Desarrollo de un sistema óptico para Interfaces Tangibles (mesa con pantalla reactiva)

Autor: Emiliano Causa

Este trabajo pertenece a los siguientes proyectos de investigación: "Convergencia de Realidad y Virtualidad en el campo de las Realidades Mixtas y la Realidad Aumentada en el campo de las artes multimediales." (Director Carmelo Saitta, Codirector Emiliano Causa) perteneciente al programa de Incentivos para la Investigación Docente de la SPU. Proyecto de investigación "Diseño y desarrollo de aplicaciones e interfases de Realidad Aumentada destinadas a síntesis y procesamiento de audio digital." (Directores Carmelo Saitta y Pablo Cetta)perteneciente al programa PICTO-ARTE
Ambos proyectos del Área Transdepartamental de Artes Multimediales del Instituto Universitario Nacional del Arte (IUNA) – Yatay 843 Ciudad Autónoma de Buenos Aires Tel. 54-11-4862-8209
emiliano.causa@gmail.com

Palabras claves

Interfaces Tangibles, Realidad Aumentada, Sistema de proyección con espejos

1. Introducción

El objetivo de este texto es mostrar el desarrollo de una mesa para pantalla sensible al tacto e interfaces tangibles. La idea consiste en diseñar una mesa que logre maximizar el tamaño de la pantalla, conteniendo la altura de la mesa dentro de un parámetro ergonómico. El problema surge a partir de que las pantallas para interfaces tangibles utilizan un proyector de video para retroproyectar la imagen que emite la pantalla. El proyector se coloca dentro del mueble, debido a que el común de los proyectores de video poseen una relacion de 1:1,4 a 1:1,8 (es decir que para que el proyector genere una imagen de 1 metro, requieren entre 1,5 y 1,8 metros de distancia a la superficie de proyector. Si pensamos que la pantalla debe estar en una posición horizontal, si queremos que la misma sea de 1 metro de ancho, se necsitaría que la mesa tenga entre 1,5 y 1,8 metros de alto. Obviamente una mesa con estas dimensiones no resulta práctica para el uso de una persona de altura promedio.

El siguiente capítulo expone el funcionamiento del sistema ReacTiVision, el mismo está extraído de otro artículo del autor y se duplica aquí a fines de ilustrar el sistema:

1.1. ReacTiVision y las Interfaces Tangibles (*)

ReacTiVision (mtg.upf.es/reactable/) es una herramienta de software desarrollada por Sergi Jordà, Martin Kaltenbrunner, Günter Geiger y Marcos Alonso, quienes conforman el Grupo de Tecnología Musical dentro del Instituto Audiovisual en la Universidad Pompeu Fabra (Barcelona España). Esta tecnología permite reconocer patrones bitonales (llamados "fiducials") impresos a piezas de interfaces tangibles que funcionan sobre una pantalla sensible al tacto. Esta interface tangible consiste en piezas de acrílico que se apoyan sobre una pantalla sensible, esta es capaz de reconocer las piezas, gracias a los patrones bitonales y generar respuestas audiovisuales en consecuencia. Los creadores, construyeron este software para desarrollar una pantalla sensible para interpretación musical en tiempo-real (un instrumento de improvisación de música electrónica) llamada ReacTable.

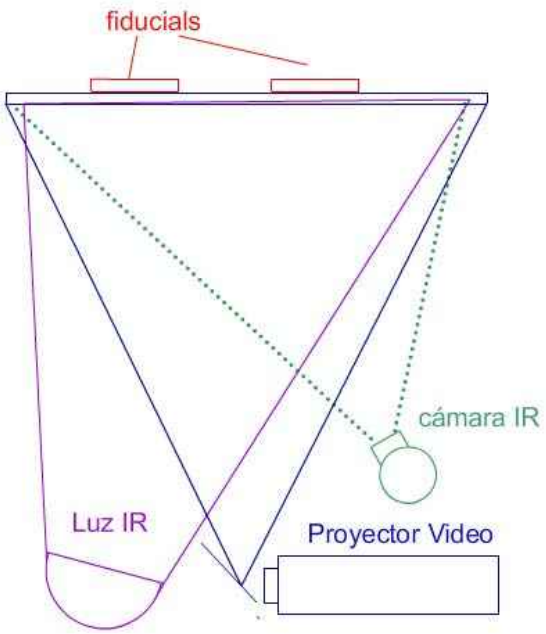


Figura: Esquema ReacTable

Como muestra la figura 3, ReacTiVision permite hacer el reconocimiento de patrones bitonales, a través de un sistema óptico, que en el caso de la ReacTable se implementa con luces y cámara infrarrojas. La pantalla es un acrílico con superficie esmerilada, las imágenes se retro-proyectan desde abajo usando un cañón de video, a su vez una luz infrarroja permite iluminar los patrones que serán captados por una cámara, también infrarroja. Dicha luz y cámara son infrarrojas para no interferir la luz del proyector de video (que pertenece al rango visible de la luz), y para que la cámara no vea a su vez las proyecciones.

2. Modelo inicial

Como modelo inicial se partió de un sistema de 3 espejos que utilizamos en un proyecto personal (anterior a esta investigación) y que había mostrado eficacia para reducir la altura de una mesa con una pantalla de retroproyección, el sistema en cuestión utiliza otra tecnología de captura y la superficie de proyección es una tela elástica, pero el principio de reducción de altura y ampliación del tamaño de la pantalla, es el mismo.

A partir de este modelo se tomó la decisión de ponerlo a prueba con un sistema que permitiera medir su eficiencia a la hora de maximizar el tamaño de la pantalla.

