Laboratorio de medios, Instituto de Computación, Facultad de Ingeniería, UDELAR.

Informe de Proyecto de Grado

Proyección sobre superficies irregulares

Estudiantes:
Daniel Gomez de Souza
Javier Fradiletti
Adriana Soucoff

Tutor:
Tomás Laurenzo

Montevideo, Uruguay, año 2012.

Índice general

Capítulo 1

Introducción

La proyección de imágenes sobre una superficie plana mediante un foco luminoso es una técnica con variadas aplicaciones, siendo de las más utilizadas la proyección de películas cinematográficas. En los últimos tiempos se introdujeron dispositivos de video que permiten proyectar la salida de una computadora, los cuales son comúnmente utilizados en ámbitos académicos y empresariales con fines de dictado de cursos y presentaciones.

En los últimos años se ha ido popularizando la proyección de imágenes y videos sobre fachadas, monumentos u otros tipos de superficies irregulares de modo de resaltar, ocultar o transformar distintas regiones de interés. Esta técnica recibe el nombre de *video mapping*. Un espectáculo de *video mapping* es, en esencia, una expresión artística en la que un $VJ^{\dagger 1}$ presenta su creación mediante la proyección de diversos efectos visuales que generalmente son acompañados por efectos de sonido. Todo esto se logra mediante la utilización de software especializado que distorsiona las imágenes y videos proyectados.

Esta técnica tiene una gran variedad de aplicaciones que van desde el ámbito artístico hasta el publicitario. Es muy común observar este tipo de proyecciones sobre fachadas con la finalidad de celebrar el aniversario de algún edificio emblemático o durante la realización de festivales que se llevan a cabo periódicamente en ciertas ciudades. Claro ejemplo de esto último es el Festival de Lyon[Fes] que se lleva a cabo en dicha ciudad, generando una atracción de interés turístico. Otro tipo de aplicación de *video mapping* se da en espectáculos de menor envergadura, realizados generalmente en espacios cerrados, los cuales se basan en proyecciones sobre objetos tridimensionales. Estos pueden ser tanto objetos creados específicamente para este propósito, por ejemplo maquetas, u objetos ya existentes de la más diversa variedad: automóviles, teléfonos móviles, calzado, etc. Una nueva tendencia en el ambiente artístico es la incorporación de esta técnica en obras teatrales, lográndose una ilusión de interacción entre los actores y las imágenes proyectadas.[2]

[1] El símbolo (†) representa las referencias al glosario.

[2] En el presente documento todas las imágenes se referencian en el índice de figuras.

Figura 1.1: Fachada festival Lyon

Figura 1.2: Instalaciones sobre maquetas.

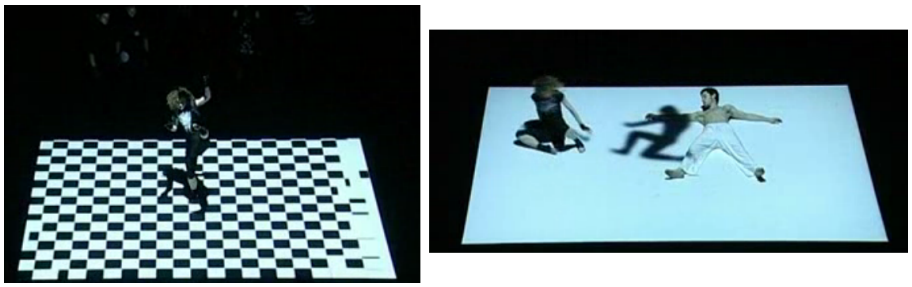

Figura 1.3: Instalación interactiva con actores.

Para la elaboración de un espectáculo con las características antedichas no existe

un procedimiento estándar a seguir sino que cada artista utiliza sus técnicas, métodos y aplicaciones de software que considere adecuadas. No obstante, se identifican etapas comunes como la construcción de un modelo de la escena, la producción del espectáculo y su proyección. A su vez se identifican dificultades comunes como la generación del modelo y la calibración de los proyectores al momento de la reproducción.

En los siguientes capítulos, se estudian distintos métodos para facilitar la obtención del modelo virtual. En particular se relevan métodos de escaneo y reconstrucción de objetos tridimensionales y se implementa una de las estrategias estudiadas para su generación. Se desarrolla también una aplicación que permite la creación, edición y reproducción de un espectáculo de *video mapping* basada en un motor tridimensional[†]. Ésta combina la edición y la reproducción permitiendo visualizar en tiempo real los distintos efectos diseñados. A su vez permite distribuir la reproducción entre varias computadoras, asociadas a uno o más proyectores, sincronizadas todas por un componente central.

El presente documento se organiza en cuatro capítulos y cuatro apéndices, cuya temática se detalla a continuación:

- Capítulo 1 - Introducción a la técnica de *video mapping*.
- Capítulo 2 - Estado del arte. Explicación de las nociones básicas de la técnica y de las etapas en el proceso de creación: modelado, producción y proyección; presentación de estas dos etapas mediante los enfoques bidimensional y tridimensional; explicación de distintos algoritmos y técnicas para la obtención automática y la reconstrucción de geometría tridimensional.
- Capítulo 3 - Solución planteada. Presentación de la solución propuesta: una herramienta para la edición y posterior ejecución de espectáculos de *video mapping*.
- Capítulo 4 - Conclusiones y trabajo futuro. Resumen de las conclusiones y posibles líneas de trabajo futuro.
- Apéndice A - Relevamiento de aplicaciones de *video mapping*. Presentación de un relevamiento de las aplicaciones existentes más populares, utilizadas para la creación de espectáculos.
- Apéndice B - Aportes. Presentación de los resultados de una serie de entrevistas realizadas a *VJs* e ingenieros que contribuyeron a la investigación del estado del arte.
- Apéndice C - Método de triangulación. Detalle del método de triangulación utilizado para la obtención automática de geometría tridimensional.
- Apéndice D - Modelo de cámara. Explicación de los fundamentos teóricos del modelo de cámara.

Capítulo 2

Estado del arte

El estado del arte se basa en el estudio de la técnica de *video mapping* y del modelado automático de geometrías. El proceso de creación de un espectáculo de *video mapping* se apoya en entrevistas a *VJs* y en un relevamiento de las aplicaciones por ellos sugeridas, anexadas en los apéndices Aportes y Relevamiento de aplicaciones de *video mapping* respectivamente. Por su parte, el modelado automático es un área no solo relacionada al *video mapping* sino que es aplicable también en distintas ramas de la ingeniería como son la visión por computadora, animación tridimensional, interfaces persona-computadora, entre otras. Es por esta razón que se decidió realizar un estudio de estas técnicas, abordándose de forma genérica los problemas de obtención de una nube de puntos[†] correspondientes a objetos tridimensionales reales y la reconstrucción de mallas[†] tridimensionales para su utilización durante el proceso de creación de espectáculos.

2.1. Creación de un espectáculo de *video mapping*

La creación de un espectáculo de *video mapping* se presenta en tres etapas: el modelado de la escena, la producción del espectáculo y la proyección del mismo. Cada una de estas etapas será abordada en esta sección desde dos enfoques: bidimensional y tridimensional, ya que tanto los problemas que plantean, así como los escenarios de aplicación difieren en cada caso, según el enfoque.

Durante la etapa de modelado de la escena, se obtiene una representación abstracta de los objetos reales sobre los que se proyectará. Este modelo constituye la base para trabajar en posteriores etapas y sobre él se diseñará el espectáculo.

La producción del espectáculo consiste en definir los distintos efectos visuales a ser aplicados sobre los objetos modelados y la forma en que serán ejecutados. Los espectáculos pueden discriminarse en dos categorías según la forma de presentación de los efectos pudiendo ser en una secuencia predefinida o en tiempo real. Si la secuencia a ejecutar es predefinida, en esta etapa se define dicha secuencia indicando el instante en que cada efecto se desplegará. Esto por lo general se hace de forma sincronizada con la música que

suele acompañar al espectáculo. Por otra parte, si la ejecución de los efectos se define en tiempo real, esta etapa es utilizada para definir los mecanismos que desencadenan la ejecución de los efectos. Así por ejemplo, se podría definir para un teclado, un efecto distinto para cada tecla para que el *VJ* los ejecute a voluntad en la presentación. En esta etapa también se diseña y produce la musicalización que será utilizada durante todo el espectáculo.

Es en la etapa de proyección del espectáculo en donde se puede contemplar el resultado de los distintos efectos visuales proyectados sobre las superficies, acompañados estos por efectos de sonido. Para lograr la correspondencia en la proyección de los objetos del modelo con las superficies se debe calibrar la proyección. Esto se logra modificando el modelo de la escena, ajustando la posición y orientación de los proyectores, así como los parámetros intrínsecos de la proyección, siendo estos el ángulo de enfoque, la resolución y distancia focal.

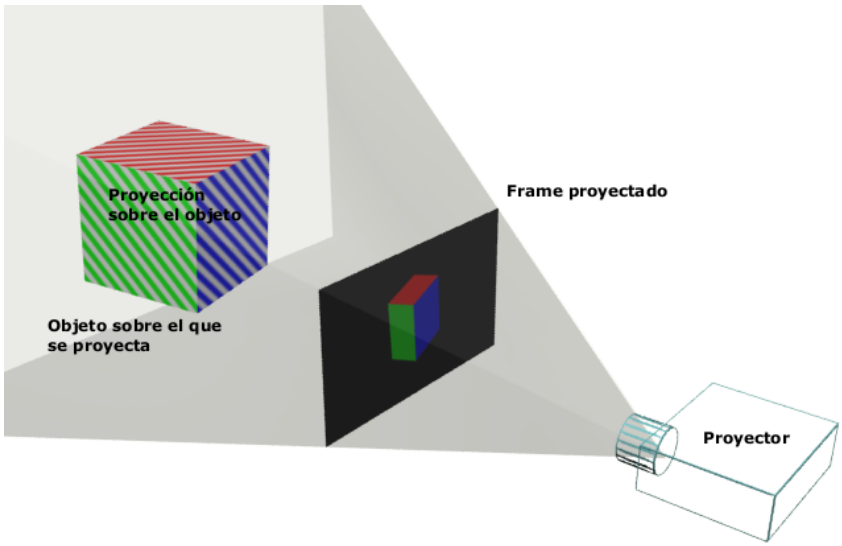

Figura 2.1: Esquema de proyección.

2.1.1. Enfoque bidimensional

Modelo

Un modelo bidimensional refleja lo que vería un observador desde un punto de vista fijo. Técnicamente es el resultado de una proyección en perspectiva [Jam96] sobre un plano de vista de los elementos de la superficie a modelar. Este punto de vista debe ser considerado al posicionar y orientar el proyector que reproducirá el espectáculo. La posición, orientación y campo de vista del proyector definirán además la sección de superficie sobre la que se proyectará. En caso de utilizar más de un proyector cada uno de estos

será posicionado en un lugar diferente y por lo tanto será necesario un modelo por cada uno de ellos.

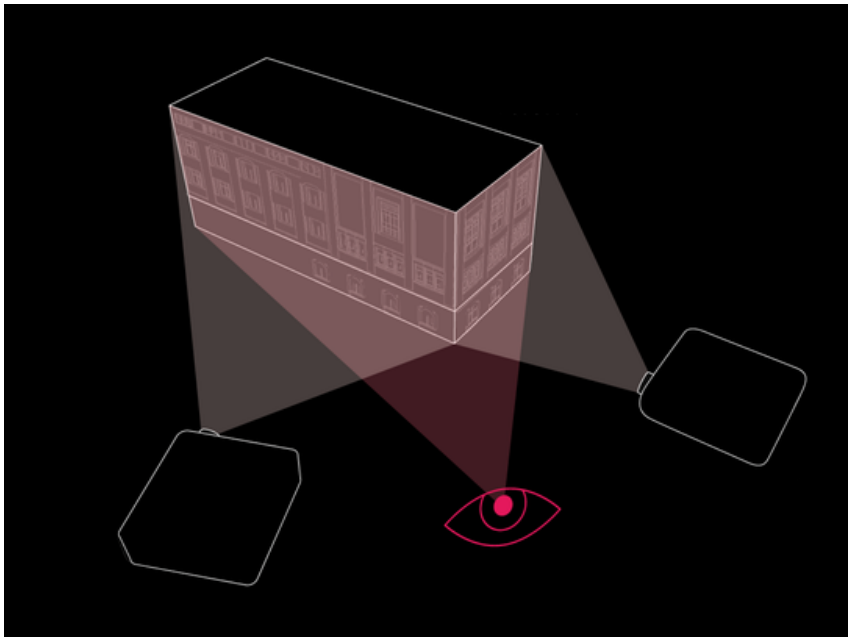

Figura 2.2: Proyectores y sus puntos de vista.

Son ejemplos de modelos: fotografías, planos arquitectónicos y figuras geométricas. Cada uno de estos requiere distintas técnicas para su creación y la forma de trabajo en las posteriores etapas dependerá del modelo utilizado.

Una fotografía captura lo que vería un observador desde el punto de vista desde donde fue tomada. Es una forma sencilla de crear el modelo de una escena, aunque usualmente requiere trabajo adicional para llegar a ser un modelo fiel. Para utilizar este modelo en etapas posteriores es conveniente que el punto de vista desde donde fue tomada la fotografía coincida con la posición desde donde se desea proyectar, minimizando de esta forma los ajustes de calibración necesarios.

Un plano arquitectónico, por ejemplo de una fachada, contiene información exacta de las medidas de la superficie que representa en una escala dada. Generalmente utiliza el método de proyecciones paralelas [Jam96] sobre un plano de proyección. Para utilizar el plano arquitectónico como modelo, se debe transformar de forma que coincida con la proyección en perspectiva desde el punto vista donde estará ubicado el proyector.

Mediante figuras geométricas se modelan sectores de la superficie a proyectar. Un método para generarlas consiste en utilizar un proyector y herramientas de software que permiten delinear el contorno de las secciones de la superficie, ajustando el modelo en el momento de la construcción. Al usar el proyector para obtener el modelo queda incorporada la perspectiva del mismo, por lo tanto no es necesaria otra calibración a menos que haya modificaciones en la ubicación u orientación del proyector. Otro método consiste en dibujar las figuras con una fotografía o plano de fondo. En este caso la construcción de las figuras geométricas se realiza delineando el contorno de la superficie en la fotografía o

plano. Este tipo de modelos también pueden ser generados de forma automática utilizando técnicas de visión por computadora† como por ejemplo el método basado en el filtrado de partículas[Geo06] para el reconocimiento de aristas.

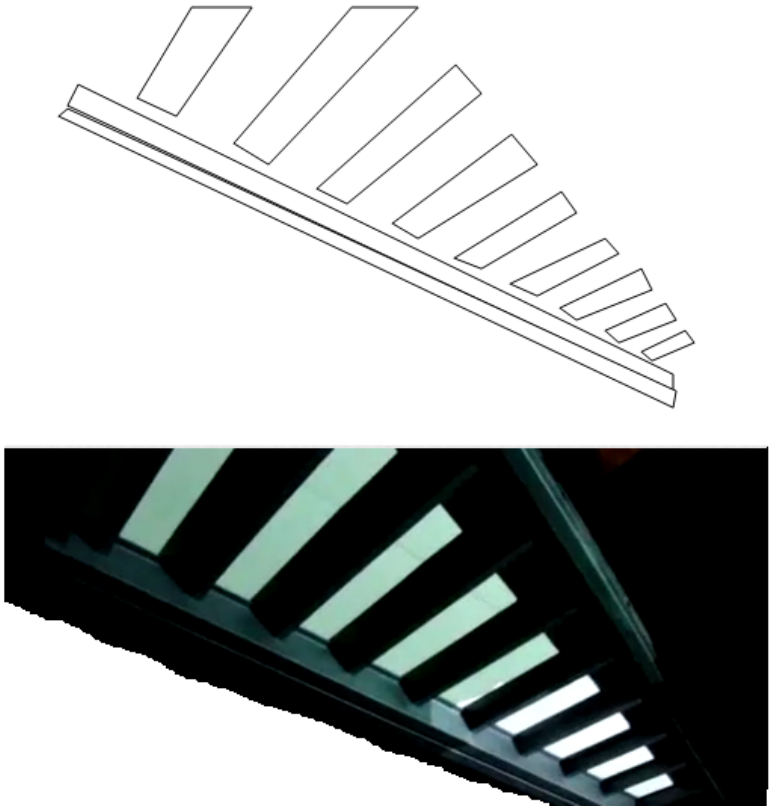

Figura 2.3: Representación con figuras geométricas.

Producción del espectáculo

La producción del espectáculo en dos dimensiones consiste en definir efectos visuales sobre regiones de un espacio bidimensional discreto† representadas en el modelo de la escena. Este espacio se representa con coordenadas de pantalla que identifican cada uno de los píxeles† del área de trabajo. Los efectos visuales se logran realizando cualquier animación computacional que genere una salida gráfica como pueden ser videos e imágenes. En esta etapa, además de definir los efectos, se planifica cuándo se mostrarán cada uno de ellos, pudiendo sincronizarse con la música que forma parte del espectáculo. Los efectos podrán ser mostrados de forma secuencial, planificando en qué instante se ejecutará cada uno de ellos, o también definiendo acciones para que un *VJ* decida en tiempo real en un espectáculo en vivo, los efectos a desplegar.

En computación gráfica se utilizan texturas para proyectar videos e imágenes sobre regiones del área de trabajo. Las texturas son mapas de bits† utilizados para cubrir la superficie de un objeto virtual. Estos mapas de bits pueden ser generados a partir de

imágenes, videos, o incluso dinámicamente mediante rutinas computacionales, permitiendo así crear efectos visuales como por ejemplo, la transición de un color a otro.

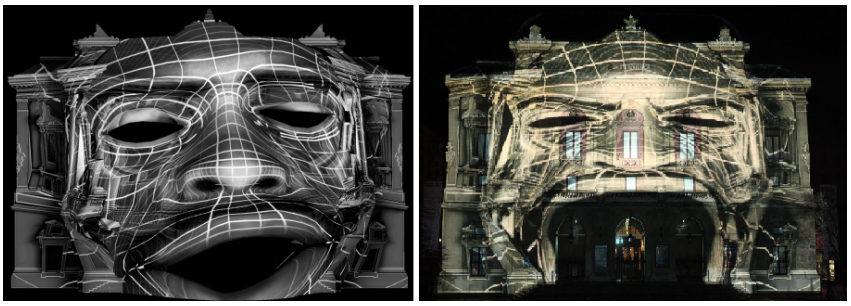

Figura 2.4: Izq. Efecto en una herramienta de software. Der. Efecto proyectado.

Al utilizar videos para crear efectos es usual que se tenga en cuenta su posterior proyección, en particular la superficie donde se proyectará y el punto de vista desde donde se contemplará el espectáculo. Un ejemplo en el que esta consideración es importante es cuando se pretende crear una ilusión tridimensional, pues la perspectiva de las formas proyectadas en el efecto debe ser coherente con la escena desde la óptica de los espectadores.

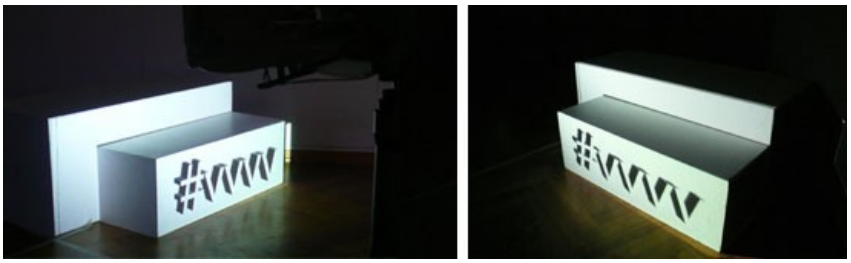

Figura 2.5: Izq. Ilusión tridimensional lograda. Der. No se logra la ilusión.

