Alternativas en la Enseñanza Intensiva de Posgrado en Computación Gráfica

Claudio Delrieux
Departamento de Ingeniería Eléctrica
Universidad Nacional del Sur
Alem 1253 - (8000) Bahía Blanca - ARGENTINA - e-mail:claudio@acm.org

Resumen

En este trabajo presentamos la experiencia de diseñar e impartir un curso intensivo de Posgrado en Computación Gráfica. Se discuten las ventajas y desventajas de las diferentes propuestas de contenidos mínimos en general, y para el caso particular de cursos intensivos. Se sugiere que una metodología experimental es más adecuada para un curso intensivo. Se presenta un detalle de los contenidos del curso, los trabajos prácticos y se muestran los resultados obtenidos por los alumnos.

Palabras Clave: Computación Gráfica, Cursos de Posgrado

1 Contexto de la experiencia

En este trabajo proponemos un modelo de curso intensivo de Computación Gráfica para alumnos de Maestría en Ciencias de la Computación, el cual fue impartido en tres diferentes oportunidades a interlocutorios distintos, con muy buenos resultados. El curso se origina dentro de un programa de posgrado en la Universidad Mayor San Simón (UMSS) en Cochabamba, Bolivia, el cual ofrece la Maestría en Ciencias de la Computación por medio de un convenio con la Universidad de Utrecht (Holanda), quien colaboró con el diseño curricular de los cursos y la provisión de docentes calificados para los mismos. Dentro de dicho plan se incluyó la materia *Algoritmos de Graficación*, la cual, como su nombre lo indica, busca enfatizar el aspecto algorítmico y de implementación de la Computación Gráfica (CG). Como ya se discutiera en muchas oportunidades, la enseñanza de la CG en materias regulares es bastante problemática por los conflictos entre las expectativas de los alumnos y las posibilidades que se plantean en el dictado de un curso [2, 3, 5, 13, 17, 18]. A esta situación además le agregamos las dificultades que normalmente se deben afrontar en un curso intensivo dictado en breve tiempo (alumnos asistiendo muchas horas semanales a clase con presiones de trabajos pendientes para otros cursos), probablemente con obligaciones laborales y familiares, por lo que no se puede contar con que lean o asimilen debidamente los conceptos y planteen sus dudas en clase. Por lo tanto, es menester realizar una cuidadosa elección de objetivos y metodologá, y también una adecuada planificación para llevarlos a cabo.

Uno de los factores que conspira contra el satisfactorio cumplimiento de un conjunto razonable de dichos objetivos es que los alumnos en general tienen una expectativa muy alta respecto de los contenidos del curso y especialmente con los trabajos prácticos. En un curso de inteligencia artificial los alumnos no presuponen que van a realizar sistemas que jueguen al ajedrez, reconozcan textos manuscritos o comprendan el lenguaje natural. Tampoco pretenden desarrollar como trabajo final de una materia de compiladores, por ejemplo, una JVM para una arquitectura paralela. Sin embargo, por causas difíciles de conocer –pero evidentemente culturales– los alumnos suponen que en un curso de CG es posible dominar

los conceptos involucrados en la elaboración de productos como la película *Shrek* o el juego *Tomb Rider*, aunque probablemente las dificultades tanto teóricas como tecnológicas sean equivalentes o mayores a los ejemplos mencionados en otras materias. Indudablemente los alumnos están bombardeados por la gran difusión cultural que tienen los gráficos por computadora, a la disponibilidad de programas comerciales con aplicaciones gráficas avanzadas, y a la proliferación de sistemas de distribución libre y de APIs gráficas.

