Universidad de la República
Facultad de Ingeniería
Uruguay

Soluciones interactivas para la danza

Ismael Pisano

Gonzalo Rosso

Proyecto de Grado
Ingeniería en Computación
Universidad de la República

Montevideo, Uruguay, Octubre de 2020

Supervisores: Dr. Ing. Daniel Calegari García
Mag. Ewelina Bakala

Resumen

Dado el incremento en la cantidad y calidad de sensores y actuadores que mejoran la interacción entre personas y computadoras, es que nos vemos tentados a explorar, investigar, e implementar nuevas herramientas que exploten la utilización de estos componentes. En esta ocasión llevándolo al terreno del arte.

El objetivo del proyecto es brindar, a artistas de la danza, una herramienta que permita autonomía a la hora de realizar obras donde se incluya la interacción con contenidos multimedia proyectados. Estos contenidos pueden estar enriquecidos mediante el uso de un sensor de profundidad Kinect que permita identificar al artista y sus movimientos en escena. Por lo tanto, se permite generar contenidos dinámicos con los que se pueda interactuar en base a estos movimientos.

Uno de los principales requerimientos es que la herramienta resulte sencilla e intuitiva para el artista y no requiriéndose conocimientos técnicos ni de programación para su utilización.

Palabras clave: Danza interactiva, kinect, processing, visión por computadora, proyección, detección de cuerpo, interacción.

Contenido

1 Introducción

Hoy en día la utilización de herramientas audiovisuales que complementan obras artísticas se está haciendo muy popular debido al creciente abanico de ofertas de artículos para proyección y captura de movimientos, así como la disminución en los precios vinculados a los mismos, y a computadoras con gran capacidad de procesamiento. Se puede apreciar el uso de las herramientas antes mencionadas en ejemplos tales como Projection Ballet [1] y Projector Dance [2].

En general, el agregado de elementos audiovisuales e interactivos a una obra de danza, es complejo. No solo por el hecho de crear y configurar estos elementos para que la obra sea agradable, sino por el hecho de coordinar a gente capacitada para el manejo y la ejecución de los mismos, principalmente en cambios de locación o giras.

Actualmente la puesta en escena de una obra con estas características conlleva a no sólo invertir en software que reúna las características deseadas, sino también a disponer de personas con el conocimiento técnico necesario para la utilización de dicho software.

Nos encontramos con la idea de poder simplificar la creación, armado, y ejecución de una obra mediante el manejo de la tecnología.

Un desafío importante surge al momento de querer lograr independencia de personas con conocimientos técnicos y software privativo para cubrir todas las etapas del desarrollo de una obra, abarcando desde la concepción hasta la ejecución final de la obra en sí.

Se toman como motivación y punto de partida en cuanto a contenidos, varios trabajos previos realizados en Talleres de Tecnología en Artes Escénicas[1] dictados en la Facultad de Ingeniería - UdelaR. Con este material también se comienza a dimensionar el posible alcance en cuanto al desarrollo e independencia de la parte técnica necesarios para ejecución de la herramienta.

[1] Taller TAEs: https://vimeo.com/146317152

Para facilitar el entendimiento del resto del documento, se definen dos conceptos que serán utilizados de forma elemental, estos son *escena* y *proyección*. Escena refiere a la partición de una obra donde permanecen constante ciertos elementos (escenografía, contenidos, personajes, etc). Adicionalmente, proyección refiere a una imagen o conjunto de imágenes que se proyectan sobre una pantalla o superficie, teniendo en cuenta que en nuestro caso particular se podrá referir a un área determinada dentro de una pantalla.

Con el fin de enriquecer las ideas planteadas se cuenta con la colaboración de la escuela de danza Espacio Pangea [2] para definir ciertos objetivos específicos:

1. Generar una herramienta que mediante una interfaz gráfica amigable, le permita al artista definir distintos espacios de proyección que requiere su obra.

2. Permitir la asignación de contenidos multimedia específicos para los espacios de proyección definidos.

3. Desplegar el contenido multimedia asociado a cada área de proyección de forma independiente y simultánea según se desee.

4. Posibilitar la carga de material multimedia, donde no exista límite en cantidad o tipo de contenido.

5. Lograr una proyección sobre el cuerpo del artista que se ajuste al contorno de la persona y acompañe los movimientos que realiza en tiempo real mediante la integración de un sensor Kinect y un proyector frontal.

El documento se organiza del modo que se describe a continuación:

El capítulo 2 presenta el marco de trabajo del proyecto, donde se brinda un resumen del estado del arte, con una breve descripción y análisis de las herramientas existentes para evaluar su potencial de utilización.

El capítulo 3 especifica los requisitos que se desean abordar para la implementación de una solución que abarque las ideas ya mencionadas.

Luego se describe la solución planteada en el capítulo 4, donde se detallan sus características, funcionalidades, componentes y limitaciones.

Se detalla la aplicación práctica de la herramienta en el capítulo 5, donde se muestra el resultado de la ejecución de una obra de danza interactiva.

Finalmente en el capítulo 6 se presentan las conclusiones y las principales líneas de trabajo a futuro.

[2] Espacio Pangea: Página en Facebook: https://www.facebook.com/espaciopangea/

2 Estado del Arte

Al comenzar el proyecto se realiza una investigación sobre herramientas existentes que pudieran servir en la implementación de la solución, ya sea como una dependencia, o simplemente de donde se puedan tomar ideas que ayuden a lograr un mejor producto.

En esta sección se presentan, en primer lugar el lenguaje de programación predefinido para la implementación del proyecto y en segundo lugar, las herramientas que son comunes a lo largo del desarrollo del mismo.

Luego se dividen de acuerdo a los objetivos específicos que se detallan en la introducción, acordes a las dos principales ideas planteadas. Por un lado el manejo y gestión de la configuración de la obra, y por otro lado la proyección de contenidos sobre un cuerpo en movimiento.

2.1 Processing

Para realizar el proyecto se utiliza Processing[3], que es un lenguaje de programación y entorno de desarrollo integrado (IDE) de código abierto, de fácil utilización. Este sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital.

El desarrollo de este lenguaje fue iniciado por Ben Fry y Casey Reas, ambos miembros de Aesthetics and Computation Group del MIT Media Lab dirigido por John Maeda.

Uno de los objetivos de Processing es el de actuar como herramienta para que diseñadores visuales, artistas, y miembros de otras comunidades ajenos a la programación, aprendan las bases de la misma a través de una muestra gráfica instantánea.

De manera muy sencilla Processing brinda la posibilidad de generar gráficos tanto en 2D como en 3D, manejar sensores como por ejemplo Kinect y también trabajar con entradas/salidas de sonido, todo esto siendo manipulado en tiempo real y permitiendo la interacción constante. Estos últimos puntos son los que permitieron su uso en este proyecto ya que era prioridad el trabajo en tiempo real y la interacción con las bailarinas.

A pesar de que el lenguaje de Processing se basa en Java, hace uso de una sintaxis simplificada y brinda un modelo para programación de gráficos.

Al ser una alternativa al software propietario carece de restricciones y costos de licencias por lo que se hace accesible a todo tipo de usuario, principalmente estudiantes. Cabe destacar que este lenguaje funciona para Mac, Windows y GNU/Linux.

Entorno de desarrollo (IDE)

El entorno de desarrollo de Processing incluye un editor de texto, un compilador y una o varias ventanas donde se muestra el resultado del código ejecutado.

Cada programa que se escribe en Processing es denominado Sketch. En la figura 2.1 se puede apreciar el editor de Processing y una ventana con el código en ejecución. Este código dibuja dos imágenes superpuestas con diferentes tamaños.

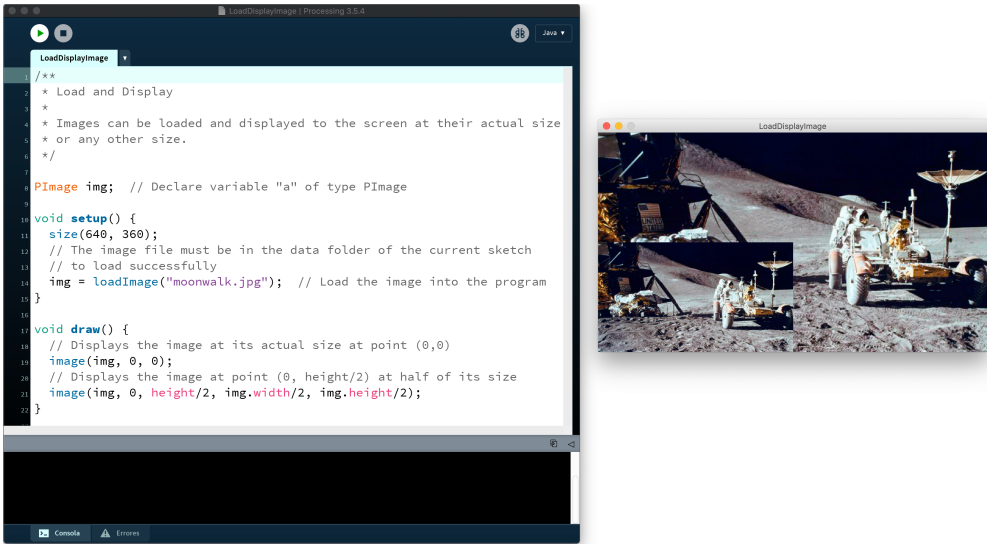


Figura 2.1: Entorno de desarrollo Processing

Se hace un breve racconto de las posibles alternativas a Processing y sus funcionalidades para entender ventajas y limitaciones que nos ayuden a delimitar de forma certera nuestros requerimientos.

Pasamos a describir brevemente cada una de las alternativas analizadas:

- **Openframeworks**[4]

  Es un kit de código abierto sobre C++. Ofrece muchas de las funcionalidades que brinda Processing. Una ventaja importante es que tiene muy buena documentación así como una gran comunidad y proyectos disponibles de ejemplo.

- **P5js**[5]

  Biblioteca de Javascript que surge a partir de Processing y comparte su objetivo, hacer que la creación sea accesible a artistas, diseñadores, educadores y principiantes. Es gratuita y de código abierto. En términos de FPS brinda menor rendimiento que Processing (50 contra 60 fps en promedio).

- **Cinder**[6]

  Librería gratuita y de código abierto basada en C++. Su utilización no es muy intuitiva y requiere conocimientos avanzados de programación.

- **MAX**[7]

Software propietario enfocado en desarrollo multimedia. Está basado en estructuras modulares. Escrito en C y C++. Su licencia cuesta entre 100 y 400 USD.

- **Codea**[8]

  Editor de código para dispositivos con iOS. Se enfoca en diseñar rápidamente ideas visuales, especialmente videojuegos.

2.2 Configuración de escenas y contenidos

En este apartado se estudian soluciones ya existentes que podrían ser de ayuda para el desarrollo del módulo de configuración de las escenas. Se hace hincapié en herramientas que permitan configurar y manipular en tiempo real varias áreas de proyección.

A continuación se detallan las herramientas estudiadas, explicando las ideas obtenidas en cada caso, y las razones por las cuales se tomaron en cuenta o fueron descartadas.

Keystone

Keystone [9] es una biblioteca que permite hacer mapeo (mapping[1]) de proyecciones en Processing. Permite que la proyección sobre superficies planas se vea correctamente, independientemente de la posición u orientación del proyector.

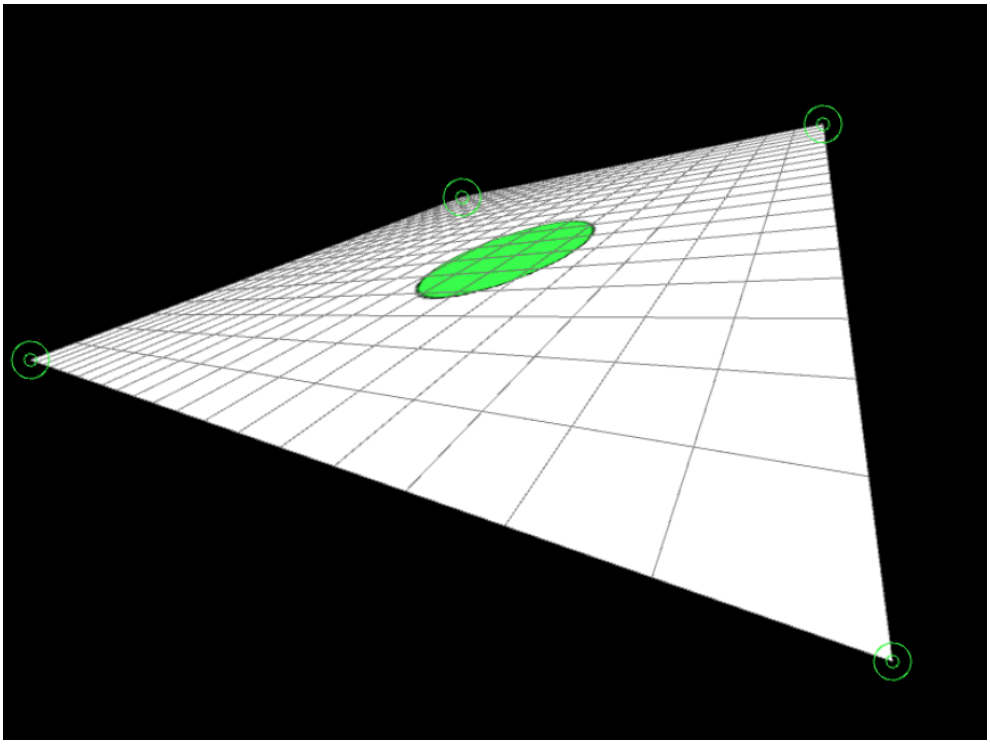


Figura 2.2: Interfaz de Keystone

[1] Ejemplo de mapping: https://www.youtube.com/watch?v=6q21a1anI6w

En la Figura 2.2 se puede apreciar que la parte blanca corresponde a la salida del proyector, los círculos verdes en los vértices se corresponden a los vértices de la superficie sobre la que se proyecta. También se puede ver que el círculo verde del centro se muestra en perspectiva acorde a la superficie definida.

