Universidad Nacional Autónoma de México
Facultad de Ingeniería

Proyectos de Investigación y Desarrollo de la DECDFI

Tesina
Para obtener título de:
Ingeniero en Computación

Presenta
César Alejandro
Arias Canto

Asesor de tesis: M. I. Rodrigo
Guillermo Tintor Pérez

México, D.F. 2011

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Contenido

Índice de contenido

Índice de tabla

Índice de ilustraciones

Introducción

La División de Educación Continua y a Distancia de la Facultad de Ingeniería ubicada en el Palacio de Minería en el Centro Histórico de la Ciudad de México se caracteriza por brindar preparación impartiendo talleres, cursos, diplomados y seminarios de diversos temas de ingeniería.[1]

El objetivo de la DECD es preparar constantemente a los alumnos y profesores de la Facultad de Ingeniería así como al público en general debido al avance exponencial de la tecnología y a que cada día el conocimiento de las ciencias y la ingeniería aumentan a gran medida.

Cumpliendo con el objetivo, la División garantiza que los integrantes de la Facultad de Ingeniería están lo suficientemente preparados y actualizados para enfrentarse a los problemas que se presentan día a día en el ámbito laboral de las ingenierías.

La responsabilidad que recae en la División de Educación Continua de la Facultad de Ingeniería es la de reforzar y actualizar el conocimiento de los ingenieros para que puedan servir a la sociedad de la mejor manera posible.

Dentro de la DECD existe el Departamento de Investigación y Desarrollo que se encarga de crear distintos proyectos para mejorar la educación tanto en la División como en la Facultad. Estos proyectos van desde recorridos virtuales del Palacio de Minería, aplicaciones interactivas y laboratorios virtuales de clases impartidas en la Facultad.

Estos proyectos son realizados por los integrantes de este departamento, ex alumnos de la Facultad de Ingeniería con los conocimientos obtenidos a lo largo de la carrera de ingeniería en computación, a su vez que se amplían y se adquiere experiencia como parte de la formación profesional.

Los proyectos que han sido creados involucran una gran cantidad de programación, modelado geométrico y conocimiento de herramientas y tecnologías de inmersión virtual.

El objetivo de esta tesina es presentar los proyectos de investigación y desarrollo que desarrollé como mi labor en la DECDFI.

El objetivo de los proyectos en los que participé fue el de utilizar las nuevas herramientas y tecnologías de la información para mejorar la enseñanza y con ello aumentar el aprendizaje de los alumnos de la facultad y de la división. La importancia de estos proyectos radica en que día a día la división necesita explorar nuevas alternativas para potenciar la enseñanza.

Mi mayor motivación fue la de adquirir nuevos conocimientos y experiencia en el uso de herramientas para el desarrollo de aplicaciones interactivas o videojuegos educativos.

1. Resumen de Proyectos

En este capítulo se habla brevemente de los proyectos en los que estuve involucrado como parte de mi labor en el Departamento de ID de la DECDFI.

Los proyectos que describo son los siguientes:

- Herramienta de visualización de geometría analítica.
- Recorrido virtual del Salón de Actos del Palacio de Minería.
- Aplicación de videoconferencia.
- Práctica 1 del nuevo laboratorio de geometría analítica.

Debo mencionar que desarrollé dichos proyectos en el orden en el que se encuentran listados.

En los primeros 3 proyectos no utilicé ninguna metodología sin embargo en el desarrollo del último participé utilizando la metodología de desarrollo ágil de *software*: SCRUM.

1.1.Herramienta de Visualización de Geometría Analítica

La herramienta de visualización de geometría analítica fue el primer proyecto en el que participé recién ingresé a trabajar en el Departamento de Investigación y Desarrollo.

Es una aplicación que permite crear geometrías virtuales en tres dimensiones con la capacidad de modificarlas en tiempo real para su observación y análisis para asistir en el aprendizaje de la materia de geometría analítica.

Este proyecto tomó 6 meses en desarrollarse ya que es una misma aplicación orientada a 3 plataformas distintas y consta de una versión para PC, *Xbox 360* y otra para *Windows Phone 7*.

Este proyecto se describe detalladamente en el siguiente capítulo.

1.2.Recorrido Virtual del Salón de Actos del Palacio de Minería

Este proyecto estuvo un poco alejado de la programación ya que utilicé únicamente el motor de juegos *UDK* para cargar los modelos del Salón de Actos así como crear los materiales y hacer las animaciones requeridas para el paseo virtual, sin embargo *Unreal* contiene la programación por defecto para el movimiento dentro de los escenarios creados, por lo tanto el salón de actos también puede ser recorrido a placer del usuario.

Entre las características incluidas en este paseo o recorrido virtual se encuentran las siguientes:

- Recorrido inicial animado.
- Recorrido por parte del usuario con las teclas W, A, S y D del teclado para moverse y el ratón para mover la cámara.
- Iluminación avanzada, mapas de normales y especulares.
- Transición entre día y noche.
- Efecto de profundidad de campo conocido domo *Depth of Field*.
- Música de fondo tomada del contenido de *UDK*.

A continuación se encuentran una serie de imágenes del recorrido virtual animado, seguido de una del recorrido personal, posteriormente colocaré 2 imágenes relativas al motor de juegos *UDK* acerca del editor, la creación de materiales y la programación visual conocida como *visual scripting*.

Esta es la escena representativa del recorrido virtual, donde se puede apreciar, el águila, la mesa y las sillas del jurado.

Ilustración 1-1. Escena del recorrido virtual.

En la siguiente imagen se puede observar el Salón de Actos completo, así como el efecto de profundidad de campo utilizado tanto en el cine como en la mayoría de los videojuego actuales.

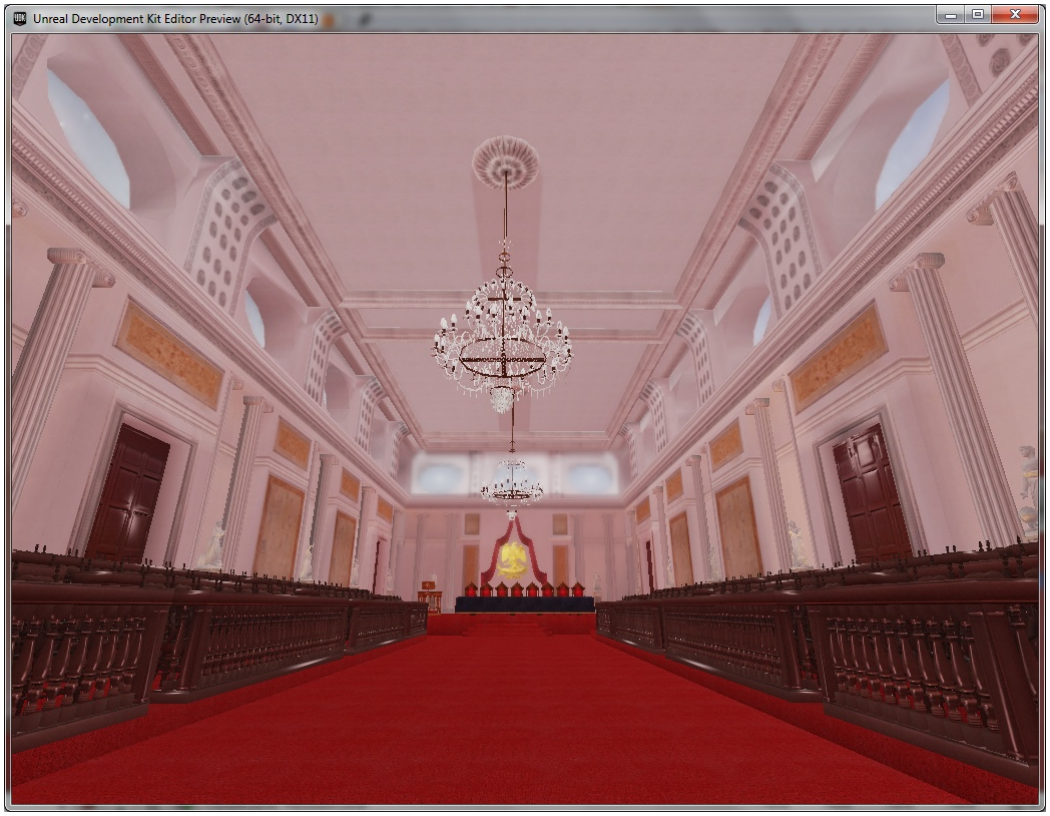

Ilustración 1-2. Escena del paseo virtual controlado por el usuario.

En la siguiente imagen se puede apreciar el uso de *UDK* para diseñar niveles para cualquier tipo de entorno virtual.

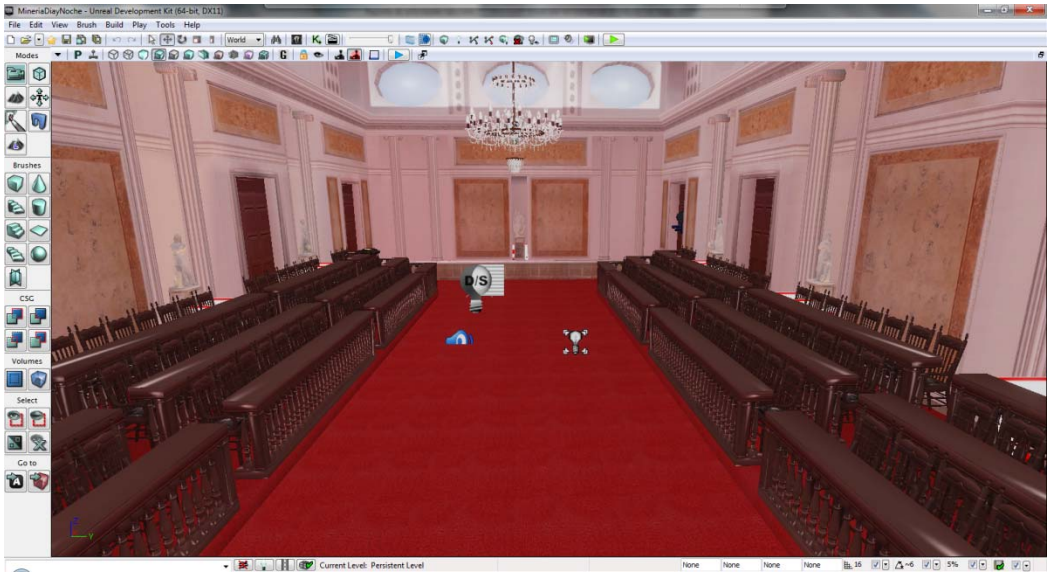

Ilustración 1-3. Interfaz del editor de niveles de UDK.

La siguiente imagen es una captura de pantalla de la programación visual que se utiliza en *UDK*.

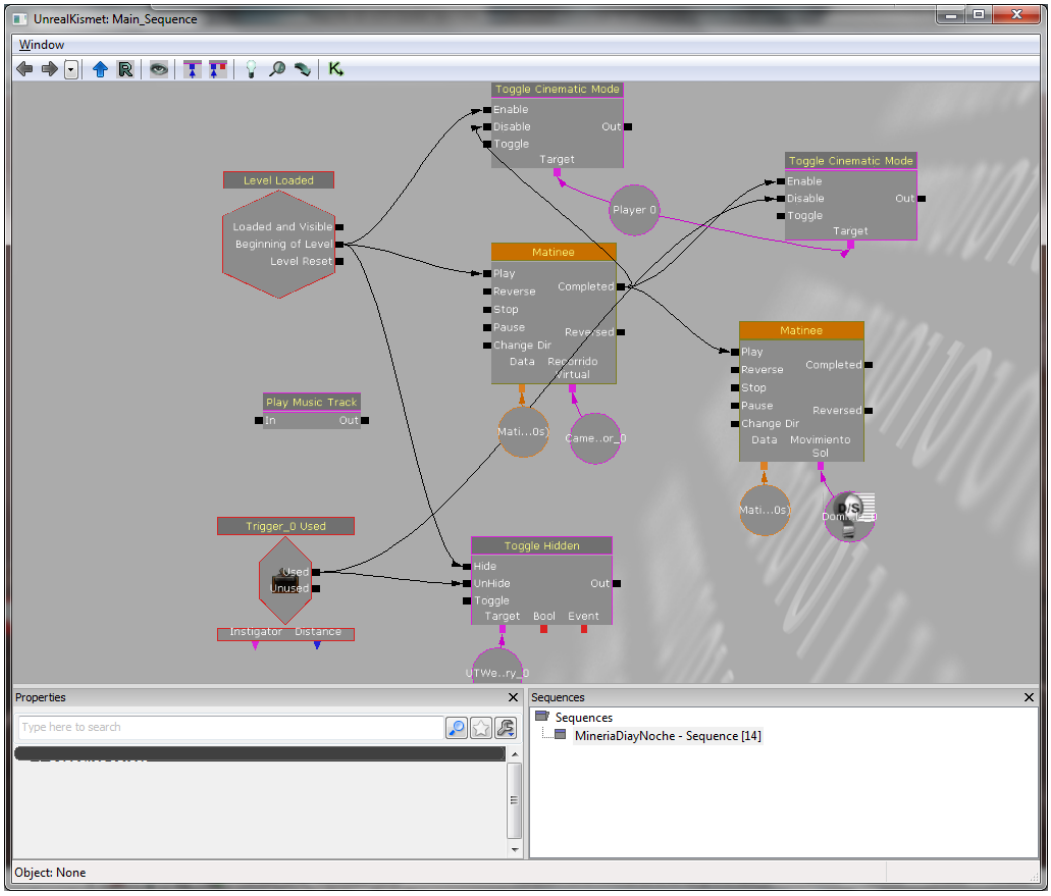

Ilustración 1-4. Programación visual de UDK.

Por último la siguiente imagen consta del editor de materiales de *UDK* en donde cree el material para el cielo que cambia entre día y noche.

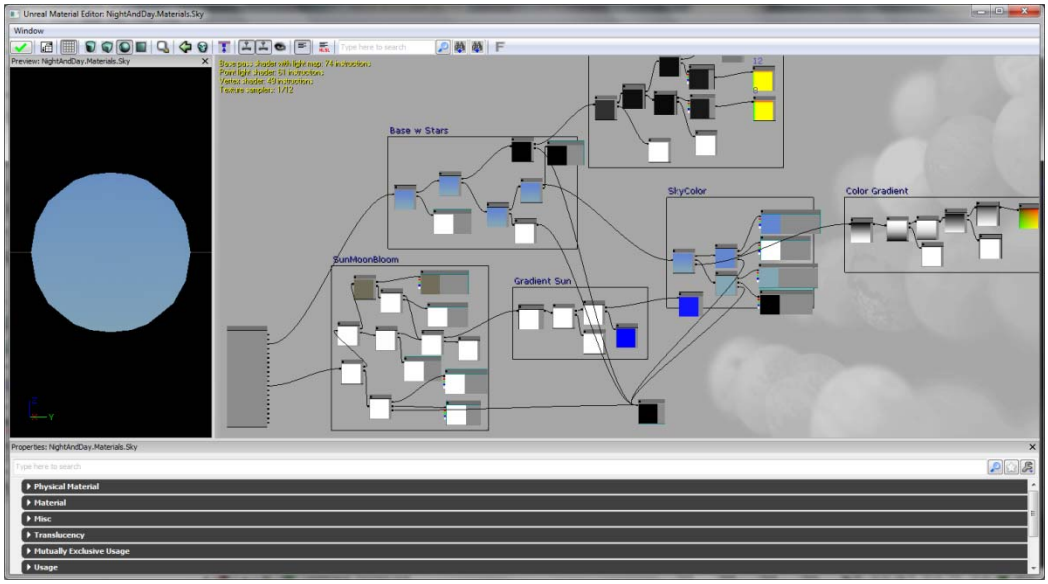

Ilustración 1-5. Editor de materiales de UDK.

El tiempo que me llevó elaborar este paseo virtual fue de 2 meses incluyendo la curva de aprendizaje ya que anteriormente había tomado un curso de desarrollo de videojuegos en Unreal Development Kit impartido por la División de Educación Continua y a Distancia.

1.3.Aplicación de videoconferencia

Una pequeña aplicación de video chat creada para una mayor aplicación de *Pizarrón Digital* que estaba siendo realizado por el Departamento de Investigación y Desarrollo antes de ser cancelado para darle prioridad a la nueva versión del laboratorio virtual de geometría.