Sin embargo, la tendencia general en la enseñanza de la CG ha sido la de no confrontar a los alumnos con las dificultades y exigencias del la CG, tanto en la formación requerida (matemática, física, tecnología informática y programación) como en la permanente necesidad de mantenerse actualizado en la aparición de publicaciones y productos de hardware y software. En general, el argumento propuesto en los foros de educación en CG ha sido otro: dado que un curso clásico (siguiendo los libros tradicionales como [7, 8, 14, 22]) puede ser insatisfactorio respecto de las expectativas de los alumnos, entonces se propone eliminar de la temática los fundamentos (que están disponibles como software gratuito, librerías, sistemas comerciales, o hasta tarjetas gráficas aceleradoras) y concentrarse en la programación desde el alto nivel, propiciando un enfoque *top-down*. En algunos casos se propone el uso de sistemas abiertos (POV-Ray, por ejemplo) o sistemas desarrollados *ad-hoc* en los laboratorios, y en otros casos se sugiere el uso de APIs gráficas como OpenGL. De esa forma –sigue el argumento– a lo largo de un cuatrimestre los alumnos pueden desarrollar una aplicación gráfica tridimensional interactiva.

Sin embargo, en nuestra opinión, si el objetivo fundamental de un curso introductorio a la CG es la implementación de una aplicación gráfica interactiva (lo cual sería también todo un tema de discusión), entonces es posible arribar a este objetivo sin apartarse tan radicalmente del enfoque tradicional. Además, depender de sistemas comerciales o programas de distribución libre agrega otro nivel de complicación al curso, además de imponer restricciones *a priori* a la conceptualización que los alumnos realizan de la materia, a sus posibilidades exploratorias y a la capacidad de expresión de ideas, limitando la creatividad y distrayendo el interés en los aspectos teóricos y de investigación (lo que probablemente debería ser uno de los objetivos del curso, al menos en nuestro contexto). En vez de aprovechar las horas de clase para presentar el rico marco conceptual y algorítmico de la CG, se utiliza el tiempo en el aprendizaje de sistemas que no enriquecen la formación de los alumnos.

Además, en el caso particular de este curso intensivo de posgrado, un cambio en el plan de la materia, que incorpore únicamente los elementos tecnológicos en el centro de la escena, puede retractarnos del objetivo curricular planteado de brindar conocimiento y experiencia de los elementos algorítmicos constitutivos de una aplicación gráfica. Al mismo tiempo, en nuestra opinión, la CG ha producido una gradual pero profunda revolución en la manera en que se utiliza la computadora en los contextos científicos y tecnológicos. Tal es el caso, por ejemplo, de la visualización científica, el diseño asistido, la realidad virtual, etc. Esto es especialmente importante si tenemos en cuenta que muchos de los alumnos de esta Maestría son docentes universitarios y profesionales, es decir, responsables a su vez de la formación de recursos humanos. Por ello es necesario contemplar su preparación y actualización en estos temas.

Teniendo en cuenta todos estos elementos, se propuso un plan de estudios para un curso intensivo de CG para alumnos de Maestría en Ciencias de la Computación. El mismo comienza al estilo de un curso clásico, e incorpora el uso de librerias y sistemas abiertos de una manera gradual, en el momento en que los trabajos prácticos realizados a la manera tradicional ya le han dejado al alumno una clara comprensión formativa de los distintos elementos teóricos y de implementación que constituyen una aplicación gráfica. Es importante impartir el conocimiento de los algoritmos fundamentales y los temas básicos y avanzados, brindando la posibilidad de conocer y comprender la importancia de los problemas de investigación que plantea la CG. Esto no excluye el uso de software de distribución libre, librerías y APIs,

siempre que aprendan a incorporar la programación de estos elementos desde sus propias aplicaciones y en el momento en el que sean indispensables.

Un elemento esencial del curso es el énfasis en realizar experiencias utilizando la computadora en clase. *"Si lo escucho, lo olvido; si lo veo lo recuerdo; pero si lo hago lo entiendo"* [21]. La idea es tratar de reconstruir el surgimiento de algunas de las ideas más importantes de la CG como un experimento. Esto ayuda a que se fijen los conceptos y puedan representarse mentalmente en forma conjunta las diferentes partes que se integran en la "tubería" (pipeline) de procesos que constituye una aplicación gráfica. Además, los programas desarrollados en clase luego son utilizados como punto de partida para la realización de los trabajos prácticos, lo cual produce cierto impulso hacia su pronta realización. Otro elemento esencial consiste en contar con un apunte electrónico que mapee fielmente el contenido de las clases teóricas. De esa forma se consigue que los alumnos aprovechen la clase para incorporar los conceptos, elaborarlos y cuestionarlos, dado que de todo ese proceso dialéctico queda un sedimento asimilado por comprensión y no por memoria.