En este contexto se habla de mapeo, no como la salida a través de un equipo proyector, sino como la operación que logra una correspondencia entre una textura y una figura geométrica que no necesariamente coinciden en tamaño y forma. Para esto se definen coordenadas de textura en cada vértice de la figura geométrica que referencian distintas ubicaciones dentro de dicha textura. Las coordenadas de las texturas tienen dos componentes: una horizontal y una vertical llamadas UV. Si el valor de estas componentes se normaliza entre 0 y 1 entonces la esquina superior izquierda de la textura se corresponderá con la coordenada $(0, 0)$, la superior derecha con $(1, 0)$, la inferior izquierda con $(0, 1)$ y la inferior derecha con $(1, 1)$. Los vértices de una figura geométrica se asocian con coordenadas UV que definen el punto de la textura que se corresponde sobre el vértice y mediante interpolación se logra mapear toda la textura a la figura geométrica.

Si bien es posible mapear una textura a cualquier figura geométrica, esta correspondencia es más directa utilizando un cuadrilátero ya que a cada uno de los vértices se lo hace corresponder con una esquina de la textura. A su vez el cuadrilátero es la figura

básica en las aplicaciones[1] de *video mapping*. Estos cuadriláteros se utilizan como piezas constructoras del espectáculo, cubriendo sectores del modelo sobre los cuales luego se aplican las texturas permitiendo crear los distintos efectos visuales.

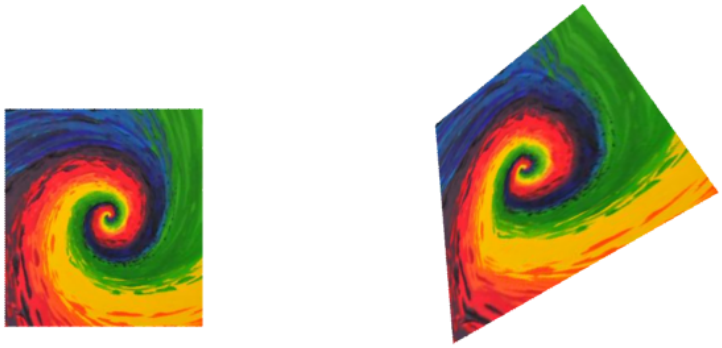

Figura 2.6: Izq. Textura. Der. Cuadrilátero con textura mapeada.

Proyección del espectáculo

La principal dificultad encontrada en la etapa de la proyección del espectáculo es lograr que el modelo previamente generado se corresponda con la escena sobre la cual se proyectará. El proceso que ajusta este modelo a proyectar con la superficie que representa es denominado calibración. Los ajustes necesarios varían dependiendo del método utilizado para obtener el modelo. En caso de utilizar una fotografía, el proyector deberá ser ubicado de forma tal que el punto de vista de la cámara con la que se tomó coincida con el del proyector y así lograr la coincidencia del centro de proyección. Igualmente son necesarios ajustes ya que los lentes de la cámara y el proyector no necesariamente coinciden en el ángulo de visión. Con el método de generación de figuras geométricas en el que se modelan las secciones de la superficie a mapear utilizando el mismo proyector, los ajustes se reducen a lograr la misma posición y orientación que tiene el proyector al momento de la captura de secciones, ya que las deformaciones relacionadas a los parámetros intrínsecos fueron implícitamente consideradas durante el proceso de captura.

En cualquiera de estos dos métodos, lograr que la posición y orientación del proyector coincidan con el punto de vista desde donde se obtuvo el modelo puede no ser una tarea sencilla. Esta tarea puede ser asistida, modificando la proyección mediante una homografía[2] que es la transformación geométrica que permite corresponder puntos en dos planos de perspectiva distintos, logrando de esta manera ajustar el modelo proyectado a la superficie. Este procedimiento deberá ser realizado nuevamente en caso de haber alteraciones en la posición y orientación del proyector ya que los parámetros de calibración obtenidos al lograr la correspondencia son particulares para una ubicación del proyector dada.

[1] Las aplicaciones relevadas utilizan el cuadrilátero como figura básica. Ver apéndice: Aplicaciones relevadas.

[2] Esto se extiende en la sección: Obtención de geometría

2.1.2. Enfoque tridimensional

Modelo

En el modelado tridimensional se representan cuerpos y superficies tridimensionales comúnmente representadas por medio de mallas de polígonos, utilizadas en disciplinas como visión computacional y computación gráfica. A diferencia de un modelo bidimensional, éste no depende de un punto de vista, lo que permite al diseñador visualizarlo desde diferentes ángulos mediante la definición de cámaras virtuales. Las entidades que conforman la malla son vértices, aristas, caras y atributos numéricos que representan la posición y normales de los vértices, coordenadas de textura y colores. Existen distintas topologías de mallas de polígonos siendo de las más utilizadas la que se basa en la cantidad de vértices que forman una cara.

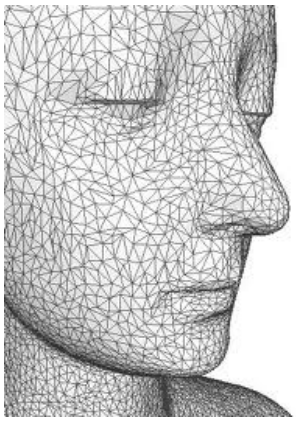

Figura 2.7: Mallas triangulares.

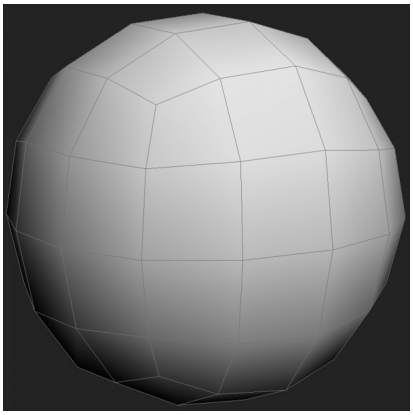

Figura 2.8: Mallas de cuadriláteros.

Técnicas como las de *remeshing*, mediante la utilización de algoritmos específicos reducen la cantidad de vértices de la malla sin perder la representatividad de la superficie. Un estudio de estas técnicas se presenta en el artículo *Recent advances in compression of 3D meshes[Pie03]*. De esta forma se logra eficiencia en el manejo de las mallas que generalmente utilizan algoritmos cuyo orden varía dependiendo de la cantidad de vértices así como también se optimiza el almacenamiento de estas representaciones.

Los modelos tridimensionales se generan utilizando distintas técnicas cuya elección depende de varios factores como el nivel de detalle deseado, uso final del modelo y habilidades del usuario. Algunas de estas técnicas son:

- Modelado utilizando polígonos: a partir de mallas que representan figuras primitivas, por ejemplo cubos o esferas, se podrán construir nuevas mallas aplicando transformaciones que modifican las aristas, vértices o caras.
- Modelado utilizando curvas: a partir de una jaula creada por curvas se aplican

transformaciones para modificarla, manipulando sus puntos de control. Es utilizado en el modelado de automóviles, edificios y mobiliario, entre otros.

- Esculpido digital: técnica digital que simula el esculpido convencional mediante software especializado que provee una interfaz para modificar el modelo de forma detallada, oprimiendo y resaltando zonas de la superficie. Es utilizado para lograr efectos especiales en películas y video juegos logrando figuras y texturas complejas.
- Reconstrucción a partir de fotografías: se obtiene la representación de la superficie con una foto. Conociendo la escala de la imagen y mediante mediciones de los objetos fotografiados se extrapola y se obtiene la distancia entre dos puntos en la superficie. Es usada en arquitectura, ingeniería, geología, arqueología, etc.
- Reconstrucción utilizando hardware especializado: mediante escáneres tridimensionales se obtiene una nube de puntos que representa la superficie. El conjunto de puntos obtenido es denso, generalmente con redundancia y errores, es por esto que comúnmente se utilizan algoritmos específicos para reducir la nube de puntos [3].

Para las técnicas de modelado utilizando polígonos y curvas, algunas de las herramientas más populares son *Maya*[May], *3D Studio Max*[3DSb] y *Blender*[Ble], mientras que para la realización de esculpido digital son también muy utilizadas *3D-Coat*[3DC], *ZBrush*[Zbr], *Sculptris*[Scu] y *Mudbox*[Mud].

Figura 2.9: *Mudbox*

[3]Esta técnica se ampliará en la sección Obtención de Geometía

Figura 2.10: *Sculptris*

Producción del espectáculo

La producción del espectáculo en tres dimensiones agrega un nivel de abstracción adicional a la producción bidimensional. Esto permite que el diseñador cree un espectáculo transformando directamente los objetos del modelo tridimensional en contraposición al anterior enfoque en el que se transforman sus perspectivas. Se plantea un cambio en la forma de trabajar en el espectáculo y planificar la producción, ya que el diseñador no estará restringido a considerar ubicación alguna de los proyectores.

El modo de trabajo se basa en mapear texturas sobre las caras de los objetos tridimensionales de forma análoga a como se realiza sobre figuras bidimensionales. En este tipo de modelos podría ser deseable mapear una textura de forma que abarque varias caras del mismo objeto tridimensional, por ejemplo, el conjunto de caras que representa la superficie lateral curva de un cilindro. Para lograrlo se deben mapear distintos sectores de la textura en cada una de las caras que conforman la superficie, utilizando coordenadas de textura en cada uno de los vértices. La diferencia radica en que en el presente enfoque los vértices no se encuentran necesariamente en el mismo plano, por lo cual ajustar una textura bidimensional a este tipo de superficies no es directo.

Existen diversas técnicas que asisten en la tarea de definir las coordenadas de textura. Una de ellas consiste en aproximar la superficie por una primitiva conocida, más simple, como ser un cubo, cilindro, esfera o plano. Para estas superficies existen funciones matemáticas que proyectan cada punto de la superficie a un plano, de esta forma se pueden determinar las coordenadas UV. Un ejemplo de esto para esferas son las proyecciones utilizadas para representar el planisferio como la proyección cilíndrica equidistante

presentada en el libro *Flattening the Earth[Joh93]*. Otra técnica es la denominada *UV unwrapping* que consiste en desenvolver una malla tridimensional, aplanándola sobre la textura. De esta manera el mapeo se realiza de forma más intuitiva ya que la superficie aplanada es bidimensional al igual que la textura. Esta técnica es soportada por la mayoría de los programas de edición de gráficos tridimensionales, los que proveen distintas herramientas para ajustar la forma en que se desenvuelve la textura.

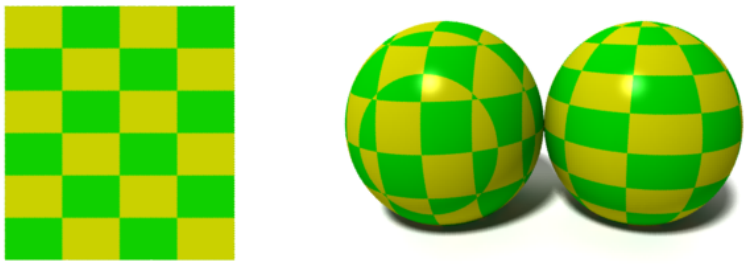

Figura 2.11: Izq. Textura. Der. Esferas con distintos mapeos de la misma textura.

En la escena tridimensional los objetos son visualizados utilizando cámaras virtuales que fijan un punto de vista. En la salida gráfica se representará entonces la escena desde la perspectiva de una cámara virtual, permitiendo visualizarla desde distintos ángulos. Esto posibilita definir el punto de vista del proyector que reproducirá la salida gráfica en el momento que se desee. Un caso sería visualizar el resultado del espectáculo luego de producido y en este momento definir el punto de vista más conveniente.

Producir el espectáculo con este enfoque, es decir, transformando directamente los objetos y no sus perspectivas, permite escalar con mayor facilidad en la cantidad de proyectores a utilizar. Esto se logra agregando cámaras virtuales que definan más puntos de vista sin necesidad de modificar el modelo ni los efectos producidos.

Si bien se está abordando un enfoque tridimensional de la producción del espectáculo, existen casos en que es más sencilla la implementación de ciertos efectos tomando un enfoque en dos dimensiones, sobre todo cuando estos se aplican sobre regiones planas de la superficie. Es por esto que es muy común combinar ambos enfoques y utilizarlos de acuerdo a las necesidades del diseñador. Cabe aclarar que al combinar los enfoques se pierden las ventajas de definir el punto de vista del proyector luego de la etapa de producción, por lo que los objetos bidimensionales y efectos sobre éstos se deben definir una vez fijada la ubicación de los proyectores.

Proyección del espectáculo

En esta etapa se fija finalmente el punto de vista desde donde se proyectará, definiendo la ubicación y orientación de los proyectores. Al igual que en el enfoque bidimensional, para lograr una correspondencia en la proyección de la escena con las superficies a proyectar se debe realizar una calibración que consiste en modificar los parámetros de ubicación,

orientación y proyección de las cámaras virtuales. La ubicación y orientación relativas de la cámara virtual y la escena deberán corresponderse a las del proyector con las superficies a proyectar.

2.2. Obtención de geometría

En esta sección se presenta la reconstrucción tridimensional automática de una superficie utilizando técnicas computacionales. Se introducen distintos métodos que permiten la construcción automática de modelos, discutiendo sus características según propiedades de la superficie a representar y tecnologías utilizadas. Inicialmente se obtiene una nube de puntos correspondiente a la superficie, utilizando técnicas y dispositivos para este propósito, para luego procesar la información obtenida y construir una malla tridimensional. Mediante este procesamiento se obtienen propiedades adicionales como grupos de puntos que representan caras de una malla y las normales que identifican la orientación de la superficie.

2.2.1. Obtención de la nube de puntos

Visión estéreo

La visión estéreo se basa en el análisis de dos imágenes observadas desde puntos de vista ligeramente diferentes, de forma similar a la visión humana, para reconstruir la estructura tridimensional de una escena. Este método entra en la categoría de los métodos pasivos por no utilizar luz auxiliar. El análisis de las imágenes se realiza por medio de algoritmos que han evolucionado en los últimos tiempos logrando la reconstrucción de escenas rápidamente. Es por esto que esta técnica es muy utilizada en implementaciones que necesitan respuesta en tiempo real. Un primer paso es el preproceso de las imágenes donde se identifican sus principales características. Este análisis definirá qué tipo de algoritmo se utilizará en el paso de correspondencia, ya que definirá las características que se estudiarán. Luego se resuelve el problema de correspondencia, en el que dadas dos imágenes de la escena, para todos los puntos de la primera se obtiene el punto correspondiente en la otra y se calcula la desigualdad de cada punto, es decir, la distancia en píxeles entre los puntos que se corresponden. Finalmente se calcula la profundidad del punto utilizando el método de triangulación[4] en el cual se obtiene la distancia focal de cada punto a cada una de las cámaras[SB].

Existen variedad de algoritmos que solucionan el problema de correspondencia y cálculo de desigualdad entre los puntos de las dos imágenes. Se pueden clasificar según el análisis de preproceso identificando características principales de las mismas destacándose dos estrategias[Ume]:

[4]ver anexo de método de triangulación

- *Separate area*: basada en correlación del brillo e intensidad. Se utilizan patrones de brillo aplicados a un píxel y sus vecinos utilizando principio de localidad. Las diferencias en la perspectiva de la imagen o cambios en luminosidad absoluta de la escena pueden generar errores.

- *Features*: las características usadas para la correspondencia son aristas, puntos o segmentos dadas por cambios de intensidad de la imagen. Esta estrategia es más estable ante la variación de luminosidad absoluta y en la práctica la correspondencia es más rápida.

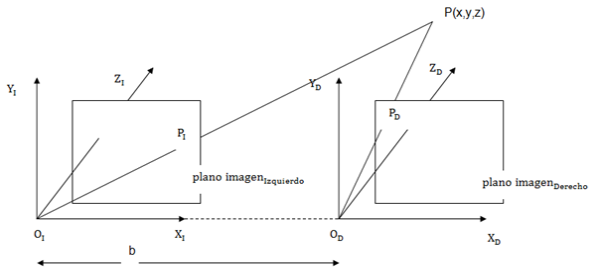

Figura 2.12: Geometría estéreo con ejes paralelos

En la figura 2.12 se muestra un caso particular de geometría estéreo que consta de ejes ópticos de las cámaras paralelos y en consecuencia los planos de imagen son coplanares. En los casos más comunes no se tienen estas características, y por ello es necesario introducir parámetros en los cálculos que hacen el ajuste del modelo *pinhole*[5] al modelo de las cámaras reales.

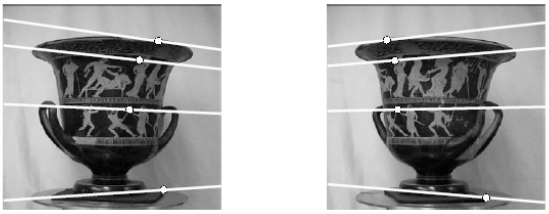

Figura 2.13: Dos imágenes con las líneas epipolares y puntos que se corresponden

En el caso mostrado en la figura 2.13 se ven los puntos correspondientes en cada imagen ubicados sobre las líneas epipolares[6].

La principal desventaja del método de visión estéreo es que en caso de oclusión hay regiones que no tienen correspondencia en las dos imágenes, por esto no se puede resolver el problema de correspondencia para estas regiones.

[5]Ver apéndice modelo de cámara: modelo *pinhole*
[6]ver anexo de geometría epipolar

Luz estructurada

Luz estructurada es una técnica de obtención de geometría tridimensional que consiste en la proyección de patrones sobre la superficie a capturar para luego analizar la deformación de los mismos y obtener una nube de puntos que la representan. Este método, a diferencia de visión estéreo, es un método activo ya que reemplaza la segunda cámara por un proyector de luz que proyecta patrones sobre la superficie a modelar. Los patrones son capturados por la cámara de video para su posterior análisis, obteniendo información tridimensional de la posición, orientación y textura de la superficie[Joa].

Dependiendo de la superficie a escanear, los patrones proyectados pueden tener problemas de oclusión, baja reflexión y puntos reflejados fuera del alcance de la cámara. Como consecuencia, la pérdida de estos puntos proyectados impide resolver el problema de la correspondencia. Estos problemas se pueden solucionar utilizando una codificación de patrones adecuada[J.].

Según dependencia temporal los patrones se clasifican en:

- Estáticos: El patrón es limitado para escenas estáticas en donde son necesarias proyecciones de varios patrones distintos. El movimiento de cualquier objeto de la escena mientras se realiza la obtención de los patrones proyectados producirá un error de correspondencia.
- Dinámicos: Los objetos en la escena se pueden mover por lo tanto es necesaria la utilización de un único patrón de proyección.

Clasificación según la luz proyectada:

- Binaria: Cada uno de los puntos del patrón tiene dos posibles valores codificados con 0 y 1 respectivamente. Este valor representa opacidad y transparencia, ausencia o presencia de la luz proyectada en el objeto.
- Escala de grises: Cada punto del patrón tiene asociado un valor de gris que representa el nivel de trasparencia o nivel de opacidad del punto para la luz proyectada. En este caso son necesarios dos pasos. Primero se obtiene una imagen de la escena iluminada con la misma luz sin variar la intensidad y luego se obtiene la referencia de luz necesaria para cancelar el efecto de reflejo de la superficie que depende directamente del tipo de superficie. La necesidad de estos dos pasos contribuye a que este patrón también sea clasificado como estático.
- Color: Cada punto del patrón es asociado con un valor de tono. Los tonos deben ser bien diferenciados para alcanzar una segmentación eficiente. Este tipo de patrones son limitados por el color de la escena. En caso de presentar colores altamente saturados se producen pérdidas de regiones en el paso de segmentación que luego provocan errores en la decodificación.

Otra posible clasificación es según la discontinuidad en la profundidad de la superficie proyectada:

- Periódica: La codificación se repite periódicamente a lo largo del patrón. Esta técnica se utiliza para reducir el número de bits que codifican el patrón. Como limitante la profundidad del objeto no puede ser mayor que la mitad de la longitud del período.
- Absoluta: Cada columna o fila del patrón proyectado tiene una única codificación. No sufre dependencia de discontinuidad de profundidad.

