UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de interfaz multi-táctil mediante el fenómeno de Reflexión Total Interna Frustrada

TESIS

QUE PARA OBTENER EL TÍTULO DE INGENIERO ELÉCTRICO-ELECTRÓNICO

PRESENTAN:

Bonilla Arroyo Juan Manuel
Quiroz Ledesma Luis Manuel
Romero Flores Luis Fernando
Sandoval Alejaldre Christian Antonio

Directora:
M.I. Norma Elva Chávez Rodríguez

Ciudad Universitaria, México2014

Agradecimientos generales

El equipo responsable de la presente tesis agradece encarecidamente a la Universidad Nacional Autónoma de México (UNAM), al Programa de Apoyo a la Titulación (PAT) de la Facultad de Ingeniería y al Palacio de Minería. Asimismo a nuestra asesora de tesis M.I. Norma Elva Chávez, al PAPIME PE100315 y al jurado de tesis: Mtro. Juan José Carreón Granados, Dra. TanyaMoreno Coronado, Ing. Gabriel Alejandro Jaramillo Morales y al Ing. Vicente Flores Olvera.

Agradecimientos de Juan Manuel Bonilla Arroyo

Un agradecimiento especial a mi madre Pilar Arroyo, quien ha guiado y procurado mi formación durante todos estos años, siempre enfocada hacia el mejor desarrollo de mis capacidades y a llevar una vida responsable y profesional. Estaré agradecido por siempre Mamá, me has mostrado el camino indicado y me has permitido ser quien soy.

Gracias a mi abuela Socorro Romero "Lala", por apoyarme desde pequeño y ser primordial en mi crecimiento; has estado en momentos muy importantes en mi vida, me has compartido tu sabiduría y tus experiencias, es un orgullo compartir este logro contigo.

Gracias a mi hermana Montserrat, quien ha sido un ejemplo a seguir en perseverancia, trabajo y talento. Me has enseñado a perseguir mis sueños con alegría, humor y actitud.

Gracias a mi hermano David por sus consejos y motivación en los caminos que hemos recorrido juntos. Tu astucia me ha permitido aprender mucho del mundo y me ha dado energías para explorar y seguir luchando. Que vivan los "Cuatro Fantásticos", los admiro muchísimo.

Gracias a mi padre por apoyarme durante mis estudios y enseñarme a ser compasivo.

Gracias a mis mejores amigos "Los Campeones" : Andrés Torres, Juan Francisco Yedra y Rodrigo Valdéz, cuya amistad me ha acompañado durante más de diez años; me han enseñado a crecer como persona y académicamente; son mis hermanos. Gracias también a

Lilián Irazú Hernández y Andrea Escobar que complementan la vida de mis amigos y con quien hemos compartido el viaje.

Gracias a mis compañeros de tesis, a Luis Quiroz por convocarnos como equipo para ingresar al PAT, así como por su disposición y compromiso para concluir el presente trabajo. Gracias Christian Sandoval por tu dedicación, diplomacia y franqueza en los retos enfrentados. Gracias Luis Fernando Romero por tu capacidad de investigación, atención al detalle y presentación de los resultados en esta tesis. Gracias amigos por su paciencia y esfuerzo. Llegamos al final después de tantos años.

Gracias a nuestra asesora de tesis M.I. Norma Elva Chávez por la motivación y entusiasmo por la conclusión de nuestros esfuerzos. Por último un agradecimiento enorme a la Universidad Nacional Autónoma de México y a la Facultad de Ingeniería, cuyo cuerpo académico e instalaciones me permiten contribuir al desarrollo de nuestro país y abren un camino para futuros trabajos. Por mi raza hablará el espíritu.

Agradecimientos de Luis Manuel Quiroz Ledesma

A mi esposa Olga. Nada de esto pudo haber sido posible sin ti, el apoyo, el amor incondicional, la paciencia y la perseverancia. Gracias por estar conmigo, gracias por ser parte de mi vida y ser la cómplice de este proyecto y de todos los que vienen. Te Amo mucho. Gracias por hacer de tu familia mi familia. Lo logramos esposa.

Agradezco a mi familia. A mi mamá Gloria por tu paciencia, bondad, todas las enseñanzas y amor incondicional. Eres la persona más buena del mundo. Te amo mucho. Lo logramos mami

A mi padre Javier por tus consejos, tu cariño y ese gran ejemplo que me has dado. Cada día intento parecerme un poco a más ti. Eres el mejor hombre de mi vida. Te amo. Lo logramos hijin

A mi hermana Tanya porque eres el mejor ejemplo de fuerza y tenacidad. Te amo mugres y gracias por estar siempre para mí, ser mi cómplice de vida y nunca dejarte caer. Te amo mucho. Lo logramos mugres.

A Angélica Nava eres el mejor ejemplo de perseverancia y dedicación sigue así, con esto te demuestro que si se puede, por más difícil que sea el camino siempre se puede.

A mi tíos Rubén y Miguel Quiroz por ser ese apoyo incondicional a mí y a mi familia. Son un gran ejemplo

A toda mi familia Quiroz Ledesma primos, primas sobrinos, sobrinas tías y tíos. (Disculpen no nombrarlos a todos, pero me llevo otra tesis).

A mis hermanos Jorge Velarde, Cesar Rosas y Alvaro Ledesma porque me enseñaron que existen hermanos sin ser de sangre. Saben que son parte importante muy de mi vida. Gracias

A mis mejores amigas Lucía Martínez y Ana Serrano por estar siempre al pendiente de mí y apoyarme siempre estar en las buenas y en las malas. Y permitirme ser testigo de sus logros. Las quiero feas.

A Mariana Gonzalez por ser mi doble y llegar en el momento justo, te quiero mucho. También a toda tu familia por esas muestras de cariño que siempre he recibido.

A la banda de la fac: Eros, Kenji, Viveros, Arturación, César, Neto, Mariano, Guillermo, Josué, Steph, Félix, Charro, La tremenda, Juanito, Fercho, Karlita, QuikeVS, Jaziel, Majo, y todos las personas que no menciono pero no dejan de ser importantes. Por molestarme a concluir esto, me hicieron pasar una excelente estancia en la facultad. Gracias amigos sin ustedes esto no hubiera sido posible.

A todos mis amigos que siempre han estado en las buenas, en las malas y en las peores: Cabe,el capitán, Carlos, tomate, core, pipo, beny, kyo, Karen, a toda la banda shaks (es imposible poner el nombre de todos mis amigos pero les agradezco a todos porque siempre he aprendido algo de ustedes.)

A mis amigos laborales Ricardito, Pamelita, Karlita, Nike, Rox, Noé, Martín, Leticia, Lalo, Adrián, Itzel, Rebe, frutos, Isra, Yuri, Eder. Porque me han apoyado, aconsejado," basuerado" para ser un profesional.

A mis compañeros de Tesis Nandito por ser un gran ser humano y excelente amigo (el primero y al final). Cris porque ser ese amigo tan positivo y entregado en cada proyecto. Juan por ser ese amigo tan interesante y comprometido. A la familia Bonilla Arroyo por abrirnos

las puertas de su casa y el cariño mostrado siempre. Son parte de este gran proyecto. Gracias amigos lo logramos (no que no!!!).
A mi jurado, a la UNAM mi alma mater y a la facultad de Ingeniería mi segunda casa.

Agradecimientos de Luis Romero Flores

El mayor agradecimiento a mi madre Ana Flores, por todo el cariño, confianza y consejo que me has dado siempre. Todo lo que bueno que imagino que una madre puede ser, lo eres tú.
Gracias a mi padre Fernando Romero, quien me ha ensañado todas las lecciones y sabiduría que me permitieron crecer y llegar a cerrar este ciclo. Tu perseverancia y ejemplo los tengo siempre en mente en cada uno de mis logros y decisiones.
A mi hermana Katy, gracias por todo el apoyo que me has dado. Admiro tu manera de reír y el humor con el que ves la vida.
Gracias a Emmanuel Romero, me inspiras con tu capacidad creativa.
Gracias a Isaac Romero, compartimos la actitud de nunca conformarnos. Espero que nuestra búsqueda por cosas nuevas y emocionantes nunca acabe.
Comparto este logro con mis primos Erwin, Jonathan, Iván, Jazmín, César Romero. Mis mejores amigos.
Gracias a Dafne por crecer a mi lado y enseñarme a pelar por nuestra felicidad, creo que nada nos puede detener.
Agradezco a mis compañeros de tesis: Quiroz por haberme acompañado todo el camino, Chris por el compromiso, apoyo y ambiente positivo que propiciaste durante el proyecto y a Juan Bonilla por tu pasión por las grandes ideas y el apoyo de tu familia, que en todas las visitas a tu casa me hizo sentir como en la mía.
Gracias a mis amigos de la facultad Viveros, Eros, Kenji, Arturo, César, Neto, Mariano, Guillermo, Steph, Mariana, Quike, Fercho y todos los que no menciono sin que sean menos importantes. ¡Fue genial compartir la universidad ustedes!
Por último agradezco a nuestra casa de estudios la UNAM y a la Facultad de Ingeniería, en cuyas aulas aprendí de grandes profesores y amigos.

Agradecimientos de Christian Sandoval

A mi mamá, por ser siempre mi gran ejemplo, mi guía, mi consejera. Te debo la vida y este trabajo también es tuyo, jamás podré regresarte todo lo que me has dado. ¡Gracias!
A mi Avis, pues sin tu perseverancia y tu fortaleza, no hubiera alcanzado este objetivo sin tu ayuda.
A Thalia y Diego, siempre están a mi lado en los momentos más divertidos y complicados. Mi vida no sería igual sin ustedes.
A mi papá, por quererme y apoyarme a tu manera.
A mis compañeros de proyecto. Gracias Quiroz, por tu iniciativa y por considerarme para este grupo. Gracias Nando, por tu compromiso para lograr el resultado, gracias Juan por permitir convertir tu casa en laboratorio y a tu familia por su hospitalidad durante nuestra estancia.
A todos mis amigos, porque nunca me dejan sólo.
A Bety, por ese “poquito” tiempo en el que nos hemos complementado y ayudado a ser mejores.
A la Universidad Nacional Autónoma de México, la Facultad de Ingeniería y todos mis profesores, por haberme brindado la oportunidad de formarme como ingeniero en sus aulas y compartirme su conocimiento.

ÍNDICE

1. Antecedentes

Una pantalla táctil es una superficie que permite la entrada de datos en una computadora mediante el toque directo en dicha superficie.Este contacto se realiza regularmente con los dedos pero también se puede realizar por medio de un lápiz óptico u otras herramientas similares. Las pantallas táctiles se hicieron populares por aplicaciones en dispositivos móviles, exposiciones de museos, cajeros automáticos, etc., donde el control conteclados o ratones no permitían una interacción intuitiva y rápida con los usuarios.

El primer dispositivo de interacción táctil se hizo con tecnología capacitiva para controlar el tráfico aéreo en el Royal Radar Establishment del Reino Unido, la tecnología fue descrita por E.A. Johnson en un artículo de 1965. El dispositivo permitía la entrada de sólo un punto de contacto. Su principio de operación sigue empleándose en cajeros ATM y dispensadores de boletos. La tecnología de pantalla táctil tuvo por primera vez la atención del público en 1971con la invención de la computadora “Elograph” por la compañía Elographics Inc. Esta empresa fue creada para producirdigitalizadores de datos empleados en investigación y aplicacionesindustriales. Un año después la Universidad de Illinois introdujo en su sistema académico la computadora PLATO IV que permitía a los estudiantes responder preguntas y consultar contenidos.

La capacidad de procesar varios puntos de contacto a la vez se propuso por Nimish Mehta de la Universidad de Toronto en 1982, una cámara de vídeo detectaba los puntos de contacto. La computadora “HP-150”, desarrollada por Hewlett Packard en 1983,es considerada la primera plataformacomercial con pantalla táctil, su principio de operación era la iluminación infrarroja. El mismo año Myron Krueger construyó un sistema óptico que detectaba manos de manera similar al sistema “Kinect” de Microsoft, se abrió así el camino

en la interacción gestual en la que combinaciones de movimientos específicos controlan operaciones gráficas como rotar, cambiar de página, amplificar, etc.

En 1993 IBM integró por primera vez la tecnología táctil a un teléfono celular en su dispositivo “Simon” con una pluma para digitalizar trazos. Académicos de la Universidad de Delaware fundaron FingerWorks en 1998 para comercializar teclados y tabletas táctiles. Siete años después Apple adquirió la compañía y tras varios productos popularizó la tecnología en dispositivos móviles.

Dentro de la tecnología de pantallas táctiles basada en luz infrarroja existen dos sistemas principales: uno basado en una rejilla interna y otro en el fenómeno de reflexión interna. Estos sistemas son muy precisos sin embargo requieren superficies más grandes. Los sistemas de reflexión interna son de grandes dimensiones para que las cámaras puedan medir de manera precisa la sombra producida por los LED´s infrarrojos.Las interfaces con tecnología de reflexión interna representan sistemas multi táctiles confiables y precisos que pueden ser fabricados a bajo costo y extenderse en aplicaciones de mayor escala.

2. Marco teórico

2.1 Refracción de ondas de luz

Cualquierondaelectromagnética se propagaen el vacío con una velocidad c=3 x 10^8 m/s. Dicha constante hacereferencia a cualquieronda monocromática o policromática queviaja en el vacío. Lavelocidad depropagación *v* en un medio diferente siempreesmenor ac de propagación en el vacío. Para una onda electromagnética que se propagapor un medio material, definimos el índice derefracción *n* de dicho medio para la onda en cuestión en larelación:

$$n = c/v$$

dondeces lavelocidad de laondaen el vacío y *v* la velocidad de la luz en el material. El índice derefracción es entonces una magnitud adimensional y caracteriza un medio material desdeel punto devista óptico. La figura 2.1.1. presenta el índice derefracción dealgunas sutancias.

Material	n_C'' 6563 Å (rojo)	n_D'' 5890 Å (amarillo)	n_F'' 4861 Å (azul)	n_G'' 4340 Å (violeta)
Agua, 20 ° C	1.3312	1.3330	1.3372	1.3404
Alcohol etílico, 20 ° C	1.3605	1.3618	1.3666	1.3700
Disulfuro de carbono, 20 ° C	1.6182	1.6276	1.6523	1.6748
Vidrio crown, n.° 123	1.51458	1.51714	1.52326	1.52859
Vidrio flint, ligero, n.° 188	1.57638	1.58038	1.59029	1.59931
Vidrio flint, denso, n.° 76	1.65007	1.65548	1.66911	1.68181

FIGURA 2.1.1 TABLA DE ÍNDICES DEREFRACCIÓN DEALGUNASSUSTANCIAS

Los materiales o medios se pueden clasificar en función desu comportamiento ante laluz en tres grupos principales:

- Transparentes. Dejan pasar parte de laluz a través deellos sin dificultad y absorben una parte como efecto de la polarización electrónica introducida en intervalos de impurezas.
- Traslúcido. Dejan pasaruna parte de la luz, transmiten unaimagen difusa y en su interior se produce una mayorcantidaddeefectos dereflexión y refracción.
- Opacos. Absorben todo el rango deluz visible.

A los medios cuyascaracterísticas físicas permiten a laonda atravesar de manerahomogénea en todos los puntosy en todas direcciones, se considera un medio homogéneo e isótropo. Un medio que permite a la luz viajar con la mismavelocidad pero que produce variaciones con respecto a ladirección sedenominan anisótropo. Si por el contrario,la velocidades la quevaría dependiendo de la dirección, se le consideraun medio heterogéneo.

El índice derefracción también puedeserdefinido en relación con lapermitividad eléctrica y la permeabilidadmagnética relativas del medio del material. Para la mayoriade lassustancias transparentes $\mu_r = 1$, deesta manera:

$$n = \sqrt{E_r}$$

Donde *Er*es la constante dieléctrica del material o medio. Por estarazón, en los camposeléctricos rápidamentevariables con el tiempo, el valor de*Er*depende de la frecuencia de las oscilaciones en el campo.

2.2 Reflexión de onda de luz

El principio derelexión fueexplicado por Pierre Fermat en 1650 utilizando el siguiente enunciado:

“Un rayo luminoso que pasa deun punto fijo aotro seguiráuna trayectoriatal que, en comparación con lastrayectoriascercanas, el tiempo requerido es mínimo, máximo o permanece inalterado”.

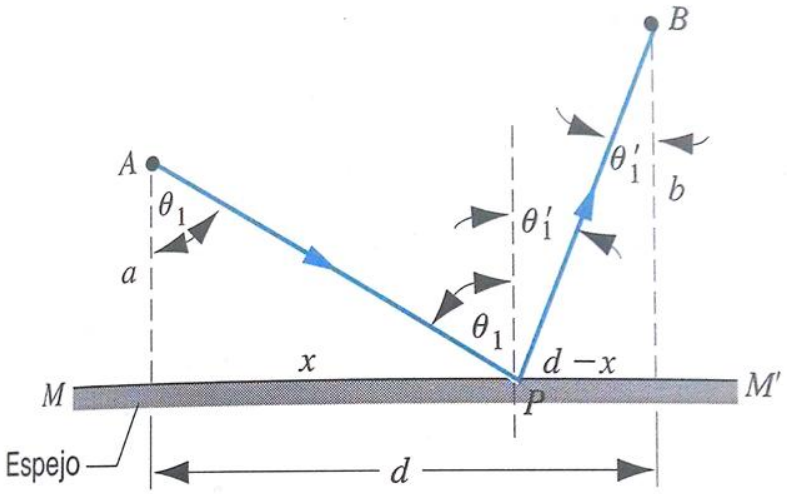


FIGURA 2.2.1 REFLEXIÓN DE UNA ONDA PLANA CONTRA UN ESPEJO PLANO

La figura 2.2.1presenta el análisis geométrico del principio de Fermat. Se pueden hacer deduccioneshasta comprobar que la ley dereflexión puede escribirse como:

$$sen\theta_1 = sen\theta_1'$$

El comportamiento del haz reflejado depende de lasuperficie incidente, entoncesse pueden obtenerdistintosefectos dereflexión. Cuando la superficie eslisa, todos losrayos son reflejados en una misma direcciόny cuando lasuperfice esrugosase obtiene una reflexión difusa, con rayoscambiando dedirección dependiendo del punto de incidencia.(Ver figura 2.2.2)

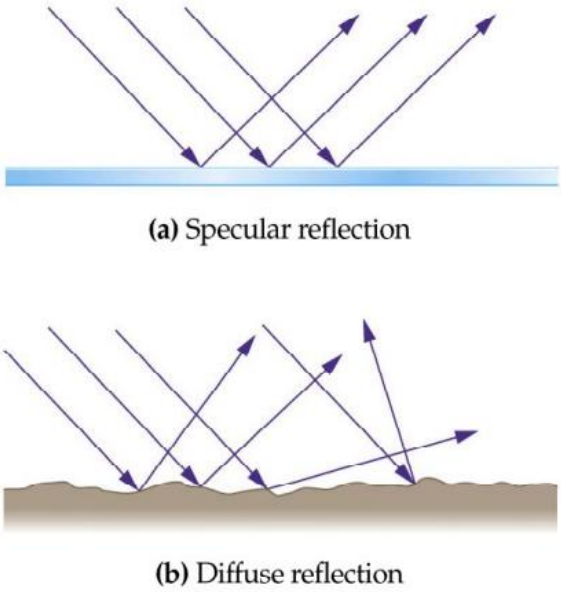


FIGURA 2.2.2. EFECTO DE REFLEXIÓN DE LA LUZ EN DIFERENTES SUPERFICIES

2.3 Reflexión interna total

Cuando se tiene unafuente puntual de luz incidiendo sobreuna interfaz vidrio-aire y el ángulo de incidencia θ aumenta, se llega a la situación de la figura 2.3.1, donde el rayo refractado apunta a lo largo de lafrontera entre los medios, con un ángulo de refracción de 90°. En los ángulos de incidencia más grandes que este ángulo críticoθ_c, no existen rayos refractados, y se puede hablar entonces deuna reflexión interna total.

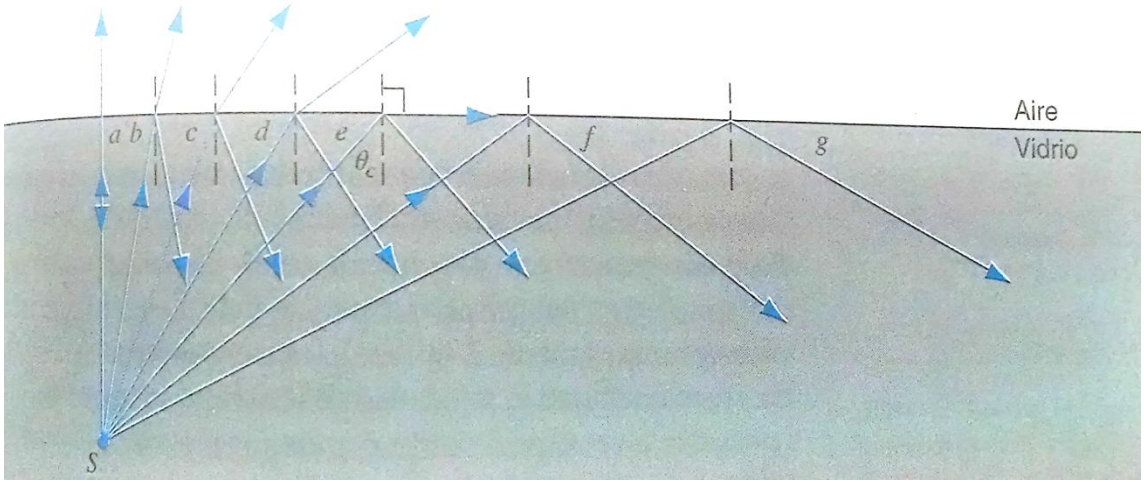


FIGURA 2.3.1EFECTO DE REFLEXIÓN DE LA LUZ EN UN MEDIO VIDRIO-AIRE

Elángulo crítico se puedecalcular con lasiguiente ecuación

$$\theta_c = csc\frac{n_2}{n_1}$$

Paraun vidrio $\theta_c = sen^{-1}(\frac{1.00}{1.50})$ = 41.8°. Por lo que el total de laenergía de la onda es reflejada cuando el ángulo deincidencia es mayora 41.8°para este par de materiales.Ya que el seno de un ángulo no puede ser mayor a la unidad, se debe tenerque n_2<n_1. El fenómeno de reflexión interna total ha permitido diseñar dispositivos de fibra óptica con una pérdida de energía prácticamentenula.Lafigura 2.3.1ilustra como la reflexión interna total de la luz desde una fuentepuntual S ocurre en todos los ángulos de incidencia mayores que el éngulo crítico θ_c.

2.4 Reflexión interna total frustrada (FTIR)

El efecto de FTIR fue identificado por primera vez por Selenyi en 1913 al recubrir la cara superior de un prisma semiesférico con un material fluorescente eiluminarlo con una fuente de gran ángulo como se muestra en la figura 2.4.2. Mediantela detección de la intensidad que escapa del prisma para ángulos mayores al crítico, se notóque una onda evanescente debe estar presente en la película fluorescente para generar el efecto de frustración y

propagarse fuera de la lente interfaz. El efecto FTIR es utilizado en aplicaciones como los lentes de inmersión sólida, almacenamiento óptico e inmersión sólida microscópica.En la figura 2.4.3 se muestra una pequeñabrecha, comúnmente aire, que es controlada entre el último elemento semiesférico óptico y un sustrato. En distancias de campo cercano, el campo evanescente en la brecha resultante de la iluminación frustrada se propaga por el sustrato con el índice de reflexión más alto.

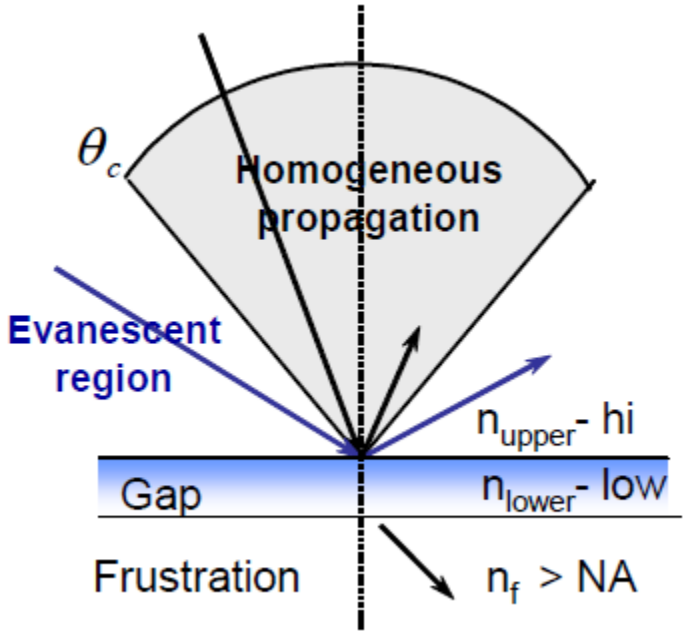


FIGURA 2.4.1 REFLEXIÓN INTERNA TOTAL FRUSTRADA DE UN CAMPO EVANESCENTE DENTRO DE UN MEDIO CON UN ÍNDICE MAYOR QUE LA APERTURA NUMÉRICA DE LA PROPAGACIÓN DE FRENTE DE ONDA

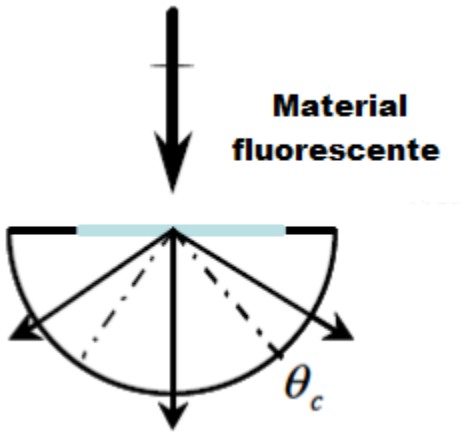


FIGURA 2.4.2 EXPERIMENTO DE SELENYI QUE DETECTA LA ENERGÍA MEDIANTE LA PROPAGACIÓN DE UN CAMPO EVANESCENTE DENTRO DE UN MATERIAL FLUORESCENTE

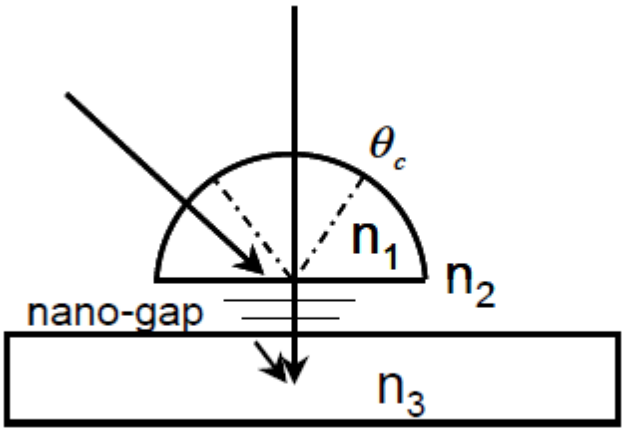


FIGURA 2.4.3 ESQUEMA DE UNA LENTE DE INMERSIÓN SÓLIDA CON UNA BRECHA DE AIRE EN UNA REGIÓN DE CAMPO CERCANO

2.5 Luz Infrarroja

La luz infrarroja se encuentra en una porción del espectro electromagnético intermedia entre la parte visible y las microondas. La luz "infrarroja cercana" es la más próxima en cuanto a longitud de onda a la luz visible y la luz "infrarroja lejana" está más cerca de la región delas microondas del espectro electromagnético. Las longitudes de onda más lejanas de la luz

visible son aproximadamente del tamaño de una cabeza de alfiler y las más cortas son del tamaño de las células.

