensayo llamado MSA FLUID. El ensayo fue hecho en base a una solución codificada de Jos Stam en su paper "Real-Time Fluid Dymanics for Games". Jos Stam toma variables estrictamente matemáticas, llamadas equaciones Navier-Stokes. Los parámetros de este código son variados, por lo cual permite tener distintos comportamientos de fluído. Se logró controlar la vida de las partículas y su movimiento. Existen factores físicos definidos como variables propias de un sistema de partículas. Por ejemplo, el concepto MOMENTUM es utilizado, lo que significa que es la cantidad de movimiento del cuerpo, que va acumulando fuerza y por eso se mantiene. El comportamiento es que el momentum es igual a la masa por velocidad, en este caso es 0.5.
Dentro de este código, las funciones que hacen que su comportamiento sea radicalmente distinto son por ejemplo las propias de este sistema de fluido:

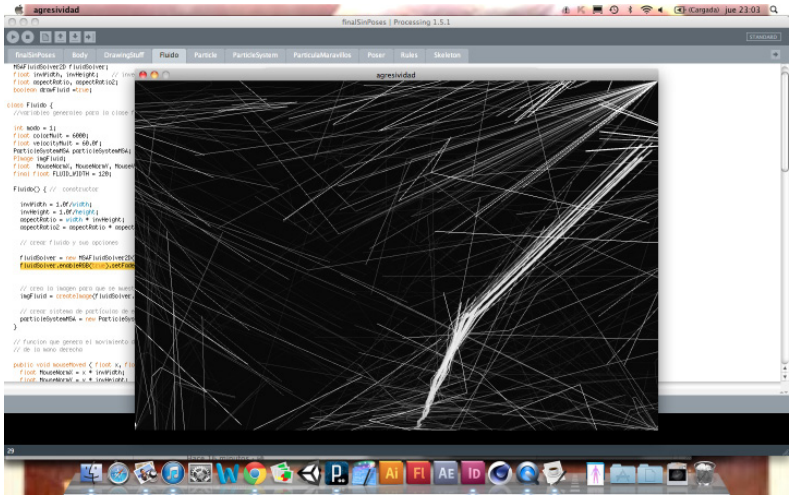
```
fluidSolver.enableRGB(true).setFadeSpeed(0.0001).setDeltaT(0.5).setVisc(0.0001);
```

Y también las variables de Open gl. Estas son al principio confusas y llenas de alternativas, pero mientras uno va analizndo el código, puede ir ententiendo ciertos

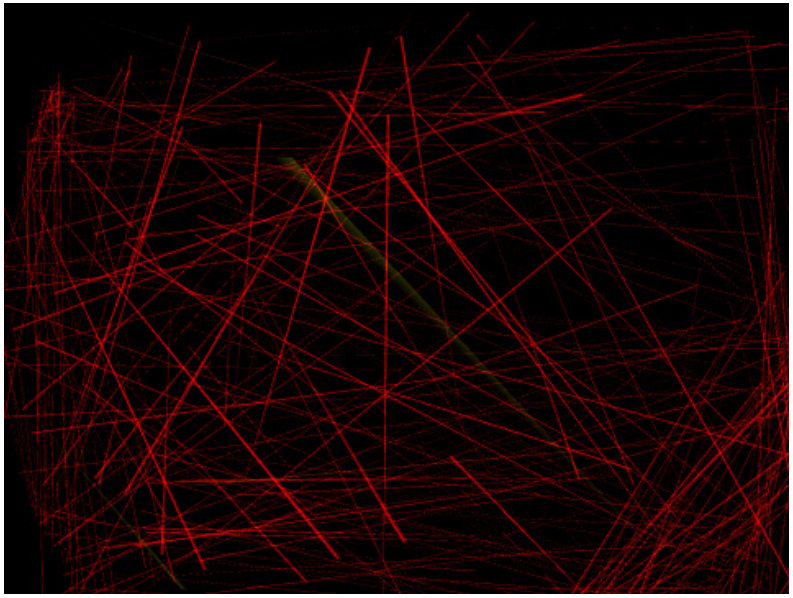
comportamientos de la librería. Por ejemplo, se recomida para la comprensión de la función blend, este sitio: http://www.andersriggelsen.dk/glblendfunc.php . Permite visualizar las funciones de blend y sus combinaciones.

```
GL.GLBLENDFUNC(GL.GL_SRC_ALPHA, GL.GL_ONE_MINUS_SRC_ALPHA);
```





Experimentos con el sistema de partículas Open GL de MSA FLUID



Conclusiones Experimento parte 1

La manera a llevar a cabo este proyecto de perfermance visual y danza fue en base a dos criterios:

1) Criterio técnico: Las posibilidades técnicas para llevar a cabo el proyecto fueron definidas en la primera etapa del proyecto. ¿Qué software utilizar para llevar a cabo un proyecto?, ¿ Qué hace optar por una plataforma en vez de otra? En este caso fueron problemas de tipo circunstancial del trabajo y de tiempo lo cual potenció el uso de una herramienta y el descarte de otra.

2) Conceptualización gráfica: Desarrollado en el Experimento parte 2
Las exigencias que tiene la conceptualización de la performance, y su desarrollo gráfico son variables que definen la manera de llevar a cabo el proyecto. Es decir, la intencionalidad del desarrollo gráfico mas las limitantes técnicas hicieron tomar decisiones para la etapa final .

1) Criterio Técnico

El IDE (entorno de desarrollo integrado) escogido para realizar la performance será **Processing (versión 1.5.1).** Se utiliza el Wrapper Simple OpenNI.

Una razón de esta elección es que Processing no tiene problemas de compatibilidad con las versiones de Mac OSX, al ser un entorno de programación que es empaquetado como un programa de aplicación.

Caso distinto era **Quartz Composer,** al momento de desarrollar ejercicios con distintos patchs y plug ins, se encontraron limitantes técnicas y de compatibilidad con la versión Mac OSX 10.6 y 10.7, el último no era capaz de ejecutar la mayoría de los patches y plug ins., por lo cual no cumplía con la totalidad de requerimientos.