2 Propuestas clásicas y modernas para la enseñanza de la CG

El enfoque tradicional para un curso introductorio de computación gráfica está orientado a brindar un conjunto de conceptos y técnicas a los estudiantes de Licenciatura en Ciencias de la Computación o de Ingeniería. Los objetivos básicos incluyen una adecuada cobertura de la organización del *hardware* y *software* de base gráficos, conceptos de representación y manipulación geométrica en 2D y 3D, presentación de información univaluada y bivaluada, y una introducción a temas de mayor complejidad como cara oculta, modelos de iluminación, aproximación de curvas y superficies y modelos procedimentales. Los requisitos mínimos para afrontar un curso de estas características consisten en un buen dominio de geometría analítica, algoritmos y estructuras de datos, y un manejo disciplinado de una plataforma de desarrollo de programas. Un curso de estas características cubre en un mínimo las expectativas de los alumnos, esencialmente porque el tiempo empleado en el desarrollo de los trabajos prácticos no permite afrontar los temas más avanzados. En su mayor parte, los temas de interés (como ser animación, juegos y sistemas interactivos) o de gran aplicabilidad (por ejemplo el modelado con curvas y superficies) no pueden ser adecuadamente cubiertos. Este era, sin embargo, el mejor contenido que se podía adecuar a un contexto como el existente hace unos 10 años, en el cual tanto las expectativas de los alumnos, la formación de los docentes y las posibilidades de equipamiento en las Universidades en eran mucho menores.

En la actualidad, tanto la motivación de los estudiantes como el manejo de computación que tienen en general, hacen que este planteo resulte totalmente insatisfactorio. Los grandes cambios tecnológicos y culturales han modificado el contexto general, haciendo que las expectativas puestas en un curso sean notablemente más sofisticadas. Uno de los primeros cambios discutidos fue publicado por el Comité de Educación de la ACM SIGGRAPH en 1995, donde se propuso una modificación sustancial a la currícula de los cursos de CG [17, 18, 20]. Se propone utilizar *software* comercial y programas de distribución libre para plataformas convencionales, para acelerar la comprensión de los principios fundamentales de la disciplina.

En esta propuesta el curso enfatiza esencialmente la producción de gráficos en 3D con realismo. Para ello, el punto de partida consiste en el aprendizaje de los aspectos físicos de la interacción de la luz con los diversos materiales, y su modelado computacional a través de algoritmos de *ray tracing*. Más adelante, o en cursos avanzados, los estudiantes toman contacto con el enfoque tradicional *scan-line* de la CG. Los alumnos utilizan desde el comienzo programas de *ray tracing* de distribución libre como el POV-RAY, para luego reprogramarlos o modificarlos según sus propios objetivos. Una vez manejados los conceptos de la primera mitad del curso, los alumnos aprenden el funcionamiento teórico de los algoritmos *scan-line*

tradicionales utilizando programas comerciales como el Renderman de PIXAR. De esa forma se familiarizan con el *pipeline* gráfico a través de una implementación particular.

Los contenidos mínimos propuestos para este y otros cursos similares [9, 16, 18, 23]) incluye principios físicos de la reflexión de la luz, la ecuación del *rendering*, modelos globales de iluminación, modelos sencillos de iluminación local (Phong), teoría e implementación de un algoritmo de *ray tracing*, métodos para acelerar el *ray tracing*, transformaciones en 2D y 3D, algoritmos *scan-line* para rectas y círculos, conversión *scan* de polígonos, cara oculta, mapeo de texturas y desplazamientos, curvas y superficies paramétricas, anti-*aliasing*, y modelos procedimentales (fractales, sistemas iterados).