Un algoritmo que utiliza codificación binaria es el llamado *Three Phase-Shifting* que utiliza tres ondas sinusoidales para la proyección del patrón. Este algoritmo fue presentado por Zhang[Son09] e implementado por Kyle McDonalds en [Kyl]. Dicho algoritmo es aplicable únicamente para superficies con curvas continuas. Superficies disjuntas o con aristas pronunciadas serán capturadas de forma distorsionada.

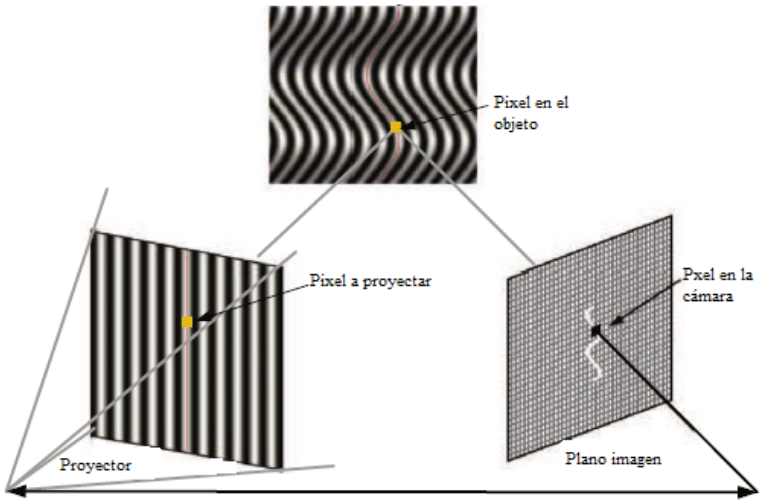

Figura 2.14: Instalación básica de proyección de patrones y sistema de *Phase-Shifting*

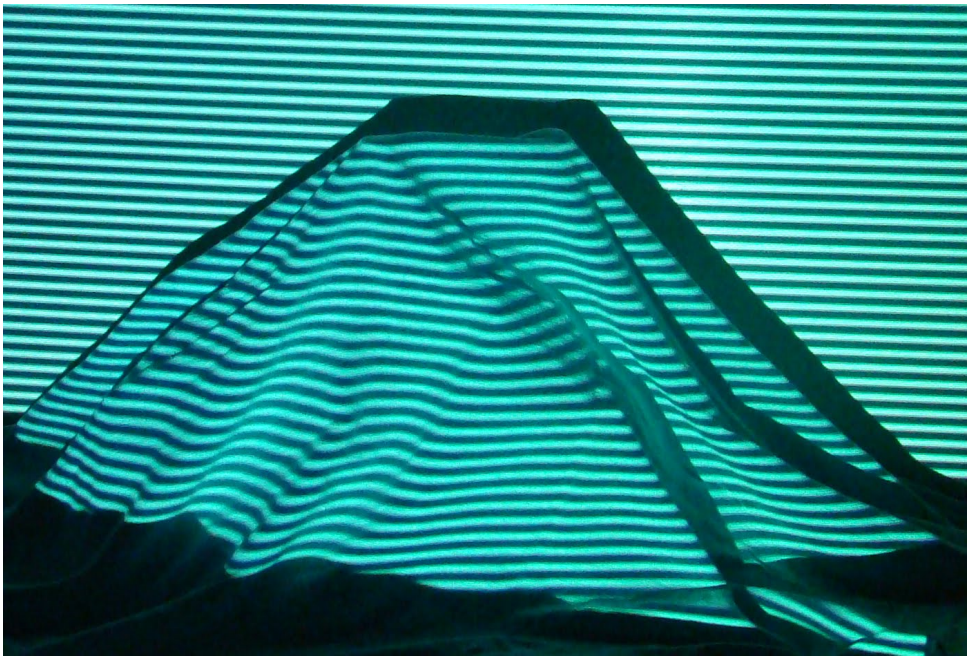

Figura 2.15: Superficie con patrón proyectado

Calibración

En esta sección se describe un método originalmente propuesto por Zhang[Son09] para la calibración de una cámara y un proyector con el objetivo de capturar correctamente los patrones proyectados para su posterior análisis. La calibración de la cámara requiere estimar inicialmente los parámetros del modelo de cámara *pinhole*, por lo tanto se deben obtener los parámetros intrínsecos como distancia focal, punto principal y factores de escala, y los extrínsecos definidos por la matriz de rotación y vector de traslación de un punto en el espacio al sistema de coordenadas de la cámara. Luego se debe capturar una secuencia de imágenes de un objeto simple, con un conjunto de características fijo y distinguible, y con un desplazamiento tridimensional conocido. Esto permite que cada imagen capturada durante el proceso de calibración provea de un conjunto de correspondencias de los puntos tridimensionales de la escena a puntos bidimensionales en el sistema de coordenadas de la cámara. Particularmente, en el método de Zhang, el objeto conocido que se observa es un tablero de damas plano en una o más orientaciones. De la secuencia capturada se pueden obtener los parámetros intrínsecos y luego, utilizando una sola toma de la secuencia, se obtienen los parámetros extrínsecos. Para la calibración del proyector, éste se modela como el inverso de una cámara, teniendo en cuenta que la luz viaja en la dirección opuesta y que un punto en el plano de la imagen se mapea a un rayo de luz saliente por el punto y por el centro de proyección. Dado este modelo, la calibración se debe realizar de forma similar a la de una cámara, con la salvedad que en lugar de tomar imágenes de un tablero de damas fijo, se proyecta un tablero en una ubicación conocida y se toman imágenes del mismo utilizando la cámara para analizar las distorsiones. Este enfoque resulta ser una extensión directa del método de Zhang para calibración de cámaras.

Existen varias implementaciones de esta técnica de calibración basado en el modelo inverso de la cámara. Una de estas implementaciones es *BYO3D*[BYO], que se encuentra disponible para *MATLAB*[MATa] y como una extensión a la biblioteca de visión computacional *OpenCV*[Opea].

Dispositivos de captura de geometría

Un dispositivo para la captura de geometría en tres dimensiones cuenta con componentes de hardware y software que implementan las técnicas antes mencionadas. En general, estos componentes son seleccionados de modo que optimicen las técnicas en las que se basan. Un ejemplo de estos dispositivos es *DAVID Structured Light Scanner[DAV]*, cuyo componente de software utiliza la técnica de luz estructurada y su hardware está compuesto por una cámara y un proyector de video. Otro es el caso *Kinect* de *Microsoft*, que también implementa la técnica de luz estructurada con la salvedad que se reemplaza el proyector y la cámara de video por un emisor y una cámara de infrarrojos.

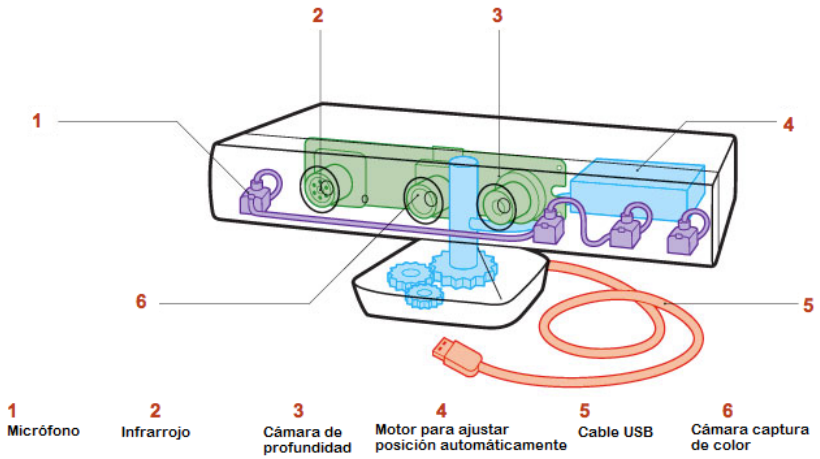

Figura 2.16: Componentes de un sensor *Kinect*

En los últimos años se ha popularizado el uso de *Kinect* por ser un dispositivo que se puede adicionar a la consola *Xbox360* de *Microsoft* con el objetivo de permitir que el usuario interactúe con la misma utilizando únicamente el movimiento de su cuerpo. Para el análisis de las deformaciones de los rayos y la construcción del mapa de profundidad, mediante el emisor infrarrojo con el que viene equipado se proyectan patrones de luz por toda la escena y utilizando la cámara de profundidad se analizan las distorsiones. También se cuenta con una cámara convencional para analizar objetos o personas en la escena y para la detección de colores.

A mediados de 2011 fue presentada una interfaz de programación gratuita que permite utilizar *Kinect* de forma directa en aplicaciones no licenciadas para diferentes propósitos, no limitándose a los videojuegos. Esto motiva la aparición de una variedad de aplicaciones que explotan las posibilidades del dispositivo *Kinect*. Una de ellas es *KinectFusion*[Sha], biblioteca orientada a la reconstrucción tridimensional de objetos en tiempo real, que

brinda la posibilidad de que tanto éstos como el sensor estén en movimiento y de esta forma realizar capturas en 360 grados.

2.2.2. Procesamiento de nube de puntos

Al utilizar técnicas de obtención de geometría el resultado generado es una nube de puntos. Esta nube de puntos por lo general es densa, por lo que es necesario simplificar este modelo con el objetivo de eliminar redundancia y mejorar el rendimiento durante su procesamiento. Un problema adicional al volumen de la información obtenida es que comúnmente se introduce ruido, es decir, información distorsionada que no corresponde con puntos de la realidad. Para solucionar este problema se realiza un suavizado en el procesamiento de la nube de puntos[Pau].

Las heurísticas existentes para reducir nubes de puntos pueden clasificarse en tres grandes grupos [Mar]:

- *Clustering methods*: consiste en obtener grupos de la nube de puntos en donde cada grupo se remplaza por un conjunto de puntos representativos en él. Los grupos se pueden construir utilizando un enfoque incremental en el cual estos son creados iniciando por un punto aleatorio y agregando puntos vecinos hasta llegar a una cantidad establecida de elementos. Otro enfoque es el jerárquico, en donde se particiona el conjunto de puntos recursivamente hasta conseguir grupos de un tamaño predefinido.
- *Iterative simplification*: se recorren iterativamente los puntos de la nube contrayendo parejas en un único punto. Se evalúa el error introducido, utilizando mínimos cuadrados, que se generan en la contracción comparándolos con el error que se obtendría al contraerse con otro punto vecino y eligiendo la contracción que introduce menor error al sistema. La simplificación se da por finalizada por haber logrado la cantidad de puntos deseada o por superar una cota de error a introducir en el sistema.
- *Particle simulation*: se generan nuevos puntos que sustituyen la nube de puntos original. Se generan conjuntos de partículas que se mueven aleatoriamente en la superficie hasta lograr un balance.

Luego de simplificar la nube de puntos se construye un modelo tridimensional a partir de ella utilizando mallas de triángulos debido a la simplicidad de los algoritmos que dibujan este tipo de figuras geométricas. Esto permite que sean implementados fácilmente en hardware además del beneficio de que cualquier polígono con más de tres caras puede representarse como un conjunto de triángulos [Ós].

A continuación se detallan los algoritmos asociados a cada paso de un típico procesamiento de malla que toma como entrada una nube de puntos, realiza un sub-muestreo y

suavizado de la misma, calcula las normales en cada punto de la nube y finalmente aplica algoritmos de reconstrucción de la malla. Esto es implementado en bibliotecas como *VcgLib*[Vcg] y *CGAL*[CGA].

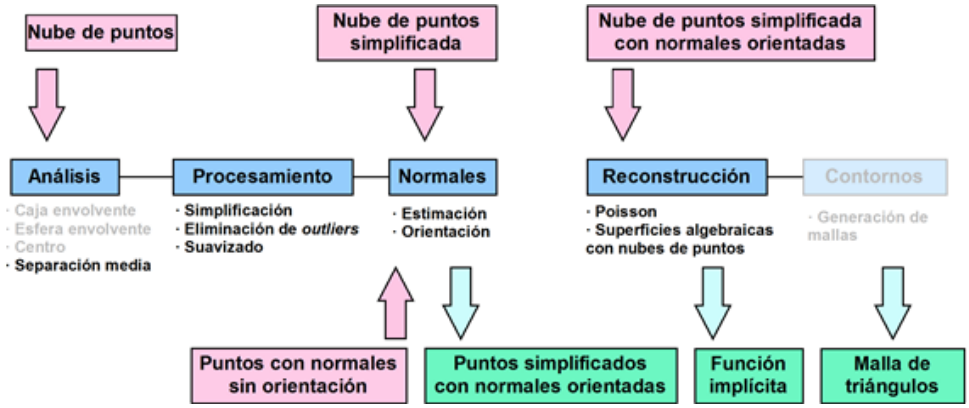

Figura 2.17: Flujo típico de procesamiento de nubes de puntos

Muestreo *Poisson-disk*

El muestreo de variables aleatorias es una técnica utilizada para una gran variedad de aplicaciones como procesamiento de imágenes y geometrías. Particularmente, el muestreo *Poisson-disk* se utiliza para la ubicación aleatoria de objetos en mundos artificiales, algoritmos de texturas procedurales y procesamiento de geometrías o mallas. Esta técnica genera conjuntos de puntos con la propiedad de que sean suficientemente cercanos pero con la restricción de no estar más próximos unos de otros que una distancia mínima predeterminada.

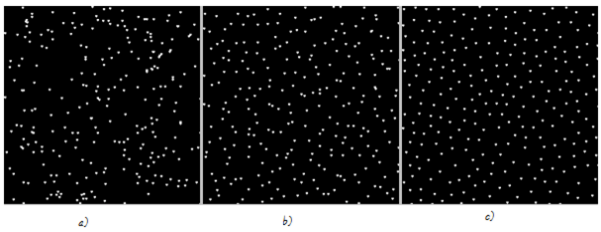

Figura 2.18: a) Posición x e y generadas aleatoriamente. b) Imagen dividida en celdas. Puntos aleatorios generados en cada celda. c) Muestreo *Poisson-disk* en dos dimensiones.

En líneas generales, este algoritmo genera puntos alrededor de los ya existentes en la muestra y valida si pueden ser agregados al conjunto final en caso de no violar la regla de la mínima distancia a los vecinos. Se genera una grilla en dos o tres dimensiones, dependiendo del escenario de aplicación, en la cual cada celda contendrá al final del proceso a lo sumo un punto. Una grilla adicional es utilizada para realizar búsquedas rápidas y dos conjuntos de puntos son mantenidos durante el procesamiento para poder diferenciar los que han sido generados y los que aún necesitan procesamiento. La implementación realizada en *VcgLib* recibe tres parámetros:

- 1) La cantidad de puntos en la muestra: el radio de cercanía es calculado en base a este parámetro.
- 2) El radio: es utilizado para calcular el tamaño de la muestra óptimo en base a la malla inicial.
- 3) Sub muestreo: indica si la muestra de Poisson es un subconjunto de la muestra inicial o si se deberán generar nuevos puntos aleatoriamente.

Reconstrucción de normales

Este algoritmo computa las normales en cada elemento de un conjunto de puntos sin la necesidad de explorar la conectividad de los triángulos. Por ello es muy útil para objetos tridimensionales sin información de caras. Se detalla un pseudo-código del método:

Paso 1: Identificar los planos tangentes para aproximar localmente la superficie y estimar así los vectores normales.

Para cada vértice:

- Calcular el centro geométrico del plano tangente en el punto como el promedio de los K puntos más cercanos.
- Calcular la normal asociada al centro geométrico. Se utiliza la matriz de covarianza en el punto, contemplando los mismos K vecinos más cercanos de la muestra y los valores y vectores propios de la matriz de covarianza. Finalmente, ordenando los vectores propios, la estimación del vector perpendicular corresponde al vector propio de menor valor. Este método es conocido como *Principal Component Analysis (PCA)*.

Paso 2: Construir un grafo en donde cada punto está conectado a los K vecinos más cercanos (grafo de Riemannian)

Se crea un grafo en cuyos nodos se guardan todas las aristas incidentes a los K vecinos más cercanos. A cada arista se le asigna un peso igual al valor absoluto del producto escalar de la normal en el punto con la normal en cada uno de los K vecinos:

$$fabs(nodoActual->normal.K\_Vecinos[n]->normal)$$

Paso 3: Calcular el árbol de expansión mínimo sobre el grafo de Riemannian y recorrerlo para orientar las normales.

Dado un grafo conexo, no dirigido, con sus aristas con un peso asignado, se llama árbol de expansión mínimo al sub-grafo con forma de árbol que conecta todos los nodos con

un peso total mínimo conteniendo todos los nodos del grafo inicial. El grafo de entrada es el construido en el paso anterior y se utiliza el algoritmo de Kruskal[†], uno de los varios algoritmos que resuelven el problema de encontrar un árbol de expansión mínima de un grafo. Una vez construido el árbol de expansión, lo único que se hace es recorrerlo en orden e invertir el sentido de los vectores normales en caso de ser necesario. La condición para efectuar dicha corrección se basa en el ángulo del nodo siendo inspeccionado en comparación a todas las direcciones de las normales de los vecinos conectados a dicho nodo.

Reconstrucción de malla de Poisson

Para reconstruir la malla a partir de la nube de puntos y sus normales, se utiliza el algoritmo de reconstrucción de Poisson[Mic]. Se computa una función indicadora χ definida de la siguiente forma:

$$\begin{cases} \chi = 1 & \text{si puntos dentro del modelo} \\ \chi = 0 & \text{si puntos fuera del modelo} \end{cases}$$

La estrategia se basa en la estrecha relación que hay entre los puntos de la muestra, orientados por sus normales, y la función indicadora de la muestra. Específicamente el gradiente de la función indicadora es un espacio de vectores, de valor nulo en todo el espacio excepto en puntos cercanos a la superficie, donde son iguales al vector normal a ella. Es por eso que puntos orientados pueden ser vistos como muestras del gradiente de la función indicadora del modelo tridimensional en cuestión y es por este mismo motivo que el problema de reconstrucción de una malla puede ser visto como un problema de Poisson estándar, es decir, computar la función escalar F cuya divergencia del gradiente se iguala a la divergencia del espacio de vectores de las normales. Fnalmente se obtiene una reconstrucción aproximada de la malla mediante la extracción de la superficie de nivel apropiada.

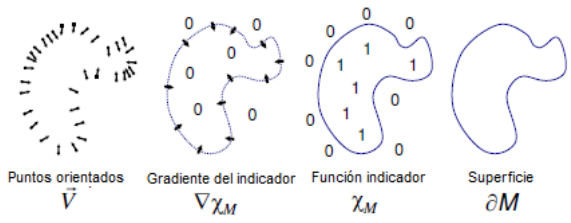

Figura 2.19: Reconstrucción de Poisson en dos dimensiones.

Capítulo 3

Solución planteada

3.1. Descripción general

En este proyecto se desarrolla una aplicación para la edición y ejecución de un espectáculo de *video mapping* que se denomina *Video Mapping Tool (VMT)* la cual está basada en los enfoques anteriormente descriptos: bidimensional y tridimensional. Esta aplicación dispone de un editor tridimensional en el cual se importan modelos en formatos específicos permitiendo visualizarlos desde distintos puntos de vista mediante cámaras virtuales. De forma análoga permite la creación de objetos bidimensionales para generar así un modelo mixto, adaptable a las distintas necesidades de los usuarios.

Para la producción del espectáculo, *VMT* brinda distintos efectos como la proyección de imágenes, videos y degradado de colores, los cuales se pueden aplicar de igual forma a objetos bidimensionales y tridimensionales. A su vez existen efectos específicos para objetos tridimensionales que permiten variar su posición en el tiempo. *VMT* brinda una línea de tiempo sobre la cual es posible definir el orden de ejecución de estos efectos, mediante su asociación a un instante del espectáculo. A su vez también es posible especificar la ejecución de efectos desencadenados por eventos de teclado o *MIDI*. Mediante el editor que brinda *VMT*, es posible asociar una pista de audio que acompañará al espectáculo.