Además de eso, dentro de las conexiones que tiene **Quartz Composer** con el dispositivo Kinect, encontramos limitantes específicas que no cumplían con expectativas que se tenía funcionamiento.El primer caso es Synapse, esta API puede ser el que recibe la aplicación para poder conectarse con el dispositivo Kinect.

Se encontraron las siguientes limitantes:

-No poder modificar la cantidad de usuarios que puede ser trackeado.
-No ser compatible con la nueva versión del dispositivo Kinect “Kinect for Windows”. En este proyecto no se utiliza esa versión, si no la versión para la XBox, pero limita potenciales proyectos que se podrían hacer con este nuevo dispositivo.
- No fue compatible con la versión de MacOSX 10.7 , por lo cual no podía desarrollarse.

El segundo caso que se analizó con **Quartz Composer**, fue el plug in Kineme Kinect Tools. Este era fácil de instalar y además fue compatible en versiones, sin embargo, este plug in solamente puede acceder a la profundidad de datos (depth image) que da la Xbox Kinect, pero no permite generar joints (articulaciones enviadas desde el dispositivo), por lo cual no cumplía con los requerimientos del proyecto.

El tercer caso de **Quartz Composer** corresponde al Plug in v002 Open Kinect Beta. Este tiene una excelente calidad para acceder a la profundidad de datos. Funciona para 32 y 64 bit. También te da el control de la variable Tilt, la cual es el motor de movimiento

del dispositivo. Accede a los datos de la modalidad de infrarrojo y RGB. Hasta este momento está desarrollado solamente para un dispositivo Kinect. Está optimizado para ir directamente a la librería Open GL. Sin embargo, tampoco cumple con los requerimientos al no poder acceder a los joints de cada parte del cuerpo.

¿Porqué **3d Unity** fue descalificado?

Al momento de realizar pruebas con 3d Unity, la situación se "alentó" en su ejecución La propia interfaz de desarrolladora de juegos, te obliga a hacer el seguimiento de los movimientos en base a un personaje en 3d que esté riggeado. Si bien seguía perfectamente los joints y no tenía problemas de seguimiento, y tiene buena fluidez de movimiento, el hacer pruebas involucraba una gran cantidad de tiempo de trabajo. Es interesante lo que se logra hacer a un nivel técnico. Pero la experimentación y la familiarización con estos software estaba sujeto a ser exhibido con la característica de ser mas un acto performativo de danza, en donde las variables gráficas responden a los movimientos propios de la bailarina, el cual buscaba una mayor amplitud de desarrollo gráfico, que interactividad con 3d. Este software, como está definido, está hecho mas para ser un realizador de juegos, donde los criterios como la acción del jugador, la interactividad con otros modelos 3d priman como variables.

Experimento Parte 2

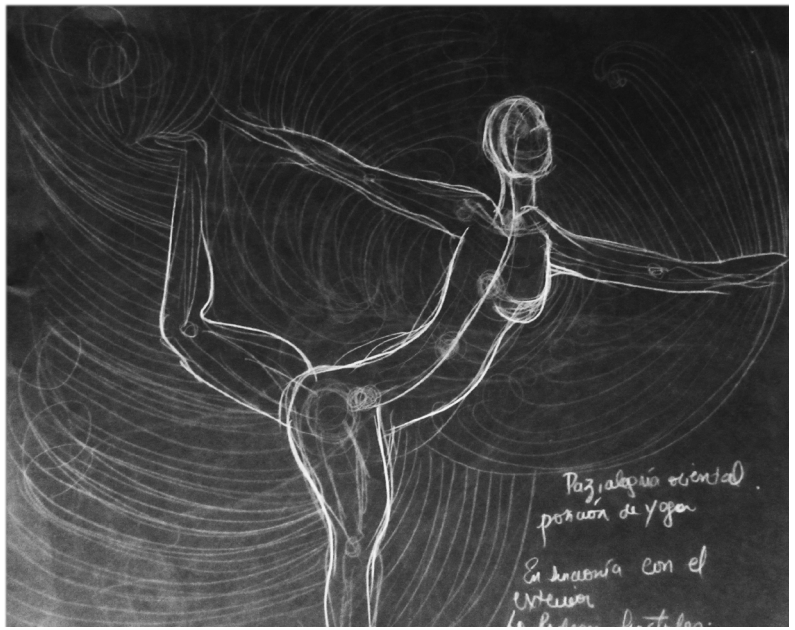
Conceptualización gráfica : elementos determinantes

Los conceptos propuestos para la performance incluye el desarrollo de 5 actos en los cuales se manifiestan dos emociones de un personaje.

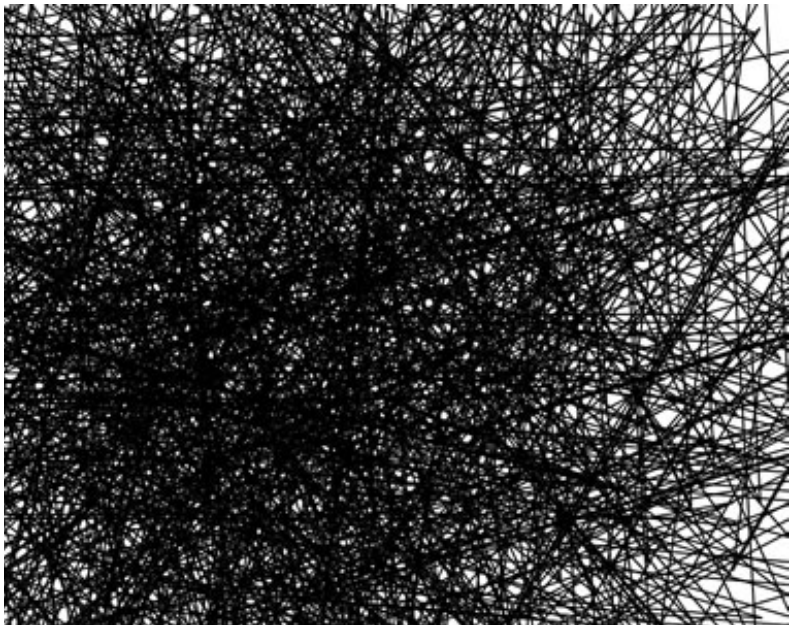
Las representación visual de las emociones no sólo responde a un código cromático, u espacial, si no también a un código que tiene que ver con el movimento y velocidad de la animación. La fluidez , el comportamiento que tiene la gráfica debe poder ser susceptible a cambios en base al movimiento de una manera parametrizable.