La región de frecuencias en la que se encuentra la luz infrarroja, como se muestra en la figura 2.5.1, va de los 0.7 a los 300 [µm] y se divide en las siguientes bandas:

- Infrarrojo cercano: 0.7 a 1.5 [µm]
- Infrarrojode longitud de onda corta: 1.5 a 3 [µm]
- Infrarrojo de longitud de onda media: 3 a 8 [µm]
- Infrarrojo de longitud de onda larga: 8 a 15 [µm]
- Infrarrojo lejano: Mayor a 15 [µm]

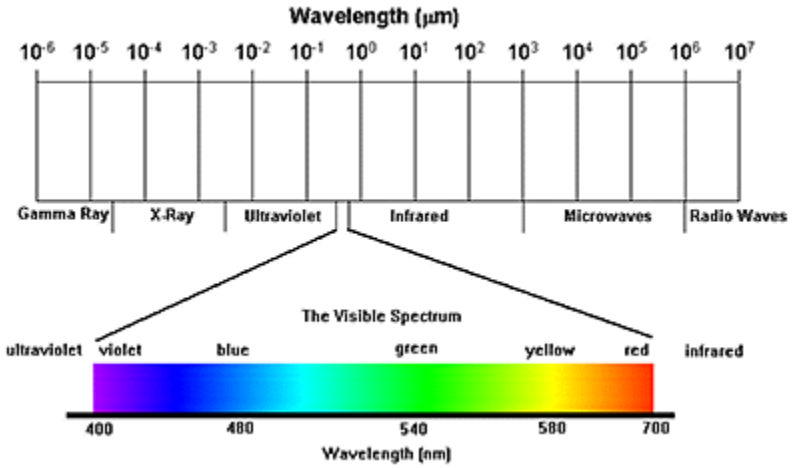


FIGURA 2.5.1 ESPECTRO ELECTROMAGNÉTICO

2.6 Interfaz Gráfica de Usuario(Graphical User Interface)

A diferencia de un interfaz mediante comandos (CLI) una GUI permite interacción gráfica con la computadora. Se originó para mejorar la utilidad en las primeras interfaces basadas en texto queconsistían en órdenes y comandos para ejecutar tareas predeterminadas. Posteriormente se mejoró la combinación con un sistema de menús e interacción con un cursor.

Existen muchas herramientas orientadas a objetos que facilitan la escritura de una interfaz gráfica de usuario. Cada elemento de la GUI se define como una clase en la cual se crean instancias de objetos para su aplicación. Se codifican métodos que un objeto utiliza para responder alestímulo de los usuarios.

2.7 Visión por computadora (Computer Vision)

El concepto de visión por computadora (CV) se basa en el análisis automático de imágenes y vídeos por computadora con el fin de procesar la información y manipularla. Una imagen se convierte en una matriz de escala de grises que facilita la detección de patrones. Elementos gráficos como contraste y posición de los píxeles son la base de los algoritmos para el reconocimiento de objetos predeterminados. La figura 2.7.1 muestratres versiones diferentes de la misma imagen convertida en arreglos.

67	67	66	68	66	67	64	65	65	63	63	69	61	64	63	66	61	60
69	68	63	68	65	62	65	61	50	26	32	65	61	67	64	65	66	63
72	71	70	87	67	60	28	21	17	18	13	15	20	59	61	65	66	64
75	73	76	78	67	26	20	19	16	18	16	13	18	21	50	61	69	70
74	75	78	74	39	31	31	30	46	37	69	66	64	43	18	63	69	60
73	75	77	64	41	20	18	22	63	92	99	88	78	73	39	40	59	65
74	75	71	42	19	12	14	28	79	102	107	96	87	79	57	29	68	66
75	75	66	43	12	11	16	62	87	84	84	108	83	84	59	39	70	66
76	74	49	42	37	10	34	78	90	99	68	94	97	51	40	69	72	65
76	63	40	57	123	88	60	83	95	88	80	71	67	69	32	67	73	73
78	50	32	33	90	121	66	86	100	116	87	85	80	74	71	56	58	48
80	40	33	16	63	107	57	86	103	113	113	104	94	86	77	48	47	45
88	41	35	10	15	94	67	96	98	91	86	105	81	77	71	35	45	47
87	51	35	15	15	17	51	92	104	101	72	74	87	100	27	31	44	46
86	42	47	11	13	16	71	76	89	95	116	91	67	87	12	25	43	51
96	67	20	12	17	17	86	89	90	101	96	89	62	13	11	19	40	51
99	88	19	15	15	18	32	107	99	86	95	92	26	13	13	16	49	52
99	77	16	14	14	16	35	115	111	109	91	79	17	16	13	46	48	51

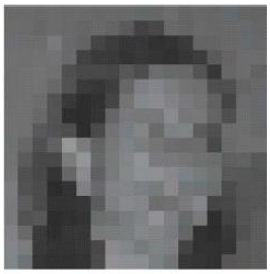


FIGURA 2.7.1: DIFERENTES VERSIONES DE UNA IMAGEN. UNA MATRIZ DE NÚMEROS (IZQUIERDA), QUE CONTIENE LOS VALORES DE LA ESCALA DE GRISES DE LA IMAGEN DE BAJA RESOLUCIÓN DE UNA CARA (PARTE SUPERIOR DERECHA)

Los patrones de detección operan enla gama de valores que se muestran en la figura 2.7.1, para homologar la visión humana se necesitaría una matriz mucho más grande (alrededor de 127 millones de elementos) y más compleja ya que cada punto debería representar tres valores con el fin de codificar la información de color. Adicionalmente el procesamiento debe hacerse en varios cuadros por segundo del sistema de adquisición de imágenes para tener una visión aproximada al tiempo real.

2.8 Protocolo de capa de transporte UDP

El protocolo UDP (User Datagram Protocol) opera en lacapa de transportedel modelo OSI. Fue definido por John Postel bajo el estándar de internet RFC 768 y permite tener un modo empaquetado de datagramas disponibles para la conexión de computadoras en un conjunto de redes interconectadas. Se considera no orientado a conexión.

2.9 Protocolo de Control de Transmisión TCP

El protocolo TCP (Protocolo de Control de Transmisión) opera en la capa de transporte del modelo OSI, es orientado a conexión por lo que se emplea en aplicaciones que requieran de una entrega confiable y ordenada de flujos de datos. Está especificado en RFC 793.

2.10 Interfaz digital para instrumentos musicales (MIDI)

MIDI es el acrónimo de Interface Digital para Instrumentos Musicales, fue definido dentro de la especificación MIDI 1.0 en agosto de 1982.Este lenguaje describe el proceso de la reproducción de la música de manera similar a las partituras, hay mensajes MIDI que indican las notas a reproducir, por cuánto tiempo y a qué niveles de volumen.Los mensajes pueden intercambiarse entre la computadora, instrumentos compatibles e interfaces.

2.11 Plataforma de desarrollo Reactivision

La plataforma Reactivision es sistema de visión por computadora para el rastreo rápido y robusto de los marcadores *fiducials*adjuntos a objetos físicos, así como para el seguimiento multi táctil. Fue diseñado en código abierto principalmente como un conjunto de herramientas para el desarrollo rápido de Interfaces Tangibles de Usuario (TUI) y superficies interactivas.

El protocolo de comunicación TUIO fue desarrollado dentro del proyecto Reactivision para codificar el estado de los objetos tangibles y eventos multi-táctiles de una mesa interactiva. Dichos mensajes son enviados como servidor a través del puerto 3333 (UDP) para un cliente configurado con las bibliotecas apropiadas en el lenguaje de programación destino. En el presente trabajo se configuró un cliente en modo “TUIOlistener” en Processing y Pure Data.

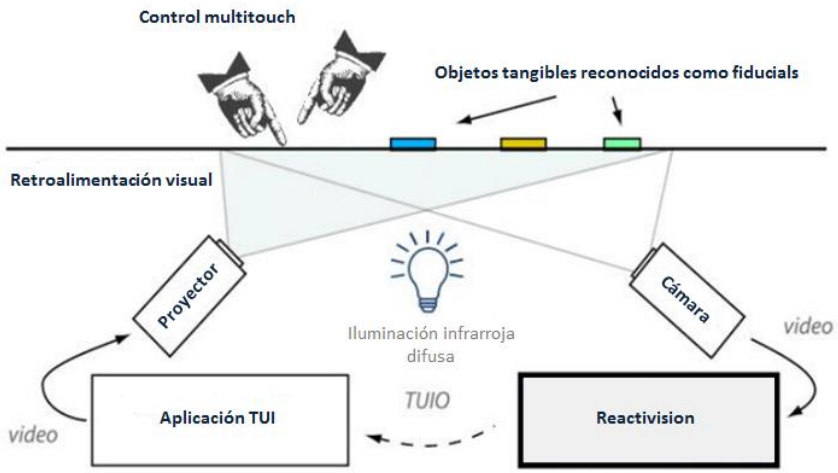


FIGURA 2.11.1 MARCO DISTRIBUIDO

El procesamiento de los marcadores comienza con una secuencia de vídeo en tiempo real que se convierte a una imagen en blanco y negro mediante un algoritmo de umbral adaptativo, de esta forma se identifican y separanlos objetos del fondo con base en la distribución de los niveles de gris. A continuación, esta imagen es segmentada en un árbol

regiones que alternan entre blanco y negro. Dentro de este gráfico se buscan secuencias de árboles (patrones únicos) que han sido codificados en un símbolo *fiducial*. Finalmente las secuencias de árboles encontradas se igualan contra un diccionario de datos para recuperar un número de identificación. El diseño del *fiducial* permite calcular de manera eficiente el punto central del marcador, así como su orientación. Los mensajes de Open Sound Control (OSC), que utiliza el protocolo TUIO, codifican la presencia, ubicación, orientación e identidad de los *fiducials* y transmiten esta información a las aplicaciones cliente.

Adicionalmente se utiliza el resultado de la segmentación de imágenes con el fin de recuperar e identificar pequeñas manchas blancas redondas como las puntas de los dedos sobre la superficie. Un algoritmo rápido de coincidencias permite seleccionar las manchas de los dedos de los posibles candidatos de la región.

Figura 2.11.2 Reconocimiento de dedos en Reactivision

2.11.1 Símbolos Fiduciales

El motor de seguimiento por defecto de fiducials utiliza la biblioteca fidtrack de Ross Bencina que es una aplicación de alto rendimiento del concepto de *d-touch* de Enrico Costanza.

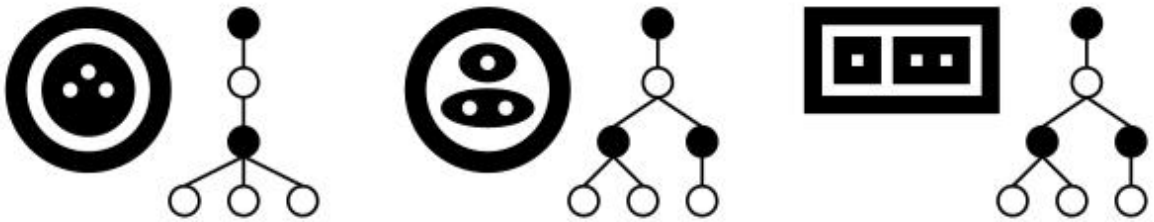


FIGURA 2.11.1.1 ALGUNAS SIMPLES TIPOLOGÍAS Y SUS CORRESPONDIENTES GRÁFICAS DE REGIÓN ADYACENTES

2.11.2 Seguimiento de dedos

El seguimiento de dedos aprovecha la infraestructura de procesamiento de imagen existente con poca sobrecarga de rendimiento adicional. Adicionalmente, los controles generales de la imagen como el brillo, la ganancia y velocidad de obturación pueden mejorar la calidad de seguimiento (tecla O). Otro parámetro importante de control es el umbral de ruido, se debe establecer tan bajo como sea posible pero que permita que la imagen sea identificable (tecla G). Finalmente el seguimiento de las puntas de los dedos puede ser configurado ajustando del tamaño medio de los dedos y la sensibilidad de seguimiento (tecla F).

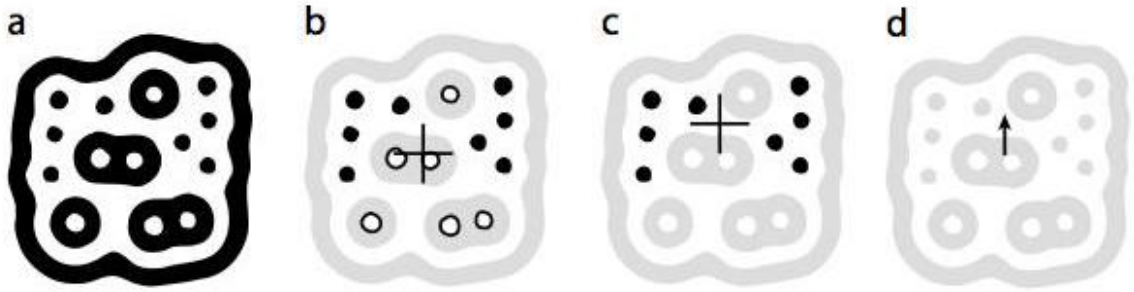


FIGURA 2.12.2.3. (A) UN FIDUCIAL DE REACTIVISION (B) HOJAS EN BLANCO Y NEGRO Y SU CENTROIDE PROMEDIO (C) HOJAS NEGRAS Y SU CENTROIDE PROMEDIO (D) VECTOR UTILIZADO PARA CALCULAR LA ORIENTACIÓN DEL FIDUCIAL

2.11.3 Funcionalidad de mensajes TUIO vs. MIDI

La aplicación puede enviar alternativamente mensajes MIDI que permiten mapear cualquier dimensión del objeto (x, y, ángulo) a un control MIDI a través de un archivo de configuración XML. La adición y eliminación de objetos se puede asignar a eventos de encendido y apagado en una nota. Los eventos MIDI tienen menos ancho de banda y resolución de datos en comparación con los eventos de Open Sound Control (OSC), por lo que la función MIDI está pensada como una alternativa conveniente en algunos casos. Agregando el comando <midi config="midi/demo.xml"/>al archivo Reactivision.xml, cambia al modo MIDI y especifica el archivo de configuración MIDI que contiene las asignaciones y la selección del dispositivo MIDI.

2.12 Protocolode comunicación Open Sound Control (OSC)

Open Sound Control (OSC) es un protocolo de comunicación para computadoras, sintetizadores de sonido y otros dispositivos multimedia. Está optimizado para aprovechar los beneficios de la tecnologíade conexiones de redes en el mundo de los instrumentos musicales electrónicos. Las ventajas de OSC incluyen precisión, flexibilidad así como una mejor organización y documentación.

Existen múltiples implementaciones de OSC, incluyendo entornos de procesamiento de sonido en tiempo realy de medios de comunicación, herramientas de interactividad web, sintetizadores de software, etc. El protocolo OSC ha alcanzado un amplio uso en campos que incluyen nuevas interfaces para la expresión musical y sistemas de música distribuidos conectados localmente.Entre lasprincipales características de OSC se encuentran las siguientes:

- Esquema de nombramiento de URL abierto y dinámico
- Argumentos numéricos y simbólicos para mensajes
- Etiquetas de tiempo
- Paquetes de mensajes cuyos efectos deben ocurrir simultáneamente

2.12.1 Formato de mensajes OSC

El formato de mensajes utilizados en OSC se escribe de la siguiente manera:

<Dirección del patrón> + <cadena de etiqueta> + <argumentos>
e.g.:
127.0.0.1 ”destino” 11

2.12.2 Transporte de mensajes OSC

Por lo general el control OSC se transmite con paquetes UDP en redes locales o públicas,aunque es posible usar TCP que es orientado a conexión.
Tipos de datos permitidos en OSC:

- 32bit integer
- 32bit float
- OSC-string
- OSC-timetag
- OSCblob

Perfiles de objetos en OSC:

- Objeto 2D: sessionID, classID, x,y,angle,xSpeed,ySpeed,rotSpeed, motionAccel, rotAccel
- Cursor 2D: sessionID, x ,y , xSpeed, ySpeed, motionAccel
- Mancha 2D: sessionID, x ,y , angle, width, height, area, xSpeed,ySpeed, rotASpeed
- Perfiles alternativos: 2.5Dobj, 2.5Dcur, 2.5Dblb, 3Dobj, 3Dcur. 3Dblb

Mensajes OSC:
set: Los parámetros actuales de cualquier objeto modificado
alive: Una lista de todos los objetos visibles al momento
fseq: El orden de la secuencia de imágenes

Ejemplos:
/tuio/2Dobj alive 7 10 14 32
/tuio/2Dobj set 10 10 0.35 0.75 3.14 0.1 0.2 0.05 -0.3 0.2
/tuio/2Dobj fseq 132

2.13 Protocolo TUIO

El protocolo TUIO encapsula el formato OSC que proporciona un método eficiente de codificación binaria para la transmisión de datos de un controlador, por lo tanto los mensajes TUIO pueden transmitirse a través de cualquier canal soportado en una implementación de OSC. El método de transporte por defecto es UDP. Este método de transporte predeterminado es conocido como TUIO / UDP.

2.13.1 Formato de paquetes y mensajes TUIO

Debido a que TUIO está basado en OSC sigue su sintaxis general. Por lo tanto, una aplicación cliente TUIO hace uso de bibliotecas apropiadas de OSC.

En el siguiente ejemplo serecibe informaciónde un cursor bi-dimensional:
/tuio/2Dcur source application@address
/tuio/2Dcur alive s_id0 ... s_idN
/tuio/2Dcur set s_id x_pos y_pos x_vel y_vel m_accel

Un paquete típico TUIO contendrá un mensaje *ALIVE*inicial, seguido de un número arbitrario de mensajes *SET* que pueden encapsularse en la capacidad del paquete actual y un mensaje *FSEQ* final.

2.13.2 ComponentesTUIO

Los componentes de TUIO son los siguientes:

- Tokens: /tuio/2Dobj: Describe los objetos físicos arbitrarios, por lo general se realiza un seguimiento con la ayuda de símbolos visuales (marcadores fiducials), etiquetas RFID o métodos similares. Los tokens no se definen por su apariencia física, sino por un ID que codifica la posición y la rotación de su ángulo
- Punteros: / tuio / 2Dcur: Describen punteros de superficie tales como toques de los dedos o múltiples punteros de dispositivos dedicados que sólo se distinguen por su posición
- Geometrías: / tuio / 2Dblb: Describen los límites de los objetos físicos sin etiquetar y codifican la posición y la caja de contorno orientado (ángulo, ancho, alto) que puede ser utilizada para describir adicionalmente la geometría aproximada de los tokens y punteros.

2.14 Programación Orientada a Objetos

La programación orientada a objetos surge como un intento para solucionar la complejidad que posee la programación de software. De manera tradicional la metodología de resolución de un problema con computadora es:

- Análisis de problema
- Diseño de algoritmo
- Codificación
- Verificación
- Depuración
- Mantenimiento
- Documentación

En la programación orientada a objetos se descompone el problema en sub-problemas más sencillos que con acciones simples se pueden resolver de manera más eficiente. Como consecuencia se tiene un problema más fácil de programar, reduciendo el tiempo que toma el diseñar un algoritmo que abarque todas las partes del problema.

Actualmente existen lenguajes orientados a objetos como C++ y Java. En ambos, el elemento básico no es la función, sino un “ente” denominado objeto.
Un objeto es la representación de un concepto para un programa y contiene toda la información necesaria para abstraerlo: datos que describen sus atributos y operaciones que pueden realizarse sobre los mismos.
De tal manera, la programación orientada a objetos es un modelo de desarrollo con teoría y metodología basada en el uso de objetos y sus interacciones para generar aplicaciones y programas informáticos.

2.14.1 Definición y características de Objetos

Un objeto no es más que un conjunto de variables (o datos) y métodos (o funciones) relacionados entre sí. Los objetos en programación se usan para modelar entidades del mundo real. Un objeto es por lo tanto, la representación en un programa de un concepto y contiene toda la información necesaria para abstraerlo, los datos que describen sus atributos y operaciones que pueden realizarse sobre los mismos.

Los atributos del objeto (estado) y lo que el objeto puede hacer (comportamiento) están expresados por las variables y métodos que componen el objeto respectivamente.

Un objeto puede tener varios métodos. A los métodos que cambian la función se les denomina métodos instancia o métodos miembro, ya que cambian el estado de una instancia u objeto particular. En instancias el programador necesita exportar alguna de las variables miembro o en caso contrario protegerla.

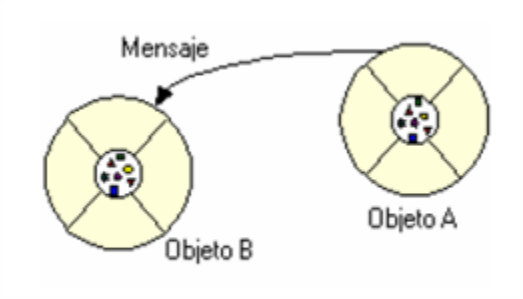
Existen distintos niveles de protección los cuales determinan que objetos y clases pueden acceder a los métodos. El hecho de poder encapsular las variables proporciona 2 importantes beneficios a los programadores de aplicaciones:

- Capacidad de crear módulos. El código fuente de un objeto puede escribirse y mantenerse independiente del código fuente del resto de los objetos
- Protección de información. Un objeto tendrá una interfaz pública perfectamente definida que otros objetos podrán usar para comunicarse con él. De esta forma, los objetos pueden mantener información privada y pueden cambiar el modo de operar de sus funciones miembros sin que esto afecte a otros objetos que usen estas funciones miembro.

2.14.2 Interacción de objetos

La interacción entre los objetos de un programa ocurre por medio de mensajes. Cuando un objeto A quiere que otro objeto B ejecute una de sus funciones miembro (métodos de B), el objeto A manda un mensaje al objeto B. El contenido del mensaje debe incluir los parámetros con la información precisa para poderse ejecutar correctamente.

El comportamiento de un objeto está completamente determinado (a excepción del acceso directo a variables miembro públicas) por sus métodos, así que los mensajes representan todas las posibles interacciones que pueden realizarse entre objetos.



Cuando un objeto A quiere que otro objeto B ejecute una de sus funciones miembro (métodos de B), el objeto A manda un mensaje al objeto B.

FIGURA 2.15.2.1 MÉTODOS Y VARIABLES

Otra propiedad de la interacción en programación orientada a objetos es que los objetos no necesitan formar parte del mismo proceso, ni siquiera residir en una misma computadora para mandarse mensajes entre sí.

2.14.3 Propiedades Clase

Existen plantillas para crear objetos denominados clases, estas definen las variables y los métodos que son comunes para todos los objetos de un cierto tipo.

La propiedad herencia permite definir nuevas clases partiendo de otras ya existentes. Las clases que derivan de otras heredan automáticamente todo su comportamiento, sin embargo, es posible introducir características particulares.
Esta propiedad también permite definir clases a partir de otras clases ya construidas. Por cada subclase hereda, los estados en forma de declaración de variables, de la superclase de la cual deriva.

Las subclases pueden añadir variables y métodos a aquellas que han heredado. Estamos limitados a un único nivel de herencia. El árbol de herencias o jerarquía de clases puede ser tan extenso como necesitemos. Los métodos y las variables miembro se heredarán hacia abajo a través de todos los niveles de la jerarquía.
La especialización de las clases se encuentra en la base de la jerarquía. Esto ayuda a la resolución de un problema, ya que la herencia es una herramienta clave para abordar un problema de forma organizada, pues permite definir una relación jerárquica entre todos los conceptos que se están manejando.
La herencia proporciona las siguientes ventajas:

- Las clases derivadas o subclases proporcionan comportamientos especializados a partir de los elementos comunes que hereda de la clase base.
- A través del mecanismo de herencia los programadores pueden reutilizar el código de la superclase tantas veces como sea necesario.
- Los programadores pueden implementar las llamadas superclases abstractas, que definen comportamientos genéricos. Las clases abstractas definen e implementan parcialmente comportamientos.

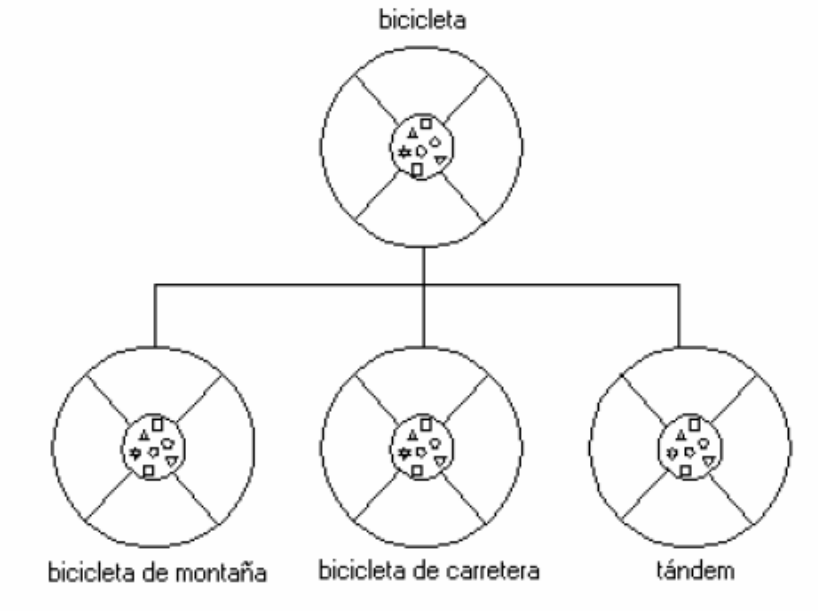


FIGURA 2.15.3.1 MÉTODOS Y VARIABLES

La figura 2.15.3.1 Ejemplifica la herencia de propiedades de la de clase bicicleta a las subclases; bicicleta de montaña, bicicleta de carretera y tándem. El propósito de una clase abstracta es servir de modelo base para la creación de otras clases derivadas, pero cuya implantación depende de las características particulares de cada una de ellas. Un ejemplo de clase abstracta podría ser en nuestro caso la clase vehículos. Esta clase sería una clase base genérica, a partir de la cual podríamos ir creando todo tipo de clases derivadas.

2.14.4 Palabras clave o reservadas

Las palabras clave o reservadas son aquellas que definen o se combinan con una definición de datos de una variable o propiedad, clase de objetos o métodos, por ejemplo *class* es una palabra clave y nos indica que a continuación pondremos el nombre a la clase de objeto que vamos a crear, si queremos que dicha clase pueda ser accedida desde dentro o fuera de nuestro paquete de clases de objetos, entonces la definiremos como pública (public), donde esta es otra palabra clave.

2.15 Lenguaje de programaciónPure Data

Pd (Pure Data) es un ambiente de programación gráfica en tiempo real para audio, vídeo y procesamiento de gráficos. Fue desarrollado por Miller Puckette a partir de MAX/MSP y se ha extendido de manera comunitaria debido a la facilidad de crear y compartir bibliotecas entre los usuarios.

Adiferencia de otros lenguajes con las mismas capacidades de procesamiento no es necesarioescribir líneas de código, se trabaja a partir de objetos gráficos con forma de pequeñas cajas que se conectan entre sí a través de canales virtuales.

Otra diferencia dePd con los lenguajes como C, C++ o Java es que la programación está orientada al flujo de datos, esto permite trabajar fácilmente con procesos en paralelo y el direccionamiento de datos determina la conexión entre los objetos a través de los canales virtuales.