3. Requisitos para el diseño de la mesa

A la hora de diseñar la mesa existen un conjunto de restricciones que la misma debería respetar:

1. La altura de la mesa no debería sobrepasar los 115 cm.
2. La utilización de espejos no debería producir deformaciones en la imagen.
3. La utilización de espejos no debería producir reflejos que dupliquen porciones de la imagen (superposiciones).
4. Los espejos deberían estar ubicados de forma tal que no proyecten sombra sobre la imagen.
5. Los espejos deberían permitir ubicar al proyector en una posición horizontal (dado que algunos proyectores no pueden ser usados en posiciones oblícuas).

Para probar el diseño se desarrolló una algoritmo que permite visualizar (modelizar) el as de luz del proyector y sus reflejos en el sistema de espejo.

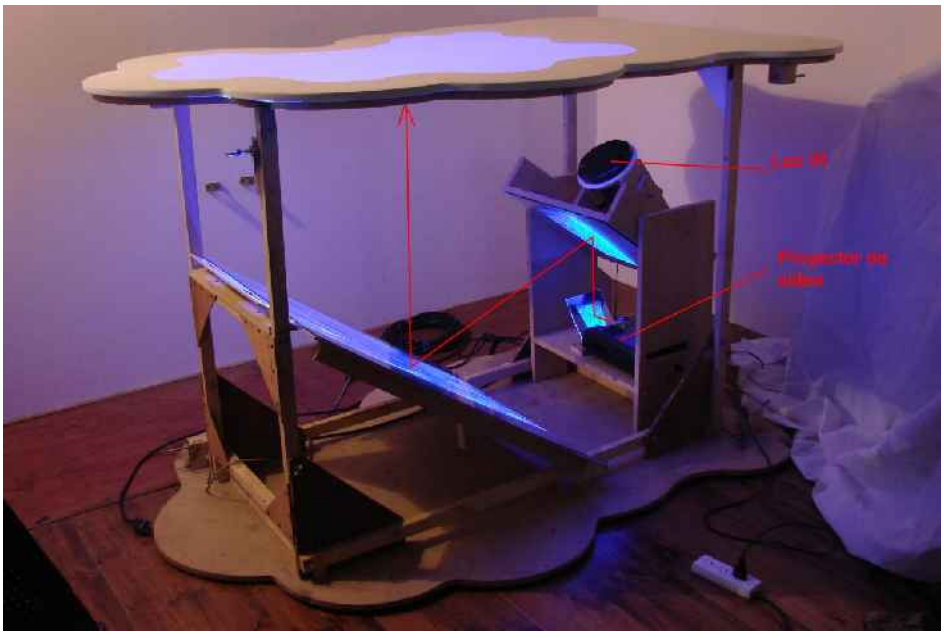
4. Sistema de 3 espejos

Como se pudo apreciar en el modelo original, un sistema de 3 espejos permiten resolver algunos de estos requisitos.

El espejo más pequeño permite reflejar el as del proyector de forma tal que el dispositivo se encuentre horizontal y la imagen se emita en forma vertical.

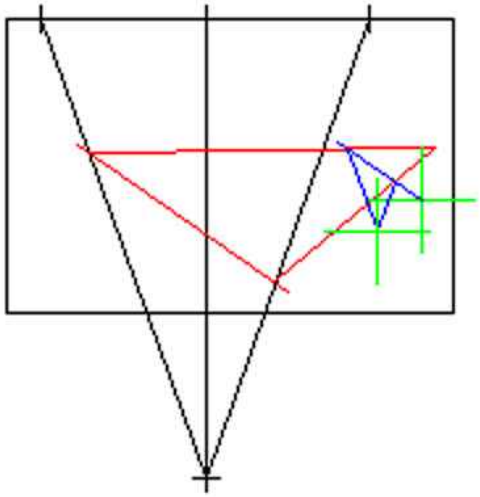
Los espejos, medio y grande permiten desviar el as para extender la longitud del mismo (es decir la distancia entre el proyector y la pantalla) sin estender la altura de la mesa.

A continuación se puede ver una foto del modelo original y la forma en que los espejos funcionan:



La proyección entre el espejo medio y el grande puede producir un efecto de trapecio sobre la imagen deformándola. La forma de solucionar esto es logrando que ambos espejos estén ubicados en forma paralela, es decir con el mismo ángulo, ya que la deformación que produce el espejo medio es corregida por el grande.

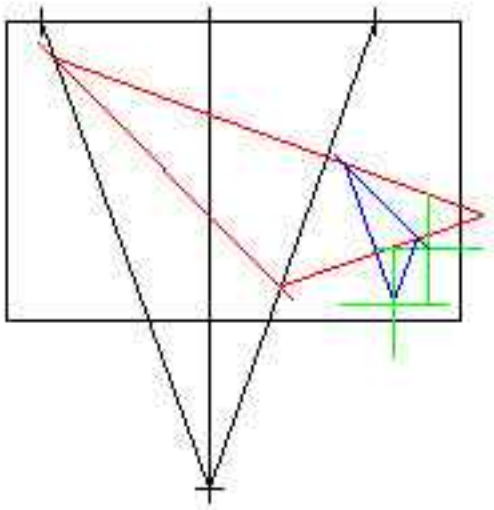
El objetivo del proyecto es hacer la mesa más chica posible que maximice el tamaño de la pantalla, por eso en el modelo computacional se partió de una caja de 112 cm (para dejar un margen para el apoyo en el piso) y se proyecto sobre su superficie una pantalla de 125cm de ancho. La relación entre ancho y distancia del as se estableción en 1:1,4. Este as se puede apreciar en el gráfico hecho por el modelo, es el triángulo de color negro. El vértice inferior corresponde a la posición en la que tengría que estar el proyector si no hubiese espejos.