Programé esta aplicación con *Flex* y *ActionScript* haciendo uso de un *script* en el mismo lenguaje encargado de realizar la conexión a un servidor *Flash Media Server*.

La aplicación soportaba 20 usuarios conectados simultáneamente con un video de buena calidad sin exigir demasiados recursos de red.

Como ya mencioné esta aplicación era parte de un pizarrón digital que podía ser ejecutado en cualquier navegador de internet.

Desarrollé esta aplicación en un periodo de 2 meses ya que tuve que aprender 2 nuevos lenguajes de programación simultáneamente además de entender el *script* de conexión al servidor *FMS*.

1.4.Programación del nuevo laboratorio de geometría analítica

Yo me encargué de la programación del contenido procedural para la primera práctica de la nueva versión del Laboratorio Virtual de Geometría Analítica en *Unity*.

Dicha práctica trata de suma de 2 vectores por el método del paralelogramo y del triángulo, resta de 2 vectores, producto punto y producto cruz entre 2 vectores, condición de perpendicularidad entre 2 vectores y por último cosenos directores de dos vectores distintos.

Este contenido fue programado en *C#* para el motor de juegos *Unity*.

Los *scripts* que realicé hacían las siguientes acciones dependiendo de la operación que se fuera a realizar.

- Suma de vectores:
    - Método del paralelogramo.

Modificaban la posición de los modelos de los 4 vectores que crean el paralelogramo y calculaban la posición inicial y final del vector resultante, ubicando en esas coordenadas geométricas el modelo de dicho vector.

    - Método del triángulo.

Manipulaba únicamente 2 vectores para formar el triángulo con el modelo del vector resultante cuyas coordenadas eran calculadas en tiempo real.

- Resta de vectores:

Modificaba la posición de los modelos de los vectores ingresados y calculaba la posición de las coordenadas del vector resultante.

- Producto punto:

Únicamente regresaba un escalar que se manejaba internamente, lo cual no tenía apreciación gráfica.

- Producto cruz:

Obtenía la posición inicial y final del vector resultante a esta operación y lo colocaba en el sistema cartesiano.

- Condición de perpendicularidad:

En caso de cumplirse la condición de perpendicularidad se dibujaba un indicador de ángulo recto entre los dos vectores.

- Cosenos directores:

Se calculaba internamente los cosenos directores de dos vectores ingresados, la suma de cosenos directores y los ángulos de los vectores con respecto a los ejes coordenados. Una vez obtenidos estos valores se calculaba y se dibujaba el ángulo entre ambos vectores y los ejes.

El desarrollo de la nueva versión del laboratorio virtual fue pausado hasta nuevo aviso ya que se ordenó dejarlo para utilizar la versión anterior.

La creación de contenido en tiempo real que programé para esta nueva versión demoró únicamente 1 mes gracias a dos elementos:

- Mi trabajo en el desarrollo de la herramienta de visualización de geometría analítica me permitió tener un buen manejo del lenguaje *C#*, el cual es utilizado por *Unity* como ya había mencionado, y la creación de gráficos por computadora creados y modificados en tiempo real, lo cual era el objetivo de ese proyecto.
- El trabajo con el motor de juegos *UDK* me permitió tener una curva de aprendizaje corta para entender el manejo del motor de juegos *Unity 3*. Sin embargo en *UDK* no programé directamente, hice lo que se conoce como programación visual, por lo que si bien ya manejaba la sintaxis de *C#*, tuve que aprender a programar con el lenguaje de programación en el motor de juegos.

Con esto he terminado de describir los 4 proyectos en los que trabajé en el Departamento de Investigación y desarrollo y puedo enfocarme a detallar el primer proyecto que es el de la Herramienta de Visualización de Geometría Analítica, el cual fue el primer proyecto en el que trabajé una vez que terminé la carrera y me enfrenté al mundo laboral.

2. Marco teórico

En este primer capítulo se presenta la información de las herramientas que utilicé para el desarrollo de los proyectos que mencioné en el capítulo anterior, en los que participé en mi labor en el Departamento de Investigación y Desarrollo de la División de Educación continua y a Distancia de la Facultad de Ingeniería.

La mayor parte de la información que conforma este capítulo trata sobre XNA, entorno de trabajo en el que desarrollé el proyecto principal, que es la Herramienta de Visualización de Geometría Analítica, con ello busco explicar de forma sencilla su funcionamiento general. También incluí información sobre las herramientas y librerías con las que desarrollé los proyectos de Videoconferencia, el Paseo Virtual y la práctica 1 del nuevo Laboratorio Virtual de Geometría Analítica.

2.1.XNA

XNA es el *framework* utilizado para desarrollar la *Herramienta de Visualización de Geometría Analítica*.

2.1.1. Definición de XNA

XNA es un *framework* o entorno de desarrollo creado por Microsoft que incluye una serie de librerías .NET de Microsoft que se utilizan para crear juegos. Otro elemento que forma parte de la solución es XNA Game Studio, el cual es una *API* que permite desarrollar juegos y aplicaciones utilizando el lenguaje *C#* y el paradigma de programación orientado a objetos para las siguientes plataformas[2]:

- Windows
- Xbox 360
- Zune
- Windows Phone

La razón por la que elegí XNA para el desarrollo de la herramienta de geometría analítica es la facilidad para desarrollar aplicaciones multiplataforma mediante un conjunto de API's o interfaces de programación de aplicaciones, normalizadas entre las plataformas mencionadas.

2.1.2. Definición de Framework y API

Un *framework* es un conjunto de códigos o librerías que tienen una funcionalidad común y amalgamada para una diversa cantidad de aplicaciones. Un *framework* ahorra el tiempo al no tener que escribir la lógica usada comúnmente en diferentes aplicaciones que utilicen funciones similares. La diferencia principal entre un *framework* y una librería es que ésta únicamente aporta una función específica y mientras que el *framework* ofrece varias.[3]

Una *Aplication Program Interface* (API) o Interfaz de Programación de Aplicaciones es un conjunto de reglas y especificaciones que un programa puede hacer uso de los servicios y recursos provistos por un programa particular que implementa esa API. Básicamente es una interfaz entre distintos programas de manera que facilita su interacción. Una API puede contener uno o varios *framework*.[4]

2.1.3. Definición de .NET

El *framework*.NET es la plataforma de Microsoft para programar aplicaciones que consiste en lo siguiente[5]:

- Rutinas de lenguaje común que proveen una capa de abstracción sobre el sistema operativo.
- Librerías de clases base o códigos pre construidos para tareas de programación de bajo nivel.
- *Framework* y tecnologías de desarrollo reusables y personalizables para tareas de programación mayores.

.NET se emplea en diversos *framework* y es compatible con la mayoría de los lenguajes de Microsoft como Visual Basic, C, C++, C#, etcétera.

Con ello tenía un *framework* suficientemente robusto para elaborar una aplicación educativa de calidad.

2.1.4. Estructura de XNA

La estructura general de XNA está dividida en las siguientes capas[6]:

- Plataforma

El nivel más bajo y consiste en interfaces de bajo nivel nativas y administradas sobre las cuales se construye el framework de XNA. Algunas interfaces son Direct3D 9(encargado de los gráficos en 3D), XACT (administra el audio y los efectos de sonido), XInput (maneja la entrada del usuario), XContent (administra la carga del contenido restante).

- Núcleo

La primera capa de XNA y provee la funcionalidad básica que requieren las demás capas. Las áreas de funcionalidad son Gráficos, Audio, Entrada, Matemáticas y Almacenaje.

- Extensión

En esta capa nos encontramos con el Modelo de Aplicación y la Tubería de Contenido (del inglés *Content Pipeline)*.

- Juegos

Esta es la capa superior y consiste del código y contenido de la aplicación o juego. Aquí se manejan las plantillas y los componentes de juego.

2.1.5. Características de XNA

Las características principales de XNA son las siguientes[6]:

- Modelo de aplicación

El propósito de dicho modelo es hacer que el programador se olvide de la plataforma en la que ejecutará su juego para así concentrarse en la concreta tarea de programar la aplicación. Así mismo el framework provee al desarrollador de un Componente de Gráficos que facilita la creación y manejo del Dispositivo Gráfico utilizado para la *renderización* tanto para la computadora, el Xbox y los dispositivos móviles. Por su parte el modelo de componentes permite agregar a la aplicación de una manera sencilla y rápida Componentes de Juego lo cual otorga al programador la capacidad de crear una librería de componentes que puede reutilizar en diversos proyectos.

- Gráficos

Los API's están basados en Direct3D 9, pero a diferencia de programar en su totalidad en Direct3D, uno de los beneficios es el uso de una tubería de renderización programable con *shaders* en lugar de una tubería de funcionalidad fija.

Un componente de XNA de suma importancia es el Efecto Básico o *BasicEffect*, el cual es un efecto de fácil uso que posee propiedades como luces, texturas, transparencia, etc. Utilizar el shader o efecto básico del framework permite a los programadores mostrar diversos elementos en pantalla sin tener que escribir complejos shaders.

- Audio

La interfaz de audio que utiliza XNA está realizada sobre XACT el cual es una API de Microsoft multiplataforma. La ventaja de utilizar XACT radica en que la herramienta permite crear paquetes de efectos de sonido con propiedades como volumen, bucles, canales, etc. Para posteriormente cargarlos y agregarlos al proyecto sin tener que inicializar los buffers o administrar los datos.

- Entrada

Las API's de Entrada están construidas a partir de XInput el cual es un API multiplataforma que funciona como driver del control de Xbox 360. La interfaz de Entrada ofrece un modo inmediato que no requiere inicialización, por lo que no hay que preocuparse por el reconocimiento o liberación del dispositivo, el modo para compartirlo, etc. Así mismo para la entrada específica por plataforma tenemos el teclado para Xbox y computadora, el mouse para computadora y para los dispositivos móviles existe la entrada táctil y los acelerómetros.

- Matemáticas

La interfaz de Matemáticas provee datos de programación de juegos como vectores en dos, tres y hasta cuatro dimensiones, matrices y planos. Dentro de la librería de las matrices podemos encontrar útiles métodos de creación de vistas, transformaciones, proyección, etc.

Ahora bien las características que ofrece XNA me permitieron realizar las geometrías sin más complicaciones que los cálculos matemáticos más óptimos para generarlas. Y realmente lo que fue más tardado de elaborar fue: la aplicación, el control táctil para la versión de Windows Phone 7 y el diseño.

Otro elemento importante que debo mencionar después de haber trabajado con XNA es el API de gráficos que utiliza el *framework*, el cuál es el más utilizado para crear gráficos en computadoras con sistemas Windows.

2.1.6. Definición de Direct3D

Direct3D de Microsoft es un API de gráficos de bajo nivel que permite manipular modelos visuales de objetos tridimensionales y hace uso de la aceleración por hardware, como las tarjetas de video.[7]

El término de aceleración por hardware básicamente es incrementar la velocidad de un proceso de software con la ayuda del hardware. Esto se logra al incluir parte del código de software en el hardware.

XNA utiliza Direct3D para crear gráficos mediante pequeños códigos o programas denominados *shader*.

2.1.7. Definición de Shader

Un *shader* es cualquier código escrito en un lenguaje de sombreado que se compila independientemente. La traducción literal de *shader* es "sombreador", la cual no se usa comúnmente, en cambio se usa la palabra efecto para describirlos debido al hecho que se utilizan principalmente para crear efectos especiales.[8]

Los lenguajes de sombreado son lenguajes de programación de alto nivel que se encargan de realizar cálculos de iluminación como indica su nombre. La principal ventaja de utilizarlos es que permiten la independencia de hardware, es decir liberan al procesador ya que se procesan en las tarjetas de video.

2.1.8. Definición de HLSL

El lenguaje de sombreado que utiliza XNA es *HLSL* o *High Level Shader Language* (Lenguaje de Sombreado de Alto Nivel) de Microsoft que utiliza *DirectX* también de Microsoft que es una implementación de Direct3D.[9]

Los tipos de *shader* que existen se ejecutan en 3 distintos procesadores que se encuentran en las tarjetas de video, estos shaders son:

- *Shader* de vértices. Realiza transformaciones en coordenadas, color, textura, vector normal, vector binormal, vector tangente, entre otros, de un vértice.
- *Shader* de geometrías. Encargado de generar dinámicamente o modificar primitivas geométricas o figuras geométricas básicas como puntos, líneas y triángulos.
- *Shader* de pixeles y fragmentos. El *pixel* es la unidad homogénea en color más pequeña de una imagen digital o *raster*. Es el que realiza transformaciones relacionadas con la profundidad, trabajar con las unidades de textura denominadas *texel*, calcular efectos de iluminación por pixel de alta precisión. Todos estos cálculos determinan el color que deberá ser aplicado a cada *pixel*.

Para concluir, de manera sencilla un shader es un código con un conjunto de instrucciones que serán ejecutadas por procesadores que contienen las tarjetas de video para obtener efectos en tiempo real y liberar al procesador de dichas operaciones y así se pueda encargar de otros procesos.

Debo aclarar que realicé dos versiones de la aplicación, la principal razón de esto fue que la versión casi terminada no era compatible con los dispositivos móviles, debido a la mayoría de los cambios aplicados a la sintaxis del lenguaje C# en la nueva versión de XNA, sin embargo esa no fue la única razón del pequeño contratiempo. La otra razón fue el hecho que se utilizaron *shaders* ajenos al efecto básico incluido en ambas versiones de XNA con las que se trabajó y debido a las restricciones de *hardware* de los dispositivos celulares y multimedia, se tuvo que trabajar con el efecto predeterminado.

Para continuar con la parte teórica de los lenguajes de sombreado es necesario definir la palabra *renderizar,* que viene del inglés *render*.

2.1.9. Definición de renderización

Renderizar o *rasterizar* es el proceso de convertir un modelo tridimensional generado por computadora en una imagen en el espacio de dos dimensiones. El resultado del proceso de renderizado se denomina *raster* o imagen digital.[1011]

Existen 2 tipos de renderizado:

- Pre-renderizado. Es el proceso computacionalmente más intensivo, se utiliza generalmente para películas y tiene como característica ser de muy alta calidad. Los cálculos son realizados por el procesador. Para la animación de escenas en 3 dimensiones, los movimientos son fijados antes de la renderización y se calculan durante ella.
- Renderizado en tiempo real. Utilizado en juegos de video. Se procesa en las tarjetas de video o aceleración 3D. A diferencia del otro método, los cambios se calculan en tiempo real.

2.1.10. XNA Game Studio

Si bien para elaborar la aplicación de geometría programé en XNA, un componente importante de la solución es XNA *Game Studio* que se define como:

"Entorno de programación que permite usar Visual Studio para crear juegos para Windows Phone, la consola Xbox 360 y equipos basados en Windows. XNA Game Studio incluye XNA Framework, que es un conjunto de bibliotecas administradas diseñadas para el desarrollo de juegos basado en Microsoft .NET Framework 2.0" – Definición de la página oficial de XNA Game Studio 4.0 en MSDN en español.[12]

La versión de XNA Game Studio 4.0 es la más reciente pero la versión 3.1 aún se encuentra vigente.

Las aplicaciones que son ejecutadas en los celulares con Windows Phone 7 son las creadas en la versión 4.0. Ahora bien estas aplicaciones también son ejecutadas en Windows y en la Xbox 360 así como las aplicaciones de XNA Game Studio 3.1. Los dispositivos multimedia Zune únicamente ejecutan programas creados en la versión 3.1

La totalidad de versiones que han existido son[13]:

- XNA Game Studio Express
- XNA Game Studio 2.0
- XNA Game Studio 3.0
- XNA Game Studio 3.1
- XNA Game Studio 4.0
- XNA Game Studio 4.0 Refresh

2.1.10.1. Requisitos de Sistema

Para comenzar con la programación de la aplicación se tuvieron que cumplir los siguientes requisitos:

- Windows XP (con la excepción que no se pueden desarrollar aplicaciones para Windows Phone 7).
- Windows Vista (el cual debe ser cuando menos Service Pack 1).
- Windows 7

Es importante mencionar que en Windows Vista y 7 se requieren tener permisos de administrador para poder instalar el entorno.

