Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Arquitectura para aplicaciones *Rich-Client* con soporte a sistemas de inmersión

Tesis que presenta

Sonia Abigail Martínez Salas

para obtener el Grado de

Maestro en Ciencias en Computación

Director de Tesis

Dr. Amilcar Meneses Viveros

México, D.F. **Noviembre 2014**

Resumen

Las representaciones visuales ayudan a tener una mejor interpretación de datos y modelos. Estudios demuestran que la interpretación adecuada de la información es mayor cuando se presenta al usuario de forma inmersa. No todos los dispositivos tienen las capacidades (*hardware*) para desplegar contenido de esta manera o no cuentan con los controladores (*software*) necesarios, impidiendo que las aplicaciones ocupen adecuadamente este recurso de los dispositivos. Las aplicaciones *Rich-Client* permiten, entre otras ventajas, acceder a las características de los dispositivos donde se ejecutan, brindando la facilidad de adaptar la aplicación dependiendo de éstas.

El objetivo de esta tesis es desarrollar una arquitectura de *software* que permita a una aplicación *Rich-Client* dar soporte a sistemas de inmersión para desplegar contenido 3D, dependiendo de las características del dispositivo donde ésta se ejecute. Se propone una aplicación que despliega contenido 3D en dispositivos con soporte a inmersión. A ésta se le adapta el manejo de contenido HTML para que cuente con las ventajas de una aplicación *Rich-Client*. Con estas ventajas se pueden acceder a las características del dispositivo (dimensiones, contraste, resolución, capacidad de despliegue de contenido en forma inmersa y consumo energético, por mencionar algunas), para que con base en esto pueda saberse el tipo de contenido adecuado a desplegar.

Abstract

Visual representations help to achieve a better interpretation of data and models. Studies show that an adequate interpretation of this data is greater when information is presented to the user in an immersive manner. Not all the devices have the capability (hardware) to display content in this way or they do not have the drivers (software) required, making that applications do not use adequately the device's resources. *Rich-Client* applications, among other advantages, allow access to the host device's features, making posible to adapt information according to them.

The objective of this tesis is to develop a software architecture that allows a *Rich-Client* application to provide support to immersive systems, in order to display 3D content, according to the host device's features. An application that display 3D content in devices with immersive capabilities has been developed. This application is adapted to handle HTML to make it have *Rich-Client* applications advantages. With this advantages the application is able to access the device's features (dimensions, contrast, resolution, immersive capability and energetic consume, to mention a few), in order to know the kind of content suitable to be displayed.

Agradecimientos

A mi madre, que desde chica me inculco el hábito del estudio y apoyó que estudiara una maestría a pesar de que ella cree que las personas que estudian un posgrado lo hacen por que no desean trabajar

A mis sinodales Sonia Mendoza Chapa y Gabriela Sánchez Morales, por su apoyo y aportaciones a mi trabajo

A los profesores Naun y Alicia que me enseñaron y ayudaron con la redacción

A mis amigos con los que compartí esta etapa y a los que ya tiene mucho más tiempo en mi vida

A todos mis profesores y personal del departamento de Computación del CINVESTAV

A mi director de tesis

Al CONACyT y al CINVESTAV-IPN

Contenido

1 Introducción **1**

1.1 Motivación . . . 1
1.2 Planteamiento del problema . . . 2
1.2.1 Problema general . . . 2
1.2.2 Problema específico . . . 3
1.3 Propuesta de solución . . . 4
1.3.1 Alcances de la solución . . . 5
1.3.2 Detalles técnicos de la solución . . . 5
1.4 Objetivos . . . 5
1.4.1 Objetivo general . . . 5
1.4.2 Objetivos específicos . . . 6
1.5 Justificación . . . 6
1.6 Metodología . . . 7
1.7 Organización del documento . . . 7

2 Visualización Científica **9**

2.1 Importancia . . . 9
2.2 Aplicaciones . . . 10
2.3 Tipos de representaciones de datos . . . 10
2.4 Dispositivos de salida . . . 11
2.5 Adaptabilidad de la información . . . 13

3 Sistemas de inmersión **17**

3.1 Importancia de los sistemas de inmersión e interfaces 3D . . . 17
3.2 Desarrollo de aplicaciones 3D . . . 18
3.2.1 Bibliotecas y *frameworks* . . . 18
3.2.2 Formatos de modelos 3D . . . 20
3.3 Aplicaciones 3D en la Web . . . 21
3.4 Formas de despliegue de contenido 3D . . . 22
3.4.1 3D estereoscópico . . . 22
3.4.2 Formas pasivas . . . 23
3.4.3 Formas activas . . . 26

4 Aplicaciones *Rich-Client* **29**
4.1 Soluciones al desarrollo de aplicaciones multi-pataforma . . . 29
4.1.1 Tipos de clientes Web . . . 29
4.1.2 Ventajas y desventajas de las aplicaciones *Rich-client* . . . 30
4.2 Herramientas de desarrollo para aplicaciones *Rich-Client* . . . 31
4.2.1 Lenguajes . . . 31
4.2.2 Alternativas a JavaScript . . . 32
4.3 Adaptabilidad de interfaces móviles . . . 33

5 Desarrollo **35**
5.1 Análisis . . . 35
5.2 Diseño . . . 36
5.3 Implementación . . . 40
5.3.1 Módulo de obtención de características . . . 40
5.3.2 Módulo de transformación de datos . . . 43
5.3.3 Módulo de despliegue de información . . . 43

6 Pruebas y resultados **47**
6.1 Pruebas . . . 47
6.1.1 Recursos de *hardware* . . . 47
6.1.2 Recursos de *software* . . . 48
6.2 Análisis del experimento . . . 48
6.3 Resultados . . . 49

7 Conclusiones y trabajo a futuro **53**
7.1 Conclusiones . . . 53
7.2 Trabajo a futuro . . . 54

A Pruebas médicas **57**

B Imágenes de la aplicación **59**

Capítulo 1

Introducción

1.1 Motivación

La visualización de información en computadoras es una tarea necesaria en el ámbito profesional y personal, ya que actualmente gran parte de la información, se acompaña por representaciones visuales para su mejor interpretación [Wright, 2007]. Como ejemplo de ésto, gráficas estadísticas son ocupadas para mostrar posibles comportamientos a futuro en las ventas de un producto [Arteaga et al., 2009]; modelos en tercera dimensión sirven para simular y reemplazar prototipos, lo que reduce tiempos y costos de elaboración [Ahrens et al., 2001]; incluso imágenes representativas del tipo de clima acompañan el reporte meteorológico para hacerlo más atractivo al público [Wright, 2007].

En áreas de aplicación como los sistemas de aprendizaje, factores como la documentación, el conocimiento y el manejo de la información son importantes para obtener el énfasis adecuado en la información [Blas and Fernández, 2008]. En estas herramientas utilizadas para la educación (sistemas de aprendizaje), debe representarse de manera óptima la información, para brindar una mejor experiencia al usuario e incluso adaptarla a las necesidades pedagógicas [Machado and Tao, 2007].

Otra de las áreas donde la representación adecuada de la información es importante es la simulación de eventos físicos, donde la cantidad de datos simulados es grande y vital para una adecuada interpretación de ellos [Kapferer and Riser, 2008]. Estas simulaciones suelen ser visualizadas en grandes dispositivos de despliegue, como paredes de video y domos [Ni et al., 2006]. Sin embargo al no tener un dispositivo que cuente con estas características, sería útil poder representar la información de otra manera que aún sea entendible y significativa, adecuándonos a las características de los dispositivos disponibles.

Estudios muestran que la inmersión es benéfica en aplicaciones donde la noción del espacio y la profundidad en el sistema son útiles [Raja et al., 2004], [Schuchardt and Bowman, 2007], [Bowman and McMahan, 2007], [Alahmari et al., 2014]; esto hace que la inmersión no sólo sea más atractiva sino también más entendible para el usuario. Las representaciones 3D en tiempo real se han vuelto una característica común en muchas aplicaciones; sin embargo, las imágenes resultantes están aún en 2D [Nocent et al., 2012].

Por otra parte, la diversidad de dispositivos hace evidente la necesidad de adaptación y

acceso ubicuo a las interfaces de usuario [Moralez, 2009]. Esto da pie a que una aplicación no sólo debe ser consciente del tipo de dispositivo en el cual se ejecuta, sino que también debe tomar "decisiones" sobre cómo representar la información o qué recursos del dispositivo huésped se deben ocupar.

Una interfaz que puede ser representada en "n" dispositivos disponibles y aún así seguir manteniendo el mismo modelo conceptual tiene beneficios como son: menor esfuerzo cognitivo para el usuario al ocupar diferentes interfaces, complejidad adecuada y reducción del mantenimiento para sistemas e interfaces y el uso adecuado de portales Web no diseñados específicamente para dispositivos móviles [de Oliverira and da Rocha, 2006].

1.2 Planteamiento del problema

En esta sección se tratarán los puntos principales se deben de tomar en cuenta para tener una solución adecuada a el problema a resolver. Se abarcan las características de cualquier sistema que brinde soporte a la adaptabilidad de información y las propias de sistemas de inmersión.

1.2.1 Problema general

Actualmente, las aplicaciones no cuentan con algún tipo de soporte para que los datos puedan visualizarse e interpretarse de manera eficiente en diversos tipos de dispositivos. Además se carece de una arquitectura general para que se utilicen de forma adecuada las ventajas de dichos dispositivos. Una aplicación que pretenda solucionar este problema debe tomar en consideración los siguientes puntos:

- Existen diferentes formatos y tipos de datos.
- Existen diferentes fuentes para obtener datos.
- Existe una amplia diversidad de dispositivos de despliegue, cada uno con sus características particulares.
- Las capacidades de los dispositivos deben utilizarse al máximo, por ejemplo la disponibilidad para el despliegue de contenido inmersivo, con el objetivo de que los datos sean entendibles al espectador.
- En caso de que el formato dado a desplegar no sea el adecuado para el dispositivo huésped, debe realizarse una transformación.
- No todos los dispositivos pueden realizar una transformación de datos. En el caso de dispositivos con recursos limitados (móviles), podría requerirse que las transformaciones se ejecuten en un servidor externo.

En la figura 1 se muestra una posible solución a este problema en donde un modelo de datos puede estar o no en el dispositivo. En caso de no estar en el dispositivo se obtendrá de un

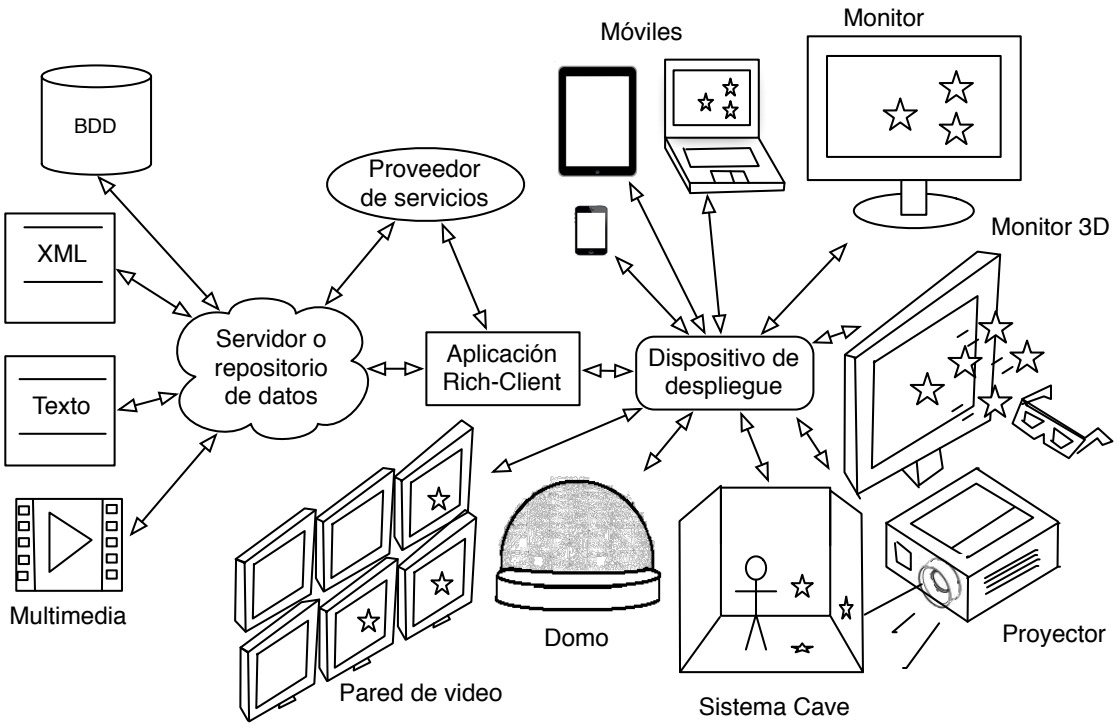


Figura 1: Diagrama general de la solución

servidor externo. El modelo de datos pude provenir de diferentes formatos, por ejemplo, de una base de datos, de archivos descriptivos o de archivos de texto plano.

Una vez ejecutada la aplicación, ésta debe determinar en qué dispositivo se encuentra, de acuerdo a sus características (dimensiones, resolución, contraste, consumo energético y capacidad de despliegue en 3D) para "decidir" cuál es la mejor presentación para los datos en el dispositivo huésped.

En caso de que los datos no se encuentren en la presentación de datos que mejor se adapta al dispositivo o incluso que el modelo de datos provenga de diferentes formatos, deberá realizarse una transformación. En particular, cuando se utilizan dispositivos móviles, no siempre es viable este recurso, debido a que su procesamiento impactaría al consumo energético del dispositivo [Kumar et al., 2012]. En estos casos también debe evaluarse si los cálculos se ejecutarán en el dispositivo o en un servidor externo [Kumar et al., 2012]. Además, en este escenario, otro factor a tomar en cuenta es la conectividad a Internet. Una vez que se tienen los modelos de datos en la representación adecuada, se procederá al despliegue de datos y su interacción con el usuario.

1.2.2 Problema específico

Es evidente que el problema de la adaptabilidad de datos y su despliegue en cualquier dispositivo es muy amplio, por lo cual ahora se hablará específicamente de la adaptabilidad de información

en dispositivos con capacidad de despliegue de contenido en 3D.

Al momento de intentar desplegar información que pueda ser representada en 3D, la aplicación debe acceder a las características del dispositivo para conocer si este tiene capacidad de desplegar dicha información. Algunos ejemplos de dispositivos que podrían ser capaces de desplegar modelos 3D son monitores 3D, proyectores 3D y pantallas volumétricas. Las características para saber si el dispositivo puede desplegar contenido 3D son:

- Tipo de dispositivo en el que se quiere desplegar información.
- Capacidad del *hardware* del dispositivo para despliegue en 3D (con o sin capacidad de desplegar contenido en 3D).
- Tipo de tecnología para despliegue 3D (activo o pasivo), en caso de contar con alguna.
- Dimensiones del dispositivo.
- Disponibilidad de controladores necesarios (*software*) para el funcionamiento de estas utilidades.

En caso de no contar con todo lo necesario para desplegar contenido en 3D, una aplicación debe presentar la información en 2 dimensiones (despliegue común), a pesar que los modelos representados estén en tres.

Ademas, los modelos de datos 3D pueden provenir de diferentes formatos, como son: archivos VRML, *obj*, en formatos X3D o archivos descriptivos. Así mismo pueden estar ubicados en diversos medios de almacenamiento como el dispositivo de despliegue, una BDD (base de datos) remota o incluso en la *nube*.

1.3 Propuesta de solución

Se propone el diseño de una arquitectura que servirá como base para solucionar este problema y la cual contempla los siguientes aspectos:

- La obtención de modelos de datos podrá ser desde un servidor remoto o desde el dispositivo donde se requiere desplegar los datos.
- Se desplegará datos de modelos 3D.
- La aplicación decidirá si el dispositivo puede o no desplegar contenido de manera inmersa (3D). Para hacer esto posible se hará uso de una aplicación *Rich-Client*, la cual será consciente de las características del dispositivo donde se está ejecutando.
- La aplicación *Rich-Client* decidirá si puede desplegarse contenido de forma inmersa (sondeo de *hardware* y *software*).
- Una vez tomada la decisión, se activarán los elementos necesarios (*hardware* y *software*) y finalmente los modelos se desplegarán en una representación adecuada.

1.3.1 Alcances de la solución

Como primer acercamiento a esta solución y dado la complejidad del problema a tratar, la infraestructura propuesta sólo cubrirá los siguientes aspectos:

- Los modelos de datos sólo podrán ser de tipo *obj* y *sdkmesh*.
- Las pruebas se harán únicamente con Sistema Operativo Windows 7 y 8. Esto es debido a que este Sistema Operativo es el que tiene más soporte a sistemas de inmersión.
- Los dispositivos de despliegue podrán ser monitores con y sin capacidad de despliegue de contenido 3D.

1.3.2 Detalles técnicos de la solución

Para el desarrollo de esta tesis se requieren los siguientes elementos:

- Una computadora con Sistema Operativo Windows a partir de Windows 7. En este caso se ocupará Windows 8, ya que es el Sistema Operativo con más soporte para dispositivos de inmersión en la actualidad.
- Un procesador Core 2 Duo o mejor.
- Un monitor 3D. En este caso se cuenta con los monitores V3D231 ViewSonic y BenQ XL2720T.
- Una tarjeta gráfica con soporte a contenido 3D. En este caso se cuentan con las tarjetas GTX530 TI, Quadro FX 540 y Quadro K2000.
- Un par de gafas activas y sincronizador.
- Algún dispositivo sin capacidad de despliegue de contenido en 3D para realizar pruebas (monitor o computadora portátil).

1.4 Objetivos

1.4.1 Objetivo general

Generar una arquitectura que permita a una aplicación *Rich-Client* dar soporte para que los sistemas de inmersión puedan mostrar contenido 3D, dependiendo de las características (capacidad de despliegue de contenido 3D, dimensiones, resolución, contraste y consumo energético) del dispositivo en el que se despliegue.

1.4.2 Objetivos específicos

- Analizar la adaptabilidad de información dependiendo del tipo de datos y dispositivos de despliegue, con el fin de saber qué tipo de dato es el mejor entendible al desplegarse en cada dispositivo y, en caso de que el formato y el dispositivo no sean compatibles, saber si existe algún tipo de transformación para obtener el tipo de dato deseado.

- Analizar los modelos de datos 3D existentes con el fin de conocer los más ocupados, su uso y determinar cúales son los más apropiados a usar por el momento en nuestra arquitectura.

- Diseñar una arquitectura para dar soporte a sistemas de inmersión.

- Desarrollar una aplicación capaz de manipular un modelo de datos 3D con base en la arquitectura propuesta para así poder probar su eficacia.

- Proveer de adaptabilidad a la aplicación 3D detectando las características del dispositivo en el que se ejecuta.

- Generar prototipos de sistemas de inmersión, lo cual permitirá visualizar modelos de datos en dispositivos con capacidad de inmersión y sin ella.