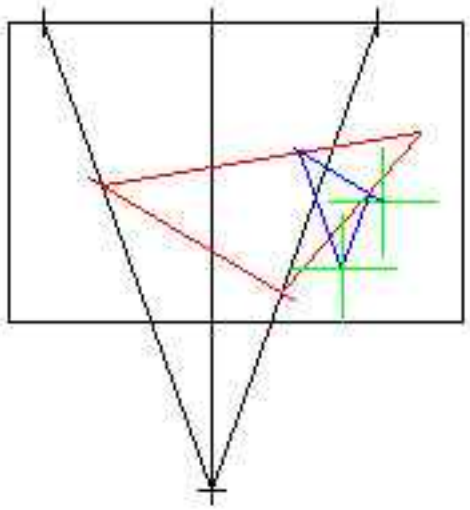
En rojo se grafica el espejo grande y la forma en que este refleja el as. Aquí cabe aclarar que el sentido de la luz es inverso, es decir que la luz sale del proyector y viaja hacia la pantalla, pero en el modelo se parte desde la pantalla por que este es un elemento constante (inamovible del sistema), mientras que las posiciones de los espejos y el proyector son los elementos variables del mismo.

Por último, el azul se puede ver al espejo medio y su reflejo. El vértice del as azul muestra la posición del proyector. El espejo chico sirve a los fines de verticalizar el as con el proyector puesto en forma horizontal. El tipo de modificación que produce en la longitud del as es despreciable y no es necesario incluirlo en el sistema.

En el gráfico mostrado arriba, el ángulo de los espejos es 34,75º, esto permite que los mismos no generen sombras sobre el as ni reflejos parásitos.



En esta nueva gráfica los espejos se ubican a 45,15º de ángulo. Con este nuevo ángulo, los tamaños de los espejos se agrandan de a 39 a 50cm (espejo medio) y 97 a 134cm (espejo grande), lo que encarece el mueble y reduce el espacio para el resto del equipo, a la vez que deja menos margen de error. Esto es importante dado que no siempre se cuenta con un proyector que respete la relación 1:1,4 y el margen de error permite absorver estos cambios.También se reduce el espacio para ubicar el proyector.



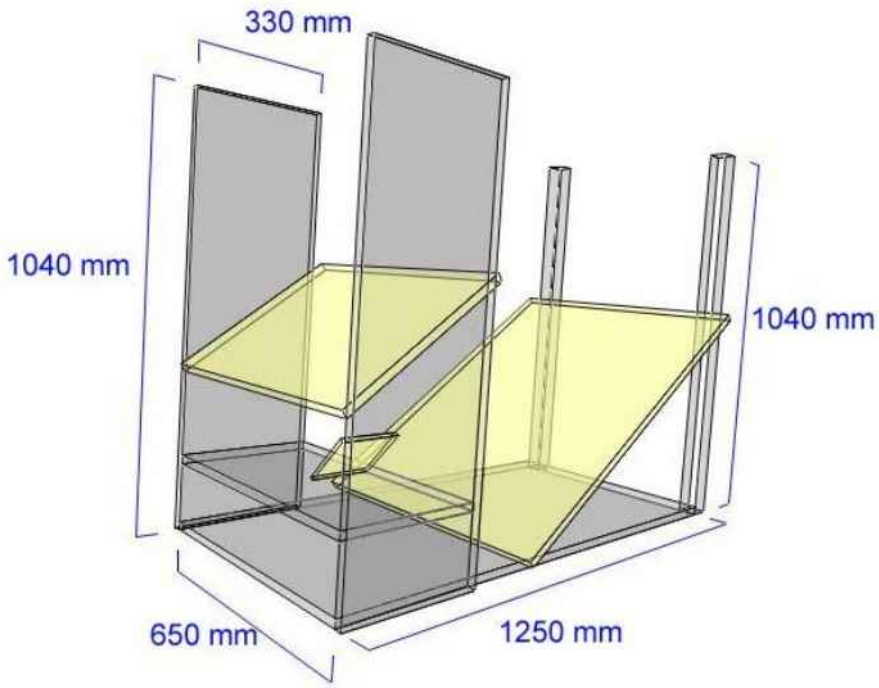
Aquí mostramos otro ejemplo en donde el ángulo se modificó a 30,57º. Como se puede ver en el gráfico, el espejo medio hace sombra en el as que parte del espejo grande, y dada la posición es probable que el proyector también lo haga. Desde esta perspectiva, los ángulos cercanos a las 35º son buenos para un as que sale de un proyector de relación 1:1,4.