Los requisitos de hardware para ejecutar y programar juegos de XNA Framework se requiere mínimo una tarjeta de video que soporte *Shader Model 1.1* y *DirectX 9.0c* pero se recomienda que la tarjeta soporte *DirectX 10* para poder programar aplicaciones para Windows Phone.[14]

El software que se requiere para poder programar en XNA Game Studio 4.0 es cualquiera de las versiones de *Microsoft Visual Studio 2010*:

- Visual Studio 2010 Express for Windows Phone.
- Visual Studio 2010 Ultimate
- Visual Studio 2010 Premium
- Visual Studio 2010 Professional
- Visual C# 2010 Express

Para programar aplicaciones con la versión anterior, es decir XNA Game Studio 3.1 es cualquiera de las versiones de *Microsoft Visual Studio 2008*:

- Visual C# 2008 Express Edition
- Visual Studio 2008 Standar Edition
- Visual Studio 2008 Professional Edition

Los requisitos mínimos de hardware son:

Computadora con sistema operativo Windows con tarjeta aceleradora de gráficos que soporte como mínimo Shader Model 1.1 y DirectX *9*.

Se recomienda una tarjeta aceleradora de gráficos que soporte el modelo de shader 2.0 (*Shader Model 2.0*).

Para ejecutar y depurar aplicaciones de Windows Phone 7 con el emulador, se requiere una tarjeta aceleradora de video que soporte como mínimo *DirectX 11*.

2.1.10.2. Herramientas empleadas

Para elaborar la herramienta de superficies geométricas hice uso de los siguientes elementos:

- Sistema operativo Windows 7 de 64 bits con permisos de administrador.
- Visual Studio Professional 2008 y 2010, para la versión de Xbox360 / PC y Windows Phone 7 respectivamente.
- XNA Game Studio 3.1 para la versión de Xbox360 / PC y 4.0 para programar para Windows Phone 7.
- Photoshop CS5 para editar las imágenes.
- Crazy Bump para obtener los mapas de normales y mapas de especularidad utilizadas por los *shaders* en la versión de XNA Game Studio 3.1.
- Softimage que es un software de modelado 3D.
- Software multimedia Zune para transferir la aplicación al celular con Windows Phone 7. Una gran ventaja es que estando conectado permite la ejecución en modo de depuración de errores.
- Software XNA Game Studio Connect para la consola Xbox 360 para poder transferir la aplicación a la consola, instalarla y ejecutarla en modo de depuración de errores.

Otros elementos que considero de gran importancia para la elaboración de ambas aplicaciones son:

- Consola de videojuegos Xbox 360.
- Control alámbrico de Xbox 360 / Windows para realizar pruebas tanto en la consola como en la computadora.
- Celular con sistema operativo Windows Phone 7 para realizar pruebas.
- Cuenta de estudiante en la página de internet de MSDN para obtener acceso a la licencia de Visual Studio Professional y a la licencia de desarrollador de XNA Game Studio.
- Licencia de desarrollador de juegos con XNA para Xbox 360, PC, Windows Phone 7 y Zune.

2.1.10.3. Perfiles de XNA

Además de los cambios obvios de sintaxis que se realizaron entre las versiones de XNA, un aspecto muy importante que se añadió al *framework*, es la posibilidad de modificar la configuración del proyecto desde las propiedades para seleccionar uno de dos perfiles de juego[15]:

- *Reach*. Para dispositivos móviles. Alcance.
- *HiDef*. Para Xbox 360 y computadora. Alta Definición.

A continuación listo las principales diferencias entre estos dos perfiles de aplicación.

Tabla 2-1. Tabla comparativa de perfiles de XNA 4.0

	Reach	HiDef
Plataformas Soportadas	Windows Phone 7, Xbox 360, cualquier computadora con Windows y una tarjeta aceleradora de video que soporte DirectX 9 con *shader model* 2.0.	Xbox 360, cualquier computadora con Windows y una tarjeta aceleradora de video que soporte DirectX 10.
Modelo de Shader (*Shader Model*)	2.0. Windows Phone no soporta *shaders* personalizados.	3.0 en adelante. (Xbox 360 maneja *shaders* con tipo vfetch pero no son soportados por Windows.
Tamaño máximo de textura	2048 pixeles	4096 pixeles
Tamaño máximo de mapa cúbico (*cubemap*)	512 pixeles	4096 pixeles
Tamaño máximo de textura volumétrica	No hay.	256 pixeles
Texturas no cuadradas	Si, sin embargo no se puede utilizar *wrap*, *mipmaps*, o compresión DXT.	Sí
Cubemaps no cuadrados	No	Sí
Texturas volumétricas no cuadradas	No	Sí
Primitivas máximas por llamada de dibujado	65535	1048575
Formatos de buffer de índices	16 bits	16 y 32 bits
Formatos de elementos de vértices	Color, Byte4, Single, Vector2, Vector3, Vector4, Short2, Short4, NormalizedShort2,	Todos los formatos de Reach más *HalfVector2* y *HalfVector4.*

	NormalizedShort4	
Formatos de textura	Color, Bgr565, Brga4444, NormalyzedByte2, NormalizedByte4, Dxt1, Dxt3, Dxt5	Todos los formatos de Reach más *Alpha8, Rg32,* Rgba64, Rgba1010102, Single, Vector2, Vector4, HalfSingle, HalfVector2, HalfVector4. Las texturas de punto flotante no soportan filtrado.
Formatos de vértices texturizados	Texturizado de vértices no está soportado	Single, Vector2, Vector4, HalfSingle, HalfVector2, HalfVector4
Formatos de objetivos de rasterizado	Variable	Variable
Objetivos de rasterizado múltiples	No	Hasta 4. Deben de tener la misma profundidad en bits. Soporta mezclado de transparencia y máscaras de escritura independientes por objetivo.
Consulta de oclusión	No	Sí
Mezcla de transparencia separada	No	Sí
Blend.SourceAlphaSaturation	Solo para SourceBlend y no DestinationBlend	Sí
Flujo de vértices máximo (*vertex streams*)	16	16
Alcance máximo del flujo (*stream stride*)	255	255

HiDef hereda de *Reach* por lo tanto, si se corre una aplicación de este último perfil en una plataforma de alta definición el *framework* seguirá aplicando las reglas de *Reach*.

El perfil de alta definición no utiliza DirectX10 directamente sino que requiere una tarjeta aceleradora gráfica con memoria de video y procesador de gráficos superior a una que soporte DirectX9.

Estos perfiles se configuran en las propiedades del proyecto dentro de Visual Studio y se explicará la manera en la que se realizó la configuración.

2.2.Unreal Development Kit

Unreal Development Kit es la versión gratuita del poderoso motor de juegos Unreal Engine 3, como dice el nombre es la tercera iteración del motor Unreal Engine.[16]

Este es un framework completo de desarrollo profesional de videojuegos y contiene todas las herramientas que se puedan necesitar para crear juegos, simulaciones, recorridos virtuales y demás aplicaciones virtuales para la PC y dispositivos móviles como *iPhone* o *iPad*.

El objetivo principal de *UDK* permitir el uso gratuito de *UnrealEngine3* para crear contenido por estudiantes, desarrolladores independientes, universidades, investigadores, etc.

Esta gran herramienta es actualizada mes con mes con el fin agregar nuevos elementos, mejorar los existentes y remover los que por estándares de la industria se dejan de utilizar.

Siendo un motor de juegos cuenta con todo lo necesario para desarrollar un videojuego sin mayor esfuerzo. Entre las características que incluye están:

- *Unreal Editor*. Un ambiente muy completo de edición de ambientes o niveles.
- *Unreal Content Browser*. Un navegador de contenido, que puede ser: texturas, modelos, música, etc.
- *Unreal Matine*. Animación artística de los modelos así como cinemáticos y escenas visualmente impactantes.
- *Unreal Script*, el cual es un lenguaje de programación de alto nivel.
- Física real.
- Iluminación y sombreado avanzado.
- Creación de terreno.
- Conectividad *LAN* y por *IP*.
- *Shaders* en tiempo real.
- Audio tridimensional y soporte de gran cantidad de formatos.
- Efectos de partículas.
- Inteligencia artificial.
- Ambientes destruibles.

2.3.Flex y ActionScript

Flex es un *framework* gratuito de *Adobe* utilizado en la creación de aplicaciones web tanto para sistemas operativos de computadoras, como para dispositivos móviles, por lo tanto tiene la ventaja de ser soportado en varias plataformas, reduciendo el tiempo de elaboración de dichas aplicaciones.[17]

ActionScript es un lenguaje de programación orientado a objetos de *Adobe* para *Flash* también de *Adobe*. Actualmente es utilizado en *Adobe Flash* y *Flex*.[18]

2.3.1. Flash Media Server

Flash Media Server es un servidor de medios de *Adobe*, el cual permite almacenar y ejecutar aplicaciones realizadas en otras plataformas de *Adobe*, estas plataformas pueden ser *Flash* y *Flex* entre otras.[19]

De igual manera este servidor permite la transmisión de audio y video en tiempo real por lo que puede ser utilizado para aplicaciones de videoconferencia o video chat así como chat sencillo entre otras.

2.4.Unity 3

Unity 3 es una herramienta de desarrollo de videojuegos muy completa que no exige demasiados recursos de *hardware*. Si bien es un motor de juegos utilizado para crear videojuegos y aplicaciones interactivas al igual que otros productos, este se caracteriza por ser más sencillo.[20]

Esta herramienta permite desarrollar videojuegos para PC, consolas de videojuegos, dispositivos móviles con sistema operativo *iOS*, *Android*, así como a diferencia de otros maneja un reproductor web que permite ejecutar las aplicaciones directamente en los navegadores de internet.

Las características de Unity son:

- Crear y editar niveles.
- Cambio instantáneo para desarrollo entre distintas plataformas.
- Método de iluminación mediante el mapeado de luces y oclusión de caras.
- *Scripting* con *C#* y *JavaScript*.
- Inteligencia artificial.
- Física real.
- Conectividad de red.
- Creación de terreno.
- Shaders en tiempo real.

La desventaja que presenta utilizar *Unity* es que la versión gratuita tiene desactivadas varias características, incluyendo la más remarcable, el sombreado.

3. Desarrollo del proyecto: Herramienta de Visualización de Geometría Analítica.

El principal aporte en cuanto a *software* de la Coordinación de Investigación y Desarrollo de la División de Educación Continua y a Distancia de la Facultad de Ingeniería es el Laboratorio Virtual de Geometría Analítica ya que ha logrado una mejora en el aprendizaje de dicha materia.

Dicho proyecto resulta de gran importancia para la comunidad estudiantil de la facultad, tanto para alumnos como para profesores, ya que aporta una gran mejoría del aprendizaje de la asignatura de Geometría Analítica impartida durante el primer semestre de todas las ingenierías debido a que es una materia de tronco común.

Si bien a lo largo de los siglos se han utilizado el papel y la pluma como principal herramienta de educación, como ingenieros en computación y desarrolladores de software, debemos de aprovechar la tecnología y las herramientas a nuestro alcance para darle un mayor aprovechamiento al aprendizaje de las primeras materias que servirán como cimiento a los futuros ingenieros de nuestro país, logrando con ello obtener un mejor desempeño y una adquisición de conocimiento básico de gran calidad.

Por lo tanto decidí que sería ideal crear una aplicación que complementara el laboratorio virtual de geometría analítica, dicho programa fue el primer proyecto que desarrollé en mi vida profesional.

El objetivo de dicho proyecto es:

Realizar una aplicación con control intuitivo que permita la visualización y modificación en tiempo real de distintas geometrías.

Las plataformas elegidas para su ejecución son la consola de videojuegos *Xbox 360*, la computadora y celulares con *Windows Phone 7*.

El desarrollo de la aplicación de geometría analítica para la Xbox y computadora se realizó en 2 fases, las cuales tuvieron una mejora al momento de migrar la aplicación y crear ciertos elementos específicos de la plataforma móvil Windows Phone 7.

La primera fase durante la elaboración de la aplicación fue programar las geometrías mientras que la segunda fase fue crear la aplicación como tal.

Se consideraron las geometrías más representativas para su programación y por lo tanto su visualización tridimensional en la consola y los celulares.

Dicha selección está compuesta por:

- Cilindros
- Conos
- Superficies de revolución

La lista de geometrías que programé, pertenecientes a las familias mencionadas, es la siguiente:

- Cilindro
  - Circular Recto
  - Elíptico
  - Parabólico
  - Hiperbólico
- Cono
  - Circular
  - Elíptico
  - Parabólico
  - Hiperbólico
- Superficie de Revolución
  - Esfera
  - Elipsoide

A pesar de haber varias geometrías, las que llegaron a la versión final de la aplicación con propósito demostrativo son los cilindros.

La manera en que programé las clases de las geometrías facilita crear la geometría deseada y cambiar los valores de sus propiedades en tiempo real, dando una gran capacidad de visualización, parte del objetivo del software creado. Así mismo de este modo se garantiza que se ahorran recursos ya que únicamente se construye y dibuja la figura geométrica elegida.

Las plataformas para las que se realizaría esta aplicación no permiten el uso de una misma versión ya que como menciono en el marco teórico de *XNA* la versión 3.1 permite crear juegos para *Xbox 360*, PC, mientras que la versión 4.0 es la única en la que se pueden realizar aplicaciones para *Windows Phone*.

Por lo tanto se crearon dos versiones de esta aplicación, compartiendo el mismo objetivo, pero con un diseño diferente.

Retomando el tema de las geometrías que programé, estas se encuentran en las dos versiones y realicé pruebas de estrés de ambas para asegurar su correcto funcionamiento, sin embargo por cuestiones de tiempo ya que el dispositivo celular en el que realizaríamos las pruebas

fue prestado por Microsoft, las únicas geometrías que se visualizan en el producto final son los cilindros.

La prueba de estrés que realicé a cada geometría fue necesaria para que supiera que al momento crearlas en la aplicación final no iba a haber ningún error o *bug* como se conoce, y con ello tener un programa de calidad. A continuación coloco impresiones de pantalla de las geometrías en ambas versiones de XNA.

3.1.Geometrías en XNA 3.1

Las imágenes que se encuentra en la siguiente sección pertenecen a la versión de prueba de *XNA* 3.1 y contienen cada una de las geometrías y los ejes coordenados.

La razón por las que las coloco es mostrarlas al lector ya que por cuestión de tiempo y para cumplir el objetivo que era demostrar el uso de la tecnología para apoyar el aprendizaje, las geometrías cónicas y superficies de revolución no llegaron a la versión final de la aplicación.

3.1.1. Cilindro Circular Recto

Este es un cilindro conformada por un círculo como directriz y una generatriz que gira entorno a dicho círculo.

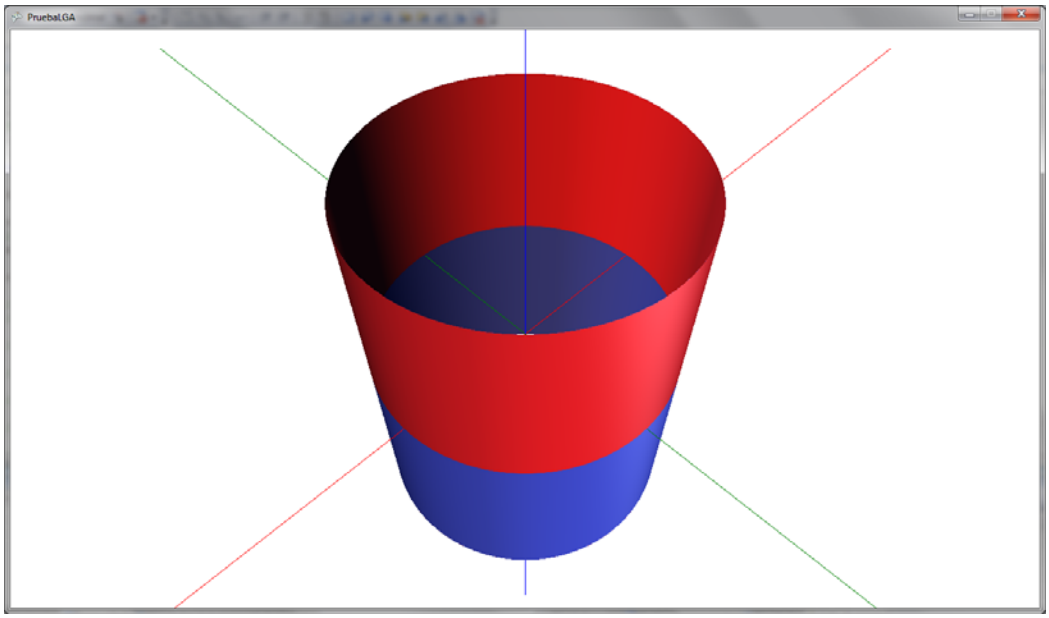

Ilustración 3-1. Cilindro circular recto en XNA 3.1.