La aplicación tiene la capacidad de utilizar múltiples proyectores, permitiendo así cubrir una amplia superficie. El resultado de estas proyecciones puede ser previsualizado de forma centralizada en el editor, logrando visualizar desde diferentes ángulos los efectos aplicados sobre el modelo. La calibración de la proyección es asistida por *VMT* mediante distintos mecanismos para objetos bidimensionales y tridimensionales utilizados en la escena.

La persistencia del espectáculo se logra almacenando y cargando la definición del mismo desde archivos *XML* estándar.

VMT es una aplicación multiplataforma de código abierto, basado en *OpenFrameworks*[opeb] y *Qt Framework*[QT] que permite su compilación y ejecución en entornos

Microsoft Windows, *Linux* y *Mac OS X*.

3.2. Funcionalidades

3.2.1. Manejo de escena

Una escena en *VMT* es una colección de objetos, efectos y cámaras que serán representados gráficamente. Se cuenta con un motor de dibujado que se encarga de generar dicha representación brindando soporte a la visualización de elementos gráficos bidimensionales y tridimensionales en pantalla o proyectados. También brinda el soporte para el mapeo de texturas de imágenes o videos sobre estos objetos. Este motor ofrece una interfaz para el manejo y edición de objetos y efectos que contiene la escena.

Los elementos bidimensionales manejados por *VMT* son cuadriláteros que en el contexto de la aplicación se denominan *quads*. Éstos se pueden posicionar en pantalla de forma tal que al ser proyectados cubran una superficie total o parcial de la escena. Luego, sobre estos *quads* se aplicarán texturas en forma de imagen o video, para que la proyección deforme estas texturas adaptándolas a la forma del *quad*. Los *quads* también pueden ser utilizados como máscaras para cubrir zonas en donde no se desee proyectar. Esto puede ser útil cuando se desea proyectar sobre una superficie con forma compleja en la que un *quad* no se adapta a la sección deseada. Una máscara se logra con un *quad* opaco de color negro ya que este color es el menos visible al ser proyectado.

Los elementos tridimensionales representan objetos más complejos y ofrecen una mayor versatilidad para realizar mapeos sobre ellos. El formato utilizado para representar los elementos tridimensionales es *3DS*[3DSa] por lo cual es posible cargar al motor cualquier geometría que lo respete. La edición de la geometría de estos objetos tridimensionales es en general un tema complejo que lo resuelven varias aplicaciones de edición tridimensional entre ellas *3D Studio*[3DSb] y Blender[Ble]. Por esta razón es que se delega la edición de las formas tridimensionales a programas para este propósito. Los programas de edición además de permitir modificar la geometría manipulando vértices y caras, también posibilitan la definición de materiales para aplicar propiedades comunes a una cierta selección de caras de la malla tridimensional[†]. *VMT* utiliza estas selecciones de caras para mapear texturas sobre ellas resolviendo de esta forma el mapeo en tres dimensiones. Este mapeo es controlado mediante la definición de coordenadas *UV* en cada vértice del modelo. Las aplicaciones de edición tridimensional antes mencionadas proveen herramientas para hacer esta tarea de forma intuitiva. Ejemplo de esto es la definición de coordenadas de mapeo a un cilindro para mapear una textura en una columna cilíndrica.

En *VMT* los elementos gráficos por sí mismos no son representables, siendo necesario asignarles un material y es a éste al que se le asigna un color o una textura con la cual se representará el objeto. En la implementación de los materiales se utilizan *shaders* que son programas que se ejecutan en los procesadores gráficos y tienen como ventaja ser altamente paralelizables, permitiendo realizar cálculos de forma rápida y eficiente. En la

aplicación se utiliza el lenguaje de *shaders GLSL*[GLS] para *OpenGL* para la implementación de un *shader* básico que combina una textura de imagen o video con un color, obteniendo así el resultado final del material. Este color tiene un canal alfa† que permite representar texturas y colores con distintos niveles de transparencia.

Para visualizar los elementos tridimensionales es necesario definir un punto de vista. Esto se logra mediante la utilización de una cámara virtual y la configuración de sus propiedades, entre ellas su ubicación y orientación. Los proyectores con los que se realiza el espectáculo de *video mapping* son también representados como cámaras, aunque no toda cámara virtual representa un proyector, ya que es posible contar con cámaras adicionales que representen puntos de vista de interés para el usuario como por ejemplo el de un observador del espectáculo. Las cámaras permiten ajustar sus parámetros de forma similar a otros paquetes de animación por software y a cámaras cinematográficas en general. Estas operaciones representan movimientos de la cámara llamados *Roll*, *Orbit*, *Dolly* y *Pan*. *Roll* se refiere al movimiento en donde la posición de la cámara y el punto de vista son fijos y ésta rota a través del eje formado por ellos. *Orbit* es el movimiento en el que la cámara orbita alrededor de su punto de vista, es decir, la posición de la cámara cambia pero siempre visualizando el mismo punto y manteniendo la distancia a este. *Dolly* es el movimiento en donde se mantienen el punto y la dirección de vista pero se mueve la posición de la cámara acercándose o alejándose del objetivo. Por último, *Pan* es el tipo de movimiento en el cual se mueve la cámara manteniendo un paralelismo con la dirección de vista, cambiando la posición y el punto de vista.

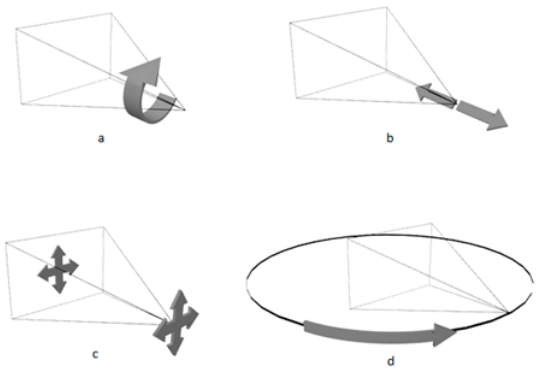

Figura 3.1: Movimientos de cámara. a) *Roll* b) *Dolly* c) *Pan* d) *Orbit*

Además de estos parámetros de ubicación, también existe el parámetro *Field Of View (FOV)* o campo de vista que es el ángulo de apertura de la cámara. Para el caso de cámaras que representan proyectores, este parámetro debe coincidir con el ángulo de apertura del proyector para ajustar correctamente las escenas con respecto a lo que muestra la cámara en pantalla. Mediante estas operaciones es que las cámaras virtuales de *VMT* ajustan sus parámetros para que la proyección de objetos tridimensionales coincida con las superficies proyectadas. Hay que destacar que las cámaras proveen un punto de vista únicamente para los elementos tridimensionales. Los *quads*, que como se mencionó, son construcciones

bidimensionales, no dependen del punto de vista de la cámara y se representan en un sector de la pantalla relativo a ésta. Es por esto que los *quads* no alteran su posición en pantalla al ajustar los parámetros de la cámara.

Los *quads* se pueden organizar en grupos que a su vez se organizan en una o más capas. Cada capa pertenece a una única cámara de manera de representar las zonas a cubrir por un solo proyector. Es posible aplicar una transformación inicial a una capa, la que se aplicará a todos los *quads* o grupos de *quads* contenidos en ella. Esto es útil para ajustar la proyección de los *quads* en caso de mover el proyector levemente y que esta proyección deje de coincidir con las superficies, lo cual provee la funcionalidad básica y de bajo nivel para el mecanismo de calibración bidimensional que se detallará más adelante.

Las texturas se asocian a grupos de *quads* que permiten variar la manera en que se mapean las texturas a los *quads* que los componen. Éstas pueden ser mapeadas por cara, en donde a cada uno de los *quads* se le aplicará toda la textura, o también de forma plana, es decir, la textura será aplicada al conjunto entero de *quads* como si estos fueran trozos de un *quad* de mayor tamaño. Con estas variantes de proyección se brindan distintas posibilidades para que el artista pueda lograr los efectos utilizados normalmente en espectáculos de *video mapping*.

Figura 3.2: Izquierda: Proyección plana. Derecha: Proyección por cara.

La escena también contiene los efectos visuales que se ejecutarán durante el espectáculo. Estos efectos son la asignación de texturas en forma de imagen o video, ya sea a grupos de *quads* u objetos tridimensionales, el *fade*[1] en un material desde un color inicial a otro final, y la animación de la posición de un objeto tridimensional de un origen a un destino en un rango de tiempo.

El efecto de mapeo de texturas a pesar de ser básico es el más potente, pues permite lograr infinidad de resultados al momento de proyectar texturas sobre superficies, basando su complejidad en la creación de las texturas de imagen o video que se deseen utilizar. Al permitirse mapear estas texturas sobre objetos tridimensionales no es necesario distorsionar la forma de los videos generados sino que será *VMT* quien los adaptará a la forma del objeto a mapear. El efecto *fade* puede ser combinado en un objeto que ya disponga de una textura y así permitir por ejemplo su atenuación o resaltado.

[1]Transición de un color inicial a uno final en un rango de tiempo.

A medida que transcurre la ejecución de los efectos se modifica sucesivamente la escena, y por lo tanto su estado en cada momento será el que resulte luego de la finalización de los efectos. Por ejemplo, si un objeto es de color rojo y se le aplica un efecto *fade* al azul su color luego de la ejecución del efecto continuará siendo azul.

Como se mencionó anteriormente, una de las tareas principales de *VMT* es la representación en pantalla o proyección de los elementos gráficos. La implementación del dibujado se realiza en un bucle principal de la aplicación encargado de actualizar todos los objetos de la escena, procesar los eventos pendientes y finalmente desplegar el resultado. Este es un proceso cíclico que se repite varias veces por segundo. En cada uno de estos ciclos se generan los cuadros, también llamados *frames*, que se muestran como resultado final. Para dar la ilusión de continuidad la frecuencia de estos ciclos debe ser alta. En el caso de *VMT*, ésta se limita a 60 ciclos por segundo aunque alcanzar esta frecuencia dependerá de varios factores como la cantidad de *quads*, el tamaño de las texturas utilizadas y capacidad de procesamiento y memoria de los equipos. Si esta frecuencia se encuentra muy por debajo de los 60 ciclos, la reproducción del espectáculo no será fluida y se podrán percibir saltos o pausas al ejecutarlo. Para que esta frecuencia efectivamente se logre es necesario que el procesamiento de cada uno de los ciclos del bucle principal sea eficiente en cuanto al costo de procesamiento en cada tarea u operación. A continuación se muestra un pseudo-código del bucle principal de dibujado y se analizan cada una de las acciones que se toman, su impacto en cuanto al rendimiento del programa y las medidas que fueron tomadas para su optimización:

Algorithm 1 Pseudo-código bucle de dibujado.

```
posicionarCamara();
for all objeto3d obj en la escena do
    for all cara c de obj do
        mat = cargarMaterial(obj);
        c.dibujar(mat);
    end for
end for
resetCamara();
for all quad q en la escena do
    mat = q.cargarMaterial();
    q.dibujar(mat);
end for
```

Un bucle de este estilo cuenta con varios cuellos de botella. La carga de materiales implica copiar una textura a la zona de memoria de video correspondiente para utilizarla. Esto podría ser un proceso lento si no se disponen de las texturas precargadas en memoria. Es por esta razón que *VMT* desde un comienzo carga las texturas en memoria y luego utiliza referencias para identificarlas y asociarlas a una unidad de textura de *OpenGL*[2] para que se encuentre disponible al momento de ser utilizada. Otra optimización realizada en la implementación de *VMT* está en el uso de *shaders* para el cálculo del resultado visual final. Los *shaders* son ejecutados en la unidad de procesamiento gráfico (*GPU*[†]) en forma

[2] www.opengl.org/wiki/Texture

paralela y por lo tanto son óptimos para ejecutar en bucles de este estilo. La visualización fluida de un espectáculo de *video mapping* es difícil de predecir ya que, como se puede ver en el pseudo-código, los tiempos de ejecución dependen de la cantidad de materiales, caras, objetos tridimensionales y *quads* incluidos en la escena, así como también del hardware disponible. Si se utilizan videos en las texturas, su compresión y resolución también afecta la fluidez del espectáculo. Preferentemente los videos deberían estar codificados con *codecs* óptimos para ser utilizados en tiempo real. El mismo problema ocurre con formatos de imágenes comprimidas, pero teniendo un menor impacto en el rendimiento de la reproducción del espectáculo.

3.2.2. Multi proyector

Uno de los puntos fuertes de *VMT* es la posibilidad de editar y visualizar un espectáculo de *video mapping* utilizando más de un proyector. Para brindar soporte a esto, *VMT* fue diseñada con una arquitectura maestro-esclavo, implementada en dos componentes ejecutables de forma independiente. El nodo maestro orquesta la reproducción del espectáculo enviando a los nodos esclavos instrucciones para la ejecución de los efectos. Estos nodos se pueden distribuir en varias computadoras, logrando la comunicación entre ellas mediante un protocolo de comunicación implementado utilizando *Open Sound Control (OSC)*, estando a su vez este último construido sobre *User Datagram Protocol (UDP†)*. Los mensajes se componen por una cadena que lo identifica y un conjunto variable de parámetros que depende de cada mensaje en particular. Dependiendo del tipo de mensaje, el nodo maestro decidirá si enviarlo a un nodo en particular, por ejemplo para el caso de operaciones sobre *quads* ya que éstos se representan únicamente en un nodo, o enviarlo a todos los nodos al mismo tiempo, por ejemplo para operaciones sobre objetos tridimensionales ya que todas las cámaras tienen visibilidad sobre ellos. Por la naturaleza del protocolo *UDP*, es preferible contar con conexiones cableadas en lugar de inalámbricas dada la alta tasa de pérdida de paquetes que existe en este tipo de redes, lo cual podría causar desincronización de algunos nodos esclavos. *OSC* también permite a cualquier aplicación capaz de enviar mensajes de este tipo y que conozca el formato de los mismos, controlar los esclavos y por consiguiente lo que se proyecta en los correspondientes proyectores.

La edición del espectáculo se realiza exclusivamente en el nodo maestro quien envía órdenes mediante el protocolo de comunicación a los nodos esclavos para que éstos representen objetos tridimensionales y *quads*, aplicándoles efectos visuales. En *VMT*, los nodos se definen agregando cámaras virtuales que representan proyectores a la escena, asignando propiedades de ubicación para cada una en particular para luego definir la red de nodos esclavo con la información de conectividad necesaria y su correspondiente asociación al proyector conectado al nodo.

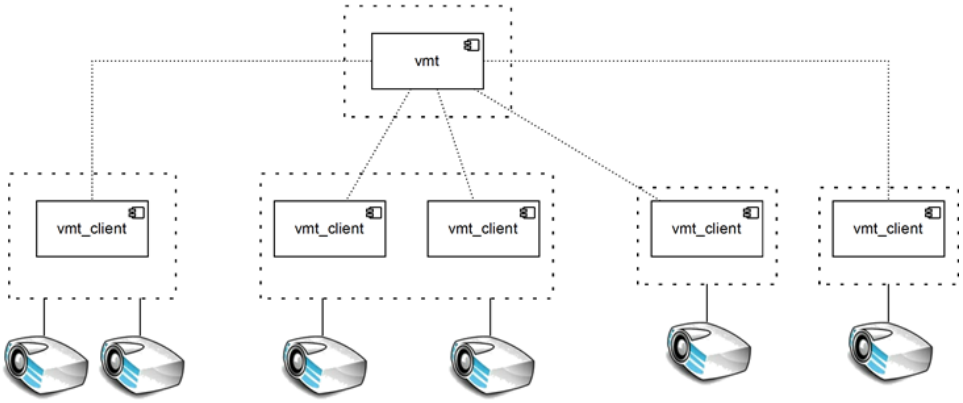

Figura 3.3: Escenarios posibles de múltiples proyectores en VMT

El soporte de múltiples proyectores puede lograrse utilizando diferentes esquemas de distribución. El más sencillo es el de conectar varios proyectores con más de una salida de video desde la misma computadora. Esto se logra mediante la utilización de una tarjeta de video con salidas múltiples, conectando cada proyector a cada una de estas salidas. Luego, se configura el sistema operativo para que extienda la resolución en las múltiples salidas y de esta forma se dispondrá de un área de trabajo mayor. Los diferentes sectores del área de trabajo serán desplegados por los proyectores conectados. El componente esclavo de *VMT* permite indicar al inicio de un espectáculo la resolución y la posición en pantalla en donde se desplegará. Controlando estos parámetros se permiten distintas variaciones en la configuración de la aplicación y en la disposición de los proyectores. Esto también se puede lograr conectando dispositivos que dupliquen o tripliquen las salidas de video como las tarjetas *DualHead2Go* o *TripleHead2Go*[Matb] y de esta manera conectar dos o tres proyectores. El resultado es similar al anterior, extendiendo el área de trabajo a la resolución duplicada o triplicada, mediante el soporte del sistema operativo. Estas variantes en la forma de conectar los proyectores son útiles cuando se desea proyectar sobre una superficie tan extensa que un único proyector no pueda abarcarla completamente, con la restricción de que los proyectores deben estar organizados de forma contigua. En cambio, si queremos proyectar sobre zonas de la escena no contiguas entre si, lo ideal es contar con proyectores conectados a nodos distribuidos e independientes tanto al momento de la producción del espectáculo como al de su ejecución. Esto es posible explotando al máximo las posibilidades de distribución de *VMT*, por ejemplo, ejecutando un nodo distinto en cada una de las computadoras distribuidas.

3.2.3. Calibración

VMT tiene la capacidad de calibrar las cámaras virtuales para que la proyección reflejada por los proyectores se corresponda y ajuste a la geometría de la escena. Como se mencionó anteriormente, *VMT* permite trabajar simultáneamente de forma bidimensional y tridimensional, por lo tanto, cada uno de estos modos de trabajo requiere distintos mecanismos de calibración.

La calibración bidimensional se basa en ajustar los vértices de todos los *quads* de una capa de forma conjunta. Inicialmente, al preparar el modelo para el espectáculo, los *quads* son posicionados en pantalla al mismo tiempo que éstos son proyectados, cubriendo las áreas de interés a proyectar. Luego de este posicionamiento inicial, cualquier movimiento de los proyectores causará que se desajuste simultáneamente la proyección de todos los *quads*. Si esto ocurriese, se hace necesario reposicionar los *quads* para que vuelvan a cubrir las áreas a proyectar. *VMT* ofrece la posibilidad de ajustar todos los *quads* pertenecientes a una misma capa de forma simultánea aplicando una homografía. Esta transformación es la que ocurre naturalmente en la proyección al variar la posición u orientación del mismo, por lo tanto es necesario aplicar la transformación inversa para que la proyección de los *quads* se ajuste nuevamente a la superficie. La homografía queda definida conociendo cuatro puntos de origen y cuatro de destino, por lo que se puede calibrar la proyección escogiendo los cuatro vértices de un *quad* y cuatro puntos en donde se desea proyectar dicho *quad*.

Por otra parte, la calibración tridimensional se basa en hacer coincidir los parámetros de posición, orientación y proyección de las cámaras virtuales con los de los proyectores físicos. Este ajuste de parámetros se realiza manualmente por el usuario, utilizando las transformaciones de cámara explicadas anteriormente, *dolly*, *pan*, *roll* y *orbit*. Esta calibración es distinta para cada cámara virtual y proyector por lo que debe hacerse de forma independiente para cada uno de estos.

3.3. Interfaz gráfica de usuario

La interfaz gráfica de *VMT* cuenta con ventanas y diálogos de altas, bajas y modificaciones para todos los tipos de objetos existentes en una escena. Fue diseñada en base a barras de herramientas flotantes de forma de tener visibilidad del editor tridimensional en todo momento sin ocupar áreas considerables del espacio de trabajo y poder observar así de mejor forma el impacto de las acciones realizadas.