Por lo cual se toma la decisión de que utiliza el recurso de sistema de partículas experimentado en el **experimento parte 1.**

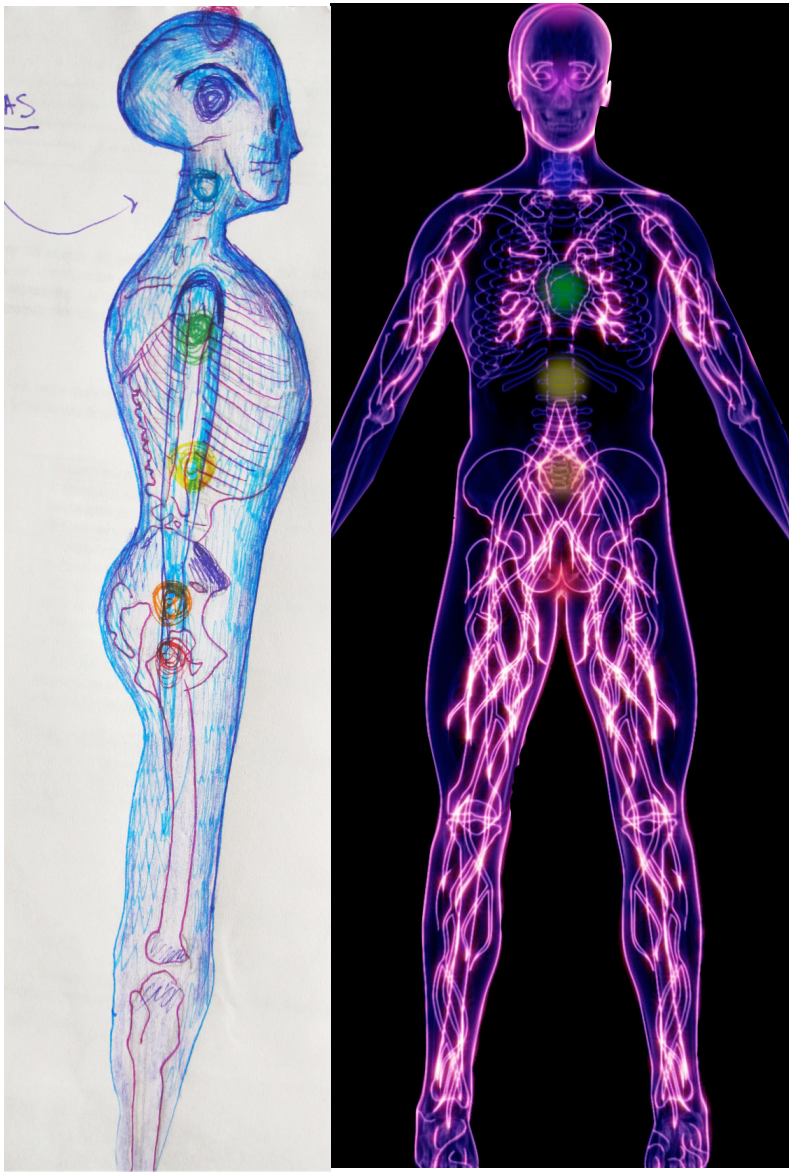
En el caso de la alegría, se juegan con ritmos armónicos, fluidos y lentos.



En el caso de la ira, se juegan con contrastes de ritmos, casi careciendo de ritmo. Se desarrolla el concepto caótico



El personaje, conceptualmente, es el "puente" entre el personaje que baila y la emocionalidad representada en el sistema de partículas. Por lo cual se toma como decisión que su nivel de abstracción es menor al resto de las escenas. La construcción original de este personaje es una animación realizada en after effects, donde se hicieron pruebas con el formato mov, gif y secuencia png. Todos para poder ser importados en Processing, ninguno con buen final: el mov se veía muy lento, era muy pesado el momento de reproducirse. Las velocidades de la composición eran más rapido de lo que por lo menos se podía reproducir. El formato gif no mantenía la calidad gráfica: se perdían los brillos, las opacidades, las saturaciones. La secuencia PNG no fue un formato aceptado para el código creado, por lo cual se llegó a la decisión de utilizar el formato JPG.



Desarrollo del código final : Observaciones

Las librerías utilizadas son una combinación de ensayos realizados en la primera parte. Simple OpenNI es la que utilizamos para poder generar datos del dispositivo Kinect.
El trabajo del desarrollo de este codigo estuvo en tratar de modularizar de mejor manera las funciones del código, con tal de que recibiera una optimización

Al momento de desarrollar el código, se encontraron dos sistemas de partículas que correspondían a distintos ensayos. Estos sistemas son motores distintos:

```
import traer.physics.*;
import msafluid.*;
```

Este encuentro de motores de partículas en un principio generó complicaciones por el encuentro de términos propios de los sistemas como, en la escena de la ira:

```
physics = new ParticleSystem();
```