3.1.2. Cilindro Elíptico.

Esta geometría está formada por una directriz elíptica y la generatriz que se mueve en torno a esa curva.

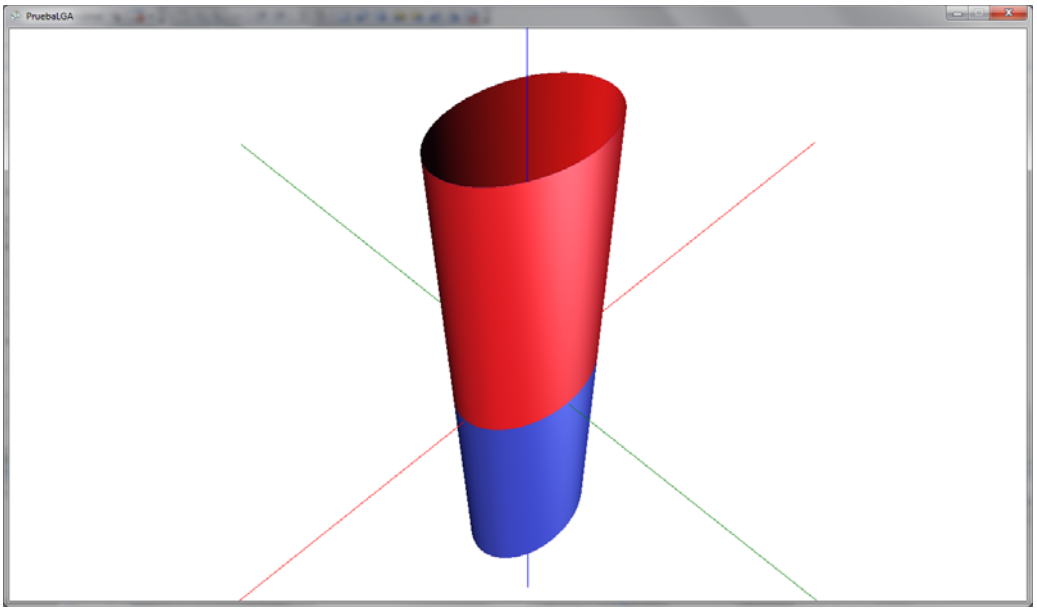

Ilustración 3-2. Cilindro elíptico en XNA 3.1.

3.1.3. Cilindro Parabólico

Geometría conformada por una parábola como directriz y la generatriz que la recorre.

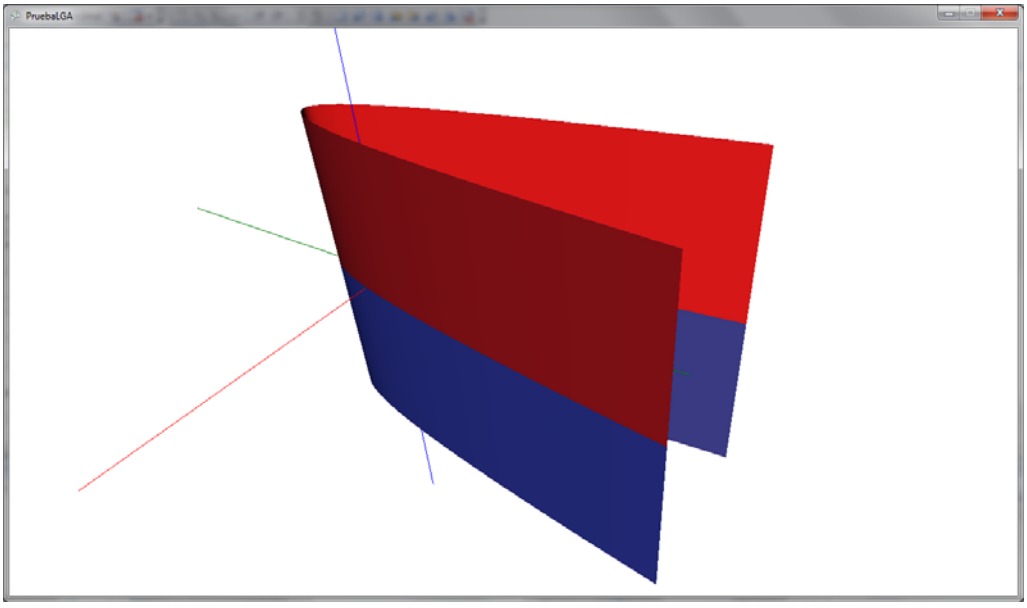

Ilustración 3-3. Cilindro parabólico en XNA 3.1.

3.1.4. Cilindro Hiperbólico

Este es la última geometría de la familia de los cilindros y construida por una directriz hiperbólica y su respectiva generatriz.

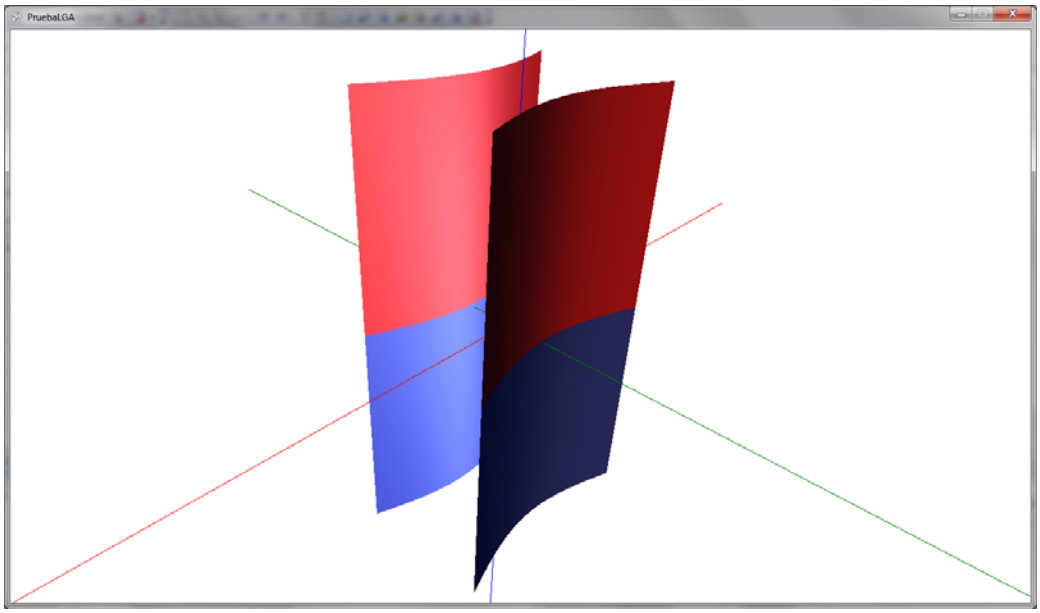

Ilustración 3-4. Cilindro hiperbólico en XNA 3.1.

3.1.5. Cono Circular

Imagen de un cono formado por un círculo por directriz y una generatriz en diagonal.

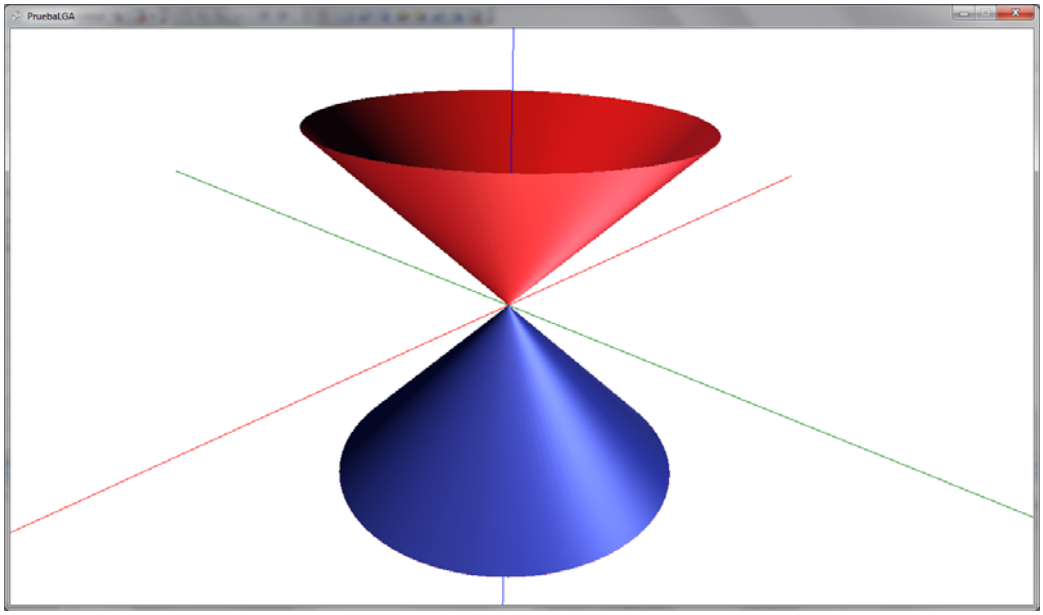

Ilustración 3-5. Cono circular en XNA 3.1.

3.1.6. Cono Elíptico

Cono conformado por una directriz elíptica.

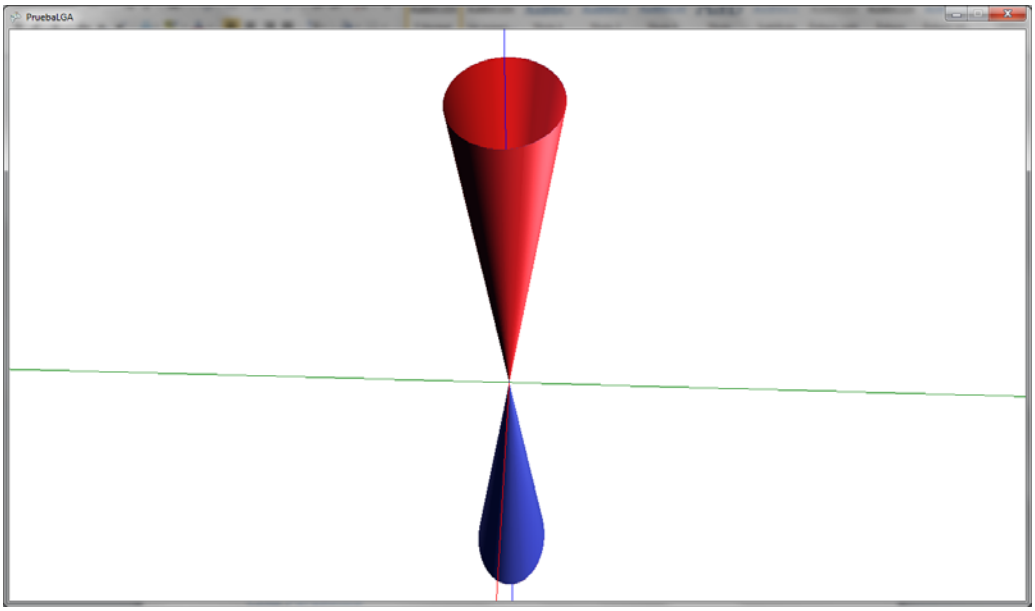

Ilustración 3-6. Cono elíptico en XNA 3.1.

3.1.7. Cono Parabólico

Cono que como bien indica su nombre está formado por una parábola.

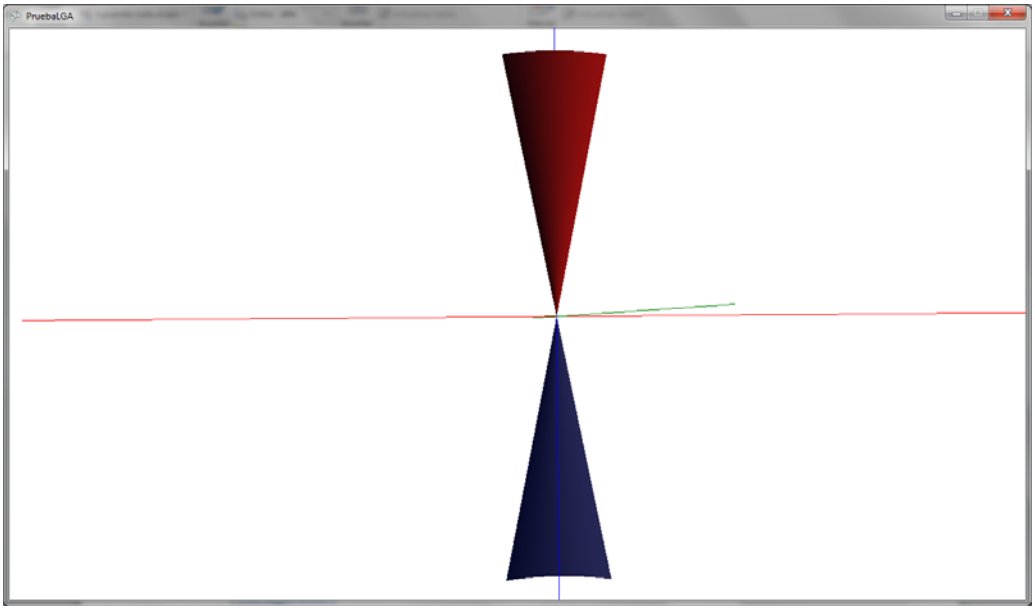

Ilustración 3-7. Cono parabólico en XNA 3.1.

3.1.8. Cono Hiperbólico

Captura de pantalla del cono hiperbólico cuya generatriz es una hipérbola.

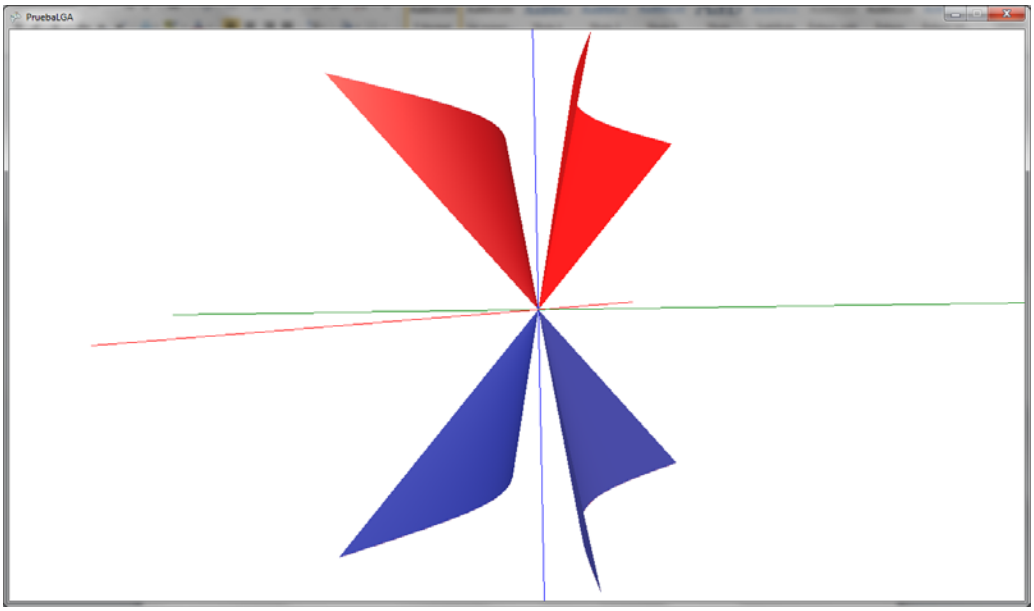

Ilustración 3-8. Cono hiperbólico en XNA 3.1.

3.1.9. Esfera

A continuación coloco una imagen de la superficie de revolución más famosa de todas.