- Probar la adaptabilidad de los modelos de datos dependiendo del dispositivo de despliegue y así comprobar que las soluciones dadas sean entendibles para el usuario.

1.5 Justificación

La visualización de información en computadoras es necesaria para lograr una interpretación adecuada de los datos. Se puede lograr un mejor entendimiento, interpretación e impacto en los usuarios presentando la información de forma inmersa. Actualmente los sistemas que pueden desplegar contenido de manera inmersa requieren que el dispositivo donde se ejecutan tenga características específicas para su funcionamiento adecuado. El simple hecho de no contar con alguna de ellas, impide su función.

La manera en que se debe desplegar la información depende, en gran parte, de las características del dispositivo en el cual se ejecuta una aplicación. Una manera adecuada para determinar las características de un dispositivo es ocupando aplicaciones *Rich-Client*, ya que éstas sólo requieren de un navegador Web para ejecutarse, funcionan en cualquier sistema operativo, son independientes a la plataforma del dispositivo sede y funcionan incluso de manera desconectada.

Esta tesis servirá como base para que un navegador Web común pueda determinar, entre cuatro tipos de modos de despliegue (3D activo, 3D pasivo, 2D e imágenes estáticas), cuál es la manera que más se adecúa a sus características, brindando así soporte a aplicaciones que pretendan desplegar información de manera inmersa. Se espera con esto, dada la literatura [Raja et al., 2004], [Nocent et al., 2012], obtener una mejor interpretación de los datos por parte de los usuarios.

1.6 Metodología

- Investigar los modelos de datos existentes y sus representaciones.
- Seleccionar los modelos de datos adecuados para el despliegue de información en cada dispositivo e indicar las transformaciones que se requerirían para una mejor representación de datos, en caso de existir alguna.
- Desarrollar una aplicación que nos permita manejar las características de inmersión del dispositivo (en dispositivos que sí cuenten con las características necesarias para desplegar este tipo de contenido).
- Desarrollar una aplicación *Rich-Client* que nos permita determinar las características del dispositivo (dimensiones, resolución, contraste y consumo energético) en el que se ejecute.
- Incorporar la aplicación *Rich-Client* con la de sistemas de inmersión.
- Realizar pruebas de experiencia de usuario en dispositivos con y sin capacidad de despliegue inversivo.

1.7 Organización del documento

Como el objetivo de este trabajo de tesis es desarrollar una arquitectura que permita el uso de sistemas de inmersión por medio de aplicaciones *Rich-Client*, con el fin de dar soporte a estos sistemas y tener información que pueda adaptar su representación, de acuerdo al dispositivo en el que se muestre, en el capítulo 2 se hablará de los avances en visualización científica en cuanto al despliegue de información en dispositivos de gran tamaño y sus aplicaciones. El capítulo 3 contiene los tipos de datos y sus representaciones, transformaciones entre ellos y características a tomar en cuenta para escoger los que serían abarcados en nuestra infraestructura. En el capítulo 4 se informa sobre las aplicaciones *Rich-Client*, su historia, avances, ventajas sobre otros tipos de clientes, herramientas para desarrollarlas y ejemplos donde podemos encontrarlas cotidianamente. En el capítulo 5 se trata todo el análisis para el desarrollo de la infraestructura y la aplicación, también se mencionarán las herramientas ocupadas para su desarrollo y diagramas descriptivos de su funcionamiento y composición. En el capítulo 6 se muestran las pruebas realizadas y su análisis. Finalmente, el capítulo 7 recopila conclusiones, discusiones y trabajo a futuro.

Capítulo 2

Visualización Científica

En esta tesis, se pretende brindar al usuario una mejor visualización de modelos en 3D de manera inmersa, por lo que deben de conocerse los diferentes tipos de datos y sus representaciones e incluso transformaciones. El área de visualización científica estudia los diferentes tipos de datos y la mejor manera de representarlos para su mejor entendimiento e impacto en el usuario. En este capítulo, se habla de la importancia de la visualización científica, los avances que se han tenido, sobretodo para el despliegue de grandes cantidades de datos y otros puntos de interés para el desarrollo de esta tesis. Se presenta un breviario de los diferentes tipos de datos y sus representaciones, así como los tipos de dispositivos de despliegue y sus características.

2.1 Importancia

La representación por medio de imágenes es casi tan antigua como el hombre mismo. Específicamente en las ciencias, las representaciones visuales sirven para explicar observaciones, hacer predicciones y entender teorías [Wright, 2007].

Actualmente estamos bombardeados de imágenes, haciendo que cualquier tipo de información parezca incompleta sin ellas [Wright, 2007], pero el mayor interés en la visualización por computadora fue el resultado de varios factores. El primero de estos factores es el incremento del poder de cómputo de los dispositivos, ahora capaces de procesar grandes conjuntos de datos. Como segundo factor se tiene la necesidad de tener representaciones que faciliten la interpretación adecuada de los datos. El último factor son los avances en herramientas gráficas por computadora que funcionan incluso generando gráficos en tiempo real. El objetivo de estas visualizaciones es hacer cada vez representaciones más reales y llamativas, sean o no sacadas del mundo real y que los datos o resultados sean fácilmente apreciados e interpretados.

Se han realizado estudios [Arteaga et al., 2009] en los cuales se demuestra que el humano entiende mejor con representaciones visuales y que el impacto que éstas tienen es más amplio que otras representaciones. Siendo aún más específicos, se ha descubierto que representaciones tridimensionales tienen mejor impacto en usuarios [Raja et al., 2004] y su interpretación adecuada es más fácil para tareas donde el sentido de profundidad es importante (como diseño de prototipos), en comparación de imágenes bi-dimensionales.

2.2 Aplicaciones

Entre las principales áreas que ocupan la visualización científica están Medicina, Biología, Astronomía, Geología, por mencionar algunos. Entre los trabajos afines a esta tesis se encuentran, estudios de tipos de datos, transformaciones existentes y formas de representarlos en diferentes dispositivos de salida (en su mayoría grandes dispositivos).

En el artículo de [Chi, 2000] se resumen los tipos de datos que ocupan algunos sistemas de visualización de datos más ocupados y sus transformaciones.

En el artículo de [Kelleher and Wagener, 2011] se dan guías para una mejor visualización y entendimiento de datos para propósitos específicos (documentos científicos), haciendo énfasis en la importancia que tiene encontrar una representación adecuada de estos.

En trabajos como [Ni et al., 2006] se estudia cómo se puede ocupar la tecnología de grandes dispositivos de despliegue para visualizar datos de mejor manera.

En trabajos de [Ahrens et al., 2001] y [Kapferer and Riser, 2008] se discuten técnicas para manejar la visualización de problemas, donde se requiere procesar y desplegar grandes cantidades de datos, como es el caso simulaciones astronómicas.

En trabajos como [Healey and Enns, 2012] se investiga cómo la percepción visual, con un foco en específico, tienen relevancia directa con la visualización y analítica visual, en especial memoria y atención visual.

En trabajos como [Yi et al., 2007] se proponen siete nuevas categorías de interacción con los datos en aplicaciones visuales, lo que va más allá de interacciones a bajo nivel, con el fin de entender mejor la ciencia de la interacción entre los tipos de datos.

Todo esto da una base para saber qué tipo de despliegue se puede representar, en algunos dispositivos, tipos de datos y cómo transformarlos, así como errores comunes a evitar en nuestro trabajo.

2.3 Tipos de representaciones de datos

Los datos se pueden clasificar por algún rubro, determinado por nosotros mismos o por una taxonomía ya existente.

Una forma de clasificación de un rubro determinado son, por ejemplo, las *variables dependientes e independientes*, con la cual no se tienen los datos en sí, sino un modelo que ayuda a reconstruirlos. Éste es el caso de las fórmulas con las cuales podemos construir gráficas, tablas u otras representaciones de datos.

En la clasificación por *dominio de datos* se pueden representar las características influyentes en el valor de los datos, como por ejemplo distancia y tiempo. Cada uno de estos elementos le da una dimensión más a los datos, los cuales pueden variar en dimensiónes desde 1D, 2D, 3D o hasta nD. En este tipo de clasificación se encuentran datos representados por coordenadas polares, coordenadas ordinales o coordenadas esféricas.

En la clasificación dada por *tipo escalar o vector*, se distinguen los datos dependiendo de si éstos son representados por dos escalares separados o si se tienen en una sola variable dependiente, la cual es a su vez, un vector en dos dimensiones.

Las representaciones anteriores son algunos ejemplos de clasificaciones de datos pero dependiendo del problema en que se encuentren se pueden tener diferentes clasificaciones. Actualmente existe una taxonomía que ya clasifica datos dependiendo de la representación que se les puede dar. En esta taxonomía existen diferentes tipos de datos, los cuales podemos agrupar en tres grandes grupos: nominales, agregados y ordinales [Wright, 2007].

Los datos *nominales* están representados por un conjunto de puntos, en donde el orden no importa para su representación. Ejemplo de representaciones de estos datos son las gráficas de barras y de pastel.

Los datos *agregados* son representaciones donde un orden determinado importa para dar significado correcto. Estos datos pueden ser representados en histogramas o zonas de dibujo, por mencionar algunos.

Los datos *ordinales* son representaciones en las que el orden es importante y la imagen es generalmente discontinua. En este caso pueden unirse los valores si la imagen es continua, pero se debe tener extremo cuidado que ésto no altere la forma de visualización.

La tabla 1, nos muestra algunos ejemplos de tipos de datos y sus representaciones, dependiendo de sus dimensiones y estructura. La primera columna contiene el tipo de dato que se desea representar. La segunda columna muestra las dimensiones en que pueden ser representados los tipos de datos. Las columnas tres, cuatro, cinco y seis son las formas en que los tipos de datos pueden representarse. Como ejemplo, los datos del tipo agregado pueden tener representaciones en una o dos dimensiones. Si se desea ocupar una dimensión, se debe emplear un histograma para representar un escalar único. En caso de requerir representar escalares múltiples se pueden ocupar histogramas superposicionados o apilados.

Para cambiar los datos de una forma en la que no se encuentran a otra representación, se debe realizar una transformación. Algunas transformaciones entre datos son muy fáciles o directas, como pasar de un histograma 3D a uno 2D, tomando rebanadas de éste o tener diferentes formas de representar el mismo modelo de datos como en histogramas o gráficas de barras. El proceso que representaría realizar transformaciones, manteniendo el significado de su información, es complicado. La adecuada interpretación de los datos depende a veces de su presentación, e incluso se puede tener el caso en que las representaciones no sólo son equivalentes, si no que se pueden complementar entre si. Un ejemplo de esto son datos presentados tanto en tablas como en gráficas.

Aún en estas representaciones equivalentes, no todas se entienden por igual o sirven para el mismo tipo de propósitos.

2.4 Dispositivos de salida

Los dispositivos donde podemos requerir desplegar nuestras aplicaciones actualmente son muy amplios y variados en cuanto a características. Estos dispositivos van desde teléfonos inteligentes y tabletas con capacidades limitadas y pequeñas pantallas hasta sistemas tipo CAVE y grandes paredes de video.

Las características a tomar en cuenta para despliegue de información son: dimensiones de pantalla, resolución, contraste, capacidad de despliegue de contenido en 3D y consumo energético.

Tipo de dato	Dim	Escalar único	Escalares múltiples	Vector	Escalar y vector
Nominal	1D	• Barras • Pay	• Barras agrupadas • Barras apiladas	• Diagramas de dispersión	
	2D	• Barras 2D			
Agregado	1D	• Histograma	• Histograma superposicionado • Histograma apilado		
	2D	• Histograma 2D • Zona de dibujo acotada			
Ordinal	1D	• Líneas	• Líneas superpuestas • Líneas apiladas	• Trayectoria	• Trayectoria coloreada • Polígono ***Swept***
	2D	• Despliegue de imágenes • Gráfica de contorno • Vista de superficie	• Gráfico **Height-Field**	• Flechas sólidas en plano • Flechas de línea • Línea de tiempo • Textura de flujo • *StreamLine*	• Líneas coloreadas • Flechas coloreadas
	3D	• IsoSuperficies • Render de volumen	• IsoSuperficie coloreada	• Flechas con volúmen • Superficie de tiempo (*StreamRibbon* o tuvo/superficie)	• Líneas coloreadas • Flechas coloreadas

Tabla 1: Tipos de datos y sus representaciones

Otro factor a tomar en cuenta, y quizá el más importante, es el tipo de dispositivo en el cual se requieren desplegar los datos, ya que las características antes mencionadas no aplican para todos. Se podría requerir desplegar información en dispositivos como son los *Google Glasses*, en los cuales, una representación de los datos de manera auditiva, sería mejor interpretada que solo imágenes visuales.

Solo dependiendo de la cantidad de información y las características del dispositivo, también pueden cambiar las formas en las que interactuamos con la aplicación o con la información misma. Para grandes volúmenes de información es importante saber cómo será la navegación entre ellos, para lo cual la característica más importante del dispositivo podría ser el tamaño. En el caso de video, imágenes u otro material multimedia, debe tenerse en cuenta que las acciones principales podrían ser reproducción y nivel de zoom/volumen: para este caso, el tamaño es menos importante que contar con hardware adecuado para la reproducción de estos materiales.

En la tabla 2 se resumen las características de diferentes dispositivos de despliegue de información tomando en cuenta el tipo de dispositivo, su funcionamiento, resolución, contraste y consumo energético. El tipo de dispositivo ayuda a determinar el tipo de información que se puede tener, así como el tamaño del modo de despliegue. La resolución hace referencia a cuánto detalle puede distinguirse en un dispositivo. No se debe desplegar información importante en dispositivos con baja resolución, ya que ciertos detalles podrían no ser visualizados correctamente. El contraste es la diferencia relativa en la intensidad entre dos puntos de una imagen. Al igual que con baja resolución, si se cuenta con bajo contraste, algún tipo de información pudiera no ser apreciada en el despliegue. El consumo energético ayuda a evaluar los costos por energía en caso de tener un dispositivo en funcionamiento por tiempo prolongado. También se sabe de dispositivos con recursos limitados que podrían no ser capaces de efectuar cierto tipo de cálculos.

Aunque no se encuentran mencionados en la tabla, otros tipos de dispositivos de salida importantes son impresoras y *plotters*, a partir de los cuales podemos obtener información en dos o tres dimensiones como resultado.

2.5 Adaptabilidad de la información

La adaptabilidad de la información es la habilidad de ajustar diferentes tipos de información a distintos tipos de dispositivos, con el objetivo de obtener una interpretación adecuada de los datos [Salas, 2014].

Para lograr este objetivo, las características del dispositivo huésped deben ser evaluadas, para determinar la representación que más se adecúe a éste, por ejemplo: en caso de querer reproducir un audio, debemos primero saber si existe un reproductor de sonido y si los controladores de la tarjeta de sonido se encuentran actualizados, de otro modo la información puede no llegar al usuario de la manera esperada. En caso de querer desplegar grandes cantidades de información en un dispositivo con pantalla pequeña, como pueden ser teléfonos o relojes inteligentes, la información debe ser reestructurada para no perder visibilidad ante el usuario e incluso evaluar si el mostrarla en otra forma, como podría ser audio, es mejor que ocupar la representación usual.

Actualmente, una forma de desarrollar aplicaciones multi-plataforma es el enfoque *Rich-*

Dispositivos de Despliegue				
Dispositivo	Funcionamiento	Resolución	Contraste	Consumo Energético
TouchScreen	Detecta la presencia de dedos o stylus en la pantalla	Hasta 1.920 x 1.080	Depende de la tecnología	Limitada
Proyectores	Luz proyectada a micro espejos	Hasta 4096x 2160	hasta 3000:01	200 V a 240 V
Tiled display	Conjuntos de pantallas LCD agrupados en un arreglo 2D	Depende de las pantallas que lo conforman	Depende de las pantallas que lo conforman	Depende de las pantallas que lo conforman y la energía que cada una de éstas consume
Pantallas de bitmaps				
CRT	Un flujo de electrones es emitido de una pistola, enfocado y dirigido por campos magnéticos. Es usado en televisores y monitores de computadoras	Entre 640 x 480 y 1600 x 1200, pero resoluciones más altas son posibles	Arriba de 15,000:1	Alta
Plasma	Pequeñas celdas que contienen iones de gas eléctricamente cargadas	Desde 853x480 hasta1,920x1,080 o mayores	hasta 50000:01	Varia dependiendo del brillo, pero usualmente es mayor a las LCD
LCD	Ocupa las propiedades de modulación de la luz de los cristales líquidos	1366 x 768 en promedio	1000:01 en contraste normal y hasta 7000:en contraste dinámico	Baja, entre el 10% y 50% de los CRT
LED	Por medio de diodos emisores de luz	Usualmente entre 1,280x768 y 1920x1080	10000:01	Varía dependiendo del brillo, pero usualmente es menor que LCD, a menos que grandes áreas claras sean desplegadas
Sistemas de inmersión				
Pantallas volumétricas	Apilan pixeles en 3D para simular profundidad	Depende de la tecnología	Depende de la tecnología	Depende de la tecnología
Proyectores 3D	Misma tecnología que pantallas 3D	1920x1080	Desde10000:1 hasta 50000:1	100V - 240V
Display 3D	Vistas separadas para diferentes ojos	1920x1080	Depende de la tecnología	Depende de la tecnología
Cascos de inmersión	Pequeñas pantallas para cada ojo que crean efecto en 3D	720 pixeles	Depende de la tecnología	Depende de la tecnología
CAVE	Rodea a los espectadores con un ambiente inmersivo con cuatro o más pantallas de pared	100 millones de pixeles o más	Depende de la tecnología	Depende de la tecnología
CAVE2	Combina características de paredes de video con sistemas de realidad virtual	3 veces la resolución de cave	Depende de la tecnología	Depende de la tecnología
Domo	Simples o múltiples proyectores funcionando en una superficie con forma de domo	Desde 1024x768 hasta 4096x2160	Depende de la tecnología	Depende de la tecnología
Display estereoscópico	Muestra dos conjuntos de pixeles (una para cada ojo del espectador) por cada imagen	Hasta 2872 X 2150	Depende de la tecnología	Depende de la tecnología

Tabla 2: Dispositivos de despliegue, pantallas de bitmaps y sistemas de inmersión

Client, también conocido como aplicaciones de Internet enriquecidas. Como se puede ver en la figura 1, para lograr una adaptabilidad de la información se necesita un agente que, independientemente del dispositivo, pueda determinar sus características, ayudando a conocer los recursos con los que contamos para desplegar e incluso interactuar con la información. Dado este enfoque, se pueden conocer las características del dispositivo y adecuar el tipo de salida deseado.

Capítulo 3

Sistemas de inmersión

El problema específico que se aborda en esta tesis es presentar modelos de datos 3D en sistemas de inmersión en un navegador Web, por lo cual debe saberse cómo funcionan los sistemas de inmersión, un poco de historia sobre ellos y qué avances se ha tenido al ocuparlos en diferentes dispositivos. También se habla de formatos para representar modelos de datos en 3D con el fin de seleccionar los más ocupados. Se mencionan algunas de las herramientas que se usan actualmente para tener conversiones entre estos modelos de datos para dar un panorama más amplio de los alcances que este proyecto tiene.