con la escena de la alegría

```
particleSystemMSA = new ParticleSystem();
```

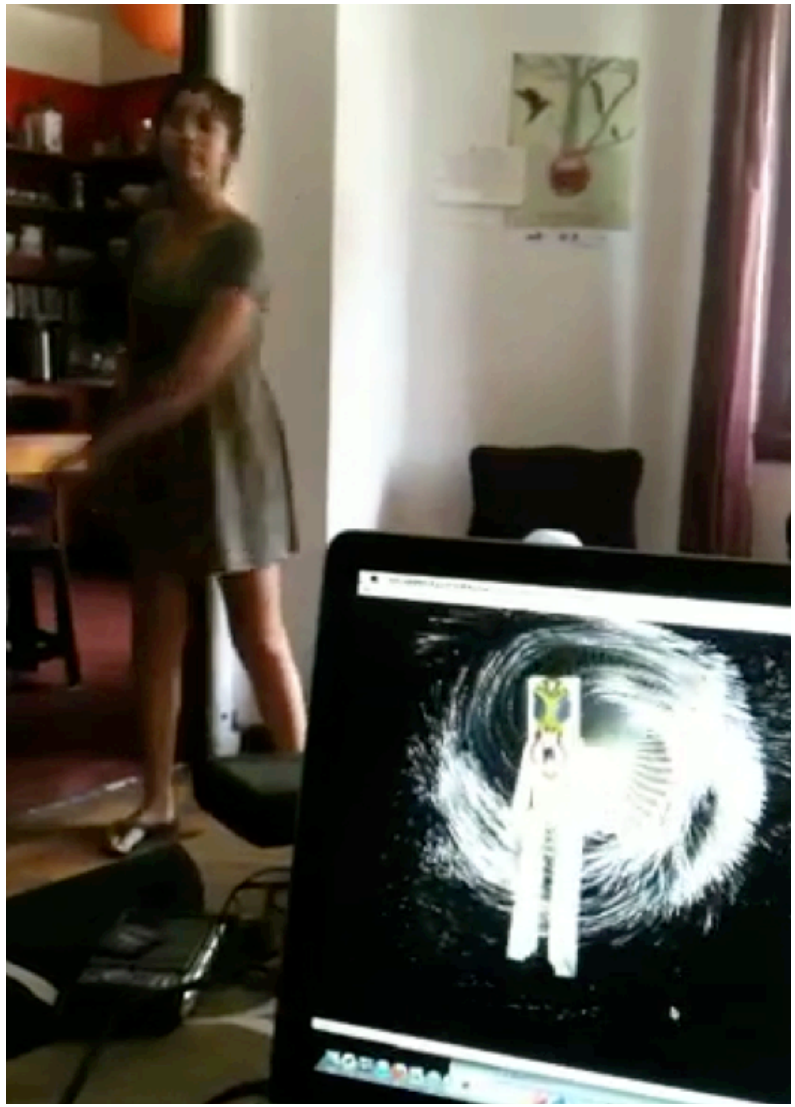
Esto se pudo solucionar lógicamente, cambiando el nombre de una de las variables del sistema de partículas y encapsulando cada escena en clases.

En este caso, el código cambiado fue el de la alegria

```
particleSystemMSA = new ParticleSystemMSA();
```

Lo mismo pasó con la variable "Partícula", la cual tuvo que ser mofiicada en el motor de la ira (physics) para ser llamada

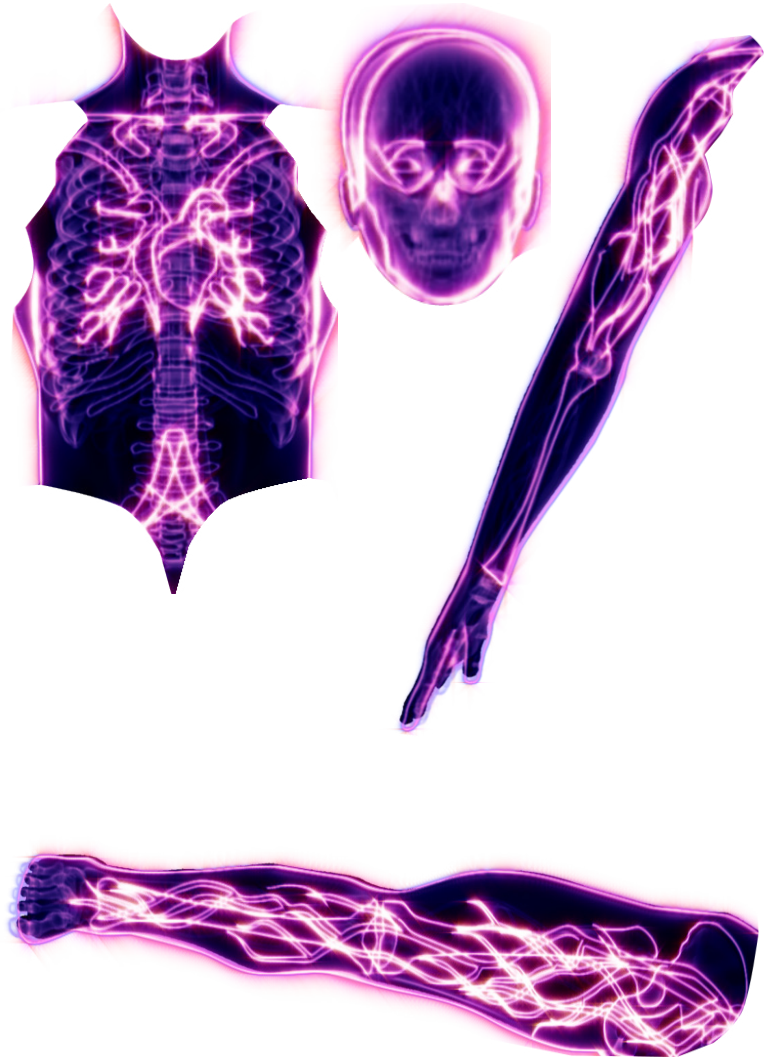
```
class ParticulaMaravillosa
```



Pruebas de unión de sistema de partículas MSA FLUID controlada por el joint de la mano izquierda

Este código debía ser capaz de poder contener las cinco escenas y poder ser cambiadas cuando el usuario quisiera. Por lo cual se realizó el código de cambio de modos el cual finalizó de la siguiente manera:

```
switch(mode) { //cambia entre los modos

  case 1:
    dibujaPuppet();
  break;

  case 2:
    dibujarFluido();
    break;

  case 3:
    dibujaPelota();
    break;

  case 4:
    kinectCirculo();
    break;

  case 5:
    //aprovecho de dibujar el sistema de partículas de la Escena 5
    maravillosa.draw(body.head.x, body.head.y);
    break;
  } // cierra cambio de modo
```



Persona del caso dibujaPupper () : Tratamiento gráfico para la importación a Processing en código final