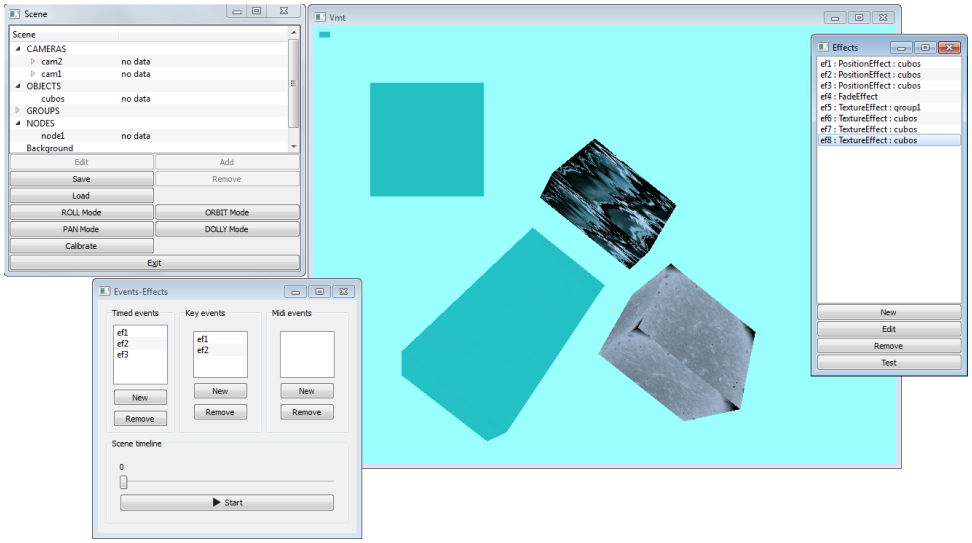

Figura 3.4: Vista general de la interfaz de usuario de *VMT*

Al inicio de la ejecución de *VMT* se cuenta con una vista gráfica de la escena, un árbol con los elementos contenidos en la escena, la lista de definición de efectos y una línea de tiempo con las listas de efectos por tiempo, evento de teclado y *MIDI*. La ventana de escena brinda las funcionalidades principales de la aplicación, desde donde se manejan los objetos bidimensionales y tridimensionales, cámaras, capas y nodos distribuidos del espectáculo. También esta ventana ofrece los controles para la calibración de las cámaras virtuales para cambiar los modos de control ofrecidos.

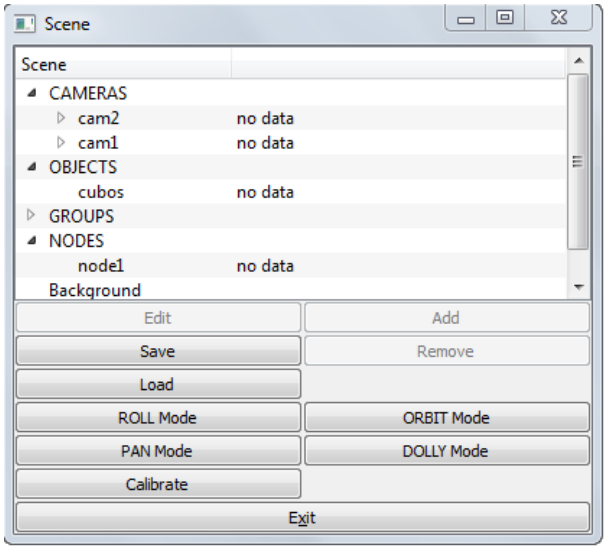

Figura 3.5: Ventana para manejar la escena

3.3.1. Cámaras, capas y *quads*

Las cámaras virtuales tienen todas las propiedades usuales que se pueden encontrar en paquetes de software de animación tridimensional como son la posición, *eye*, *up*, *FOV*, aspecto, *near* y *far*. *Eye*, *up* y posición son ternas de coordenadas tridimensionales que identifican respectivamente hacia dónde apunta la cámara, su dirección vertical y su punto de origen. El parámetro *FOV* o campo de vista, es el ángulo de apertura de la cámara virtual. A mayor ángulo de apertura, mayor cantidad de objetos serán incluidos en el campo visual de la cámara. Por su parte el aspecto es un valor que representa la relación entre la resolución horizontal y vertical tanto de cámaras como de proyectores. Finalmente, los parámetros *near* y *far*, son propios del modelo y no representan propiedades de los proyectores físicos. Éstos definen la distancia mínima y máxima de los objetos a la cámara, lo que será considerado para restringir la representación de los mismos.

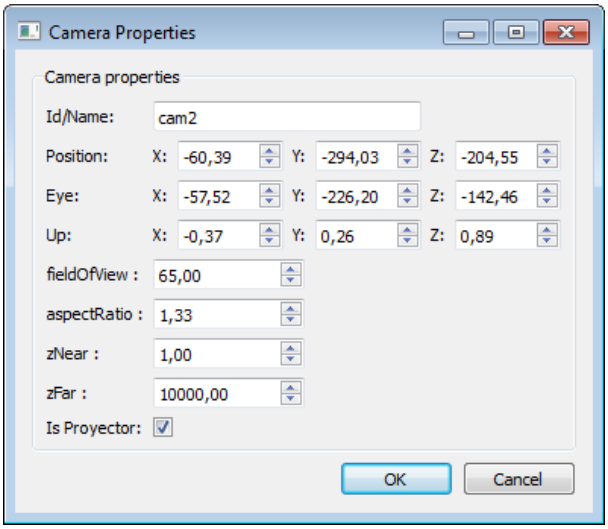

Figura 3.6: Diálogo de propiedades de cámara

Las cámaras virtuales en *VMT* representan puntos de vista desde donde visualizar una escena tridimensional. Estas cámaras virtuales poseen la propiedad *isProjector* que diferencia si esta cámara es utilizada para representar el punto de vista de un proyector o un punto de vista arbitrario. La aplicación utiliza esta información para asociar las cámaras que representen proyectores a los nodos esclavos en una configuración multiproyector.

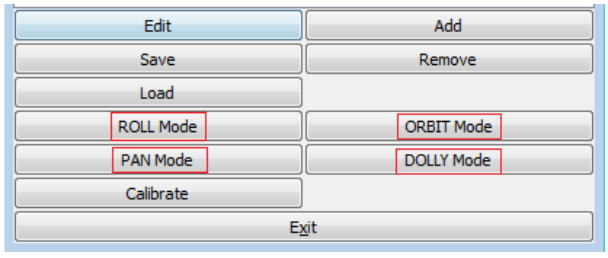

Figura 3.7: Acciones para seleccionar modo de movimiento de la cámara activa

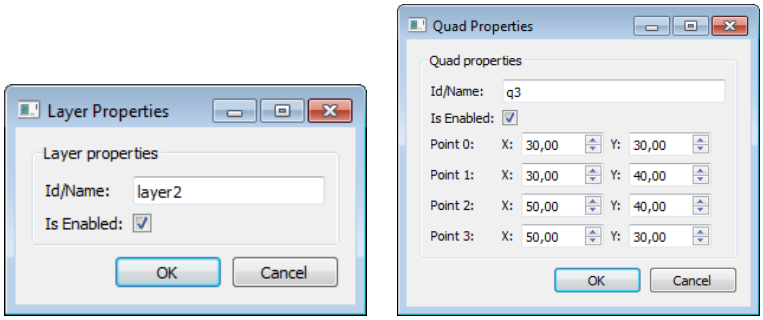

Figura 3.8: Propiedades de los objetos capa y *Quad*

Una vez seleccionado el modo de la cámara, arrastrando el ratón manteniendo presionado el botón izquierdo se mueve la cámara de manera acorde al modo seleccionado. Cada cámara es a su vez un contenedor de capas, cuyo propósito es manejar todo lo relacionado al mapeo sobre estructuras bidimensionales. Por esta razón, las capas son a su vez contenedores de *quads* sobre los cuales se realizan todas las operaciones y efectos disponibles.

Tanto *quads* como capas tienen propiedades que pueden ser modificadas en tiempo de edición para mostrarse u ocultarse completamente. Si se oculta una capa, todos los *quads* contenidos en ella también se ocultarán. En caso de revelarla, todos los *quads* visibles volverán a desplegarse.

3.3.2. Objetos tridimensionales

VMT permite el manejo de objetos tridimensionales, importándolos de archivos con formato de malla triangular de tipo *3DS*. Fue escogido este formato debido a que es un estándar utilizado por los programas de modelado tridimensional más populares, y por la disponibilidad de bibliotecas que facilitan su manipulación. Como se mencionó anteriormente, la edición de los objetos tridimensionales se delega a aplicaciones para este propósito, limitándose *VMT* a su posicionamiento.

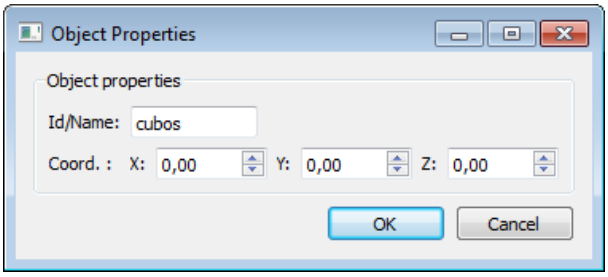

Figura 3.9: Diálogo de propiedades de objeto tridimensional

3.3.3. Efectos

Se proveen ventanas flotantes tanto para la definición de efectos como para la asociación con sus disparadores como ser un instante específico, un evento de teclado o un evento *MIDI*. Si un efecto es definido para ser desplegado en un instante dado, éste se podrá visualizar durante el transcurso de la línea de tiempo. En caso de ser definido por evento de teclado o *MIDI*, éste será ejecutado cada vez que el evento asociado suceda.

El efecto de posición de objetos tridimensionales permite animar el objeto trasladándolo de una posición inicial a una final en un rango de tiempo.

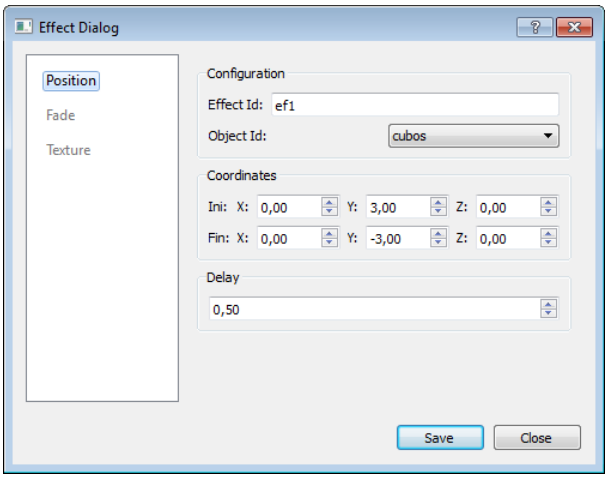

Figura 3.10: Diálogo de definición de efecto de tipo posición

El efecto *fade* es aplicable a grupos de *quads* y permite colorear todos los *quads* del grupo variando desde un color inicial y pasando por toda la gama de colores intermedios hasta llegar al color final en un rango de tiempo. Para la selección de colores se utiliza un diálogo específico para este propósito.

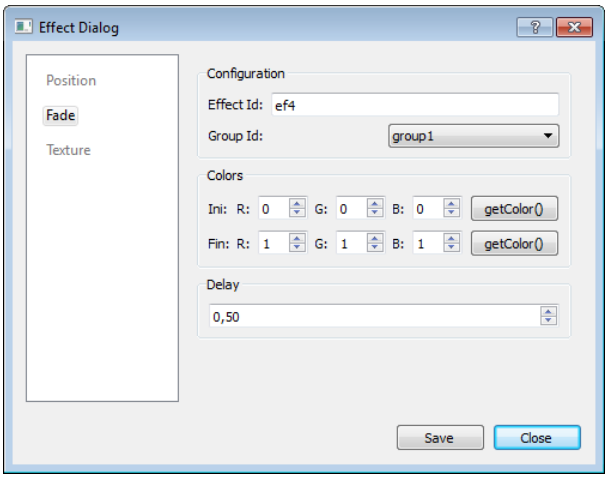

Figura 3.11: Diálogo de definición de efecto de tipo *fade*

Los efectos de tipo textura son aplicables tanto a grupos de *quads* como a objetos tridimensionales, definiendo texturas de tipo imagen o video. En ambos casos es necesario proporcionar la ruta al archivo multimedia correspondiente. Para el caso específico de un efecto de textura asociado a un objeto tridimensional, es necesario proporcionar además la cara o conjunto de caras sobre los cuales se mapeará dicho efecto. Este conjunto de caras se representa en el formato *3DS* asociándole un material, para luego identificarlas mediante su nombre.

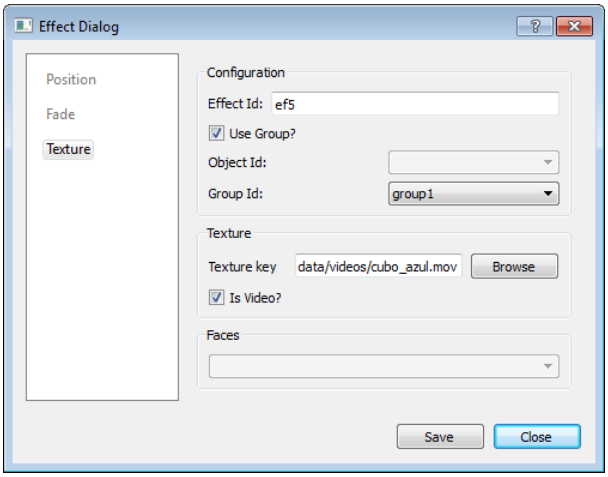

Figura 3.12: Diálogo de definición de efecto de tipo textura

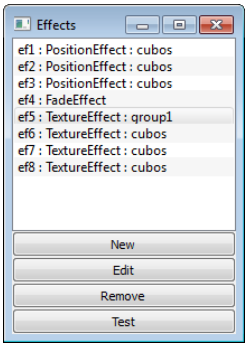

Figura 3.13: Lista de efectos definidos en la escena

Es posible visualizar el efecto que se está creando presionando el botón *Test* de la lista de efectos. Esto disparará la ejecución del efecto seleccionado por una única vez.

3.3.4. Calibración

El proceso de calibración en *VMT* se basa en posicionar cuatro puntos de la superficie donde se proyecta y cuatro puntos de la capa contenedora de *quads* para de esta forma calcular la matriz de transformación de la homografía que los hace corresponder.

Figura 3.14: Diálogo para configurar y ejecutar la calibración

Al ingresar en el modo calibración se muestran en la pantalla de edición cuatro puntos azules de destino y cuatro puntos rojos de origen junto con el diálogo de configuración de la calibración. El diálogo posee controles para posicionar cada uno de los ocho puntos mostrados en pantalla. Se deberán posicionar los cuatro puntos de origen en regiones significativas de la capa donde hayan vértices de *quads*, preferiblemente alejados entre si para de esta forma minimizar el error obtenido al calcular la matriz. Los cuatro puntos de destino se deberán posicionar observando la proyección de éstos sobre la superficie, trasladándolos hasta que coincidan con la superficie que se desea cubrir. Una vez que los ocho puntos estén posicionados se podrá calcular y aplicar la nueva matriz de calibración confirmando con el botón *OK*. Mediante el botón *ReCalibrate* se reestablece la calibración original, descartando los cambios realizados.

3.3.5. Línea de tiempo

Tanto para las fases de producción y previsualización del espectáculo como para su reproducción en vivo, es necesario contar con controles para el manejo de la línea de tiempo.

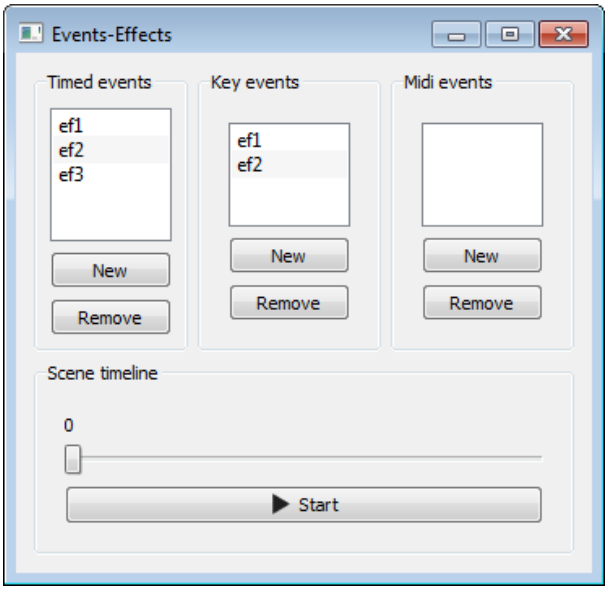

Figura 3.15: Diálogo con línea de tiempo y lista de efectos asociados a tiempo y eventos

La aplicación permite iniciar la ejecución de la línea de tiempo mediante el botón *Start*.

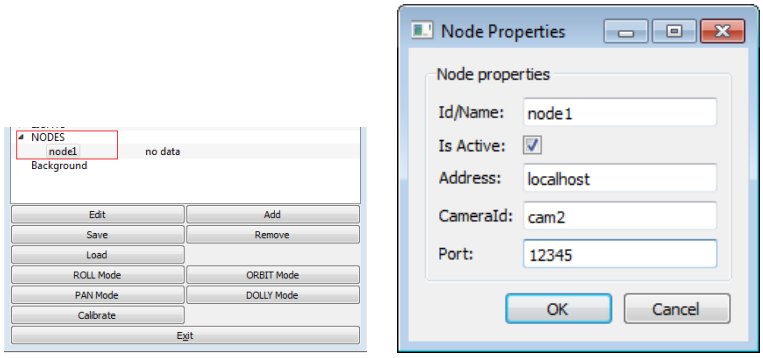

Figura 3.16: Lista de nodos y diálogo de propiedades de nodos

Se provee una vista gráfica representada con una barra deslizante y una vista numérica para reflejar el avance en el tiempo en relación a la duración total del evento. Utilizando la barra deslizante es posible navegar por la línea de tiempo e iniciar la visualización del espectáculo desde cualquier instante deseado. Esto último es muy útil para la fase de producción del espectáculo en la cual es muy común que el usuario se quiera concentrar en un lapso específico en el que se está trabajando. El diálogo contiene los efectos asociados a eventos agrupados por tipo, pudiendo estos referenciar un instante dado, un evento de teclado o proveniente de dispositivos de entrada *MIDI*.

3.3.6. Nodos remotos

Para modelar la red de nodos esclavos de *VMT* en los cuales se conectarán proyectores remotos es preciso definir para cada uno de ellos, su dirección *IP* y número de puerto en donde estarán configurados así como también la cámara definida en la escena que representará al proyector asociado.

3.3.7. Escena en XML

El almacenamiento de la escena y su carga se realiza mediante archivos *XML* estándar. La interfaz provee diálogos específicos para guardar y cargar estos archivos.

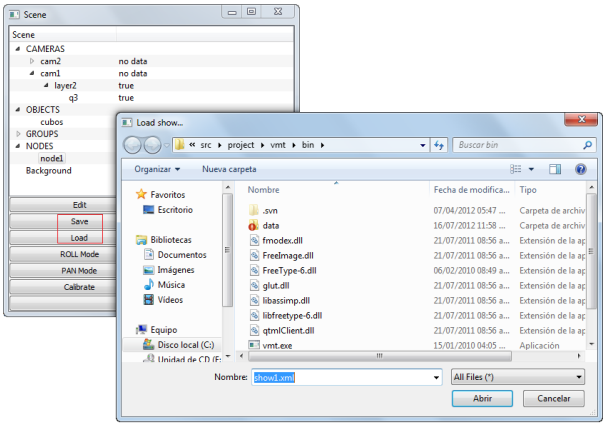

Figura 3.17: Diálogo para cargar y guardar archivos *XML*.