3.1 Importancia de los sistemas de inmersión e interfaces 3D

Se refiere a inmersión física al grado en el cual un usuario puede estar rodeado por el espacio o ambiente [Raja et al., 2004]. Algunos ejemplos de dispositivos con capacidades de inmersión son los sistemas tipo CAVE y CAVE2, monitores y proyectores 3D, domos y dispositivos estereoscópicos.

Estos sistemas son importantes en áreas como simulación [Kapferer and Riser, 2008], videojuegos y mundos virtuales [Katz et al., 2011] y visualización de datos [Ni et al., 2006] por mencionar algunas. Aplicaciones de esta índole han demostrado tener éxito en áreas como terapia de rehabilitación, simulación de operaciones, diseño y simulación de sistemas de automatización, descubrimiento de drogas, visualización científica y entretenimiento [Bowman et al., 2005].

Estudios muestran que la inmersión es benéfica en aplicaciones donde la noción del espacio en el sistema es útil para su interpretación [Raja et al., 2004], [Schuchardt and Bowman, 2007], [Bowman and McMahan, 2007], [Alahmari et al., 2014], haciéndola no sólo más atractiva, sino también más entendible al usuario.

A pesar de que las representaciones 3D en tiempo real se han vuelto una característica común en muchas aplicaciones, las imágenes resultantes son usualmente desplegadas en 2D [Nocent et al., 2012]. Éste es el caso de aplicaciones accedidas desde navegadores Web, en los cuales aún no se tiene acceso (por razones de seguridad) al hardware necesario para procesar modelos en 3D y desplegarlos [Nocent et al., 2012], [Ortiz, 2010].

Desde el lanzamiento de la película *Avatar*, surgió un gran interés de los consumidores por todo tipo de dispositivos capaces de desplegar contenido en 3D y, en el 2011, se esperaba una nueva aparición de contenido 3D que pudiera ser reproducido en aplicaciones o navegadores Web [NVIDIA, 2011], el cual no fue el esperado debido a la falta de contenido en el formato adecuado.

3.2 Desarrollo de aplicaciones 3D

Existe gran cantidad de modelos, lenguajes y bibliotecas para representar objetos y escenarios en tres dimensiones. A continuación se mencionan algunos de los más ocupados y sus características.

3.2.1 Bibliotecas y *frameworks*

OpenGL

OpenGL (*Open Graphics Library*) es una interfaz de programación al *hardware* de gráficos. Esta interfaz sirve para definir objetos y operaciones para producir aplicaciones interactivas en 3D [Shreiner et al., 2007], [Buss, 2003].

Con OpenGL deben construirse modelos basados en *geometrías primitivas* (líneas, puntos, y polígonos). GLU (*OpenGL Utility Library*) forma parte de la implementación de OpenGL y provee características para crear modelos más complejos, como son superficies cuadráticas y curvas *NURBS*.

Con esta interfaz se pueden definir y construir objetos, partiendo de formas primitivas y también como representación o descripción matemática de los objetos. Estos objetos pueden colocarse en espacios tridimensionales y seleccionar un punto de vista desde el cual se compondrá la escena. Los colores de todos los objetos pueden ser calculados y asignados.

Los programas en OpenGL que funcionan en un esquema cliente-servidor ocupan un protocolo que es siempre el mismo. Este protocolo permite que las aplicaciones funcionen sin importar que el cliente y el servidor no tengan la misma plataforma (*hardware* y *software*).

OpenGL está diseñada para operar en diferentes plataformas de *hardware*. Para lograr ésto, no se incluyen en OpenGL comandos para el manejo de ventanas. Este manejo debe realizarse por medio del controlador de ventanas del sistema que se esté manejando.

Para trabajar las ventanas independientemente de la plataforma de desarrollo, puede usarse GLUT (*OpenGL Utility Toolkit*).

DirectX

Son interfaces de desarrollo de aplicaciones para videojuegos, diseñadas por Microsoft en 1995 para dar acceso a capacidades avanzadas de gráficos 3D. Al proporcionar independencia entre el dispositivo y la abstracción de *hardware* se pretende, que el código escrito para dispositivos en Direct3D funcione sin importar la versión de Direct3D con la que cuenta el dispositivo [Engel, 2003].

3DMLW

3DMLW (*3D Markup Language for Web*) es una plataforma de código abierto para crear aplicaciones 3D y 2D para Web o escritorio, de manera rápida [Juursalu et al.,]. Ocupa *OpenGL* como motor gráfico. Esta plataforma se ejecuta en los Sistemas Operativos *Windows*, *Linux*, *BSD* y *OS X*.

Ventajas:

- En esta plataforma el lenguaje descriptivo es similar a XHTML.
- Acepta entradas XML.
- Abierto.
- Puede integrarse con otras aplicaciones.

Desventajas:

- Requiere de *plug-ins* para su funcionamiento.
- Ocupa el lenguaje *Lua* como lenguaje de *scripting*.

WebGL

WebGL (*Web Graphics Library*) es una tecnología que permite crear poderosos gráficos 3D, por medio de un navegador Web [Danchilla, 2012]. Esto se logra usando JavaScript para interactuar con la unidad de procesamiento Gráfico (GPU).

La versión actual de WebGL está basada en la versión 2.0 de OpenGL (ES), aunque su API es más reducida, ya que está diseñada para poder funcionar aún en dispositivos con capacidades limitadas. Ya que se trata de una API en JavaScript, no se requiere de algún *pug-in* extra para su funcionamiento.

WebGL funciona de forma regular en sistemas operativos Windows, a partir de Vista y Mac OS a partir de 10.5. También funciona, aunque no con tanto soporte, en Linux. Tiene soporte para los siguientes navegadores:

- Firefox: a partir de la versión 4.
- Safari: a partir de la versión 5.1 y únicamente para el sistema operativo OS X.
- Chrome: a partir de la version 9.
- Internet Explorer: el soporte no es brindado de manera nativa, pero es posible configurarlo con *plug-ins* extras.

3.2.2 Formatos de modelos 3D

VRML

Virtual Reality Modeling Language (VRML) es un lenguaje para describir simulaciones interactivas de múltiples participantes. La intención de este lenguaje es convertirse en estándar para simulaciones iterativas en la Web [Gavin Bell, 1995].

Desde sus primeras versiones, permite crear mundos virtuales con comportamiento interactivo pero limitado, que funciona por medio de *hiper-links* hacia otros mundos o documentos HTML. Esté lenguaje de modelado es independiente de plataforma, extensible y trabaja bien con conexiones de poco ancho de banda.

Teóricamente, los objetos VRLM pueden contener cualquier cosa (geometrías 3D, datos MIDI, imágenes, etc.). Las estructuras de VRML se ordenan jerárquicamente por gráfos de escenas, las cuales definen el orden de los nodos que a su vez pueden afectar a los nodos subsecuentes, por ejemplo, un nodo de rotación o de material afectará a todos sus nodos subsecuentes. Los nodos tienen nombres para identificarlos, así como otros campos para describirlos. Estos nodos también son capaces de formar grupos de nodos.

VRML ya no es ocupado pues ha sido reemplazado por X3D.

X3D

Es un lenguaje para gráficos vectoriales y puede emplear sintaxis similar a XML y a VRML. X3D amplía VRML con extensiones que dan la posibilidad de emplear XML, para modelar escenas completas en tiempo real. X3D permite ocupar objetos definidos por el usuario, usar herramientas de animación (temporizadores y operadores), así como animaciones y morfología humanoide [Brutzman and Daly, 2007].

X3DOM

X3DOM (X-Freedom) es un *framework* de código abierto para gráficos 3D en la *Web*. Está diseñado para seguir las especificaciones de HTML5, W3C y Web3D [Brutzman and Daly, 2007].

X3DOM permite tener escenas de X3D en un DOM de HTML. Estos elementos pueden ser manipulados por medio de JavaScript y CSS3 de manera habitual.

Para poder hacer uso de la aceleración por hardware ocupa WebGL, lo que permite poder ejecutarlo sin necesidad de *plug-ins* adicionales. Los navegadores en los cuales se indica que X3DOM tendrá un adecuado funcionamiento son:

- Internet Explorer: desde la versión 11. En las versiones 9 o 10 se necesita instalar *plug-ins* adicionales.
- Google Chrome: desde la version 9.x.
- Firefox: nativo en versiones de escritorio y móviles.
- Safari: (sólo para OS X) desde la versión 10.6. En la versión para Windows deben cambiarse algunas configuraciones.

- Mobile IOS: aún no es soportado de manera nativa, pero es posible avilitarlo por medio de 3rd.
- Sony Ericsson Mobiles: creados para soportar WebGL y HTML estándar en navegadores Web Android.

Formato *obj*

Es un formato de archivo que define geometrías, desarrollado en un principio por *Wavefront Technologies*. Es un formato abierto y ha sido adoptado por muchas aplicaciones de gráficos en 3D, hasta ser considerado un formato universal. Al sólo representar geometrías 3D, dadas las posiciones de sus vértices, se considera un formato muy fácil de entender [Tamayo, 2013].

Las líneas que empiezan con "v" denotan vértices, las líneas con "vt" denotan texturas, "vn" para vectores normales y "f" lista de vértices que forman una cara.

Formato *sdkmesh*

Formato empleado para ser ocupado con los ejemplos del SDK de Microsoft para DirectX. Este formato no es recomendado para aplicaciones ajenas a este lenguaje o para aplicaciones robustas [Engel, 2003].

Formato *fx*

Formato creado para trabajar efectos y propiedades como programación en DirectX. En estos archivos puede especificarse texturas, sombras, luces, procesamiento, entre otros efectos 3D; en ellos se declara el flujo definido por un efecto [MWDeveloperCenter, 2010]. Un efecto puede ser de tres tipos de categorías:

- Variables: usualmente declaradas al principio del efecto.
- Funciones: implementan código o son ocupadas como funciones internas a otras.
- Técnicas: implementan sentencias de procesamiento usando uno o más efectos.

3.3 Aplicaciones 3D en la Web

En el trabajo de [Katz et al., 2011] debido a la demanda de videojuegos y redes sociales dedicadas a vidas virtuales como *SecodLife*, se desarrolló un *plugin* para navegadores Web para permitir a los usuarios de estas aplicaciones accederlas, sin necesidad de una aplicación dedicada en el cliente o realizar todo el procesamiento en el servidor. Los desarrolladores descartan estas dos formas de funcionamiento, debido a que desean dar soporte a usuarios ocasionales. En este artículo se toma en cuenta que no todos los usuarios tienen máquinas con características adecuadas para realizar el procesamiento. Si no se tienen las características para realizar el procesamiento de información, el trabajo computacional dependerá de mantener comunicación constante con el servidor (el cual realiza el procesamiento). Se sabe que entre más difícil sea

instalar una característica adicional en un dispositivo, más fácil será que sea ocupada, sobre todo si se encuentra en un dispositivo sólo momentáneamente.

Esta propuesta es desarrollada con WebGL y es un primer acercamiento a aplicaciones que muestran contenido 3D y cuenta con gran parte de las características que queremos tener en nuestra infraestructura, pero el contenido 3D se despliega únicamente en dispositivos 2D y no se adapta dependiendo de las características del dispositivo sede.

En el artículo de [Nocent et al., 2012] se discuten las implicaciones, necesidades y ventajas de que los navegadores Web puedan desplegar contenido en 3D, sin requerir de la instalación de aplicaciones extras. En este trabajo se menciona el uso de dispositivos extras, como Kinect, para hacer el manejo de los modelos de datos. No se habla de poder hacer una transformación de datos en caso de ser requerido, ni de tomar decisiones dependiendo de las características de los dispositivos donde se ejecute.

Ellos mencionan que la limitante más grande para realizar aplicaciones que cubran estos requerimientos es la seguridad, ya que los lenguajes en que se encuentran desarrolladas estas aplicaciones (principalmente JavaScript, HTML5, y CSS) no tienen acceso directo a las características de hardware como para determinar si el contenido puede o no ser desplegado.

Como solución a lo antes mencionado, ellos proponen ocupar *WebSockets* para tener comunicación con un módulo que pueda acceder a estas características.

A pesar de que las soluciones propuestas por este trabajo parecen prometedoras e interesantes, no se encontraron evidencias de que algún avance se ha desarrollado, con lo cual no puede darse por hecho que sus soluciones funcionarán.

Esfuerzos de algunas instituciones, como Nvidia, han desarrollado APIs, con el objetivo de desplegar contenido multimedia en la Web de manera inmersa [NVIDIA, 2011]. Estas herramientas evalúan si el dispositivo cuenta con las características adecuadas para el despliegue del contenido. Gracias a esta API, ahora es sencillo convertir videos de páginas como YouTube en videos con salida en 3D, si se cuenta con las características adecuadas en su dispositivo. Desafortunadamente estas herramientas aún funcionan únicamente en ciertos dispositivos y con formatos multimedia limitados.

3.4 Formas de despliegue de contenido 3D

La necesidad de incrementar el realismo con el cual los objetos 3D se representan es cada vez mayor [NVIDIA, 2010]. Además, la tecnología ha mejorado a tal grado que cada vez puede obtenerse una mejor experiencia en efectos y 3D.

Los datos para representar 3D pueden provenir de diferentes medios como escáneres, modelos creados en programas para modelado 3D y hasta cámaras diseñadas para capturar un mundo en tres dimensiones [Gateau, 2010].

3.4.1 3D estereoscópico

La técnica que está ganando popularidad, coloquialmente llamada "3D", es más correctamente descrita como "3D estereoscópico" o sólo "estéreo" [NVIDIA, 2010]. Para lograr una representación 3D se necesita crear dos copias de la imagen que nuestro cerebro percibirá, una para

el ojo izquierdo y otra para el derecho. Cada una de las imágenes anteriores tiene una perspectiva relativa con la otra, la cual a su vez es relativa a la separación inter-ocular. Gracias a la distancia inter-ocular se logra engañar al cerebro para que perciba los objetos a una distancia y profundidad relativa [Gateau and Nash, 2010].

La tabla 3 muestra la forma en que se procesa una imagen para ser presentada de forma 2D en un dispositivo. Esta muestra el centro del "*fustrum*" o "rango de visión", el cual es una pirámide trunca que parte desde el centro de visión del usuario. En estos casos debe procesarse la imagen una vez tomando en cuenta todo lo que queda dentro del rango de visión.

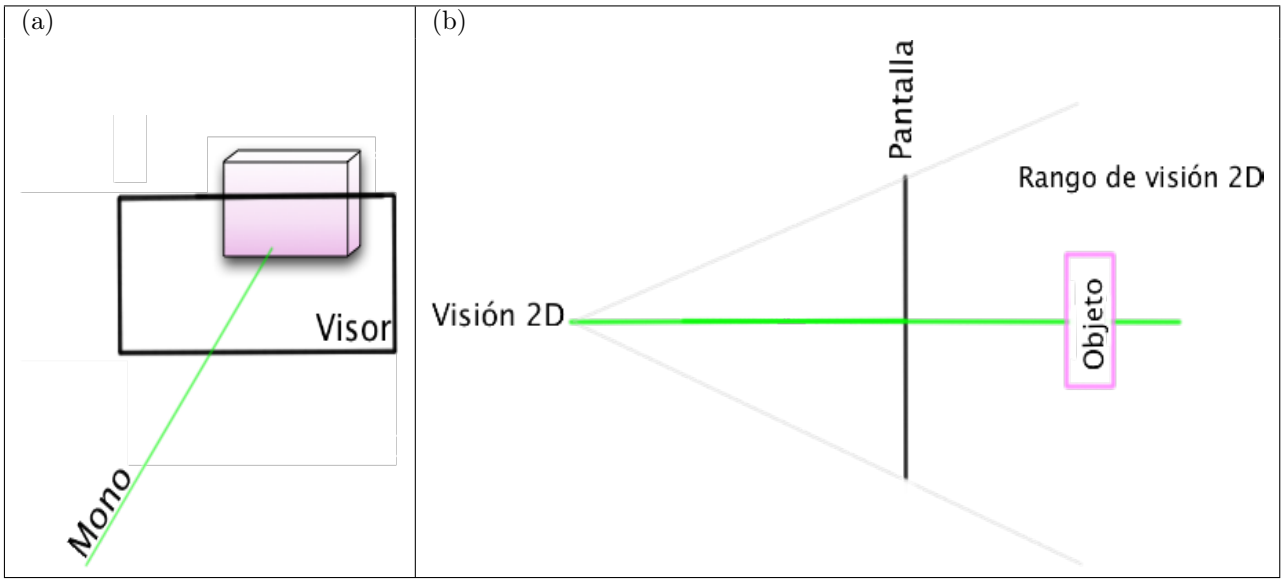


Tabla 3: Ejemplo del rango de visión a procesar en 2D

El método ocupado para lograr 3D estereoscópico es diferente dependiendo del objetivo deseado, por lo que la técnica para presentar 3D a una audiencia es diferente a la ocupada para presentarlo en una computadora personal.

3.4.2 Formas pasivas

Nos referimos con formas pasivas a las técnicas que no requieren un *hardware* específico en las gafas para poder implementarse o mostrar las formas estereoscópicas al usuario.

Anaglifo

Es la forma más sencilla de lograr un efecto estereoscópico. Para poder lograr esta técnica se toman dos puntos de visión para cada ojo (ver fig 2) y debe tenerse en cuenta que la distancia ocular puede variar de persona a persona. Se procesa la imagen desde cada uno de los puntos de visión creando una imagen para cada ojo como se muestra en las figuras 3 y 4. En este caso cada imagen se crea ocupando un filtro de color. Finalmente, ambas imágenes se sobreponen para ser mostradas al usuario (imagen 5). La imagen debe ser vista con lentes cromáticamente opuestos, los cuales actúan como filtros, revelando únicamente una imagen para cada ojo.

Posteriormente el cerebro fusiona esta imagen compuesta como una sola, percibiendola en un espacio tridimensional. Los colores más ocupados en esta técnica son rojo y cyan.