2.15.1 Ambiente de programación.

Al abrir el programa aparece la ventana principal de Pdque es donde se imprimen los mensajes enviados desde el entorno, las librerías que fueron cargadas así como los errores I/O.

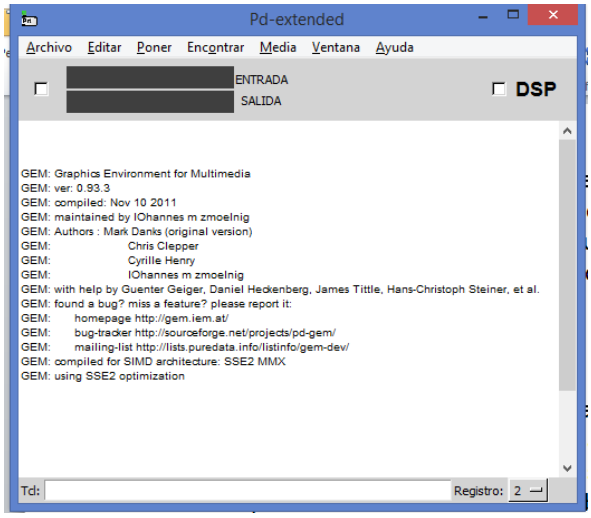


FIGURA 2.14.1.1 PD - VENTANA PRINCIPAL

Las unidades donde se programa el código se llaman parches (patches). Los patches constan de diferentes objetos interconectados entre ellos. Por convención en su parte superior se encuentranlas entradas donde se les envían valores numéricos u otros tipos de datos y en la parte inferior se ubican las salidas de datos.

El programa tiene dos estados en los que se puede encontrar el usuario, el modo de edición o el modo de ejecución. Para cambiar de un estado a otro se teclea Ctrl+E.

En el modo edición se puede modificar el contenido de las cajas o la conexión entre ellas. En el modo de ejecución se tiene la posibilidad de poner en marcha todo el patch e ir modificando los valores durante su reproducción o cuando esté detenido.

Se pueden crear patchs secundarios conocidos como sub parches (subpatchs). Estos se encuentran dentro del patch principal, se crean escribiendo en un objeto las letras “Pd” seguidas de un espacio y el nombre que se le quiera dar a ese subpatch.

La forma de comunicación entre los tipos de cajas es en forma de cables. Los cables gruesos transmiten señales, mientras que los cables delgados sólo transmiten datos.

2.15.2 Tipos de objetos

- **Objetos.**El comportamiento de los objetos depende del texto introducido en él. Pd tiene unos objetos predefinidos, programados por terceras personas en diferentes lenguajes como C. El programa reconoce el tipo de objeto y esa caja se comportacomo fue definida internamente. Ver figura 2.14.2.1

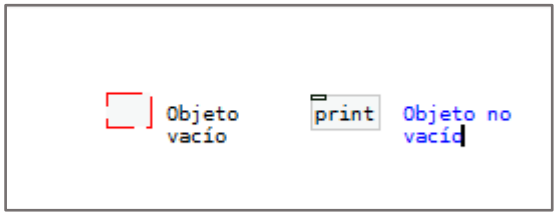


FIGURA 2.14.2.1 OBJETO

- **Números**. Su utilidad es diversa, desde controlar el valor que tiene la señal en diferentes puntos del patch hasta la de inicializar valores que se pasan a objetos que pueden controlar acciones. Su forma de caja tiene una hendidura en la esquina superior derecha. Verfigura 2.14.2

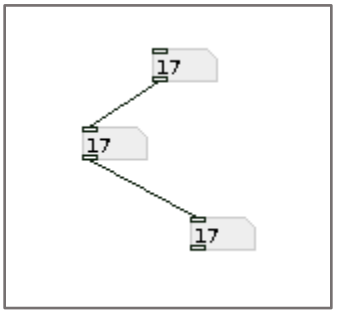


FIGURA 2.14.2.2 NÚMERO

- **Mensajes**. Están provistos de información que pasan a los objetos, su forma es una caja rectangular con esquinas salientes del lado derecho. Ver figura 2.14.2.3

FIGURA 2.14.2.3 MENSAJE

- **Bang**. Es la segunda forma de datos que Pd utiliza adicionalmente a los números. Algunos objetos reconocen ciertas palabras y trabajan estas como entradas. Dado que el bang se utiliza frecuentemente, existe una representación gráfica especial que utiliza un círculo que se oscurece cuando se activa. En la entrada derecha se introduceel número que rige la periodicidad del envío de bangs, en milisegundos. Ver figura 2.14.2.4

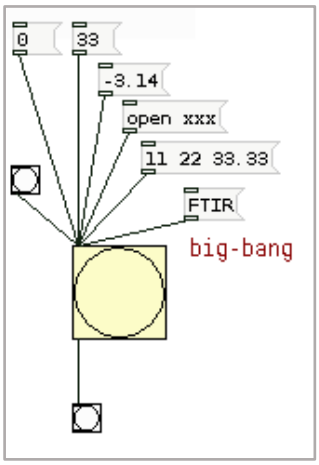


FIGURA 2.14.2.4 BANG

- **Símbolo.** Este objeto guarda un símbolo hasta que recibe un “bang” u otro símbolo. El símbolo accionado sale del objeto, por la parte inferior de la caja. Ver figura. 2.14.2.5

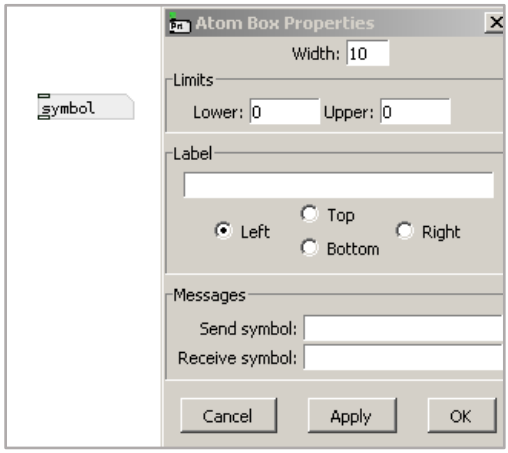


FIGURA 2.14.2.5 SÍMBOLO

- **Comentario**. Se utiliza para incluir aclaraciones dentro de los diferentes pasos que sigue el código del programa. Pueden colocarse en cualquier espacio del margen de programación. Ver figura 2.14.2.6

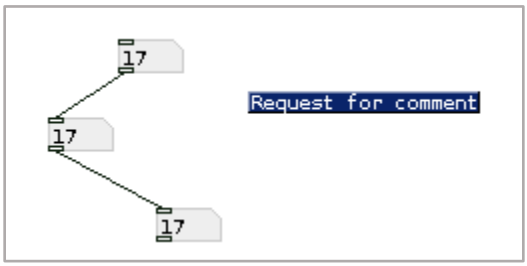


FIGURA 2.14.2.6 COMENTARIO

- **Metro**. Envía series de bangs periódicamente. Se genera escribiendo la palabra metro dentro de un objeto.La entrada izquierda es un interruptor que al recibir un número distinto de cero o un bang comienza.Acepta mensajes de texto, stops, start, etc. La entrada de la derecha es el delta de tiempo en [ms] entre cada bang que envía. Ver fig 2.14.2.7

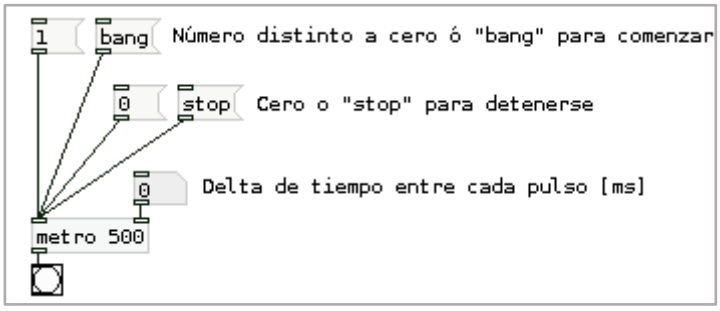


FIGURA 2.14.2.7 METRO

- **Expr**. Se utiliza para evaluar expresiones de datos de control donde se pueden definir múltiples variables para operarlas matemáticamente. También es posible realizar operaciones sobre señales de audio escribiendo una tilde después del objeto. Ver fig 2.14.2.8

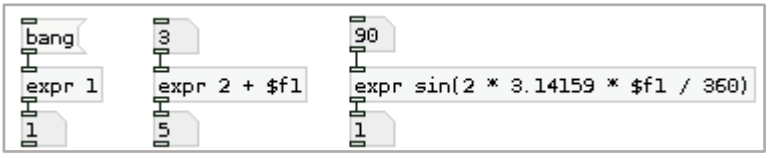


FIGURA 2.14.2.8EXPR

- **Delay**. El objeto de retardo (delay) envía un bang a su salida después de un retraso en milisegundos especificados por su entrada o su argumento de creación.

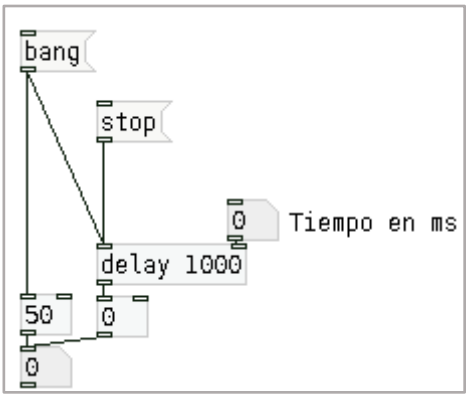


FIGURA 2.14.2.9 DELAY

- **Toggle**. Cuando se hace clic en el objeto palanca (toggle), este envía uno de dos posibles valores de acuerdo en la selección - cero cuando no está marcado y un número distinto de cero cuando se comprueba. El número distinto de cero es 1 por defecto, sin embargo, esto se puede cambiar en la opción "Propiedades". Toggle también tiene una entrada, que puede ser utilizada para mostrar si un número en la entrada es cero o no. Ver figura 2.14.2.10

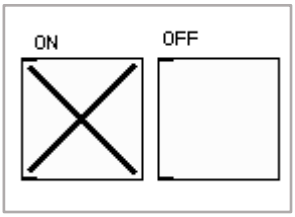


FIGURA 2.14.2.10 TOGGLE

- **Read sf.** El objeto readsf lee un archivo de sonido y lo envía como señal de audio. El archivo debe abrirse previamente con el mensaje “open” para cargarlo en la memoria.

La reproducción del sonido se hace con el mensaje “start” o con el número 1.Ver fig 2.14.2.11

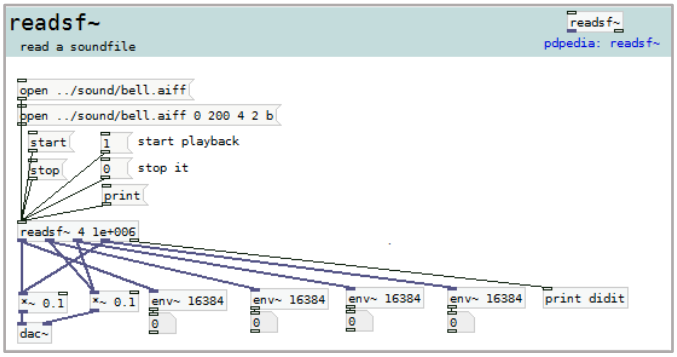


FIGURA2.14.2.11READ SF

- **Open panel.**Cuando openpanel recibe unbang en su entrada aparece un explorador de archivos en la pantalla; al seleccionar un archivo, su nombre y ruta aparecen en la salida. Esto puede usarse para facilitar la inclusión de archivos o para enviar la ruta de un archivo como variable de texto.

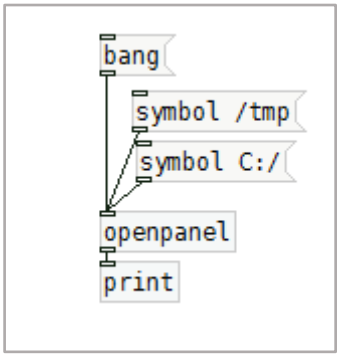


FIGURA2.14.2.12OPENPANEL

- **ROUTE.**El objeto ruta (route)encamina las entradas al puerto 1 o 2en valores de comparación. El objeto puede tomar “symbols” o “floats” como argumentos, el primer argumento determina qué modo de direccionamiento realiza.

Si seintroduce un “float” seoperará en modo de “float mode”.Si se utilizan símbolos, el float funciona en modo “selector mode”. Cuando llega una entrada que no cumple se envía a la última salida. Ver figura 2.14.2.13

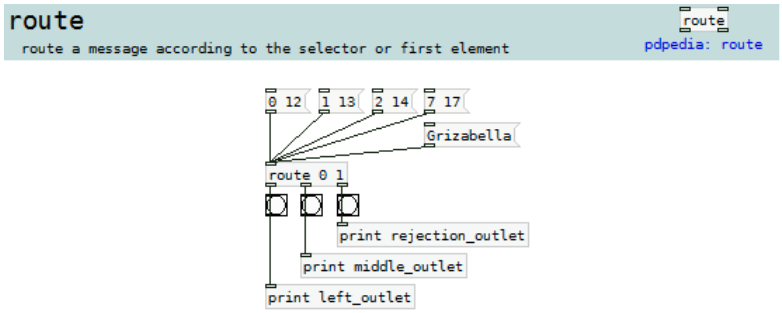


FIGURA 2 .14 .2.13 ROUTE

- **PACKOSC / UNPACKOSC.**Convierten mensaje de Pd a un mensaje OSCde forma viceversa. Ver figura 2.14.2.14. Se puede enviar el tiempo de envío en una cola indicando estampa de tiempo en uno de los argumentos.

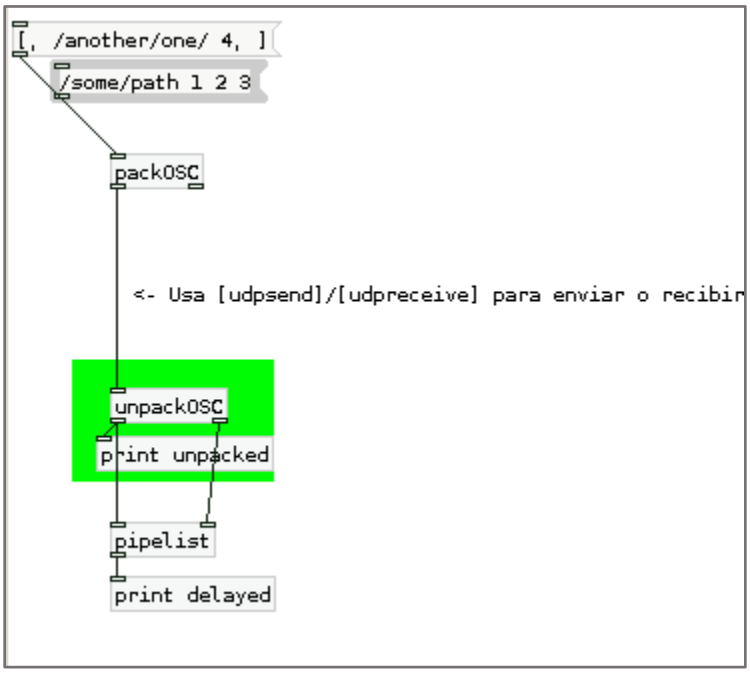


FIGURA 2 .14 .2.14PACK OSC

- **RouteOSC.**Direcciona los mensajes OSC a una salida dependiendo del primer elemento en la entrada. En la figura 2.14.2.15 tos los mensajes que llegan al argumento izquierdo del objeto se discriminan de acuerdo con la cadena después de "/". Si no se encuentra una coincidencia los mensajes se manda por la última salida de izquierda a derecha.

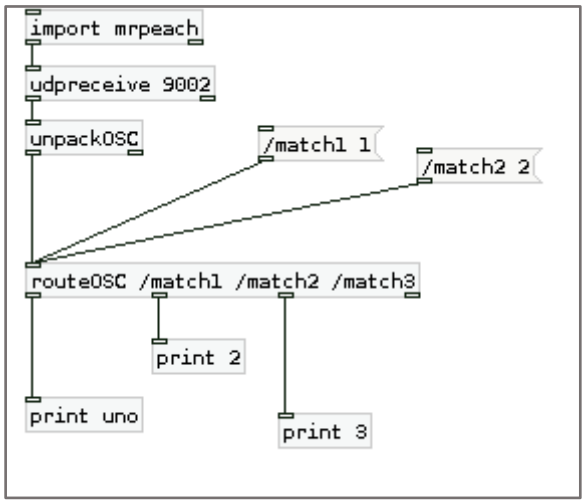


FIGURA 2 .14 .2.15 ROUTE OSC

- **UDPreceive/UDOsend:** Biblioteca procesa envío y recepción de mensajes mediante el protocolo UDP. La dirección IP y el puerto de recepción se configura con un mensaje en la entrada del objeto (fig. 2.14.2.16)

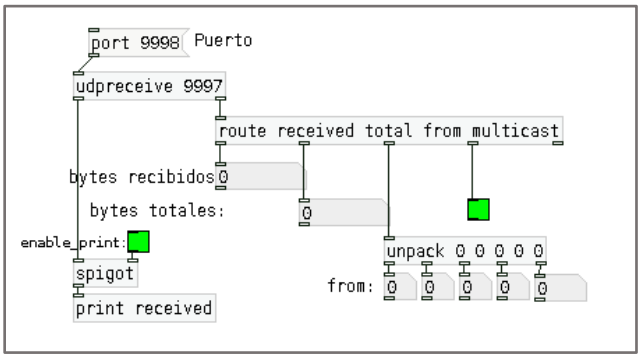


FIGURA 2 .14 .2.16UDPSEND/RECEIVE

2.16 Lenguaje de programación Processing

Processing es un lenguaje de programación en el cual se utiliza texto como entrada para obtener una imagen en la pantalla e interaccionar a través del mouse u otro dispositivo de entrada. Es reconocido por ser un ambiente para programar animaciones y sonidos, tiene aplicaciones artísticas, de diseño, arquitectura e investigación. Es usado regularmente como herramienta de enseñanza de ambientes ilustrados y profesional de producción.

Otra característica del software es que está construido en ambiente Java y Open source lo que contribuye a la generación de ideas y expansión de características.

2.16.1 Operación de píxeles en Processing

Se utiliza un sistema de posicionamiento en la pantalla. El origen lo referencia en la esquina superior izquierda e incrementa conforme se acerca al vértice contrario. La figura 2.16.1 Muestra gráficamente el espacio definido en Processing.

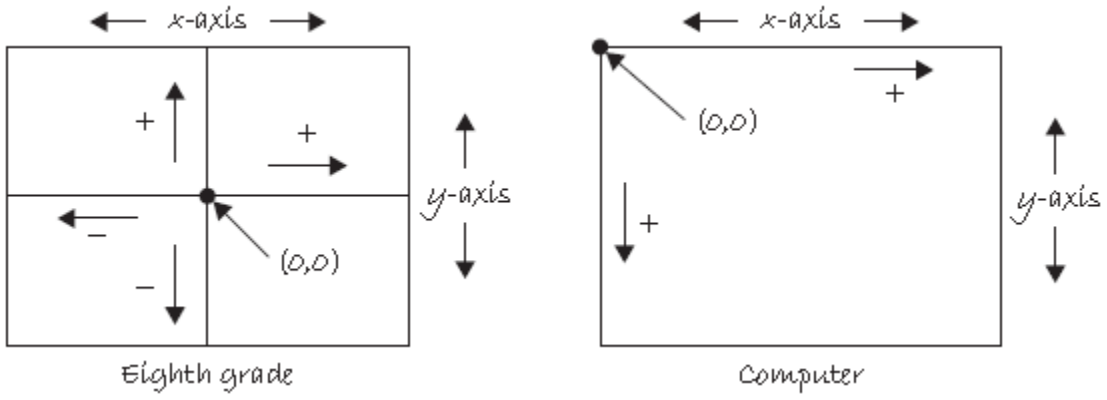


FIGURA 2.16.1OPERACIÓN DE PÍXELES

El código que se escribe en processing utiliza coordenadas. Los comandos puntuales son: POINT (X,Y), Line(X,Y,X.Y) , rect (X,Y,W,H). Para rect, el lenguaje define a X, Y como los puntos donde se inicializa la figura, W como el ancho y H es la altura.

La mayoría de los comandos utilizan coordenadas y después definen las dimensiones mediante posición del punto final o un tamaño en píxeles.

Processing utiliza un algoritmo interno de selección de píxeles para generar figuras predeterminadas, de tal manera que las figuras se generan con un mismo patrón. También reconoce parámetros en figuras como trazo grueso y fondo. Por lo que al programar figuras en el software se puede utilizar una variación escribiendo el código deestas mismascaracterísticas.

2.16.2 Escala de grises y color RGB

La escala de grises que utiliza el software está definida por una rango de 256 opciones (0,255), siendo 0 el color blanco y 255 el color negro. Por lo que programar un rectángulo de línea negra, relleno gris y fondo negro se haría de la siguiente manera:

```
background(255);
stroke(0);
fill(150);
rect(50,50,75,100);
```

Processing utiliza colores mediante las combinaciones RGB y usando el comando fill. La intensidad de cada uno de los colores estará definida mediante la misma escala de 256 opciones separadas en el comando por comas. Ejemplo:

Fill(255,200,200)= rosa

2.16.3 Interfaz processing

La interfaz de Processing es muy parecida a cualquier pantalla de editor de código combinado con un reproductor multimedia, como se muestra en la figura 2.16.3. Los programas en Processing se definen como esquemáticos (sketch) y se generan con la extensión ".PDE" Estos pueden contener, además del código, algunos archivos con elementos como imágenes y música, entre otros.

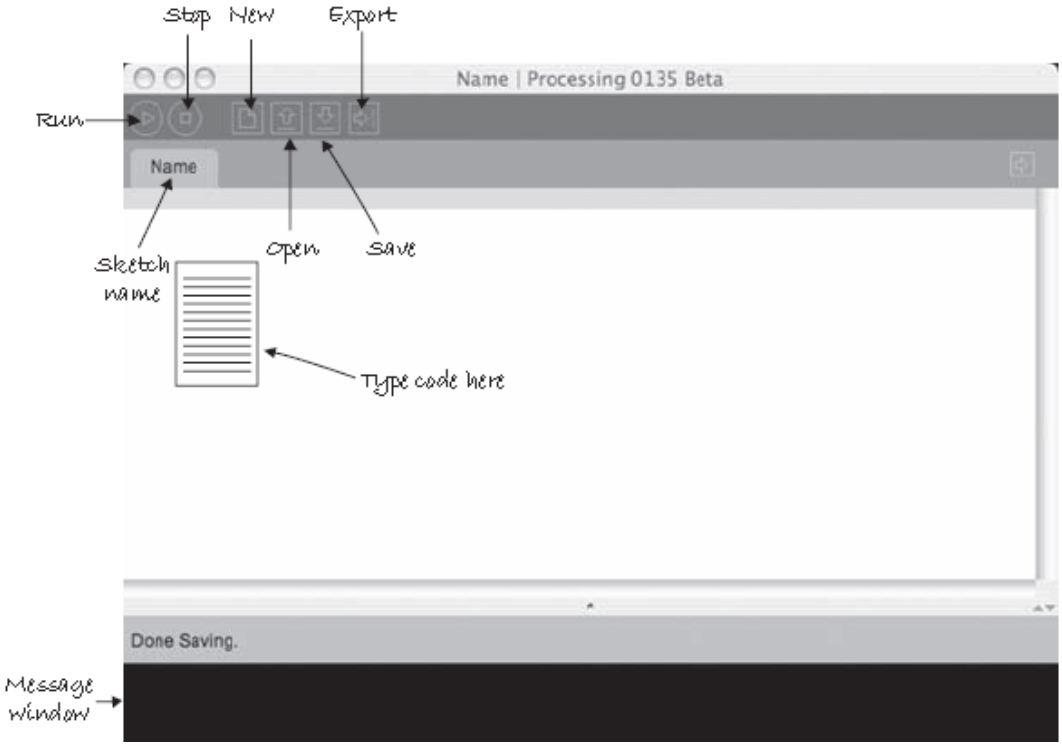


FIGURA 2.16.3 OPERACIÓN DE PÍXELES

Cada vez que un Sketch es ejecutado se puede exportar como un archivo de Processing, una aplicación o elemento web en alguna de las siguientes extensiones:

- .html - Despliega la aplicación HTML
- .gif - Despliega imágenes
- .jar —Genera Aplicación en plataforma.
- .java —Genera aplicación de contenido traducido a Java
- .Pde —Archivo de processing

2.16.4 Código de programaciónde Processing

El código de Processing es muy similar a la programación orientada a objetos, como el lenguaje Java, además de contener un reproductor de imágenes en el editor de texto. Por ejemplo, la ejecución de un archivo compilado o “sketch” se puede comparar con una clase creada en Java. Incluso las aplicaciones corren en una máquina virtual Java, lo que le da mayor versatilidad al sistema, pudiéndose utilizar en ambientes Mac, Windows, Android y otros sistemas operativos.

El código que se ejecutaen Processing funciona iterativamente, el programa siempre tratará de ejecutar las imágenes a la velocidad fijada por el usuario con la función “frameRate(n)”. Lavelocidadcon laque el software presenta lasimágenes por default tiene una frecuencia de imagen de 60 cuadros por segundo. Esteparámetro se puedemodificar y funcionaráapropiadamente dependiendo de las limitaciones del equipo. La variable frameCount()contiene el número de fotogramas o imágenes desplegadas desdeque el inicio de la ejecución del programa.

Algunas funciones se ejecutan una vez, en lugar de cada cuadro. La función setup () se ejecuta antes de draw()para que funciones como size() o loadImage() no se ejecuten en cadacuadro. El código fuera de setup()y draw()es declarado primero, como consecuencia el código dentro de setup() se ejecuta una vez, y finalmente el código dentro de draw() se ejecuta en un bucle continuo de arriba abajo.

2.16.5 Clase Display

Processing puede cargar imágenes, mostrarlas en la pantalla,cambiar su tamaño, posición, opacidad, y color utilizandola instrucción “Pimage(file.ext)”.De la misma maneraque cualquier otra variable, el softwarereconoceimágenes con distintostipos de extensión, paravisualizarlas es necesario previamentecargar laimagen con la función “loadImage” en el sketch.

También se puedeagregar texto con distintastipografías al espacio. Lasfunciones loadfont(data, x, y) y textfont (stringdata, x, y, ancho, altura) permiten definir perfectamenteel espaciodondeaparecerá el texto y su formato.

2.16.6 Clase Ratón

El diseño del ratón ha pasado por muchas revisiones pero su función sigue siendo la misma. En la solicitud de patente original de Douglas Engelbart en 1970 se refirió al ratón como un "indicador de posición XY".

Processing no es la excepción y el objeto del ratón o mousees controlar la posición del cursor en la pantalla y seleccionar elementos de la interfaz. La posición del cursor es leído por los programas como dos números. Estos números se pueden utilizar para controlar los atributos de los elementos en pantalla.

Las variables de Processing mouseX y mouseY,guardanla coordenada ***XY*** del cursor en relación con el origen en la esquina superior izquierda de la ventana de visualización.