Esta herramienta fue investigada a raíz de no saber la ubicación que podrían tener los proyectores en un escenario, pero luego se descartó al trabajar sobre la hipótesis que siempre se utilizarían centrados y de forma frontal.

SketchMapper

Esta biblioteca [10] para Processing ofrece una interfaz que permite crear diferentes sketches y mapearlos en superficies independientes.

Está basado en surfaceMapperGUI [11] desarrollado por Jason Webb para la definición de superficies y en Surface Mapper [12] desarrollado por Ixagon.

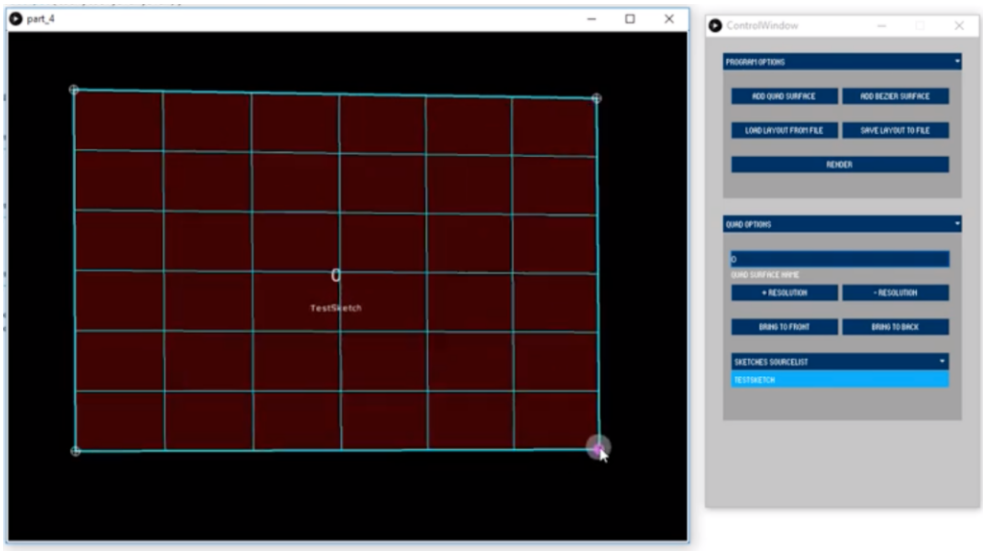


Figura 2.3: Interfaz de Sketch Mapper

Si bien esta herramienta brinda una gran cantidad de funcionalidades que se buscaban (destacando la definición/ajuste del área a proyectar y asociación de contenido), no se utilizó dado que al momento de su análisis solo existía su versión para Processing 2, lo que nos requería un esfuerzo importante para adaptarlo a Processing 3. Para el momento que se lanzó su versión para Processing 3 ya teníamos nuestra propia herramienta desarrollada.

ControlP5

Esta librería [26] provee controles gráficos que permiten desarrollar interfaces gráficas amigables.

Cuenta con una amplia gama de controles como se puede apreciar en la figura 2.4, que van desde botones, selectores, áreas de texto, checkbox y sliders, entre otros.

Todos los controles se pueden manipular de forma muy sencilla dentro de cualquier sketch de Processing y además es posible guardar el estado de cada control dentro de un archivo de formato JSON.

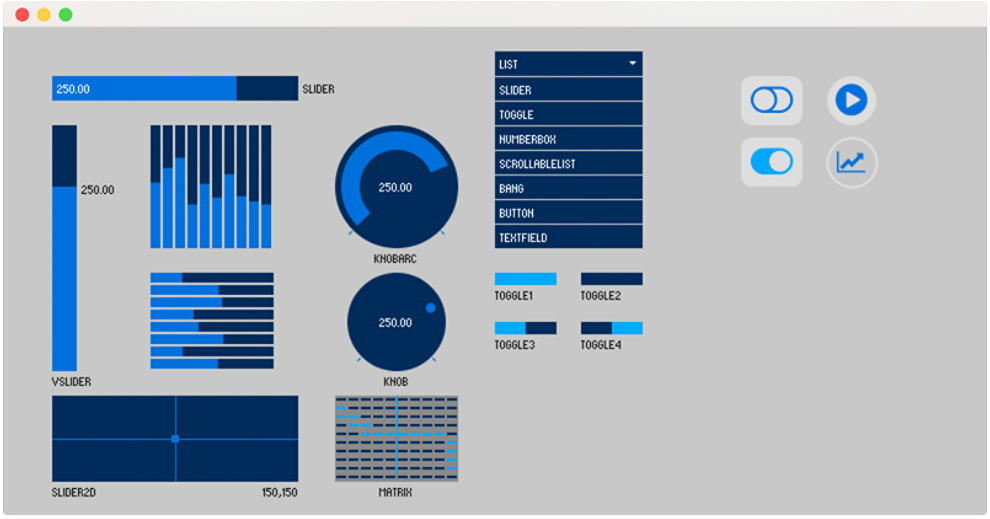


Figura 2.4: Controles de la librería ControlP5

2.3 Proyección sobre el cuerpo

Para cumplir con el desafío de poder proyectar sobre el cuerpo se analizan y evalúan varias herramientas y soluciones existentes.

En particular se analizan las ventajas de cada solución, teniendo en cuenta el rendimiento y la facilidad de integración que cada una presenta.

SimpleOpenNI

Esta librería [14] para Processing permite el manejo del Kinect [13] de forma sencilla.

Está creada sobre OpenNI [15] y NITE [16]. Entre sus principales funcionalidades se pueden encontrar el reconocimiento de personas (individuos), reconocimiento del esqueleto y gestos, como se muestra en la figura 2.5.

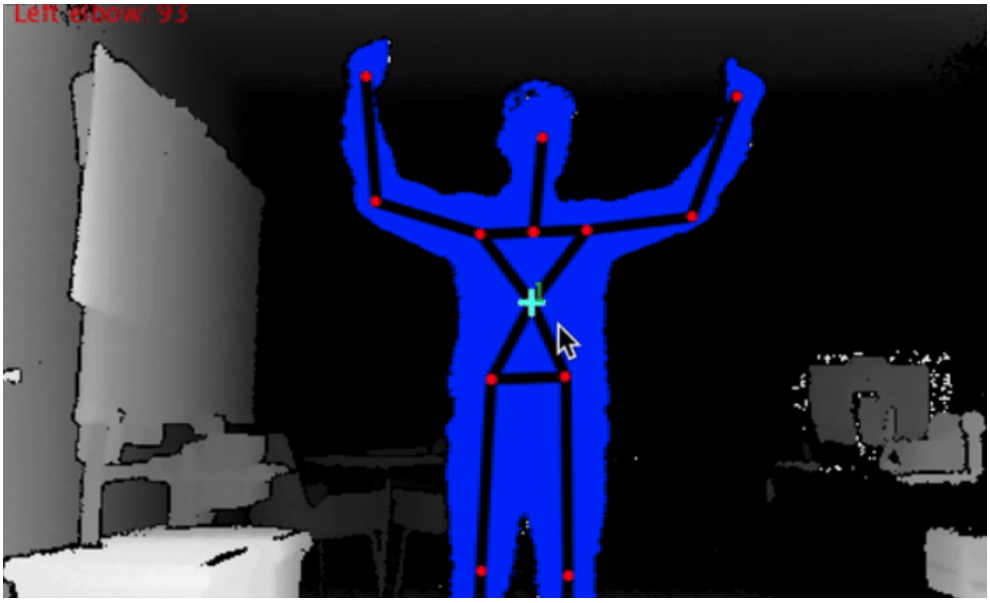


Figura 2.5: Reconocimiento de esqueleto

Principalmente la funcionalidad de reconocimiento de personas resulta atractiva pero presenta una clara desventaja al momento de realizar el reconocimiento ya que solo identifica correctamente a personas paradas y en donde cualquier distorsión de postura o movimiento dificulta dicha tarea de reconocimiento.

(a) Reconocimiento de una persona cuando hay dos

(b) Reconocimiento de dos personas

Figura 2.6: Reconocimiento de varios cuerpos

En la figura 2.6 se puede apreciar como la herramienta reconoce los cuerpos y dibuja ciertos puntos de referencia en función a cada reconocimiento.

Por un lado, en la figura 2.6a se ve claramente la falla en el reconocimiento de la segunda persona, siendo que se encuentra simplemente parada y su posición no presenta ninguna complejidad.

Por otra parte, en la figura 2.6b vemos que con un cambio mínimo en la postura del segundo individuo, el reconocimiento se realiza exitosamente.

Open Kinect for Processing

Biblioteca [17] que utiliza drivers de código abierto del proyecto libfreenect [18] (librerías y drivers de Kinect para Windows, Linux y OS X) para facilitar la conexión entre el sensor Kinect y Processing.

Provee una capa de abstracción para facilitar la conexión y el acceso a las cámaras RGB e IR del Kinect, así como también controlar el ángulo de las cámaras.

KinectProjectorToolkit

Librería [19] de Processing para lograr calibrar el Kinect y la salida del proyector, ajustando las proyecciones a cuerpos o superficies en movimiento.

Mediante la identificación de 12 vértices sobre una grilla y en repetidos pasos se genera una nube de puntos que permiten ajustar la visión del Kinect con la proyección. Con esto se logra

hacer coincidir lo que el Kinect está viendo con lo que se está proyectando sin necesidad que ambos dispositivos estén en posiciones específicas o alineados.

En la figura 2.7 se puede ver la interfaz de la herramienta. Se puede apreciar, sobre la parte izquierda de la imagen, la grilla que se utiliza para obtener los puntos en el espacio, y sobre la derecha de la pantalla se selecciona la configuración deseada en cuanto a la posición y tamaño de la grilla.

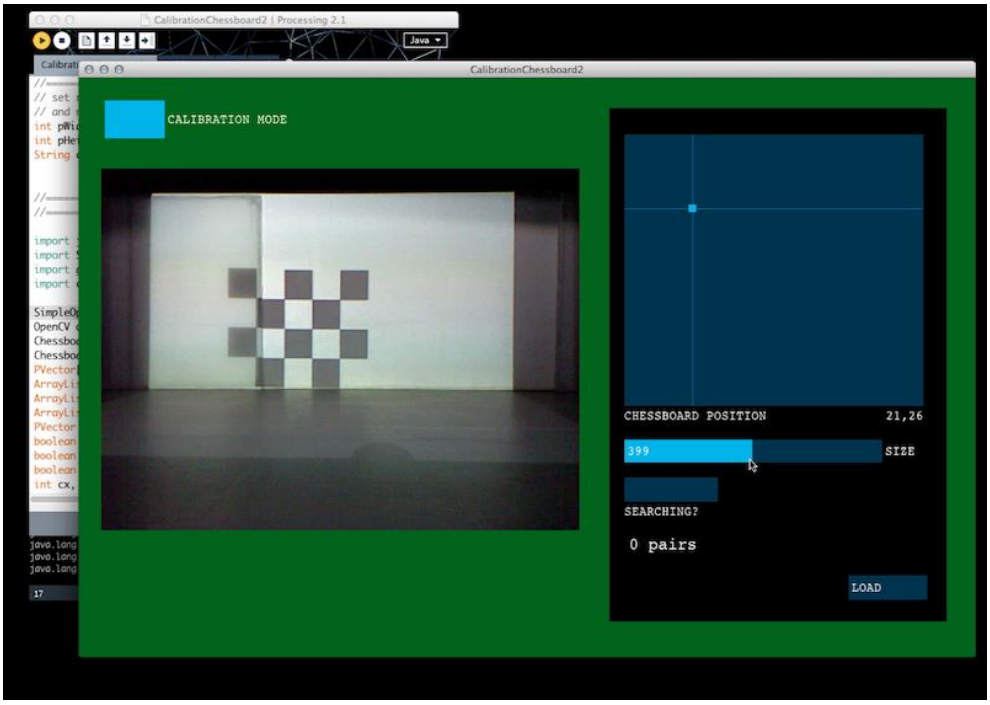


Figura 2.7: Interfaz de Kinect Projector Toolkit - Modo Calibración

Se considera una herramienta útil ya que se utiliza para obtener la nube de puntos de configuración que permite hacer coincidir la lectura del Kinect con la salida del proyector.

OpenCV for processing

OpenCV[20] es una biblioteca de código abierto basada en OpenCV Java API y que está enfocada en la visión por computadora.

Entre sus funcionalidades podemos destacar el reconocimiento de caras, el encontrar contornos o líneas rectas, realizar sustracción de fondo, hacer seguimiento de objetos en movimiento

y detectar patrones. En nuestra solución utilizamos sus métodos referidos a contornos y áreas, de manera que a partir de una serie de puntos desordenados en el plano, podemos encontrar el contorno que contiene a todos los puntos.

3 Requisitos

En esta sección se detallan los requisitos funcionales definidos para cada uno de los objetivos específicos detallados en el capítulo 1 Introducción, como también los requisitos no funcionales inherentes a la solución que se busca.

Estos objetivos incluyen la generación de una herramienta amigable y sencilla de utilizar por parte del artista sin conocimientos técnicos de programación, donde se le permita definir y configurar todos los aspectos relacionados a una obra de danza.

Asimismo, se especifican detalladamente los requerimientos a lo pertinente con la generación y asignación de los contenidos multimedia, donde se involucra la proyección sobre el cuerpo del artista en escena.

3.1 Requisitos funcionales

3.1.1 RF1 - Configuración de escena

Mediante un módulo de configuración, el usuario debe poder agregar, modificar, duplicar, o borrar varios espacios de proyección dentro de una única proyección correspondiente a cada escena.

3.1.2 RF2 - Configuración de espacios de proyección

Cada espacio de proyección se compondrá de una o varias figuras, estas figuras podrán ser polígonos y/o elipses.

3.1.3 RF3 - Configuración de figuras

Se permitirá al usuario agregar, modificar, y borrar las figuras que definen cada espacio de proyección. La modificación se podrá realizar mediante el teclado para permitir un ajuste fino y preciso.