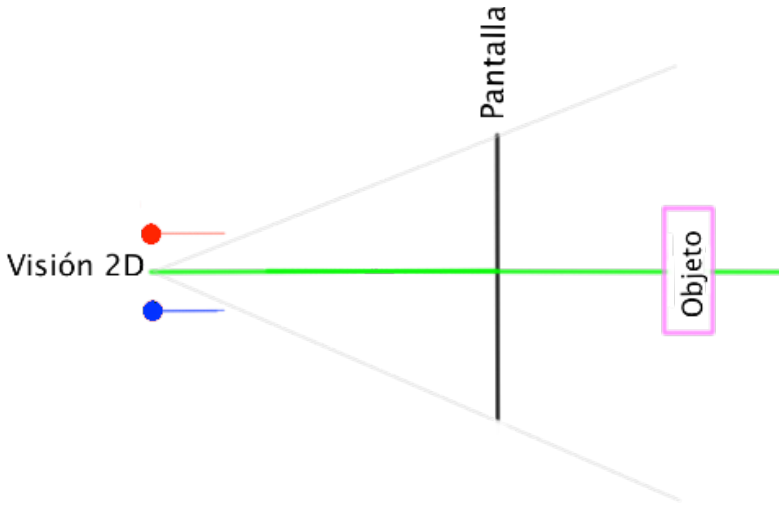


Figura 2: Creación de puntos de vista u "ojos"

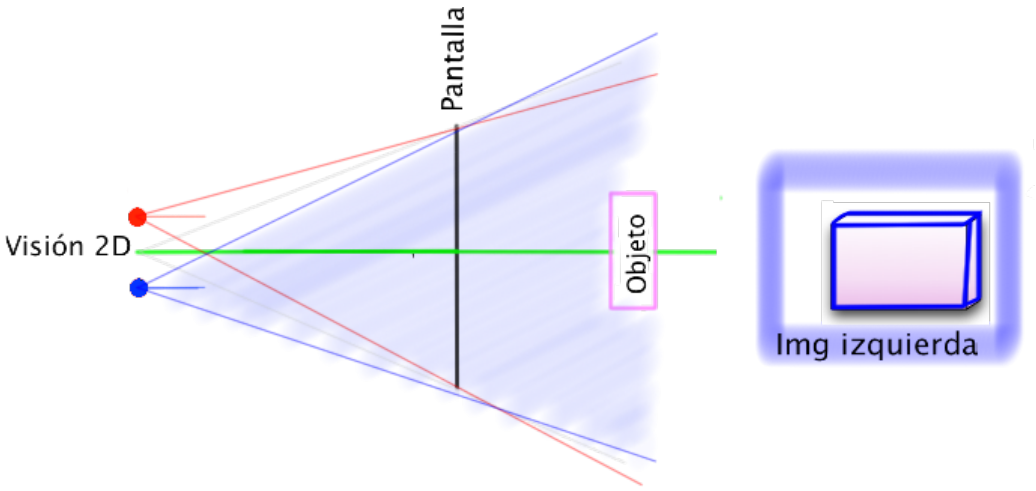


Figura 3: Procesamiento de la imagen para el ojo izquierdo

Esta misma técnica descrita se ocupa en todos los tipos de 3D, sólo que en vez de separar las imágenes por rangos de color, se toma alguna otra técnica para generar estas dos imágenes para cada ojo.

La principal ventaja de este método es que no se requiere de *hardware* especializado para lograr un efecto 3D y las gafas, que son el único requisito para el usuario, son muy económicas en comparación con otras tecnologías. La mayor desventaja de esta técnica es que no puede representarse todo el espectro de colores.

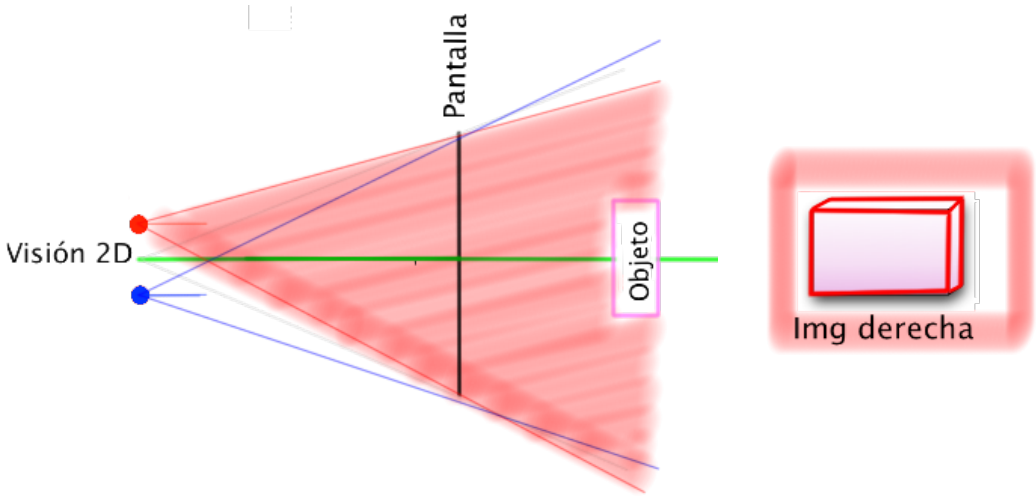


Figura 4: Procesamiento de la imagen para el ojo derecho

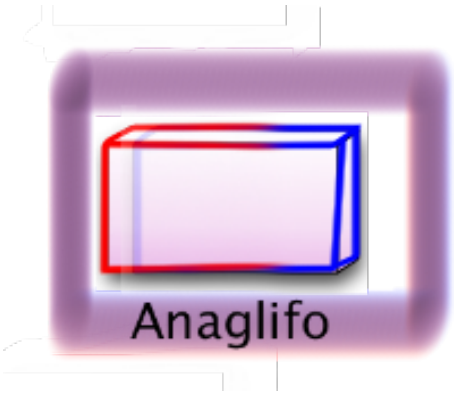


Figura 5: Imagen anaglifo resultante

Luz polarizada

En este método las imágenes de ambos ojos se distinguen por polarizar la luz en una manera circular o lineal. Al igual que en el modo anaglifo, los lentes funcionan como filtro, dejando pasar solo la luz del ojo correcto.

La luz emitida por una fuente generalmente va en todas direcciones, por lo que la luz emitida por un monitor de luz polarizada se encuentra en haces con diferentes direcciones. Los lentes para este tipo de tecnología son realmente filtros que dejan pasar únicamente componentes de la luz. Estos componentes pueden ser verticales, horizontales o incluso en espiral, dependiendo del fabricante. De esta manera, la luz que percibe cada uno de nuestros ojos depende del filtro que ocupó, por lo cual puede percibirse una imagen diferente para cada ojo. Esta funcionalidad se ilustra en la figura 6.

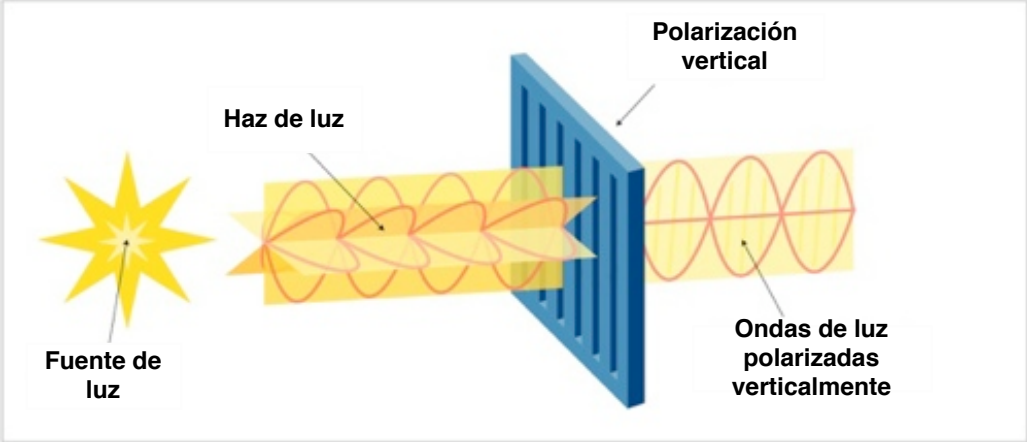


Figura 6: Funcionamiento de la luz polarizada

Ocupando este método se puede emplear todo el espectro de colores en las imágenes mostradas, pero se requiere de monitores especiales que preserven la polaridad de la luz.

3.4.3 Formas activas

En este tipo de formas se requiere de *hardware* específico en las gafas, así como en el dispositivo, con el fin de poder lograr un buen efecto 3D.

Lentes LCD y disparador

Este método ocupa unos lentes con LCD para cada ojo. Cada lente es sincronizado para abrir y cerrar en el tiempo que corresponda a la imagen que debe visualizar, como se ilustra en la figura 7. A los lentes se les indicará la velocidad y el tiempo en el que está por medio del disparador o sincronizador. Las imágenes del ojo izquierdo y derecho son presentadas alternadamente, lo que permite que cada ojo tenga tanto el espectro completo de colores, como alta resolución [NVIDIA, 2010]. La frecuencia de actualización con la cual cambian las imágenes depende de la frecuencia de actualización del monitor y es normalmente de 120 Hz (60 para cada ojo) e

incluso de hasta 150 Hz en algunos monitores compatibles con esta tecnología. Otra ventaja de este método es que no hay restricción del ángulo de visión para lograr el efecto 3D.

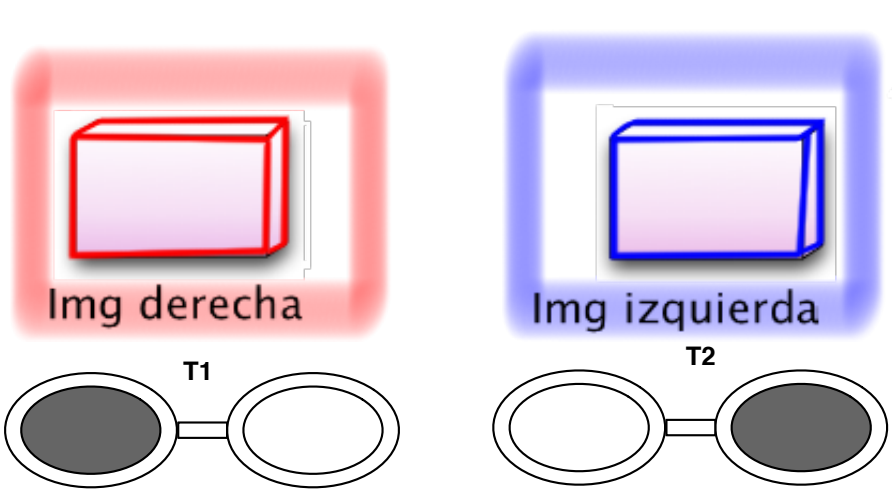


Figura 7: Funcionamiento lentes LCD

La principal desventaja de esta tecnología es el alto costo de las gafas en comparación con las tecnologías pasivas, el peso de las mismas, al tener su propia circuitería, es mayor y para algunos usuarios puede ser incómodo, al igual que a otros el parpadeo de los lentes les puede causar dolor de cabeza de acuerdo al tiempo de uso.

En esta tesis se opta por ocupar 3D activo con gafas LCD y dispositivo disparador, dado que con este método el efecto de inmersión que puede lograrse es mejor que con las otras tecnologías mencionadas anteriormente.

Capítulo 4

Aplicaciones *Rich-Client*

Se propone ocupar una aplicación *Rich-Client* para solucionar nuestro problema, dadas sus principales capacidades de acceso a las características del dispositivo sede, lo cual es necesario para deducir la forma en la cual los datos serán desplegados y su facilidad para ejecutar de manera nativa las partes de la aplicación que son importantes para la activación del *hardware* y *software* necesarios para el despliegue de contenido inmersivo. En este capítulo se hablará de los tipos de clientes Web, las ventajas que tienen las aplicaciones *Rich-Client*, herramientas que facilitan su desarrollo y aplicaciones que ocupamos cotidianamente que ya funcionan con esta tecnología.

4.1 Soluciones al desarrollo de aplicaciones multi-pataforma

Con el objetivo de tener una aplicación, que pueda ejecutarse en diferentes plataformas, se tienen dos opciones. La primera es desarrollar una aplicación nativa para cada plataforma, lo que consumiría mucho tiempo y requeriría de un grupo de desarrollo con conocimiento de diferentes lenguajes de programación que se ocupe en dichas plataformas. La segunda opción es hacer una aplicación Web [Hernández et al., 2013a], la cual debe conocer las características del dispositivo en el que se ejecuta. La aplicación Web podrá ejecutarse independientemente de las características del dispositivo, teniendo como único requisito que el dispositivo cuente con un navegador Web.

Se asume que hoy en día la mayor parte de los dispositivos electrónicos se conectan a Internet [David, 2010]. Actualmente, existen incluso electrodomésticos inteligentes con capacidad de conectarse a la Web, lo cual aumenta la gama de dispositivos para los cuales pueden desarrollarse aplicaciones y pueden interactuar entre sí para facilitar la vida del usuario. Esto hace más factible crear una aplicación Web que funcione en varios dispositivos y pueda adaptarse dependiendo de sus características.

4.1.1 Tipos de clientes Web

Dado el gran número de dispositivos en que puede desearse desplegar contenido, si queremos que nuestra aplicación funcione en todos, puede desarrollarse una aplicación nativa para cada dispositivo, lo cual es altamente complicado dado el gran número de dispositivos y plataformas

en las que nuestra aplicación se podría utilizar. Otra solución a esto es ocupar algún tipo de cliente Web para desplegar la información.

En las aplicaciones Web tradicionales la forma en que los contenidos están distribuidos entre clientes y servidores es limitada [Preciado et al., 2007]. Los tres tipos de clientes Web son *Thin*, *Fat* y *Rich-Client* [Hernández et al., 2013a].

Si ocupamos un cliente tipo *Thin*, el dispositivo únicamente sería usado para el despliegue de datos y todas las transformaciones se realizarían en un servidor remoto. En caso de un cliente tipo *Fat*, todos los cálculos se ejecutarían en el cliente, lo cual no es adecuado para todo tipo de dispositivo, por lo antes mencionado.

Las aplicaciones *Rich-Client* ocupan lo mejor de los tipos de clientes anteriores, comportándose de acuerdo al dispositivo en el que se encuentran ejecutando. La figura 8 muestra la arquitectura de una aplicación *Rich-Client*. La arquitectura consta de tres capas para presentación, negocios y acceso a datos respectivamente. La capa de datos accede a recursos y servicios de manera local o Web.

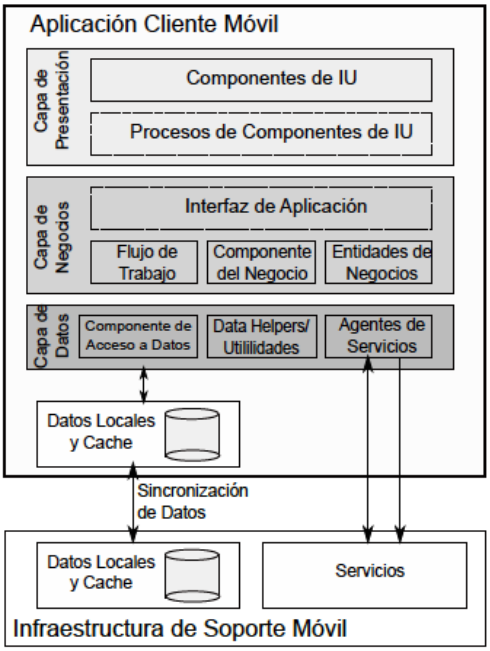


Figura 8: Arquitectura de una aplicación *Rich-Client*.

4.1.2 Ventajas y desventajas de las aplicaciones *Rich-client*

Los clientes *Rich-Client* tienen la ventaja de poder ejecutar cálculos como en una aplicación nativa [Meier et al., 2009], ya que pueden acceder a las características del dispositivo para determinar, de acuerdo a éstas, dónde ejecutar estos cálculos [Hernández et al., 2013a]; son capaces de hacer procesamientos en el cliente para después únicamente actualizar los cambios o resultados en el servidor (también llamadas aplicaciones temporalmente desconectadas) [Preciado et al., 2007]; son independientes del Sistema Operativo, lo que ayudaría a no requerir

muchas aplicaciones que hagan lo mismo en diferentes plataformas; finalmente, no necesitan de instalación, lo cual haría su uso más fácil incluso que aplicaciones nativas, ya que cualquier usuario con un navegador Web actualizado puede ocupar la aplicación en cualquiera de sus dispositivos y en cada uno de ellos ver la representación más adecuada de sus datos.

4.2 Herramientas de desarrollo para aplicaciones *Rich-Client*

Dado que las aplicaciones Web son muy variadas y requeridas en estos tiempos, han surgido diversas herramientas de desarrollo y APIs en los lenguajes más comunes [Hernández et al., 2013b]. A continuación, se mencionarán algunas de estas herramientas y cómo pueden servir para el desarrollo de esta tesis en la obtención de características y conocimiento del dispositivo sede.

4.2.1 Lenguajes

HTML5

HTML (*Hypertext Markup Language*) se desarrolló hace más de 20 años. En 1997 apareció en el mercado HTML4 y XHTML re-surgió, pero hubo muy pocos avances en todo este tiempo a la estructura general de HTML. HTML5 empezó a desarrollarse en el 2007, con el objetivo de desarrollar un estándar de HTML capaz de ejecutar aplicaciones completas en un navegador Web.

Las nuevas tecnologías incluidas en HTML5 incluyen características como: geolocalización, soporte a audio y video, canvas gráficos de SVG, CSS3, animaciones en dos y tres dimensiones y JavaScript 2.0 [Freemand and Robson, 2011]. Gracias a estas características, actualmente los desarrolladores Web pueden construir sitios aun mejores que aplicaciones de escritorio [David, 2010].

JavaScript

HTML5 permite crear aplicaciones tan poderosas como aplicaciones de escritorio, pero para complementar esto se necesita un lenguaje de desarrollo que permita crear soluciones sofisticadas. La respuesta a esto la encontramos en JavaScript [Wright, 2007].

JavaScript debe de ser interpretado por una máquina virtual dentro del navegador Web, lo que lo hace lento, pero ya se están trabajando en soluciones para esto [Wright, 2007]. Otra desventaja de la obtención de características en JavaScript es que, esta tarea se realiza antes de que se despliegue por completo la aplicación y, dependiendo del número de características que se requiera obtener, la carga de una aplicación puede tomar mucho tiempo.

CSS3

CSS3 es un estandar para CSS (*Cascading Style Sheets*). CSS3 sirve para definir archivos de estilo, los cuales van a describir como es la apariencia (capa de presentación) de una página HTML [Freemand and Robson, 2010]. Gracias a CSS, un sitio puede transformarse fácilmente en un sitio móvil, ya que en cada CSS se indica la apariencia de la página Web. Esto se logra

sólo referenciando a un diferente archivo CSS [Hernández, 2011]. Debido a que los navegadores móviles son los primeros en adoptar los estándares del W3C, es sencillo diseñar estilos para estos mismos.

4.2.2 Alternativas a JavaScript

Una alternativa más rápida para detectar características que JavaScript, es ocupar agentes de usuario. Los **agentes de usuario** son aplicaciones que funcionan como clientes en un protocolo de red. Al conectarse con un servidor envían una cadena con prefijo *"User-agent"* o *"User-Agent"*. Esta cadena contiene información del nombre de la aplicación, versión, tipo de Sistema Operativo e idioma.

La principal desventaja de los agentes de usuario es que la cadena no siempre está compuesta de la manera correcta, por lo que puede darnos información incorrecta o poco exacta del dispositivo a pesar de ser muy rápida.

Ua-parser, Jsperf y Platform

Son APIs que sirven para detectar características de dispositivos ocupando agentes de usuario. Dado el enfoque no garantizan que las características indicadas sean correctas.