Esta propuesta enfatiza la producción de gráficos con realismo, y por lo tanto exige desde el comienzo tener una noción adecuada del rendering y de los principios físicos involucrados. Un aspecto positivo es que se motiva a los estudiantes a producir imágenes que pueden ser comparables en calidad a las que perciben en juegos, publicidades, video, etc. Sin embargo, desde nuestro punto de vista, esta orientación tiene desventajas profundas que es necesario analizar. La más importante es que, a nuestro entender, el uso de sistemas preexistentes tiende a generar un contacto superficial con la problemática del tema, produciendo *usuarios* más que solucionadores de problemas. Si el objetivo es proveer entrenamiento para salidas laborales rápidas, éste enfoque puede ser adecuado, pero si el perfil de alumno que se busca es más académico, entonces es preferible que los alumnos comprendan lo antes posible la real complejidad de un sistema de CG, lo cual solo se logra con la experiencia en el desarrollo de programas propios. Otras desventajas provienen de la formación necesaria para poder aprovechar un curso de esta naturaleza. Desde el comienzo se insiste en los aspectos físicos de la luz, el anti-*aliasing* y los modelos de muestreo estocástico. Una asimilación adecuada de esta temática requiere que los alumnos conozcan óptica física, probabilidad, análisis de variable compleja y procesamiento de señales discretas. Es decir, el curso pasa a ser mucho más exigente en sus correlativas.

Otra propuesta más seductora fue realizada por Edward Angel en 1997 [2]. Angel propone centrar el dictado del curso en la librería OpenGL, la cual está disponible como API para la gran mayoría de equipos y sistemas operativos [12]. OpenGL surgió como un intento de Silicon Graphics por estandarizar el mercado de la programación de tarjetas gráficas. La idea fue proveer una interfase de software que permita simular por librería el comportamiento de las tarjetas gráficas provistas por sus equipos, las cuales implementan por *hardware* una proporción considerable del *pipeline* gráfico. Si bien dicho objetivo no se cumplió, la alianza entre Silcon y Microsoft permitió que OpenGL se incorporara como DLL estándard del Windows a partir de la versión 97, y desde entonces su uso se masificó, esencialmente entre los desarrolladores de juegos y sistemas gráficos interactivos bajo Windows. Un resultado de estos cambios tecnológicos fue que los programadores que utilizan plataformas que manejan la WINAPI estuvieron en condiciones de incorporar OpenGL a sus aplicaciones en forma libre y gratuita. Esto se da en forma natural en el lenguaje C, dado que OpenGL está implementado en este lenguaje y por lo tanto la interfase de software entre la API y la aplicación es similar a la de cualquier otra libreria escrita para C. Para las plataformas de desarrollo más primitivas, como por ejemplo DOS, se incluyo la GL Utility Toolkit (GLUT), que permite una incipiente programación de interfases gráficas dirigidas por eventos. De esa forma, es posible trabajar con una máquina modesta, una PC por ejemplo, como si contáramos con la tremenda potencia gráfica de una Silicon Graphics (doble *frame buffer*, doble *alpha buffer*, doble *z-buffer* de 24 bits, mapeo de texturas, sombrado de Gouraud, *scissors*, *stencils*, etc.).

Esta posibilidad evidentemente modifica los horizontes del dictado de un curso de CG. Por ejemplo, las arduas disquisiciones e implementaciones relacionadas con el problema de la cara oculta se vuelven irrelevantes al contar con *z-buffer*, la astucia requerida para optimizar un modelo de iluminación razonable se vuelve innecesaria al tener sombreado de Gouraud, o las complicaciones matemáticas de los modelos de curvas y superficies pueden ser eludidas si se las incluye como primitivas. La atención, entonces, se centra exclusivamente en el uso de la

librería para desarrollar aplicaciones gráficas específicas. De esa forma, es posible elaborar un curso de CG orientado a la programación, donde en relativamente poco tiempo los alumnos están desarrollando aplicaciones gráficas tridimensionales interactivas. Este es el enfoque sugerido por Angel en su libro [1]. Básicamente la propuesta consiste en aprender primero a utilizar las librerias y luego ver cómo funcionan (de ahí que su propuesta sea *top-down*).