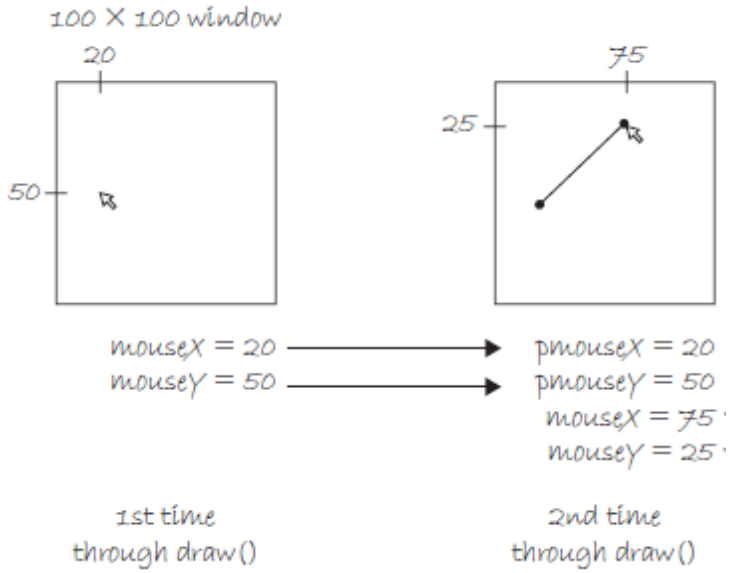


FIG 2.16.6CAMBIO DE POSICIÓN REPORTADO POR MOUSEX, MOUSE Y, PMOUSEX Y PMOUSEY

Cuando se inicia un programa, los valores mouseX y mouseY son 0. Si el cursor está en la izquierda, el valor mouseX es 0 y el valor aumenta a medida que el cursor se mueve a la derecha. Si el cursor se encuentra en la parte superior, el valor mouseY es 0 y el valor aumenta a medida que el cursor se mueve hacia abajo.

Las variables de proceso pmouseX y pmouseY almacenan los valores del ratón delúltimo fotograma con respecto a su posición anterior. Si el ratón no se mueve, los valores serán los mismos, pero si elratón se está moviendo rápidamente puede haber grandes diferencias entre los valores.La figura 2.16.6 muestra los datos obtenidos en lasvariables pmouse y mouse al cambiarde posición en el espacio gráfico.

La posición del cursor es un punto dentro de la ventana de visualización que actualiza cada fotograma. Es posible restringir los valores de ratón a una gama específica, calcular la distancia entre el ratón y otra posición, interpolar entre dos valores, determinar la velocidad del movimiento del ratón, y calcular el ángulo del ratón en relación con otra posición.
Por ejemplo,la función de constraint(valor, min, max) tiene tres parámetros. El parámetro valor es el número del límite, el parámetro min es el mínimo valor posible, y el parámetro max es el valor máximo posible. Esta función devuelve el número min, si el parámetro valor es inferior o equivalente a min; devuelve el número máximo, si el parámetro valor es mayor o equivalente a un máximo, y regresa el valor “0”si es entre el mínimo y el máximo. Cuando se utiliza con mouseX o mouseY, esta función puede establecer los valores máximos y mínimos paralos datos de coordenadas del ratón.
Otra función es“dist()”, lacual calcula la distancia entre dos coordenadas. Este valor se puede utilizar para determinar la distancia del cursor desde un punto en la pantalla, además de su posición actual. La función dist (x1, y1, x2, y2) tiene cuatro parámetros. Los parámetros x1 y y1 fijan la coordenada del primer punto, yx2 y y2definen la coordenada del segundo punto. La distancia entre los dos puntos se calcula como un númeroflotante.El valor devuelto por dist () se puede utilizar para establecer las propiedades de las formas.

2.16.7 Método Clic

Processing puede detectar cuando se pulsan los botones del mouse. El estado de los botones y la posición del cursor juntos permiten al ratón llevar a cabo diferentes acciones. Por ejemplo, al pulsar un botón cuando el ratón está sobre un icono, se puede seleccionarel

icono y moverlo a una ubicación diferente en la pantalla. Parahacer esto seutiliza la variable mousePressed, cuyalógica esverdadera si se pulsa un botón del ratón y falsa si no se pulsa ningún botón alguno del ratón.

Las funciones para los eventos de clic en el ratón son mousePressed, mouseReleased, mouseMoved y mouseDragged. Susaplicaciones se describen a continuación:

- mousePressed. Código dentro de este bloque se ejecuta una vez cuando se presiona un botón del ratón
- mouseReleased. Código dentro de este bloque se ejecuta una vez cuando se suelta un botón del ratón
- mouseMoved. Código dentro de este bloque se ejecuta una vez cuando se mueve el ratón
- mouseDragged. Código dentro de este bloque se ejecuta una vez cuando se mueve el ratón mientras se pulsa un botón del ratón

2.16.8 Clase Overcircle y Overrect

Las dos formas que reconoce el ratón más fácilmente dentro de sus fronteras son el círculo y el rectángulo. Las clases OverCircle y OverRect se han escrito para que quede claro cuando el ratón está sobre estos elementos.

La clase OverCircle tiene cuatro campos. Establece la coordenada X, coordenada Y,“d” del diámetro, y el valor de gris para el relleno. El método update () se ejecuta cuando el ratón está sobre el elemento.La función dist () dentro de update () calcula la distancia desde el ratón al centro del círculo; si la distancia es menor que el radio del círculo, el valor de gris se establece a blanco.

Los campos y métodos para la clase OverRect son idénticos a los de OverCircle, pero el criterio está en función del ancho y la altura del rectángulo y no el diámetro de un círculo. La expresión relacional dentro de update() comprueba para ver si los valores mouseX y mouseY entrantes están dentro del rectángulo

2.16.9 Clase botón.

La clase Botón o Button tiene un estado adicionala la clase OverRect. Mientras OverRect reconoce cuando el ratón está sobre la forma, la clase Button hace que sea posible determinar si el ratón está sobre la forma y desencadenar un evento. Adicionalmente si se presiona el ratón se pueden desencadenar eventos por separado.

El método update () establece elcampo,verdadero o falso, dependiendo de la ubicación del ratón. El método press() se puede ejecutar desde de la función del programa principal mousePressed(). Cuando se ejecuta, establece campo press() a verdadero si el ratón está actualmente sobre el botón. El método de liberaciónse debe ejecutar desde el programa principal.

Cuando se suelta el ratón del campo pulsado se re-establece en falso, lo que prepara el botón a que sea pulsado de nuevo. El método de visualización () dibuja el botón a la pantalla y establece su valor de gris en función del estado actual del ratón en relación con el botón. La figura 2.16.19 muestra el cambiodeestado cuando un botón es activado o desactivado

FIGURA 2.16.19 CAMBIO DEESTADO EN BOTÓN

Para utilizar la clase Button,se tiene que añadir a un sketch y llamar a sus métodos desde las funciones de eventos de ratón. Como la mayoría de los objetos, se debe crear dentro de setup () y se actualizará y se mostrará dentro de draw (). Los métodos del objeto relacionados con el estado del ratón se deben ejecutar de forma explícita desde las clases mousePressed () y mouseReleased ()

2.16.10 Casillas de verificación y botones de radio

Una casilla de verificación es un botón con dos estados, ON y OFF, que cambian cuando se selecciona este elemento. Los campos y métodos son similares a los de la clase Button, pero el método display () es único. Cuando el campo de la press() es cierto, una X se dibuja en el centro de la caja para mostrar que el estado está en ON. La figura 2.16.10.1 ejemplifica la selección de una casilla de verificación activada por el método clic.

Las casillas de verificación y botones de radio forman otra categoría de elementos de la interfaz. Las casillas de verificación funcionan de forma independiente a los demás, mientras que el estado de un botón de radiodepende de otros elementos en su grupo. Así casillas de verificación se utilizan como un elemento de la interfaz cuando más de una opción está disponible, y los botones de opción se utilizan cuando sólo un elemento dentro de una lista de opciones se puede seleccionar.

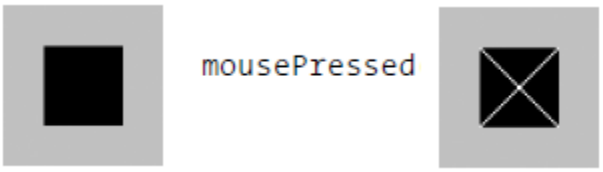


FIGURA2.16.10.1SELECCIÓN DE CASILLA DEVERIFICACIÓN

Al igual que una casilla de verificación, un botón de radio también tiene dos estados, ON y OFF. A diferencia de las casillas de verificación, los botones de opción se utilizan en grupos de dos o más, y sólo uno de los elementos en el grupo se puede seleccionar a la vez. La figura 2.16.10.2 muestra como solo un radio puede serseleccionado dentro de varias opciones.

Cada botón tiene que ser capaz de convertir los otros elementos en estado OFF cuando esté seleccionado. La clase Radio es únicaya que está diseñada para utilizarse como parámetro,esto hace que sea posible crear para cada elemento dentro de la matriz de Radio una referencia a los otros elementos de la matriz.

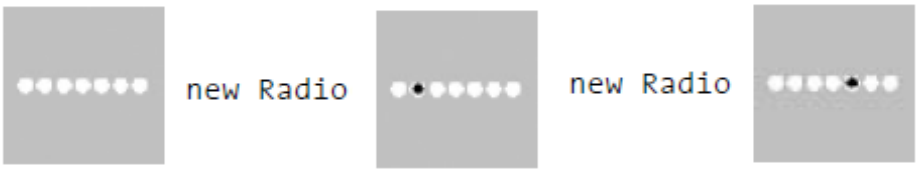


FIGURA2.16.10.2 SELECCIÓN DE BOTÓN DERADIO CADA OBJETO RADIO ESTABLECE EL ESTADO DELOS DEMÁS OBJETOS DE RADIO EN OFF. SI EL ESTADO DEL OBJETO ES ON, UN CÍRCULO INTERIOR NEGRO, INSCRITO EN EL MÉTODO DISPLAY()

2.16.11 Barra de desplazamiento (Scrollbar)

Una barra de desplazamiento (scrollbar) o simplemente scroll, es un elemento de interfaz para la selección de un valor en un rango de valores posibles. El scrollbar es típicamente un elemento rectangular estrecho con un elemento interior móvil típicamente denominadopulgar. El usuario puede mover el pulgar entre los extremos de la barra de desplazamiento arrastrándolo a una nueva posición.

La clase Scrollbar es similar a las otras clases de Processing, sin embargo el campo bloqueado es único, por lo que es posible que el cursor que siga actualizando la posición del pulgar incluso si el cursor se mueve fuera de la zona de la barra de desplazamiento. Esta clase fue diseñada para que los objetos que tenganuna Scrollbar tengan la posibilidad de seleccionar losvalores mínimos y máximos. El método getPos () devuelve el valor actual del elemento de botón dentro del rango de la barra de desplazamiento definido por los campos MINVAL y MAXVAL.

Un objeto de la clase Scrollbar se declara en la parte superior del código, al igual queotroselementos, creado en el setup (), muestra y se actualiza en draw () yes controlado por los eventos mousePressed () y mouseReleased () descritos en la figura 2.16.11.1

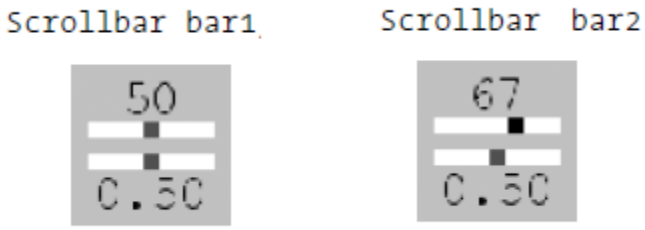


FIGURA 2.16.11.1 REPRESENTACIÓN DESCROLLBAR

El número que regresa getPos() de la barra de desplazamiento se puede utilizar para controlar cualquieraspecto de un programa, como el volumen del audio o la frecuencia de la señal desalida.

3. Construcción de pantalla

3.1 Introducción

El área de estudiode interacción humana con computadoras HCI (Human Computer Interface)es una rama de las ciencias de la computación que tiene como propósito desarrollar

tecnologías que faciliten la comunicación humano-máquina. A partir del crecimiento masivo del uso de dispositivos móviles se han modificado las técnicas de interacción hasta llegar al control “multi-touch” el cual se refiere a la manipulación deaplicaciones gráficas utilizando varios puntos de contacto. Esencialmente las interfaces de este tipo consisten en una pantalla táctil, software con la capacidad de reconocer múltiples entradas al mismo tiempo además de un sistema audiovisual de realimentación.

La detección de puntos de contacto se puede realizar mediante sensores de proximidad capacitivos o resistivos. Existen cinco técnicas ópticas principalespara la construcción de sistemas multi-touch: FTIR,iluminación difusa (Rear ID: Rear Diffused Illumination) usada en la Surface Table de Microsoft, LLP (Laser Light Plan) usada en LaserTouch de Microsoft, LED -LP (LED Light Plane) y finalmente DSI (Difussed Surface Illumination). En el presente trabajó se aplicó la técnica FTIR.

El sistema se conforma de cuatro elementos principales:

- Iluminación infrarroja
- Cámara IR
- Superficie
- Realimentación audiovisual (proyector y sonidos)

Los elementos de realimentación audiovisual están basados en software libre y son los siguientes:

- Reactivision. Procesa puntos de contacto y manda mensajes de control (TUIO) a las aplicaciones audiovisuales.

- Processing. Genera interfaz gráfica a partir de los mensajes TUIO (posición x, y, ángulo de las figuras)
- PureData. Procesamiento de audio y ejecución de secuenciador a partir de los puntos activados por el usuario en la interfaz

3.2 Iluminación infrarroja

Se utiliza este tipo de luz para aislar el reconocimiento táctil de la realimentación audiovisual. El fenómeno óptico que permite iluminar la pantalla de acrílico se denomina FTIR que es una técnica empleada también en biometría para adquisición de huellas digitales. Cuando la luz viaja de una interfaz a un medio con menor índice de refracción (acrílico hacia aire), se refracta en función del ángulo de incidencia y del ángulo crítico hasta formar reflexión interna total como se muestra en la figura 3.2.1.

El principio es usado en fibra óptica y otras guías de onda para transportar luz con muy pocas pérdidas. Es posible acoplar otro material que frustre ese fenómeno y permita que la luz escape de la guía de onda en cierto punto de contacto.

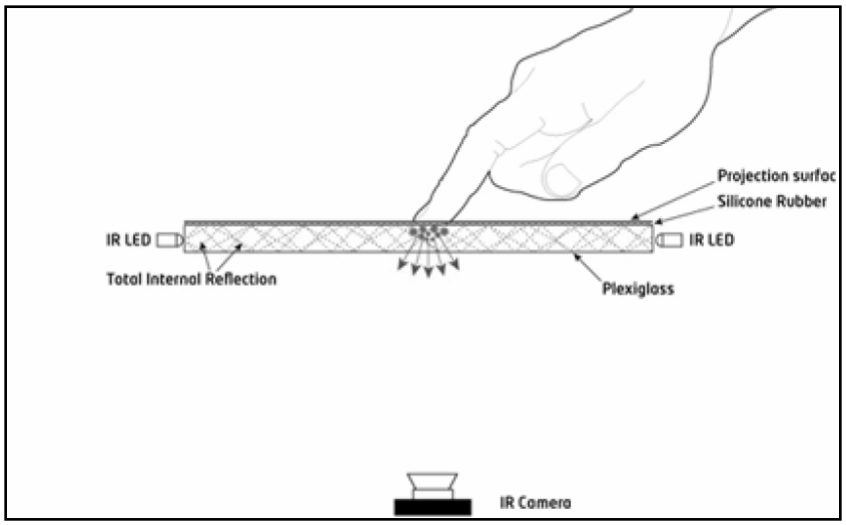


FIGURA 3.2.1FENÓMENO FTIR

El LED utilizadoes el modelo IR383Everlightdebido a la potencia y bajo costo. EL LED opera en 940 nm, longitud de onda a la cual el sensor de la cámara es sensible. Se requiere cubrir los cuatro lados de la pantalla que en su totalidad suman 420 [cm].El ángulo devista recomendado es menor a 48° pero se trabaja con uno de 20° y se fija a 1.5 [cm] la disposición de los mismos para aprovechar la iluminación de cada LED.

Los criterios de diseño son los siguientes:

- Longitud total pantalla: 420 [cm]
- Separación entre cada LED: 1.5 [cm]
- Número de LED´s : 224

De acuerdo con la hoja de especificaciones se calculan los requisitos de la fuente:

V = 1.5 [V]
R = 1 [Ω]
Imax= 1 [A]
V_fuente = 12 [V]
I_fuente = 10 [A]

Resulta el siguiente circuito con 28 ramas en paralelo, cada rama con 8 LEDs y una resistencia de 1 [Ω] como se muestra en el esquemático de la figura 3.2.2

- Cada resistencia disipa 0.4 [mW]
- Todas las resistencias disipan 11.2 [mW]
- Todos los diodos disipan 6720 [mW]
- La energía disipada por el arreglo es de 6731.2 [mW]
- El circuito demanda una corriente de 560 [mA] de la fuente

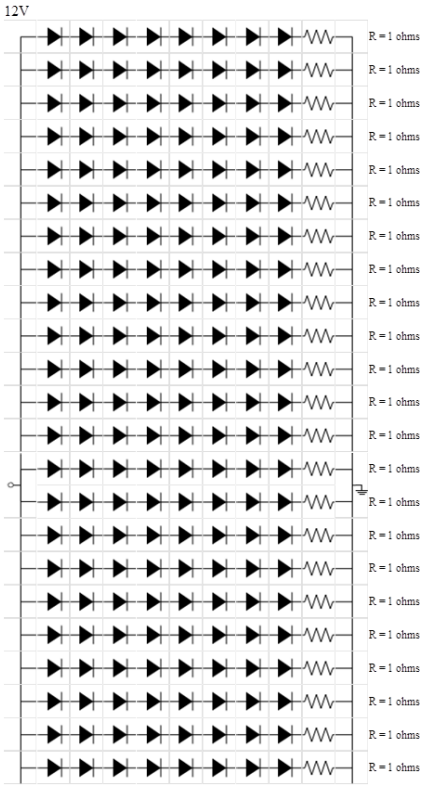


FIGURA 3.2.2CIRCUITO PARA ILUMINACIÓN IR

Para facilitar la aplicación de soldadura se perfora una superficie de cartón en la que se acomodaron los 8 LEDs de cada rama como se muestra en la figura 3.2.3.

FIGURA 3.2.3CIRCUITO PARA ILUMINACIÓN IR

Las tiras deLEDs se montanen una vía para garantizar una iluminación uniforme, evitarel calentamiento y garantizar que la luz incida directamente en los cantos de la pantalla como se muestra en la figura 3.2.4.

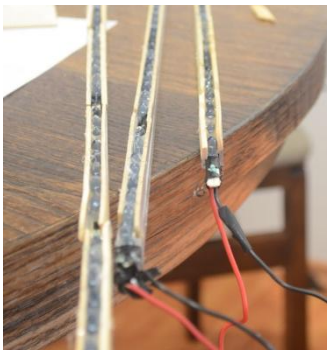


FIGURA 3.2.4GUÍA DE LEDS

3.3 Cámara Infrarroja

Las cámaras digitalesson sensibles a una porción del espectro IR pero se fabrican con un filtro para capturar sólo la luz visible. Se sustituye dicho filtro con unoque elimine el espectro visible para tener una cámara sensible sólo al infrarrojo. La cámara de elección es PS3 Eye, figura 3.3.1, usada para interacción web en consola PlayStation 3.Se opta por este modelo debido a su costo asequible, facilidad para modificarla e integración con los sistemas operativo Mac y Windows.

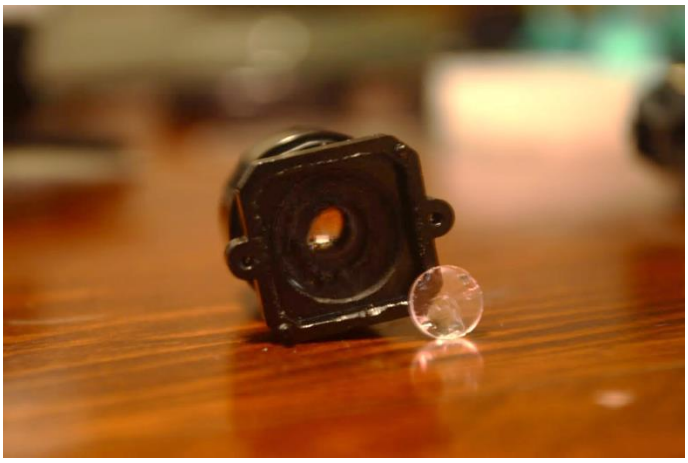


FIGURA 3.3.1SENSOR PS3 EYE & FILTRO IR

Como filtro de luz visible se agregaun pedazo de placa de rayos X en el sensor, de esta forma sólo se permite el paso de los rayos IR

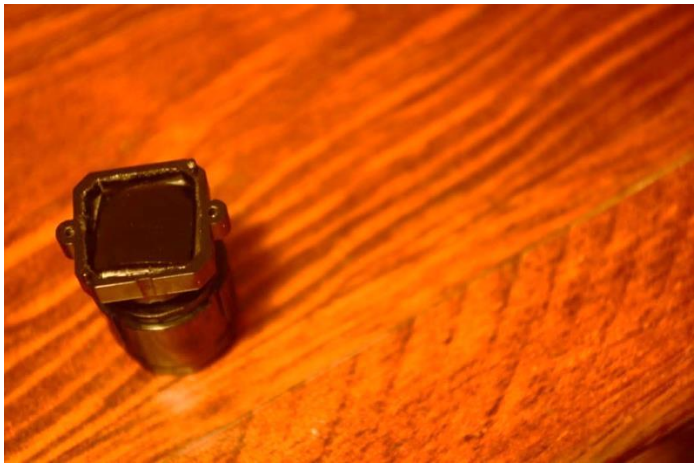


FIGURA 3.3.2SENSOR PS3 EYE &FILTRO LUZ VISIBLE

Para utilizar la cámara desde la computadora en lugardel Play Station 3 se necesita instalar drivers de acuerdo con el sistema operativo. Para el sistema operativo Mac se optó por la aplicación “Macam”.El componente build macam.component debe agregarse a la ruta “/Library/QuickTime/” y al abrir macam se verifica que la cámara se reconozca adecuadamente como se muestra en la figura 3.3.3

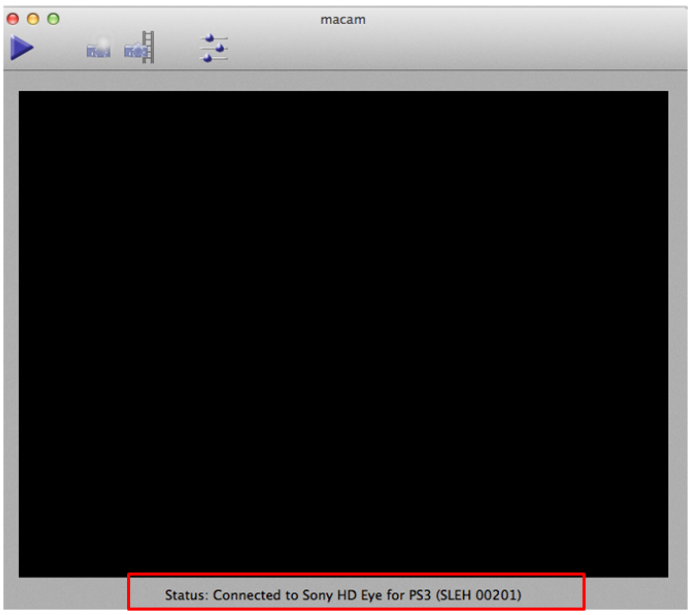


FIGURA 3.3.3PS3EYE EN SOFTWARE MACAM

Los ajustes comunes se pueden editar en el archivo "Reactivision.xml", donde todos los cambios se guardan automáticamente cuando se cierre la aplicación. En Mac OSX, el archivo de configuración XML se puede encontrar dentro de la carpeta de recursos del paquete de aplicaciones. Seleccionar"Mostrar contenido del paquete" en el menú contextual de la aplicación con el fin de acceder y editar el archivo.

La aplicación Reactivision normalmente envía los mensajes TUIO al puerto 3333 en el *localhost* (127.0.0.1). Se puede cambiar esta configuración mediante la adición o edición de la etiqueta XML <tuio host="127.0.0.1" port="3333">
La etiqueta <fiducial engine="amoeba" tree="default"/> XMLpermite seleccionar el motor de fiducial o una orden de árbol *amoeba* alternativa. El motor por defecto usa el conjunto fiducial 'amoeba' más rápido y eficaz.

El atributo de visualización define la pantalla por defecto al arrancar la aplicación. El comando <image display="dest" equalize="false" gradient="32" /> permite ajustar el valor de la puerta gradiente por defecto. La aplicación Reactivision viene con un módulo de

sustracción de fondo, que en algunos casos puede simplificar el rendimiento del reconocimiento del dedo y los marcadores *fiducial*. Dentro de la aplicación en ejecución puede alternar esto con la tecla 'E' o volver a calcular la sustracción de fondo presionando la barra espaciadora.

Por defecto Reactivison elegirá la cámara integrada en la laptop. Para permitir al usuario qué cámara elegir es necesario editar el archivo camera.xml, como en la figura 3.3.5, en el folder de instalación. Es necesario cambiar el campo “camera id” de 0 a “auto”.

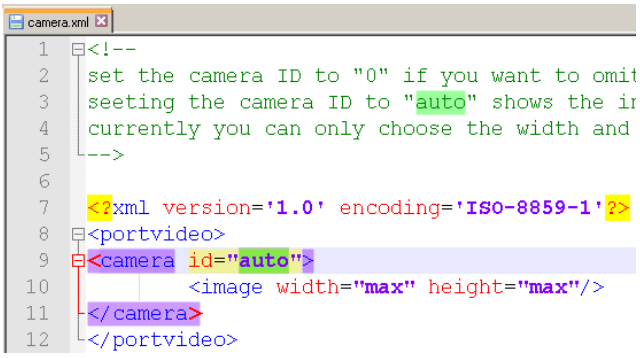


FIGURA 3.3.5ARCHIVO DE CONFIGURACIÓN CAMERA.XML

De esta forma se eligió la cámara modificada y se validó la visualización de los LEDs en funcionamiento:

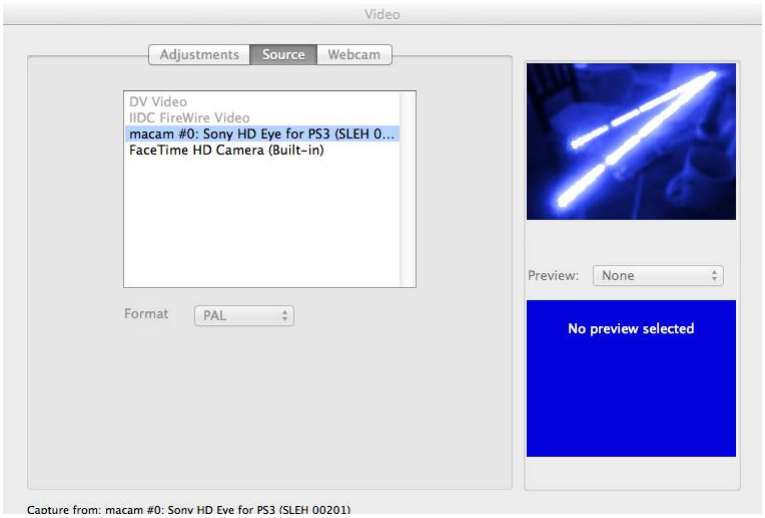


FIGURA 3.3.6DETECCIÓN DE LEDS INFRARROJOS

La cámara usada en este trabajo es de lente gran angular con el fin de aumentar el área visible a una distancia mínima. Estas lentes lamentablemente distorsionan la imagen y es necesario corregir distorsión y alineación general de la imagen para tener un cursor en la posición correcta al tocar la superficie. Se imprime una hoja de calibración, como se muestra en la figura 3.3.7, y se coloca en la pantalla para ajustar los puntos visualizados por Reactivision con los puntos reales de contacto.