Modernizr

Es un entorno de desarrollo que permite de manera gráfica seleccionar qué características del dispositivo se quieren sensar o evaluar en JavaScript. Estas características se obtienen antes de cargar la aplicación. Las características incluidas en este *framework* se dividen en cuatro categorías principales: CSS3 (muchos fondos de página, transformaciones 2D, translaciones, etc.), HTML5 (Web *Sockets*, memoria, video, etc.), Variadas (*webGL* eventos *touch*, geo-localización, etc.) y Extra (Clases CSS, etc.).

Battery Status API

Es un API definido bajo los estándares de W3C que proporciona funciones para determinar el estado de la batería del dispositivo. Algunas de las funciones más importantes son: detectar si el dispositivo está cargándose, tiempo que tiene en estado de carga, nivel de batería y tiempo que lleva descargándose.

FormFactor

Esta biblioteca nos permite saber el tipo de dispositivo en el cual se ejecuta nuestra aplicación. Con esta herramienta pueden detectarse dispositivos desde televisores inteligentes hasta teléfonos inteligentes.

Nvidia 3dvisionlive

Es una API en Java Script creado por NVIDIA para ayudar a detectar características de la tarjeta gráfica del dispositivo y *drivers* con los que cuenta. La API cuenta sólo con 4 funciones (obtener *driver*, versión de *driver*, capacidad estéreo, disponibilidad de estéreo). Sólo funciona con tarjetas gráficas de la familia Nvidia, Sistema Operativo Widows 7+ y en navegadores Mozilla Firefox.

4.3 Adaptabilidad de interfaces móviles

Actualmente, es común que al abrir una página en un navegador Web móvil ésta proponga al usuario que descargue una aplicación para la misma. Estas aplicaciones descargadas son aplicaciones nativas al Sistema Operativo del dispositivo móvil. Algunas páginas no se adaptan dependiendo del dispositivo, y otras re-ordenan los elementos con el fin de que sean mejor apreciados.

Algunos *frameworks* han sido desarrollados para facilitar la transformación de interfaces usando tres categorías de cambios: migración directa, transformación lineal y transformación de visión de conjunto [de Oliverira and da Rocha, 2006]. Desafortunadamente, la mayoría del trabajo realizado es sólo para móviles.

Algunas aplicaciones están diseñadas para usarse en televisores inteligentes y tiene su versión para Web, pero éstas también son aplicaciones nativas. Algunas aplicaciones sirven para adaptar aplicaciones Web a móviles removiendo colores, botones y *links*, para adaptarse a los estándares móviles [Yingling, 2006], pero el tamaño de la información presentada es limitada y la omisión de elementos afecta a la interpretación.

Capítulo 5

Desarrollo

En el presente capítulo se describe la forma en que se desarrolló este trabajo de tesis, se detallan todos los componentes de la infraestructura diseñada, así como la especificación del funcionamiento de los módulos realizados en este trabajo. Se detalla las herramientas ocupadas para la obtención de características de los dispositivos y se incluyen los diagramas de su estructura y funcionamiento.

5.1 Análisis

A lo largo de este trabajo de tesis se han desarrollado los temas importantes a considerar para lograr nuestros objetivos. El objetivo principal es diseñar una arquitectura que permita a aplicaciones *Rich-Client* dar soporte a sistemas de inmersión. Para probar la eficacia de nuestra arquitectura se desarrolla una aplicación que se apegue a la arquitectura propuesta.

Para que la arquitectura propuesta pueda ser ocupada para dar soporte a otros sistemas de despliegue de datos diferentes a los sistemas de inmersión, se propone primero una arquitectura general para dar soporte a la adaptabilidad de la información y de acuerdo a ésta se obtiene la arquitectura que funcione para dar solución al problema específico de los sistemas de inmersión. Para lograr tener una sola aplicación que se ejecute en cualquier dispositivo, independientemente de su plataforma (*hardware* y *software*), una solución adecuada y frecuentemente ocupada son las aplicaciones *Rich-Client*. Con el objetivo de lograr que en una aplicación Web se despliegue información de la mejor manera para los usuarios, se deben tomar en cuenta las características del dispositivo en el cual se ejecuta (*hardware* y *software*). Las aplicaciones *Rich-Client* también sirven para determinar las características del dispositivo en el que se ejecutan, para así determinar la manera en la que deben de comportarse o desplegar información. Las características principales que ayudan a determinar que tipo de despliegue se requiere en un dispositivo son: tipo de dispositivo, sistema operativo, dimensiones, contraste, resolución, y capacidades de despliegue especiales (capacidad 3D). Características como la capacidad de despliegue en 3D, deben de ser siempre ocupadas, ya que al desplegar información de manera inmersa, se logra una mejor interpretación de los datos y un mayor impacto en el usuario. La información que se requiere presentar al usuario, puede ser representada por muchos modelos de datos pero no todos son apropiados para todos los dispositivos. En dado caso de que la información no se encuentre disponible en el modelo de datos que mejor se adapte al dispositivo

sede, se debe realizar una transformación del modelo de datos disponible, al deseado. No todos los dispositivos son capaces de realizar estas transformaciones. En el caso de los dispositivos móviles, la ejecución de una transformación podría requerir un consumo energético significativo. En este caso, podría ser preferible realizar los cómputos de la transformación en un servidor externo.

5.2 Diseño

La arquitectura que se propone en este trabajo abarca la detección de las características del dispositivo, la obtención de información representada por modelos de datos, las conversiones entre modelos de datos como alternativa a formas de representar información y el despliegue de información de acuerdo a las características. Estas cuatro funciones se alberga en diferentes módulos que colaboran entre sí para dar soporte a la adaptabilidad de información en sistemas de inmersión.

En la figura 9 se muestra la estructura general de la arquitectura requerida por una aplicación para dar solución a la adaptabilidad de datos. La arquitectura esta segmentada en los diferentes módulos que mencionamos anteriormente. En esta estructura se toma en cuenta que la información puede tener cualquier representación.

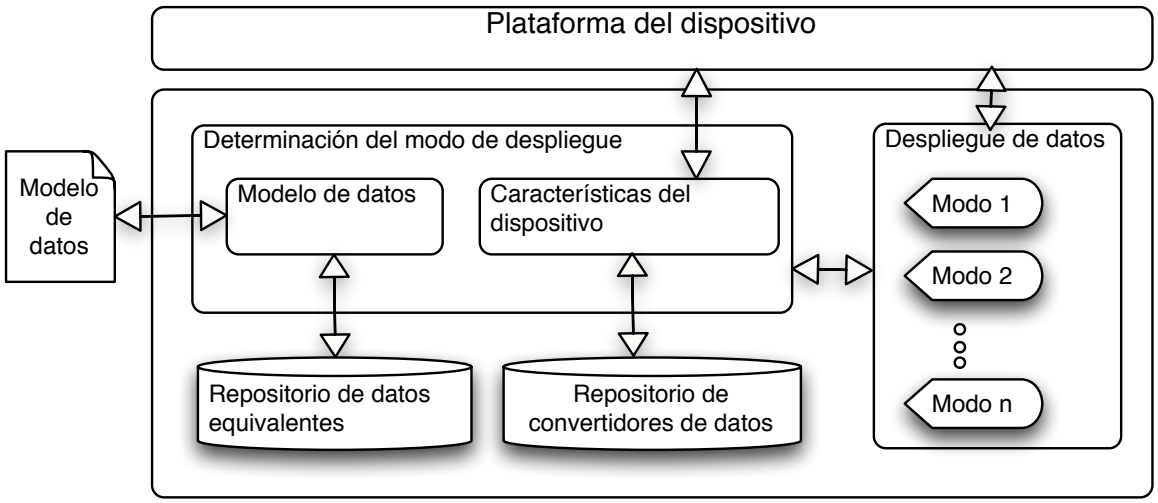


Figura 9: Estructura de aplicaciones *Rich-Client* para brindar adaptabilidad de información

El módulo de determinación del modo de despliegue es en sí la aplicación *Rich-Client*. El módulo de despliegue de datos está integrado, por el momento por dos aplicaciones independientes y el despliegue mismo de la aplicación *Rich-Client*. Cada aplicación despliega un modelo en tres dimensiones dependiendo de los requerimientos de entrada del usuario y el modo de despliegue que decida la aplicación *Rich-Client*. Las aplicaciones están desarrolladas en DirectX y OpenGL.

El módulo principal de la arquitectura es el encargado de la determinación del modo de despliegue, el cual accede a la plataforma del dispositivo. Nos referimos al término "plataforma

del dispositivo" a todas las características, de *hardware* y de *software*, que influyen en el tipo de despliegue que se tendrá. Después de evaluar la plataforma, se toma en cuenta el formato de los modelos que el usuario desea desplegar y los requerimientos de cada tipo de despliegue. En caso de no contar con un modo de despliegue compatible con los modelos de datos requeridos, se accede al repositorio de datos equivalentes y en caso de no contar con alguno, se ocupa el módulo de transformación de datos. Una vez realizadas estas acciones, y habiendo decidido el tipo de despliegue que se efectua en el dispositivo, se llamará al módulo de despliegue de datos. Estas funciones se detallarán en la siguiente sección.

Para nuestro caso de estudio específico, los modelos de datos ocupados pueden ser únicamente representaciones de objetos tridimensionales. Estas representaciones contienen información de los vértices de la figura, posiciones y aristas entre los vértices (caso de los formatos *obj*), por mencionar algunos.

Estos modelos pueden ser representados de diferentes maneras, como se menciona en la sección 3.4. Para el problema que nosotros abarcamos en el caso de despliegue de contenido 3D activo ocupamos los formatos *sdkmesh* para el modelo de datos, *fx* para el efecto y *png* para la textura. En el caso de despliegue de contenido pasivo (anaglifo) se ocupa únicamente el formato *obj*.

La figura 10 ilustra la relación que se tiene entre el tipo de datos, las formas que existen para representarlo y los dispositivos en que pueden ser visualizados correctamente.

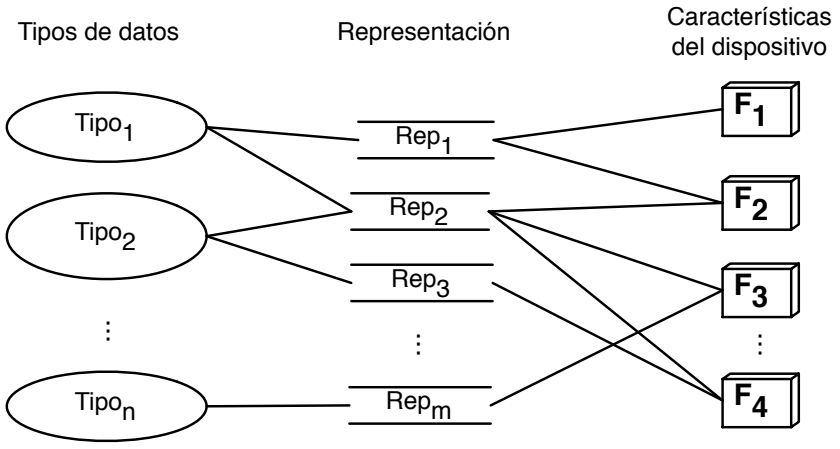


Figura 10: Mapeo de tipo de datos a dispositivos

Un tipo de datos puede tener diferentes representaciones. Cada representación es adecuada para su despliegue en una plataforma, la cual está compuesta por las características del dispositivo sede. Un modelo u objeto tridimensional puede estar albergado en diferentes formatos que representen y describan la misma información, pero cada una de estas podrá desplegarse según las especificaciones del módulo de despliegue de información de la aplicación.

En el caso particular de los sistemas de inmersión, un modelo de un objeto tridimensional puede ser representado en muchas formas, entre estos formatos están el *obj* y el *sdkmesh*. Dependiendo del formato o representación que se tenga de estos modelos, se sabe en qué tipos

de dispositivos puede desplegarse adecuadamente. Si no se tiene la adecuada relación de la representación y el tipo de dispositivo, se busca una representación equivalente a la pedida, para lo cual se pueden requerir transformaciones de modelos de datos.

Se han definido diversos protocolos para la solicitud de información, los cuales son aplicables a la adaptabilidad de datos en general y a este caso de estudio. La aplicación desarrollada se apega a protocolos propuestos para el funcionamiento adecuado de la adaptabilidad de información, de los cuales a continuación se mencionarán los más importantes.

El primer caso en el que se aplican estos protocolos es cuando el usuario pide un modelo de datos en específico e indica la forma en que quiere que éste se despliegue, independientemente de las características del dispositivo. En este caso la aplicación *Rich-Client* evalúa las características y decide que el dispositivo no va a ser capaz de desplegar el modelo de una forma adecuada, así que le envía una advertencia. El usuario debe hacer otra solicitud. Un ejemplo de este caso es si el usuario requiere desplegar información de manera inmersa pero el dispositivo sede no tiene las características específicas para este tipo de despliegue. Si el dispositivo no provee su lista de características, la aplicación no será capaz de decidir si el dispositivo podrá desplegarlos de la manera solicitada, por lo que mandará una advertencia de que no podrá determinar la forma correcta de despliegue. Este protocolo está ilustrado en el diagrama de secuencia de la figura 11.

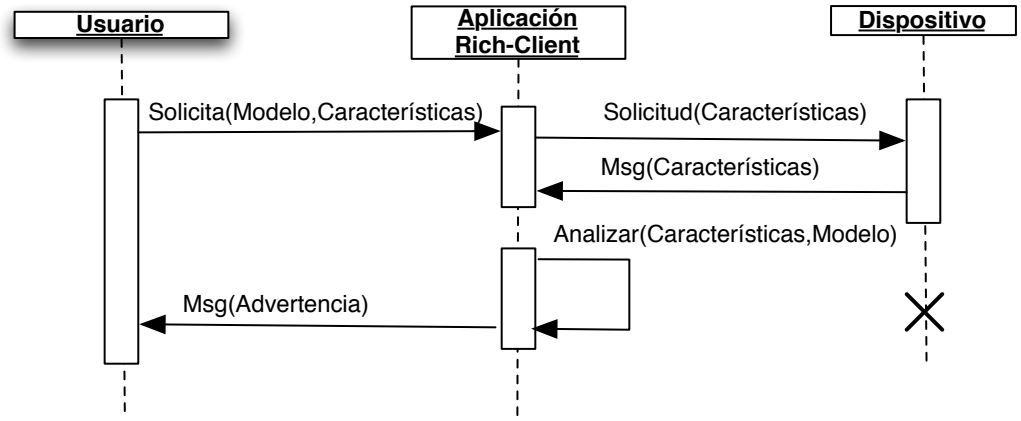


Figura 11: Protocolo del caso 1

La figura 12 ilustra el caso ideal en el cual la aplicación decide que el modelo de datos solicitado por el usuario podrá ser desplegado de manera correcta en el dispositivo sede. La aplicación *Rich-Client* indica al dispositivo qué modo de despliegue se ocupará, para que éste prepare todo lo necesario, por ejemplo la activación del modo estéreo, en el caso del 3D activo, y así pueda desplegar el contenido de la manera adecuada.

La figura 13 muestra el caso donde el modelo proporcionado por el usuario no está en la representación requerida por el modo de despliegue que la aplicación *Rich-Client* determinó como la más adecuada para el dispositivo, por lo tanto debe realizarse una transformación. La aplicación *Rich-Client* evalúa las características del dispositivo para saber si el dispositivo es capaz de realizar los cálculos computacionales para obtener el mejor modelo de datos que se adecúe a sus características. Si no pueden realizarse los cálculos, se intentará al menos desplegar

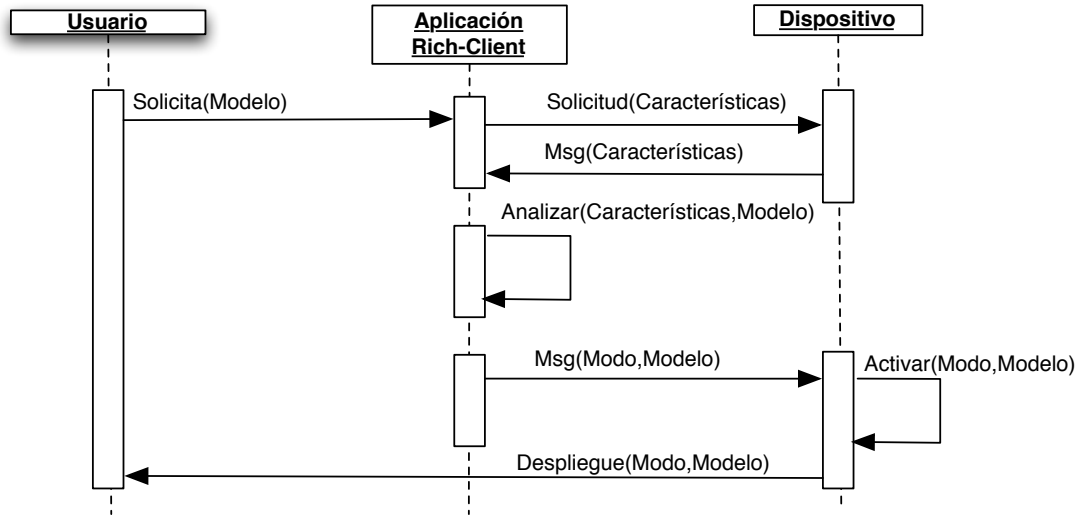


Figura 12: Protocolo del caso 2

algún tipo de información que dé idea del modelo que el usuario solicitó. La aplicación *Rich-Client* buscará en el repositorio de modelos equivalentes algo que pueda desplegar (generalmente imágenes o archivos multimedia). En caso de que sí se tenga una representación equivalente, la aplicación *Rich-Client* la desplegará para que el usuario al menos visualize un poco de la información solicitada.

Una ventaja de este método es que el despliegue es independiente a las características del dispositivo, por lo que si se tiene un modelo equivalente, en cualquier dispositivo se podrá obtener al menos parte de la información solicitada.

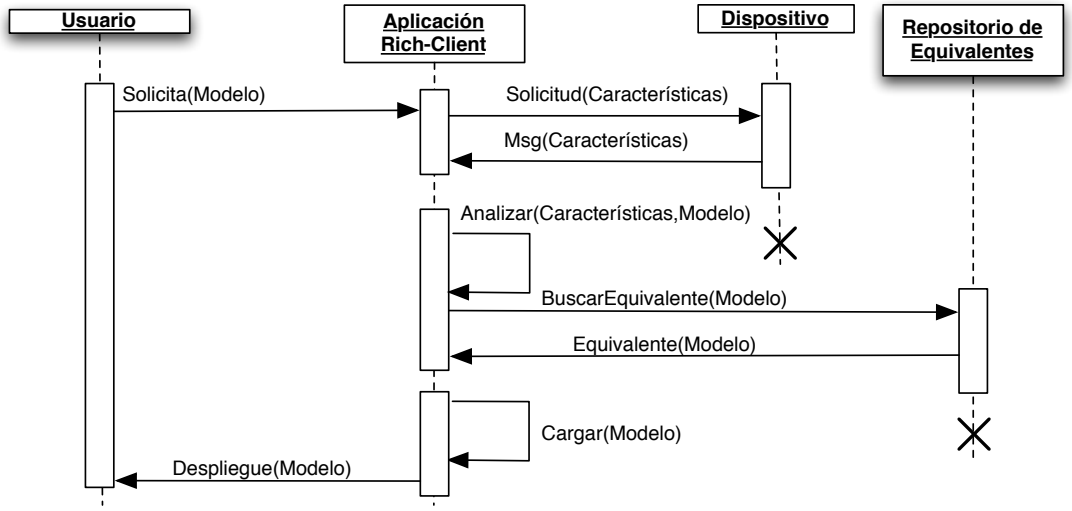


Figura 13: Protocolo del caso 3

El caso mostrado en la figura 14 es un caso similar al caso tres, con la diferencia de que la aplicación *Rich-Client* decide que el dispositivo va a ser capaz de realizar la transformación

entre tipos de representaciones del modelo. La aplicación *Rich-Client* pide al módulo de transformación que efectúe el cambio en los modelos y si existe una transformación del modelo A al modelo B, éste último será regresado a la aplicación *Rich-Client*. Una vez terminada la transformación, la aplicación *Rich-Client* indica al dispositivo el modo de despliegue que se tomará para que pueda activarse y preparar lo necesario para el despliegue. Finalmente, el dispositivo despliega la información ocupando la nueva representación del modelo de datos.