3.1.4 RF4 - Elección de contenido

Se podrá seleccionar desde una lista predefinida qué contenido se desea proyectar en cada espacio de proyección definido.

3.1.5 RF5 - Ajuste de contenido

Se facilitará la posibilidad de elegir de qué forma se ajusta el contenido al área de proyección. Brindando la posibilidad de redimensionar, recortar o no realizar modificación alguna del contenido, según se desee.

3.1.6 RF6 - Exportar/Importar archivo de configuración

El módulo de configuración permitirá importar y/o exportar un archivo de configuración que defina las escenas y sus componentes para su posterior utilización.

3.1.7 RF7 - Visualización de contenidos

Se desea que se visualice mediante una única proyección el contenido definido para cada espacio de proyección de una escena. Las figuras definidas crearán una máscara para esta visualización.

3.1.8 RF8 - Reconocimiento del cuerpo

Se desea reconocer un cuerpo humano mediante la utilización de un dispositivo Kinect.

3.1.9 RF9 - Proyección en el cuerpo detectado

Interesa proyectar el contenido especificado en el área detectada. Funcionando el área del cuerpo como un espacio de proyección en movimiento.

3.1.10 RF10 - Definición de contenido

El usuario podrá elegir el contenido que se proyectará en el área del cuerpo detectado.

3.1.11 RF11 - Adaptación de contenido

Brindar la posibilidad de adaptar contenidos externos y/o ya existentes para su utilización en el proyecto, mediante modificaciones mínimas.

3.2 Requisitos no funcionales

3.2.1 RNF1 - Rendimiento

La solución deberá buscar optimizaciones para que el uso de la herramienta no presente retardos y despliegue los contenidos de forma fluida.

3.2.2 RNF2 - Fácil utilización

Dado que se desea que la herramienta sea utilizada por personas no técnicas ni con experiencia en programación, se desea que la interfaz sea simple y permita una utilización sencilla.

3.2.3 RNF3 - Detección en tiempo real

Es fundamental el procesamiento inmediato de la detección del cuerpo ya que se debe proyectar sobre un cuerpo en movimiento.

3.2.4 RNF4 - Detección de cuerpos en distintas posiciones

Al ser aplicado en obras de danza, los cuerpos no siempre están en una posición reconocible por detectores de cuerpos humanos. Es por esto que se busca que la solución detecte cuerpos en cualquier posición.

4 Descripción de la Solución

El objetivo de este capítulo es describir cómo se implementa la solución propuesta que permite definir escenas y sus distintos espacios de proyección mediante la conjugación de elipses y polígonos. Luego, se detalla cómo se vinculan estos espacios de proyección a distintos contenidos audiovisuales.

Posteriormente, se explica la manera en que se realiza la importación y exportación de la configuración de las escenas ya mencionadas.

Se investigan técnicas de calibración que permitan hacer coincidir puntos en el espacio captados por el Kinect con lo que se debe proyectar con el fin de evitar problemas de alineación.

Para finalizar, se describen diferentes técnicas investigadas para el reconocimiento del cuerpo, describiendo sus ventajas y desventajas en relación a su desempeño y resultados obtenidos.

4.1 Proceso de desarrollo

Para ordenar el proceso de desarrollo se comienza por agrupar en dos módulos los requerimientos ya definidos en el capítulo 3.

1. **Manejador**: Involucra la configuración, manipulación, y visualización en tiempo real de varios espacios de proyección (sub-proyecciones) en una única proyección. Abarca fundamentalmente a los requisitos funcionales desde RF1 al RF7 definidos en la sección 3.1, en conjunto al RF11 que refiere a la adaptación de contenidos y que se encuentra definido en la misma sección.

2. **Proyección sobre el cuerpo**: Implica seguimiento del cuerpo y ajuste de las proyecciones teniendo en cuenta: la distancia entre el sensor, la persona a seguir y las características del proyector. Involucra específicamente a los requisitos funcionales RF8, RF9, RF10 definidos en la sección 3.1.

Luego se organiza el trabajo en 4 etapas bien diferenciadas, donde en cada una de ellas se plantea un objetivo específico.

- **Primera etapa**: Familiarización con Processing y bibliotecas a utilizar.
- **Segunda etapa**: Investigación e implementación del Módulo 1 - Manejador
- **Tercera etapa**: Investigación e implementación del Módulo 2 - Proyección sobre el cuerpo
- **Cuarta etapa**: Preparación y ejecución de una obra en conjunto con Espacio Pangea.

A lo largo de las 4 etapas se mantienen reuniones semanales en el Espacio Pangea con el fin de probar el avance de cada módulo y a su vez intercambiar conocimientos con la bailarina/directora del espacio, Cristina Ibarra.

Para la segunda y tercera etapa se realiza una subdivisión, que consiste en una fase inicial de investigación y una fase posterior de implementación. En esta última fase se incluyen a su vez las pruebas de uso y rendimiento.

4.2 Implementación

La solución propuesta es desarrollada utilizando Processing como lenguaje de programación y haciendo uso de un sensor Kinect que permita obtener información para utilizar al momento de generar interacción entre los contenidos audiovisuales y los individuos en escena.

Se busca que la solución planteada no presente limitaciones para ser ejecutada en distintos sistemas operativos. Con el fin de evitar que el sistema operativo sea una barrera es que se realizan pruebas para que la solución funcione en Windows, MacOS y Linux.

Se agrupan los requisitos funcionales definidos en el capítulo 3 en dos módulos, por un lado un módulo referente al manejo de las escenas y por otro lado un módulo asociado a la proyección sobre el cuerpo.

En los siguientes apartados se pueden apreciar todos los detalles vinculados a cada uno de los módulos antes mencionados.

4.2.1 Manejador de escenas

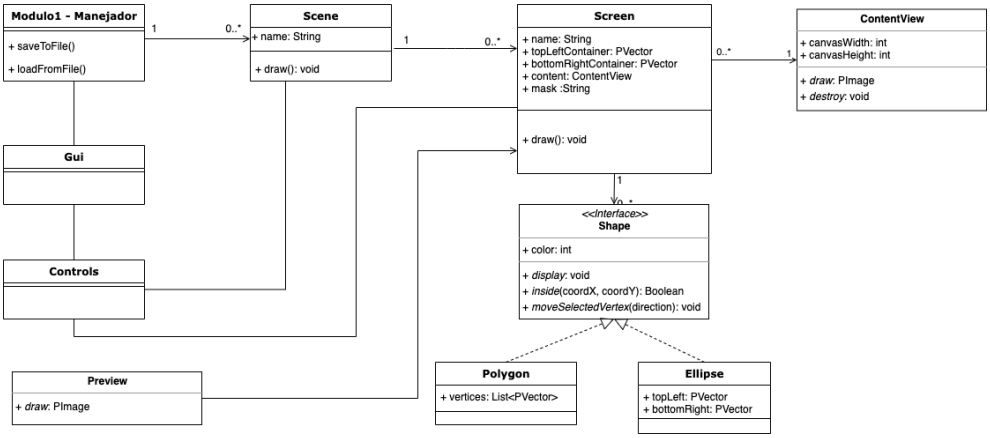


Figura 4.1: Arquitectura de la solución

Se comienza por el desarrollo del manejador de escenas, que es el encargado de proveer al usuario una interfaz para configurar todo lo que tenga que ver con los aspectos previos a la obra. Estos aspectos están vinculados a la creación de diferentes escenas que contengan distintas áreas de proyección que se forman utilizando figuras (elipses y/o polígonos).

A su vez se brinda la posibilidad de asociar contenidos a cada una de la áreas de proyección.

Por último, se detalla la forma en que se guarda las configuración con todos los elementos

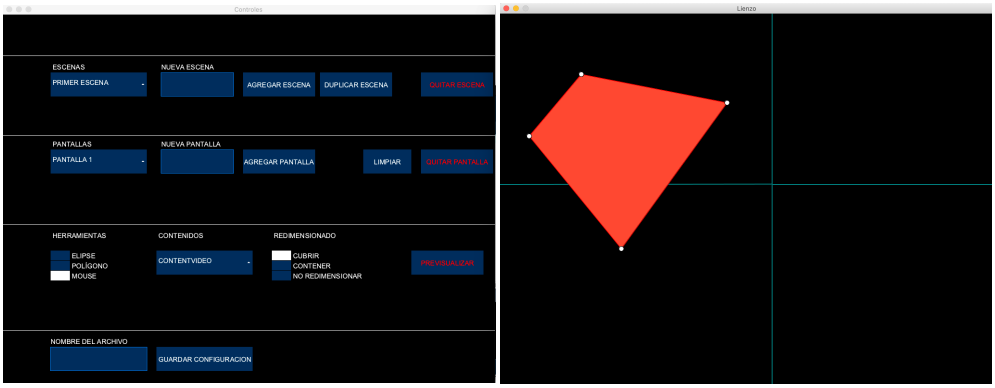
antes mencionados para su posterior utilización.

INTERFAZ DE USUARIO

Tal como se define en el requisito no funcional RNF2 de la sección 3.2.2, uno de los puntos importantes es la usabilidad de la herramienta, por lo que se busca una interfaz de usuario sencilla y simple.

Se utiliza la librería controlP5 [26] para modelar la interfaz de usuario mediante controles gráficos, tal como se puede apreciar en la figura 4.2.

Para realizar la configuración inicial de la obra se idea un flujo que facilite al usuario la definición de cada escena y área de proyección sin la necesidad de tener conocimientos técnicos y/o de programación.



(a) Controles (b) Lienzo

Figura 4.2: Interfaz de usuario

Este flujo consiste en iterar los siguientes pasos para cada una de las escenas que se desee definir:

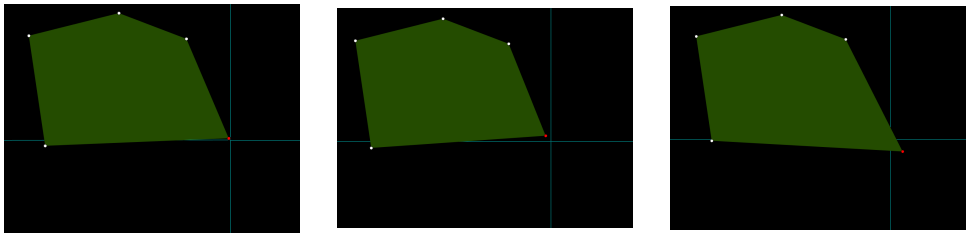
1. Crear una escena o duplicar una escena ya existente.
2. Crear uno o más espacios de proyección dentro de cada escena.
3. Agregar figuras dentro del espacio de proyección.
4. Asignar un contenido al espacio de proyección definida.
5. Seleccionar el tipo de ajuste, acorde a lo detallado en 4.2.1, que se le realizará al contenido asignado dentro del espacio asociado. Esto implica definir si el contenido se ajusta de forma particular al espacio asociado o simplemente se muestra en su tamaño original.
6. Opcionalmente se permite la previsualización de la pantalla seleccionada.
7. Guardar la configuración para su uso en la obra o para cargar una configuración preexistente y continuar su edición.

Por más detalles sobre la utilización de la solución se puede consultar el Manual de Uso presente en el apéndice B.

AJUSTE FINO DE FIGURAS

Se brinda la posibilidad de realizar un ajuste de mayor precisión sobre las figuras y sus vértices, mediante la utilización de las flechas del teclado.

En cualquiera de los dos casos se debe seleccionar el elemento a ajustar (figura o un vértice particular) para luego comenzar con el movimiento píxel a píxel en cualquier dirección que se desee.



(a) Selección de vértice (b) Ajuste Noroeste (c) Ajuste sureste

Figura 4.3: Ajuste fino de figura

AJUSTE DE CONTENIDOS

Con el fin de brindar alternativas al momento de visualizar los contenidos (RF5), se brinda la posibilidad de ajustar el contenido dentro del espacio en que se proyecta.

A su vez, se denota que esta funcionalidad permite facilitar el uso de la herramienta desarrollada ya que el usuario no debe proveer los contenidos tal cual se deben utilizar y puede variar la configuración según se desee.

Estos ajustes corresponden a las siguientes 3 opciones:

- **Cubrir**: Redimensiona el contenido para que cubra todas las superficies de las figuras en el espacio de proyección.
- **Contener**: Se ajusta el contenido de manera que se muestre en su totalidad dentro del espacio definido.
- **No redimensionar**: El contenido se visualiza en tamaño original, sin alineación particular más que el anclaje desde la esquina superior derecha del espacio de proyección.

En la figura 4.4 se explican cada una de estas tres opciones de ajuste de contenido, tomando como espacio de proyección un círculo.

(a) Cubrir (b) Contener (c) No redimensionar

Figura 4.4: Ajuste de contenidos

GENERACIÓN DE MÁSCARAS

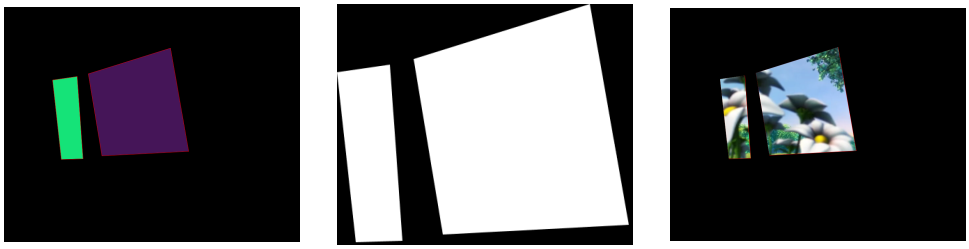
Conforme al RF7 y con el fin de respetar la configuración establecida es que se deben generar máscaras para permitir la correcta visualización de los contenidos.

Para ello se genera una imagen en formato TIFF [1] basada en las figuras que se definen para cada área de proyección. Cabe destacar que cada máscara generada es del tamaño del rectángulo que contiene a todas las figuras que la definen, tal como se explica en la figura 4.6.

Asimismo, también se referencia la posición donde se ubica cada máscara generada ya que éstas no ocupan la totalidad del espacio de proyección, sino que tan solo el área definida por

[1] TIFF: https://www.adobe.io/open/standards/TIFF.html

las figuras que las definen.