Esta propuesta es seductora, porque da la apariencia de centrar el curso en aspectos tecnológicos de última generación. Además simplifica enormemente el trabajo de los docentes. Sin embargo hay algunos puntos que pueden ser insatisfactorios a largo plazo, muchos de los cuales son semejantes a las objeciones que hicimos a la propuesta de Owen. Volvemos a correr el riesgo de ser usuarios sin demasiado criterio propio, podemos confundir las posibilidades y limitaciones de OpenGL con las de la CG en general. Le prestamos poca o ninguna atención a la riqueza conceptual teórica de la computación gráfica y a sus aspectos algorítmicos. Además, en nuestro caso particular de un curso intensivo, corremos el riesgo de que el tiempo alcance solamente para la parte *top* del enfoque *top-down* (si es que el *top* está lo suficientemente claro, lo cual no siempre se cumple). Como el desarrollo se basa en una librería, no parece necesario realizar investigación en el tema (se crea la falsa expectativa de que una librería o un *hardware* específicos solucionarían todos los problema). Temas de gran importancia que no pueden ser manejados con el modelo de iluminación de OpenGL (como por ejemplo iluminación especular, interreflexión, iluminación global, rendering de volúmenes, etc.) pueden ser mencionados solamente de una forma superficial porque escapan a las posibilidades de la librería.

3 Descripcion del curso intensivo

Como mencionáramos, el objetivo curricular del curso propuesto requiere un énfasis en la parte algorítmica. Al mismo tiempo, como decisión personal, elegimos no renunciar a introducir a los alumnos a los temas avanzados y algunas líneas acutales de investigación. Esto requiere del alumno un entendimiento profundo del tema, lo cual únicamente se logra en base al desarrollo de un sistema gráfico de cierta complejidad. Sin embargo, es necesario buscar una forma de disminuir el costo en tiempo que tal desarrollo implica, agilizando al máximo la elaboración de los programas interactivos por parte de los alumnos. Una parte de este objetivo se logra utilizando una herramienta de programación tipo RAD, como Delphi o C++ Builder, o sus versiones para LINUX.

El primer caso es normalmente el que utilizamos, dado que en su gran mayoría los alumnos de Ciencias de la Computación han desarrollado sus habilidades como programadores en el lenguaje Pascal, y conocen en detalle el Turbo Pascal. Para aquellos alumnos un poco rezagados en los detalles de implementación, llevamos adelante al mismo tiempo los ejemplos en Pascal para DOS, utilizando los drivers VESA más modernos que permiten utilizar con mayor poder a las actuales tarjetas gráficas. Esto garantiza cierta "compatibilidad" con los alumnos que llevan algunos años de graduados, y que no se han familiarizado con las tecnologías más modernas. Eventualmente, durante este trabajo la gran mayoría comprende las ventajas del desarrollo de aplicaciones en Delphi por la experiencia, y el Pascal puede ser abandonado rápidamente.

Los alumnos cuentan con un apunte electrónico conciso que sigue casi textualmente el contenido de las clases. De esa forma pueden prestar atención al desarrollo de ejemplos, y participar activamente (muchas veces para corregir errores voluntarios o involuntarios del docente!). Los ejemplos vistos en clase también están disponibles *on-line* y forman la base de los trabajos prácticos que se deben realizar. De esa manera los alumnos se sienten capacitados para el desarrollo de sistemas de mediana envergadura a partir de las propuestas elaboradas en clase, y adquieren experiencia en la discusión de las ideas, y muchas veces realizan observaciones que contribuyen al enriquecimiento de la cátedra. Afortunadamente