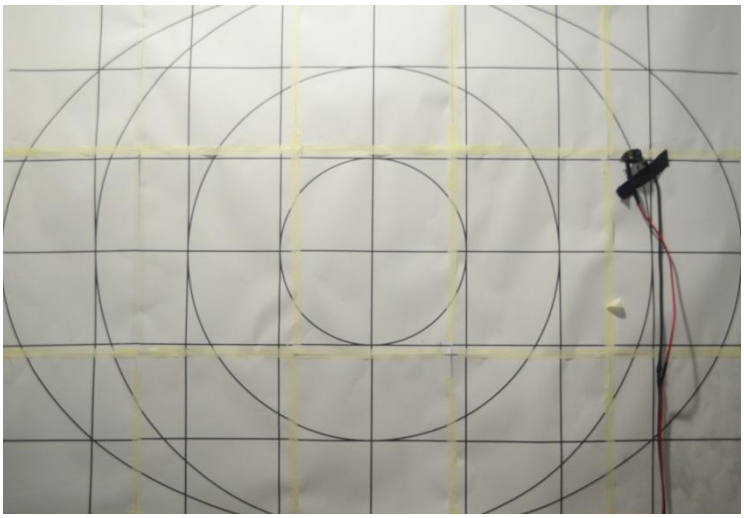


FIGURA 3.3.7 REJILLA DE CALIBRACIÓN CON LED FIJO EN PUNTO DE CONTACTO

En la figura 3.3.8 se muestra la ejecución del modo de calibración de Reactivision, este se activa con la tecla ‘C’. Las teclas A, D, W, X permiten navegar dentro de la rejilla y con las teclas de cursor se ajustan los puntos en la cuadrícula. Con la tecla 'J' se restablece toda la cuadrícula de calibración, con la tecla 'U' se restablece el punto seleccionado y con la tecla "K" se revierte a la red guardada.

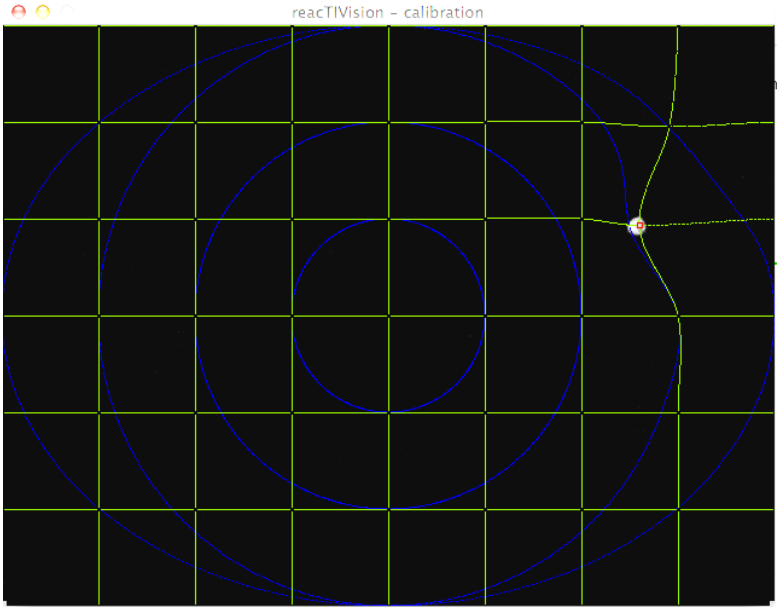


FIGURA 3.3.8 AJUSTE DE PUNTO FIJO MEDIANTE REJILLA DE CALIBRACIÓN

Para comprobar si la distorsión está funcionando correctamente se presionala tecla "R". Esto muestra la imagen de vídeo en directo totalmente distorsionada en la ventana de destino. El algoritmo de distorsión sólo corrige las posiciones encontradas en el lugar de la imagen completa. En la figura 3.3.9 se muestra el punto de contacto fijo y el punto F corregido mediante una transformación geométrica.

FIGURA 3.3.9 PUNTO FIJO Y PUNTO CORREGIDO

3.4 Superficie

Se prefiereuna pantalla de acrílico debido a sus propiedades ópticas. Las dimensiones se especifican en la figura3.4.1. El espesor es importante a dimensiones grandes para evitar que se doble al tacto. La placa tiene una superficie irregular en sus lados debido al proceso de corte en fábrica. Se solicitó a un taller realizar un pulido de los mismos para permitir que la iluminación sea uniforme y facilitar el fenómeno de FTIR (figura 3.4.2)

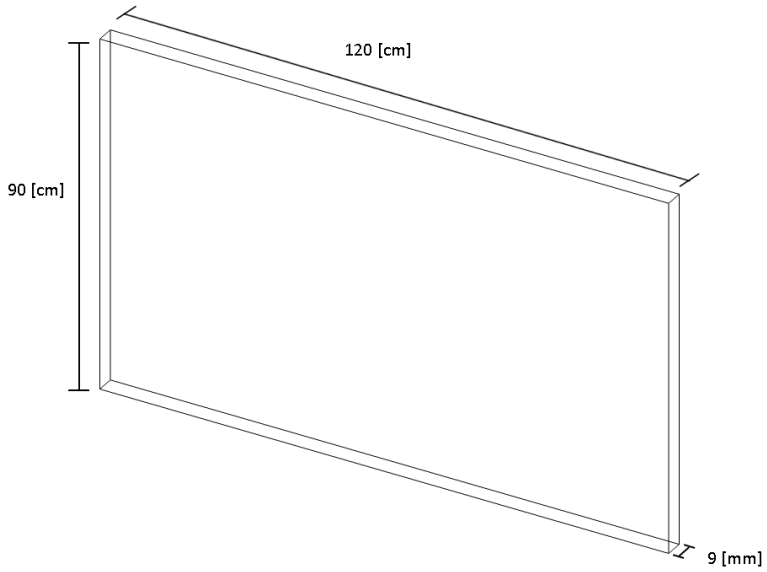


FIGURA 3.4.1DIMENSIONES DE PLACA ACRÍLICA

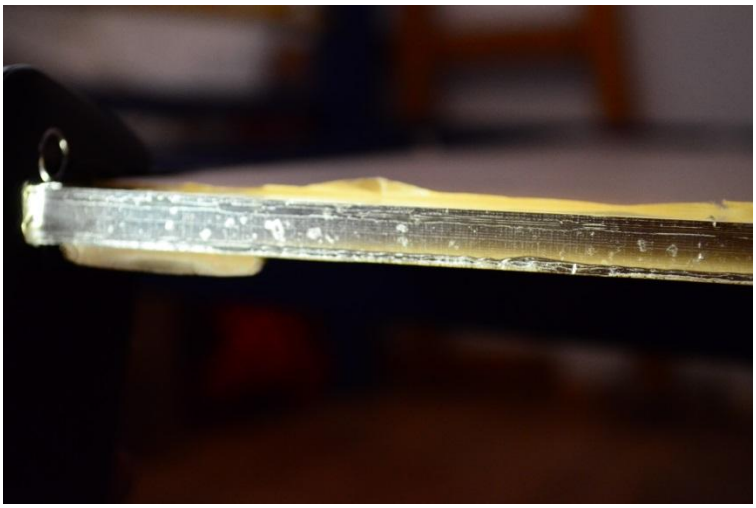


FIGURA 3.4.2CANTOS PULIDOS DE PLACA

Entre el proyector y la pantalla de acrílico es necesario agregar un difusor;sin él la cámara no sólo ve los puntos de contacto, también ve todos los objetos detrás de la superficie.

Al usar un difusor sólo los objetos brillantes son visibles y la aplicación Reactivision capta menos ruido. En la figura 3.4.3 se muestra la luz de un celular a través del difusor.

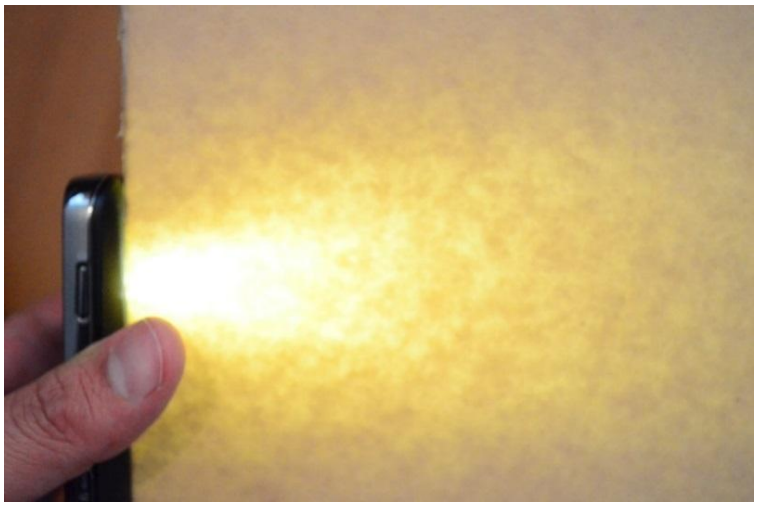


FIGURA 3.4.3DIFUSOR

Los dedos húmedos hacen mejor contacto sobre la superficie y los dedos secos no permiten completamente frustrar la reflexión interna. Para solventar este problema se añadió una superficie de silicón sobre el acrílico llamada “capa de acoplamiento”.

Esta capa ayuda también a la detección ya que la superficie de contacto debe ser de $\eta>$ índice de refracciónmayor al del acrílico para la longitud de onda que viaja en el acrílico, de tal forma que acople la superficie del acrílico bajo presión, permita el fenómeno de FTIR y desacople una vez que la presión del usuario se ha liberado.

El producto utilizado fue SORTA Clear 18, mostrado en la figura 3.4.4, del fabricante SMOOTH-ON, material utilizado en la industria de efectos especiales.Primero se mezclaron las masas respetando la proporción 100 a 10 entre los componentes A y B respectivamente. Se eliminaron burbujas de aire implicadas en el proceso.

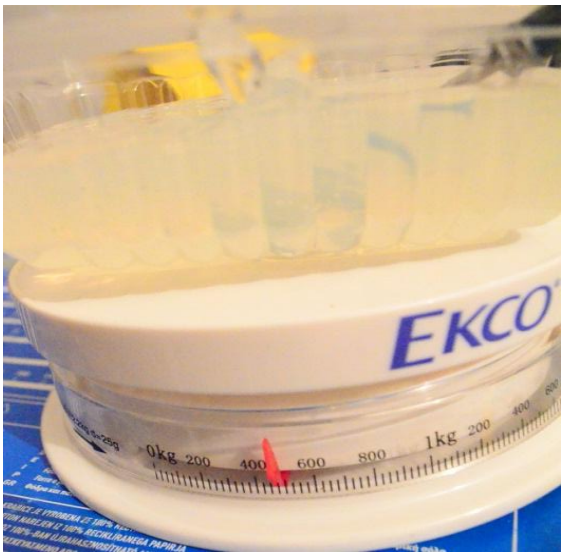


FIGURA 3.4.4MEZCLA SORTA CLEAR

Una vez que se tuvo una mezcla uniforme se aplica con una barra de metal sobre la placa de acrílico, cuidando que los lados estuvieran nivelados para evitar desplazamientosdurante el proceso de secado. El secado del silicón se hace atemperatura de 35 [° C] durante 12 horas:

FIGURA 3.4.5APLICACIÓN SORTA CLEAR

FIGURA 3.4.6SECADO DE CAPA DE ACOPLAMIENTO

En los lados se montaronlas tiras de LEDs previamente alambradas. Por último la placa se monta en un soporte de metal de las siguientes dimensiones para propósitos de transporte y utilidadespara el usuario. En la figura 3.4.7 se observa el marco de LEDs encendidos sobre el acrílico.

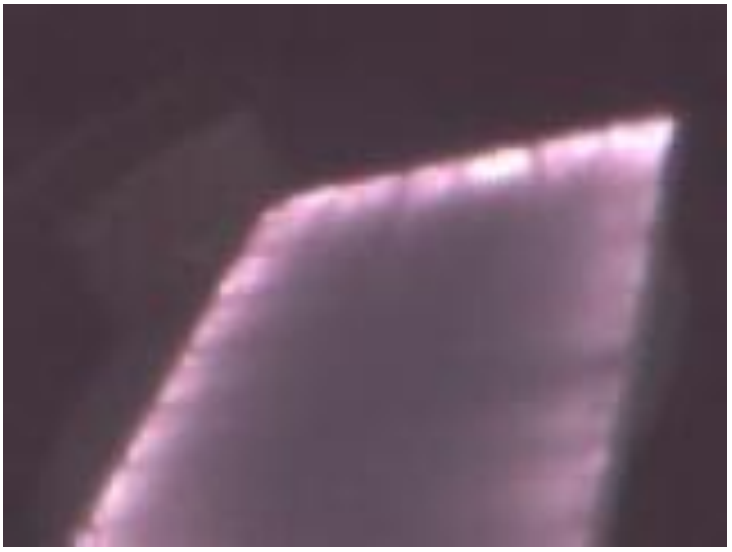


FIGURA 3.4.7MARCO DE LEDS SOBRE ACRÍLICO

FIGURA 3.4.8SOPORTE PARA PANTALLA

3.5 Realimentación audiovisual

La etapa final de ensamble requiere del ambiente audiovisual que logrará la realimentación con el usuario a partir de los puntos de contacto. Para ello se instala un proyector de la marca EPSON modelo PowerLite Presenter cuya capacidad es de 2000lúmenes. Los elementos de interacción se tratarán en capítulos subsecuentes así como la descripción de los algoritmos y estructuras de control.

El esquema de funcionamiento se muestra en la figura 3.5.1

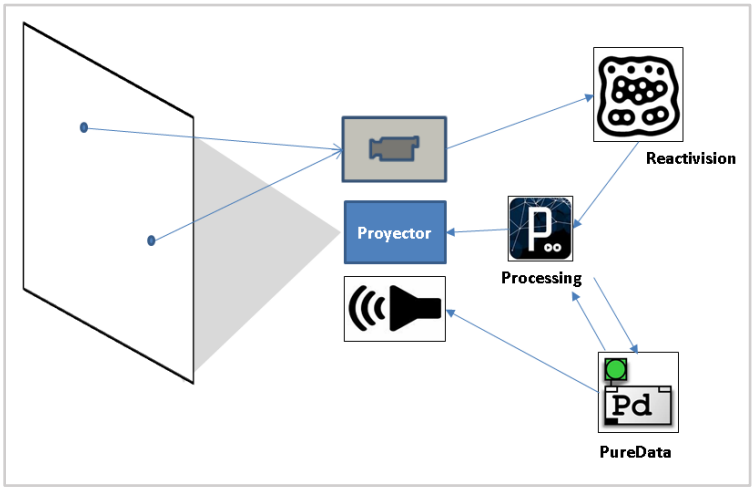


FIGURA 3.5.1ESQUEMA DE INTERACCIÓN AUDIOVISUAL

4. Secuenciador

4.1 Introducción

Un secuenciador es una aplicación capaz de reproducirsonidos en determinados intervalos de tiempo de manera cíclica. La distribución de los tiempos en los que se activan los sonidos corresponde a la notación tradicional en la música occidental. Este proyecto contiene un secuenciador de 16 pasos distribuidos en bloques de 4 pasos cada uno para acompañar un compás de cuatro cuartos (4/4).

Un concepto heredado de la músicaes el de “voz” y se refiere a un instrumento que suena simultáneamente con otro. En el presente trabajo cada paso tiene la capacidad de reproducir cuatro sonidos al mismo tiempo, en otras palabras es un secuenciador de 16 pasos y cuatro voces. Cada voz puede programarse para un sonido en particular, por ejemplo los componentes básicos de una batería tradicional. En la figura 4.1.1 se aprecia el control de un secuenciador de ocho pasos (columnas) y cuatro voces (filas)

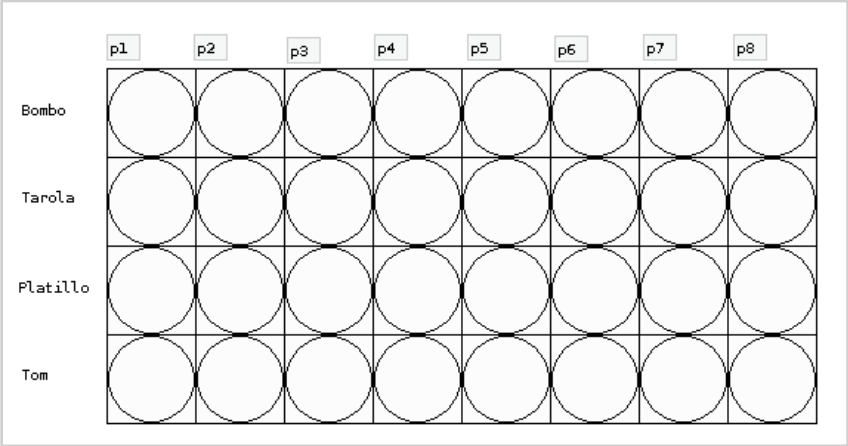


FIGURA 4.1.1CONTROL DE SECUENCIADOR DE OCHO PASOS

4.2 Señal de reloj

Para lograr la sincronización en las distintas voces que se ejecutarán es necesario definir una señal de reloj común y repartirla en los 16 pasos. Cada pasoactivalas 4 voces que le correspondan. Existe en Pd un objeto que funciona como metrónomo llamado [metro]:

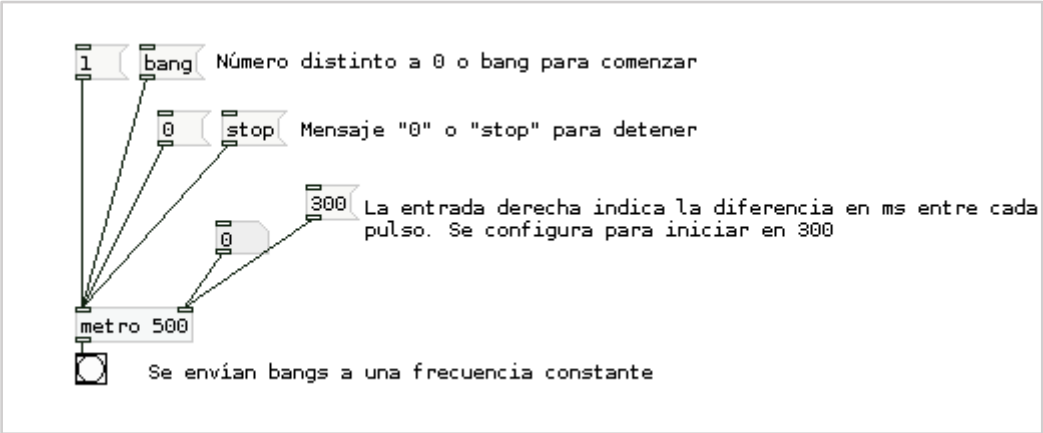


FIGURA 4.2.1OBJETO METRO EN PD

Debido a que el secuenciador corre de manera iterativa se necesitaun contador que registre cada incremento en el tiempo. Se agrega el objeto [float] que almacena una variable y la manda al objeto al que está conectado en su salida. Al conectar con el objeto [+1] se incrementa en 1 el valor del contador a partir de cada pulso que reciba [float] en su entrada.

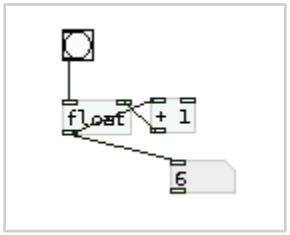


FIGURA 4.2.2OBJETO +1 PD

El contador se incrementa durante el tiempo que el programa corra. El secuenciador debe repartir ese número en 16 valores distintos: 1, 2, 3,…hasta llegar a 16 y volver a 1 al terminar

la secuencia. Para ello se usa el objeto [mod] que proporciona el resto de la división (módulo) entre un entero determinado. Se tienen por ejemplo los siguientes posibles combinaciones:

1 / 16- - > Módulo = 1

16 / 16 - - >Módulo= 0

17/16- - > Módulo= 1

De esta forma se logra una secuencia que se repite durante la ejecución del programa. Las conexiones en Pdson:

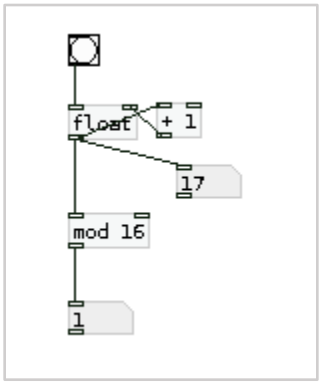


FIGURA 4.2.3CONEXIONES PARA CONTADOR

Cada resultado debe direccionarse a un paso en el secuenciador. El objeto [route] manda un [bang] a una salida de acuerdo con el valor de comparación, en este caso cada [bang] será asignado a una señal de reloj individual.

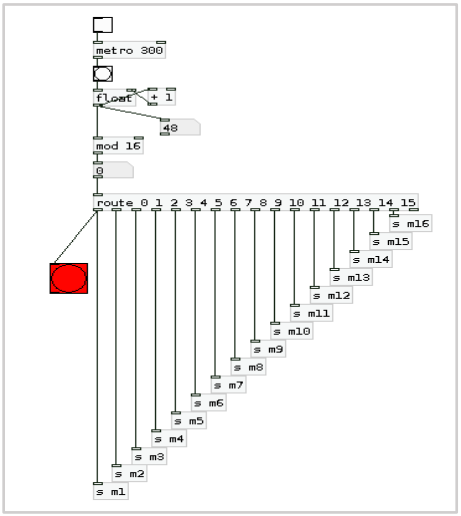


FIGURA 4.2.4ASIGNACIÓN DE OBJETO BANG A SEÑAL DE RELOJ

De esta forma se reparte la señal de reloj original en 16 variables de reloj distintas. La notación de las variables es: m<tiempo> (m1, m2, m3, etc.)

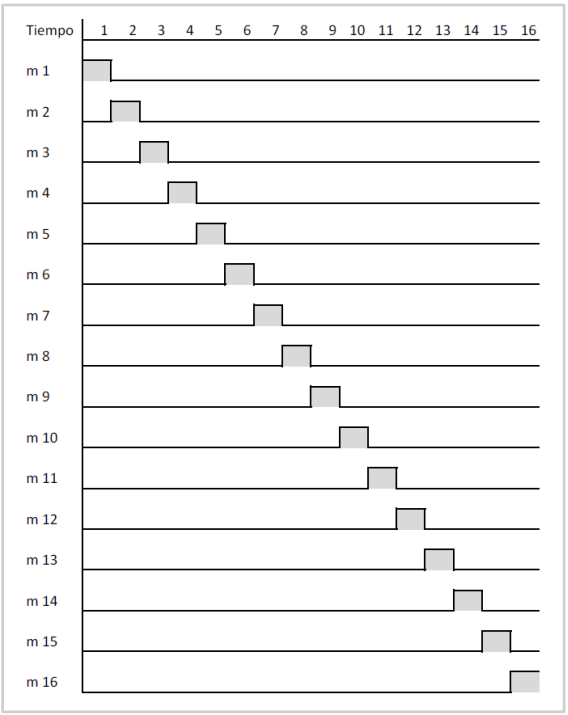


FIGURA 4.2.5SEÑALES DE RELOJ

Para facilitar el control del usuario se realiza una conversión de tiempo en [BPM] (beats por minuto) a [ms]. Se programa con el objeto [expr] el cual toma una variable en su entrada izquierda y realiza una operación con la variable definida tras el signo $.

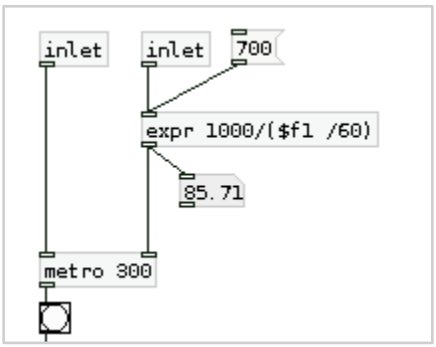


FIGURA 4.2.6CONVERSIÓN DE BPM A MS

En la figura 4.2.6 el número 700 [BPM] equivale a una diferencia de 85.71 [ms] en cada pulso de reloj. Esta conversión se encapsula con los objetos mencionados en un patch llamado “metro_16_pasos.Pd”. La entrada izquierda enciende o apaga la señal de reloj general y la entrada izquierda convierte los valores de tiempo a ms para el objeto [metro]. Un slider de control se acota en el rango (70, 500) [bpm].

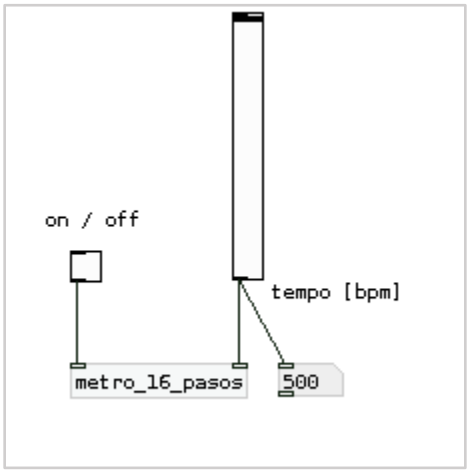


FIGURA 4.2.7CONTROL DE TIEMPO

4.3 Tablero de pasos

Se requiere un mecanismo para controlar el envío de un bang en cada voz, dicho de otra forma es necesario discriminar de acuerdo con un tablero qué sonido debe ser activado en cada tiempo. Se crea una estructura con un toggle para cada combinación paso – voz y un bang para activar los sonidos en cada combinación paso-voz.En la figura se activa en el paso 1 sólo el sonido de la voz 1 (bombo), en el paso 2 el sonido de la voz 3 (Platillo), en el paso 3 sólo la voz uno, etc.