3.3.8. Decisiones de implementación de la interfaz gráfica de usuario

Para la implementación del módulo de interfaz gráfica se utilizó *Qt Framework*[Qt-] como biblioteca de base para la creación de las ventanas, controles gráficos y manejo de la comunicación que ocurre entre las ventanas y los modelos de *VMT*. Esta biblioteca utiliza el patrón de diseño *Model-View-Controller (MVC)*[QTM], lo que exige generar para cada ventana, modelos que manejan sus datos relevantes. Se hizo uso de la funcionalidad para la internacionalización de las interfaces gráficas, las que por el momento están tomando valores por defecto en idioma inglés. *Qt Framework* es multiplataforma y puede ser utilizado en varios de los sistemas operativos más populares como *Linux*, *Mac OS X* y *Windows*, por esta razón es que fue escogido como base para el módulo de interfaz gráfica.

3.4. Prototipo de calibración tridimensional

En el marco de este proyecto se desarrolló un prototipo de calibración basado en un método cuyo objetivo es obtener la posición del proyector con respecto a un sistema de coordenadas ubicado en un punto relativo a la escena. Para ello se utiliza una superficie plana de calibración que puede existir ya en la escena o puede ser ubicada sobre ésta de forma temporal. Esta superficie de calibración deberá ser un plano rectangular en donde uno de los vértices será el centro de coordenadas que se desea determinar. La orientación del plano determinará la alineación del centro de coordenadas con sus ejes X e Y alineados con dos de los bordes del plano y el eje Z perpendicular a éste. Se utiliza además un proyector el cual es representado mediante el modelo *pinhole*.

Para encontrar el sistema de coordenadas buscado, se resuelve el problema opuesto que es encontrar la posición del punto que será origen del nuevo centro de coordenadas con respecto al centro de proyección ubicado dentro del proyector. Este método utiliza tres de los vértices de la superficie de calibración que formarán una base del nuevo sistema

de coordenadas. Las medidas de la superficie de calibración son conocidas por lo que las distancias entre sus vértices también lo son. Desde el centro del proyector se proyectan rayos de luz, tres de los cuales pasan por los vértices de la superficie de calibración y también por el centro de proyección con coordenadas $(0, 0, 0)$. Para obtener la ecuación de estos rayos se precisa únicamente un punto distinto del origen. La ecuación de este punto puede ser obtenida utilizando las coordenadas en pantalla del píxel que se proyecta en cada vértice de la superficie de calibración.

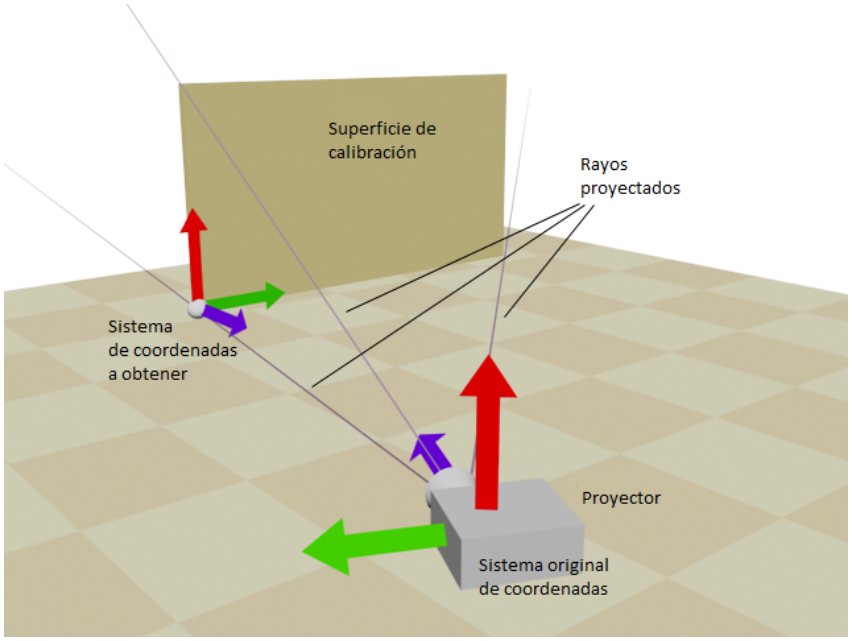

Figura 3.18: Esquema de calibración.

Usando las coordenadas en la pantalla de estos puntos junto con los parámetros intrínsecos del proyector que son su resolución y ángulo de proyección, es posible encontrar las coordenadas del punto con respecto al centro de proyección y por tanto la ecuación del rayo.

$$a(t) = \vec{P}.t, \text{ ecuación del rayo en dónde } \vec{P} \text{ es cualquier punto del rayo.}$$

$$\begin{cases} P(X) = \frac{res_x}{2} - s_x \\ P(Y) = \frac{res_y}{2} - s_y \\ P(Z) = \frac{res_y}{2 \cdot \tan \frac{fov_y}{2}} \end{cases}$$

en donde s_x y s_y son las coordenadas del píxel, res_x y res_y son la resolución horizontal y vertical del proyector y fov_y es el ángulo de proyección vertical del proyector.

Aplicando esta ecuación a cada uno de los puntos a determinar se obtienen las ecuaciones paramétricas de los tres rayos. Vale aclarar que estos puntos P son puntos de los rayos pero no necesariamente coinciden con los vértices de la superficie de calibración. Resta hallar las coordenadas de los vértices determinando el parámetro t para cada ecuación. Para hallar este parámetro se resuelve un sistema de ecuaciones utilizando las tres

ecuaciones de los rayos y las distancias conocidas entre estos puntos.

$$\begin{cases} \|a(t_0) - b(t_1) = j\| \\ \|b(t_1) - c(t_2) = k\| \\ \|c(t_2) - a(t_0) = l\| \end{cases}$$

siendo a, b y c los rayos y j, k y l las distancias entre las parejas de puntos. La solución será entonces los parámetros t_0, t_1 y t_2 que satisfacen el sistema de ecuaciones no lineal. Una aproximación a esta solución se puede obtener utilizando métodos numéricos, por ejemplo, el método *trust-region-dogleg*[Ya-]. Una vez hallados los valores para t_0, t_1 y t_2, las coordenadas de los vértices de la superficie de calibración son $a(t_0)$, $b(t_1)$ y $c(t_2)$. La base para el sistema de coordenadas con origen en el punto P es:

$$x' = \frac{b(t_1) - a(t_0)}{\|b(t_1) - a(t_0)\|}, \quad y' = \frac{c(t_2) - a(t_0)}{\|c(t_2) - a(t_0)\|}, \quad z' = -x' \times y'$$

La ubicación del proyector en este nuevo sistema se obtiene proyectando cualquiera de los rayos en la nueva base de coordenadas.

3.5. Tratamiento de malla

Para generar mallas que puedan ser manejadas por *VMT* se desarrolló separadamente una aplicación que recibe como entrada una nube de puntos a procesar. Como salida se generan mallas triangulares en el formato estándar *OBJ*.

Para la implementación de este módulo se utilizan los algoritmos descriptos en el capítulo de estado del arte, incluidos en *VcgLib* y que forman parte del procesamiento de malla propuesto. Para visualizar y evaluar los resultados esperados fue utilizada la aplicación de código abierto para la manipulación de mallas tridimensionales en diferentes formatos *MeshLab*[Mes]. Particularmente se utilizan los algoritmos de muestreo *Poisson-disk* para reducir y normalizar los puntos de la malla inicial, *normal extrapolation* para el cálculo de normales y reconstrucción de superficies de Poisson para la reconstrucción de la malla final.

Esta aplicación cuenta con una sencilla interfaz de usuario, con una única ventana que recibe los parámetros para la configuración de los algoritmos que se ejecutan en cada uno de los pasos del procesamiento.

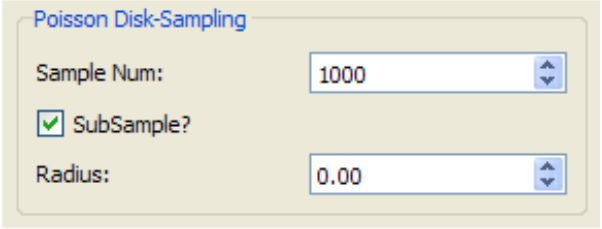

Figura 3.19: Configuración de parámetros del algoritmo *Poisson-disk*

Fase	Vértices	Caras
Nube inicial	5021	9608
Poisson-disk	1776	0
Extrapolación Normales	1776	0
Reconstrucción de Poisson	1959	3910

Tabla 3.1: Comparación de estructura de mallas de entrada y salida en cada fase

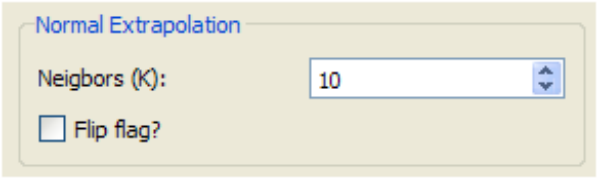

Figura 3.20: Configuración de parámetros para reconstrucción de normales

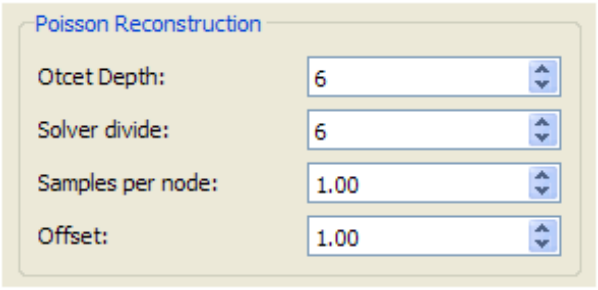

Figura 3.21: Configuración de parámetros para reconstruccion de malla de Poisson

3.5.1. Pruebas y resultados

Para validar el correcto funcionamiento de esta técnica mediante los tres algoritmos descriptos, fue utilizada una malla inicial de 5021 vértices y 9608 caras triangulares. Cabe destacar que si bien se ha mencionado que la malla de entrada debe ser simplemente una nube de puntos, se pueden utilizar mallas con caras, aunque estas serán ignoradas e incluso eliminadas de la malla de salida del primer paso del procesamiento (muestreo de *Poisson-disk*). Luego de experimentar con varios juegos de datos iniciales durante varias ejecuciones del procesamiento, se fijaron de manera personalizada para la malla de entrada algunos parámetros clave. Dado que la muestra inicial tiene alrededor de 5000 puntos, fueron elegidas 5000 muestras para el algoritmo de *Poisson-disk*. Luego, para la extrapolación de normales se utilizan $K = 15$ vecinos para la toma de decisiones locales de aproximación. La aplicación de estos algoritmos resultó ser lo esperado en términos estructurales de cada malla procesada en cada uno de los pasos.

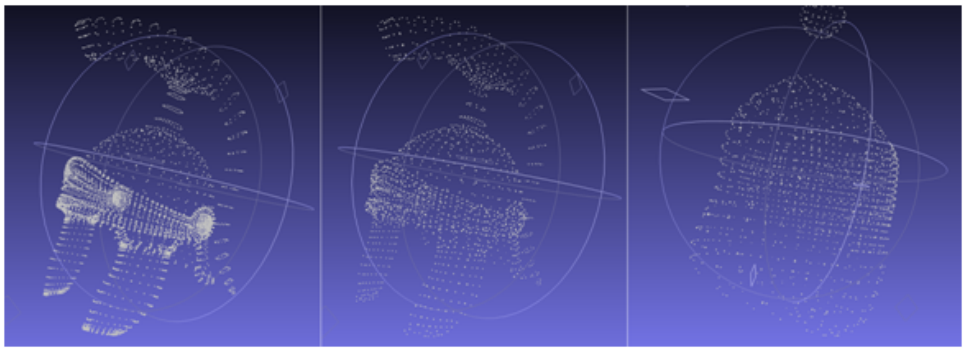

Figura 3.22: 1) Nube de puntos inicial con 5021 vértices. 2) Resultado de muestreo *Poisson-disk* con 1776 vértices. 3) Luego de extrapolar normales y reconstruir la malla con 1959 vértices y 3910 caras.

Capítulo 4

Conclusiones y trabajos futuros

En el marco de este proyecto se desarrolló la herramienta *VMT* para la realización de espectáculos de *video mapping* que implementa un enfoque novedoso, permitiendo la utilización de modelos tridimensionales para representar las superficies a proyectar y posibilitando la aplicación de efectos directamente sobre ellos. Cuenta con una interfaz de usuario interactiva en donde es posible visualizar los efectos en tiempo real a medida que se van diseñando. Se lograron identificar etapas claramente diferenciadas en el proceso de creación de un espectáculo, lo cual fue muy útil para delinear requerimientos específicos para el desarrollo de *VMT*.

En las entrevistas que se mantuvieron con los *VJs* éstos manifestaron un problema común que se da al utilizar más de un proyector cuyas proyecciones se solapan, causando deformaciones no deseadas en la visualización. *VMT* mediante el soporte de múltiples proyectores evita este problema al tener control sobre qué objetos y efectos se muestran en cada proyector. Más en general, el manejo de múltiples proyectores distribuidos posibilita la creación de distintos esquemas de proyección soportando áreas amplias y disjuntas e incluso mapeos en 360 grados.

Los avances durante el transcurso del presente proyecto se han ido publicando en el *blog*[BLO] elaborado para este propósito y permitiendo el seguimiento del mismo. El código fuente de las aplicaciones, así como también otros materiales utilizados durante este proyecto se encuentran disponibles en los repositorios públicos [SOUb] y [SOUa].

Obtención automática de la geometría

Se logró resolver parcialmente el problema de reconstrucción automática de la escena a mapear mediante la implementación de una aplicación que toma una nube de puntos ya capturada, la procesa y genera un objeto tridimensional para luego ser utilizado por *VMT*. Se trabajó en el estudio de las técnicas y componentes existentes para la reconstrucción tridimensional de la escena, principalmente mediante la utilización del método de luz estructurada en donde se experimentaron varios problemas, la mayoría de ellos en la etapa de calibración del sistema cámara-proyector.

Se logró realizar exitosamente una prueba de concepto del método de escaneo de Kyle McDonald *Three-Phase shift*. Si bien este método logra obtener una representación tridimensional, tiene limitantes en cuanto a que las superficies a escanear deben ser continuas y sin cambios bruscos en la profundidad. Es por esta razón que no es aplicable para obtener geometrías correspondientes a escenas comúnmente utilizadas en espectáculos de *video mapping*, ya que en general éstas contienen objetos que no cumplen estas características.

Figura 4.1: Izq: Patrón proyectado sobre sujeto de prueba. Der: Nube de puntos obtenida

Se relevaron otros métodos de escaneo con luz estructurada que se adaptan mejor a un mayor rango de superficies, incluyendo las no soportadas por *Three-Phase shift*, pero no se encontraron prototipos ya desarrollados para realizar pruebas de concepto y fue considerado fuera de los objetivos del proyecto. Adicionalmente, durante el transcurso del proyecto surgió el dispositivo *Kinect* para escanear objetos en tiempo real utilizando una implementación del método de luz estructurada. Con la liberación de su kit de desarrollo se incentivó su uso a desarrolladores de todo el mundo para una variedad de propósitos, lo que motivó a enfocar el alcance del proyecto en el procesamiento de nubes de puntos, independientemente de cómo estos puntos se obtienen, ya que *Kinect* logra buenos resultados y es accesible en términos de costos.

Se implementó un prototipo de calibración tridimensional para obtener de forma automática la ubicación del proyector con respecto a una superficie. Si bien este prototipo se pudo evaluar experimentalmente dando resultados que se aproximaban a lo esperado, necesitaba ajustes debido a las diferencias entre el funcionamiento de un proyector con el modelo ideal asumido. A su vez, el método de calibración se basa en parámetros intrínsecos del proyector como son su ángulo de enfoque el cuál debe ser medido experimentalmente, agregando errores a los cálculos. En etapas posteriores del proyecto, al desarrollar la interfaz gráfica, se implementaron las transformaciones de cámaras virtuales mediante las cuales, si bien de forma manual, se puede lograr un ajuste más preciso.

Espectáculos en vivo

Se realizaron dos espectáculos en vivo con la herramienta *VMT* en una versión alfa: el evento de cierre de Ingeniería de Muestra del año 2010[1] de Facultad de Ingeniería de la Universidad de la República y un espectáculo durante la noche de fallos de fin de curso de Facultad de Arquitectura de la Universidad de la República.

Figura 4.2: Ingeniería de muestra 2010, *quads* se iluminan al ritmo del audio.

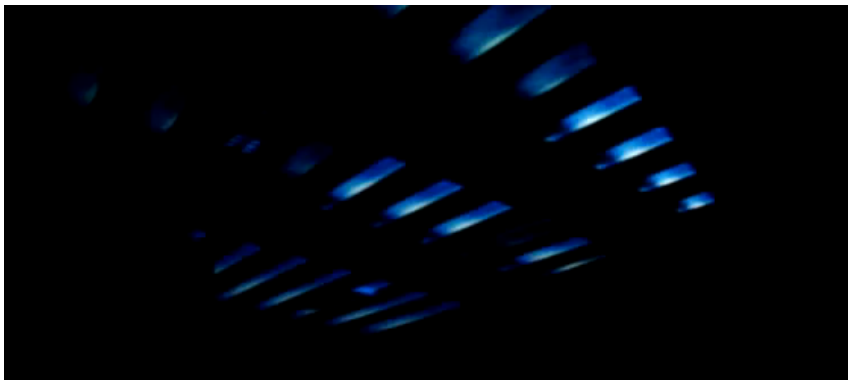

Figura 4.3: Ingeniería de muestra 2010, videos en cada *quad*.

[1] `http://www.fing.edu.uy/eventos/ingenieria_demuestra/2010/index.html`

Figura 4.4: Noche de fallos 2010, Fac. Arquitectura, casilleros iluminados.

Figura 4.5: Noche de fallos 2010, Fac. Arquitectura, casilleros y columnas iluminados.

Figura 4.6: Noche de fallos 2010, Fac. Arquitectura, casilleros y columnas cambiando de colores.

Durante la preparación y ejecución de los mismos se profundizó el conocimiento sobre lo necesario para llevar adelante un espectáculo de estas características, lo que ayudó a asentar las funcionalidades básicas que la aplicación debía cubrir. También se experimentó la importancia de la robustez y tolerancia a fallos que debe tener una aplicación que tiene como propósito la ejecución de espectáculos en vivo.

Desde el punto de vista funcional, se identificaron varios aspectos de la aplicación en las que se debía trabajar para mejorarlos, como ser el agregado de un mismo evento en varios momentos en la línea de tiempo, la agrupación de *quads* para aplicarles el mismo efecto especificándolo una única vez y la ejecución del espectáculo a partir de un instante dado de la línea de tiempo. Todo lo anterior siendo accesible mediante una interfaz gráfica de usuario que adopte criterios comúnmente utilizados en aplicaciones orientadas al diseño.

En las instancias iniciales del desarrollo de la aplicación, se había decidido distribuir los archivos contenedores de la definición del espectáculo en todos los nodos esclavo de *VMT*, enviando desde el nodo maestro únicamente órdenes para ejecutar los efectos en el instante requerido. Esto presentaba varios problemas, por ejemplo durante la edición del espectáculo, ante cualquier modificación en los *quads* u objetos tridimensionales era necesario actualizar el archivo de definición en cada nodo y posteriormente reiniciarlo. Debido a esta fuerte limitación se rediseñó la arquitectura y el protocolo de comunicación entre nodos maestro y esclavo, centralizando la definición del espectáculo en el nodo *VMT* maestro y soportando la creación de todos los elementos mediante nuevos mensajes que fueron agregados a dicho protocolo. Estos mensajes envían a cada nodo esclavo los objetos bidimensionales, tridimensionales, grupos de objetos, y efectos necesarios para la visualización del espectáculo en ese nodo en particular. De estas experiencias también surgió el inconveniente de pérdida de mensajes durante la ejecución debido al uso de redes inalámbricas. Estas redes proveen facilidades para conectar nodos debido a que no requieren cableado pero tienen una alta tasa de colisiones. En el caso de *VMT*, el uso de

estas redes provocó que la pérdida de mensajes durante la ejecución ocasionara un desfasaje en la reproducción en los diferentes nodos esclavo. Por ello se decidió utilizar para el espectáculo de Facultad de Arquitectura una red cableada *Ethernet* y se observaron mejoras notorias en cuanto a la calidad de la transmisión sin notar pérdida de mensajes.