es posible también programar a OpenGL desde Delphi con poco esfuerzo (ver [10]), y por lo tanto los alumnos ven el pasaje de la programación de sus propias primitivas al uso de librerías como un paso natural y necesario. A tal efecto, se desarrolló una componente para manejar OpenGL desde Delphi sin inconvenientes. En síntesis, la propuesta es brindar los conocimientos básicos, experiencia, y las herramientas necesarias para continuar con un autoaprendizaje. De esa forma, es posible introducir a los alumnos a la problemática teórica y algorítmica de la computación gráfica de una manera cercana al enfoque tradicional, pero incorporando más adelante el uso de librerías.

Los contenidos del curso no difieren, en su introducción, del contenido tradicional. Se agrega como tema de importancia el modelado con curvas y superficies modeladas con puntos de control [6], Otro tema que es profundizado es el de los modelos de iluminación y sombreado, incluyéndose una introducción al *ray tracing* y otros modelos no locales como radiosidad y la ecuación del rendering. Se finaliza con una exposición breve, enfocada hacia el interés de los alumnos, sobre sistemas de animación, métodos procedimentales (fractales, modelos gramáticos), visualización científica y otros temas avanzados. La lista de contenidos agrupados por trabajo práctico es la siguiente:

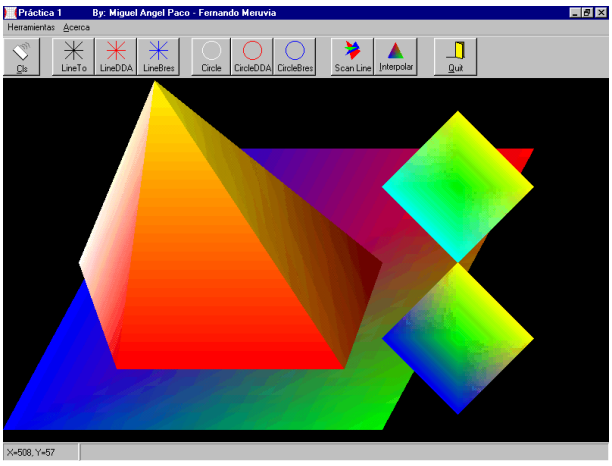
- Introducción a los sistemas de *hardware* y *software* gráficos de una computadora. Desarrollo de programas gráficos de base (líneas y círculos). Conversión-*scan* de polígonos.
- Geometría básica en 2D. Transformaciones. Coordenadas Homogéneas. Modelado jerárquico de escenas. *Clipping* y *windowing*.
- Modelado con curvas (Bézier, B-Splines recursivo y cúbico uniforme, curvas racionales).
- Visión. Teoría del color. Espacios cromáticos.
- Geometría en 3D. Transformaciones. Proyecciones y perspectiva. Línea y cara oculta.
- Modelos locales de iluminación y sombreado.
- OpenGL.
- Modelado con superficies (Bézier y B-splines).
- Modelos de iluminación scan-line avanzados. Mapeo de atributos.
- Modelos de iluminación no locales (*ray-tracing*, radiosidad, ecuacion del rendering).
- Introducción a los modelos procedimentales.
- Introducción a los sistemas de animación.
- Introducción a la visualización científica.

Con respecto a los trabajos prácticos, se enfatiza su desarrollo en comisiones de dos personas, de modo de poder distribuir la carga de la implementación pero no por ello desconocerla por completo. Se atiende fundamentalmente el aspecto creativo en los trabajos prácticos, no prohibiéndose el intercambio razonable de módulos entre las comisiones. Como los programas de los primeros prácticos constituyen la base de los prácticos siguientes, los alumnos aprenden por experiencia las bases y dificultades de la ingeniería de *software*, sobre todo en lo concerniente al reuso y portabilidad de sistemas.