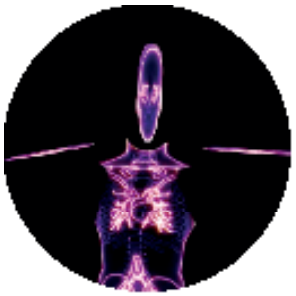
Con este codigo se puede cambiar de modos, pero aún no se especifica cómo el cuerpo puede controlar la aparición de las escenas. Esto se hace con el concepto de “reglas” las cuales generan poses que activan o desactivan alguna funciión, por ejemplo:

```
////Definir Poses////
SkeletonPoser pose1, pose2, pose3, pose4, pose5;

pose1 = new SkeletonPoser(context);
  // agrega regla para pose: las manos arriba
    pose1.addRule(SimpleOpenNI.SKEL_LEFT_HAND, PoseRule.ABOVE, SimpleOpenNI.
SKEL_LEFT_ELBOW);
   pose1.addRule(SimpleOpenNI.SKEL_RIGHT_HAND, PoseRule.ABOVE, SimpleOpenNI.
SKEL_RIGHT_ELBOW);
   pose1.addRule(SimpleOpenNI.SKEL_LEFT_ELBOW, PoseRule.ABOVE, SimpleOpenNI.
SKEL_LEFT_SHOULDER);
  pose1.addRule(SimpleOpenNI.SKEL_RIGHT_ELBOW, PoseRule.ABOVE, SimpleOpenNI.
SKEL_RIGHT_SHOULDER);


  // initialize the pose object 2
  pose2 = new SkeletonPoser(context);
  // agrega regla para pose manos en las rodillas
   pose2.addRule(SimpleOpenNI.SKEL_RIGHT_HAND, PoseRule.BELOW, SimpleOpenNI.
SKEL_RIGHT_KNEE);
    pose2.addRule(SimpleOpenNI.SKEL_LEFT_HAND, PoseRule.BELOW, SimpleOpenNI.
SKEL_LEFT_KNEE);


  pose3 = new SkeletonPoser(context);
  // agrega regla para pose mano derecha bajo rodilla derecha y mano izquierda sobre
hombro izquierdo
   pose3.addRule(SimpleOpenNI.SKEL_RIGHT_HAND, PoseRule.BELOW, SimpleOpenNI.
SKEL_RIGHT_KNEE);
    pose3.addRule(SimpleOpenNI.SKEL_LEFT_HAND, PoseRule.ABOVE, SimpleOpenNI.
SKEL_LEFT_SHOULDER);

pose4 = new SkeletonPoser(context);
  // agrega regla para que el cuerpo esté “acostado” y la pierna derecha esté a la misma
altura de la cabeza
   pose4.addRule(SimpleOpenNI.SKEL_RIGHT_FOOT, PoseRule.EQUAL, SimpleOpenNI.
SKEL_HEAD);

  pose5 = new SkeletonPoser(context);
  // agrega regla para manos cruzadas
   pose5.addRule(SimpleOpenNI.SKEL_RIGHT_HAND, PoseRule.EQUAL, SimpleOpenNI.
SKEL_LEFT_SHOULDER);
    pose5.addRule(SimpleOpenNI.SKEL_LEFT_HAND, PoseRule.EQUAL, SimpleOpenNI.
SKEL_RIGHT_SHOULDER);
  pose5.addRule(SimpleOpenNI.SKEL_RIGHT_ELBOW, PoseRule.BELOW, SimpleOpenNI.
SKEL_RIGHT_SHOULDER);
   pose5.addRule(SimpleOpenNI.SKEL_LEFT_ELBOW, PoseRule.BELOW, SimpleOpenNI.
SKEL_LEFT_SHOULDER);
```

Lo que gatilla el cambio de un estado a otro, de una escena a otra es el gesto corporal. El gesto corporal se define como pose. Las poses que tendrá el usuario hacen que se activen o desactiven ciertas gráficas. Para esto se usa el concepto de regla. Como se señala anteriormente, el cuerpo es trackeado gracias a sus articulaciones (joints) , y a los vectores que se forman de cada extremidad del cuerpo.

Cada pose se relaciona con un elmento gráfico, como veremos en las tablas relacionadas a continuación:

Acto 1



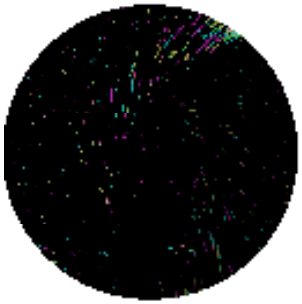
Desarrollo Gráfico

Posición cuerpo: poses

```
pose1 = new SkeletonPoser(context);
 // agrega regla para pose: las manos arriba
 pose1.addRule(SimpleOpenNI.SKEL_LEFT_HAND, PoseRule.ABOVE, SimpleOpenNI.SKEL_LEFT_ELBOW);
 pose1.addRule(SimpleOpenNI.SKEL_RIGHT_HAND, PoseRule.ABOVE, SimpleOpenNI.SKEL_RIGHT_ELBOW);
 pose1.addRule(SimpleOpenNI.SKEL_LEFT_ELBOW, PoseRule.ABOVE, SimpleOpenNI.SKEL_LEFT_SHOULDER);
 pose1.addRule(SimpleOpenNI.SKEL_RIGHT_ELBOW, PoseRule.ABOVE, SimpleOpenNI.SKEL_RIGHT_SHOULDER);
```

Código que define las reglas de cada pose.

Acto 2



Desarrollo Gráfico

Posición cuerpo: poses

```
// initialize the pose object 2
  pose2 = new SkeletonPoser(context);
  // agrega regla para pose manos en las rodillas
  pose2.addRule(SimpleOpenNI.SKEL_RIGHT_HAND, PoseRule.BELOW, SimpleOpenNI.SKEL_RIGHT_KNEE);
  pose2.addRule(SimpleOpenNI.SKEL_LEFT_HAND, PoseRule.BELOW, SimpleOpenNI.SKEL_LEFT_KNEE);
```

Código que define las reglas de cada pose.