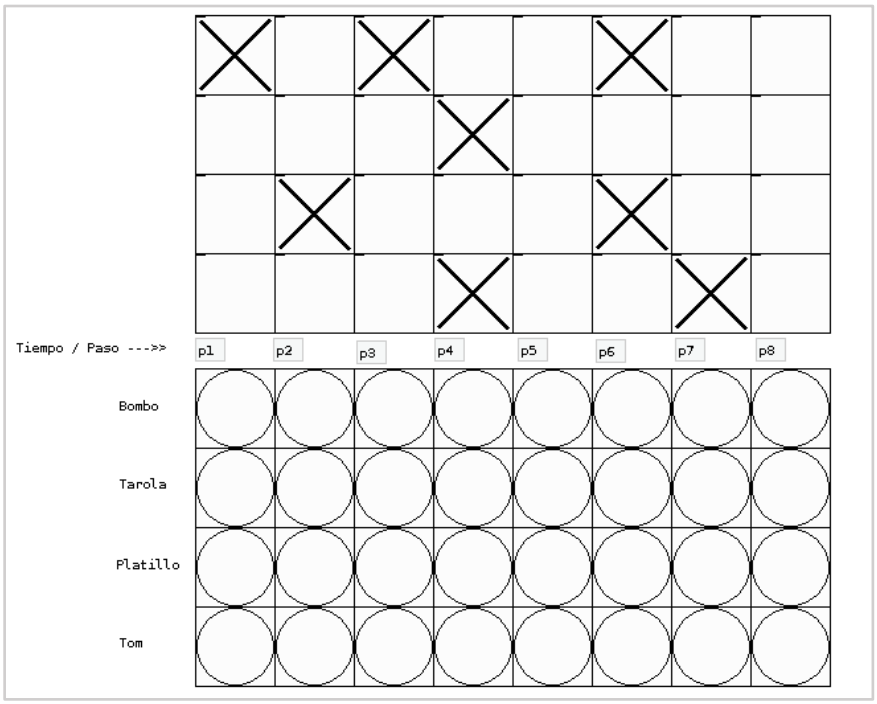


FIGURA 4.3.1TABLERO DE PASOS

Se asigna una variable de salida a cada [toggle] dando clic derecho en el objeto y después en “Properties”

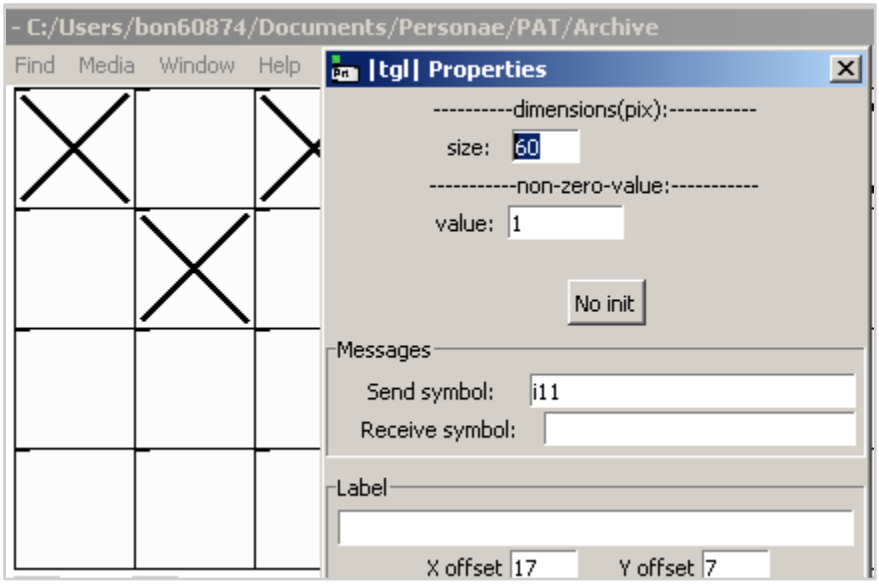


FIGURA 4.3.2CONFIGURACIÓN DE INTERRUPTOR

El objeto [gate] cierra o permite el flujo de un bang hacia su salida de acuerdo con un [toggle] en su entrada izquierda:

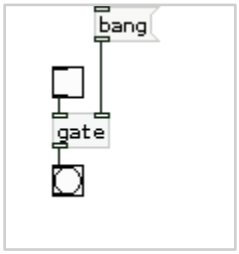


FIGURA 4.3.2OBJETO GATE

Se genera este control para cada voz y se define el patch paso_4_voces.Pd, con cuatro entradas (una para cada voz), una señal de reloj general (variable m_<número_de_paso>) y cuatro salidas. De esta forma se controlan las 4 salidas simultáneas de cada paso de acuerdo con un interruptor de cada voz.

A cada paso debe asignarse un control de encendido apagado para activar o no el sonido correspondiente a esa voz en un tiempo determinado. Para ello se programa el patch "paso_4_voces.Pd"

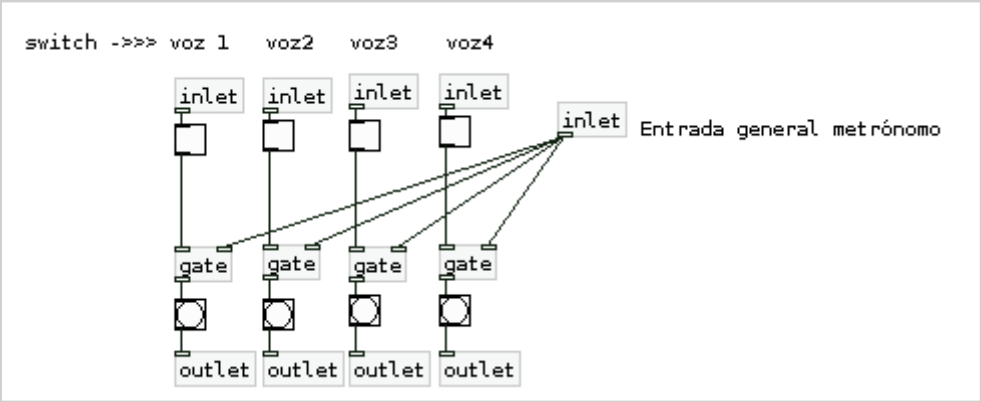


FIGURA 4.3.3CONTROL DE ENCENDIDO

El interruptor de cada voz se define como variable "i<paso><voz>", de esta forma i11 controla el [toggle] de la voz 1 en el paso 1. De manera similar se definen las variables "o<paso><voz>" que disparan los sonidos de cada voz en cada paso.

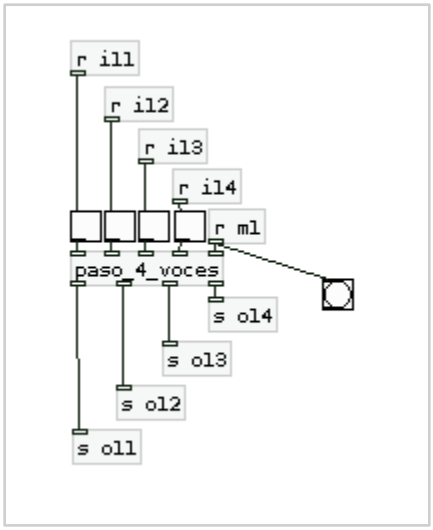


FIGURA 4.3.4VARIABLES PARA DISPARO DE SONIDOS

Se crea un patch para cada paso que agrupa los patches anteriores, la nomenclatura usada es “p<número_de_paso>” yson integrados al tablero:

FIGURA 4.3.5INICIALIZACIÓN DE PASOS

4.4 Banco de sonidos

Se direccionan todos los [bang] de cada voz a una variable común llamada “voz_<número_de_voz>”.:

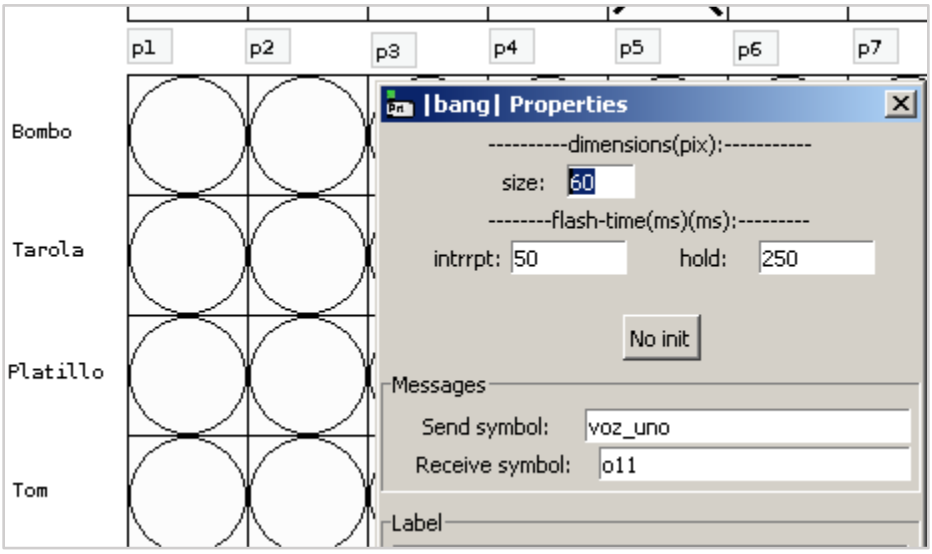


FIGURA 4.4.1CONFIGURACIÓN DE OBJETO BANG

Dicha variable activa la reproducción de un sonido:

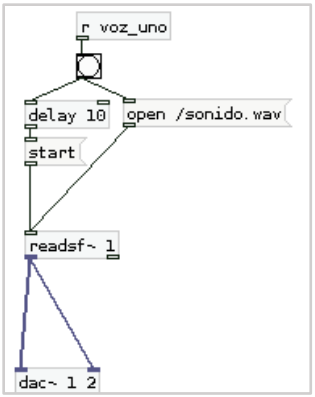


FIGURA 4.4.2ACTIVACIÓN DE SONIDO

Cada banco recibe la entrada de la voz correspondiente. Las salidas de cada banco de sonidos se agrupan con el objeto [+~]. De esta forma se unen las cuatro señales de audio en una señal común:

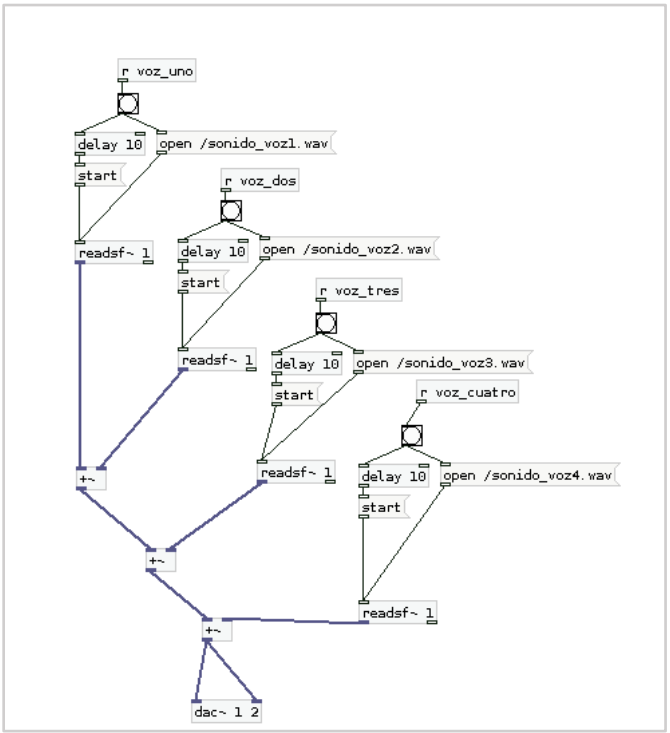


FIGURA 4.4.3UNIÓN DE SEÑALES DE AUDIO

Para poder cambiar rápidamente de banco sonoro en cada voz se desarrollaun sistema de casillas con el objeto HRadio de Pd. Se define una variable para cada voz que puede tomar un valor entre 0 y 6: b<número_de_voz>. Dicha variable es procesada en un parche para cada voz mediante el objeto [gate]. La selección introducida por el usuario condiciona el banco que será usado en la reproducción

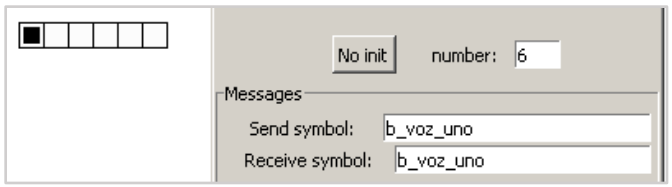


FIGURA 4.4.4 PROPIEDADESHRADIO

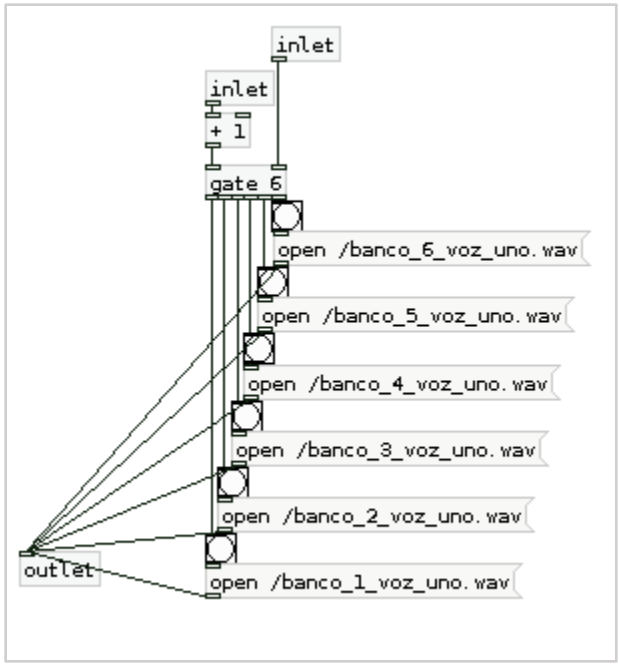


FIGURA 4.4.5BANCO VOZ UNO

4.5 Control de amplitud

El posible modificar la amplitud de las señales de audio usando el objeto[*~]. A la izquierdase conecta la señal a modificar y a la derecha un valor entre 0 y 1 con el que se multiplica. En la aplicación se definen variables de volumen para cada voz y una para el volumen general: “vv<número de voz>”, “master_vol”. Pd recibirá de Processing una variable asociada a dichos controles.

FIGURA 4.5.1MULTIPLICACIÓN DE SEÑALES

4.6 Selector de ritmos

Con el propósito de aumentar las capacidades de ejecución de la interfaz se agrega un módulo selector de ritmos. De esta forma es posible generar una serie de pulsos a partir del pulso que activa cada paso. El usuario puede así reproducir un sonido en patrones rítmicos afines a la notación musical tradicional, por ejemplo dos golpes por pulso, tres golpes (tresillo) o un golpe sincopado.

Se divide entre un número “n” el tiempo en [ms] entre cada pulso. Esa división de tiempo se usa como argumento a unaserie de retrasos (objetos [delay]). La selección del ritmo a ejecutarse se realiza con un sistema de casillas del mismo modo que la selección de bancos y el objeto [gate] activa la secuencia rítmica apropiada.

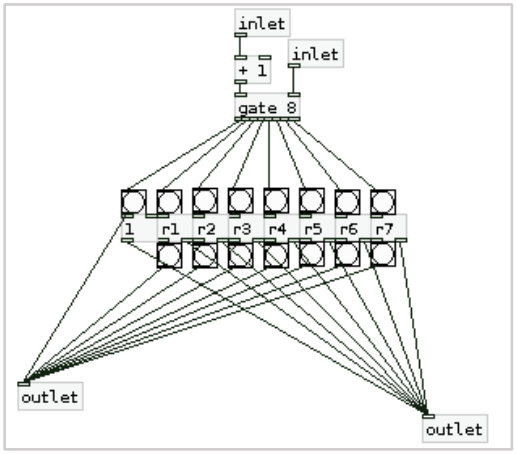


FIGURA 4.6.1RITMO.PD

Se asigna una amplitud a cada golpe dentro de la secuencia rítmica para proporcionar un control más parecido a la ejecución de un instrumento real, así en el primer golpe de la secuencia se reproduce con una amplitud total y en golpes sucesivos con una amplitud atenuada.

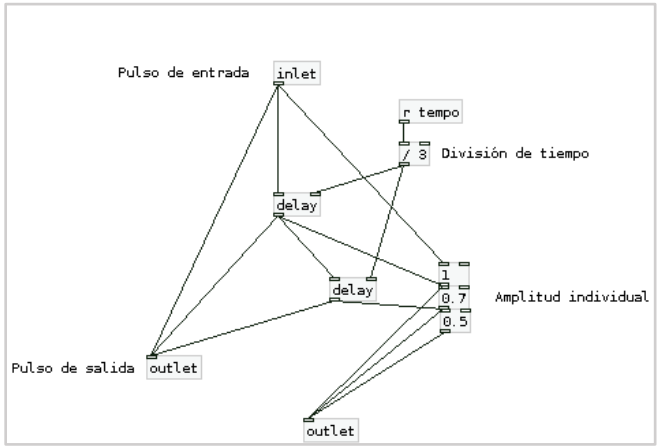


FIGURA 4.6.2RITMO 1 – TRESILLO

4.7 Envío de mensajes OSC a Processing

Para el envío de mensajes OSC en Pd se usa una librería llamada [mrpeach] en la cual debe abrirse un canal de comunicación indicando dirección IP y puerto destino. Posteriormente se envía el identificador y valor del mensaje con el objeto [sendOSC]. Pd envía el número de paso en el que se encuentra el metrónomo y Processing dibuja un círculo que cambia de coordenadascíclicamente de acuerdo con el valor, de esta forma se le proporciona al usuario unindicador de tiempo.

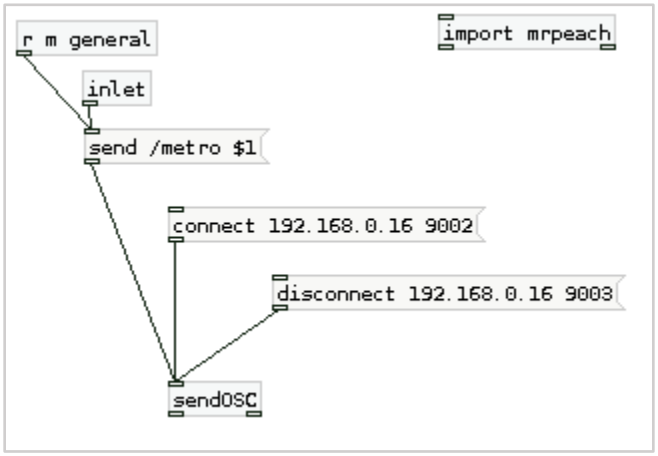


FIGURA 4.7.1ENVÍO DE MENSAJES OSC

5. Interfaz gráfica

5.1 Introducción

La interfaz gráfica permite al usuario tener una realimentación visual a partir de los puntos de contacto así como disparar las señales de audio en la aplicación. El programa enProcessing consta de una pantalla de bienvenida y un tablero de control. La interacción con Pd se realiza mediante mensajes OSCenviados por los marcadores visuales en cada paso del secuenciador.

Las partes que componen el código de la interfaz gráfica son:

- Definición de bibliotecas y variables globales
- Generación de gráficos
- Pantalla de bienvenida
- Tablero de control
- Objetostáctiles

5.2 Instalación y configuración

Se instala Processing versión 1.5.1 para asegurar la compatibilidad con el protocolo TUIO de Reactivision. Las bibliotecas necesarias son:

- OSCP5 - Permite comunicación Processing – Pd mediante protocolo OSC

- TUIOProcessing - Recibe los mensajes TUIO de Reactivision.

Se copian a la carpeta /Processing/sketchbook/libraries creada tras la instalación.

5.3 Definición de bibliotecas y variables globales

Al inicio del programa se indica qué bibliotecas son utilizadas. Posteriormente se enlistan los nombres de las variables así como el tipo de dato (cadena, entero, imagen, etc.). Para facilitar la legibilidad del código se crean arreglos multidimensionales que pueden crear y actualizar los objetos TUIO mediante un ciclo de control.

```
// Importación de bibliotecas
import TUIO.*;
import oscP5.*;
import netP5.*;

// Fuente mensajes TUIO
TuioProcessing tuioClient;

//Destino mensajes OSC
OscP5 oscP5;
NetAddress myRemoteLocation;

//Arreglo de objetos TUIO
BotonTUIO inicioTUIO, atrasTUIO, reset;
ToggleTUIO on_off;
ToggleTUIO[][] grid_paso = new ToggleTUIO[4][16];
V_Scrollbar[] v_barras = new V_Scrollbar[10];
RadioTUIO[][] ritmos = new RadioTUIO[4][8];
RadioTUIO[][] bancos = new RadioTUIO[4][5];

//Dimensiones del tablero
int l_t = 1200;  // Largo
int a_t = 900; // Alto


//Variables adicionales
PImage UNAM,FI, bkg;
PFont f,f2,f3;
int i,j, espacio;
int pantalla=0;
float cursor_size = 15;
float table_size = 760;
float scale_factor = 1;
float obx, oby;
```

FIGURA 5.3.1DEFINICIÓN DE BIBLIOTECAS Y VARIABLES

El método setup() realiza la inicialización de las variables con el constructor apropiado de cada objeto. Las dimensiones de cada botón, el nombre de las imágenes a importar así como el color y la posición se definen en este método.

Antes de inicializar una variable de texto es necesario crear la tipografía a partir de estilos predeterminados. Se realiza siguiendo la rutaProcessing - >Tools -> Create Font. Tras seleccionar un estilo, tipo de teclado y tamaño,se almacena en formato .vlw en la carpeta de desarrollo /<nombre>/data. Las imágenes almacenadas se colocan en la misma carpeta.
Se usan dos archivos en la pantalla de bienvenida.

En este método se define también la dirección de red a la que se envían los mensajes OSC. En el cliente OSC se abre un túnel UDP en el mismo puerto especificado en el servidor OSC como se muestra en la figura 5.3.2

```
void setup() {
  //Dimensiones del tablero
  size(l_t, a_t);
  //Clientes OSC & TUIO
  oscP5 = new OscP5(this,9001);
  myRemoteLocation = new NetAddress("192.168.0.8",9001);
  tuioClient  = new TuioProcessing(this);
```

FIGURA 5.3.2DEFINICIÓN DE IPY PUERTO DESTINO PARA MENSAJES OSC

Las variables TUIO definidas previamente en arreglos multidimensionales se inicializan en un ciclo para facilitar la legibilidad y modificación posterior. La posición de cada botón o barra de control se define en cada iteración y aumenta de acuerdo conlos valores de las variables “i y j”.

```
for (int i=0;i<4;i++) {
  for (int j=0;j<16;j++) {
  grid_paso[i][j]=new ToggleTUIO((j+1)*width/18 , (i+1)*height/16 + espacio, height/20, height/20,color(204),color(255,0,0),color(0));
    if(j<8){
    ritmos[i][j]=new RadioTUIO((j+1.2) * width/18,(i+2.2)*height/16 + espacio ,height/40,color(255), color(0),j,i, 8, ritmos);  }
    if(j>10){
    bancos[i][j-11]=new RadioTUIO((j-12+1.2) * width/18 + width/1.5 ,(i+2.2)*height/16 + espacio ,height/40,color(155), color(0),j-11,i, 5, bancos);}
  }           espacio = espacio + height /20; }
espacio = 0;
for (int i=0;i<8;i++) {
  //Controles volumen
  if(i<4){
  v_barras[i]=new V_Scrollbar((i+1)*width/25,6.5*height/12 + 30,height/40,height/4,1,12, color(0),color(155),color(255));
  v_barras[i].pos = 7*height/12;        }
  //Controles reverberación
  if(i>3){
  v_barras[i]=new V_Scrollbar((i+1)*width/25,6.5*height/12 + 30,height/40,height/4,1,12, color(155),color(255),color(0));
  v_barras[i].pos = 9.6*height/12;  }     }

//Controles de tempo
 v_barras[8]=new V_Scrollbar(0.85*width,6.5*height/12 + 30,height/40,height/4,1,12, color(155),color(255),color(0));
 v_barras[8].pos = 8*height/12;
 //Volumen general
 v_barras[9]=new V_Scrollbar(0.9*width,6.5*height/12 + 30,height/40,height/4,1,12, color(0),color(155),color(255));
 v_barras[9].pos = 7*height/12;
```

FIGURA 5.3.3INICIALIZACIÓN DE ARREGLO DE OBJETOS TUIO

5.4 Generación de gráficos

La generación de gráficos se realiza con el método **draw()** el cualse ejecuta iterativamente después de **setup**() hasta que el programa se detenga o hasta que se llame al método **noLoop()**. En la figura 5.4.1 la sentencia switchselecciona la rutina apropiada de acuerdo con la variable de control **pantalla**.

```
void draw() {
  background(204);
  stroke(255);
  switch(pantalla) {
  case 0:
    pantallaA();
    break;
  case 1:
    pantallaB();
    break;
  }
}
```

FIGURA 5.4.1MÉTODO DRAW()

El diagrama que describe la ejecución general del programa como se muestra en la figura 5.4.2

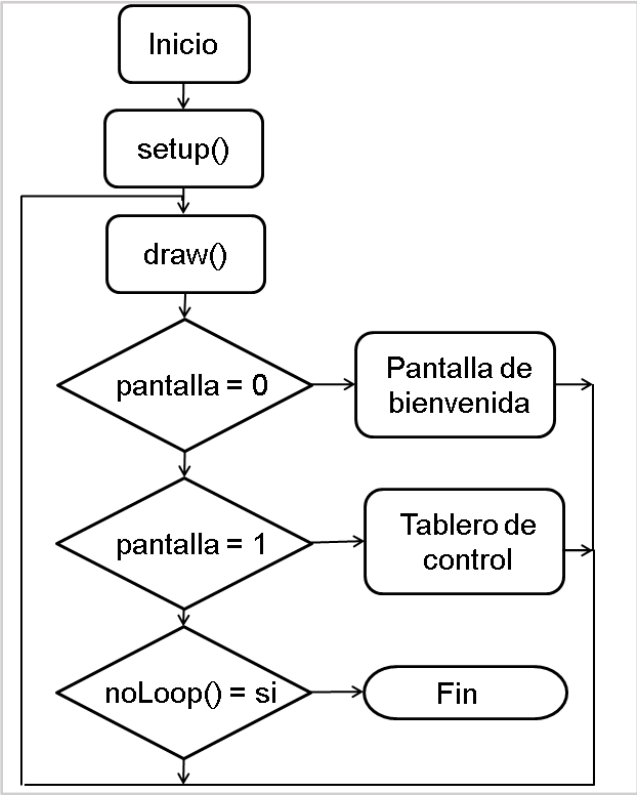


FIGURA 5.4.2DIAGRAMA GENERAL DE PROGRAMA

5.5 Pantalla de bienvenida

En la figura 5.5.1 se muestra la pantalla de bienvenida, mostrando los escudos de la UNAM y de la Facultad de Ingeniería, así como información del proyecto de tesisy un botón para comenzar con la aplicación.

FIGURA 5.5.1PANTALLA DE BIENVENIDA

Al iniciar esta pantalla se manda un mensaje OSC a Pd para detener la ejecución del metrónomo.Se inicializa el mensaje **mReset**, se agrega un valor 0 que interpreta Pd y por último se envía a la dirección de red definida en **setup().**

```
OscMessage mReset = new OscMessage("/reset");
mReset.add(0);
oscP5.send(mReset, myRemoteLocation);
```

FIGURA 5.5.2 MENSAJE DE INICIO A PD

La generación de cursores que reciben información de Reactivision se realiza con el método **dibujaEventosTUIO()**el cual registra los mensajes TUIO, extrae coordenadas y agrega cada objeto a una lista dinámica de objetos por dibujar. El dibujo yla actualización de los objetos se hace con el método de muestreo definido para cada uno.