Trabajo futuro

Durante el transcurso del proyecto en sus etapas de investigación del estado del arte y desarrollo de la aplicación *VMT*, así como también producto de la experiencia obtenida en los dos espectáculos en vivo mencionados, se identificaron varias oportunidades de mejora de lo realizado y de esto se desprendieron nuevas líneas de trabajo a seguir.

Un aspecto en donde hay posibilidades de mejoras es la ampliación de las funcionalidades brindadas por la interfaz gráfica de usuario para permitir un uso más directo e intuitivo de la aplicación. Esto se puede lograr permitiendo posicionar y editar elementos bidimensionales y tridimensionales directamente mediante acciones de ratón, sin dejar de soportar la opción de posicionamiento mediante el ingreso de coordenadas ya que esto permite un control más preciso.

VMT permite únicamente utilizar *quads* para representar figuras bidimensionales. Esto provoca que de necesitarse modelar figuras con más vértices para cubrir una superficie compleja sea necesario construirla utilizando más de un *quad*. Una mejora posible sería permitir definir figuras bidimensionales con más de cuatro vértices para simplificar la definición en estos casos. Esto tiene la dificultad de que el mapeo de texturas no es directo para figuras con más de cuatro vértices por lo que la aplicación debería proveer un mecanismo para definir las coordenadas de mapeo para cada vértice. De todas formas, esta funcionalidad se puede cubrir mediante el uso de objetos *3DS* que ya incorporen las coordenadas de mapeo en los vértices.

Para el manejo de objetos tridimensionales, *VMT* depende de herramientas externas que permitan el manejo de archivos *3DS*. En particular, las tareas que serían deseables incluir en *VMT* son las de edición de vértices y caras de mallas tridimensionales y asignación de materiales a grupos de caras.

Actualmente *VMT* no dispone de una interfaz visual que asista a la sincronización de la ejecución de los efectos visuales con el audio que acompaña el espectáculo. Una forma en que esto podría realizarse es desplegando una visualización de la onda de audio, para tener una idea de en qué instantes se provocan cambios importantes en el sonido y asignar efectos directamente en los instantes visualizados.

Por último, debido al amplio estudio en métodos de obtención automática de geometría tridimensional y a la más reciente liberación del kit de desarrollo de *Kinect* sería de interés integrar la captura de geometría con *VMT* para permitir un uso más dinámico de la aplicación, completando de esta forma el ciclo de trabajo para la construcción de un espectáculo de *video mapping*.

Apéndice A

Aportes

A continuación se presentan los resultados de una serie de entrevistas realizadas a *VJs* e ingenieros contactados a partir de la investigación del estado del arte del *video mapping*. Se les consultó sobre sus trabajos realizados y las técnicas y herramientas que utilizaron para su producción. Fueron introducidos brevemente al proyecto para recibir comentarios y conocer sus expectativas en cuanto a herramientas que puedan asistirlos en la resolución o simplificación de problemas con los que se encuentran actualmente al momento de realizar espectáculos de *video mapping*.

A.1. Marcelo Vidal (VJ Chindogu)

Marcelo Vidal[Chi] es un *VJ* local referente del *video mapping* en nuestro medio y quien ha desarrollado varios de los espectáculos más importantes y de gran repercusión. Nos comenta que para sus espectáculos trabaja principalmente con modelos en dos dimensiones, por lo que la fase inicial de su trabajo se basa en transformar la fachada o superficie a mapear en un modelo bidimensional. Para este propósito su forma de trabajo consiste en fijar la posición y orientación del proyector en el lugar desde donde se realizará la proyección, para luego capturar las formas y construir el modelo bidimensional.

Una vez modelada la escena, se realiza todo el trabajo creativo y de diseño visual del espectáculo, para lo que utiliza *AfterEffects*, *Photoshop* y *3D Studio*. Luego, al momento de la proyección, posiciona nuevamente el proyector en el lugar original y realiza los ajustes previos. Entre estos ajustes se encuentra la calibración y posicionamiento del proyector en donde fue resaltada la dificultad existente para mover los proyectores que se utilizan en espectáculos de gran porte, los que pueden llegar a pesar hasta 200 kilogramos, lo que dificulta realizar movimientos milimétricos para ajustes finos. Por esto último considera importante la disposición de herramientas de ajuste o calibración del modelo de software sobre la superficie. Destacó que utiliza exactamente el mismo proyector tanto para la obtención de la geometría como para la posterior ejecución del espectáculo, lo cual según su experiencia reduce el trabajo posterior de calibración.

En cuanto al mapeo, fue destacado como muy importante poder tener la posibilidad de modificar en tiempo real el espectáculo. En referencia al tratamiento de las deformaciones que se producen al proyectar sobre una superficie irregular, comentó que maneja diferentes estrategias. Puede tanto utilizar la deformación dada por la superficie junto con deformaciones de la imagen o video a proyectar para lograr un efecto en conjunto, como también intentar modificar el video o imagen para minimizar los efectos dados por la superficie irregular. Comenta que esta última opción es la menos interesante en el ámbito artístico de los *VJs*. Las herramientas que utiliza para la realización de sus espectáculos son *QuickTime*, *Module8*, *VDMX* y *Resolume PC*.

A.2. Martin Borini (*VJ Ailaviu*)

La principal inquietud que nos transmitió Martín Borini[Ail] fue en relación al manejo de información tridimensional durante el proceso de producción de un espectáculo y así poder conocer las deformaciones que se darán en la superficie donde se proyectará. También se muestra preocupado por el problema de la correspondencia entre el modelo tridimensional y la superficie, lo que lo lleva a estar interesado en la posibilidad de la calibración de un modelo de ese estilo. En sus trabajos realizados sobre fachadas ha tomado fotos a nivel desde el mismo lugar donde se coloca el proyector. Según su experiencia esto produce mejores resultados que trabajar sobre planos o medidas tomadas por él. Otro punto en el cual expresó interés fue en casos en los que se utilizan más de un proyector, en donde surge el problema de las costuras o uniones de las imágenes proyectadas por los diferentes proyectores.

A.3. Viktor Vicsek

Viktor Vicsek[Vik] es un *VJ* de nivel internacional y se ha establecido contacto con él luego de observar los espectáculos de más relevancia y popularidad referenciados en sitios relacionados al tema. Comenta que ha implementado sus propios módulos de software para asistirle en la producción. También ha realizado una mejora a la técnica de calibración automática de proyectores de Johnny Lee[1] utilizando cámaras y visión por computadora en lugar de sensores de luz, lo que le ha resultado particularmente útil al trabajar con grandes superficies o fachadas de edificios. También ha implementado su propia línea de tiempo de videos utilizando *Adobe Air*. Para la realización de los mapeos y distorsiones de imágenes y videos utiliza la herramienta de programación visual *VVVV*.

Su proceso de producción consta en tomar fotografías y obtener los planos arquitectónicos de la escena en cuestión y la creación del modelo, según sus necesidades, con herramientas de modelado como *3D Studio*. Posteriormente realiza el mapeo de texturas de imagen o video aplicando efectos que utilizan *shaders* sobre el modelo mediante la utilización de *VVVV*.

[1] http://johnnylee.net/projects/thesis

Apéndice B

Relevamiento de aplicaciones de *video mapping*

B.1. *Modul8*

Modul8 [Mod] es una herramienta profesional para realizar espectáculos de proyección de video en vivo diseñada por *VJs* y se encuentra disponible exclusivamente para plataforma *Mac OS X* mediante licenciamiento propietario.

Figura B.1: Pantalla principal de *Modul8*

La interfaz de usuario está pensada para los espectáculos en vivo, para lo que cuenta con cuatro paneles con un fin específico: un panel principal donde crear y editar las composiciones de video, sonido y efectos; un panel multimedia para administrar los archivos de medios (videos, imágenes, pistas de audio, etc.); un panel de previsualización donde se puede ir observando la ejecución de una parte de la composición del espectáculo; y

por último el panel de salida, donde se observa la composición final que será proyectada. Todas las funcionalidades provistas pueden ser asignadas a eventos *MIDI*, por lo que es posible utilizar cualquier dispositivo que los genere para controlar enteramente la aplicación.

Modul8 maneja hasta siete salidas simultáneas de proyección y una más para la interfaz de usuario. Para hacer uso de esta funcionalidad es necesario contar con una tarjeta de video que dé soporte a la salida múltiple. Esto activa el modo de salida avanzado donde se puede escoger entre varias alternativas para la selección de la composición o porción de la misma a ser emitida en cada una de las salidas disponibles.

Modul8 maneja el concepto de capa, con un máximo de diez, las cuales pueden contener sus propios medios, efectos y configuraciones. Si bien no hay un manejo tridimensional de la escena, la aplicación provee algunas transformaciones a aplicar a los medios para ajustarlos a la representación bidimensional de los objetos de la escena. La arquitectura de la herramienta permite su extensión mediante la programación de módulos en el lenguaje *Python*[1]. En relación al rendimiento, se hace un uso intensivo de la *GPU* para todo lo que es dibujado, composición y transformaciones de gráficos.

B.2. *VDMX*

VDMX[VDM] es una aplicación para el procesamiento multimedia en tiempo real, disponible exclusivamente para plataforma *Mac OS X* mediante licenciamiento propietario.

Figura B.2: Ejemplo de consola personalizable avanzada realizada en VDMX

Su arquitectura modular la hace altamente personalizable mediante los denominados *inspectores* que proveen la interfaz para controlar distintos aspectos de la aplicación. El *inspector* del grupo de trabajo expone todo lo que está sucediendo en *VDMX* a bajo nivel, por ejemplo, los archivos multimedia disponibles, las entradas de video configuradas,

[1] www.python.org

las capas, los *plugins* existentes, etc. Luego, el *inspector* de interfaz de usuario permite ver y editar las propiedades de cada control que se selecciona en la interfaz. Adicionalmente, permite crear interfaces propias a partir de controles básicos como deslizadores, botones, ventanas emergentes, etc, lo que permite a los *VJs* tener toda la información que consideren relevante a su disposición en el momento de la representación en vivo. La aplicación permite asociar eventos de entrada *MIDI*, *OSC* o de teclado a cualquier control de la interfaz de usuario, así como también generar eventos *MIDI* u *OSC* para ser consumidos por otras aplicaciones.

Las capas contienen un único origen multimedia, el que puede ser de tipo video, imagen o composición de *QuartzComposer*[2], y adicionalmente se define una cadena de efectos a ser aplicados. La ejecución de la secuencia de efectos puede ser configurada mediante cualquiera de los mecanismos de eventos mencionados anteriormente así como también por tiempo.

No hay un máximo establecido en el número de capas que se pueden crear en *VDMX*, y es posible agruparlas para darle un manejo diferenciado o aplicar efectos a todas a la vez. Se cuenta con tres modos de salida de video: ventana, pantalla completa y avanzado. En el modo avanzado se permite definir dispositivos de salida, sin un límite predeterminado, y asociarlos a las capas o dispositivos de entrada que se desee.

B.3. *VVVV*

VVVV[VVV] es una herramienta de programación de propósito general que provee un entorno híbrido de programación gráfica y textual. Es de uso gratuito para propósitos no comerciales y se encuentra disponible únicamente para plataforma *Windows*.

[2] quartzcomposer.com

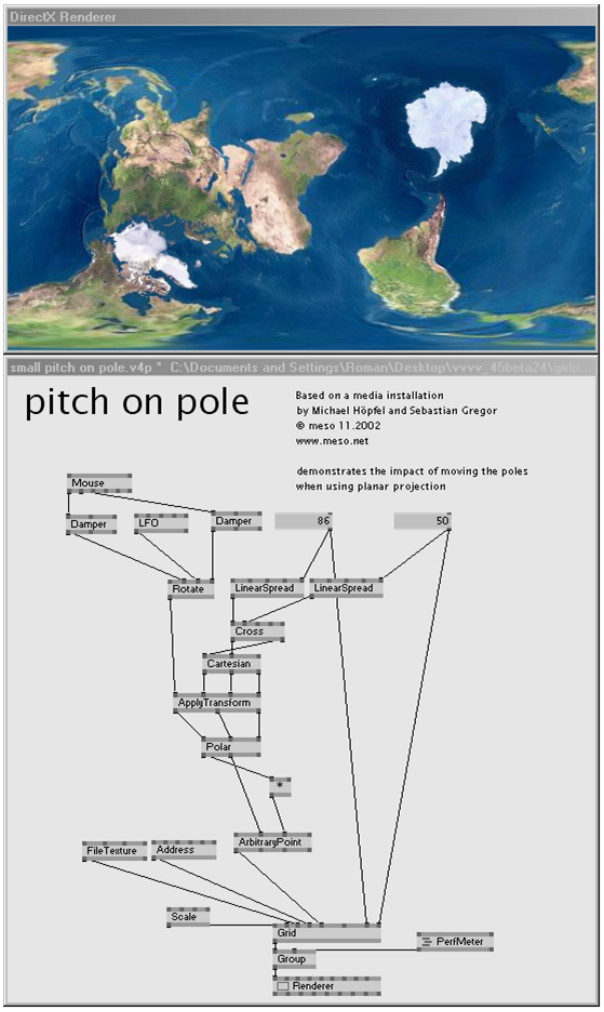

Figura B.3: Entorno de programación de *VVVV* y salida de video correspondiente

Para la construcción de un programa se utilizan nodos que representan operaciones o funciones individuales que pueden tomar una entrada, procesarla, y entregar datos a la salida. Las conexiones entre nodos se llaman *links* y es como se representa el flujo de información entre nodos. Existen varios tipos de nodos para diferentes propósitos. A modo de ejemplo, el nodo utilizado para desplegar la salida visual del programa es del tipo *Renderer*, para el agregado de cuadrantes se utiliza el nodo *Quad*, para el manejo de transformaciones, el nodo *Transform* y para redimensionar a escala, el nodo *Scale*. También hay nodos para cargar objetos tridimensionales, manejar texturas, vectores, temporizadores, etc.

A diferencia de otros lenguajes de programación, que cuentan con diferentes modos para construir y ejecutar los programas, *VVVV* todo el tiempo trabaja en el mismo modo, el modo de ejecución. En él, constantemente se están ejecutando cálculos y dibujando gráficos mientras se está generando o editando un programa.

Para dar soporte multiproyector, *VVVV* implementa una arquitectura cliente-servidor, la que permite contar con cualquier cantidad de clientes manejados desde una instancia

de *VVVV* servidor. Toda la configuración y ejecución de un espectáculo distribuido se implementa en un módulo llamado *boygrouping*[3]. *VVVV* provee soporte para varios protocolos de entrada y salida como *TCP*, *UDP*, *MIDI* y *OSC* entre otros, y gracias a aportes de la comunidad de programadores *VVVV* es posible también interactuar con dispositivos como *Wii*, *PSP* y *Kinect*. Existen nodos que proveen funcionalidad avanzada como ser soporte para animación y líneas de tiempo, detección y seguimiento de objetos en tiempo real utilizando técnicas como *ARToolkit* y *OpenCV*, emisión de flujos de video, reproducción y mezcla de archivos de audio, simulación de movimiento, colisiones y efectos físicos en general. El motor gráfico tridimensional de *VVVV* está basado en *Direct3D*, lo que permite hacer un uso intensivo de las placas gráficas modernas para el dibujado de gráficos de alto rendimiento. *VVVV* ofrece la posibilidad de crear nodos personalizados a través de una interfaz *COM* para la implementación de *plugins*, lo que ofrece alternativas en cuanto a la implementación, pudiéndose utilizar *C#*, *Delphi* o *C++*.

B.4. *VPT - Video Projection Tool*

VPT[VPT] es una herramienta para la realización de proyecciones en tiempo real. Es una aplicación de uso y distribución gratuita que se encuentra disponible para plataformas *Windows* y *Mac OS X*.

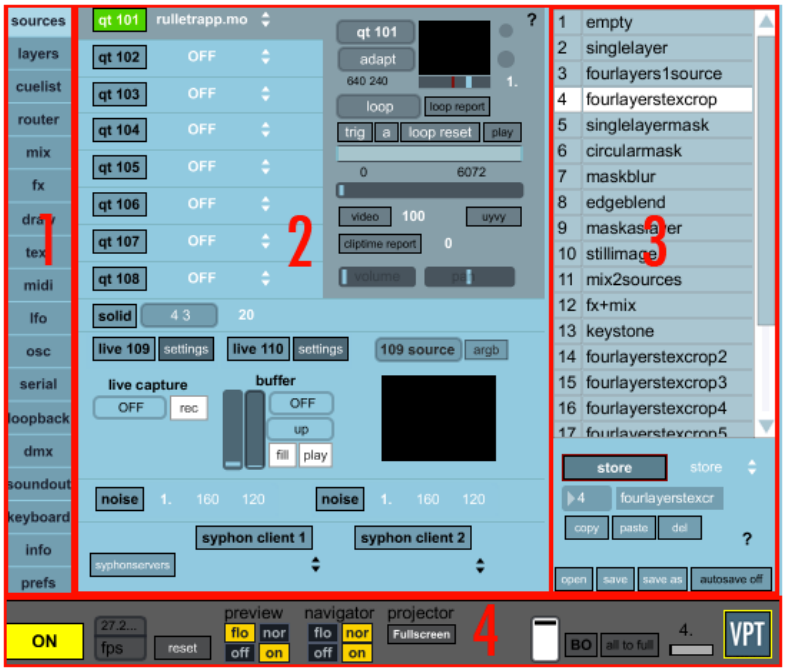

Figura B.4: Pantalla principal de *VPT*

La interfaz gráfica es amplia y permite el acceso a variadas funcionalidades. Dispone de una barra de herramientas con cada una de las secciones, las que se muestran en

[3]http://vvvv.org/documentation/boygrouping-basics

detalle en el panel principal de la aplicación. Siempre disponible a la derecha, están las configuraciones predefinidas. La barra sobre el borde inferior de la pantalla es desde donde se controlan las salidas de video, la previsualización y las opciones de navegación. En cuanto a los posibles orígenes de medios, se puede contar con hasta ocho videos de *Quicktime*, dos entradas en vivo, las que pueden ser asociadas a cualquier dispositivo de video soportado en el sistema, y una entrada de dibujado en la que es posible crear máscaras personalizadas para objetos sobre los cuales se proyectará.

Hay una cantidad máxima de 32 capas que se pueden utilizar simultáneamente en un espectáculo. Cada una está superpuesta sobre la siguiente, y puede ser escalada, posicionada, rotada, distorsionada y enmascarada de manera independiente. Maneja el concepto de capa activa para la edición y dibujado en la ventana de salida y cuenta con una secuencia de eventos a partir de efectos predefinidos almacenados y configurados previamente en el sistema. Para la definición de efectos se cuenta con tipos predefinidos y una nomenclatura para agregarlos a la lista, por ejemplo “F 15 16 5.00” representa una transición del efecto 15 al 16 con duración 5 segundos, “C 5” ejecuta el efecto número cinco, “L 3” genera un bucle y vuelve al tercer lugar de la lista de eventos, “D 2” genera una demora de dos segundos y continúa con el siguiente evento.