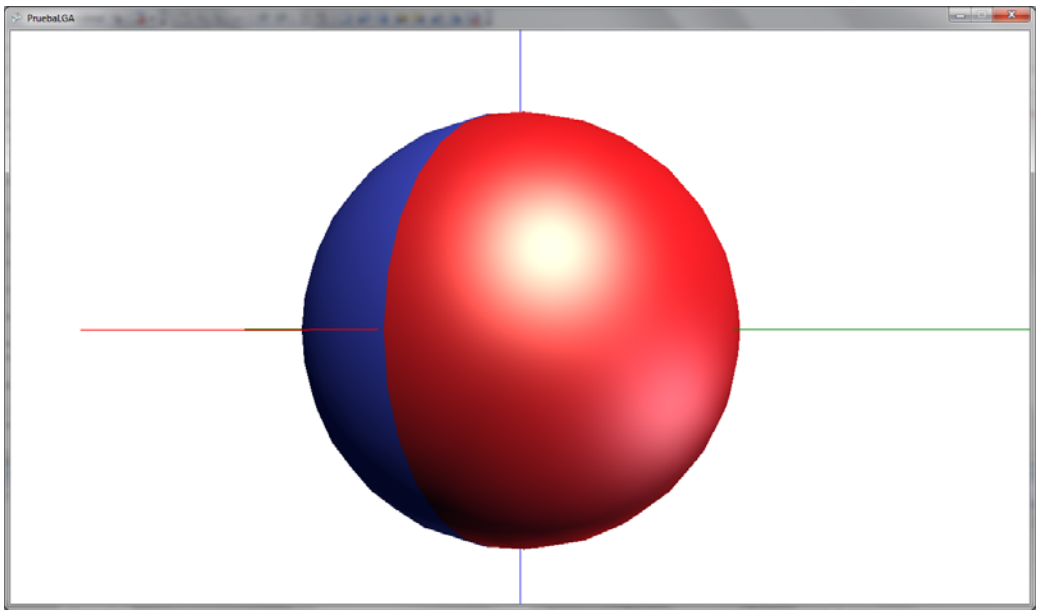

Ilustración 3-9. Esfera en XNA 3.1.

3.1.10. Elipsoide

Esta es la última geometría que programé para la aplicación.

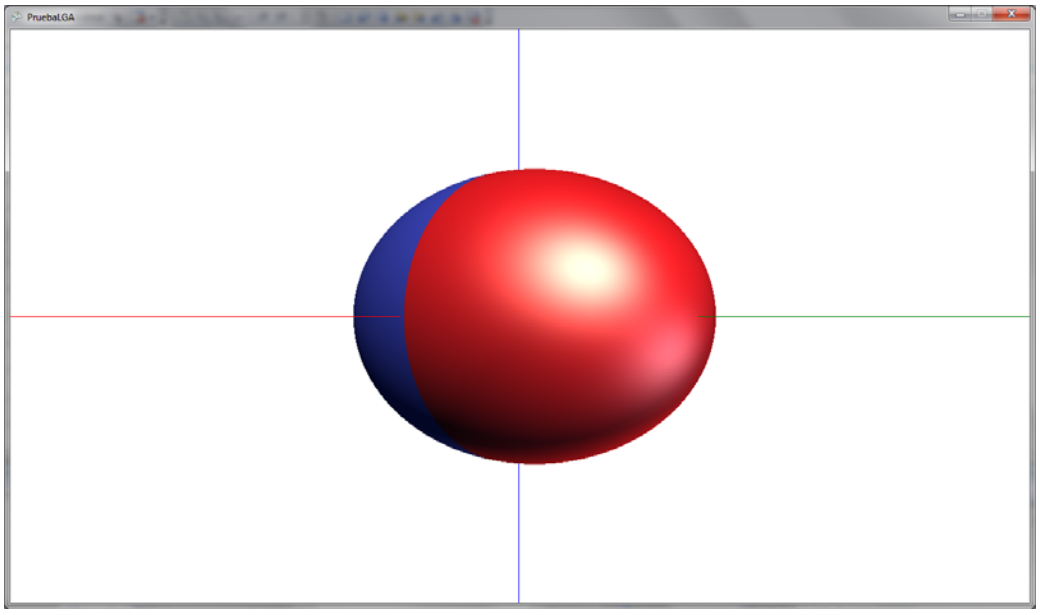

Ilustración 3-10. Elipsoide en XNA 3.1.

3.2. Geometrías en XNA 4.0

Para la última versión de *XNA* con la que trabajé, se realizaron las mismas pruebas, sin embargo por cuestiones de agilidad, se eliminaron los ejes coordenados de dichas pruebas.

Por otra parte aunque hubo algunos cambios en la programación de las geometrías para la versión de Windows Phone 7 visiblemente no se percibe, más bien busqué mejorar y optimizar los algoritmos utilizados para la *renderización* de estas geometrías.

3.2.1. Cilindro Circular Recto

Cilindro con base circular de la versión de móvil.

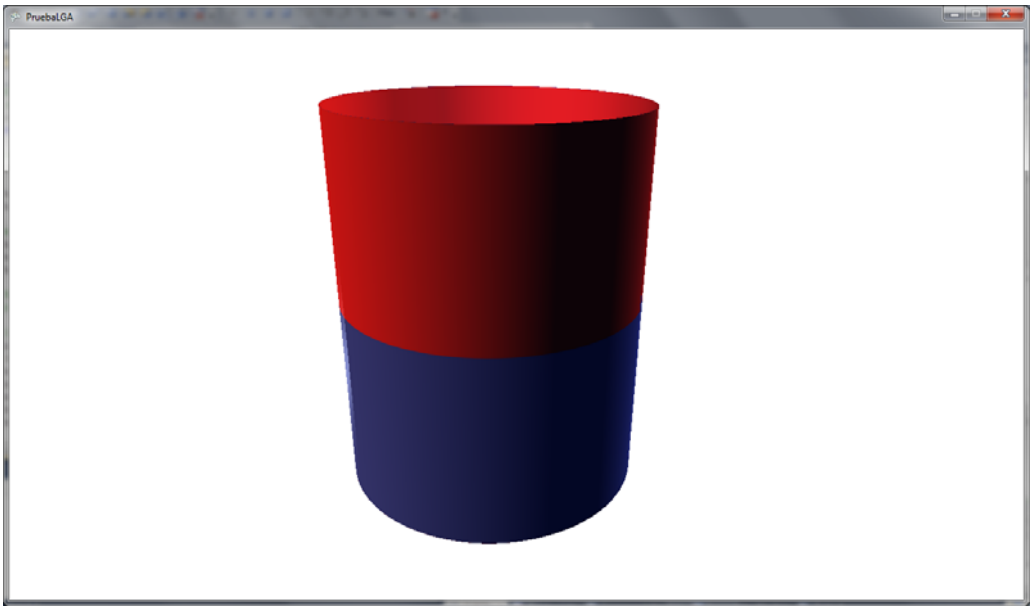

Ilustración 3-11. Cilindro Circular recto en XNA 4.0.

3.2.2. Cilindro Elíptico

Geometría cilíndrica de base elíptica.

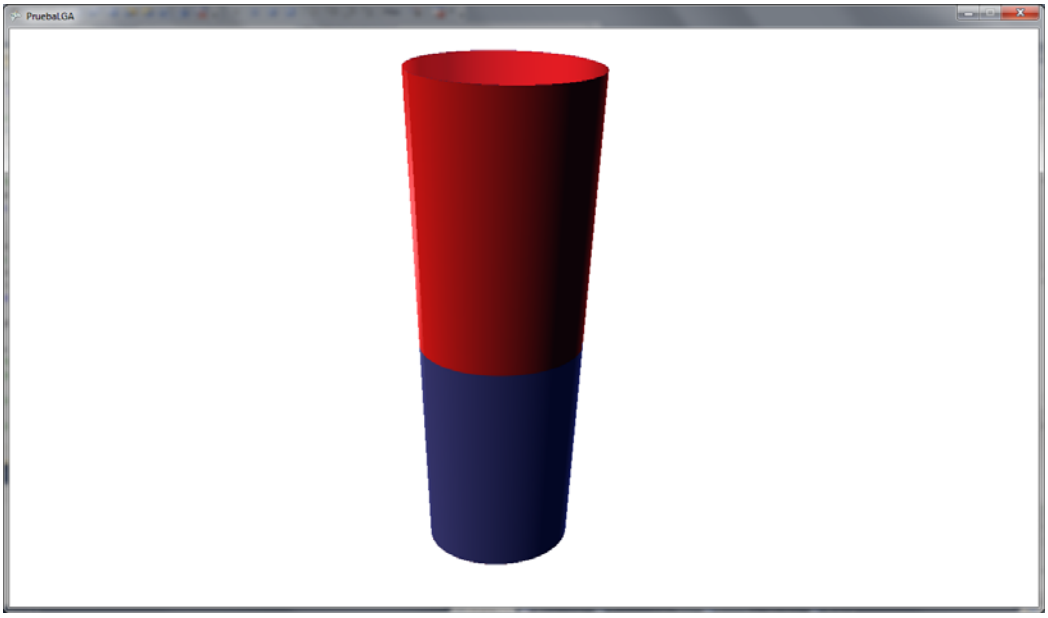

Ilustración 3-12. Cilindro elíptico en XNA 4.0.

3.2.3. Cilindro Parabólico

Geometría integrante de la familia de los cilindros probada para los dispositivos móviles.

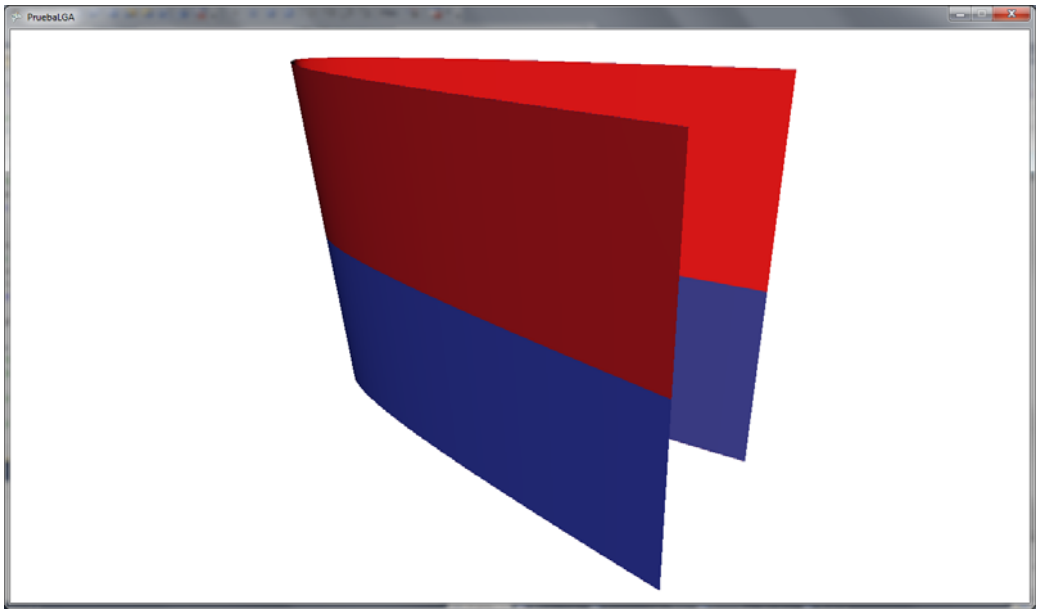

Ilustración 3-13. Cilindro parabólico en XNA 4.0.

3.2.4. Cilindro Hiperbólico

Última geometría de la familia de los cuerpos cilíndricos.

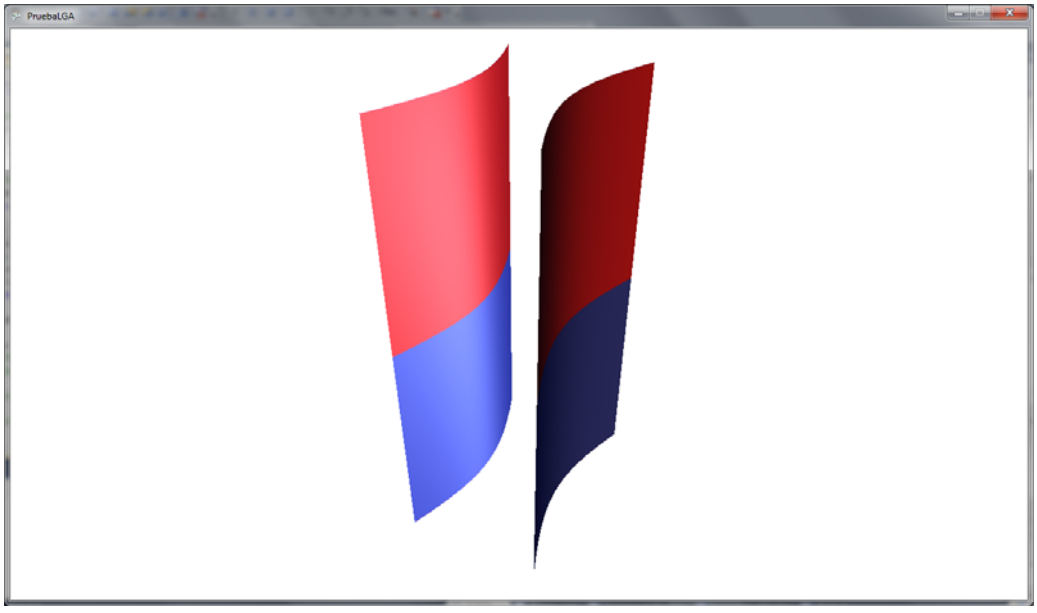

Ilustración 3-14. Cilindro hiperbólico en XNA 4.0.

3.2.5. Cono Circular

Geometría cónica más básica y sencilla de programar.

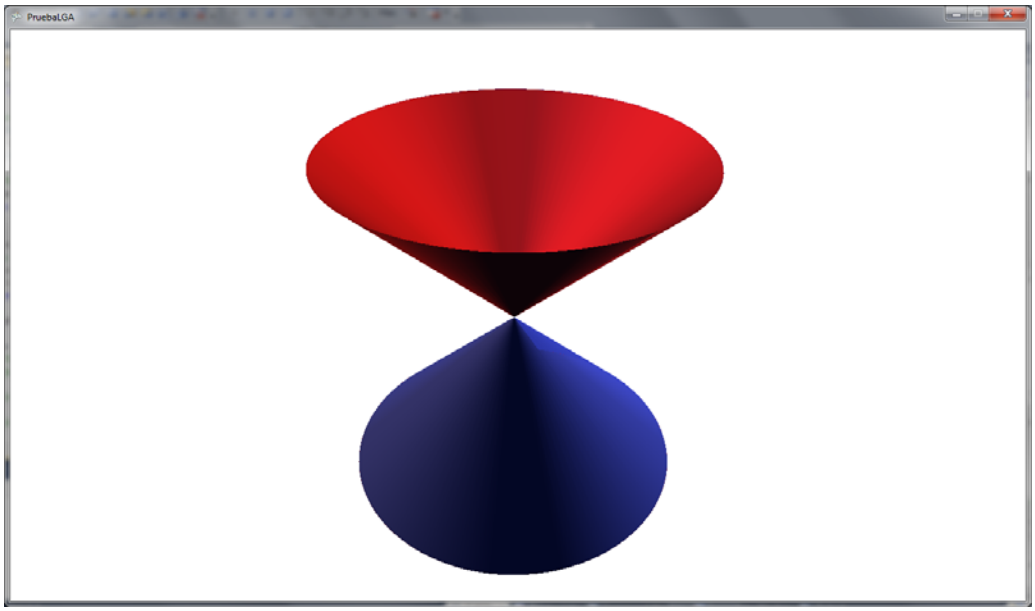

Ilustración 3-15. Cono circular en XNA 4.0.

3.2.6. Cono Elíptico

Imagen del segundo cuerpo cónico que programé.