(a) Figuras (b) Máscara generada (c) Previsualización

Figura 4.5: Generación y utilización de la máscara

Como se puede apreciar en la figura 4.5, se crea una máscara en donde las figuras se marcan en color blanco plano. Luego esa máscara es utilizada para recortar el contenido que se desea mostrar en ese espacio.

Tal como ya se detalló, se puede apreciar que la máscara generada (Figura 4.5b) no es del tamaño total de la proyección sino que tan solo es una porción de la misma. Finalmente, para su utilización se la posiciona donde corresponde y se ajusta el contenido dentro de dicha máscara según se defina.

GUARDADO Y CARGA DE CONFIGURACIÓN

Siguiendo lo definido en el requisito funcional RF6, se define la estructura con la que se guardan y se utilizan los datos relacionados a la configuración de las escenas.

Estos datos son almacenados en formato JSON [27] ya que es un formato liviano que permite una lectura fácil para humanos, y a su vez es fácil de analizar y generar a nivel de máquina.

Se pasan a detallar cada una de las partes que componen la estructura del archivo resultante. El resultado final se puede apreciar completo en el ejemplo B.2.5

Lo primero que se requiere es conocer el tamaño de la proyección que se está realizando, es decir el tamaño de la pantalla sobre la que se definen las escenas. Es por ésto que se comienza definiendo el alto y ancho en píxeles de la proyección.

```
"canvasWidth": 1024,
"canvasHeight": 768,
```

Luego se especifican los detalles de cada una de las escenas, mediante la asignación de un nombre y una lista de pantallas que las componen.

```
"scenes": [{
    "name": "Escena 1",
    "screens": [{ ...}]
}]
```

Para cada una de los espacios de proyección que componen una escena se tiene:

- *Nombre* (name) que la identifica
- Lista de *figuras* (shapes) que contiene
- Posición que ocupa basados en los puntos superior izquierdo (contentTopLeft) e inferior derecho (contentBottomRight) del rectángulo que contiene a todas sus figuras. En la figura 4.6 se puede apreciar exactamente cuales son los puntos antes mencionados.
- Referencia al archivo que contiene la máscara generada en base a sus figuras (mask).
- En caso de tener contenido asignado, se detalla el tipo de contenido (contentClass) y el tipo de ajuste sobre el contenido seleccionado (resizeMode).

```
"screens": [{
      "contentTopLeft": {
        "x": 141,
        "y": 126
      },
      "contentBottomRight": {
        "x": 823,
        "y": 378
      },
      "color": 255,
```

```
      "shapes": [...],
      "name": "Pantalla A",
      "contentClass": "ContentVideo",
      "resizeMode": "Cover",
      "mask": "data/test/maskE0S0.tif"
```

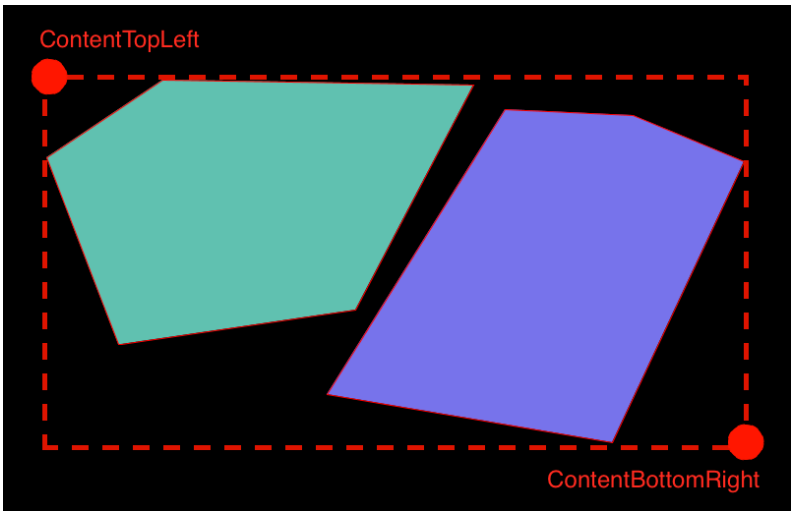


Figura 4.6: Puntos de referencia del contenedor de figuras.

Finalmente, se detallan cada una de las figuras presentes dentro del área de proyección mediante la especificación de su tipo (Polígono o Elipse), posición, y el color que tienen asignado que ayuda a su visualización durante la configuración.

Para el caso del Polígono, se detallan las coordenadas de cada uno de sus vértices y para la Elipse se detallan tan sólo los puntos que corresponden a los valores superior izquierdo e inferior derecho del rectángulo que la contiene, tal como se muestra en la figura 4.7.

Obteniendo como resultado la porción del archivo JSON que se ve a continuación:

```
[{
"shapeType": "Polygon",
"color": -10436176,
"coords": [
  {
    "x": 75,
    "y": 318
  },
  {
    "x": 193,
    "y": 239
  },
  {
    "x": 509,
```

```
      "y": 244
    },
    {
      "x": 389,
      "y": 473
    },
    {
      "x": 151,
      "y": 508
    },
    {
      "x": 148,
      "y": 508
    }
  ]
},
{
  "shapeType": "Ellipse",
  "color": -11555003,
  "coords": [
    {
      "x": 606,
      "y": 301
    },
    {
      "x": 958,
      "y": 460
    }
  ]
}
]
```

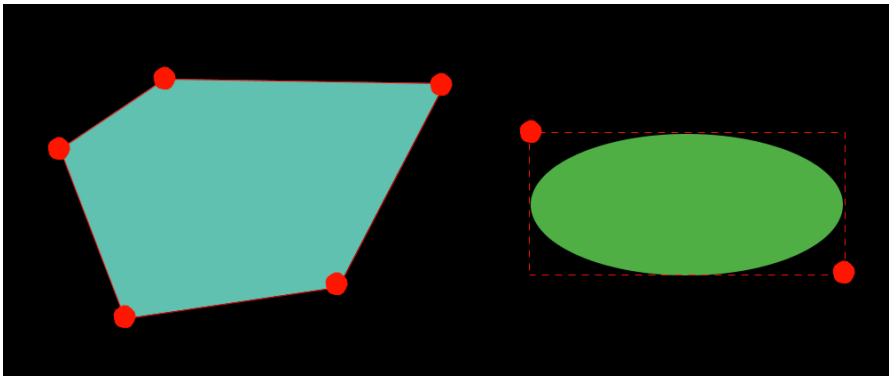



Figura 4.7: Puntos de referencia de cada figura.

ADAPTACIÓN DE CONTENIDOS

Debido a la necesidad de utilización de distintos tipos de contenidos externos ya existentes, es que se brinda la posibilidad de adaptación de los mismos. Con esto se cumple lo definido en RF11.

Esta adaptación se basa en seguir los siguientes pasos:

1. Convertir el contenido a una clase.
2. Extender a la clase `ContentView`, encargada de brindar el contenido de cada espacio de proyección.
3. El constructor de la clase debe aceptar como parámetros `PApplet container, int canvasWidth, int canvasHeight`.
4. En el constructor agregar la línea aparte de cualquier inicialización que se desee: `super(container, canvasWidth, canvasHeight)`
5. Sobrescribir el método `PImage draw()`. Este método debe retornar una imagen del tipo `PImage`.
6. Sobrescribir el método `void destroy()`.

A continuación se puede ver un ejemplo sencillo donde se muestra una clase encargada de cargar un video.

```
class ContentVideo extends ContentView{
    Movie movie;
    Movie myMovie;

    ContentVideo(PApplet container, int canvasWidth, int canvasHeight) {
        super(container, canvasWidth, canvasHeight);

        this.myMovie = new Movie(container, "video.mp4");
        this.myMovie.loop();
    }

    @Override
    PImage draw() {
        if (this.myMovie.available()) {
            this.myMovie.read();
        }
        return this.myMovie.get();
    }
    @Override
    void destroy(){
      println("Destroy movie");
      this.myMovie.stop();
      this.myMovie.dispose();
    }
```

```
25 }
```

4.2.2 Proyección sobre el cuerpo

La proyección sobre el cuerpo implica la utilización de un cuerpo humano como pantalla sobre la cual se verán desplegados los contenidos deseados.

Al momento de realizar una proyección de este tipo, se desea lograr que el encaje a los contornos del cuerpo sea lo más certero posible.

A su vez, es muy importante evitar retardos en el ajuste de la silueta cuando el cuerpo realiza movimientos, logrando así una sensación agradable para los espectadores.

El primer paso en esta modalidad es detectar la superficie y los contornos del cuerpo sobre el cual proyectar.

Identificación de un cuerpo en escena

Una parte fundamental para lograr la proyección en el cuerpo es encontrar la zona donde se realizará la proyección. Al disponer de las dos cámaras del Kinect, la cámara infrarroja (IR) y la cámara RGB, se posibilitan más de una forma para encontrar esta zona.

En primer lugar, se utiliza la cámara RGB para la detección del cuerpo. Con esta cámara se obtienen buenos resultados en casos donde la iluminación es controlada y los colores de los objetos se mantienen en el tiempo.

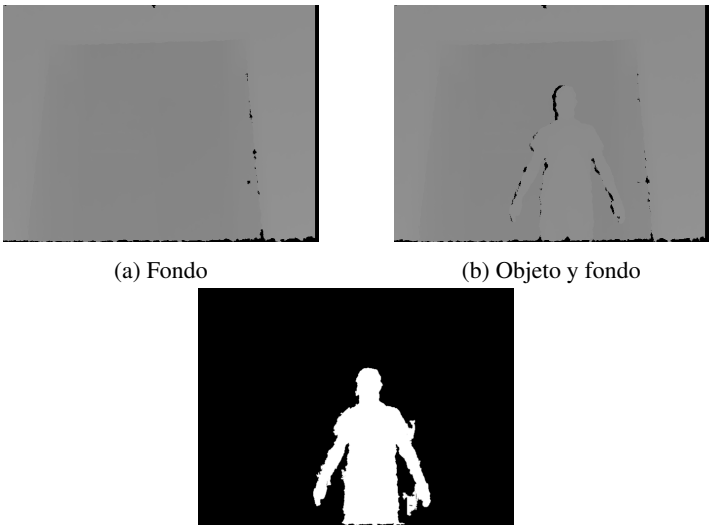
Por otra parte, sabemos que generalmente en obras de danza el dinamismo de la iluminación es una parte importante, lo que hace que la detección utilizando simplemente la cámara RGB se dificulte y no se obtengan buenos resultados.

Por esto se descarta la idea de utilizar la cámara RGB únicamente y se opta por probar distintas técnicas mediante el uso de la cámara infrarroja.

Sustracción de Fondo Para la identificación del cuerpo, se utiliza la técnica de sustracción de fondo. Esta técnica se basa en tener una imagen de la escenografía que se utilizará como fondo fijo y otra imagen con el cuerpo en el escenario. De esta manera al restar las dos imágenes se obtiene como resultado una nueva que contiene únicamente el área del cuerpo.

Cada imagen de la cámara IR es un mapa de puntos donde cada coordenada (x, y) se corresponde con un valor de profundidad en el rango de 0 a 2048.

En definitiva lo que se realiza con esta técnica es restar valores que corresponden a profundidades, obteniendo como resultado un 0 en cada coordenada que permanece inalterada y obteniendo valores positivos en valor absoluto en aquellas coordenadas donde se pueden apreciar nuevos objetos. Estos valores mencionados son teóricos ya que existe un margen de error en las lecturas del sensor por lo que en ciertos casos no es posible obtener 0 como resultado de la resta por más que no se produzcan cambios en escena.



(a) Fondo (b) Objeto y fondo

(c) Objeto

Figura 4.8: Resultado de sustracción de fondo

Como se puede apreciar en la figura 4.8, el resultado es bastante acertado, pero es posible que dependiendo de la composición del escenario, se pueden encontrar falsos positivos. Estos falsos positivos se corresponden a zonas que no pertenecen al objeto buscado pero que aparecen en la imagen resultante, asociados al margen de error antes mencionado. Esto se debe a que la cámara IR del Kinect no es perfecta, principalmente cuando se busca captar superficies brillantes que reflejan los rayos infrarrojos en distintas direcciones.

Para depurar la imagen resultante de la resta, se pueden descartar algunas zonas y puntos que se consideren que no forman parte del cuerpo en cuestión, dado a que son zonas muy pequeñas. En la figura 4.9 se puede apreciar un ejemplo de lo mencionado.

Por más que se descartan la mayor parte de falsos positivos, la imagen resultante aún podría contener zonas no deseadas.

Con el fin de obtener un mejor resultado final es que se investigan alternativas que permitan una mejor detección del cuerpo y nos ayuden a descartar porciones de la imagen que no se corresponden a lo que buscamos.

Es aquí que entra la idea de BLOB, que refiere a los conjuntos de puntos que forman una zona. Esta sigla proviene de las iniciales de **B**inary **L**arge **OB**ject, que se refiere a un grupo de puntos conectados en una imagen.

Para la identificación del cuerpo simplemente se asume que el mismo estará contenido en el BLOB de mayor área, dado que se puede apreciar que el ruido y los falsos positivos siempre

(a) Diferencia con mucho ruido (b) Diferencia mas limpia

Figura 4.9: Depuración

son áreas pequeñas.

Comenzamos por identificar los BLOBs presentes en una imagen y para ello se investigan las siguientes técnicas:

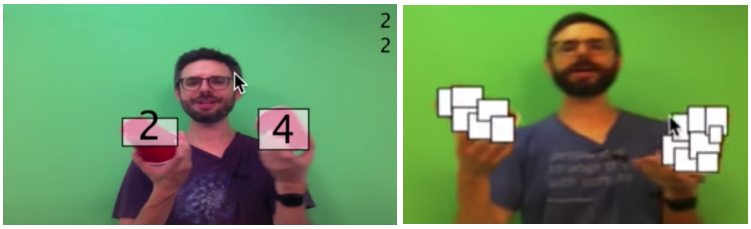
1. Computer Vision: Blob Detection [22]
2. Blob detection by labeling [24]
3. Grass-fire [25]

Computer Vision: Blob Detection

Este algoritmo es utilizado por Shiffman [28] en un video explicativo básico de detección de BLOBs [29]. El procedimiento se basa en recorrer la imagen de izquierda a derecha y de arriba hacia abajo e identificar los puntos pertenecientes a un BLOB.