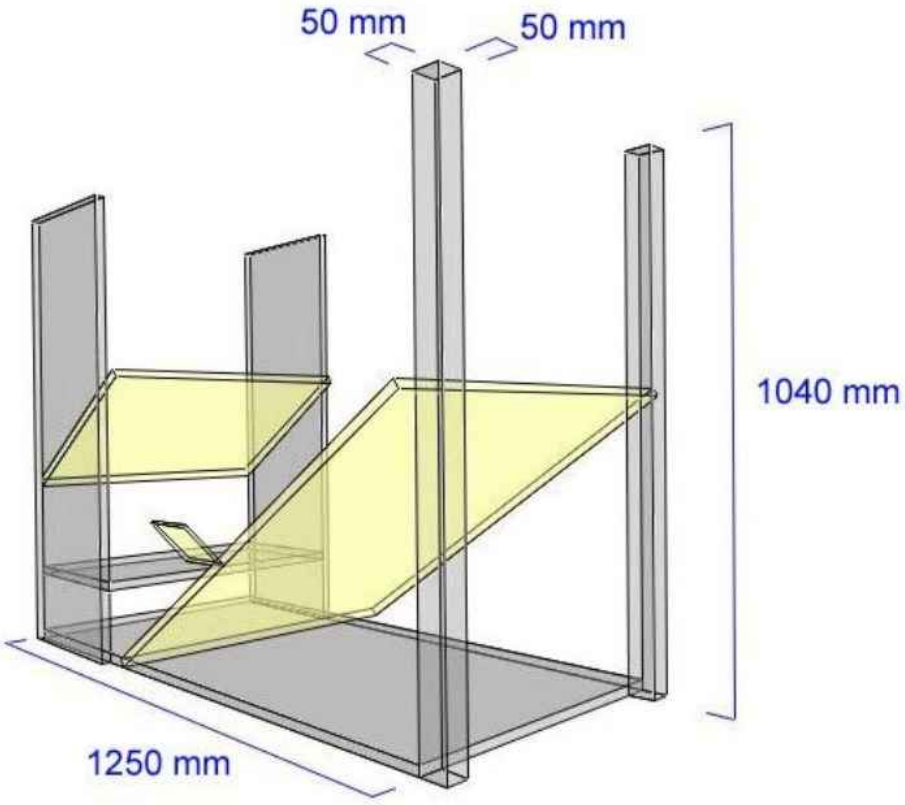
4.1. Del método

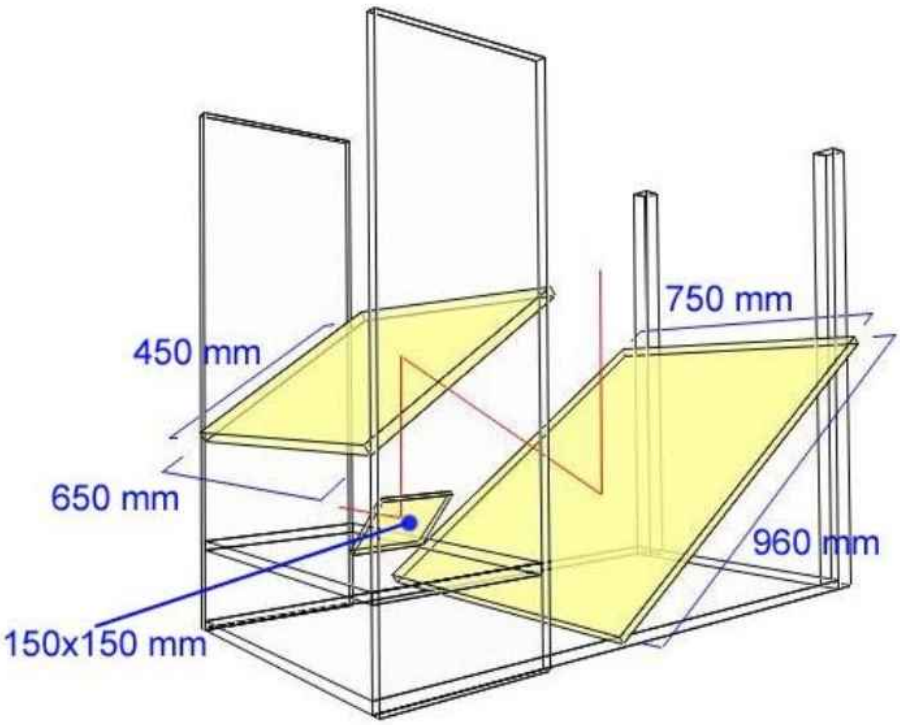
Esta forma de simulación permite probar diferentes ángulos y posiciones hasta entontrar los adecuados para cumplir con los requisitos del diseño. Es cierto que sería posible realizar un programa que busque por sí mismo dichas posiciones y ángulos haciendo el recorrido de un espacio solución, el equivalente a plantear un sistema de funciones a ser maximizadas, cuyas variables son las posiciones y los ángulos. Pero dado que la mesa está pensada para tener cierta tolerancia respecto del tipo de proyector (específicamente, respecto de su relación ancho/profundidad), estas funciones son más complejas de ser definidas. En este punto, la complejidad y tiempo necesario para programar este algoritmo, puede ser reemplazado por un trabajo más sencillo y la utilización de un ciertos criterios heurísticas, como se ha hecho en este caso.