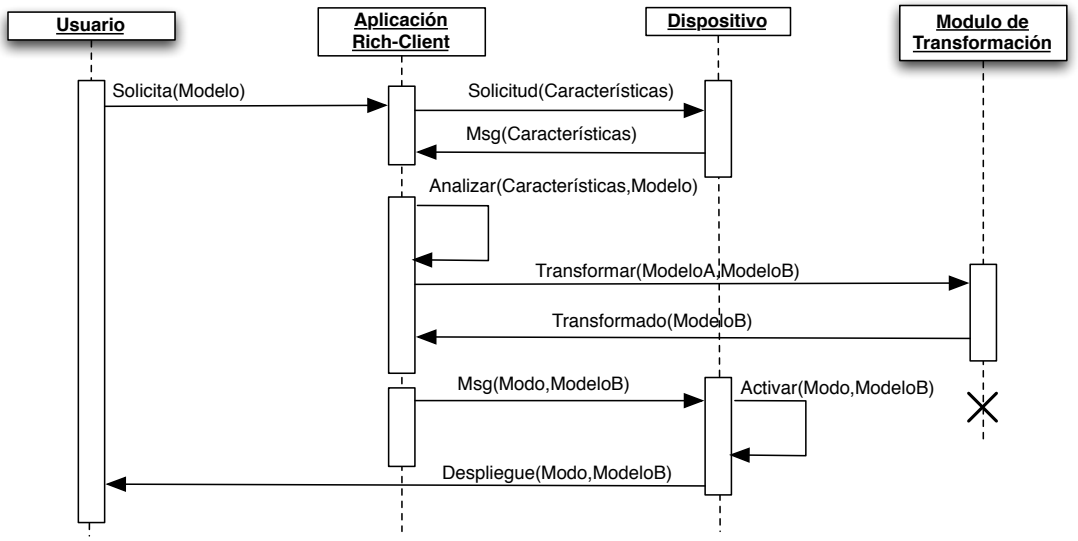


Figura 14: Protocolo del caso 4

5.3 Implementación

Como se mencionó en las secciones anteriores, se diseñó una aplicación para poder probar la manera en que funciona nuestra arquitectura y saber si es eficaz y adecuada. La implementación, para apegarse a la arquitectura, debe constar de tres módulos principales: módulo de determinación del modo de despliegue, módulo de transformación u obtención de datos equivalentes y módulo de despliegue. En esta sección se detalla la implementación, herramientas y tecnología ocupada para el desarrollo de cada uno de estos.

5.3.1 Módulo de obtención de características

Aplicación *Rich-Client*

La aplicación *Rich-Client* es la parte fundamental de esta implementación ya que es la parte encargada de obtener las características básicas del dispositivo sede y determinar la forma en que debe ser desplegada la información.

En este módulo las características del dispositivo a tomar en cuenta son: Sistema Operativo, dimensiones, capacidad de despliegue de contenido 3D activo (lentes 3D activos de Nvidia, monitor activo y controladores), capacidad de despliegue 3D pasivo (OpenGL 4.0 o más reciente)

y capacidad de despliegue 2D (OpenGL 2.0). En caso de que no se cuente con Sistema Operativo Windows o no se tengan las dimensiones requeridas (a partir de 10 pulgadas) no se intentará desplegar contenido en 3D o 2D y únicamente se desplegará alguna salida en caso de tener un equivalente en formato imagen.

Para saber si un dispositivo puede desplegar contenido 3D activo se ocupó la biblioteca *NVStereoUtils* de Nvidia, la cual sirve para determinar dentro de aplicaciones Web si se cuenta con el *hardware* y *software* necesario para ocupar Nvidia 3D Vision. Esta biblioteca toma en cuenta versión y tipo de Sistema Operativo, navegador Web en que se ejecuta la aplicación, versión de los controladores y estado del modo estéreo.

Para saber si puede desplegarse 3D anaglifo o un modelo en 2D sólo se verifica la versión de OpenGL que tiene el dispositivo sede.

El orden en que ésta determina el mejor modo para desplegar información en un dispositivo, de acuerdo a sus características, se describe en el diagrama de flujo de la figura 15.

El primer paso en la determinación del modo de despliegue es saber el formato en el cual se encuentra el modelo de datos que el usuario desea desplegar. Esto se realiza únicamente con el conocimiento del nombre del modelo de datos y la extensión del archivo que el usuario solicita.

Se determinará cuál es la mejor forma de despliegue de datos, dadas las características del dispositivo, descartando las formas para las cuales no se tengan recursos, empezando por despliegue 3D activo, 3D pasivo, 2D y finalizando con equivalente.

Para el despliegue de datos de manera 3D activa se necesita tener tres archivos en formatos *sdkmesh*, *fx* y *png*, respectivamente. En caso de sólo proporcionar los primeros dos se tomará un *png* por omisión para tomar el rol de la textura, aunque esto no garantiza una adecuada visualización del modelo.

Para despliegue 3D pasivo y 2D se requiere únicamente un modelo de datos en formato *obj*. Se escogió este formato dado que es uno de los más sencillos de ocupar, debido a su estructura descriptiva; también ha sido fácilmente adoptado por muchas otras aplicaciones e implementaciones 3D, como son Maya y 3Ds Max.

Si los requerimientos del tipo de datos que puede desplegarse coinciden con los formatos del modelo de datos solicitado por el usuario se procede al despliegue de información. En caso contrario se buscará si se tiene en el repositorio de datos el formato equivalente del modelo de datos, que coincide con el mejor tipo de despliegue permitido por el dispositivo. En caso de contar con él se desplegará.

Si no se cuenta con un equivalente, se buscará un convertidor del modelo de datos existente al modelo de datos requerido por el tipo de despliegue permitido por el dispositivo. Si existe este convertidor se realiza la transformación y el despliegue de información.

En caso de no contar con ninguno de los anteriores se despliega un mensaje con la explicación y tal vez alguna sugerencia como podría ser la actualización de los controladores.

Otras características

A parte de las características obtenidas por el módulo de determinación de características, cada una de las formas de despliegue de datos cuenta con una parte dedicada a la obtención de otras características del dispositivo como son contraste y resolución. Estas últimas mencionadas se obtienen por medio de la API *nvapi* de Nvidia.

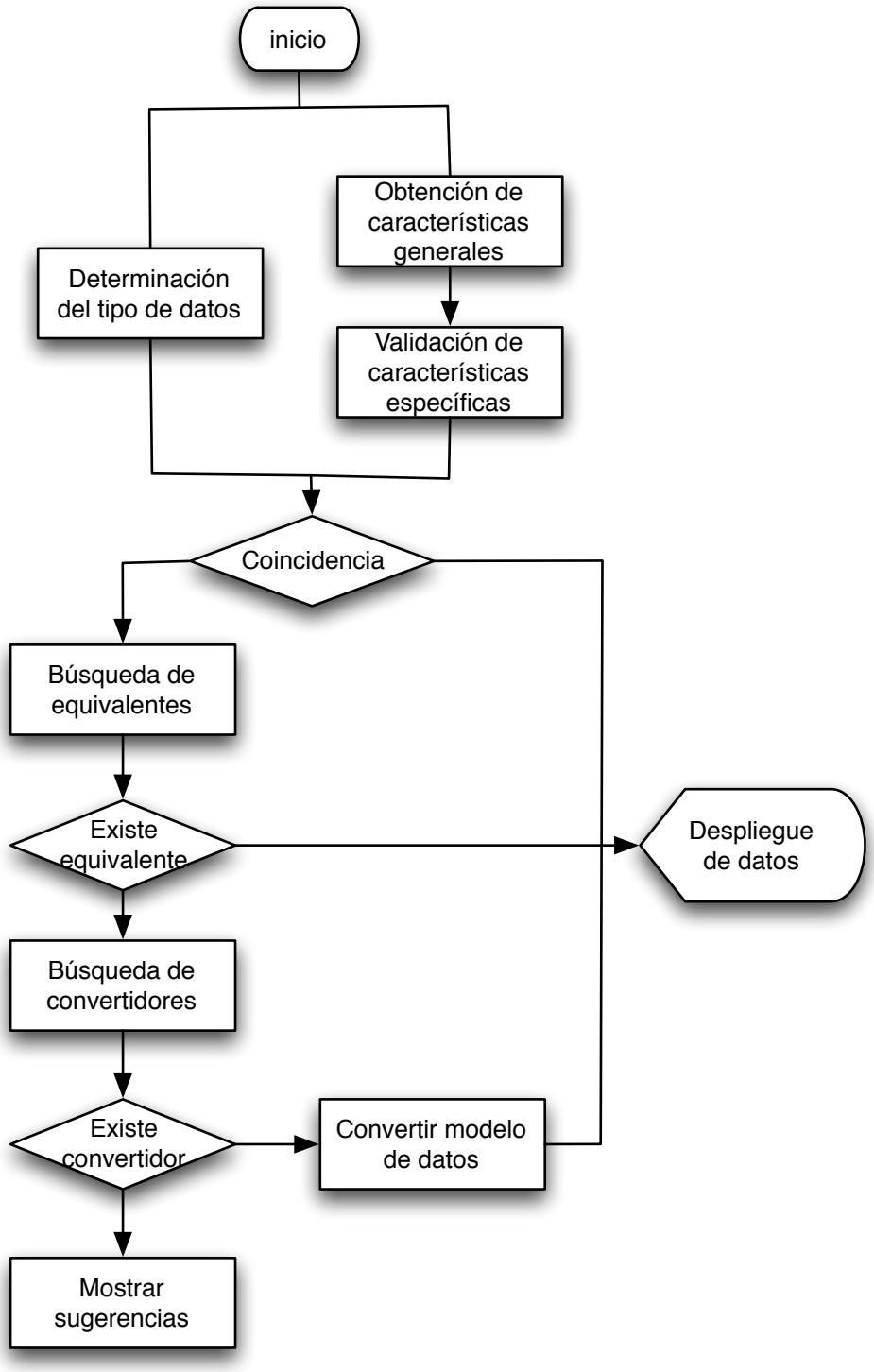


Figura 15: Funcionamiento de aplicación *RichClient*

Por el momento estas no influyen en la forma en que se despliegan los modelos de datos, pero se deja abierto para averiguar qué tanto es que estos dos factores influyen en la manera en que el usuario percibe los datos.

5.3.2 Módulo de transformación de datos

En caso de que la información esté representada en un modelo de datos que no se encuentre en el formato en el cual la mejor forma de despliegue de nuestro dispositivo requiere, debe hacerse una transformación. En nuestro caso particular la única conversión generada es del formato *sdkmesh* a formato *obj*. Esta conversión se realiza en el caso de que el usuario cuente con un modelo *sdkmesh* pero de acuerdo con las características de su dispositivo la aplicación *Rich-Client* determina que el mejor despliegue es 3D pasivo anaglifo o una salida 2D, salidas para las cuales se requiere únicamente de un formato *obj*.

El convertidor con el que se cuenta es ocupado únicamente por DirectX, por lo cual se requiere tenerlo instalado. Como el formato *sdkmesh* es únicamente ocupado por DirectX, se consideró que los usuarios, con modelos en este formato, cumplirían fácilmente con este requerimiento.

En caso de conversiones en los cuales se requiera transformar modelos de *obj* a modelos en formato *sdkmesh*, no se proporciona aún una forma de conversión, ya que a parte de un modelo en el formato *sdkmesh*, una aplicación para despliegue activo requerirá de texturas en formato *png* y efectos en formato *fx*, los cuales no se podrían obtener del modelo *obj*.

5.3.3 Módulo de despliegue de información

Este módulo está constituido por diferentes aplicaciones que despliegan modelos de datos dependiendo del tipo de despliegue requerido por el dispositivo. Dados los requerimientos y las técnicas ocupadas para la creación de inmersión en 3D, estas aplicaciones operan de manera diferente e incluso se encuentran codificadas con diferentes lenguajes y herramientas. A continuación se detalla la implementación de estas aplicaciones.

Inmersión con 3D activo

Las aplicaciones con despliegue de contenido 3D son más atractivas y entendibles para usuario. En el caso del 3D activo puede lograrse que sean aún más llamativas que otro tipo de despliegue 3D, ya que con éste puede representarse toda la gama de colores. Es por esto que la implementación desarrollada tiene como objetivo principal el poder desplegar datos con tecnología activa.

Se decidió ocupar la tecnología activa de Nvidia 3D Vision, ya que hoy en día es, lo más ocupado en la industria de los videojuegos y simulaciones. Para ocupar esta tecnología se requiere un monitor listo para el despliegue de contenido 3D compatible con Nvidia (3D Vision Ready Monitor), tarjeta gráfica con soporte para 3D y un par de gafas LCD de Nvidia (3D Vision, 3D Vision 2 o 3D Vision Pro). 3D Vision da soporte para desarrolladores que deseen crear aplicaciones en DirectX y a partir de 3D Vision Pro se incorporó también soporte para

OpenGL. Para desarrollar aplicaciones es necesario contar con una tarjeta gráfica de la familia Quadro de Nvidia con soporte a 3D.

Dado que para el desarrollo de esta tesis sólo se cuenta con gafas 3D Vision 2, la aplicación se desarrolló con DirectX 9. La ventaja de ocupar esto es que nuestra aplicación tiene soporte para los 3 tipos de gafas con los que Nvidia cuenta actualmente.

Para el desarrollo de la aplicación se ocupó el API *nvapi* para obtener las características del dispositivo. Estas características son resolución de pantalla, contraste del monitor, versión de los controladores (de tarjeta gráfica, gafas y disparador), disponibilidad de estéreo y estado de estéreo (activado o desactivado). Para la correcta utilización del dispositivo y manejo de ventanas en Windows, se ocupó la biblioteca SDL (Simple DirectMedia Layer), la cual sirve para acceder por medio de bajo nivel al hardware dedicado a gráficos tanto en OpenGL como DirectX.

Para poder desplegar un modelo, la aplicación requiere como entrada tres archivos. El primer archivo es de formato *sdkmesh* y contiene la información de creación del modelo, el segundo es de formato *fx* y contiene la información del efecto del modelo y por último un archivo *png* para la textura del modelo. Esté último archivo no es obligatorio como entrada del usuario; en caso de que no se proporcione, se tomará un *png* por omisión (imagen totalmente blanca), aunque con esto no se garantiza una adecuada visualización del modelo de salida. No se crea un archivo con formato *fx* por omisión para el efecto, ya que este formato es complejo e importante para la manera en que se despliega el modelo. De no ser compatible con el modelo, puede ser que incluso no se visualice información alguna en pantalla.

A pesar de que la aplicación *Rich-Client* haya determinado que la mejor forma para desplegar información en un dispositivo es de forma inmersa con 3D activo, puede ser que por algún error no pueda activarse correctamente el estéreo; en dado caso el modelo sólo se desplegará en dos dimensiones. En caso de que no se tenga ningún error, se activará el modo estéreo y el modelo se desplegará en 3D.

En todas las aplicaciones que ocupan 3D es posible que aún activando el modo "estéreo", el usuario no visualice el modelo adecuadamente en tres dimensiones. Esto puede deberse a la falta de sincronización entre las gafas LDC y el disparador o incluso a las condiciones médicas del usuario. Para evitar daños a la salud del usuario o detectar algún problema en el funcionamiento del 3D, se aplican imágenes llamadas *pruebas médicas*, las cuales consisten en mostrar imágenes en modo 3D al usuario, pedirle que cierre un ojo a la vez y preguntarle qué es lo que ve. Para tener una validación de este tipo, en nuestra implementación ocupamos las imágenes para pruebas médicas que Nvidia ocupa para la instalación de 3D Vision. Esta prueba únicamente se ejecutará siempre que el usuario ocupe el modo de 3D activo en nuestra aplicación, debido a que la sincronización entre las gafas y el disparador puede variar al haber cambiado desde la última vez que se ocupó este modo.

Inmersión con 3D pasivo y despliegue 2D

El contenido 3D pasivo de manera anaglifo es sencillo de generar y requiere de costos mínimos por parte del usuario al sólo requerir un par de gafas de colores; ésta es la razón por la cual este tipo de despliegue tiene el segundo lugar en preferencia entre los modos de despliegue en nuestra implementación.

Este tipo de despliegue sólo requiere que el dispositivo tenga instalada una versión de OpenGL a partir de la 4.0 y como datos de entrada se requiere únicamente un modelo tridimensional en formato *obj*. Para el desarrollo de esta aplicación se ocupó el kit *Nvidia core SDK* para la obtención y el manejo de los modelos.

Para lograr el efecto deseado en la imagen, debe realizarse un procesamiento de la imagen dos veces, cada uno con un filtro de color, en este caso rojo y cyan. Una cámara debe situarse en el foco de cada uno de los "ojos del espectador" (como se indica en la sección 3.4), para así obtener la imagen visualizada por cada foco. Al final, las imágenes obtenidas por cada foco se sobreponen en una sola imagen y el cerebro humano realiza el trabajo restante. Los objetos ocupados para este proceso se pasaron primero a escala de grises, para maximizar el contraste y por lo tanto lograr un mejor efecto de inmersión.

La única manera de saber si un usuario cuenta con gafas de tipo anaglifo es ejecutando una prueba médica. De nueva cuenta se ocuparon las imágenes de pruebas médicas de Nvidia en su versión anaglifo. En este caso la prueba médica se ejecuta únicamente una vez, ya que a diferencia del 3D activo con gafas LCD, las gafas del tipo anaglifo no requiere de sincronización. Esto también se justifica bajo la suposición de que si el usuario contó con los lentes, seguirá contando con ellos.

En caso de que la prueba médica indique que el usuario no cuenta con gafas de tipo anaglifo, no puede percibir correctamente el 3D en este estilo o cuente con OpenGL, pero no en una versión mayor a 4.0, el modelo será procesado una sola vez de manera normal y el resultado será desplegado de forma plana en pantalla.