Acto 3

Desarrollo Gráfico

Posición cuerpo: poses

```
pose3 = new SkeletonPoser(context);
 // agrega regla para pose mano derecha bajo rodilla derecha y mano izquierda sobre hombro izquierdo
 pose3.addRule(SimpleOpenNI.SKEL_RIGHT_HAND, PoseRule.BELOW, SimpleOpenNI.SKEL_RIGHT_KNEE);
 pose3.addRule(SimpleOpenNI.SKEL_LEFT_HAND, PoseRule.ABOVE, SimpleOpenNI.SKEL_LEFT_SHOULDER);
```

Código que define las reglas de cada pose.

Acto 4

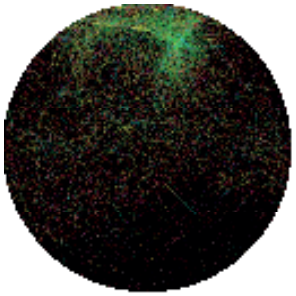
Desarrollo Gráfico

Posición cuerpo: poses

```
pose4 = new SkeletonPoser(context);
 // agrega regla para que el cuerpo esté "acostado" y la pierna derecha está a la misma altura de la cabeza
 pose4.addRule(SimpleOpenNI.SKEL_RIGHT_FOOT, PoseRule.EQUAL, SimpleOpenNI.SKEL_HEAD);
```

Código que define las reglas de cada pose.

Acto 5



Desarrollo Gráfico

Posición cuerpo: poses

```
pose5 = new SkeletonPoser(context);
// agrega regla para manos cruzadas
pose5.addRule(SimpleOpenNI.SKEL_RIGHT_HAND, PoseRule.EQUAL, SimpleOpenNI.SKEL_LEFT_SHOULDER);
pose5.addRule(SimpleOpenNI.SKEL_LEFT_HAND, PoseRule.EQUAL, SimpleOpenNI.SKEL_RIGHT_SHOULDER);
pose5.addRule(SimpleOpenNI.SKEL_RIGHT_ELBOW, PoseRule.BELOW, SimpleOpenNI.SKEL_RIGHT_SHOULDER);
pose5.addRule(SimpleOpenNI.SKEL_LEFT_ELBOW, PoseRule.BELOW, SimpleOpenNI.SKEL_LEFT_SHOULDER);
```

Código que define las reglas de cada pose.

Al momento de implementar estas poses, se puedo ver en la prueba de usuario, que las poses son muy dificiles de diferenciar una u otra, pues las reglas pueden estar cumpliendose en un gesto, sin tener la intención. Por ejemplo cuando uno declara que la mano izquierda esté igual a la mano derecha

```
pose5.addRule(SimpleOpenNI.SKEL_RIGHT_HAND, PoseRule.EQUAL, SimpleOpenNI.SKEL_LEFT_SHOULDER);
```

Puede darse el caso de que las manos estén "iguales" pero sin poder controlarlo, activando alguna escena que no debería., lo que limita muhco los movimientos de la bailarina .
Se intentó adicionar reglas, pero la pose que quedaba de resultado era muy difícil de reproducir y extraño para el movimiento fluido que se necesitaba en ciertas ocasiones. Por lo cual se deciidó el hacer otro código mas que no involucrara las poses y solamente se dejara controlar por el teclado, exclusivamente para la performance.

Condiciones lumínicas, observaciones

Las pruebas lumínicas arrojaron las siguientes observaciones:

1) La luz y su efectos en la percepción de los colores

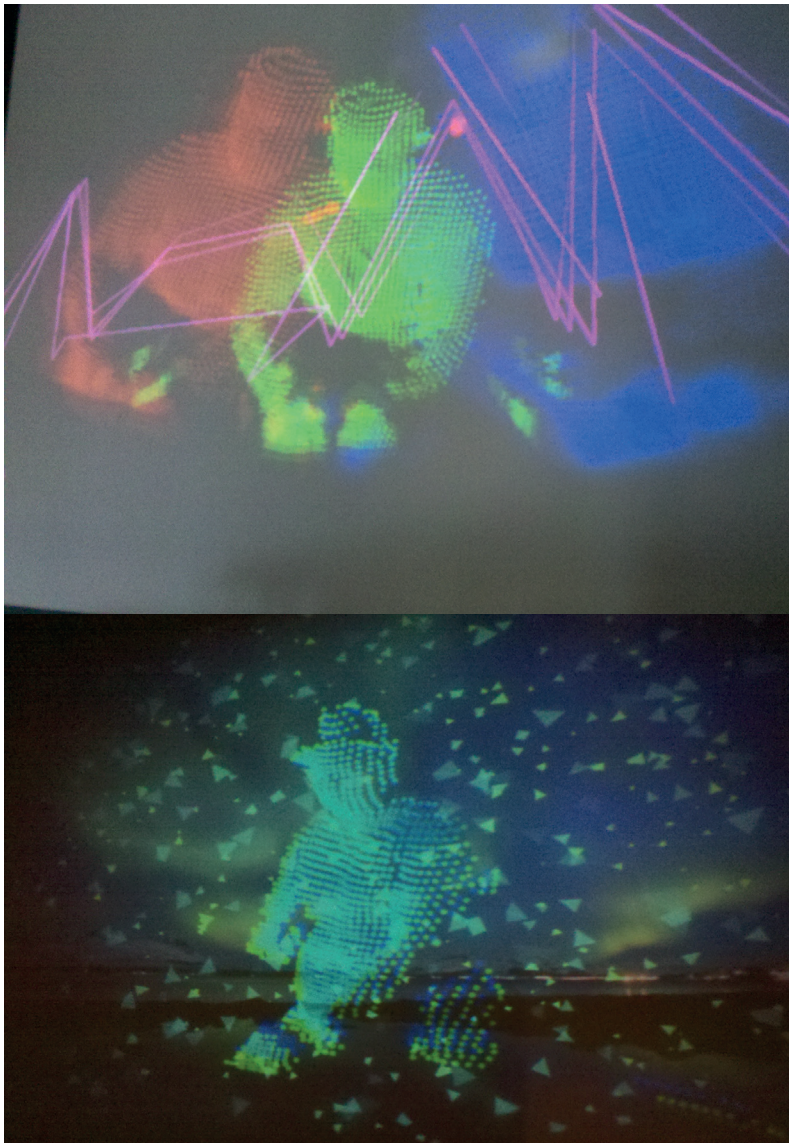
Los colores se ven más opacos al ser proyectados que en la pantalla, por lo cual, tienen que ser más saturados (tener mayor saturación).
El negro “no se ve”, los colores muy luminosos , como los amarillos claros, se ven blancos.