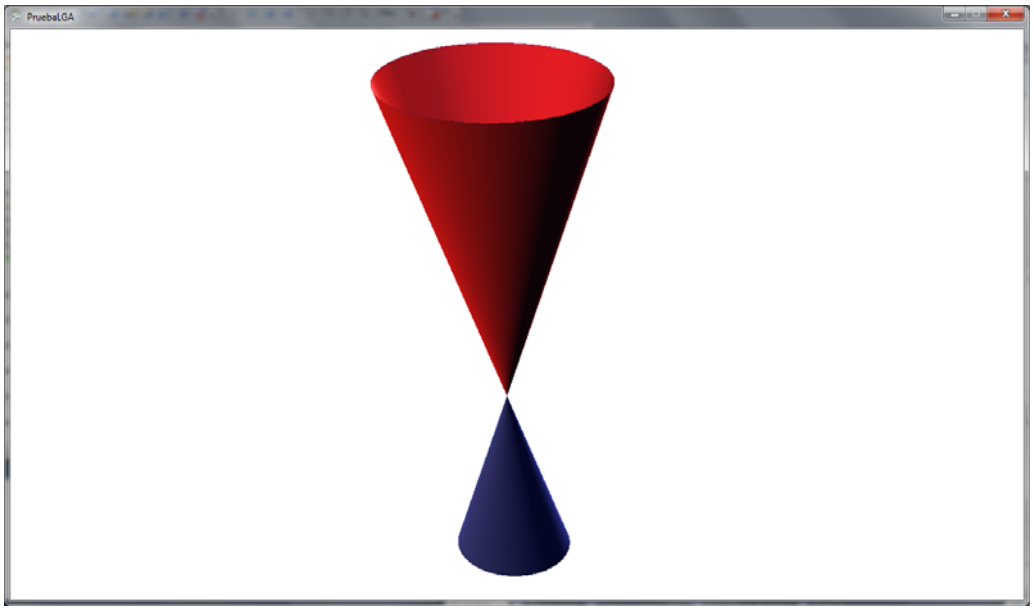

Ilustración 3-16. Cono elíptico en XNA 4.0.

3.2.7. Cono Parabólico

La siguiente captura de pantalla es del cono parabólico.

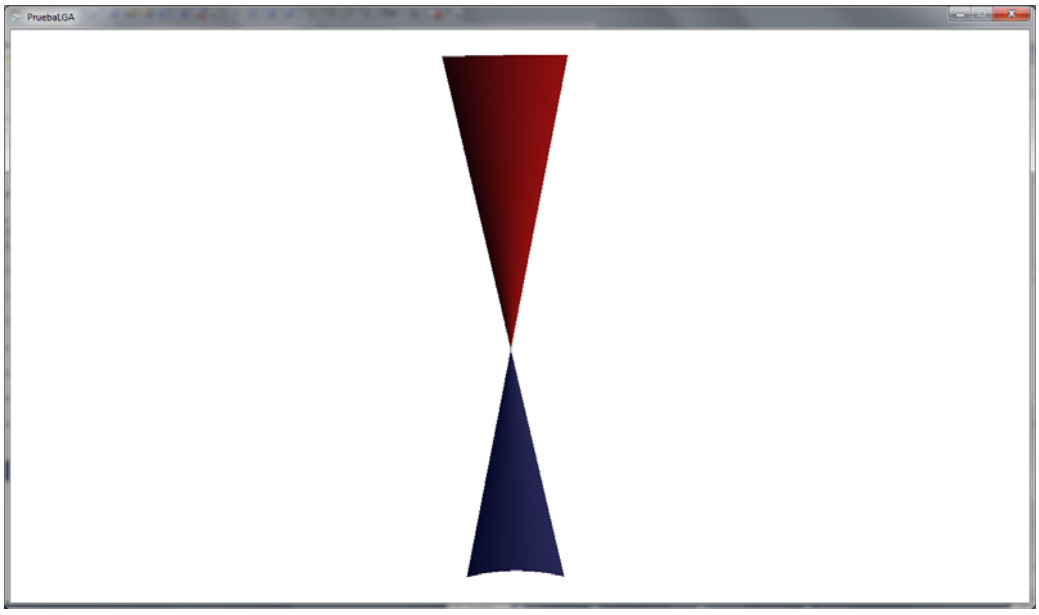

Ilustración 3-17. Cono parabólico en XNA 4.0.

3.2.8. Cono Hiperbólico

Último cuerpo geométrico cónico que programé para esta versión.

Ilustración 3-18. Cono hiperbólico en XNA 4.0.

3.2.9. Esfera

La captura de pantalla siguiente pertenece a la esfera de la versión de *WP7*.

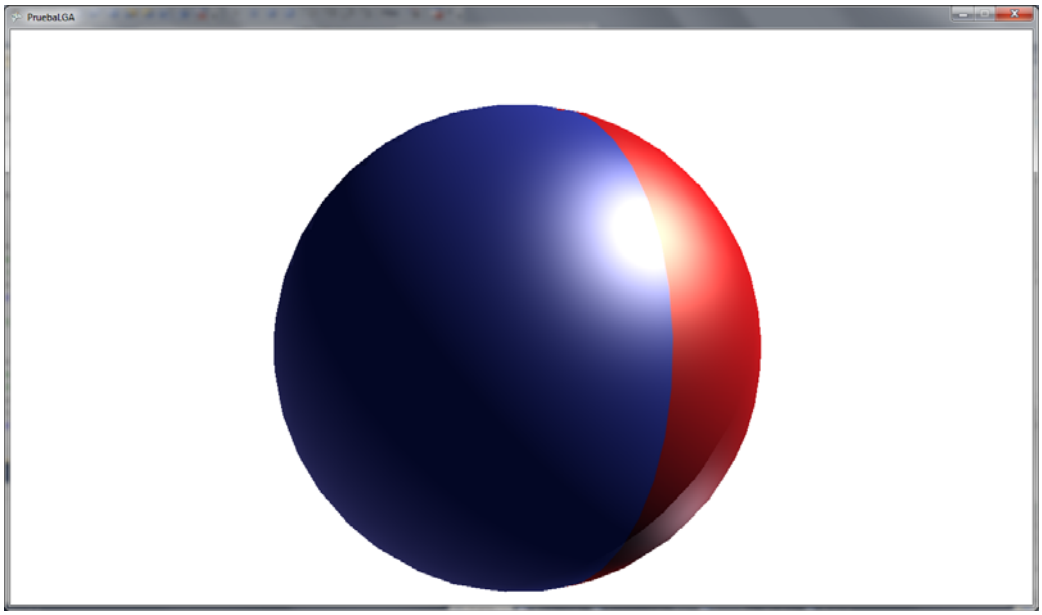

Ilustración 3-19. Esfera en XNA 4.0.

3.2.10. Elipsoide

Esta fue la geometría que más trabajo me costó programar, ya que los algoritmos se iban complicando como avanzaba en la lista de geometrías.

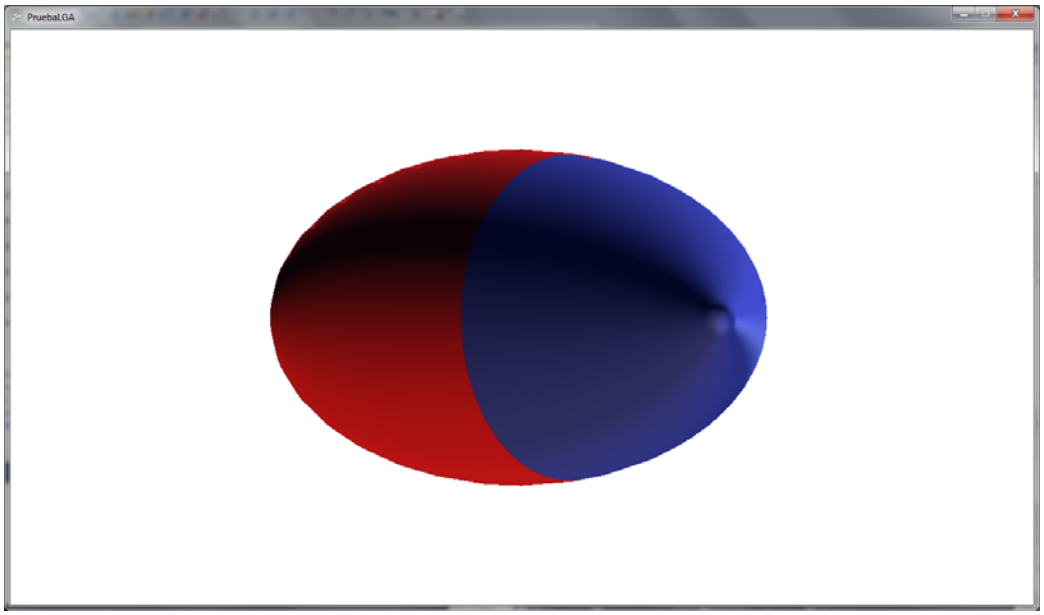

Ilustración 3-20. Elipsoide en XNA 4.0.

Con esto concluyo la sección de imágenes de prueba de las geometrías y continúo a presentar el resultado de ambas versiones de la aplicación.

Como se puede observar en esta serie de imágenes, visualmente no hay algún factor a considerar, sin embargo debido al cambio entre las versiones de XNA tuve que modificar los algoritmos de dibujado e indexado de las geometrías, así como el método de *rasterización*, ya que la manera en la que dibuja la última versión sufrió varias mejoras que tuve que ir adaptando a mis algoritmos anteriores.

4. Resultados del proyecto de visualización de geometrías

En esta sección se detalla la aplicación de visualización de geometrías así como sus características principales y capturas de pantalla en ejecución.

Los elementos que vale la pena destacar caen dentro de la lista de los aspectos que el usuario puede apreciar, cuya importancia dentro de una aplicación radica en hacer el software más llamativo, fácil de comprender y de utilizar. Y en el caso de la herramienta de visualización debo destacar las siguientes características:

- Interfaz gráfica
- Entrada del usuario
- Gráficos

4.1.Interfaz gráfica

La interfaz gráfica es la parte de la aplicación que el usuario observa, con la que interactúa. Debido a que realicé dos versiones de la aplicación, la primera para la consola de videojuegos y computadora, mientras que la segunda para celulares, el diseño cambió radicalmente debido al tamaño reducido de la pantalla de los dispositivos móviles y por ello la interfaz debía estar más limpia, es decir, mínima para permitir observar las geometrías con mayor detalle.

En la interfaz gráfica podemos apreciar los siguientes elementos:

- Menús
- *HUD* (*Head's Up Display*). Término utilizado en los videojuegos que es la información que se muestra en la pantalla incluyendo íconos y texto.[21]

4.1.1. Interfaz en XNA 3.1

El menú principal está conformado de la siguiente manera:

Ilustración 4-1. Menú inicial en XNA 3.1.

En esta pantalla se presentan cada una de las geometrías cilíndricas que el usuario puede seleccionar. Posteriormente al seleccionar una figura geométrica comienza la visualización y cambia el *HUD* para ajustarse a esta tarea.

A continuación enumero los elementos que se presentan en pantalla.

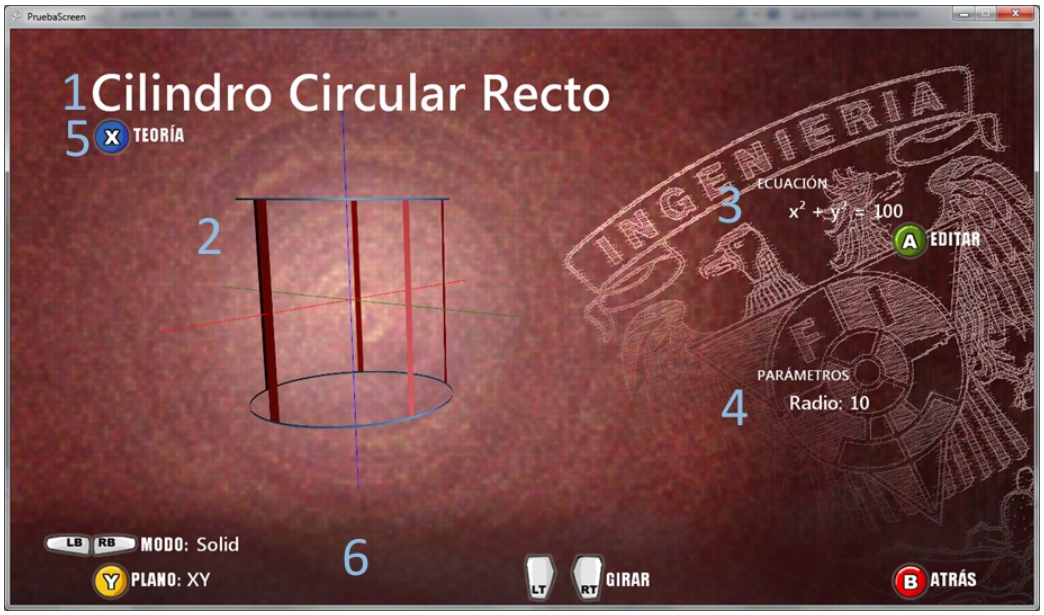

Ilustración 4-2. Interfaz de la aplicación en Xbox 360 y PC.

La lista de los elementos que integran el *HUD* es:

1. El nombre de la geometría.
2. La geometría a estudiar.
3. La ecuación de la geometría, la cual se puede modificar.
4. El o los parámetros modificables.
5. El botón X para leer la teoría.
6. La barra de ayuda de los controles.

4.1.2. Interfaz en XNA 4.0

Como mencioné anteriormente la interfaz de la versión de la aplicación para el celular se diseñó con estilo minimalista.

Coloco primero el menú principal que es la primera pantalla que aparece al ejecutar la aplicación.

Ilustración 4-3. Menú inicial en XNA 4.0.

El cambio más evidente es el uso de íconos representando las geometrías en lugar de texto, ahorrando espacio y obteniendo una mejor experiencia visual. A continuación coloco una captura de pantalla de la interfaz de visualización de la geometría seleccionada.

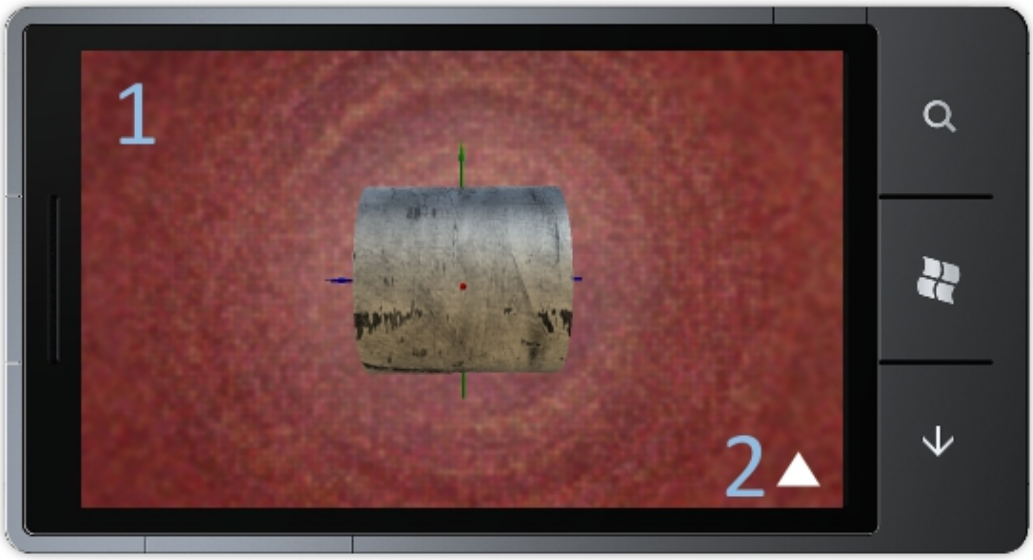

Ilustración 4-4. Interfaz en XNA 4.0.

En esta imagen podemos observar la siguiente lista de elementos presentes en el *HUD*, el cual fue reducido a lo mínimo posible.

1. Área de visualización.
2. Botón de despliegue de ecuación, parámetros y título.

4.2.Entrada del usuario

La entrada o *input* es la manera en la que el usuario interactúa con un programa. En la herramienta de visualización de geometrías programé tres métodos de entrada distintos, los cuales se describen a continuación.

4.2.1. Control

El control es utilizado por la versión de XNA 3.1 y se puede conectar a la consola Xbox 360 así como la computadora. El manejo general de la aplicación se realiza con 4 botones frontales, 2 superiores y los 2 gatillos.

Los botones que se utilizan son los siguientes:

- Los botones frontales A, B, Y y X.
- Los botones superiores LB y RB.
- Los gatillos L y R.

Ilustración 4-5. Vista frontal del control de Xbox 360.

Ilustración 4-6. Vista superior trasera del control de Xbox 360.

A continuación detallo la manera en que responde la aplicación a la entrada del usuario, es decir las funciones de los botones que programé para controlar la aplicación.