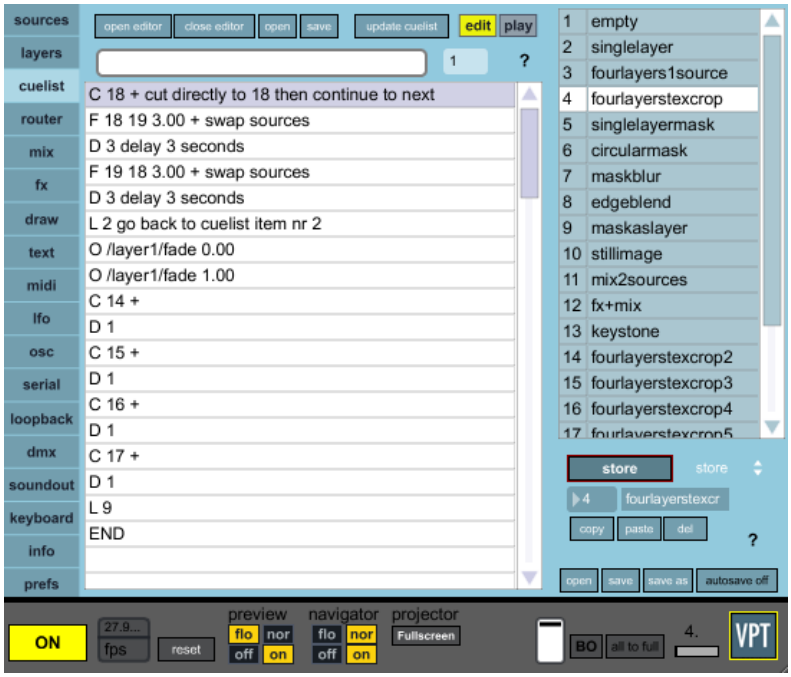

Figura B.5: Efectos y secuencia de eventos de un espectáculo realizado en VPT

Dentro del modo de edición es posible agregar o eliminar elementos a la lista sin que éstos sean reflejados en la salida de video, mientras que en modo de reproducción se está recorriendo la lista de efectos constantemente, generando la salida correspondiente. Dependiendo de la tarjeta de sonido con la que se cuente, es posible direccionar las entradas de video en hasta ocho canales de salida de audio. El módulo de mezclado permite combinar dos orígenes multimedia distintos utilizando diferentes modos. También es posible controlar la aplicación o enviar directamente indicaciones a las capas, orígenes

de medios, mezclador, etc, utilizando *OSC*, *MIDI* o vía puerto serial con dispositivos como *arduino*[4]. Para simular proyecciones tridimensionales se cuenta con la posibilidad de subdividir las texturas con grillas y aplicar así distorsiones a distintos sectores de la misma[5].

[4]http://www.arduino.cc

[5]Ejemplo de uso de mallas http://hcgilje.wordpress.com/2011/05/26/vpt-5-5-preview

Apéndice C

Método de triangulación

Mediante este método se determinan las coordenadas (x, y, z) de un punto utilizando la posición bidimensional dada por las perspectivas de dos proyecciones de las que se conocen los centros de perspectiva y planos de proyección[Pre].

Escena con dos dimensiones:

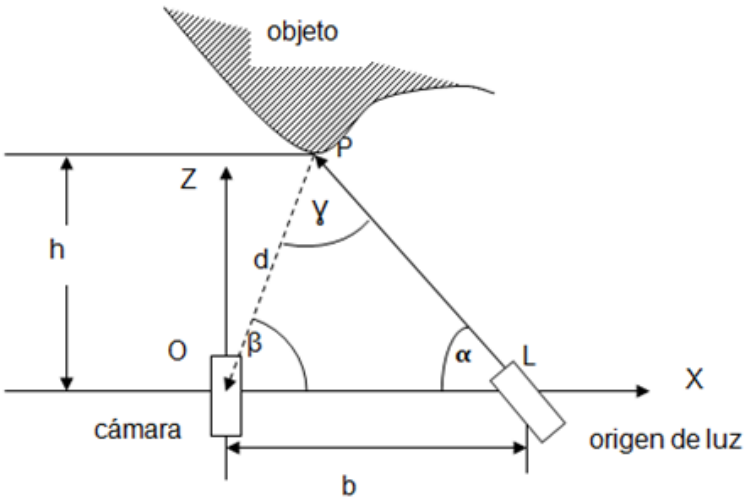

Figura C.1: Propiedades trigonométricas para escena bidimensional.

Este método tiene como objetivo calcular la distancia d de la cámara al punto P a partir de los ángulos α, β y la distancia b entre el proyector y la cámara. El ángulo α y la distancia b son dados por la configuración de la escena. El ángulo β está dado por la geometría de la proyección.

$$\left.\begin{array}{l}\frac{d}{\sin(\alpha)} = \frac{b}{\sin(\gamma)} \\ \gamma = \pi - (\alpha + \beta) \\ \sin(\pi - \gamma) = \sin(\gamma)\end{array}\right\} \frac{d}{\sin(\alpha)} = \frac{b}{\sin(\pi - \gamma)} = \frac{b}{\sin(\alpha + \beta)} \Rightarrow d = b.\frac{\sin(\alpha)}{\sin(\alpha + \beta)}$$

Las coordenadas cartesianas quedan determinadas por:

$$X_0 = d.\cos(\beta)$$

$$Z_0 = d.\sin(\beta) = h$$

Escena con tres dimensiones:

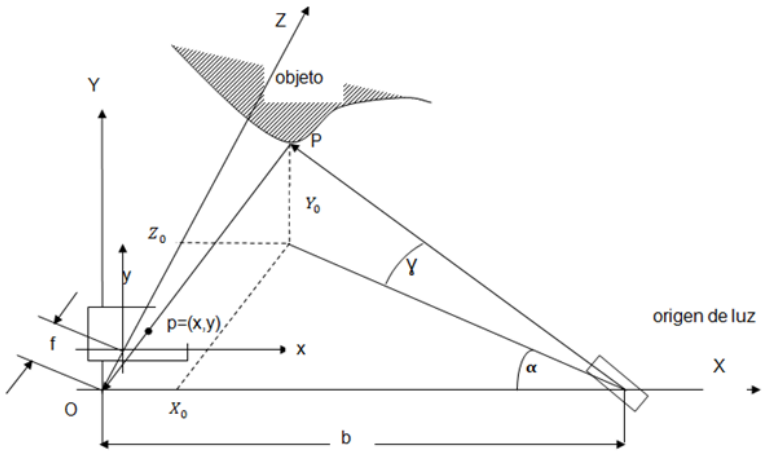

Figura C.2: Propiedades trigonométricas para escena tridimensional.

Se asume $Z = f$ siendo f el plano en el cual se proyecta el punto $P(X_0, Y_0, Z_0)$, obteniendo como resultado de la proyección el punto $p(x, y)$. El centro óptico del proyector está situado en el eje X. Se considera realizada una pre-calibración en la cual se define:

$$P = (x, y), \quad \frac{X_0}{x} = \frac{Z_0}{f} = \frac{Y_0}{y} = k \rightarrow (k * x, k * y, k * f)$$

Por trigonometría:

$$\tan(\alpha) = \frac{Z_0}{(b - X_0)} \Rightarrow Z_0 = \tan(\alpha)(b - X_0)$$

$$k * f = \tan(\alpha)(b - k * x)$$

$$k(f + x * \tan(\alpha)) = b * \tan(\alpha)$$

$$k = \frac{b * \tan(\alpha)}{f + x * \tan(\alpha)}$$

$$X_0 = \frac{x * b * \tan(\alpha)}{f + x * \tan(\alpha)}, \quad Y_0 = \frac{y * b * \tan(\alpha)}{f + x * \tan(\alpha)}, \quad Z_0 = \frac{f * b * \tan(\alpha)}{f + x * \tan(\alpha)}$$

Apéndice D

Modelo de cámara

D.1. Modelo *pinhole*

Una cámara, por medio de una proyección central, establece una correspondencia entre puntos del espacio con puntos bidimensionales en su plano imagen. En particular se estudia el modelo de cámara *pinhole* [Ric00].

Se considera el centro de proyección C, también centro óptico de la cámara, como el origen de un sistema de coordenadas euclideano, y $Z = f$ el plano de la imagen o plano focal.

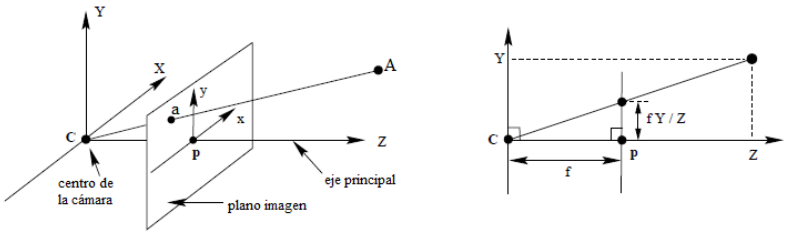

Figura D.1: Geometría de cámara *pinhole*. C es centro de la cámara y p el punto principal

Un punto en el espacio $A = (A_X, A_Y, A_Z)^T$ se corresponde con el punto a en el plano imagen, dado por la intersección del rayo que pasa por el centro de cámara y el punto A con el plano imagen. El punto A se corresponde con $(\frac{fA_X}{A_Z}, \frac{fA_Y}{A_Z}, f)^T$ en el plano imagen considerando que el centro de coordenadas del plano coincide con el punto principal P.

Considerando la representación de los puntos como vectores homogéneos se expresa la proyección central como una correspondencia lineal entre las coordenadas homogéneas.

$$\begin{pmatrix} A_X \\ A_Y \\ A_Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fA_X \\ fA_Y \\ A_Z \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} A_X \\ A_Y \\ A_Z \\ 1 \end{pmatrix}$$

Dado $A_h = (A_X, A_Y, A_Z, 1)^T$ y la proyección a sobre el plano imagen, la correspondencia utilizando el método pinhole es: $a = PA_h$

Siendo

$$P = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

D.2. Geometría epipolar

La geometría epipolar[Ric00] es la geometría proyectiva intrínseca entre dos puntos de vista. Es dada por la intersección de los planos imagen de cada cámara con el plano epipolar. Este plano es el definido por el punto a representar X y los dos centros de las cámaras C y C'.

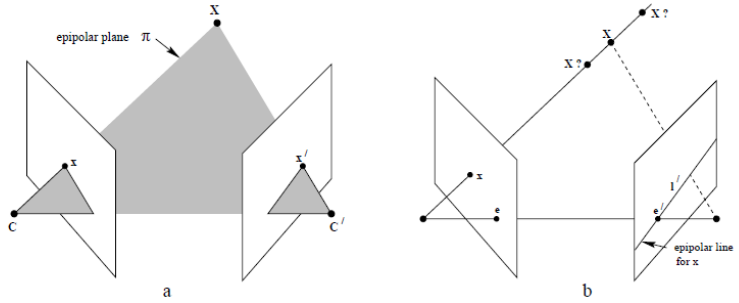

Figura D.2: C y C' centros de las cámaras.

La matriz fundamental F es la representación algebraica de la geometría epipolar. Dado un par de imágenes, como se muestra en la figura para cada punto x en una imagen existe una línea epipolar correspondiente l' en la otra imagen. Cualquier punto x' en la segunda imagen que corresponde al punto x, debe pertenecer a la línea epipolar l'. La línea epipolar l' es la proyección en la segunda imagen del rayo que parte del punto x y llega al centro C de la primera cámara. Se establece por tanto un mapeo de un punto en una imagen con la línea epipolar en la otra imagen.

$$x \rightarrow l'$$

El mapeo de puntos a líneas es representado por F la matriz fundamental.

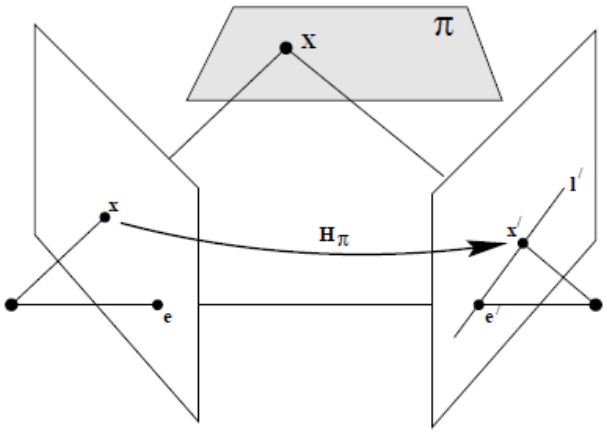

Figura D.3: Matriz fundamental F.

Dado x en una imagen se corresponde con el punto x' en la segunda imagen vía una trasferencia establecida por la intersección del plano π con los planos imagen de cada cámara. La línea epipolar que contiene x' se obtiene uniendo x' con el epipolo e', esto es:

$$x' = H_\pi * x$$

siendo H_π una homografía 2D que establece la correspondencia entre cada x_i de la primer imagen a x'_i en la segunda imagen.

$$l' = e' * x' = e' * H_\pi * x = F * x$$

$F = e' * H_\pi$es la matriz fundamental

Glosario

VJ Un *Video Jockey* es un artista quien genera sesiones visuales mezclando en directo pistas de video con música u otro tipo de elemento.

Motor tridimensional Término que hace referencia a una serie de rutinas de programación que permiten el diseño, creación y representación de aplicaciones tridimensionales en tiempo real. El motor tridimensional acepta comandos de una aplicación y construye imágenes y texto que son dirigidos al hardware gráfico.

Nube de puntos Colección de vértices que definen la superficie externa de un objeto.

Malla tridimensional Conjunto de vértices, aristas y caras en un sistema de coordenadas tridimensional que representan la superficie externa de un objeto.

Visión por computadora Rama de la inteligencia artificial que mediante el procesamiento de imágenes por computadora extrae información de las mismas para la toma de decisiones.

Mapa de bits Estructura de datos que representa una rejilla rectangular de píxeles que se puede visualizar en un monitor u otro dispositivo de representación.

Algoritmo de Kruskal Algoritmo de la teoría de grafos para encontrar un árbol de expansión mínimo en un grafo conexo y ponderado.

Eje óptico Eje perpendicular al plano imagen que pasa por el centro óptico de la cámara.

GPU Procesador suplementario a la unidad de procesamiento central cuya función específica es el procesamiento de gráficos.

Canal alfa Porción del píxel en el cual se guarda información de su opacidad.

Píxel Acrómino del inglés *picture element* o elemento de imagen, es la unidad básica de color de una imagen digital.

UDP Protocolo usado ampliamente para consultas de petición y respuesta de una sola ocasión, del tipo cliente-servidor y en aplicaciones en las que la entrega pronta es más importante que la entrega confiable, como las transmisiones de voz y video.

Bibliografía

[3DC] 3D Coat. http://3d-coat.com/.

[3DSa] 3DS File Format. www.the-labs.com/Blender/3dsspec.html by Jeff Lewis.

[3DSb] 3D Studio Max. http://usa.autodesk.com/3ds-max/.

[Ail] Web de Martin Borini. http://vimeo.com/ailaviu.

[Ble] Blender. http://www.blender.org.

[BLO] Automatic Modelling & Video Mapping - Blogspot. http://automodellingandvideomapping.blogspot.com/.

[BYO] Web del proyecto BYO3D. http://mccammon.ucsd.edu/ bgrant/bio3d.

[CGA] CGAL: Computational Geometry Algorithms Library. http://www.cgal.org.

[Chi] Web de Marcelo Vidal. http://vimeo.com/vjchindogu.

[DAV] DAVID Structured Light Scanner. http://www.david-laserscanner.com.

[Fes] Festival de luces en Lyon Francia. http://www.lumieres.lyon.fr/lumieres/sections/fr.

[Geo06] Georg Klein and David Murray. ARTS: Full-3D Edge Tracking with a Particle Filter. , 2006.

[GLS] Lenguaje de shaders para OpenGL. www.lighthouse3d.com/opengl/glsl.

[J.] J. BATLLE; E. MOUADDIB; J. SALVI. RECENT PROGRESS IN CODED STRUCTURED LIGHT AS A TECHNIQUE TO SOLVE THE CORRESPONDENCE PROBLEM: A SURVEY.

[Jam96] James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, Richard L. Phillips. *Introducción a la Graficación por Computador*. Addisson-Wesley Iberoamericana, S.A., 1996.

[Joa] Joaquim Salvi;Jordi Pagès;Joan Batlle. Pattern codification strategies in structured light systems.

[Joh93] John P. Snyder. *ARTS:Flattening the Earth: Two Thousand Years of Map Projections*. University of Chicago - ISBN 0-226-76747-7, 1993.

[Kyl] Web de Kyle McDonald. http://www.kylemcdonald.net.

[Mar] Mark Pauly; Markus Gross; Leif P. Kobbelt. *Efficient Simplification of Point-Sampled Surfaces.*

[MATa] Web de MATLAB. http://www.mathworks.com/products/matlab.

[Matb] Adaptadores gráficos. http://www.matrox.com/graphics/en/products/gxm.

[May] Autodesk Maya. http://usa.autodesk.com/maya/.

[Mes] Software de código abierto para manipulación de mallas tridimensionales en varios formatos. http://meshlab.sourceforge.net.

[Mic] Michael Kazhdan, Matthew Bolitho and Hugues Hoppe. Poisson Surface Reconstruction.

[Mod] Web de Module8. http://www.modul8.ch.

[Mud] Mudbox. http://usa.autodesk.com/adsk/servlet/pc/index?id=13565063&siteID=123112.

[Opea] Web de OpenCV. http://opencv.org.

[opeb] Openframeworks. http://www.openframeworks.cc.

[Pau] Paul S. Heckbert;Michael Garland. Survey of Polygonal Surface Simplification Algorithms.

[Pie03] Pierre Alliez and Craig Gotsman. ARTS:Recent Advances in Compression of 3D Meshes. , 2003.

[Pre] Structured Lighting / Computer Vision (EEE6503) Fall 2009, Yonsei Univ. http://web.yonsei.ac.kr/hgjung/Ho%20Gi%20Jung%20Homepage/Lectures/2009%20Fall%20Computer%20Vision/Handouts/7%20Structured%20Lighting.pdf.

[Qt-] Qt - framework multiplataforma. http://qt.nokia.com/products/.

[QT] Qt framework. http://qt.nokia.com.

[QTM] Qt Model-View-Controller. http://doc.qt.nokia.com/4.7-snapshot/model-view-programming.html.

[Ric00] Richard Hartley y Andrew Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press 2000, 2003, 2000.

[Ós] Óscar Belmonte;Inmaculada Remolar;José Ribelles;Miguel Chover;Marcos Fernández. Efficient Implementation of Multiresolution Triangle Strips.

[SB] Luca Iocchi Shahram Bahadori. *A Stereo Vision System for 3D Reconstruction and Semi-Automatic Surveillance of Museum Areas.* Dipartimento di Informatica e Sistemistica Universida di Roma La Sapienza, ¡bahadori,iocchi¿@dis.uniroma1.it.

[Scu] Sculptris. http://www.pixologic.com/sculptris/.

[Sha] Shahram Izadi;David Kim;Otmar Hilliges;David Molyneaux;Richard Newcombe;Pushmeet Kohli;Jamie Shotton;Steve Hodges;Dustin Freeman;Andrew Davison;Andrew Fitzgibbon. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. http://events.ccc.de/congress/2011/Fahrplan/attachments/1969_kinectfusion-uist-comp.pdf.

[Son09] Song Zhang. Recent progresses on real-time 3D shape measurement using digital fringe projection techniques. Opt LaserEng(2009). , 2009.

[SOUa] Fuentes del proyecto - GitHub. https://github.com/jefa/video-mapping-tool.

[SOUb] Fuentes del proyecto - Google code. http://code.google.com/p/automodeling-and-video-mapping.

[Ume] Umesh R. Dhond and J.K.Aggarwal 1989. Structure from Stereo - A Review.

[Vcg] Biblioteca de computación grafica portable escrita en C++ para manipulación, procesamiento y despliegue con OpenGL de mallas triangulares. Liberada bajo licencia GPL por parte del Visual Computing Lab (VCGLab) del Institute of the Italian National Research Council, http://www.isti.cnr.it. http://vcg.sourceforge.net/tiki-index.php.

[VDM] Web de VDMX. http://vidvox.net.

[Vik] Web de Viktor Vicsek. http://vimeo.com/viktorvicsek.

[VPT] Web de Video Projection Tool. http://hcgilje.wordpress.com/VPT.

[VVV] Web de VVVV. http://vvvv.org.

[Ya-] Ya-xiang Yuan. A review of trust region algorithms for optimization. .

[Zbr] Zbrush. http://www.pixologic.com/zbrush/.

Índice de figuras