4.2. Diseño de la estructura interna

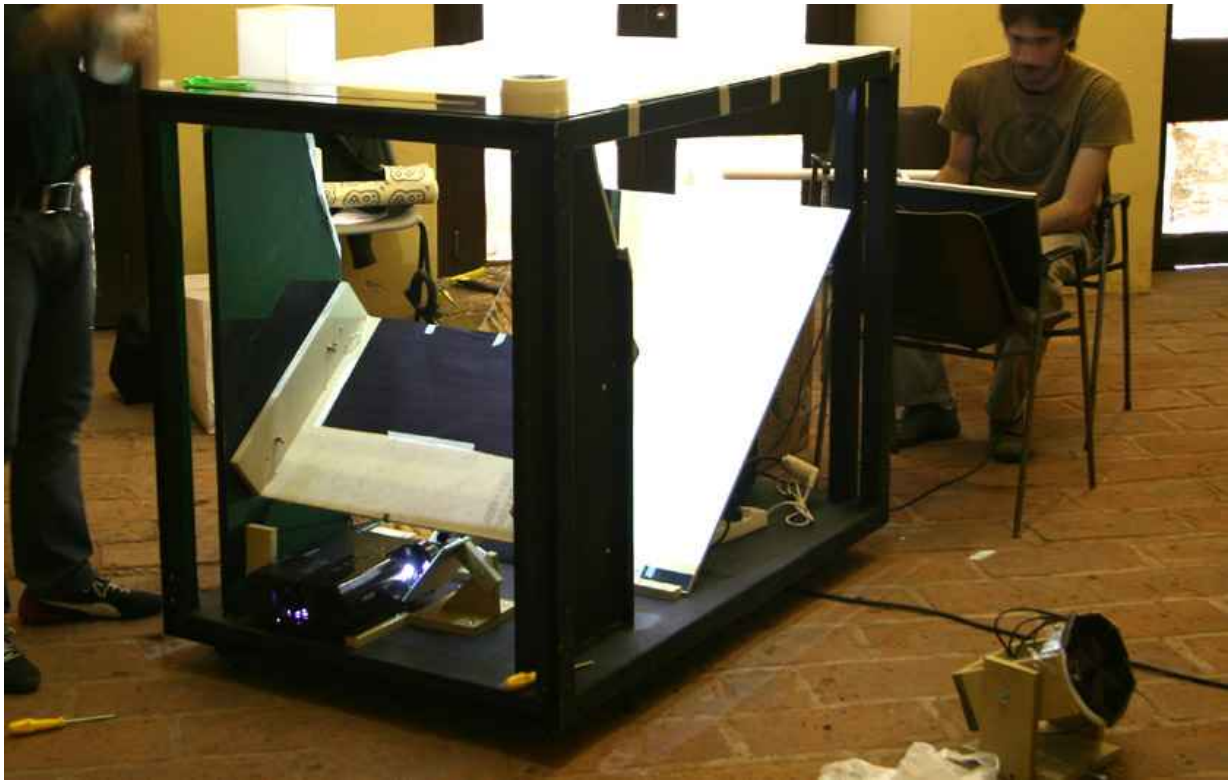
En función de los resultados obtenidos con el programa se diseño la esrtuctura interna del mueble. A continuación puede verse una serie de gráficos que muestra sus medidas:


330 mm
1040 mm
1040 mm
650 mm
1250 mm

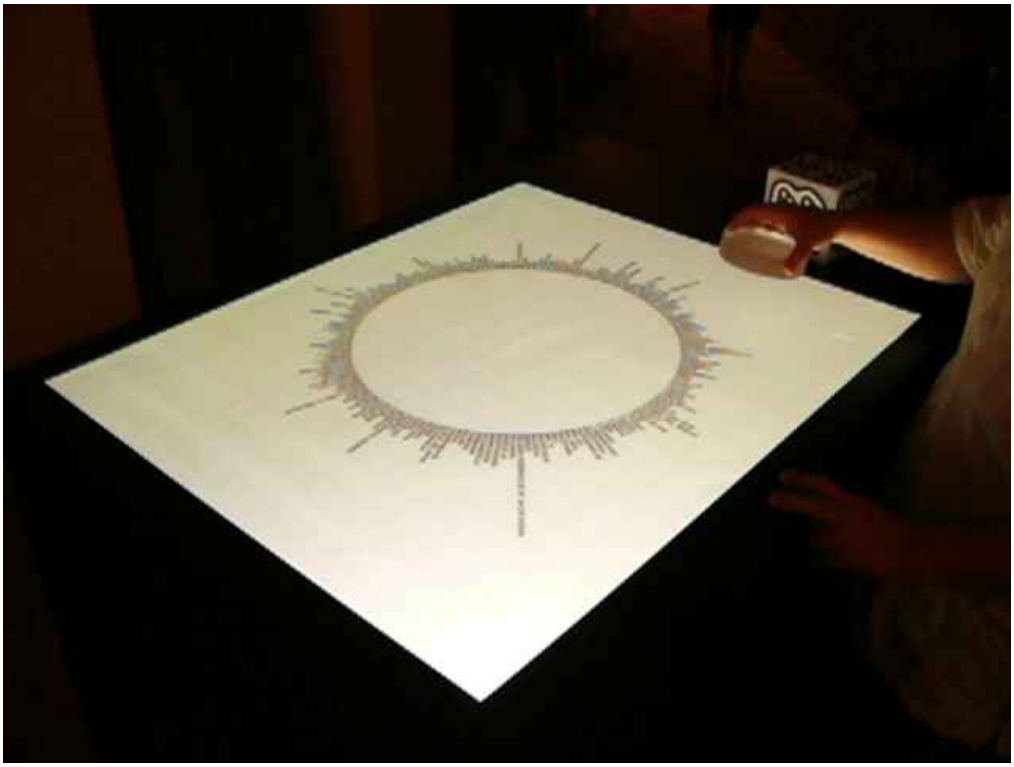

50 mm
50 mm
1040 mm
1250 mm



La construcción de la estructura, así como el diseño de la estructura externa del mueble estuvo a cargo de Diego Pimentel y Mariano Cataldi. Debajo se puede ver una foto del mueble construido antes de ser cerrado con sus tapas.



Las siguientes fotos muestram el mueble ya cerrado:



5. Deficiencias del diseño

El diseño realizado a partir del uso del programa resultó eficiente a la hora de realizar la proyección de la imagen, coincidiendo en la realidad con los resultados mostrados en el modelo computacional. Sin embargo aparecieron problemas en la implementación con aquellos elementos que no estuvieron en el modelo, particularmente con el sistema de captación.

5.1. Ubicación de la cámara

El primer problema surge a la hora de querer ubicar la cámara adecuado para poder captar la totalidad de la pantalla. El tamaño de la pantalla es de 125cm de ancho, siendo que la cámara tiene una relación entre el ancho del cuadro y la distancia cercana a 1:1,10 , la distancia a la que tendría que estar una cámara es aproximadamente de 137cm. Obviamente la altura del mueble no alcanza y por otra parte la ubicación del espejo grande impide poner la cámara en esa posición, sin embargo es posible ubicar a la cámara mirando hacia el espejo grande para extender la distancia a la pantalla.