En el momento que se encuentra un punto que pertenece a un BLOB, para descubrir si es parte de un BLOB ya existente o es uno nuevo, simplemente se calcula la distancia hacia el BLOB más cercano, y si dicha distancia es menor a un umbral, se agrega al BLOB. En caso contrario lo identificamos como un BLOB nuevo. Repetimos el procedimiento hasta que se hayan recorrido todos los puntos de la imagen.

Este método en particular depende fuertemente del umbral que se defina, por lo que en muchos casos se obtienen resultados poco precisos. Como se puede apreciar en la figura 4.10, se tienen dos casos con distintos valores para el umbral antes mencionado, donde en 4.10b el valor utilizado es bajo y genera un resultado de baja precisión.



(a) Detección (b) Detección de baja precisión

Figura 4.10: Computer Vision: Blob Detection

BLOB detection by labeling

Esta técnica se basa en recorrer la imagen píxel por píxel e ir etiquetando numéricamente cada punto perteneciente a un BLOB. Esta etiqueta se calcula de la siguiente manera:

1. Se obtienen las etiquetas de los puntos vecinos que ya fueron etiquetados.
2. Se calcula el mínimo valor en estas etiquetas.
3. Si algún vecino tiene etiqueta, se agrega el punto al BLOB, sino se etiqueta como un nuevo BLOB.

Este proceso encuentra los BLOBs pero puede pasar que encuentre más de los deseados como se puede ver en la imagen 4.11.

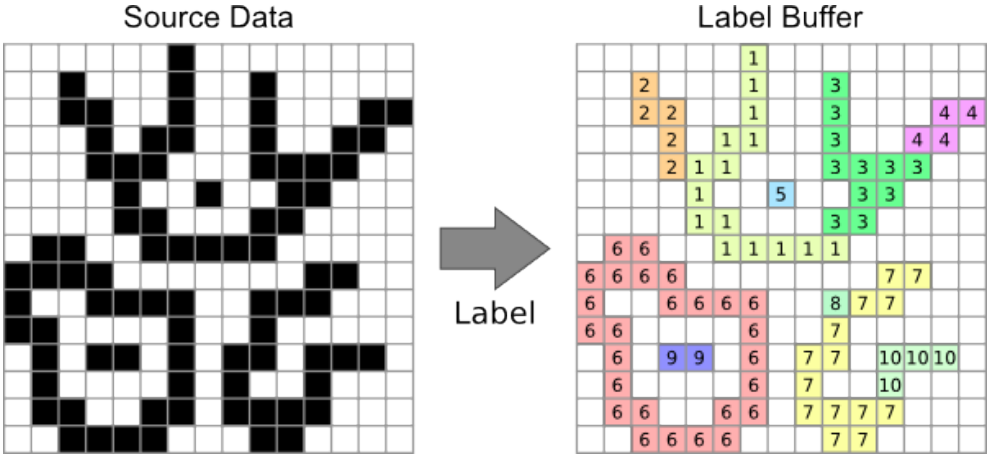


Figura 4.11: BLOBs y etiquetado

Para resolver este problema se recorren los puntos etiquetados, y si son adyacentes y tienen

distinta etiqueta, se actualizan para que se etiqueten de igual forma.

Con este algoritmo se logra obtener una imagen en blanco y negro donde se puede identificar el cuerpo en escena mediante la sustracción de fondo. Si se aplica este método en cada cuadro, la imagen resultante se puede utilizar como máscara y así proyectar sobre el objeto en escena en tiempo real.

Grass-fire

El algoritmo comienza en la esquina superior izquierda y recorre la imagen píxel por píxel de izquierda a derecha y de arriba hacia abajo. En el momento que se encuentra un punto perteneciente a un BLOB, se numera, y se analizan los puntos que se encuentran arriba, a la derecha, abajo, y a la izquierda del mismo.

En el caso que alguno de estos puntos pertenezcan a un BLOB, se numera con el mismo número que el anterior y se comienza el mismo ciclo de analizar los adyacentes. Este proceso se repite hasta que no existan más puntos adyacentes que pertenezcan a un BLOB. En este momento habremos marcado un conjunto de puntos adyacentes que pertenecen a un mismo BLOB.

Luego se continúa el recorrido inicial, salteando los puntos que ya fueron enumerados anteriormente. Para el final del proceso habremos obtenido la lista de BLOBs numerados como se deseaba.

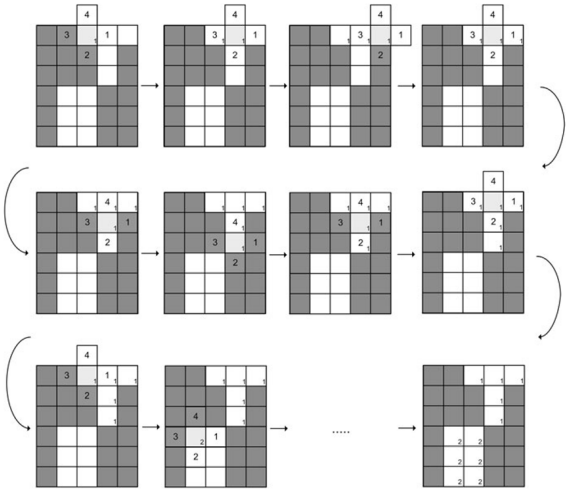


Figura 4.12: Grass Fire

Luego de implementar estas tres técnicas, concluimos que el algoritmo de etiquetado, Blob detection by labeling, resulta ser el que arroja mejores resultados para nuestra solución ya que es la que recorre menos veces los puntos de la imagen.

Esto implica que se requiere menos procesamiento y nos permite trabajar en tiempo real sin demasiados inconvenientes, cumpliendo con el RNF3 - 3.2.3.

Calibración

En la proyección sobre el cuerpo o cualquier objeto en movimiento es importante que exista alineación entre la lectura del Kinect y la salida del proyector. Mediante una calibración se busca alinear los puntos en el espacio que capta la cámara IR del Kinect con lo que se va proyectando. [30]

En la figura 4.13 se puede apreciar lo que capta el Kinect y la imagen proyectada, en el caso que el proyector y el Kinect se encuentran en distintas posiciones. En base a las imágenes captadas y proyectadas podemos entender que se debe realizar una transformación de los valores leídos por el Kinect para luego proyectarlos en la posición correcta.

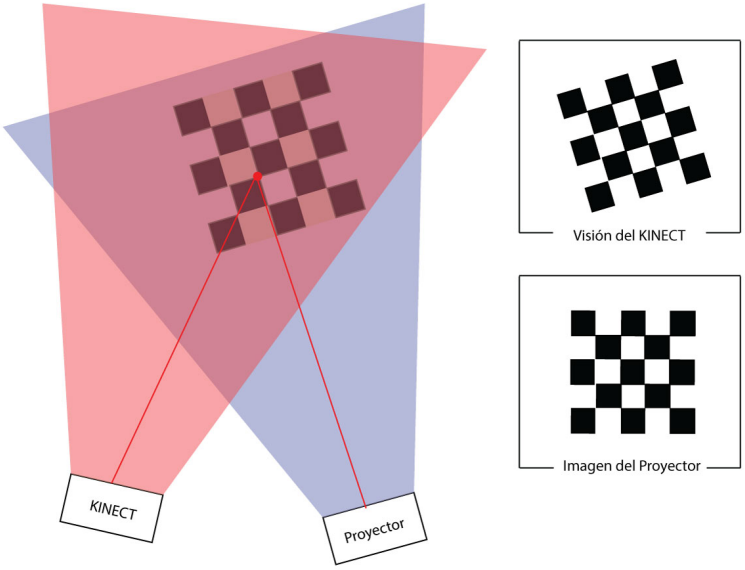


Figura 4.13: Visión del Kinect e imagen del Proyector

Para la resolución de esta problemática, se utiliza la herramienta KinectProjectorToolkit 2.3.

Esta herramienta consiste en utilizar la proyección de una imagen tipo tablero de ajedrez para captarla con la cámara IR del Kinect. El software se encarga de identificar las esquinas que se forman en el tablero y almacena su posición en el espacio.

La idea es cambiar la posición del tablero varias veces de manera de identificar una gran cantidad de puntos espaciados.

Dada serie de puntos el software retornará los valores de calibración para la posición del Kinect, la posición del proyector, y el espacio utilizado.

Luego se utiliza la nube de puntos generada para permitir proyectar los contenidos sobre las

áreas que se desee de forma correcta, incluso sobre el cuerpo en movimiento.

Es importante que una vez obtenida la configuración, no se cambie las posiciones del Kinect y del proyector ya que los parámetros de calibración no serán correctos para nuevas posiciones.

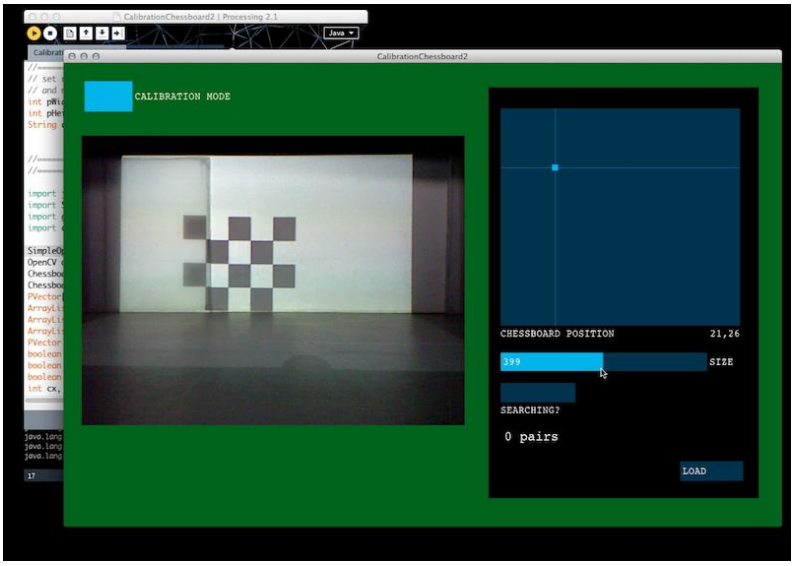


Figura 4.14: Calibración con KinectProjectorToolkit

La figura 4.14 muestra la interfaz de la herramienta donde se proyecta un tablero de ajedrez para identificar puntos en el espacio y así poder obtener la transformación necesaria para mantener el Kinect y el proyector calibrados entre sí.

4.3 Problemas encontrados

A lo largo del proyecto, se encontraron algunos problemas que no afectaron al proceso de desarrollo, pero tenían un impacto sobre la calidad del producto final.

1. Fotogramas por segundo
2. Transiciones

Es por esto que se deciden realizar algunas pruebas específicas con el fin de recabar información detallada y así poder tomar acciones y brindar recomendaciones para no incurrir en ciertos problemas.

Fotogramas por segundo

El término fotogramas por segundo (FPS) se refiere a la cantidad de imágenes o cuadros que se muestran a lo largo de un segundo en videos. A mayor cantidad de FPS, mayor será la sensación de fluidez que perciba el observador, y por lo general, una visualización por encima de los 24 FPS se considera aceptable al ojo humano.

En el correr del proceso de desarrollo y en la adaptación de los contenidos para el caso de estudio, es que identificamos que la carga de videos y la simultaneidad en que estos se muestran en una escena tiene un fuerte impacto sobre los FPS.

Al ser un tipo de contenido que intuimos se utilizará muy a menudo decidimos realizar una serie de pruebas para inferir qué tipo de videos serían los óptimos para usar.

Para comenzar, obtuvimos 4 videos de distintas resoluciones, pesos, y duraciones para tener un espectro amplio de muestra.

Nombre de video	Peso	Tamaño (px)	Duración (min:seg)
1.mp4	31.5 MB	1280x720	2:51
2.mp4	95 MB	854x480	22:21
3.mp4	62.9 MB	320x240	23:03
4.mp4	89 KB	256x144	0:04

Luego realizamos las pruebas evaluando los FPS, variando la cantidad de pantallas en escena, y la cantidad y tipo de videos. A su vez evaluamos si el ajuste del tamaño (resize) del video en tiempo real es un factor que deberíamos tener en cuenta.

Nº de prueba	Cantidad de pantallas	Cantidad de videos	FPS	FPS Sin resize
1	0	0	60	60
2	1	1 (1.mp4)	25	31
3	2	2 (1.mp4)	15	18
4	3	3 (1.mp4)	10	12
5	1	1 (2.mp4)	33	37
6	2	2 (2.mp4)	23	25
7	3	3 (2.mp4)	18	20
8	1	1 (3.mp4)	42	44
9	2	2 (3.mp4)	28	30
10	3	3 (3.mp4)	23	24
11	3	1 (4.mp4)	42	4
12	1	2 (4.mp4)	28	31
13	2	3 (4.mp4)	23	24
14	7	7 (1.mp4)	2	2
15	7	7 (2.mp4)	5	7
16	7	7 (3.mp4)	12	13
17	7	7 (4.mp4)	12	12

La prueba número 1 es de control para confirmar que sin videos en escena obtenemos 60 FPS que es la cantidad de fotogramas por defecto.

Luego, se advierte que si bien el hecho del redimensionado tiene un impacto sobre los FPS, no es el aspecto de mayor preocupación.

Se nota que tampoco lo es el peso del video ni la duración ya que estos recursos se precargan al comenzar con la ejecución del programa.

Se puede ver finalmente que las dimensiones de los videos en términos de resolución, como también la cantidad de videos en pantalla, son los factores que mayor impacto presentan sobre la cantidad de FPS finales resultantes.

Es por esto que se recomienda que los videos que se utilicen sean de una resolución acorde y que se evalúe la cantidad de videos a utilizar en base al rendimiento que se obtiene.