2) Su efecto en las formas

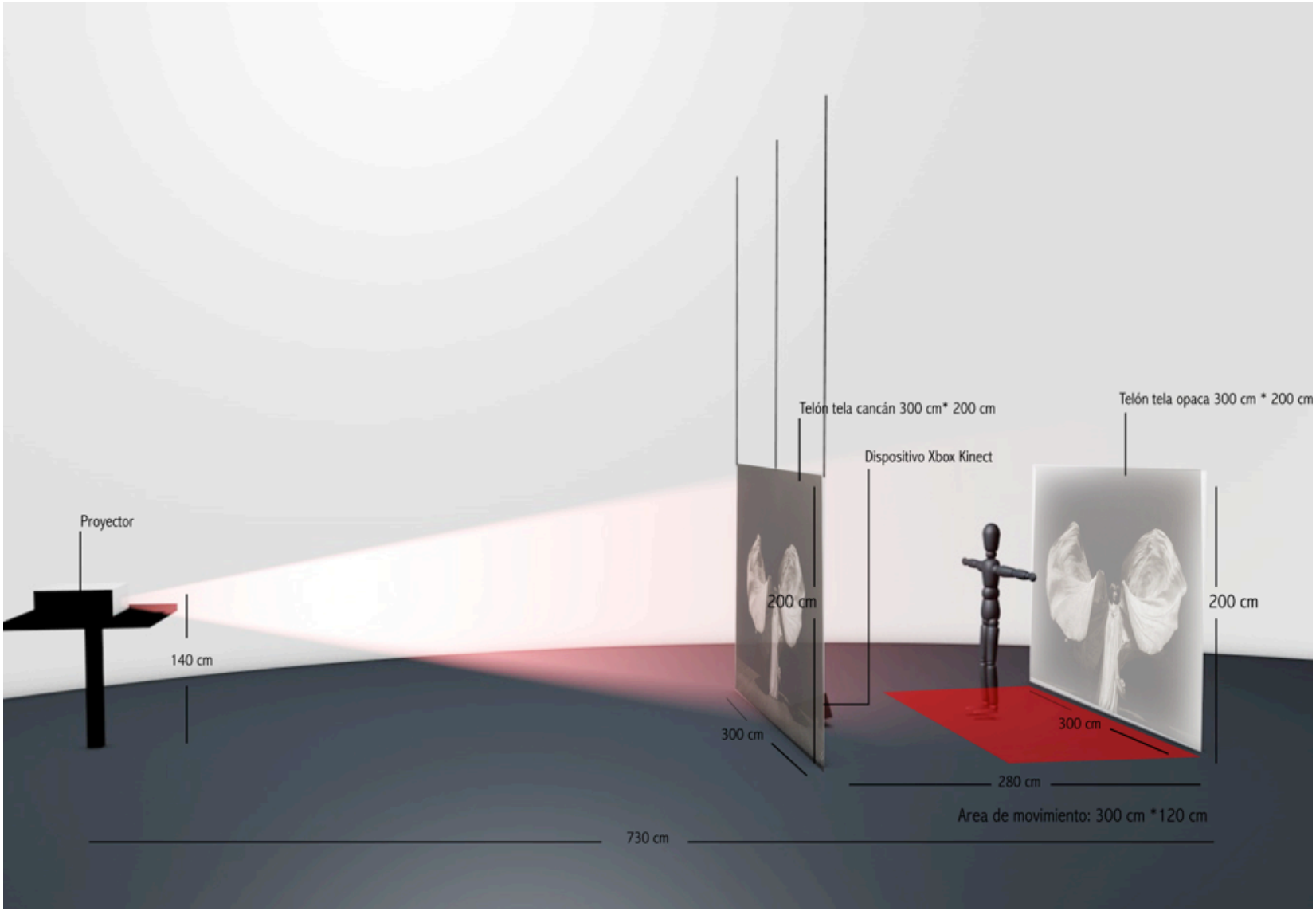
En el desarrollo gráfico de la pose o escena 5 (ira), las líneas del sistema de partículas eran inicialmente muy delgadas, por lo cual se perdían en la proyección.
No se distinguían del fondo. Se hace gruesa la línea en el siguiente códigoEsto pasa dadas las condiciones específicas de este caso. El ambiente será con luz de día, dentro de un espacio cerrado pero con ventanales, es por eso que entra mucha luz. También por la calidad del proyector, aunque es bastante buena, si se quiere mayor definición para las animaciones, y en otras condiciones, se necesitan proyectores de más lúmenes ASI (luminosidad, cantidad de luz) y mejor resolución.

Distancias, observaciones

Las distancias son elementales para que se pueda capturar la imagen y para que se pueda proyectar de manera adecuada. La posición del dispostivo Kinect con relación a la persona también es una variable definitiva para la visualización de la gráfica. En este caso el dispositivo está anguloso, en contrapicado hacia el personaje, por lo cual la imagen se va a ver ligeramente deformada. Para ver al distribución de espacio y sus distancias, ver la página siguiente. En esta imagen se puede considerar que existen dos telones. Se puso uno transparentoso (tul llamado cancan) para emular doble fondo y la mejor visualización de las gráficas, las cuales se perdían si eran proyectadas directamente en el cuerpo de la bailarina. Para elegir el telón se experimento con distintas telas, siendo escogida finalmente el cancán, pues recibía la luz de mejor manera, no reflejándolo y no absorbiendo tanto la luz. Para ver las pruebas, en las siguientes páginas hay fotos de la experimentación con la tela.