En el complejo sistema de espejos, pensados para el proyector, las posiciones que la cámara puede usar no son las más adecuadas, obligando a obtener cuadros torcidos y deformados de la pantalla. Este defecto, por suerte es corregible con el sistema de calibración que utiliza ReacTiVision.

5.2. Resolución del sistema

Otra paradoja que surge del agrandamiento de la pantalla es que el tamaño que debe alcanzar los patrones (fiducials) es mayor ya que la resolución de la cámara no alcanza para captar patrones pequeños. Esto no es en sí una deficiancia del diseño, ya que es corregible mediante el uso de cámaras de mayor calidad.

5.3. La superficie de la pantalla y los reflejos

Uno de los mayores focos de problema está en el material que se elige para usar de superficie de retroproyección (este tema es ampliamente tratado en el texto de Diego Pimentel). Esta superficie debe cumplir con una serie de requisitos complejos de lograr:

1. **Resistencia:** debe ser lo suficientemente resistente para permitir apoyar objetos y posar los dedos para interactuar (a esto se suma el hecho de que la altura de la mesa invita al público a apoyarse sobre esta).
2. **Opacidad**: debe ser lo suficientemente opaca para permitir ver nítidamente (desde abajo) sólo los objetos apoyados, pero no aquellos por encima. Por ejemplos, debe poder verse los dedos apoyados, pero no la mano por encima de ellos, ni las caras de las personas.
3. **Transparencia y nitidez:** tambien debe poder ser lo suficientemente transparente para dejar ver la imagen proyectada por detrás, pero lo suficientemente opaca como para atrapar la luz y no dejarla pasar. Por ejemplo: si es muy transparente, la proyección seguirá de largo (proyectándose en el techo, en vez de la pantalla) , si es muy opaca no podrá verse bien la imagen (hasta puede perder nitidez).
4. **Antireflejo:** la superficie debajo de la pantalla debe poseer un tratamiento antireflejo ya que el sistema de captura es muy sensible a los altos contrastes, principalmente si los producen los reflejos en este superficie. Cuando la superficie no es antireflejos, funciona como un espejo que refleja el sistema de iluminación.

En nuestra implementación no pudimos encontrar una superficie que cumpla con todos estos requisitos. El material que mejor cumplía el segundo y tercer requisito es el papel vegetal (de tipo calco), pero claramente no cumplía el primer requisito, lo que obligó a apoyarlo sobre un acrílico que no cumplía con el cuarto requisito.

5.4. La iluminación

La iluminación que este sistema utiliza es infrarroja. Nosotros optamos por utilizar lámparas incandecentes filtradas con acetatos de color rojo y azul, que permiten generar luz infrarroja con focos comunes. Este tipo de iluminación tiene como ventaja su bajo costo y la facilidad de encontrar esos elementos en el mercado de nuestro pais. Una de las más importantes desventajas es el calor que genera, lo que obliga a tomar medida de ventilación. El sistema alternativo que más se utiliza es el de reflectores de leds, que no genera calor y tienen una mayor durabilidad.

El mayor problema encontrado con la iluminación fue con la superficie de la pantalla, que al no ser antireflejo funcionaba como un espejo más. Este pseudo-espejo termina haciendo que la cámara vea las lámaras, lo que produce ruido en la captación de patrones. Debido a esto nos vimos obligados a poner la lámparas lo más apartadas hacia los lados posible, para que no entre en el área del reflejo, pero esta solución genera un tipo de iluminación desigual que complica nuevamente la captura.

5.5. Conclusión respecto de la deficiancias

Las deficiencias que hemos enunciado obligan a repensar en el futuro el diseño de este mueble, integrando a su evaluación estas nuevas variables que se han encontrado durante la implementación. Particularmente, la posición de la cámara, la posición y el tipo de iluminación, y el material de la superficie de proyección. Considero que los logros obtenidos en cuanto a la extensión del tamaño de la pantalla no terminan de justificar algunos de los problemas encontrados, sin embargo la mayor parte del problema se relaciona con el tipo de material y si este factor se resolviese el balance del diseño sería positivo y simplemente habría que corregir la posición de la cámara y la iluminación, pero de seguro sería sencillo de resolver.

6. Algoritmo de cáculo de reflexiones

A continuación se muestra el algoritmo utilizado para el cáculo de la reflexiones. Este se organiza a través de una serie de objetos (clases) que son: Cono_Luz (que es usado para modelar el as de luz), Segmento (utilizados para calcular las reflexiones) y Punto (para establecer ciertas posiciones en el modelo).

La principal parte del algoritmo es la función “reflejar” que permite establecer la reflexión de una as de luz en un espejo:

```
void reflejar( Segmento espejo , Segmento original , Segmento antes , Segmento despues ){

  float anguloEspejo = angulo( espejo );
  float anguloOriginal = angulo( original );

  float largoTotal = largo( original );

     Punto contacto = obtieneCruceDosLineas( espejo.a , espejo.b , original.a , original.b );

  antes.iniciar( original.a , contacto );
  float largoAntes = largo( antes );

  float diferencia = anguloOriginal-anguloEspejo;
```

```
  float nuevoAngulo = anguloEspejo - diferencia;
  float largoDespues = largoTotal-largoAntes;

  despues.iniciar( contacto , largoDespues , nuevoAngulo );
}
```

7. Referencias bibliográficas

[1] www.processing.org

[2] mtg.upf.es/reactable/

[3] www.biopus.com.ar

(*) Este capítulos está extraído del artículo “Desarrollo de una Aplicación con Interfaces Tangibles” del mismo autor, fueron agregados (a pesar de la obvia duplicación) debido a la pertinencia de sus contenidos para el presente texto.