Shaders[31]

Por otro lado, también se aprecia que ciertos contenidos generados (aparte de los videos) también generan un impacto en la cantidad de FPS, reduciendo drásticamente el rendimiento. Se analizan las razones por las que se puede dar este fenómeno y se concluye que se debe a la complejidad en el dibujado en pantalla de los contenidos. Lo que viene dado por la gran cantidad de cálculos necesarios para lograr la imagen final.

Es por esto que se investiga el manejo de Shaders por parte de Processing. Los Shaders son programas que corren en GPU y generan una salida visual determinada permitiendo extender la capacidad de dibujar de Processing. Mediante su uso se puede lograr crear escenas 3D

muy sofisticadas en términos de luces y texturado, como también aplicar efectos en post-procesamiento de imágenes en tiempo real, entre otras posibilidades. Todo esto lo logra mediante la creación de objetos complejos que con otras técnicas serían imposible de generar.

En el anexo C se pueden apreciar un ejemplos de Shaders.

Se realizan pruebas utilizando Shaders personalizados en lugar de los utilizados por Processing de forma predeterminada. El resultado final resulta en que, para los contenidos trabajados, no se encuentran Shaders que tengan mejor rendimiento que los establecidos por defecto y por ende utilizados por Processing en lo que refiere a FPS.

TRANSICIONES

Con el fin de mejorar la experiencia a los espectadores de la obra que se está generando, es que se investiga la posibilidad de utilizar transiciones personalizadas entre las distintas escenas y sus contenidos.

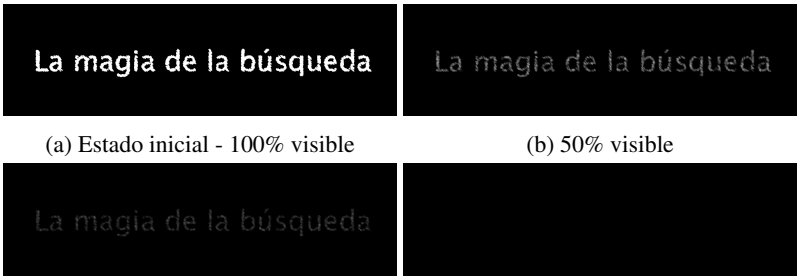
Se hace foco en dos puntos principales, por un lado las diferentes posibilidades de transición desde lo visual y por otro lado, la simplicidad en su implementación y utilización.

En primer lugar se trabaja en las posibles variantes o alternativas para generar transiciones, definiendo una lista de efectos deseables:

1. **Fade Out** - desaparición del contenido suavemente, pasando de 100% a 0% de visibilidad.
2. **Fade In** - aparición del contenido suavemente, pasando de 0% de visibilidad a 100%.
3. **Coloreado** - pintado de pantalla de un color plano hasta completar la totalidad de la misma.
4. **Pixelado** - efecto de pixelado del contenido que se incrementa/decrementa.

Se realizan pruebas de cada uno de los efectos en contenidos puntuales con el fin de evaluar su complejidad de implementación.

En la figura 4.15 se puede apreciar la prueba realizada sobre un contenido donde varias partículas forman el texto "La magia de la búsqueda" aplicando los efectos **Fade Out** y **Fade In**. Vale destacar que ambos efectos son opuestos en cuanto al manejo del porcentaje de opacidad por lo que su implementación se puede realizar en conjunto.



(a) Estado inicial - 100% visible (b) 50% visible

(c) 10% visible (d) 0% visible

Figura 4.15: Evolución de transiciones - Fade Out / Fade In

En la figura 4.16 se puede apreciar la evolución en la aplicación del efecto que colorea una imagen, **Coloreado**. Tanto en este caso como en el **Pixelado**, figura 4.17, es que se comienza a apreciar que debe existir relación entre las transiciones de salida y entrada de contenidos consecutivos que permitan un cambio agradable entre escenas.

(a) Pantalla sin colorear (b) Progresión (c) Pantalla coloreada

Figura 4.16: Evolución de transición de efecto de coloreado

Luego de evaluar el esfuerzo en la implementación de cada caso, se denota que si bien es posible generar transiciones de forma genérica o global, es mejor implementar los diferentes efectos en cada uno de los contenidos generados.

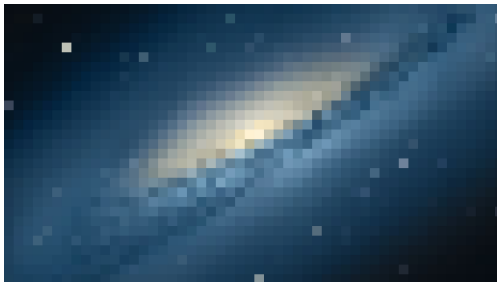
Esto se debe a que en cada transición se manipula el tipo de contenido de forma diferente y al implementarlas de manera global sólo se está trabajando sobre el resultado obtenido en cada paso. Una solución que se presenta es generar modificaciones a la generación de los contenidos personalizados para que se adapten a la utilización de las transiciones.

Otro inconveniente que surge de lo antes mencionado es que la transición de salida de un contenido y la transición de entrada del contenido siguiente deben presentar coherencia para evitar saltos abruptos y efectos visuales poco agradables para los espectadores.

Al momento de realizar pruebas de transiciones con los contenidos que Cristina de Espacio Pangea deseaba utilizar, es que se nota la necesidad de que la duración de cada transición depende del momento en que se utiliza. Esta forma de utilización agrega complejidad a la implementación por lo que requiere más tiempo y dedicación.

(a) Estado inicial

(b) Tamaño de bloque 50px



(c) Tamaño de bloque 100px

Figura 4.17: Evolución de transición de efecto de Pixelado

Por razones de tiempo se hace imposible avanzar con dicho desarrollo por lo que se detallan en la sección 6.2 todas la ideas que se plantean referentes a las transiciones para su trabajo a futuro.

5 Aplicación práctica

El caso de estudio se basa en un obra de danza realizada en conjunto con la bailarina Cristina Aguirre de Espacio Pangea.

5.1 Herramientas utilizadas

Para el desarrollo y la utilización de la solución se emplean las siguientes herramientas:

- Sensor Kinect [A.1]
- Proyector gran angular Hitachi Maxell CP-CX301WN [A.2]
- Proyector Viewsonic PA503W [A.3]
- Computadora Macbook pro [A.4]

5.2 Implementación

Desde el comienzo se mantienen reuniones semanales donde se evalúa el progreso de la solución, se discute el contenido que se debe incluir y se define la disposición deseada de la escenografía.

Se utiliza una sala lo suficientemente grande y vacía de Espacio Pangea que sirve de forma ideal para aplicar nuestra solución. Estas condiciones hacen que el Kinect funcione de forma óptima.

A este espacio se le agrega una telón a modo de pantalla para el proyector. De la forma que

se dispone el telón, se genera un espacio de 1,5 metros aproximadamente detrás del mismo. Esto brinda la oportunidad de probar una retro proyección con un proyector gran angular, con un resultado muy positivo como se puede apreciar en la figura 5.1

Figura 5.1: Espacio Pangea - telón con retroproyección

A lo largo de las reuniones se prueban distintas posiciones para el Kinect y proyector, así como diferentes técnicas para lograr una mejor experiencia para los espectadores.

Estas pruebas incluyen distintos tipos de materiales para el telón, la utilización de un sobretelón semi-transparente (Figura 5.2), que el telón tenga ondas, simultaneidad en la proyección frontal y retroproyección, entre otras variantes.

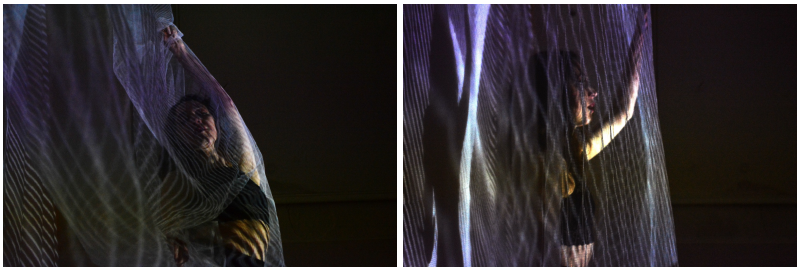


Figura 5.2: Pruebas con sobretelón de tul

A pesar de que el objetivo del proyecto es brindar una herramienta a las personas involucradas en una obra de danza, nos encontramos en la situación de proveer contenidos para la obra a realizar ya que al momento no se dispone de ejemplos para que Cristina pueda idear su obra.

Es por esto que se reúnen contenidos realizados en materias dictadas por Ewelina Bakala, de manera de mostrar las potencialidades y limitaciones que podría tener la herramienta. Se pueden apreciar las pruebas realizadas con contenidos existentes en la figura 5.3.

Cabe destacar que varios de estos contenidos utilizados para las pruebas fueron adaptados para su utilización en la obra final.

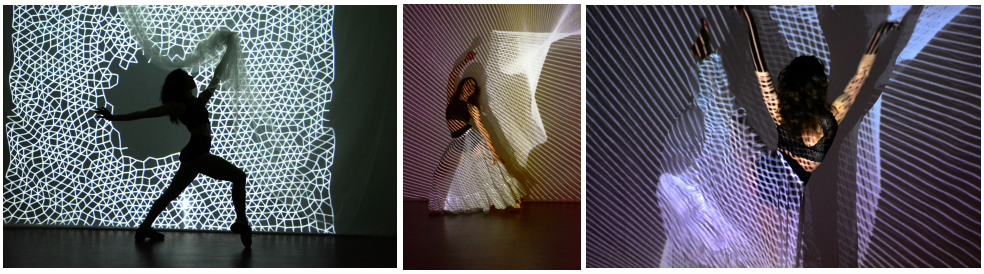


Figura 5.3: Prueba de contenidos

Luego de que Cristina ideara la obra, se procede al ajuste de los contenidos para la misma y a la preparación de la muestra.

En la obra se utiliza una configuración de cinco escenas y una única zona de proyección en cada una que ocupaba la totalidad de la amplitud del proyector. A continuación se detallan los contenidos asociados a cada escena.

5.2.1 Escenas

Escena 1 - Nube de puntos formando "La magia de la búsqueda". La frase permanece estática cuando no hay cuerpos en escena y cuando un cuerpo aparece en escena, la nube de puntos orbita sobre el punto medio del cuerpo detectado. Ejemplo en la figura 5.4.

(a) Estado inicial (b) Partículas orbitando

Figura 5.4: Escena 1

Escena 2 - Video de estrellas en movimiento. Se muestran algunas capturas del video utilizado en la figura 5.5 para que se entienda la estética utilizada.

Escena 3 - Líneas desde los bordes del área de proyección en dirección al cuerpo, como se puede apreciar en la figura 5.6 con la transición incluída.

Escena 4 - Puntos unidos entre sí que gravitan hacia o desde el cuerpo y hacia el piso. Se muestran las distintas etapas de esta escena en las figuras 5.7

Escena 5 - Máscara con el cuerpo que destapa una imagen y perdura de forma progresiva. Se pueden ver capturas de esta escena en la figura 5.8.

Figura 5.5: Escena 2

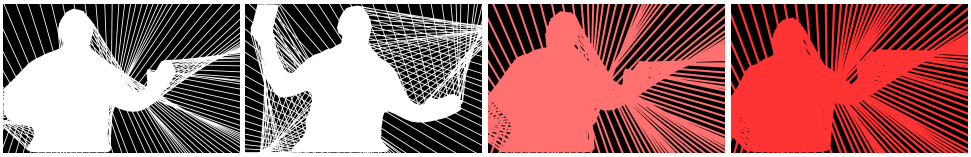


Figura 5.6: Escena 3

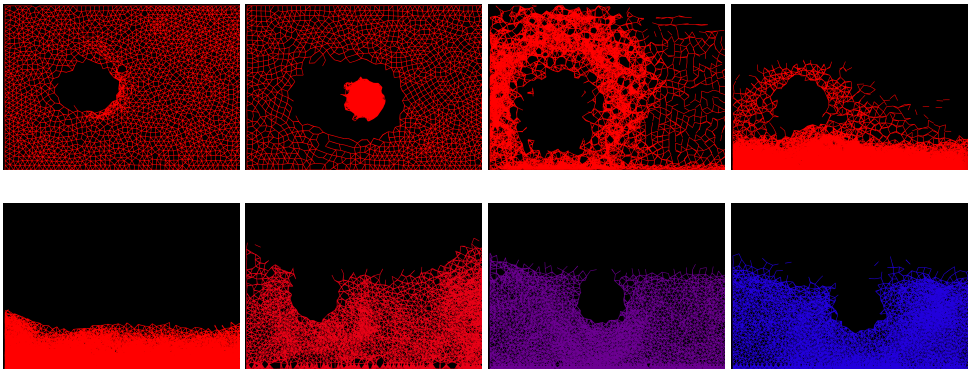


Figura 5.7: Escena 4

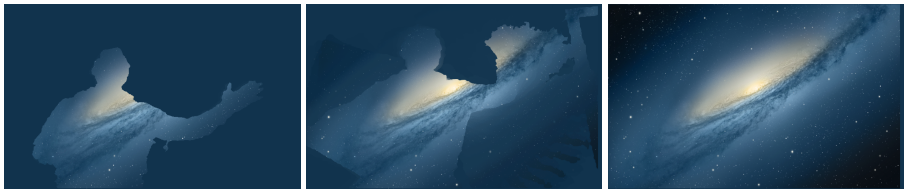


Figura 5.8: Escena 5

Cada una de las escenas se compone de contenido realizado exclusivamente para la obra, alcanzando una duración total de 10 minutos aproximadamente.

El proceso de definición de la obra se extiende durante aproximadamente dos meses, donde se realizan uno o más ensayos semanales. En cada ensayo se efectúan ajustes para que la obra culmine de la forma deseada.

La muestra se lleva a cabo los días 13 y 15 de diciembre de 2018 en el marco de eventos de fin de año del Espacio Pangea, con la presencia de un público de aproximadamente 50 personas por día. La misma se puede ver en este video publicado en Youtube[1].