La ejecución de la pantalla de bienvenida se muestra en la figura 5.5.3

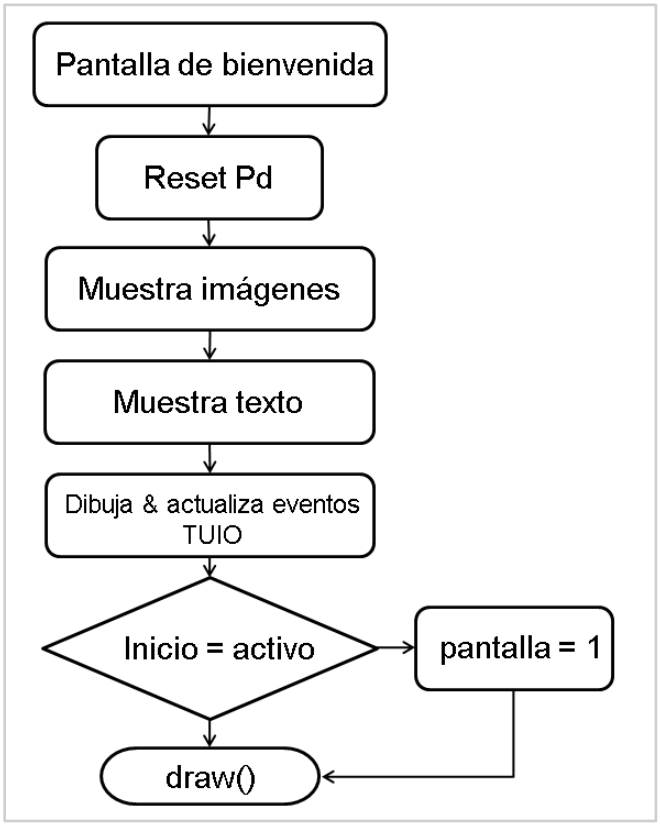


FIGURA 5.5.3 DIAGRAMA DE PANTALLA DE BIENVENIDA

El código del método de la pantalla se describe en la figura 5.5.4

```
void pantallaA() {
  OscMessage mReset = new OscMessage("/reset"); //Pone en 0 el arreglo en PUREdata
  mReset.add(0);
  oscP5.send(mReset, myRemoteLocation);

  f = loadFont("BrowalliaNew-48.vlw");
  f2 = loadFont("BrowalliaNew-Bold-48.vlw");
  background(255);
  image(UNAM, width/30, height*0.03);
  image(FI, width - width/5, height*0.03);
  textFont(f);
  textAlign(LEFT);
  fill(1);
  textAlign(CENTER, TOP);
  text("Facultad de Ingeniería",width/2,height*0.45);
  text("Desarrollo de interfaz multi-táctil mediante el fenómeno de FTIR", width/2, height*0.55);
  textFont(f2);
  text("Universidad Nacional Autónoma de México",width/2, height*0.4);
  text("Iniciar",width - width/6.8, height*0.858);

  atrasTUIO.pressed=false;
  inicioTUIO.pressed=false;

  fill(obx*25,oby*25,0);
  dibujaEventosTUIO();
  inicioTUIO.display();
  inicioTUIO.update();

  if (inicioTUIO.pressed==true && inicioTUIO.over==true){
  atrasTUIO.over=atrasTUIO.pressed=false;
  tiempos[0]=0;
  pantalla=1;
  reset();
  }
}
```

FIGURA 5.5.4 CÓDIGO DE PANTALLA DE BIENVENIDA

5.6 Tablero de control

En la figura 5.6.1 se encuentra el tablero que contiene los botones para activar cada paso,controles de volumen, reverberación & tempo. También se define un botón que puede activar o desactivar el metrónomo y uno para restablecer todos los controles a valores por defecto.

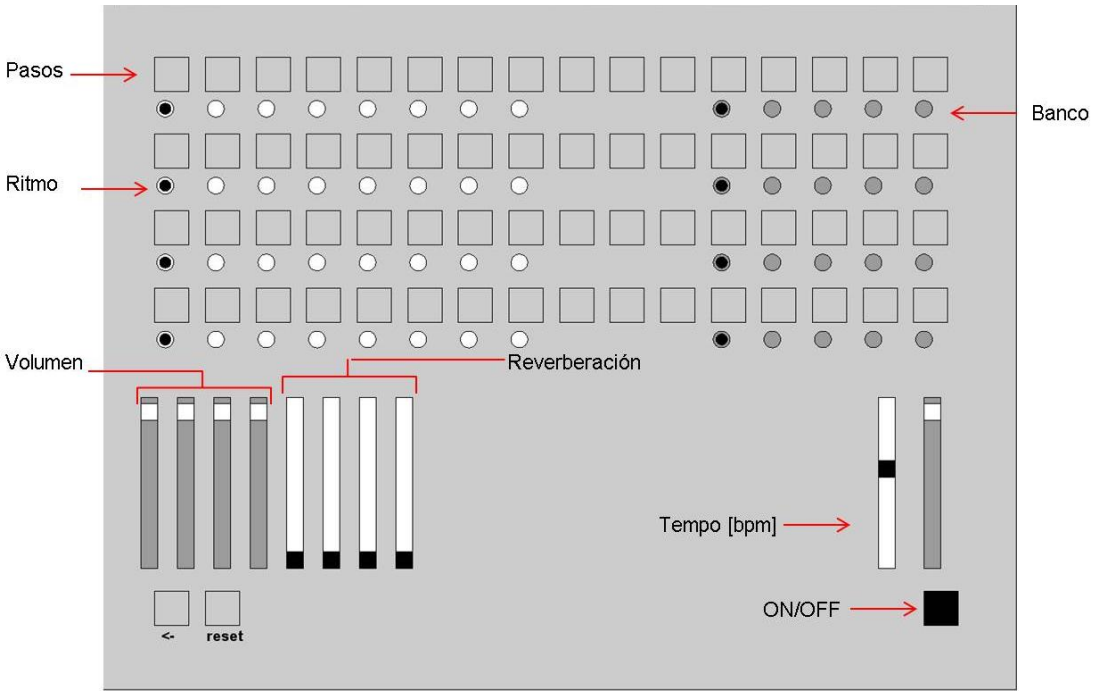


FIGURA 5.6.1 TABLERO DE CONTROL

De manera similar a la pantalla de bienvenida se crean los mensajes OSC necesarios como se muestra en la figura 5.6.2.

```
void pantallaB() {
  String send_s;
  f3 = loadFont("BrowalliaNew-Bold-30.vlw");
  OscMessage mReset = new OscMessage("/reset");
  OscMessage o_toggle = new OscMessage("/toggle");
```

FIGURA 5.6.2 CREACIÓN DE MENSAJES OSC

La cadena “/reset” y “/toogle” permiten identificar los mensajes en el cliente OSC.

La supervisión de los botones y pasos se hacen con un ciclo similar al que las variables fueron inicializadas. Esto facilita la creación de mensajes OSC ya que simplemente se indica el número de columna y fila de cada arreglo multidimensional para ser procesado por Pd. En la figura 5.6.3 describe elsalto del ciclo y la definición de los mensajes OSC.

```
for (int i=0;i<4;i++) {
     for (int j=0;j<16;j++) {
       grid_paso[i][j].display();
       grid_paso[i][j].update();
         if( grid_paso[i][j].over==true) {
         o_toggle.add(i);
         o_toggle.add(j);
         o_toggle.add(grid_paso[i][j].on);
         oscP5.send(o_toggle, myRemoteLocation);
         //tiempos[0]=0;
         }
     if(j<8){
       ritmos[i][j].displayR();
       ritmos[i][j].press(obx, oby);      }

      if(j>10){
       bancos[i][j-11].displayR();
       bancos[i][j-11].press(obx, oby);      }

 }        }

for (int i=0;i<10;i++) {
 v_barras[i].update();
 v_barras[i].display();}
```

FIGURA 5.6.3 BARRIDO DE ARREGLO DE OBJETOS TUIO

El método **reset**() es llamado por el usuario para hacer un barrido de todas las variables y asignar un valor por defecto. El método también se llama antes de regresar a la pantalla de bienvenida. Las acciones de disparadas por ese método se muestran en la figura 5.6.4.

```
void reset(){

   for (int i=0;i<4;i++) {
        ritmos[i][0].checked = true;
        bancos[i][0].checked = true;

        for (int j=0;j<16;j++) {
          grid_paso[i][j].on = false;
          if(j<8 && j !=0){
          ritmos[i][j].checked = false;
          }

          if(j>10){
          bancos[i][j-11].checked = false;
          }
      bancos[i][0].checked = true;
  }}


     for (int i=0;i<8;i++) {
      //volumen
      if(i<4){
      v_barras[i].pos = 7*height/12;      }
      //reverb
      if(i>3){
      v_barras[i].pos = 9.6*height/12;}   }

    //tempo
     v_barras[8].pos = 8*height/12;
     //master
     v_barras[9].pos = 7*height/12;
}
```

FIGURA 5.6.4 MÉTODO RESET

La actualización de los objetos TUIO se hace con el método **update**() definido individualmente en cada clase y se llama dentro de ambas pantallas para los elementos de interacción.La ejecución del programa para el tablero de control se describe en el figura 5.6.5

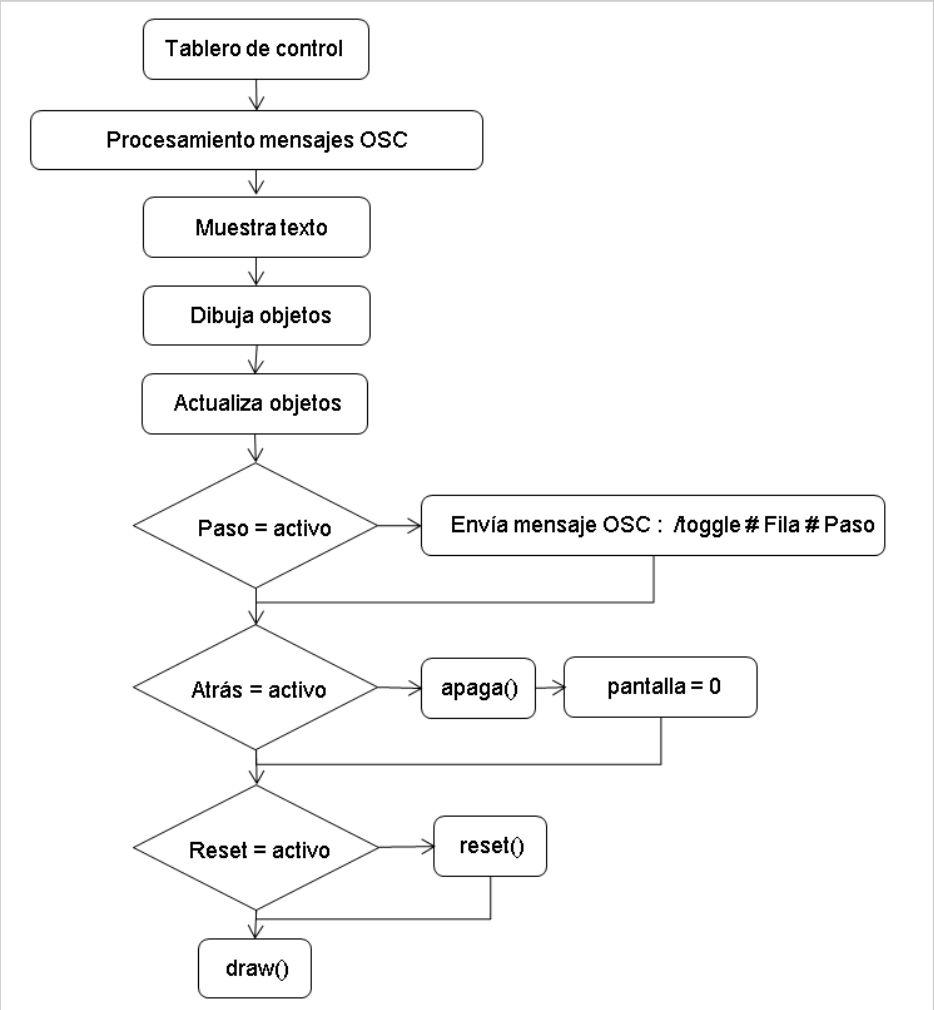


FIGURA 5.6.5 DIAGRAMA DE TABLERO DE CONTROL

5.7 Objetos táctiles

Al final del programa se definenlas clases y objetos usados en la interacción táctil: cursor, botón, scroll bar, etc).Los objetos RadioButtonTUIO y el ScrollBar incluyen un envío de mensajes OSC a Pd para informar sobre cambios en su valor como se muestra en lassiguientes figuras.

```
boolean press(float mx, float my, String message) {

   OscMessage o_ritmo = new OscMessage(message);

   if (dist(x, y, mx, my) < size/2) {
   checked = true;

   o_ritmo.add(my_row);
   o_ritmo.add(me);

   oscP5.send(o_ritmo, myRemoteLocation);

   for (j=0; j < cols; j++){
       if (j != me) {
       others[my_row][j].checked = false; }  }

   return true;
   } else {
   return false;
   }
   }
```

FIGURA 5.7.1 MÉTODO DE ACTUALIZACIÓN DE BOTÓN

```
class BotonTUIO {
  float x, y;
  float size;
  color baseGray;
  color overGray;
  color pressGray;
  boolean over = false;
  boolean pressed = false;

  BotonTUIO(float xp, float yp, float s, color b, color o, color p) {
    x = xp;
    y = yp;
    size = s;
    baseGray = b;
    overGray = o;
    pressGray = p;  }

  void update() {
    if ((obx >= x) && (obx <= x+size) &&
      (oby >= y) && (oby <= y+size)) {
      over = true;
    }
    else {
      over = false;
    }
    if(tiempos[2]<500 && over==true) {
    pressed=true;} }

  void display() {
    if (pressed == true) {
      fill(pressGray);
    }
    else if (over == true) {
      fill(overGray);
    }
    else {
      fill(baseGray);
    }
    stroke(1);
    rect(x, y, size, size);
  }

}
```

FIGURA 5.7.2 BOTÓN TUIO

```
class ToggleTUIO {
  float x, y, alto_b, ancho_b;
  color baseGray;
  color overGray;
  color pressGray;
  boolean over = false;
  boolean on = false;

  ToggleTUIO(float xp, float yp, float ancho, float alto, color b, color o, color p) {
    x = xp;
    y = yp;
    alto_b = alto;
    ancho_b= ancho;
    baseGray = b;
    overGray = o;
    pressGray = p;
    }

  void update() {
    if ((obx >= x) && (obx <= x + ancho_b) &&
      (oby >= y) && (oby <= y + alto_b)) {
      over = true;
      if(tiempos[2]>100 && tiempos[2]<1000) {
        on= !on;}   }
    else {
      over = false;  }
   }

  void display() {
    if (on == true) {
      fill(pressGray);
    }
    else if (over == true) {
      fill(overGray);
    }
    else {
      fill(baseGray);
    }
    stroke(0);
    rect(x, y, ancho_b, alto_b);  }

}
```

FIGURA 5.7.3 INTERRUPTOR TUIO

```
class RadioTUIO{

 float x, y;
 float size, dotSize;
 color baseGray, dotGray;
 boolean checked = false;
 int me,cols, my_row;
 RadioTUIO[][] others;

 RadioTUIO(float xp, float yp,float s,color b,color d,int m,int mr,int columns,RadioTUIO[][] o){
 x = xp;
 y = yp;
 size = s;
 dotSize = size - size/3;
 baseGray = b;
 dotGray = d;
 others = o;
 cols=columns;
 my_row = mr;
 me = m;
 }


 boolean press(float mx, float my) {
   if (dist(x, y, mx, my) < size/2) {
   checked = true;
     for (j=0; j < cols; j++){
       if (j != me) {
       others[my_row][j].checked = false; }     }
   return true;
   } else {
   return false;    }  }

void displayR() {
   stroke(0);
   fill(baseGray);
   ellipse(x, y, size, size);
   if (checked == true) {
     fill(dotGray);
     ellipse(x, y, dotSize, dotSize); }  }
 }
```

FIGURA 5.7.4 RADIO TUIO

```
class V_Scrollbar {
  float x, y; // The x- and y-coordinates
  float sw, sh; // Width and height of V_Scrollbar
  float pos; // Position of thumb
  float posMin, posMax; // Max and min values of thumb
  boolean rollover; // True when the mouse is over
  boolean locked; // True when its the active V_Scrollbar
  float minVal, maxVal; // Min and max values for the thumb
  color fondo;
  color marker;
  color hold;


  V_Scrollbar (float xp, float yp, float w, float h, float miv, float mav,color f, color m, color hl) {
    x = xp;
    y = yp;
    sw = w;
    sh = h;
    minVal = miv;
    maxVal = mav;
    pos = y + sh/2 - sw/2;
    posMin = y;
    posMax = y + sh - sw;
    fondo = f;
    marker = m;
    hold = hl;
  }

  void press(int mx, int my) {
    if (rollover == true) {
      locked = true;
    }
    else {
      locked = false;
    }}

  // Resets the V_Scrollbar to neutral
  void release() {
    locked = false;
  }

  // Returns true if the cursor is over the V_Scrollbar
  boolean over(float mx, float my) {
    if ((mx > x) && (mx < x+sw) && (my > y) && (my < y+sh)) {
      return true;
    }
    else {
      return false;}  }
```

FIGURA 5.7.5 BARRA VERTICAL

El envío de los mensajes OSC con la posición del cursor en el ScrollBar se realiza en el método update() para enviar mensajes sólo cuando se detecte un cambio en la posición. El identificador “/scroll” es detectado por Pd y direccionado a las variables de audio correspondientes.

```
  void update(int slider ) {
  if (over(obx, oby) == true) {
    rollover = true;
    pos = constrain(oby-sw/2, posMin, posMax);
    OscMessage o_scroll = new OscMessage("/scroll");
    o_scroll.add(slider);
    o_scroll.add(pos-posMax);
    oscP5.send(o_scroll, myRemoteLocation);
    println(pos);
  }
  else {
    rollover = false;
  }
}
```

FIGURA 5.7.6 MENSAJE OSC EN BARRA VERTICAL

5.8 Mensajes de control de audio.

Los mensajes OSC se reciben por Pd mediante el objeto [udpreceive] el cual abre un canal de comunicación en el mismo puerto en el que se configuró el envío en Processing. Es necesario desempacar cada mensaje y separarlo de acuerdo con la etiqueta creada antes de su envío (/scroll, /toggle, /reset, etc). Los objetos [unpackOSC] y [routeOSC] permiten manipular los valores que contenidos en cada mensaje y direccionar a los objetos de audio apropiados.

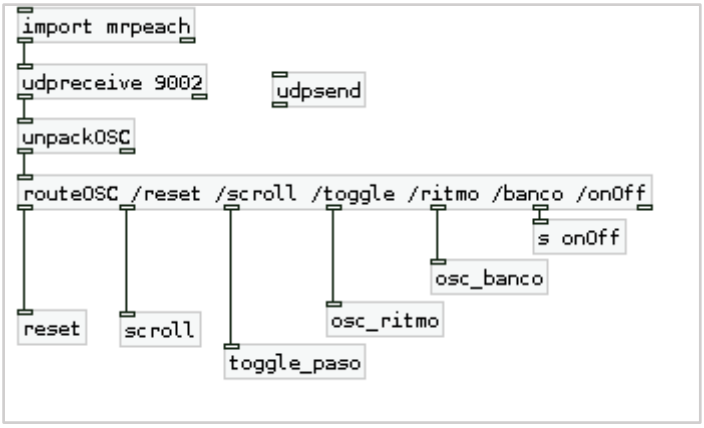


FIGURA 5.8.1RECEPCIÓN DE MENSAJES OSC EN PD

Para recibir mensajes OSC en Processing es necesario abrir un canal de comunicación y definir un método para interpretar los mismos. Se recibe de Pd el número de paso y se asigna ese valor a una variable que permite dibujar un círculo en una posición distinta en cada paso. De esta forma el indicador de paso recorre la interfaz de acuerdo con el metrónomo en Pd.

```
void oscEvent(OscMessage theOscMessage)
{
  if(theOscMessage.addrPattern().equals("/metro") &&  // Evaluación de patrón y tipo de dato del mensaje OSC
  theOscMessage.typetag().equals("i")) {
     int a = theOscMessage.get(0).intValue();
     metro_x=a; // Variable global para posición en X de indicador de paso
  }
}
```

FIGURA 5.8.2RECEPCIÓN DE MENSAJES OSC EN PROCESSING

5.9 Simulaciónde interfaz

Para depurar el código en Processing y visualizar de manera rápida el control de las señales de audio sin tener en consideración el hardware y la pantalla ensamblada, se hace uso de la herramienta TUIOSimulator.El mouse toma el papel del cursor y se generan mensajes TUIO como si Reactivision estuviera vigilando la pantalla en un ambiente real. De esta forma el desarrollo es más ágil.

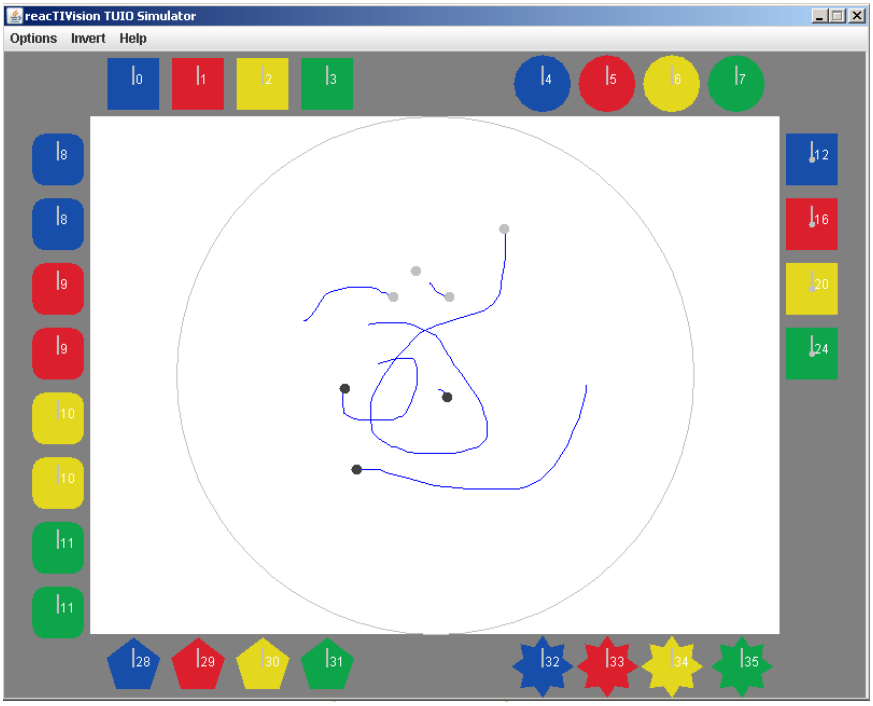


FIGURA 5.9.1SIMULADOR DE MENSAJES TUIO

6. Conclusiones

En el presente trabajo se logró la construcción de un sistema de interacciónmulti-táctil mediante una pantalla iluminada por el principio de Reflexión Interna Total Frustrada. La aplicación desarrollada es un secuenciador para propósitos musicales. Diversos bancos sonoros y ritmos pueden programarse lo que hace que las funcionalidades sean apropiadas para unentorno de ejecución sonora.

El costo de una pantalla táctil comercial de 60 pulgadas es mucho mayor a la suma de los costos de los materiales empleados en este proyecto, aproximadamente el doble de acuerdo el análisis presentado en el anexo de la tesis. La escalabilidad que puede lograrse con la tecnología tratada podría hacer que esa diferencia de costo sea mucho mayor, teniendo en cuenta que los fabricantes de pantalla apuestan por dispositivos portátiles y no de gran escala.

Desde el punto de vista del software, el código abierto desarrollado permite ser compartido y reusado lo queabre un camino para desarrollos en otras disciplinas afines a la rama de HCI:mostradores, procesamiento de imágenes, etc. En este sentido la interfaz ofrece flexibilidad para investigaciones y futuros prototipos. La capacidad de enviar mensajes MIDI desde Pd permite también la integración con plataformas de audio especializadas.

Las mayores dificultades presentadas fueron causadas por el hardware y las condiciones de iluminación. Para una respuesta óptima se requiere de un ambiente libre deruido lumínico que no interfiera el sistema de detección de puntos táctiles.En otras palabras es necesario instalar la solución en un recinto apropiado para la ejecución lo que dificulta que el sistema sea robusto y portable. La alineación precisa de los LED´s y la deformación que provoca el peso de la placa también afectaron la iluminación uniforme.

Adicionalmente es necesario realizar una rutina de calibración para ajustar contraste, tamaño de los cursores yhacer un mapeo con el área de contacto a los puntos de captura del software de CV.Una mejora considerable vendría al aumentar la resolución de la adquisición utilizando un arreglo de cámaras en paralelo y combinar la imagen mediante SW lo que sale de los alcances del proyecto. El uso de una cámara explícitamente desarrollada para infrarrojo y comunicada mediante FireWire en vez de USB también podría mejorar el tiempo de respuesta de la detección de puntos.

7. Anexos

7.1 Comparativa de costos

Se presenta el siguiente reporte de costos para comparar las interfaces táctiles en el mercado con el costo de los materiales usados en el proyecto. Los datos fueron obtenidos de los sitios web de los fabricantes.