1. El botón X muestra el texto de la teoría.

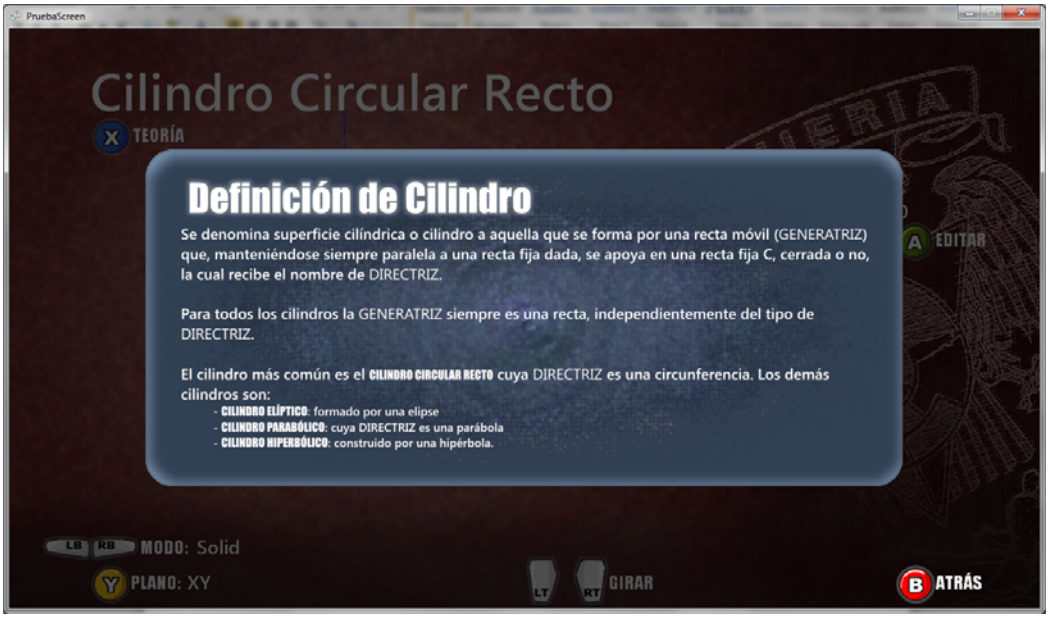

Ilustración 4-7. Teoría en XNA 3.1.

2. El botón Amuestra el menú de edición.

Ilustración 4-8. Interfaz de edición de la geometría en XNA 3.1.

Una vez en este menú se puede utilizar el *stick* o palanca izquierda para modificar el valor, o se puede presionar el botón A para ingresar el valor exacto. Detallo esta acción en el apartado del teclado.

Para el caso de geometrías con más de un valor modificable la función de la palanca derecha es seleccionar el coeficiente a modificar y moviendo el cursor color rojo que indica el valor a modificar.

Lo anterior se puede apreciar en la captura de pantalla que coloco a continuación.

Ilustración 4-9. Edición de geometría con múltiples coeficientes.

3. El botón Y cambia el plano en el que se crean las directrices.

Ilustración 4-10. Cambio de plano de directrices en XNA 3.1.

4. El botón B regresa al elemento anterior.

5. Los botones superiores LB y RB cambian el modo de *rasterizado* de la geometría.

Ilustración 4-11. Cambio de modo de rasterizado.

6. Los gatillos L y R permiten rotar la geometría en torno al eje z.

Ilustración 4-12. Rotación de la geometría.

Con esto concluyo la explicación del manejo de la aplicación utilizando los botones, gatillos y palancas del control de Xbox 360 y puedo comentar que este puede fungir como un dispositivo que permite fácilmente la visualización de objetos virtuales.

4.2.2. Pantalla táctil

Como mencioné previamente, para la mayoría de los estudiantes que tienen o alguna vez han jugado un videojuego les resultará sencillo entender el manejo de la aplicación, esto se comprobó en la semana SEFI cuando se mostró la aplicación a los estudiantes, quienes dieron una retroalimentación positiva.

Antes de comenzar a describir el manejo de la visualización de las geometrías en el celular, debo aclarar que la teoría de las geometrías abarca a toda la familia geométrica por lo tanto en la versión móvil la encontramos en el menú principal.

En la siguiente imagen se puede observar el texto de la teoría de la familia de los cilindros en la versión móvil de la herramienta.

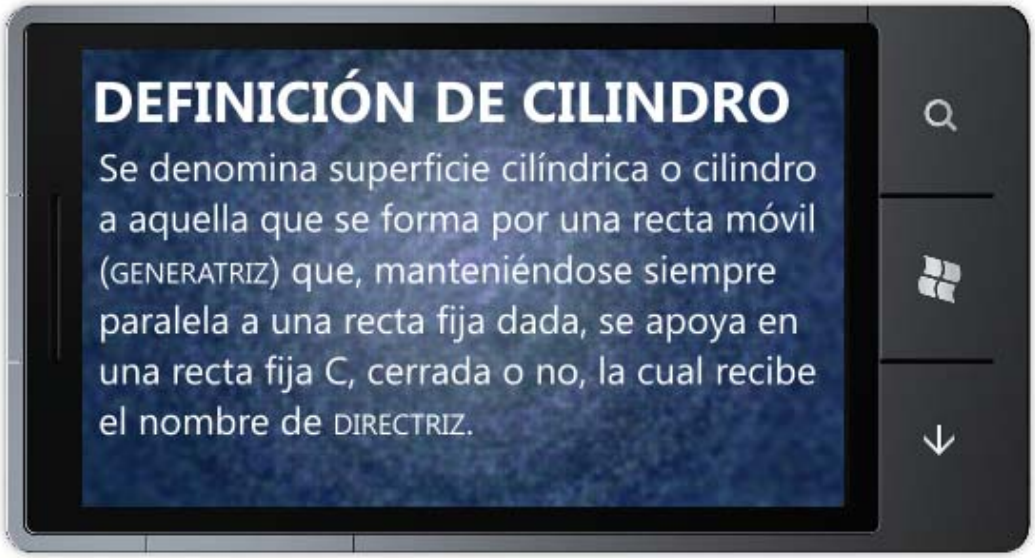

Ilustración 4-13. Teoría en XNA 4.0.

De nuevo comento que la orientación que tomó el diseño y el control de la versión de *Windows Phone* cambió radicalmente.

Los celulares con *WP7* tienen una pantalla capacitiva táctil *multitouch* o multi-toque que le permite recibir hasta 4 toques, la única función que utiliza más de un dedo es el *zoom* o acercamiento. Gracias a este tipo de entrada el control de la visualización de las geometrías se volvió más sencillo e intuitivo.

La acción más básica que se puede hacer para controlar la visualización es mover la geometría libremente con un dedo.

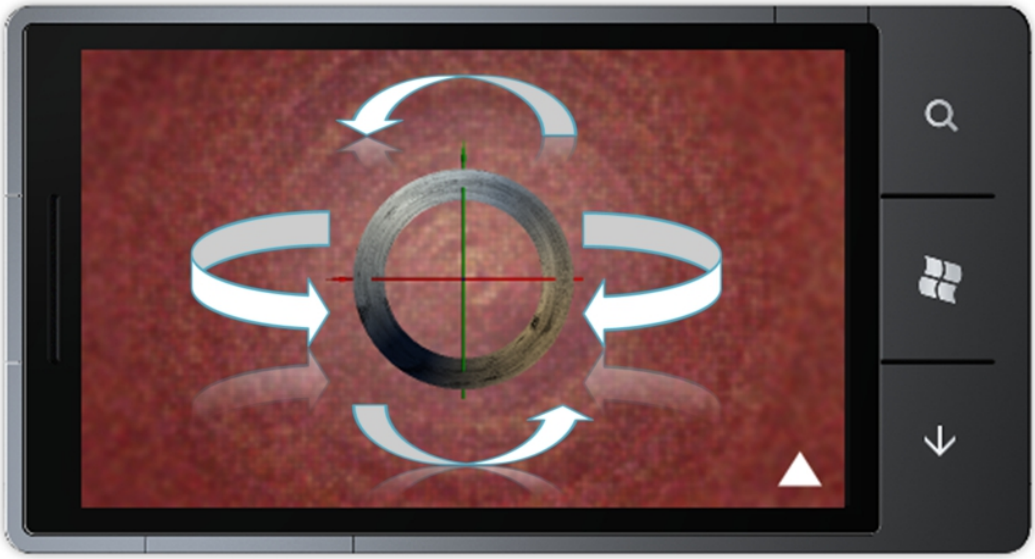

Ilustración 4-14. Movimiento libre de la geometría.

El *zoom* se realiza colocando 2 dedos en la pantalla y se acercan o separan dependiendo lo que se desee.

El tercer tipo de entrada que coloqué para controlar la aplicación es el toque simple aplicado en la flecha ubicada en la esquina inferior derecha que despliega el siguiente menú de edición.

Captura de pantalla del menú de edición de la geometría

Ilustración 4-15. Interfaz de edición en XNA 4.0.

Esta interfaz está compuesta por los siguientes elementos:

1. Nombre de la geometría
2. Parámetro o parámetros
- Barra de edición que no se encuentra numerada ya que tiene lo siguiente:
3. Ecuación
4. Cambio de plano de las directrices
5. Flecha de cierre del menú de modificación

Debo mencionar que para modificar los coeficientes de la ecuación hay que tocar el o los números que se desea, pero hay ocasiones en que no se pueden modificar todos, por lo que los que sean modificables serán color blanco y los que no gris.

En esta imagen se puede observar los coeficientes modificables.

Ilustración 4-16. Modificación de los coeficientes en XNA 4.0.

Al momento tocar uno de los coeficientes de la ecuación se desplegará el teclado para ingresar el valor deseado, eso se explica en el próximo apartado referente al teclado.

La última función permite cambiar el plano en el que se encuentran las directrices de la geometría y ello se realiza tocando el texto del plano, número 4 en la lista de los elementos de la interfaz.

La siguiente imagen presenta la misma geometría cuyas directrices se encuentran en otro plano.

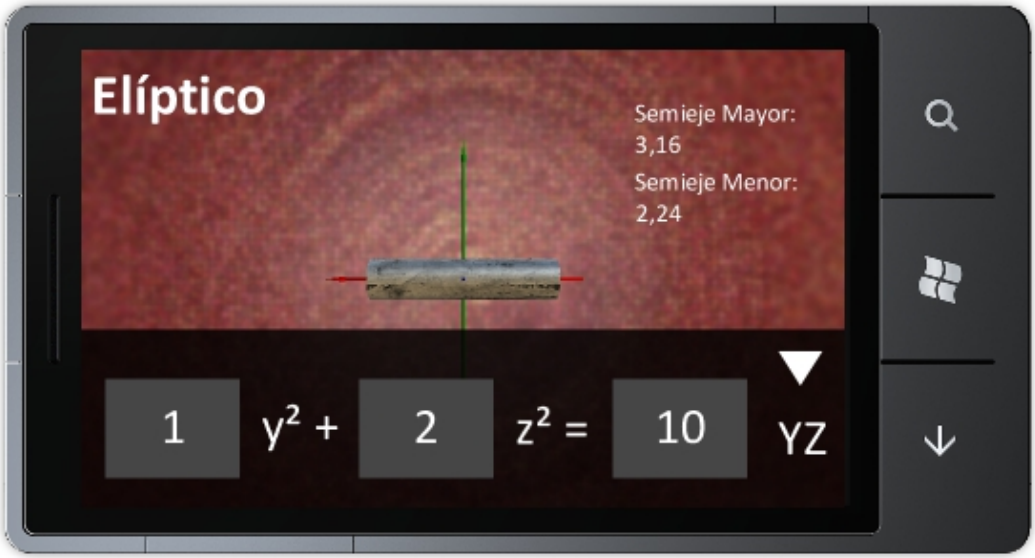

Ilustración 4-17. Cambio de plano de directrices en XNA 4.0.

Para finalizar la edición y volver a tener el control de la cámara es necesario tocar la flecha que se encuentra en la barra de edición, la cual es el elemento listado con el número 5.

Además de la pantalla táctil el teléfono tiene botones que presentan funcionalidad en la aplicación.

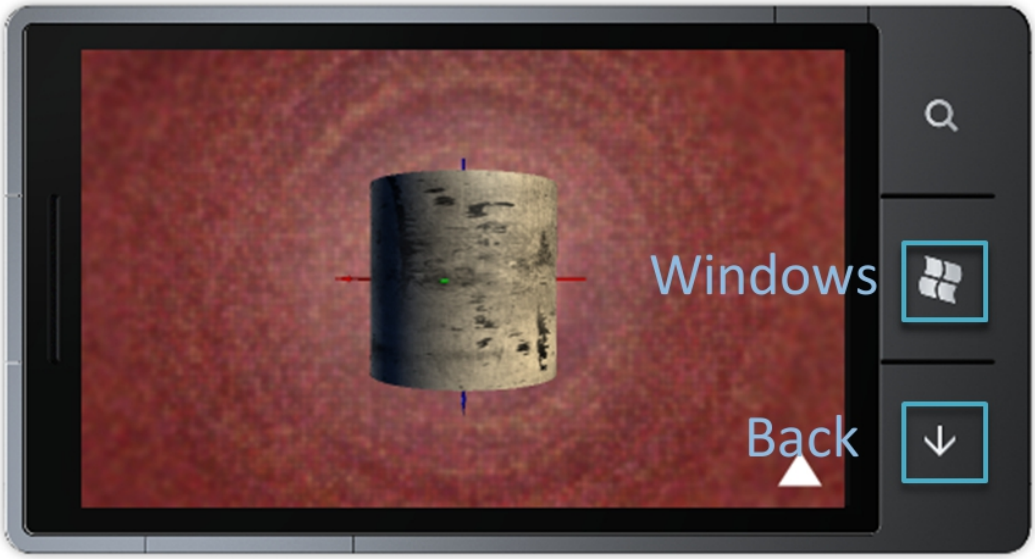

Ilustración 4-18. Uso de los botones del dispositivo.

El botón *Back* o atrás sirve para regresar al menú anterior y si se presiona estando en el menú inicial cierra la aplicación. El botón de *Windows* por su parte minimiza la aplicación como lo hace con cualquier otro programa instalado en el dispositivo.

4.2.3. Teclado

La modificación de las geometrías en la versión de consola y computadora se puede realizar tanto con los *sticks* como con el teclado virtual o físico en una computadora, sin embargo en la versión portátil tenemos el teclado emergente como el único método de ingreso de datos.

En el apartado del control se dijo que una vez en el menú de edición de una geometría, presionando el botón A al tener seleccionado un coeficiente de la ecuación, se despliega el menú de *Xbox LIVE* o *Games for Windows LIVE* que se observa a continuación.

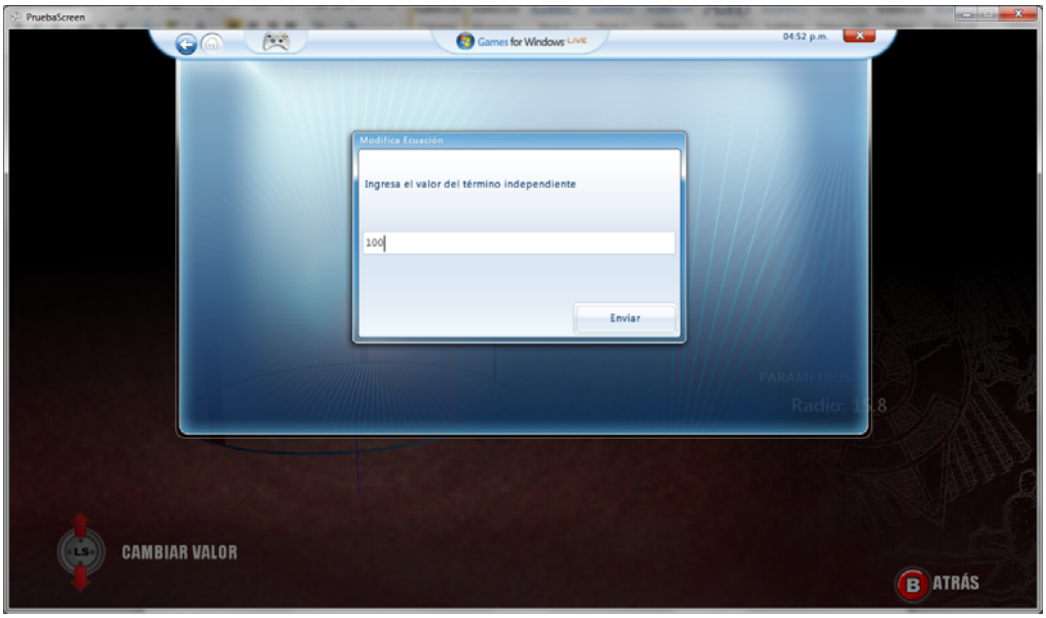

Ilustración4-19. Interfaz de Games for Windows LIVE.