Se pasa a mostrar parte del resultado de la ejecución de la obra, mediante capturas de cada una de las escenas antes descritas.

En la figura 5.9 se puede apreciar la escena 1, nube de puntos que forman una frase y que luego orbitan sobre el cuerpo detectado. Luego se puede identificar en la figura 5.10 el video desplegado en la escena 2. En las figuras 5.11 y 5.13 se muestran imágenes de las escenas 3 y 4 respectivamente, que corresponden a contenidos que se basan en la detección del cuerpo para permitir la interacción. Finalmente, la figura 5.15 enseña la última escena de la muestra, la número 5, donde se utiliza el contorno del cuerpo para generar una máscara que permite destapar una imagen de fondo.

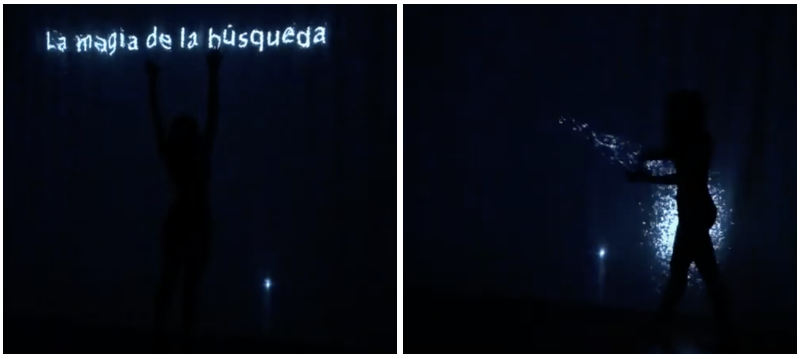


Figura 5.9: Escena 1

[1] URL del video : https://www.youtube.com/watch?v=QGPZd04b5Qs

Figura 5.10: Escena 2

Figura 5.11: Escena 3

Figura 5.12: Resultado de la ejecución de los contenidos

Figura 5.13: Escena 4

Figura 5.14: Resultado de la ejecución de los contenidos

Figura 5.15: Escena 5

6 Conclusiones y trabajo a futuro

En este capítulo se detallan las conclusiones obtenidas como resultado de la implementación y ejecución de la solución planteada.

A su vez, se mencionan las principales líneas de trabajo a futuro, explicando las razones por las que no se incluyen en el alcance de este proyecto.

6.1 Objetivos alcanzados

Para comenzar, se puede afirmar que se cumplieron los objetivos planteados al comienzo del proyecto ya que se logra desarrollar una herramienta que permite generar la configuración necesaria para una obra en lo que respecta a creación de áreas de proyección, asignación y carga de contenidos. Asimismo, se logra una solución eficiente en cuanto a la detección y proyección sobre un cuerpo en movimiento.

En cuanto a la herramienta de configuración, la solución lograda cumple con los objetivos específicos desde el 1 al 4. Con esto se permite la creación y edición de archivos de configuración que luego pueden ser ejecutados con la herramienta correspondiente.

Durante la implementación de ésta herramienta se hace gran foco en la usabilidad de la misma, con lo que se define la interfaz de usuario de la forma más amigable posible siguiendo recomendaciones de todos los involucrados. La mayoría de estas recomendaciones surgen tras realizar pruebas de uso con varios de los involucrados a lo largo del proyecto.

Se presentaron ciertos inconvenientes técnicos durante el desarrollo, como lo son la cantidad de FPS, el enmascarado de los contenidos a proyectar, la calibración entre el proyector y el sensor Kinect, los cuales fueron resueltos de forma eficiente tal como se puede apreciar en el capítulo 4, más específicamente en los apartados 4.2.1, 4.2.2 y 4.3.

Tanto para la detección del cuerpo en movimiento y proyección sobre el mismo, se considera que la solución alcanzada logra un buen desempeño al no presentar retardos en tiempo real y una adaptación a los contornos del cuerpo bastante exacta, cumpliendo con el quinto de los objetivos específicos planteados.

Por otra parte, en algunas de la soluciones alcanzadas se denota que es posible buscar mejoras, las cuales se detallan en la sección de trabajo a futuro 6.2.

Finalmente, podemos concluir que se lograron conseguir herramientas y soluciones acorde a los objetivos planteados y que cumplen con los requisitos detallados. Las herramientas brindadas permiten cumplir con las etapas de creación, configuración, y ejecución de obras interactivas, de forma simple y sin necesidad de incurrir en costos asociados al software.

6.2 Trabajo a futuro

FPS La cantidad de FPS que se logra está directamente relacionada con la cantidad y complejidad de los contenidos mostrados simultáneamente, así como con el hardware que se esté utilizando. A pesar de esto, creemos que es posible mejorarlo optimizando el código fuente y cómo se muestran los contenidos en pantalla. A pesar de que el uso de shaders por defecto que utiliza Processing es bueno, posiblemente para cierto tipo de contenidos, puedan existir

shaders específicos que permitan mejorar los FPS.

Interfaz del manejador de escenas Para definir la interfaz de la herramienta se utilizó la librería ControlP5[26] con el fin de presentar de la forma más amigable posible los controles necesarios. La librería, a pesar de brindar estos controles haciendo que su uso sea sencillo, no provee una interfaz gráfica demasiado atractiva visualmente. Creemos que debería tomar una mayor importancia que la herramienta sea visualmente atractiva así como usable.

Suavizar bordes en sustracción de fondo La técnica de sustracción de fondo utilizada genera un conjunto de puntos que al unirlos forma la figura deseada. Debido a que lo captado por la cámara IR del Kinect en cada frame difiere ligeramente, los puntos de la sustracción de fondo generan figuras con bordes irregulares. Esto incrementa si existen elementos que reflejen los rayos infrarrojos, como el pelo, la lana, etc. La unión de estos puntos se podría realizar de manera más curva y así evitar ángulos que no definan al objeto correctamente.

Transiciones Siguiendo lo detallado en la sección 4.3, queda para trabajar a futuro la implementación de transiciones ya sea de forma global a la herramienta o independiente en cada uno de los contenidos a utilizar. En este sentido es importante mantener una coherencia entre la transición de salida de cada contenido con la transición de entrada del que le sigue. A su vez, sería bueno poder manipular la duración de cada una de las transiciones para permitir una mayor personalización de la obra generada.

Referencias

[1] Projection Ballet - dancing routine lighting effects (fx) using projection DMX.
URL: https://www.youtube.com/watch?v=JA6kl4RlA0I

[2] KINECT | PROJECTOR DANCE.
URL: https://www.youtube.com/watch?v=FnulH8TrZVo

[3] Processing.
URL: https://processing.org

[4] Openframeworks.
URL: https://openframeworks.cc

[5] P5js.
URL: https://p5js.org

[6] Cider.
URL: https://libcinder.org

[7] MAX.
URL: https://cycling74.com/products/max

[8] Codea.
URL: https://codea.io

[9] Keystone.
URL: http://www.deadpixel.ca/keystone

[10] Sketch Mapper.
URL: http://josephtaylor.github.io/sketch-mapper

[11] Surfacce Mapper GUI.
URL: http://jasonwebb.io/2013/11/surfacemappergui-a-simple-processing-interface-for-projection-mapping

[12] SurfaceMapper by Ixagon.
URL: https:

//josephtaylor.github.io/sketch-mapper/SketchMapper/reference/ixagon/surface/mapper/SurfaceMapper.html

[13] Kinect.
URL:
https://azure.microsoft.com/en-us/services/kinect-dk

[14] SimpleOpenNI.
URL: https://github.com/totovr/SimpleOpenNI

[15] OpenNI.
URL: https://openni.org

[16] NITE.
URL: http://www.dfki.de/nite

[17] Open Kinect for Processing.
URL: https://github.com/shiffman/OpenKinect-for-Processing

[18] Libfreenect.
URL: https://github.com/OpenKinect/libfreenect

[19] KinectProjectorToolkit.
URL:
https://github.com/genekogan/KinectProjectorToolkit

[20] OpenCV for Processing.
URL: https://opencv.org/about

[21] OpenCV for Processing.
URL: https://github.com/atduskgreg/opencv-processing

[22] Blob detection by Shiffman.
URL: https://www.youtube.com/watch?v=ce-2l2wRqO8

[23] Luciano Gervasoni, Juan Pablo D'amato, Rosana Barbuzza. Método de sustracción de fondo a partir de imágenes de video-vigilancia. (15th Argentine Symposium on Technology, AST 2014)
URL: http://sedici.unlp.edu.ar/bitstream/handle/10915/41763/Documento_completo.pdf?sequence=1&isAllowed=y

[24] Blob detection by labeling.
URL: http://www.labbookpages.co.uk/software/imgProc/blobDetection.html

[25] Blob detection. Grassfire algorithm.
URL: http://what-when-how.com/introduction-to-video-

and-image-processing/blob-analysis-introduction-to-video-and-image-processing-part-

[26] Controlp5.
URL: https://github.com/sojamo/controlp5

[27] JSON.
URL: https://www.json.org

[28] Daniel Shiffman. Sitio oficial.
URL: https://shiffman.net

[29] Daniel Shiffman. Detección de BLOBs.
URL: https://www.youtube.com/watch?v=ce-2l2wRqO8

[30] Kinect Projection Mapping.
URL: https://pdfs.semanticscholar.org/7bf7/c9a841a751deca895a009954eb3677e1be88.pdf

[31] Shaders en Processing.
URL: https://processing.org/tutorials/pshader/

Anexos

A Herramientas utilizadas

A.1 Sensor Kinect

El sensor Kinect es un accesorio de la consola de juegos XBox 360 de Microsoft. Se creó con el propósito de funcionar como un controlador que identificara los movimientos del jugador y actuara en consecuencia.

Figura A.1: Kinect 1414

En este proyecto utilizamos el modelo 1414 ya que era el disponible al momento. Este modelo captura imágenes de 640*480 pixels a 30 imágenes por segundo en sus cámaras.

Este dispositivo esta compuesto por:

- 1-3: Cámara infraroja (IR)
- 2: Cámara de color (RGB)
- 4: Motor de inclinación
- 5: Micrófonos

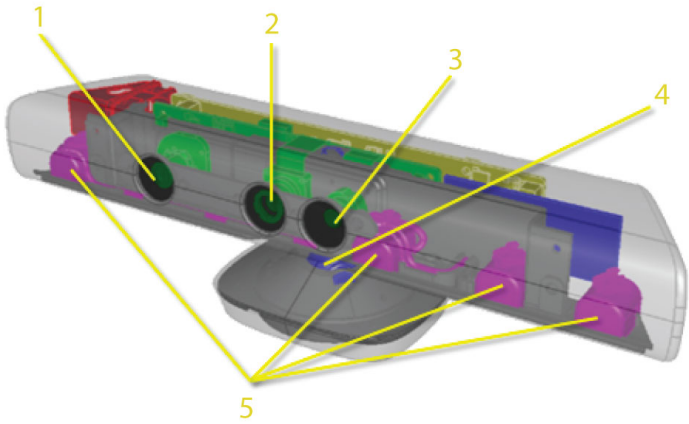


Figura A.2: Composición del Kinect

A.2 Proyector gran angular Hitachi Maxell CP-CX301WN

Resolución de 1024x768
Lente Enfoque manual, zoom digital x 1,35
Distancia de enfoque 0,691 m 1,838 m
Contraste 3000:1

A.3 Proyector Viewsonic PA503W

Resolución 1280x800
Lente Enfoque manual, zoom digital 1.1x
Distancia de enfoque 1m-10.98m
Contraste 22.000:1
Compatibilidad de Video: NTSC, PAL, SECAM
Frecuencia horizontal: 15K 102KHz
Vertical Scan Rate: 23 120Hz

A.4 Computadora Macbook Pro

Para la presentación utilizamos una Macbook Pro 13" early 2015 con las siguientes características:

Procesador: 2,7 GHz Intel Core i5 de dos núcleos
Memoria RAM: 8 GB 1867 MHz DDR3
Tarjeta gráfica: Intel Iris Graphics 6100 1536 MB
Disco duro: 256 GB SSD

B Manual de uso

B.1 Descargar herramientas

Las distintas partes que componen la solución lograda se pueden descargar desde un repositorio [1] exclusivo, el cual contiene:

1. Manejador de escenas / Herramienta de configuración
2. Herramienta de ejecución
3. Ejemplos de contenidos para utilizar
4. Código fuente de cada solución

Para la ejecución de las herramientas se debe seleccionar la que se corresponda con el sistema operativo en el que se va a ejecutar, teniendo disponible versiones para Windows, MacOS y Linux.

B.2 Manejador de escenas

Este módulo se corresponde con la herramienta que nos permite generar la configuración de la obra a ejecutar.

Al iniciar su ejecución se puede apreciar la ventana inicial donde se permite especificar las dimensiones en píxeles del lienzo sobre el que se definirán las pantallas de cada escena. A

[1] Repositorio: https://gitlab.com/proyecto-de-grado-fing/soluciones-interactivas-para-la-danza

su vez, se permite tanto la creación de una nueva configuración como también la carga de un archivo de configuración ya existente. En la figura B.1 se pueden percibir los detalles antes mencionados.

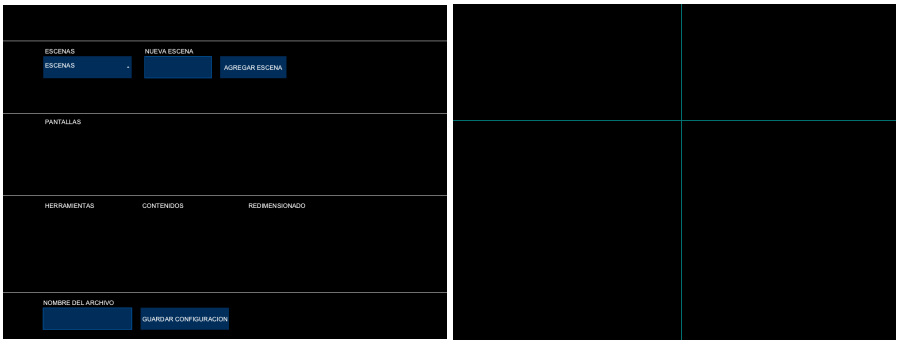
Figura B.1: Pantalla inicial del manejador de escenas

Una vez seleccionados el ancho y alto del lienzo, se pasa a tener 2 ventanas, una donde se muestran los controles y en la otra el lienzo que se debe proyectar para facilitar la definición de las áreas de proyección. En la figura B.3 se muestran ambas ventanas, donde B.3a se corresponde a la interfaz con controles (llamada controles) para generar la configuración deseada y en B.2.1 tenemos el lienzo sobre el cual definiremos las figuras asociadas a cada pantalla (área de proyección).