8. Anexo: código del algoritmo

```
Cono_Luz cono;

float mx = 100;
float my = 300;

float botton = 0;
float top = -112;
float left = 0;
float right = 170;

void setup(){

  size( 800 , 600 );
  background( 255 );
  rectMode(CORNERS);
  noFill();

  translate( mx , my );

  float anchoPantalla = 125;
  float relacion = 1.4;
  cono = new Cono_Luz( anchoPantalla , relacion );

  float profundidad = cono.profundidad;
  float margen = 13;
  float xproyector = margen + anchoPantalla/2;
  float yproyector = profundidad + top;
  cono.ubicar( xproyector , yproyector , radians(270) );

  cono.dibujar();

  float correr = 0;
  float x1esp = 107+correr;
  float y1esp = -8.5;
  float x2esp = 27;
```

```
  float y2esp = -64+correr;

  Segmento espejo = new Segmento( x1esp , y1esp , x2esp , y2esp );

  float anguloEspejo = angulo( espejo );
  float tamanioEspejo = largo( espejo );

  println( "anguloEspejo = " + (180+degrees( anguloEspejo )) );
  println( "tamanioEspejo = " + tamanioEspejo );

  rect( left , top , right , botton );

  Segmento antes_1 , despues_1 , original_1 ;
  original_1 = new Segmento( cono.extremoR , cono.foco );
  antes_1 = new Segmento();
  despues_1 = new Segmento();
  reflejar( espejo , original_1 , antes_1 , despues_1 );

  Segmento antes_2 , despues_2 , original_2 ;
  original_2 = new Segmento( cono.extremoL , cono.foco );
  antes_2 = new Segmento();
  despues_2 = new Segmento();
  reflejar( espejo , original_2 , antes_2 , despues_2 );

  stroke(255,0,0);
  espejo.dibujar();
  despues_1.dibujar();
  despues_2.dibujar();

  Punto baseEspejo2 = new Punto( 158, -43 );
  float tamEspejo2 = 39;
    Segmento espejo2 = new Segmento( baseEspejo2 , tamEspejo2 ,
anguloEspejo );

   Segmento antes_3 , despues_3 , original_3 ;
  antes_3 = new Segmento();
  despues_3 = new Segmento();
  reflejar( espejo2 , despues_1 , antes_3 , despues_3 );

  Segmento antes_4 , despues_4 , original_4 ;
  antes_4 = new Segmento();
```

```
  despues_4 = new Segmento();
  reflejar( espejo2 , despues_2 , antes_4 , despues_4 );

  baseEspejo2.dibujarCruz();
  stroke(0,0,255);
  espejo2.dibujar();
  despues_3.dibujar();
  despues_4.dibujar();

  despues_4.b.dibujarCruz();
  despues_4.b.string();
}
void draw(){

}

//CLASE CONO DE LUZ

class Cono_Luz{

  float relacion;
  float anchoPantalla;
  float profundidad;

  float x,y,angulo;

  Punto foco, centro, extremoR , extremoL;

  //----------------------------------------

  Cono_Luz( float anchoPantalla_ , float relacion_ ){

    anchoPantalla = anchoPantalla_;
    relacion = relacion_;

    profundidad = anchoPantalla * relacion;

    ubicar(0,0,0);
    calcularCono();
  }
```

```
  //-----------------------------------------

  void ubicar( float x_ , float y_ , float angulo_ ){
    x = x_;
    y = y_;
    angulo = angulo_;
    calcularCono();
  }
  //-----------------------------------------

  void calcularCono(){

    foco = new Punto( x , y );
    centro = puntoUbicadoA( foco , profundidad , angulo );
       extremoR = puntoUbicadoA( centro , anchoPantalla/2 , angulo +
HALF_PI );
       extremoL = puntoUbicadoA( centro , anchoPantalla/2 , angulo -
HALF_PI );
  }
  //-----------------------------------------

  void dibujar(){
    foco.dibujarCruz( color(0,0,0) , 5 );
    centro.dibujarCruz( color(0,0,0) , 5 );
    extremoR.dibujarCruz( color(0,0,0) , 5 );
    extremoL.dibujarCruz( color(0,0,0) , 5 );

    dibujaLinea( foco , extremoR );
    dibujaLinea( foco , extremoL );
    dibujaLinea( extremoL , extremoR );
    dibujaLinea( foco , centro );
  }
}

//-----------------------------------------------------------------
// FUNCIONES MATEMÁTICAS
//-----------------------------------------------------------------

float linea( float x , float x1 , float x2 , float y1 , float y2 ){
  return (x-x1)/(x2-x1)*(y2-y1)+y1;
}
```

```
//------------------------------------------------------------------

float lineaLim( float x , float x1 , float x2 , float y1 , float y2 ){
  float valor = (x-x1)/(x2-x1)*(y2-y1)+y1;
  valor = min(valor,y2);
  valor = max(valor,y1);
  return valor;
}
//------------------------------------------------------------------

float menorDistAngulos( float origen , float destino ){
  float distancia = destino - origen;
  return anguloRangoPI( distancia );
}
//------------------------------------------------------------------

float anguloRango2PI( float angulo ){
  float este = angulo;

  for( int i=0 ; i<100 ; i++ ){
    if( este >= TWO_PI ){
      este -= TWO_PI;
    }
    else if( este < 0 ){
      este += TWO_PI;
    }
    if( este >= 0 && este <= TWO_PI ){
      break;
    }
  }
  return este;
}
//------------------------------------------------------------------

float anguloRangoMenos2PI( float angulo ){
  float este = angulo;

  for( int i=0 ; i<100 ; i++ ){
    if( este > 0 ){
      este -= TWO_PI;
```

```
    }
    else if( este <= -TWO_PI ){
      este += TWO_PI;