Despliegue con contenido equivalente

Esta opción fue pensada dada la posibilidad de que un gran número de dispositivos no cuenta con todas las características que requerimos para los tipos de despliegue anteriores, en cuyo caso no se desplegaría información alguna. Para solucionar ésto se ideó tener una alternativa a algunos modelos como son imágenes de sus perspectivas. En caso de que el usuario desee desplegar un modelo y no cuente con las características básicas para despliegue en dos dimensiones (OpenGL 2.0), el cual es el modo de despliegue más básico, se buscará si se tiene un equivalente, en este caso las imágenes de la perspectiva del modelo, y se desplegará directamente en la aplicación *Rich-Client*, evitando dejar al usuario sin visualización de su información.

Capítulo 6

Pruebas y resultados

En este capítulo se describe la forma en que se probó la arquitectura propuesta en esta tesis. Como se ha mencionado antes, para probar la arquitectura se desarrolló una aplicación para así conocer qué tan adecuada resulta para resolver el problema de adaptabilidad de datos en sistemas de inmersión. Para probar la aplicación se proponen diferentes casos de prueba o escenarios, en los cuales diferentes plataformas (configuraciones de *hardware* y *software*) fueron ocupadas, así como también se probaron diferentes tipos de modelos de entrada o representaciones.

6.1 Pruebas

Se realizaron pruebas de efectividad para saber si la aplicación determina correctamente el tipo de salida que debe desplegarse en cada plataforma. Se considera que una prueba fue adecuada cuando el tipo de representación final que se despliega coincide con el modelo de datos de entrada, dadas las características del dispositivo, y la mayor cantidad de información (modelo, efecto y textura) se le presentó al usuario.

A continuación se mencionan los recursos con los cuales se realizaron las pruebas, así como sus características.

6.1.1 Recursos de *hardware*

- Tarjetas gráficas de la familia Nvidia:
  - GeForce GTX 550 TI (adecuada para desplegar contenido 3D activo).
  - Quadro FX 540 (no adecuada para el despliegue de contenido 3D activo).
  - Quadro K2000 (adecuada para la creación y el despliegue de contenido 3D activo).
- Monitores:
  - ViewSonic V3D231 (despliegue de 3D pasivo con tecnología de luz polarizada).
  - BenQ XL2720T (despliegue de 3D activo).
- Kit 2 de Nvidia para visión 3D (gafas LDC y disparador).

6.1.2 Recursos de *software*

- Sistema Operativo: Windows 8 a 64 bits.
- OpenGL versión 4.3.
- DirectX versiones 9 y 10.

Además de los recursos previamente mencionados, se requiere un dispositivo que no cuente con las características necesarias para el despliegue de modelos 3D, con lo que se podrá probar también este caso.

6.2 Análisis del experimento

Para comprobar la eficacia de nuestra implementación se ocuparán los siguientes casos de prueba (CP):

- **CP1-** Prueba con tarjeta gráfica Quadro K2000 y modo estéreo activado: se muestra el caso idóneo en el cual la tarjeta gráfica tiene capacidad para el despliegue de contenido 3D y el estéreo ya se encuentre activado.
- **CP2-** Prueba con tarjeta gráfica Quadro K2000 y modo estéreo desactivado: caso en el cual la tarjeta gráfica tiene capacidad para despliegue de contenido 3D, pero al no encontrarse el modo estéreo activado, será el módulo de despliegue el que tenga que activarlo.
- **CP3-** Prueba con tarjeta gráfica GeForce GTX 550 TI y modo estéreo activado: al ser las tarjetas de la familia GeForce adecuadas para desplegar contenido 3D activo pero no para desarrollar aplicaciones que lo ocupan. Este casó servirá para saber si el funcionamiento de nuestra implementación se acota únicamente a tarjetas gráficas de la familia Quadro o si también se incluye la familia GeForce.
- **CP4-** Prueba con tarjeta gráfica GeForce GTX 550 TI y modo estéreo desactivado: este casó demostrará si en tarjetas de la familia GeForce, el módulo de despliegue de datos de nuestra implementación, puede activar o no el modo estéreo para ocuparlo en el despliegue.
- **CP5-** Prueba con tarjeta gráfica FX 540: este caso demostrará qué ocurre cuando nuestra aplicación detecta una tarjeta gráfica que no tiene capacidad de despliegue de contenido en 3D.
- **CP6-** Prueba con tarjeta gráfica Quadro K2000 y modo estéreo activado y monitor no compatible con Nvidia 3D Vision: este caso servirá para saber qué ocurre cuando lo único que no coincide con el caso idóneo es el monitor. Se probará con un monitor adecuado para despliegue de 3D de manera pasiva con tecnología de luz polarizada.

- **CP7-** Prueba con tarjeta gráfica Quadro K2000 y modo estéreo activado y controladores de 3D Vision no adecuados: este caso servirá para saber qué ocurre cuando los controladores de las gafas LCD y disparador no se encuentran correctamente instalados.
- **CP8-** Prueba con tarjeta gráfica Quadro K2000 con controladores no adecuados: este casó servirá para saber qué ocurre cuando los controladores de la tarjeta gráfica no se encuentran correctamente instalados.
- **CP9-** Prueba con dispositivo no compatible en lo absoluto con los requerimientos de nuestra implementación.
- **CP10-** En este caso de prueba, se tienen las características necesarias para despliegue de manera activa, pero se requerirá un modelo en formato *obj*.
- **CP10-** Se requerirá desplegar un formato *sdkmesh* en un dispositivo con las características necesarias para hacerlos, pero sólo se proveerá el formato *fx* para el efecto, sin proveer el *png* para la textura.
- **CP11-** Se requerirá desplegar un formato *sdkmesh* en un dispositivo con las características necesarias para hacerlo, pero no se proveerán ni el formato *fx* para el efecto ni el *png* para la textura.

Todos los casos, a excepción del caso CP9, se probarán con un Sistema Operativo Windows 8, OpenGL 4.0 y DirectX 9+. El caso CP7 se probará en una Macbook Pro de 13 pulgadas, con Sistema Operativo OS X 10.9.4 y tarjeta gráfica Nvidia GeForce 320M (no compatible con Nvidia 3D Vision). También en los casos del CP1 hasta el caso CP9, el modelo requerido está en formato *sdkmesh* y se especifica el archivo *fx* para efecto y *png* para textura a utilizar. A partir del caso CP10 se especifica los modelos de entrada y sus formatos.

Cada uno de los casos de prueba pueden ser evaluados en los siguientes rubros:

- Eficacia de la aplicación para determinar el mejor modo de despliegue de un dispositivo dado sus características.
- Empleo correcto de los protocolos propuestos para el funcionamiento de la arquitectura.
- Transformación del modelo de datos requerido, en caso de ser necesario.
- Despliegue adecuado de los modelos de acuerdo al resultado de la aplicación *Rich-Client*.

6.3 Resultados

CP1

En esta prueba se esperaba poder ver el modelo requerido en forma inmersa y activa. La prueba salió como se esperaba, ya que la aplicación *Rich-Client* indica que la mejor forma para desplegar información era con 3D activo y el módulo de despliegue mostrando correctamente la información.

CP2

El módulo de despliegue decidió adecuadamente que el sistema cuenta con lo necesario para hacer despliegue de forma inmersa y activa, por lo que la aplicación *Rich-Client* determinó de manera correcta que el modo adecuado de despliegue era 3D activo. En este caso la variante es que al ejecutarse el módulo de despliegue, el modo estéreo del dispositivo no se encuentra activado en el sistema. Al ejecutarse el módulo de despliegue, éste pudo activar adecuadamente el modo estéreo del dispositivo y posteriormente desplegar el modelo de la manera esperada.

CP3

Al igual que en los casos anteriores, esta prueba pudo desplegar el modelo 3D de forma activa. Dos casos más se realizaron de esta prueba. El primero ejecutando el código desde visual C++ y el otro únicamente con el ejecutable. En ambas formas, la aplicación *Rich-Client* determinó que el mejor modo para desplegar el modelo era 3D activo y el módulo de despliegue de datos presentó el modelo de manera correcta.

CP4

De esta prueba también se realizaron dos casos, uno con el código ejecutado desde visual C++ y el otro únicamente con el ejecutable. La aplicación *Rich-Client* determinó que el mejor modo era despliegue 3D activo en ambos casos, pero el módulo de despliegue no logró activar el modo estéreo, por lo cual a pesar de ejecutar la aplicación en DirectX y con los modelos que ésta ocupa, se mostró la información de forma 2D.

CP5

La aplicación *Rich-Client* determinó que el modo adecuado para esa configuración era desplegar el modelo de datos con 3D pasivo (anaglifo). El módulo de transformación de datos transformó el modelo en formato *sdkmesh* a uno con formato obj. Una vez hecha esta transformación, el módulo de despliegue funcionó correctamente.

CP6

La aplicación *Rich-Client* determinó que el modo adecuado para esa configuración era desplegar el modelo de datos con 3D pasivo (anaglifo) y los módulos de transformación y de despliegue funcionaron correctamente.

CP7

La aplicación *Rich-Client* determinó que el modo adecuado para esa configuración era desplegar el modelo de datos con 3D pasivo (anaglifo) y los módulos de transformación y de despliegue funcionaron correctamente. En este caso, adicional a esta salida, la aplicación *Rich-Client* indica que el no haber desplegado la información de manera activa se puede deber a un error en los controladores.

CP8

La aplicación *Rich-Client* determinó que el equipo no contaba con las características suficientes para desplegar 3D activo y ocupó 3D pasivo. El módulo de transformación de datos convirtió el modelo de datos de un formato *sdkmesh* a un formato *obj*. El módulo de despliegue funcionó correctamente después de esta transformación.

CP9

La aplicación *Rich-Client* determinó que no se contaba con características ni para despliegue 3D activo, ni 3D pasivo ni 2D, por lo que buscó un equivalente en el repositorio de datos. Al tener el repositorio un "equivalente" al modelo (vista del modelo en formato imagen), éste se desplegó directamente en la aplicación *Rich-Client*.

CP10

La aplicación *Rich-Client* determinó que la manera idónea para desplegar el modelo solicitado era 3D activo. El módulo de despliegue mostró el modelo adecuadamente, para lo cual, el módulo de determinación de características creó previamente un archivo *png* en blanco. Al no contar con las dimensiones del modelo o información alguna para crear la textura, ésta no se ajustó correctamente a las dimensiones del modelo, por lo cual se percibió de manera no idónea.

CP11

Al no contar con un efecto en formato *fx*, el cual es necesario para el despliegue 3D activo en nuestra implementación y a pesar de tener todas las características de *software* y *hardware* necesarias para su implementación, la aplicación *Rich-Client* determinó que la manera adecuada para desplegar el modelo era 3D pasivo. El módulo de transformación funcionó correctamente, ya que el módulo de despliegue de información pudo mostrar correctamente el modelo del modo determinado.

Capítulo 7

Conclusiones y trabajo a futuro

En este capítulo, se presentan las conclusiones a las que se llegaron al realizar esta tesis y se dan algunas propuestas para continuar con el desarrollo de este trabajo.

7.1 Conclusiones

La forma en la que se visualiza la información, repercute en la interpretación de los datos y, dependiendo del modo de despliegue e incluso hasta de la manera en que se interactúa con estos. El éxito o fracaso de una aplicación puede depender de la forma en que se visualice su información y la manera en que se interactúa con ella.

Al desarrollar una aplicación que funcione en diferentes plataformas, la primera dificultad que se encuentra es la variedad de lenguajes de programación y herramientas de desarrollo que se pueden ocupar. Cada uno de estos cuenta con sus propias características y acepta diferentes tipos de datos como entrada al sistema. Al no tener una manera de convertir los recursos disponibles, a los recursos que ocupa una herramienta, se disminuyen las oportunidades de poder obtener el resultado esperado. La escalabilidad de la arquitectura y de otros sistemas adaptables depende de los métodos que se diseñen para transformar información, encontrar representaciones equivalentes e incluso se puede mejorar realizando un análisis de los datos, sabiendo el contenido y/o significado.

Las herramientas para desarrollo de sistemas de inmersión existentes requieren recursos de *hardware* y *software* específicos. El conocimiento de las características de un dispositivo es indispensable para saber el tipo de despliegue de información que puede ocuparse en el dispositivo.

Al ejecutar las pruebas se notó que los dispositivos con capacidad de despliegue de forma inmersa generalmente cuentan con pantallas grandes en las cuales se pueden presentar diferentes representaciones de la información al mismo tiempo. En algunos casos, más de un modo de despliegue son posibles y dependerá de la aplicación determinar el mejor o incluso decidir si son complementarios entre sí, como pueden ser representaciones en tres dimensiones de un modelo desplegado con algún tipo de forma 3D y las vistas principales (superior, lateral derecho y lateral izquierdo, por ejemplo) del mismo modelo.

El despliegue de modelos en 3D puede no ser inmediato, por lo que incluso en caso de no tener una forma de despliegue correspondiente a las características del dispositivo sede, algún

tipo de información debe de ser desplegada al usuario en forma de retroalimentación. Estos despliegues pueden ser en forma básica equivalente como se propuso en este trabajo, o algún tipo de alerta indicando los motivos por los cuales no pude desplegarse información. Otra opción de despliegue son sugerencias que ayuden al usuario a saber qué es lo que le falta para alcanzar algún tipo de despliegue que concuerde con la aplicación y su dispositivo. En caso de no presentar cambio alguno, el usuario puede llegar a pensar que se trata de un error o retraso en el despliegue de información.

Se probaron aplicaciones que ocupan diferentes tipos de datos, lo cual nos ayudó a notar cuales son más fácilmente ocupados para lograr adaptabilidad de la información. Los datos más simples son los más fácilmente adaptables en aplicaciones. En el caso del formato *obj*, fue el que más fácilmente pudo ser ocupado en las implementaciones, ya que no contiene características referentes a cada sistema o lenguaje con el que se utiliza y almacena únicamente los datos necesarios para representar objetos.

Para dar soporte a cada característica tomada en cuenta en este trabajo de tesis se ocuparon diferentes lenguajes y herramientas. Entre más características se intente abarcar en un sistema de adaptabilidad de la información, se tendrá una implementación más grande, a la cual se le debe brindar robustez en todos sus componentes para evitar fallos. Una buena solución a este problema es una segmentación adecuada de las aplicaciones. El uso de servidores de datos externos y proveedores de servicios son una buena alternativa para mantener las aplicaciones de un tamaño adecuado.

Las aplicaciones *Rich-Client* son una excelente forma de dar soporte a la adaptabilidad de información, al poderse ejecutar en cualquier dispositivo que cuente con un navegador Web, no depender en lo absoluto del dispositivo y poder acceder a las características más importantes del dispositivo, como son Sistema Operativo y dimensiones de la pantalla. Con el uso único de las aplicaciones *Rich-Client*, una aplicación puede decidir acertadamente el tipo de despliegue adecuado en un dispositiva. En nuestra implementación todo el peso de la decisión del modo de despliegue y obtención de características recae en la aplicación *Rich-Client* y en todos los casos probados la forma de despliegue decidida fue la adecuada. Las otras alternativas para detectar y modificar características, como es el caso de *nvapi*, son buena opción si se conocen al menos las características básicas del dispositivo.

La arquitectura propuesta en esta tesis da un buen acercamiento a la adaptabilidad de información y soporte a sistemas de inmersión. Funcionó adecuadamente como solución al problema específico que tratamos y puede ser fácilmente escalable a más lenguajes, Sistemas Operativos, hardware y demás componentes que influyan en el despliegue de información en dispositivos no abarcados en este trabajo.

Un factor importante a considerar, y probablemente el que más consumió trabajo y tiempo en esta tesis, fue lidiar con las incompatibilidades entre plataformas, por lo cual puede afirmarse que el camino a la adaptabilidad de información es aún muy largo, si no es que infinito.

7.2 Trabajo a futuro

Como se ha mencionado antes, el camino a la adaptabilidad de datos es aún muy largo. En esta tesis se mencionaron las características que deben de cumplirse para lograrlo y se trabajó con

sistemas de inmersión como primer acercamiento, pero dada la amplitud de esta propuesta, aún hay mucho trabajo por desarrollar. A continuación se mencionan algunas ideas que podrían servir para continuar este trabajo.

En cuanto a los dispositivos, se propone abarcar una gama más amplia, pudiendo empezar con dispositivos móviles que tienen gran demanda en la actualidad. Dispositivos como *Google Glasses* y relojes inteligentes, que son dispositivos interesantes al trabajar adaptando los tipos de datos a pantallas con tamaños pocas veces manejados en la actualidad, con lo cual se tendrá que pensar en diferentes formas (tal vez no visuales) de representar la información.

Por el momento, debido al tiempo y amplitud del proyecto, sólo se desarrollaron soluciones a este problema para desplegar datos en el Sistema Operativo Windows. En un futuro esta aplicación debería estar disponible para cualquier Sistema Operativo. También queda abierta la implementación para tarjetas gráficas y componentes dedicados de otras compañías diferentes a las ocupadas en esta tesis.

Los tipos de datos abarcados deben ser no únicamente modelos de datos, sino también multimedia o representaciones abstractas de información.

Se propone hacer un estudio más a fondo sobre la interpretación que los usuarios tienen de los datos en diferentes dispositivos de salida.

Se pueden determinar cómo influyen las características de contraste y resolución (ya obtenidas en nuestra implementación) a la forma en que el usuario percibe la información.

Es necesario agregar más equivalencias entre formatos al módulo de transformación de datos. Para el caso de despliegue equivalente, se propone generar un convertidor de modelos a algún formato de representaciones 3D en la Web, como es el vado de WebGL. Este tipo de formatos usualmente requieren de pocas características específicas en un dispositivo y su uso logra gran impacto en usuarios.

Una idea importante es incluir repositorios externos de datos, servidores de transformación de datos y de ejecución de cómputos. Esto ayudará a poder contar con más recursos de los que cuenta el dispositivo sede.

Apéndice A

Pruebas médicas

Las pruebas médicas ayudan a saber si un usuario cumple con los requerimientos de *hardware* y de *software* para desplegar contenido en 3D de manera inmersa. En otros casos también ayudan a conocer si el usuario puede percibir el 3D de manera correcta, de no ser así podría sufrir de molestias leves, como son dolores de cabeza hasta convulsiones.

Las opciones de la aplicación desarrollada ocupan las pruebas médicas establecidas por Nvidia, tanto en su versión de 3D activo como en la 3D pasivo con anaglifo. En este apartado no se incluyen imágenes de la versión para 3D activo ya que su captura de pantalla las almacena en 2D y no dan información relevante. Estas son las imágenes que ocupamos para la versión anaglifo.

Al mostrar la figura A.1 se le pide al usuario que se tape un ojo e indique la imagen que ve y luego haga lo mismo con su otro ojo. Esta prueba puede ayudarnos a saber si el usuario tiene o no un par de gafas del tipo anaglifo. La imagen A.2 muestra las posibles respuestas a la prueba médica uno.