B.2.1 Gestión de escenas

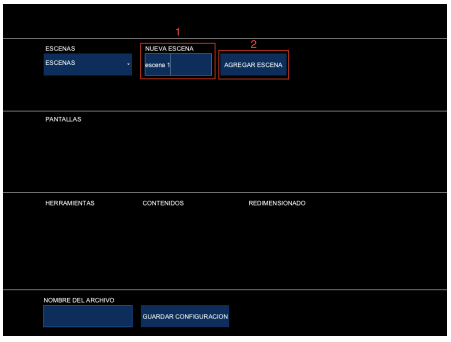
En la parte superior de la ventana correspondiente a la interfaz de controles es donde se muestran los controles que permiten la gestión de escenas. Esta gestión incluye el agregado y eliminación de escenas.

Para agregar una escena se debe ingresar el nombre que la identificará y luego hacer click en el botón con la leyenda *Agregar Escena*. En la figura B.3a se detalla el orden de los pasos antes mencionados.



(a) Controles (b) Lienzo

Figura B.2: Ventanas del manejador de escenas



(a) Pasos para agregar una escena

(b) Listado con la nueva escena

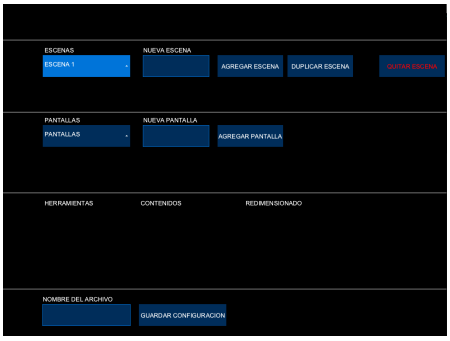
Figura B.3: Agregado de escena

Luego de agregar una escena, la misma se desplegará en el listado de escenas, tal como se puede apreciar en la figura .

Una vez se selecciona una escena del listado y tal como se muestra en la figura B.4a, se se habilitan los botones que permiten duplicar y eliminar la escena en cuestión.

También se puede apreciar, gracias a lo detallado en la figura B.4b, que se habilita la gestión de pantallas o áreas de proyección asociados a la escena, que se explica más adelante.

(a) Botones asociados a la escena



(b) Gestión de pantallas habilitada

Figura B.4: Ventanas del manejador de escenas

B.2.2 Gestión de pantallas

Las pantallas o áreas de proyección se gestionan de forma similar a lo ya explicado para las escenas. Permitiendo agregar o quitar pantallas asociadas a una escena. Adicionalmente se permite, mediante el botón *Limpiar*, eliminar todas las figuras definidas para la pantalla que estamos definiendo/editando. Todos estos detalles se pueden apreciar en la figura B.5.

(a) Agregar pantalla

(b) Botones asociados a la pantalla

Figura B.5: Gestión de pantallas

Luego al definir las figuras asociadas a la pantalla podremos seleccionar el contenido que se desea asignar y la forma en que éste se ajustará para ser proyectado.

B.2.3 Gestión de figuras y contenidos

Una vez se selecciona una pantalla se podrá comenzar a definir las figuras que definirán el espacio a utilizar como área de proyección.

Tal como podemos ver en la figura B.6, podemos dividir esta parte de los controles en cuatro partes:

1. **Herramientas / Figuras** - mediante el uso de polígonos y elipses se permite delimitar el área de proyección que se desee. Mediante la herramienta Mouse se permite la edición de las figuras para poder ajustar su posición y sus dimensiones.

2. **Contenidos** - permite asignar un contenido específico a la pantalla en cuestión.
3. **Redimensionado** - permite especificar de que forma se ajusta el contenido dentro del área de proyección definida, tal como se detalla en 4.2.1.
4. **Previsualización** - permite previsualizar el contenido seleccionado en la pantalla definida. Más detalles en la subsección B.2.4

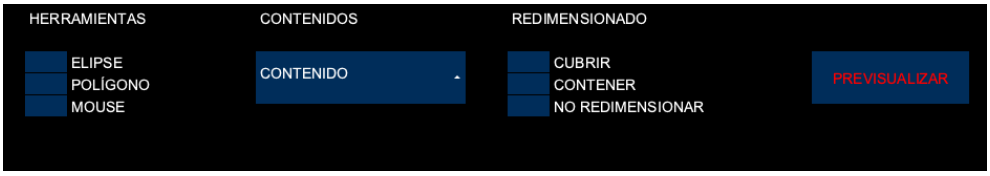
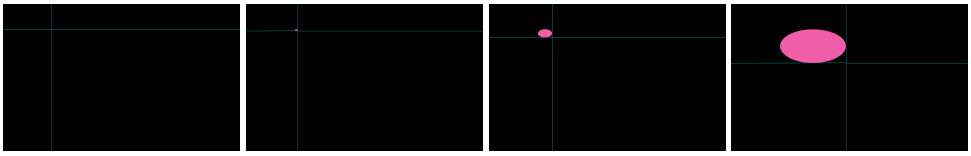


Figura B.6: Gestión de figuras y contenidos

Para dibujar una elipse se debe hacer click y mantener presionado mientras se mueve el cursor y se va definiendo la figura. En el caso del polígono, mediante sucesivos clicks se definen cada uno de los vértices. Para cerrarlo se debe hacer doble click en el último vértice.

Se pueden apreciar ejemplos en cuanto a la utilización de elipses y polígonos en las figuras B.7 y B.8 respectivamente.



(a) Lienzo vacío (b) Definiendo elipse (c) Definiendo elipse (d) Finalización

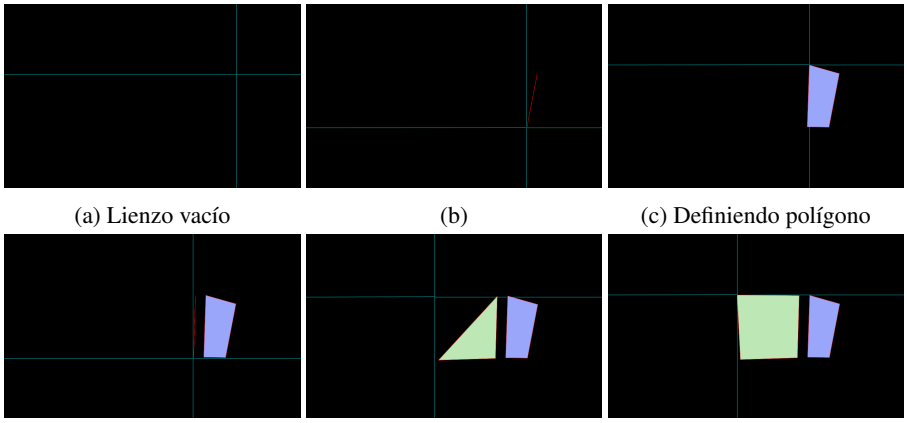
Figura B.7: Definiendo elipse

B.2.4 Previsualización

Se permite previsualizar el contenido seleccionado dentro de la pantalla definida mediante el botón con leyenda *Previsualizar* que ya fue mencionado sobre la figura B.6.

Una vez que el contenido se este previsualizando, la leyenda del botón *Previsualizar* cambiará a *Detener*, permitiendo parar la previsualización del contenido para poder continuar con la edición de la configuración.

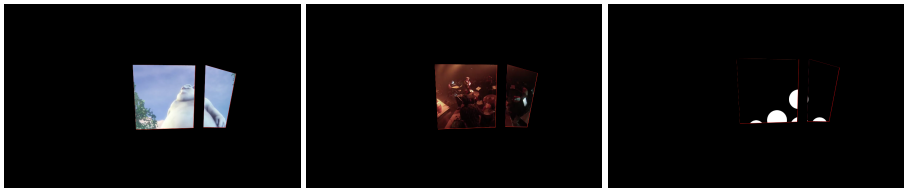
Se pueden apreciar ejemplos de previsualización de distintos contenidos sobre la pantalla definida con polígonos B.8



(a) Lienzo vacío (b) (c) Definiendo polígono

(d) Polígono finalizado (e) Definiendo otro polígono (f) Finalización

Figura B.8: Definiendo polígonos



(a) Video animado (b) Video recital (c) Contenido personalizado

Figura B.9: Previsualización

B.2.5 Guardado de datos de configuración

Una vez finalizada la creación de la configuración, la misma se puede guardar en formato JSON siguiendo la estructura detallada en B.2.5.

En la figura B.10 se puede ver que para realizar la acción de guardado se debe introducir un nombre para el archivo de configuración y luego hacer click en el botón con leyenda *Guardar Configuración*.

Una vez la configuración queda guardada de forma correcta se desplegará en verde el mensaje de exito correspondiente, tal se aprecia en la figura B.10b

(a) Controles para guardar

(b) Guardado exitosamente

Figura B.10: Guardado de la configuración

Ejemplo de datos en formato JSON

```
{
  "canvasWidth": 1024,
  "canvasHeight": 768,
  "scenes": [{
    "name": "Escena 1",
    "screens": [{
      "contentBottomRight": {
        "x": 823,
        "y": 378
      },
      "color": 255,
      "shapes": [
        {
          "shapeType": "Polygon",
          "color": -13593118,
          "coords": [
            {
              "x": 823,
              "y": 271
            },
            {
              "x": 696,
              "y": 378
            },
            {
              "x": 141,
              "y": 286
            },
            {
              "x": 443,
              "y": 126
            }
          ]
        },
        {
          "shapeType": "Polygon",
          "color": -5891098,
          "coords": [{
            "x": 443,
            "y": 126
          }]
        }
      ],
```

```
      "name": "d",
      "contentClass": "ContentVideo",
      "contentTopLeft": {
        "x": 141,
        "y": 126
      },
      "resizeMode": "Cover",
      "mask": "data/das/maskE0S0.tif"
    }],
  }]
}
```

B.3 Herramienta de ejecución

Esta herramienta permite llevar a cabo la ejecución de la obra configurada mediante la carga de un archivo de configuración. El archivo a cargar es el ya explicado en el apartado anterior.

Una vez cargada la configuración se cuentan con las siguientes acciones posibles:

1. **Comenzar la ejecución**: se consigue al presionar la tecla S (start).
2. **Detener la ejecución y volver al comienzo**: mediante la tecla R (restart).
3. **Avanzar/retroceder entre escenas**: se consigue utilizando las flechas del teclado, $\leftarrow$ para retroceder a la escena anterior y $\rightarrow$ para avanzar a la siguiente.

C Shaders

C.1 Ejemplo - Esfera

Código del Programa

```
PShader toon;

void setup() {
  size(640, 360, P3D);
  noStroke();
  fill(204);
  toon = loadShader("frag.glsl", "vert.glsl");
  toon.set("fraction", 1.0);
}

void draw() {
  shader(toon);
  background(0);
  float dirY = (mouseY / float(height) - 0.5) * 2;
  float dirX = (mouseX / float(width) - 0.5) * 2;
  directionalLight(204, 204, 204, -dirX, -dirY, -1);
  translate(width/2, height/2);
  sphere(120);
  println(frameRate);
}
```

Shader Vert: encargado de manejar las posiciones y luminosidad de los vértices.

```
uniform mat4 transform;
uniform mat3 normalMatrix;
uniform vec3 lightNormal;

attribute vec4 position;
attribute vec4 color;
```

```
attribute vec3 normal;

varying vec4 vertColor;
varying vec3 vertNormal;
varying vec3 vertLightDir;

void main() {
  gl_Position = transform * position;
  vertColor = color;
  vertNormal = normalize(normalMatrix * normal);
  vertLightDir = -lightNormal;
}
```

Shader frag: encargado de manejar el color del fragmento.

```
#ifdef GL_ES
precision mediump float;
precision mediump int;
#endif

uniform float fraction;

varying vec4 vertColor;
varying vec3 vertNormal;
varying vec3 vertLightDir;

void main() {
  float intensity;
  vec4 color;
  intensity = max(0.0, dot(vertLightDir, vertNormal));

  if (intensity > pow(0.95, fraction)) {
    color = vec4(vec3(1.0), 1.0);
  } else if (intensity > pow(0.5, fraction)) {
    color = vec4(vec3(0.6), 1.0);
  } else if (intensity > pow(0.25, fraction)) {
    color = vec4(vec3(0.4), 1.0);
  } else {
    color = vec4(vec3(0.2), 1.0);
  }

  gl_FragColor = color * vertColor;
}
```

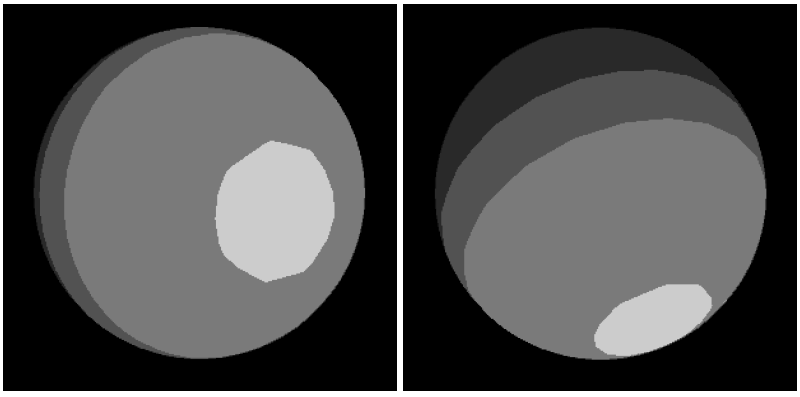


Figura C.1: Resultado