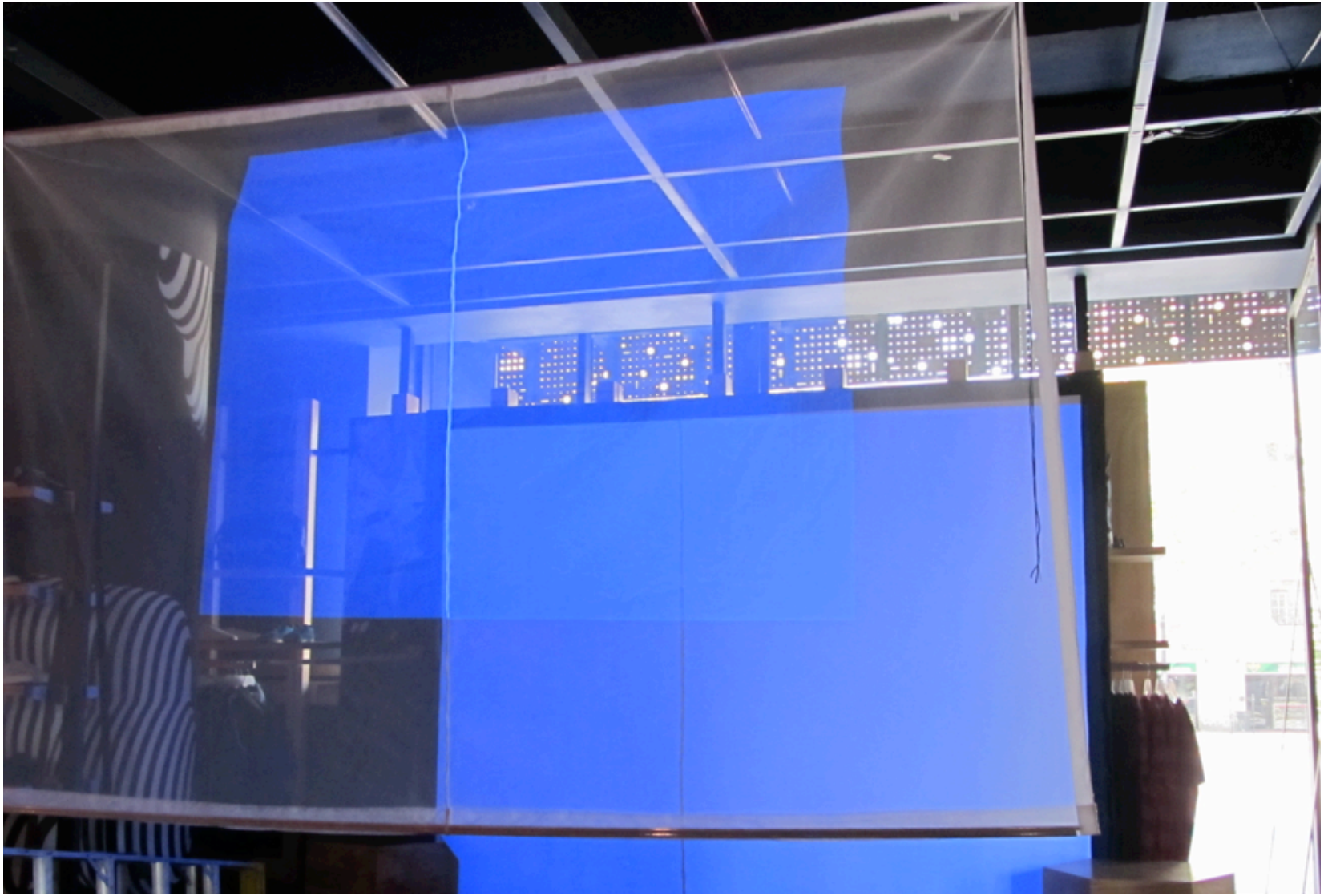
Prueba Lumínica realizada para el festival de Puma Lab, 2012



Montaje con distancias

Prueba de tela para doble fondo: en la imagen, el cancán.

Prueba de tela para generar doble fondo: En este caso vemos el tul clásico. Asimila menos la luz, pues relfeja mas la luz, al ser sus celdas mas cercanas que en el can can.



Fotografía tomada el primer ensato con doble telón, lugar: puma lab

Conclusiones

Al comenzar la realización del documento, la finalidad principal era generar una instancia de diseño experimental basado en la necesidad de sistematizar conocimientos transferibles, de un método fundamentado en la aplicación de los tan llamados Nuevos Medios.
Es así como surge el planteamiento de aplicar métodos de diseño de muestras para documentar la captura del lenguaje corporal por medio de ejercicios, que a su vez, al poder mezclarlos, visualizan la potencialidad de producir una amplia gama de combinaciones y potencialidades.

Estas muestras o ensayos culminan con este documento primario, el cual su destino es siempre ir transformándose y corrigiendose. Espero que les sirva a la gente que recién esté empezando y que puedan tomar los ejercicios para definir nuevas áreas de uso de la interfaz gestual, con el fin de aumentar nuestro conocimiento entorno a este ámbito.

A continuación se expone una serie de fotografías tomadas en el primer ensayo con el telón y el código finalizado.

Agradecimientos

Qusiera agradecer la colaboración a los desarrolladores y artistas:

Oscar LLauquén (Galactic dog)
José Concha
Orlando Bascuán
Diego Gómez
Nicole Sazo
A todo el equipo Delight Lab
Christián Oyarzún
Roy Mac Donald
A todo el equipo Puma Lab

Bibliografía

Borenstein, G. (2012). *Making Things See.* Sebastopol, CA: O'Reilly.

Greenberg, I. (2007). *Processing: Creative coding and computational art*. New York, NY: friendsofED.

Jean, J. (2012). *Kinect Hacks.* Sebastopol, CA: O'Reilly.

Kean, S., Hall, J. & Perry, P. (2011). *Meet the Kinect: Programming and scripting natural user interfaces*. New York, NY: friendsofED.

Machanick, P. (1994). *C and C++ in 5 days.*

Noble, J. (2009) . *Programming interactivity*. Sebastopol, CA: O'Reilly.

Shiffman, D. (2008). *Learning processing: A beginner's guide to programming images, animation, and interaction*. Burlington, MA: Morgan Kaufmann.

Reas, C. & Fry, B. (2007). *Processing: A programming handbook for visual designers and artists*. Cambridge, MA: MIT Press books.