4 Trabajos prácticos

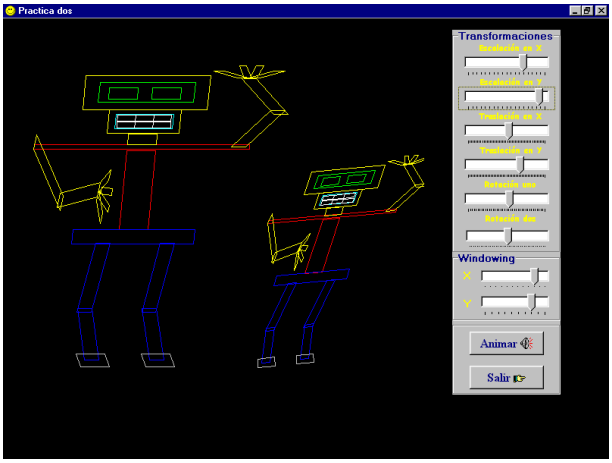
En esta Sección describiremos los contenidos y metodología utilizada en los trabajos prácticos. Mostraremos también los resultados obtenidos por los alumnos de la UMSS durante el cursado del año 2000. El contenido del curso, el apunte utilizado, y un grupo seleccionado de programas ejecutables de los trabajos prácticos están disponibles en `www.ingelec.uns.edu.ar/umss`. El listado completo de los trabajos prácticos y una ilustración de los mismos realizada por alumnos es el siguiente:

1. Algoritmos básicos.



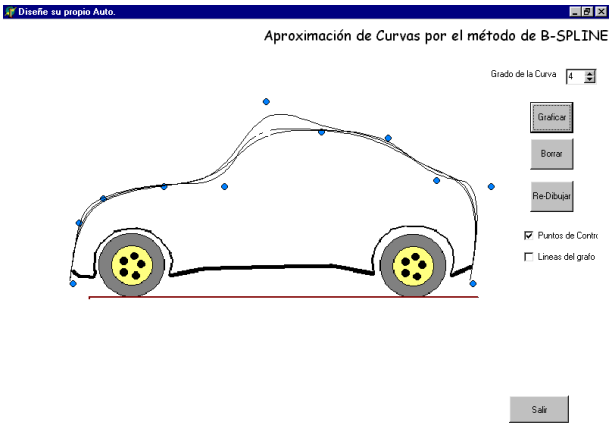
   - Implementar los algoritmos de segementos de recta y círculos por DDA y Bressenham contemplando todas las simetrías y casos posibles.
   - Implementar la conversión-*scan* de triángulos.
   - (optativo): Implementar la conversión-*scan* anterior con color interpolado (sombreado de Gouraud).

2. Transformaciones 2D.



   - Implementar transformaciones en 2D en coordenadas homogéneas.
   - Implementar un pequeño sistema que permita modelar objetos 2D por medio de transformaciones estructuradas.
   - (optativo) Implementar *windowing* utilizando el *clipping* de Cohen-Sutherland.

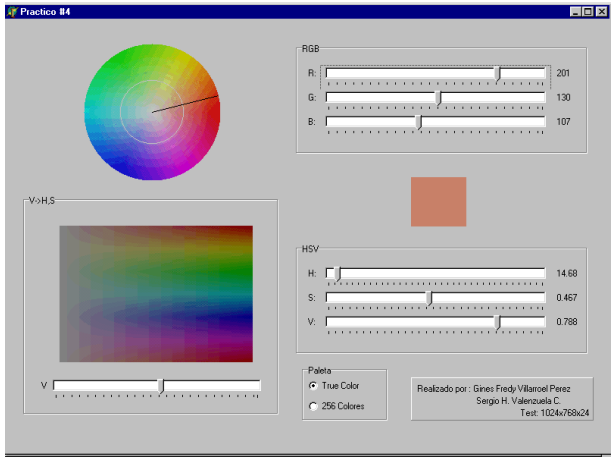
3. Aproximación de curvas.



   - Implementar el ajuste de curvas de Bézier, B-Splines recursivo, y B-Splines cúbicos uniformes.
   - Aplicar los algoritmos en un sistema que permita modificar interactivamente las curvas para resolver algún problema práctico.
   - (optativo) Implementar curvas racionales para los métodos anteriores.