Figura A.1: Imagen médica 1

Figura A.2: Imagen con las respuestas a la imagen médica 1

La siguiente prueba médica mostrada en la figura A.3, ayuda a saber si un usuario puede percibir de manera correcta el 3D pasivo. En caso de no ser así se le sugiere consultar con un médico antes de ocupar cualquier tipo de aplicaciones 3D. La figura A.4 muestra las posibles soluciones a esta prueba, siendo la opción izquierda la correcta.

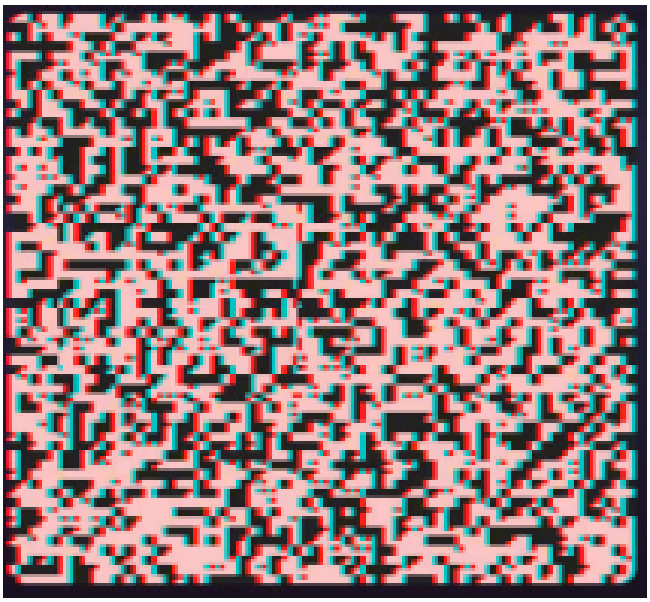


Figura A.3: Imagen médica 2

Figura A.4: Imagen con las posibles respuestas a la imagen médica 2

Apéndice B

Imágenes de la aplicación

En este apartado se muestran imágenes de las diferentes salidas que tiene la aplicación propuesta. Sólo se incluyen un par de imágenes de cada tipo de salida, ya que la forma en que pueden guardarse capturas de pantalla desde aplicaciones que ocupen 3D activo, es guardándolas en un formato específico. Este formato sólo es reproducible por aplicaciones compatibles con despliegue activo. Si se intenta obtener capturas de pantalla de otra manera, las imágenes se guardan en dos dimensiones, por lo cual no son relevantes como prueba del funcionamiento de la aplicación.

En la tabla B.1, se muestran imágenes de la aplicación activa con DirectX. La figura B.1 a) muestra un cubo con la textura creada por omisión, la figura B.1 b) muestra un cubo con una textura indicada por el usuario, la figura B.1 c) muestra otro modelo con una textura indicada por el usuario.

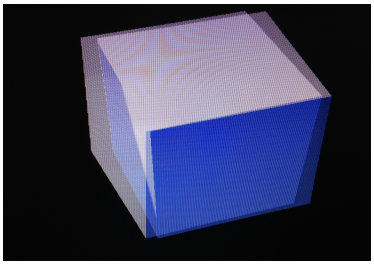
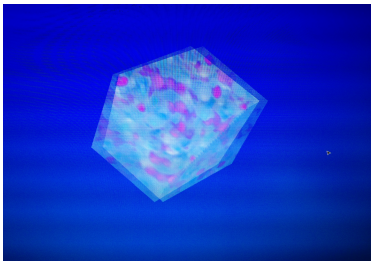
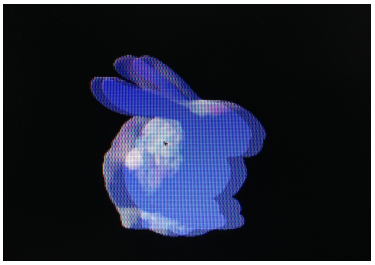
a)	b)	c)
		

Tabla B.1: Imágenes 3D activo en DirectX

En la tabla B.2, se muestran imágenes de la aplicación con DirectX en las cuales no se pudo activar el modo estéreo del dispositivo. La figura B.2 a) muestra un cubo con la textura creada por omisión, la figura B.2 b) muestra otro objeto con una textura indicada por el usuario, la figura B.2 c) muestra el mismo objeto de la figura anterior, pero con una textura creada por

omisión.

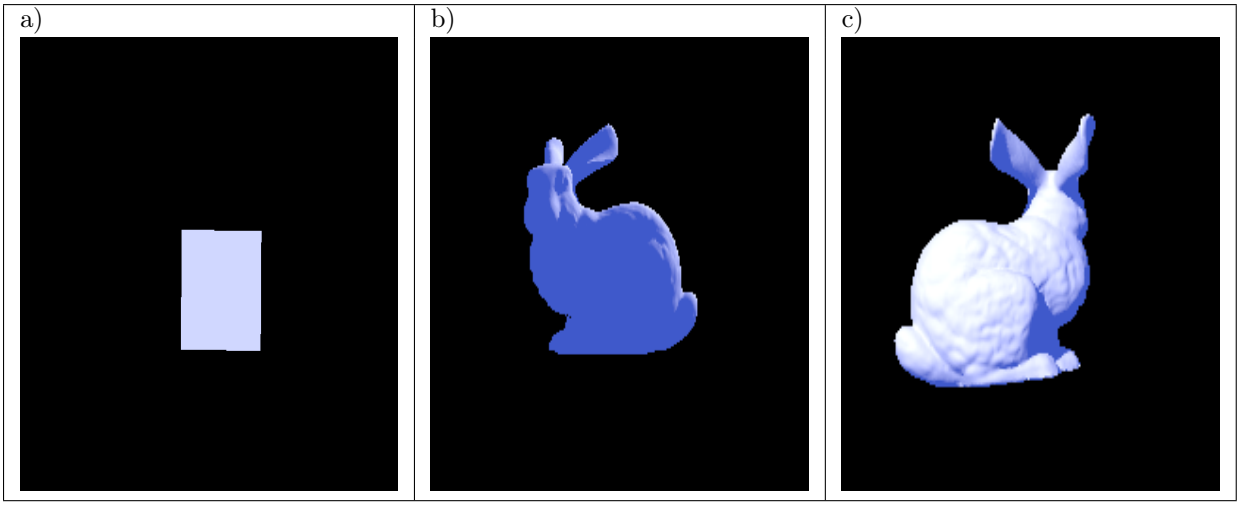


Tabla B.2: Imágenes 2D en DirectX

En la tabla B.3, se muestran imágenes de manera anaglifo en la aplicación con OpenGL. Las figuras B.3 a) y B.3 b) muestran un cubo con sus caras coloreadas mientras que la figura B.3 c) muestra sólo las aristas del cubo.

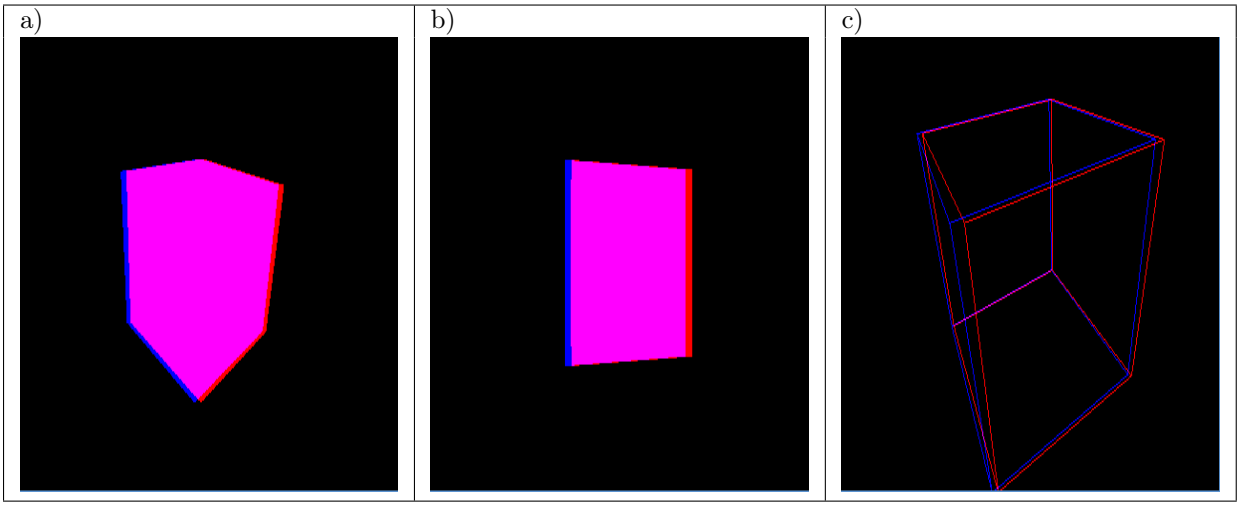


Tabla B.3: Imágenes 3D anaglifo en OpenGL

Bibliografía

[Ahrens et al., 2001] J. Ahrens, K. Brislawn, K. Martin, B. Geveci, C. C. Law, and M. Papka, *Large-Scale Data Visualization Using Parallel Data Streaming*, IEEE Computer Graphics and Applications, vol. 21, no. 4, pp. 34–41, July 2001.

[Alahmari et al., 2014] K. Alahmari, P. Sparto, G. Marchetti, M. Redfern, J. Furman, and S. Whitney, *Comparison of Virtual Reality Based Therapy With Customized Vestibular Physical Therapy for the Treatment of Vestibular Disorders*, IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 22, no. 2, pp. 389 – 399, March 2014.

[Arteaga et al., 2009] P. Arteaga, C. Batanero, C. Días, and J. M. Contreras, *El lenguaje de los gráficos estadísticos*, Revista Iberoamericana de Educación Matemática, vol. 1, no. 18, pp. 93–104, June 2009.

[Blas and Fernández, 2008] T. M. Blas and A. S. Fernández, *The role of new technologies in the learning process: Moodle as a teaching tool in Physics*, Computers and Education, vol. 52, no. 1, pp. 35–44, June 2008.

[Bowman and McMahan, 2007] D. Bowman and R. McMahan, *Virtual Reality: How Much Immersion Is Enough?*, IEEE Computer Society, vol. 40, no. 7, pp. 36–43, July 2007.

[Bowman et al., 2005] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev, *3D User Interfaces*, 3rd edn., Addison-Wesley, Boston MA (USA), March 2005.

[Brutzman and Daly, 2007] D. Brutzman and L. Daly, *X3D: 3D Graphics for Web Authors*, Series in interactive 3D technology, 1st edn., Morgan Kaufmann Publishers, San Francisco CA (USA), April 2007.

[Buss, 2003] S. R. Buss, *3-D Computer Graphics. A methematical introduction with openGL*, 1st edn., Press syndicate of the university of Cambrige, New York (USA), May 2003.

[Chi, 2000] E. H. Chi, *A Taxonomy of Visualization Techniques using the Data State Reference Model*, J. D. Mackinlay, S. F. Roth, and D. A. Keim (Eds.) *IEEE Symposium on Information Visualization, 2000. InfoVis 2000*, pp. 69–75, IEEE Computer Society, Salt Lake City UT (USA), 09-10 October 2000.

[Danchilla, 2012] B. Danchilla, *Beginning Webgl for HTML5*, 1st edn., Apress, New York (USA), September 2012.

[David, 2010] M. David, *HTML5 designing rich internet applications*, Visualizing the Web, 1st edn., Focal Press, BurlingtonMA (USA), July 2010.

[de Oliverira and da Rocha, 2006] R. de Oliverira and H. V. da Rocha, *Mobile Access to Web Systems Using a Multi-Device Interface Design Approach*, Proceedings of the 2006 International Conference on Pervasive Systems and Computing, June 2006.

[Engel, 2003] W. F. Engel, *Beginning direct3D game programming*, Course Technology, 2nd edn., Stacy L. Hiquet, 25 Thomson Place Boston Ma (USA), May 2003.

[Freemand and Robson, 2010] E. Freemand and E. Robson, *Head first HTML with css and XHML*, Head First, 1st edn., O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., April 2010.

[Freemand and Robson, 2011] E. Freemand and E. Robson, *Head first HTML5 Programming*, Head First, 1st edn., O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., October 2011.

[Gateau, 2010] S. Gateau, *The in and out: Making Games Play Rigth with stereoscopic 3D Technologies*, NVIDIA Corporation, 2701 San Tomas Expressway 2701 San Tomas Expressway, Santa Clara, CA, 1st edn., October 2010.

[Gateau and Nash, 2010] S. Gateau and S. Nash, *Implementing Stereoscopic 3D in your applications*, NVIDIA Corporation, 2701 San Tomas Expressway 2701 San Tomas Expressway, Santa Clara, CA, 1st edn., September 2010.

[Gavin Bell, 1995] M. P. Gavin Bell, Anthony Parisi, *The Virtual reality modeling lenguage Version 1.0 Specification*, November 1995.

[Healey and Enns, 2012] C. G. Healey and J. Enns, *Attention and Visual Memory in Visualization and Computer Graphics*, IEEE Transactions on Visualization and Computer Graphics, vol. 18, no. 7, pp. 1170 – 1188, July 2012.

[Hernández, 2011] I. M. T. Hernández, *Framework multiplataforma para reconocimiento de voz en aplicaciones open rich-client para dispositivos móviles*, Master's thesis, CINVESTAV-IPN, October 2011.

[Hernández et al., 2013a] I. M. T. Hernández, A. M. Viveros, and E. H. Rubio, *Analysis for the design of open applications on movile devices*, *2013 International Conference on Electronics, Communications and Computing (CONIELECOMP)*, pp. 126–131, IEEE, Puebla (México), October 10 2013a.

[Hernández et al., 2013b] I. M. T. Hernández, A. M. Viveros, E. H. Rubio, and B. C. Gamez, *Voice recognition framework for open rich-client movile applications*, *2013 10th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, pp. 302–306, IEEE, D.F (México), Septembre 30 - October 4 2013b.

[Juursalu et al.,] T. Juursalu, J. Uri, and V. Capitalists, *3D Thecnologies R and D*, URL `http://www.3dmlw.org`, June ????

[Kapferer and Riser, 2008] W. Kapferer and T. Riser, *Visualization needs and techniques for astrophysical simulations*, New Journal of Physiscs, vol. 10, no. 12, pp. 15–25, December 2008.

[Katz et al., 2011] N. Katz, T. Cook, and R. Smart, *Extending Web Browsers with a Unity 3D-based Virtual Worlds Viewer*, Internet Computing, vol. 15, no. 5, pp. 15–21, September/October 2011.

[Kelleher and Wagener, 2011] C. Kelleher and T. Wagener, *Ten guidelines for effective data visualization in scientific publications*, Environmental Modelling and Software, vol. 26, no. 6, pp. 822–827, June 2011.

[Kumar et al., 2012] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, *A Survey of Computation Offloading for Mobile Systems*, Mobile Networks and Applications, vol. 12, no. 1, pp. 129–140, April 2012.

[Machado and Tao, 2007] M. Machado and E. Tao, *Blackboard vs. Moodle: Comparing User Experience Blackboard vs. Moodle: Comparing User Experience of Learning Management Systems*, Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports, 2007 FIE '07 37th Annual, pp. S4J–7 – S4J–12, October 2007.

[Meier et al., 2009] J. Meier, A. Homer, D. Hill, J. Taylor, P. Bansode, L. Wall, R. B. Jr, and A. Bogawat, *Patterns and practices Application Architecture Guide 2.0*, Patterns and practices, 2nd edn., Microsoft Corporation, USA, November 2009.

[Moralez, 2009] G. S. Moralez, *Redistribución semi-plástica mixta: el caso de estudio de un pizarrón compartido*, Computer sciences, CINVESTAV-IPN, México D.F., January 2009.

[MWDeveloperCenter, 2010] M. W. MWDeveloperCenter, *Effect Format (Direct3D 10)*, web tutorial, May 2010.

[Ni et al., 2006] T. Ni, G. S. Schmidt, O. G. Staad, M. A. Livingston, R. Ball, and R. May, *A Survey of Large High-Resolution Display Technologies Techniques and Applications*, Vistual Reality Conference, pp. 223–236, March 2006.

[Nocent et al., 2012] O. Nocent, S. Piotin, A. Benassarou, M. Jaisson, and L. Lucas, *Toward an immersion platform for the World Wide Web using auto stereoscopic displays and tracking devices*, Proceedings of the 17th International Conference on 3D Web Technology, pp. 69–72, 2012.

[NVIDIA, 2010] NVIDIA, *NVIDIA 3D Vision pro and stereoscopic 3D*, NVIDIA Corporation, 2701 San Tomas Expressway 2701 San Tomas Expressway, Santa Clara, CA, 1st edn., October 2010.

[NVIDIA, 2011] NVIDIA, *Nvidia 3D Vision Streaming Suport for HTML5 and SilverLight*, NVIDIA Corporation, 2701 San Tomas Expressway 2701 San Tomas Expressway, Santa Clara, CA, 1st edn., May 2011.

[Ortiz, 2010] S. Ortiz, *Is 3D Finally Ready for the Web?*, Technology News IEEE, vol. 43, no. 1, pp. 14–16, January 2010.

[Preciado et al., 2007] J. Preciado, M. Linaje, S. Comai, and F. Sanchez-Figueroa, *Designing Rich Internet Applications with Web Engineering Methodologies*, 9th IEEE International Workshop on Web Site Evolution, pp. 23–30, October 2007.

[Raja et al., 2004] D. Raja, D. Bowman, J. Lucas, and C. North, *Exploring the benefits of immersion in abstract information visualization*, Proc Of IPT (Immersive Projection Technology), 2004.

[Salas, 2014] S. A. M. Salas, *Adaptability of Information*, *HCI 2014 International*, pp. 206–212, Crete Grece, June 2014.

[Schuchardt and Bowman, 2007] P. Schuchardt and D. A. Bowman, *The benefits of immersion for spatial understanding of complex underground cave systems*, Virtual Reality Software and Technology, vol. 1, pp. 121–124, November 2007.

[Shreiner et al., 2007] D. Shreiner, M. Woo, J. Neider, and T. Davis, *OpenGL Programming Guide, the official guide to learning OpenGL, version 2.1*, vol. 1, 6th edn., Addison-Wesley, 75 Arlington Sreet, suite 300, Boston MA 02116, July 2007.

[Tamayo, 2013] C. D. V. Tamayo, *Ray Tracing en tiempo real con GPGPUs*, Master's thesis, CINVESTAV-IPN, D.F (México), Diciembre 2013.

[Wright, 2007] H. Wright, *Introduction to Scientific Visualization*, vol. 1, 1st edn., Springer, 2007.

[Yi et al., 2007] J. S. Yi, Y. ah Kang, J. Stasko, and J. Jacko, *Toward a Deeper Understanding of the Role of Interaction in Information Visualization*, IEEE Transactions on Visualization and Computer Graphics, vol. 13, no. 6, pp. 1224–1231, November-December 2007.

[Yingling, 2006] M. Yingling, *MOBILE MOODLE*, Journal of Computing Sciences in Colleges, vol. 21, no. 6, pp. 280–281, June 2006.