En la computadora se observa el menú de Games for Windows LIVE de esta manera y se puede ingresar el valor deseado con un teclado físico, sin embargo en la consola aparece el menú de Xbox LIVE con un teclado emergente similar al de la versión de Windows Phone 7.

Una vez ingresado el dato se puede presionar *Enter* en el teclado o se puede utilizar el *stick* izquierdo para poner el cursor sobre el botón ***Enviar*** y se debe presionar el botón A.

En la siguiente imagen se observa el cursor resaltando el botón de ***Enviar***.

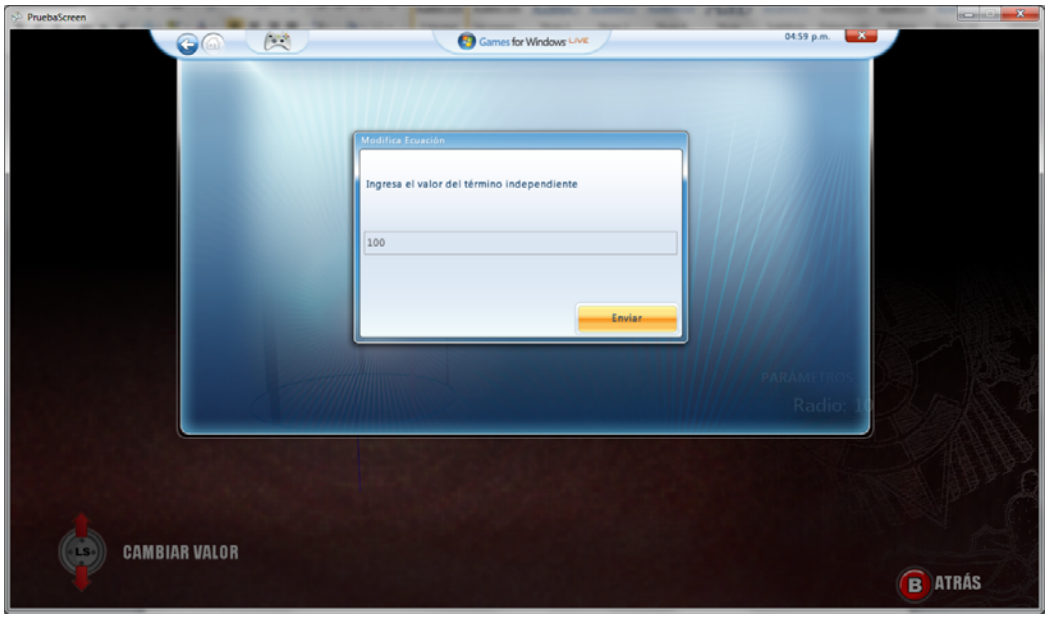

Ilustración 4-20. Ingresando el valor del coeficiente en PC.

En la versión de celular al tocar un coeficiente modificable se abre la interfaz de edición de la geometría, donde se despliega el teclado emergente en el cual se puede ingresar el dato siguiendo las instrucciones que se encuentran a continuación.

1. Tocar el botón de teclado numérico que se encuentra en la esquina inferior izquierda de la pantalla.

Ilustración 4-21. Botón de cambio a modo numérico.

2. Tocar los números para ingresar el valor deseado.

Ilustración 4-22. Ingresando el valor del coeficiente en WP7.

3. Finalmente se debe tocar el botón OK para ingresar el dato.

Ilustración 4-23. Finalizando el ingreso del valor.

Como mencioné anteriormente el teclado emergente en la versión de XNA 3.1 para consola Xbox se presenta de manera similar al de Windows Phone 7 con la diferencia que la interfaz presentada es parte de Xbox LIVE.

4.3.Gráficos

La versión de la aplicación para consola y PC presenta tres distintos modos en que se puede *rasterizar* la geometría, esto se debe a que las capacidades del *hardware* lo permiten, sin embargo las restricciones del celular incluyen el uso únicamente del efecto básico de XNA.

4.3.1. Gráficos en XNA 3.1

Para cambiar de modo de *rasterización* se pueden oprimir tanto el botón *LB* como *RB* que se encuentran en la parte superior del control.

4.3.1.1. Modo de transparencia

En este primero modo que coloqué por defecto se pueden apreciar ciertas áreas de la geometría transparentes permitiendo observar lo que se encuentra detrás. Esto se logra utilizando una textura con transparencia o canal *alpha* y un vector normal asignado a cada vértice.

Ilustración 4-24. Geometría con transparencia.

4.3.1.2. Modo de alambrado

El modo *wireframe* o alambrado permite observar la manera en que se comporta el algoritmo de indexado de los vértices y se generan los triángulos que forman las caras de las geometrías.

Ilustración 4-25. Modo firewire.

4.3.1.3. Modo de efecto avanzado

En este modo se utiliza un *shader* o efecto en el cual hay 6 luces de diferentes colores en distintas posiciones iluminando la geometría que se rasteriza con una textura difusa, una textura especular y una textura de normales.

La textura difusa es una imagen que contiene la información de los colores, la textura especular es una imagen en blanco y negro que indica las áreas de brillo y por último la textura de normales es una imagen azul y magenta que indica las normales del objeto.

Con las texturas adecuadas se obtiene el siguiente resultado:

Ilustración 4-26. Shader con múltiples luces, especular y normal.

Este *shader* le da un efecto de apariencia metálica con bordes y relieves que se ilumina por varias luces. La geometría incluso parece ser parte de una galería de arte.

4.3.2. Gráficos en XNA 4.0

Como se mencioné anteriormente los dispositivos celulares con *Windows Phone 7* utilizan el modo *Reach* de *XNA4.0*, por lo que tanto a nivel de hardware como a nivel de software no se pueden utilizar *shaders* ajenos al incluido en *XNA*. Por ello en ésta versión se maneja únicamente el efecto básico utilizando una textura difusa y la información del vector normal de cada vértice.

El resultado que se obtiene no es visualmente impactante, sin embargo conserva la estética minimalista del diseño de esta versión. A su vez los ejes coordenados fueron sustituidos por modelos tridimensionales elaborados en un programa de modelado 3D.

A continuación se encuentra la captura de pantalla que muestra lo anterior.

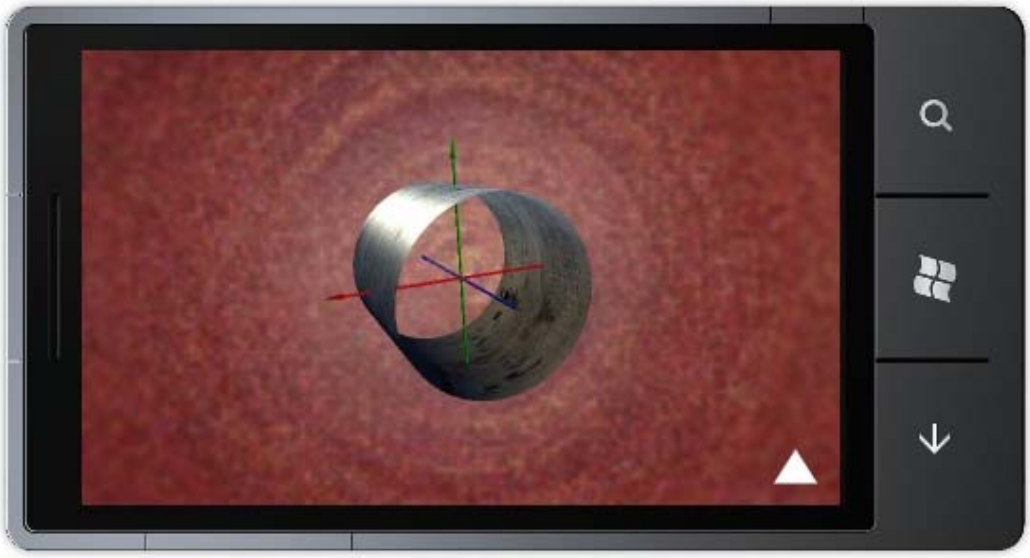

Ilustración 4-27. Gráficos en XNA 4.0

Con esto concluyo el capítulo de resultados de la Herramienta de Visualización de Geometría Analítica tanto para Windows y Xbox 360 como para Windows Phone 7.

Debo añadir que esta herramienta de visualización de geometrías, tanto en su versión de consola o computadora como en la versión portátil puede ser de gran utilidad para el estudio de la materia de geometría analítica, el cuál es el objetivo inicial de dichos proyectos como mi labor en la División de Educación Continua y a Distancia de la Facultad de Ingeniería.

Conclusiones

A lo largo de este reporte he mencionado los proyectos que desarrolle en mi labor en el Departamento de Investigación y Desarrollo de la División de Educación Continua y a Distancia de la Facultad de Ingeniería, además describí detalladamente ambas versiones del proyecto de la Herramienta de Visualización de Geometría Analítica.

Debo mencionar que cumplí satisfactoriamente el objetivo inicial de los todos los proyectos en los que participé incluyendo el Recorrido Virtual, la aplicación de Videoconferencia y la primera práctica de la nueva versión del Laboratorio Virtual de Geometría Analítica y por supuesto la Herramienta de Visualización.

Por otra parte la experiencia que obtuve desarrollando el primer proyecto me fue de gran ayuda para poder participar en el último. Y el curso que tomé de Desarrollo de Videojuegos en UDK aceleró el desarrollo de segundo proyecto y me dio las bases para manejar las herramientas utilizadas durante el último.

La conclusión final a este documento es que gracias a mis labores en esta dependencia de la U.N.A.M. he aprendido a programar en tres nuevos lenguajes, manejar dos motores gráficos, adquirí la experiencia profesional necesaria hoy en día en la industria del *software,* me desarrollé profesionalmente y por último extendí el conocimiento adquirido en el estudio de la carrera de ingeniería en computación.

Glosario

ActionScript : Lenguaje de programación de Adobe orientado a objetos utilizado diversas plataformas como Flex y Flash.

API: Siglas de Aplication Program Interface, en español Interfaz de programación de aplicaciones.

C#: Lenguaje de programación de Microsoft orientado a objetos.

Direct3D : API utilizado para la programación y procesamiento de gráficos en 3D.

DirectX: Colección de interfaces de programación para la programación de juegos y video.

Depth of Field: Efecto de profundidad de campo utilizado en películas, fotografía y videojuegos.

BasicEffect: Efecto básico por defecto de XNA.

Flash Media Server: Servidor de transferencia de medios de Adobe.

Framework: Entorno de trabajo de programación de aplicaciones.

Games for Windows LIVE: Servicio de interacción social en línea dentro de los juegos, además de permitir la compra y descarga de videojuegos.

HiDef: Perfil de XNA de gráficos en alta definición para Xbox 360 y PC.

High Level Shader Language: Lenguaje de Sombreado de Alto Nivel.

HUD: Head's Up Display, es la información que se muestra en pantalla en un videojuego.

Input: Entrada del usuario a una aplicación.

iOS: Sistema operativo de dispositivos y computadoras de la marca Apple.

iPad: Tableta electrónica de Apple.

iPhone: Dispositivo celular de Apple.

JavaScript: Lenguaje de programación orientado a objetos.

.NET: Entorno de trabajo de Microsoft.

Pizarrón Digital: Programa de ayuda para la educación a distancia en línea.

Raster: Imagen digital resultante de la rasterización.

Reach: Perfil de XNA de gráficos sencillos para Xbox 360 y Windows Phone 7.

Renderizar: Proceso de generar imágenes en dos dimensiones a partir de elementos tridimensionales.

Script: Programa usualmente sencillo.

Shader: Código en un lenguaje de sombreado que trabaja en los procesadores de los dispositivos de video y es capaz de crear distintos efectos visuales.

Shader Model : El estándar de shaders en la industria.

UDK: Siglas de Unreal Development Kit.

Unity: Motor de juegos gratuito.

Unreal Development Kit: Versión gratuita del motor de juegos Unreal Engine 3.

Visual scripting: Término acuñado a la programación visual.

Windows Phone 7: Sistema operativo de Microsoft para dispositivos celulares.

Wireframe: Modo de renderización de XNA y otras interfaces de programación cuya traducción es alambrado.

Xbox 360: Consola de videojuegos de Microsoft.

Xbox LIVE: Servicio de la consola de videojuegos que permite jugar e interactuar en línea, comprar y descargar juegos, películas y música, tener un perfil de jugador y demás.

Referencias

[1]http://www.mineria.unam.mx/– Página de inicio de la División de Educación Continua y a Distancia de la Facultad de Ingeniería, U.N.A.M. – última revisión 4 de septiembre del 2011.

[2]http://create.msdn.com/en-US/– Página en inglés de inicio del centro de aplicaciones de XNA, MSDN – última revisión 4 de septiembre del 2011.

[3]http://docforge.com/wiki/Framework– Página con definición en inglés de *Framework*, DocForge – última revisión 4 de septiembre del 2011.

[4]http://en.wikipedia.org/wiki/Application_programming_interface– Página con definición en inglés de *API*, Wikipedia – última revisión 4 de septiembre del 2011.

[5]http://www.microsoft.com/net/overview.aspx– Página de inicio en inglés de *.NET*, Microsoft – última revisión 4 de septiembre del 2011.

[6]http://blogs.msdn.com/b/xna/archive/2006/08/25/724607.aspx– Página en inglés con descripción del *framework* de *XNA*, MSDN Blogs – última revisión 4 de septiembre del 2011.

[7]http://msdn.microsoft.com/en-us/library/bb153256(v=VS.85).aspx– Página con definición en inglés de Direct3D, MSDN – última revisión 4 de septiembre del 2011.

[8]http://es.wikipedia.org/wiki/Shader– Página con definición de *Shader*, Wikipedia – última revisión 4 de septiembre del 2011.

[9]http://msdn.microsoft.com/en-us/library/bb509561(v=VS.85).aspx – Página con definición en ingles de *HLSL*, MSDN – última revisión 4 de septiembre del 2011.

[10]http://es.wikipedia.org/wiki/Renderizaci%C3%B3n – Página con definición de *renderización*, Wikipedia – última revisión 4 de septiembre del 2011.

[11]http://www.alegsa.com.ar/Dic/renderizar.php– Página con definición de *renderización*, Alegsa – última revisión 4 de septiembre del 2011.

[12]http://msdn.microsoft.com/es-es/library/bb200104.aspx – Página con definición de *XNA Game Studio 4.*0, MSDN – última revisión 4 de septiembre del 2011.

[13]http://msdn.microsoft.com/en-us/library/bb401006.aspx – Página en inglés con versiones de *XNA*, MSDN – última revisión 4 de septiembre del 2011.

[14]http://msdn.microsoft.com/en-us/library/bb203925.aspx – Página en inglés con información de requisitos de hardware de *XNA*, MSDN – última revisión 4 de septiembre del 2011.

[15]http://blogs.msdn.com/b/shawnhar/archive/2010/03/12/reach-vs-hidef.aspx – Página en inglés con información de perfiles de *XNA*, MSDN Blogs – última revisión 4 de septiembre del 2011.

[16]http://www.udk.com/ – Página inicial en inglés de *UDK*, UDK – última revisión 4 de septiembre del 2011.

[17]http://www.adobe.com/es/products/flex/ – Página inicial de Flex, Adobe – última revisión 4 de septiembre del 2011.

[18]http://es.wikipedia.org/wiki/ActionScript – Página con información de *ActionScript*, Wikipedia – última revisión 4 de septiembre del 2011.

[19]http://www.adobe.com/es/products/flashmediaserver/ – Página con información de *Flash Media Server*, Adobe – última revisión 4 de septiembre del 2011.

[20]http://unity3d.com/unity/ – Página en inglés con información y características de *Unity 3*, Unity – última revisión 4 de septiembre del 2011.

[21]http://es.wikipedia.org/wiki/HUD_(inform%C3%A1tica) – Página con definición de *HUD*, Wikipedia – última revisión 4 de septiembre del 2011.