Modelo	Pulgadas	Costo aproximado (MXP)
NEC P463 LCD Touchscreen Display with 3M DST Touch Scree	46	41075
NEC V463 LCD Touch Screen Monitor	46	37033
Sharp PN-L603B AQUOS Touch Screen LED Display	60	62937
Sharp PN-L602B1080p 60Hz LED TV	60	44930
Sharp -LCD Touchscreen Monitor	60	49489
Elo 7001LDigital Signage Multi-Touch Display ET7001L	70	77102
AQUOS BOARD LED Interactive Display System Sharp PN-L802B	80	139112

Materiales en el proyecto

Proyector	6000
Cámara	400
Pantalla de acrílico + Materiales de iluminación	5000
Computadora	14000

Total	25400

7.2 Programa en PureData

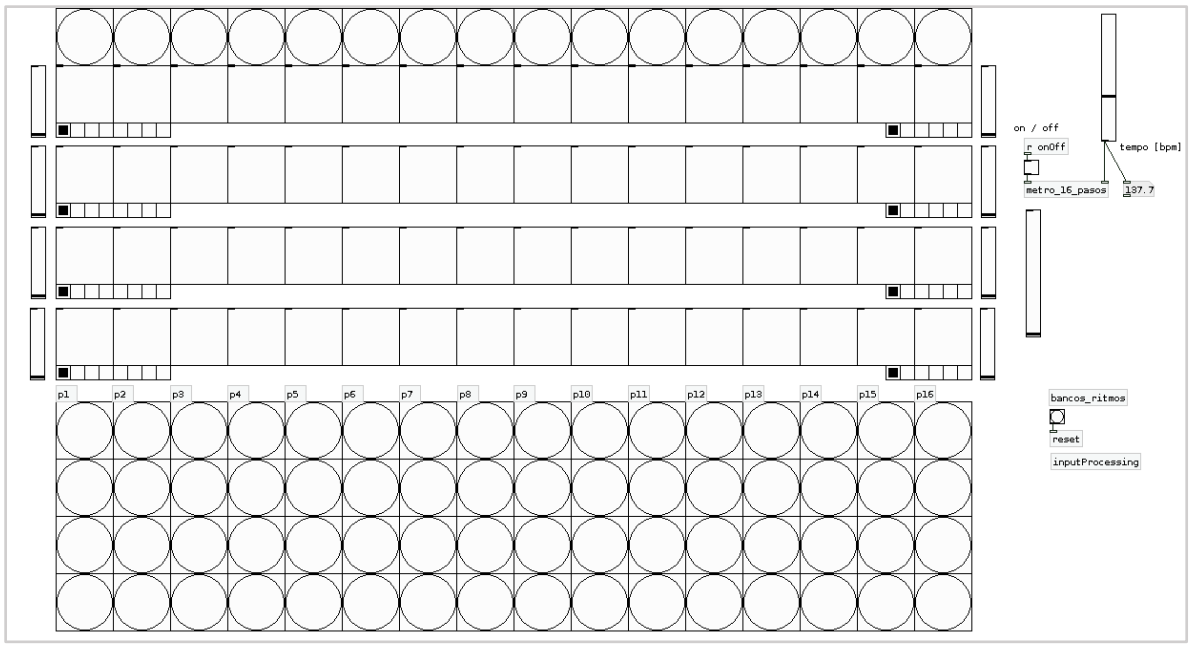


FIGURA 7.2.1TABLERO.PD

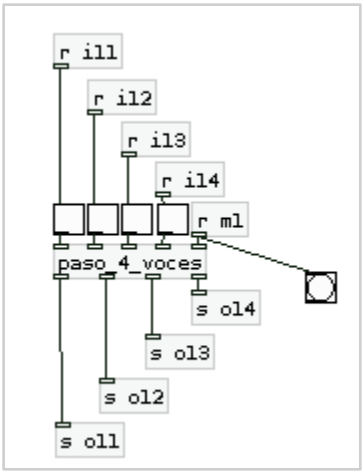


FIGURA 7.2.2P1.PD

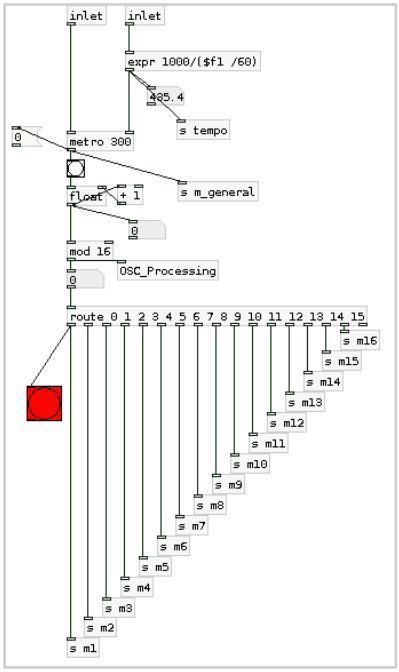


FIGURA 7.2.3METRO_16_PASOS.PD

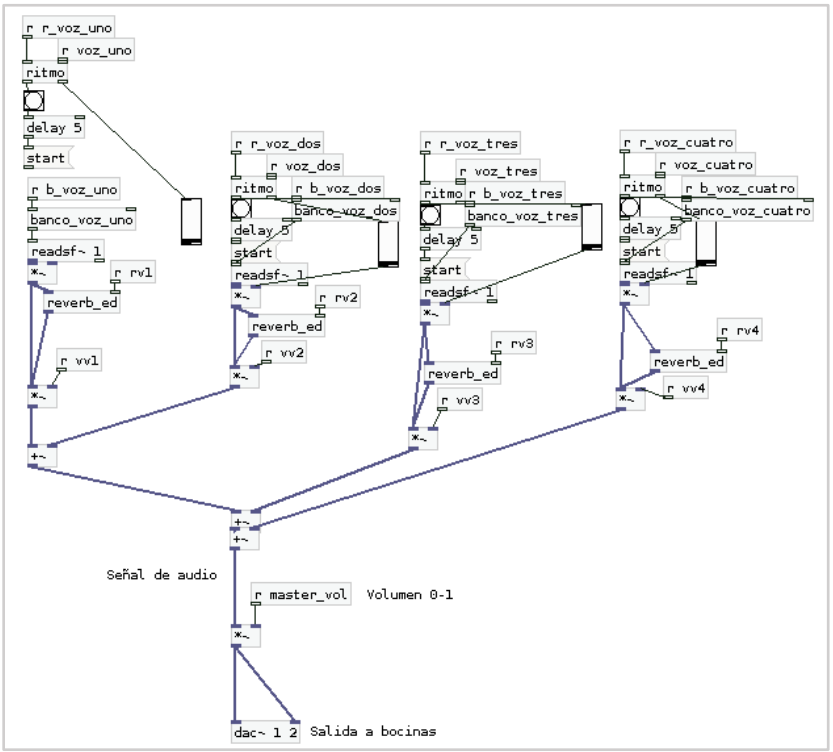


FIGURA 7.2.4BANCOS_RITMOS.PD

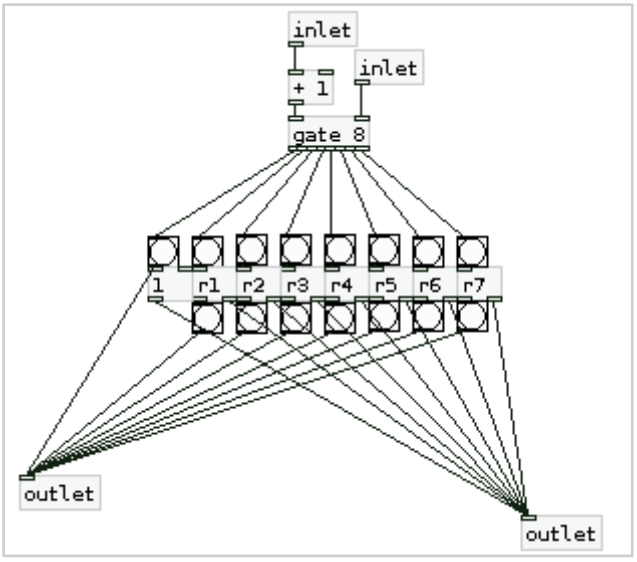


FIGURA 7.2.5RITMO.PD

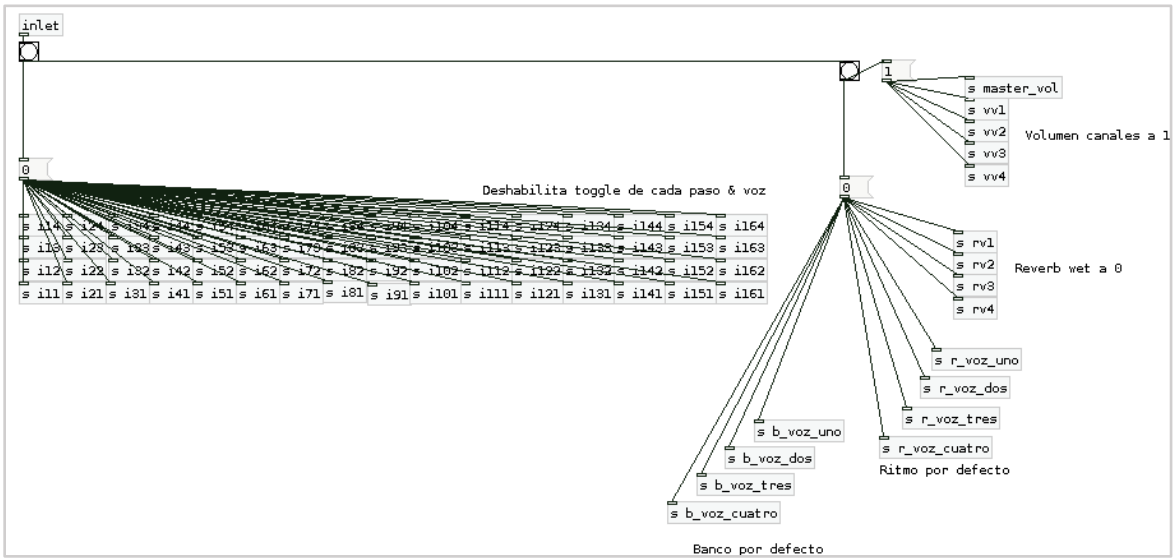


FIGURA 7.2.6RESET.PD

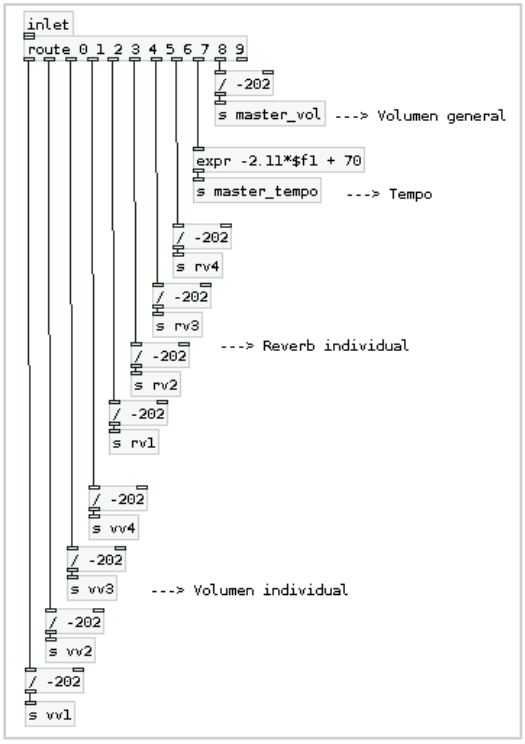


FIGURA 7.2.7SCROLL.PD

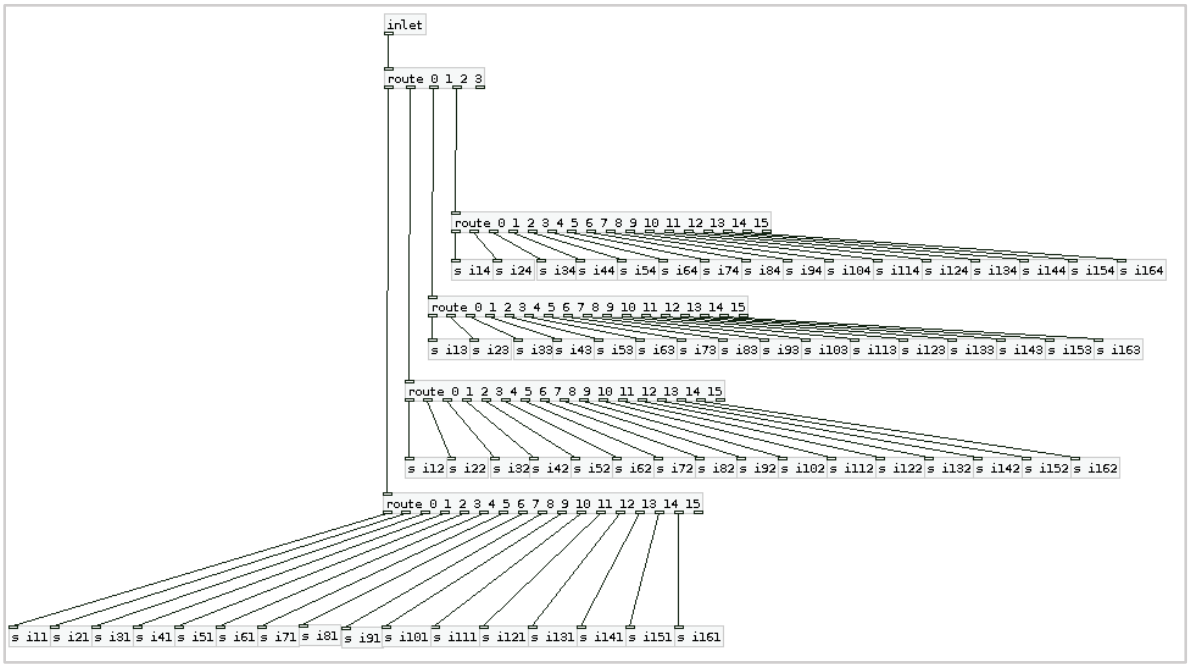


FIGURA 7.2.8TOGGLE_PASO.PD

7.3 Programa en Processing

```
import TUIO.*;
import oscP5.*;
import netP5.*;

TuioProcessing tuioClient;
OscP5 oscP5;
NetAddressmyRemoteLocation;
BotonTUIOinicioTUIO, atrasTUIO, reset;
ToggleTUIOon_off;
ToggleTUIO[][] grid_paso = new ToggleTUIO[4][16];
V_Scrollbar[] v_barras = new V_Scrollbar[10];
RadioTUIO[][] ritmos = new RadioTUIO[4][8];
RadioTUIO[][] bancos = new RadioTUIO[4][5];
intl_t = 1200;
inta_t = 900;
intmetro_x;
inti,j, espacio;
intpantalla=0;
float cursor_size = 15;
float table_size = 760;
float scale_factor = 1;
float obx, oby;
long[] tiempos = newlong[3];
PImage UNAM,FI, bkg;
PFont f,f2,f3;

void setup() {
size(l_t, a_t);
oscP5 = new OscP5(this,9002);
myRemoteLocation = new NetAddress("192.168.0.2",9002);
tiempos[0]=0;
tuioClient= new TuioProcessing(this);
UNAM = loadImage("UNAM.jpg" );
FI = loadImage("FI.jpg" );
bkg = loadImage("IIMAS.jpg");
espacio = height /65;
scale_factor = height/table_size;
inicioTUIO = new BotonTUIO(width - width/10, height*0.858, height/20,color(204), color(255,0,0), color(0));
atrasTUIO = new BotonTUIO(width/18, height*0.858, height/20,color(204), color(255,0,0), color(0));
reset = new BotonTUIO(2*width/18, height*0.858, height/20,color(204), color(255,0,0), color(0));
on_off= new ToggleTUIO(3*width/18 , height*0.858 , height/20, height/20,color(204),color(255,0,0),color(0));

for (int i=0;i<4;i++) {
for (int j=0;j<16;j++) {
grid_paso[i][j]=new      ToggleTUIO((j+1)*width/18      ,      (i+1)*height/16      +      espacio,      height/20,
height/20,color(204),color(255,0,0),color(0));
if(j<8){
ritmos[i][j]=new  RadioTUIO((j+1.2)  *  width/18,(i+2.2)*height/16  +  espacio  ,height/40,color(255),  color(0),j,i,  8,
ritmos);
```

```
}
if(j>10){
bancos[i][j-11]=new RadioTUIO((j-12+1.2) * width/18 + width/1.5 ,(i+2.2)*height/16 + espacio ,height/40,color(155),
color(0),j-11,i, 5, bancos);
}
}espacio = espacio + height /20; }
espacio = 0;

for (int i=0;i<8;i++) {
//volumen
if(i<4){
v_barras[i]=new V_Scrollbar((i+1)*width/25 + 20,6.5*height/12 + 30,height/40,height/4,1,12,
color(0),color(155),color(255));
v_barras[i].pos = 7*height/12;}
//reverb
if(i>3){
v_barras[i]=new V_Scrollbar((i+1)*width/25 + 20,6.5*height/12 + 30,height/40,height/4,1,12,
color(155),color(255),color(0));
v_barras[i].pos = 9.6*height/12; } }
        //tempo
v_barras[8]=new V_Scrollbar(0.84*width,6.5*height/12 + 30,height/40,height/4,1,12, color(155),color(255),color(0));
v_barras[8].pos = 8*height/12;
 //master
v_barras[9]=new V_Scrollbar(0.92*width,6.5*height/12 + 30,height/40,height/4,1,12, color(0),color(155),color(255));
v_barras[9].pos = 7*height/12;
}

void draw() {
background(204);
stroke(255);
switch(pantalla) {
case 0:
pantallaA();
break;
case 1:
pantallaB();
break;}}

///////////////Metodo para dibujar eventos y trayectoria del cursor
void dibujaEventosTUIO() {
Vector tuioCursorList = tuioClient.getTuioCursors();
for (int i=0;i<tuioCursorList.size();i++) {
TuioCursortcur = (TuioCursor)tuioCursorList.elementAt(i);
Vector pointList = tcur.getPath();
stroke(192,192,192);
fill(tcur.getScreenX(width)/4,tcur.getScreenY(height)/4, tcur.getScreenX(width)/50);
ellipse(tcur.getScreenX(width), tcur.getScreenY(height),width/50 + tcur.getScreenX(width) / 50,width/50 +
tcur.getScreenY(width)/50);
textAlign(TOP, LEFT);
text("X = " + tcur.getScreenX(width),width - width/5, height*0.05);
text("Y = " + tcur.getScreenY(height),width - width/5, height*0.1);
```

```
fill(0);}}

//////////////Metodos TUIO
voidaddTuioCursor(TuioCursortcur) {
tiempos[0]=tiempos[1];
tiempos[1]=System.currentTimeMillis();
tiempos[2]=tiempos[1]-tiempos[0];
obx=tcur.getX()*width;
oby=tcur.getY()*height;}

voidupdateTuioCursor (TuioCursortcur) {
obx=tcur.getX()*width;
oby=tcur.getY()*height;}

voidremoveTuioCursor(TuioCursortcur) {
println("remove cursor "+tcur.getCursorID()+" ("+tcur.getSessionID()+")");
obx=oby=0;}

void refresh(TuioTimebundleTime) {
redraw();}

void pantallaA() {
OscMessagemReset = new OscMessage("/reset");
mReset.add(0);
oscP5.send(mReset, myRemoteLocation);
f = loadFont("BrowalliaNew-48.vlw");
f2 = loadFont("BrowalliaNew-Bold-48.vlw");
background(255);
image(UNAM, width/30, height*0.03);
image(FI, width - width/5, height*0.03);
textFont(f);
textAlign(LEFT);
fill(1);
textAlign(CENTER, TOP);
text("Facultad de Ingeniería",width/2,height*0.45);
text("Desarrollo de interfaz multi-táctil mediante el fenómeno de FTIR", width/2, height*0.55);
textFont(f2);
text("Universidad Nacional Autónoma de México",width/2, height*0.4);
text("Iniciar",width - width/6.8, height*0.858);
atrasTUIO.pressed=false;
inicioTUIO.pressed=false;
fill(obx*25,oby*25,0);
dibujaEventosTUIO();
inicioTUIO.display();
inicioTUIO.update();

if (inicioTUIO.pressed==true &&inicioTUIO.over==true){
atrasTUIO.over=atrasTUIO.pressed=false;
tiempos[0]=0;
pantalla=1;
reset();}
```

```
}


void pantallaB() {
String send_s, mensaje;
f3 = loadFont("BrowalliaNew-Bold-30.vlw");
OscMessagemReset = new OscMessage("/reset");
OscMessageo_toggle = new OscMessage("/toggle");
OscMessageonOff = new OscMessage("/onOff");
stroke(0);
fill(0,255,0);
ellipse((metro_x+1)*width/18, height/16, 20, 20);
textFont(f3);
textAlign(LEFT);
fill(1);
textAlign(LEFT, BOTTOM);
text(" <---",width/18, height*0.94);
text("reset",2*width/18, height*0.94);
text("onOff",3*width/18, height*0.94);
text("tempo",0.84*width, height*0.87);
text("m_vol",16.5*width/18, height*0.87);
text("vol",width/18, 6.8*height/12);
text("rev",4*width/18, 6.8*height/12);
dibujaEventosTUIO();
atrasTUIO.display();
atrasTUIO.update();
reset.display();
reset.update();
on_off.display();
on_off.update();

if (atrasTUIO.pressed==true &&atrasTUIO.over==true){
inicioTUIO.over=inicioTUIO.pressed=false;
tiempos[0]=0;
pantalla=0;
reset();
onOff.add(0);
oscP5.send(onOff, myRemoteLocation); }

//RESET
if (reset.pressed==true &&reset.over==true){
reset();
mReset.add(1);
oscP5.send(mReset, myRemoteLocation);
reset.pressed=false;}
textFont(f);
fill(0);
textAlign(RIGHT);

// TUIO - Button - Grid /RadioTUIO
 for (int i=0;i<4;i++) {
```

```
for (int j=0;j<16;j++) {
grid_paso[i][j].display();
grid_paso[i][j].update();
if(grid_paso[i][j].over==true) {
o_toggle.add(i);
o_toggle.add(j);
o_toggle.add(grid_paso[i][j].on);
oscP5.send(o_toggle, myRemoteLocation);
print(o_toggle);}
if(j<8){
ritmos[i][j].displayR();
mensaje = "/ritmo";
ritmos[i][j].press(obx, oby, mensaje); }
if(j>10){
bancos[i][j-11].displayR();
bancos[i][j-11].press(obx, oby,"/banco");}}}

if (on_off.over==true){
onOff.add(on_off.on);
oscP5.send(onOff, myRemoteLocation);
reset.pressed=false;}

 for (int i=0;i<10;i++) {
v_barras[i].update(i);
v_barras[i].display();}}

void reset(){
on_off.on = true;
 for (int i=0;i<4;i++) {
ritmos[i][0].checked = true;
bancos[i][0].checked = true;
for (int j=0;j<16;j++) {
grid_paso[i][j].on = false;
if(j<8 && j !=0){
ritmos[i][j].checked = false; }
if(j>10){
bancos[i][j-11].checked = false;}
bancos[i][0].checked = true; }}

 for (int i=0;i<8;i++) {
//volumen
if(i<4){
v_barras[i].pos = 7*height/12;}
//reverb
if(i>3){
v_barras[i].pos = 9.6*height/12;} }
//tempo
v_barras[8].pos = 8*height/12;
//master
v_barras[9].pos = 7*height/12;
}
```

```
//V_Scrollbar
class V_Scrollbar {
float x, y;
float sw, sh;
float pos;
float posMin, posMax;
boolean rollover;
boolean locked;
float minVal, maxVal;
color fondo;
color marker;
color hold;

V_Scrollbar (float xp, float yp, float w, float h, float miv, float mav,color f, color m, color hl) {
x = xp;
y = yp;
sw = w;
sh = h;
minVal = miv;
maxVal = mav;
pos = y + sh/2 - sw/2;
posMin = y;
posMax = y + sh - sw;
fondo = f;
marker = m;
hold = hl; }

void update(int slider ) {
if (over(obx, oby) == true) {
rollover = true;
pos = constrain(oby-sw/2, posMin, posMax);
OscMessageo_scroll = new OscMessage("/scroll");
o_scroll.add(slider);
o_scroll.add(pos-posMax);
oscP5.send(o_scroll, myRemoteLocation);
println(pos);}
else {
rollover = false;} }

void press(int mx, int my) {
if (rollover == true) {
locked = true; }
else {
locked = false; }}

void release() {
locked = false;}

booleanover(float mx, float my) {
```

```
if ((mx > x) && (mx <x+sw) && (my > y) && (my <y+sh)) {
return true; }
else {
return false;}}

void display() {
fill(marker);
rect(x, y, sw, sh);
if ((rollover==true) || (locked==true)) {
fill(fondo);}
else {
fill(hold);}
rect(x, pos, sw, sw); }

float getPos() {
float scalar = sh/(sh-sw);
float ratio = (pos - y) * scalar;
float offset = minVal + (ratio/sw * (maxVal-minVal));
return offset;}}

//BotonTUIO
class BotonTUIO {
float x, y;
float size;
color baseGray;
color overGray;
color pressGray;
boolean over = false;
boolean pressed = false;

BotonTUIO(float xp, float yp, float s, color b, color o, color p) {
x = xp;
y = yp;
size = s;
baseGray = b;
overGray = o;
pressGray = p;}

void update() {
if ((obx>= x) && (obx<= x+size) &&
(oby>= y) && (oby<= y+size)) {
over = true; }
else {
over = false;}
pressed=true;}

void display() {
if (pressed == true) {
fill(pressGray); }
else if (over == true) {
fill(overGray);}
```

```
else {
fill(baseGray);}
stroke(1);
rect(x, y, size, size); }}

//ToggleTUIO
class ToggleTUIO {
float x, y, alto_b, ancho_b;
color baseGray;
color overGray;
color pressGray;
color triplet;
boolean over = false;
boolean on = false;

ToggleTUIO(float xp, float yp, float ancho, float alto, color b, color o, color p) {
x = xp;
y = yp;
alto_b = alto;
ancho_b= ancho;
baseGray = b;
overGray = o;
pressGray = p; }

void update() {
if ((obx>= x) && (obx<= x + ancho_b) &&
(oby>= y) && (oby<= y + alto_b)) {
over = true;
on= !on;         }
else {
over = false;
}}

void display() {
if (on == true) {
fill(pressGray);}
else if (over == true) {
fill(overGray);}
else {
fill(baseGray);}
stroke(0);
rect(x, y, ancho_b, alto_b);}}

class RadioTUIO{
float x, y;
float size, dotSize;
color baseGray, dotGray;
boolean checked = false;
intme,cols, my_row;
RadioTUIO[][] others;
```

```
RadioTUIO(float xp, float yp, float s, color b, color d, int m, intmr, int columns,
        RadioTUIO[][] o) {
x = xp;
y = yp;
size = s;
dotSize = size - size/3;
baseGray = b;
dotGray = d;
others = o;
cols=columns;
my_row = mr;
me = m;}
booleanpress(float mx, float my, String message) {
        OscMessageo_ritmo = new OscMessage(message);
        if (dist(x, y, mx, my) < size/2) {
checked = true;
o_ritmo.add(my_row);
o_ritmo.add(me);
oscP5.send(o_ritmo, myRemoteLocation);
for (j=0; j < cols; j++){
if (j != me) {
others[my_row][j].checked = false; }}
return true;
} else {
return false;}}

void displayR() {
stroke(0);
fill(baseGray);
ellipse(x, y, size, size);
if (checked == true) {
fill(dotGray);
ellipse(x, y, dotSize, dotSize);}}}

void oscEvent(OscMessagetheOscMessage)
{if(theOscMessage.addrPattern().equals("/metro") &&theOscMessage.typetag().equals("i")) {
int a = theOscMessage.get(0).intValue();
metro_x=a;}}
```

8. Glosario

ATM : Automated Teller Machine

BPM : Beats per minute

CLI : Command Line Interface

CV : Computer Vision

FTIR: Frustrated Total Internal Reflection

GUI: Graphical User Interface

HCI : Human Computer Interface

IETF : Internet Engineering Task Force

IP: Internet Protocol

IR: Infra red

JRE : JAVA Runtime Environment

LCD : Liquid Crystal Display

LED : Light Emitting Diode

MIDI: Musical Instrument Digital Interface

OOP: Object Oriented Programming

OSC: Open Sound Control

OSI : Open Systems Interconnection

PD: Pure Data

RFC: Request for Comments

TCP: Transmission control protocol

TUIO: Tangible User Interface Object

UDP: User Datagram Protocol

URL : Uniform Resource Locator

USB : Universal Serial Bus

9. Referencias bibliográficas

Bencina, R, Kaltenbrunner, M. (2005).*The design and evolution of fiducials for the Reactivision system*.Proc. 3rd International Conference on Generative Systems in the Electronic Arts.

Bencina, R,Kaltenbrunner, M. &Jorda, S. (2005). *Improved topological fiducial tracking in the Reactivision system*.Computer Vision and Pattern Recognition-Workshops, 2005.CVPR Workshops.IEEE Computer Society Conference.

Geiger, G., Alber, N., Jordá, S., & Alonso, M. (2010).*The reactable: A collaborative musical instrument for playing and understanding music*.Her&Mus. Heritage &Museography.

Halliday, D., Resnick, R., & Walker, J. (2010). *Fundamentals of physics*.John Wiley & Sons.

Han, J. Y. (2005). *Low-cost multi-touch sensing through frustrated total internal reflection*. Proceedings of the 18th annual ACM symposium on User interface software and technology. ACM.

IETF RFC 768. User Datagram Protocol

IETF RFC 793. Transmission Control Protocol

Kaltenbrunner, M. (2009).*Reactivision and TUIO: a tangible tabletop toolkit*. In Proceedings of the ACM international Conference on interactive Tabletops and Surfaces.ACM.

Kaltenbrunner, M., Bovermann, T., Bencina, R., &Costanza, E. (2005).*TUIO: A protocol for table-top tangible user interfaces*. P. of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation.

Shneiderman, B., & Ben, S. (2003). *Designing the user interface*.Pearson Education.

Puckette,M. (1996). *Pure Data: another integrated computer music environment*. Proceedings of the Second Intercollege Computer Music Concerts.

Puckette, M. (2007).*The theory and technique of electronic music*.

Reas, Casey, Fry, Ben. (2007). *Processing A Programming Handbook for Visual Designers and Artists*. MIT Press.

Schöning, J., Brandl, P., Daiber, F., Echtler, F., Hilliges, O. (2008).*Multi-touch surfaces: A technical guide*. IEEE Tabletops and Interactive Surfaces, 2.

Schöning, J., Brandl, P., Daiber, F., Echtler, F., Hilliges, O., Hook, J., ...& von Zadow, U. (2008). *Multi-touch surfaces: A technical guide*. IEEE Tabletops and Interactive Surfaces, 2.

Schöning, J., Hook, J., Bartindale, T., Schmidt, D., Oliver, P., Echtler, F.,& von Zadow, U. (2010). *Building interactive multi-touch surfaces*.Tabletops-Horizontal Interactive Displays. Springer London.

Wright, M. (1997).*Open Sound Control-A New Protocol for Communication with Sound Synthesizers*. In Proceedings of the 1997 International Computer Music Conference.