    }
    if( este > - TWO_PI && este <= 0 ){
      break;
    }
  }
  return este;
}
//--------------------------------------------------------------------

float anguloRangoPI( float angulo ){
  float este = angulo;
  for( int i=0 ; i<100 ; i++ ){
    if( este > PI ){
      este -= TWO_PI;
    }
    else if( este <= -PI ){
      este += TWO_PI;
    }
    if( este >= -PI && este <= PI ){
      break;
    }
  }
  return este;
}
//--------------------------------------------------------------------
// CLASE: SEGMENTO
//--------------------------------------------------------------------
class Segmento{
  Punto a,b;
  //--------------------------

  Segmento(){
    a = new Punto();
    b = new Punto();
  }
  //--------------------------
```

```
  Segmento( Punto a_ , Punto b_ ){
    iniciar ( a_ , b_ );
  }
  //--------------------------

  Segmento( float x1 , float y1 , float x2 , float y2 ){
    iniciar( x1 , y1 , x2 , y2 );
  }
  //--------------------------

  Segmento( Punto a_ , float distancia , float angulo ){
    iniciar( a_ , distancia , angulo );
  }
  //--------------------------

  void iniciar( float x1 , float y1 , float x2 , float y2 ){
    a = new Punto( x1 , y1 );
    b = new Punto( x2 , y2 );
  }
  //--------------------------

  void iniciar( Punto a_ , float distancia , float angulo ){
    a = new Punto();
    a.copiarDe( a_ );
    b = puntoUbicadoA( a , distancia , angulo );
  }
  //--------------------------

  void iniciar ( Punto a_ , Punto b_ ){
    a = new Punto();
    b = new Punto();
    a.copiarDe( a_ );
    b.copiarDe( b_ );
  }
  //--------------------------

  void dibujar(){
    line( a.x , a.y , b.x , b.y );
  }
}
```

```
//--------------------------
float largo( Segmento este ){
  return dist( este.a.x , este.a.y , este.b.x , este.b.y );
}
//--------------------------
float angulo( Segmento este ){
  return atan2( este.b.y - este.a.y , este.b.x - este.a.x );
}
//--------------------------
void reflejar( Segmento espejo , Segmento original , Segmento antes , Segmento despues ){

  float anguloEspejo = angulo( espejo );
  float anguloOriginal = angulo( original );

  float largoTotal = largo( original );

    Punto contacto = obtieneCruceDosLineas( espejo.a , espejo.b , original.a , original.b );

  antes.iniciar( original.a , contacto );
  float largoAntes = largo( antes );

  float diferencia = anguloOriginal-anguloEspejo;

  float nuevoAngulo = anguloEspejo - diferencia;
  float largoDespues = largoTotal-largoAntes;

  despues.iniciar( contacto , largoDespues , nuevoAngulo );
}
//-----------------------------------------------------------------
// CLASE: PUNTO
//-----------------------------------------------------------------
class Punto{
  float x,y;
  //--------------------------

  Punto(){
    iniciar( 0 , 0 );
  }
  //--------------------------
```

```
  Punto( float x_ , float y_ ){
    iniciar( x_ , y_ );
  }
  //--------------------------

  void iniciar( float x_ , float y_ ){
    x = x_;
    y = y_;
  }
  //--------------------------

  void copiarDe( Punto otro ){
    x = otro.x;
    y = otro.y;
  }
  //--------------------------
  void string(){
    println( "(  "+x+"  ;   "+y+"  )");
  }
  //--------------------------

  void dibujarCruz( ){
    dibujarCruz( color(0,255,0) , 20 );
  }
  //--------------------------

  void dibujarCirculo( color esteCol , float ancho ){
    stroke( esteCol );
    ellipse( x , y , ancho*2 , ancho*2 );
  }
  //--------------------------

  void dibujarCruz( color esteCol , float ancho ){
    stroke( esteCol );
    line( x-ancho , y , x+ancho , y );
    line( x , y-ancho , x , y+ancho );
  }

}
```

```
//--------------------------------
Punto puntoUbicadoA( Punto a , float distancia , float angulo ){
  Punto aux = new Punto();

  float xx = a.x + distancia * cos( angulo );
  float yy = a.y + distancia * sin( angulo );

  aux.iniciar( xx , yy );

  return aux;
}
//--------------------------------
float distancia( Punto a , Punto b ){
  return dist( a.x , a.y , b.x , b.y );
}
//--------------------------------
void dibujaLinea( Punto a , Punto b ){
  line( a.x , a.y , b.x , b.y );
}
//--------------------------------
Punto obtieneCruceDosLineas( Punto p1 , Punto p2 , Punto p3 , Punto p4 ){
  Punto aux = new Punto();

  float x = 0;
  float y = 0;

  float a = p2.y - p1.y;
  float b = p2.x - p1.x;
  float c = p4.y - p3.y;
  float d = p4.x - p3.x;

  if( b != 0 && d != 0 ){

    float ab = a/b;
    float cd = c/d;

    float e = p3.y - p1.y + p1.x*ab - p3.x*cd;

    float p = ab-cd;
```

```
    if( p != 0){
      x = e / p;
      y = linea( x , p1.x , p2.x , p1.y , p2.y );
    }

  }
  else{
    if( b == 0 && d != 0 ){
      x = p1.x;
      y = linea( x , p3.x , p4.x , p3.y , p4.y );
    }
    else if( b != 0 && d == 0 ){
      x = p3.x;
      y = linea( x , p1.x , p2.x , p1.y , p2.y );
    }
  }

  aux.iniciar(x,y);

  return aux;
}
```