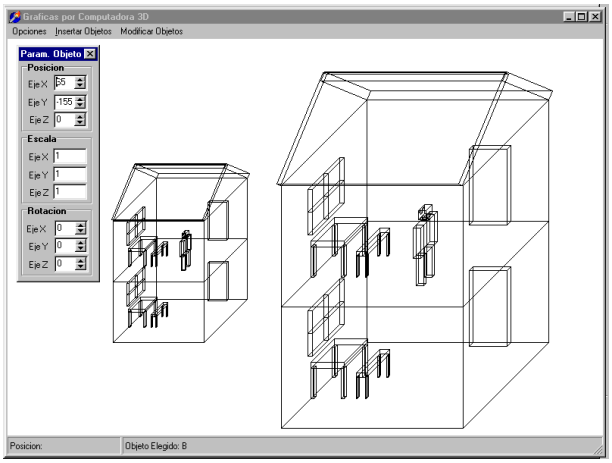
4. Color y espacios cromáticos.

   - Implementar la conversión entre espacios cromáticos RBG y HSV (CSV) y viceversa. Utilizarlo para pintar un polígono variando su cromaticidad (*hue*) según su altura, y su saturación según su ancho.
   - Rehacer el ejercicio anterior utilizando una paleta estática de 256 colores.



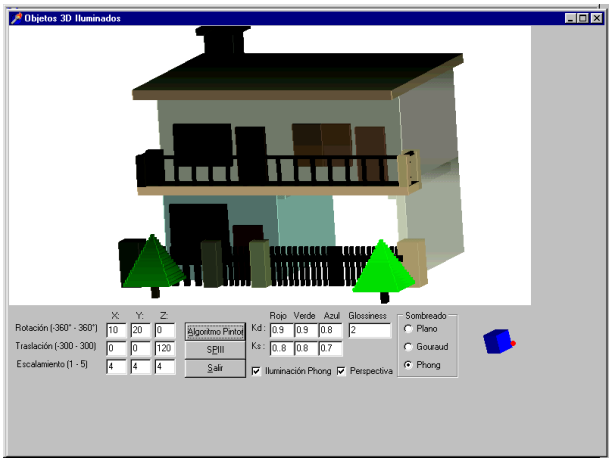
5. Transformaciones 3D.

   - Implementar un pequeño sistema que permita modelar objetos 3D por medio de transformaciones estructuradas, utilizando proyecciones ortográficas.
   - Modificar el ejercicio anterior para utilizar perspectiva.
   - Agregarle al ejercicio anterior eliminación de caras por *backface-culling* y por prioridades.
   - (optativo) Implementar algún otro método de cara a oculta a elección.

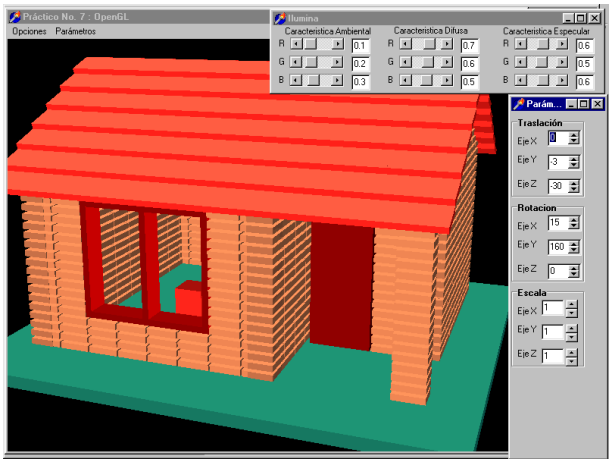


6. Modelos de Iluminación.

   - Agregar al ejercicio 5 un modelo de iluminación con luz ambiente y reflexión difusa, utilizando sombreado plano.
   - Agregar al ejercicio anterior sombreado interpolado y modelo de iluminación de Phong.
   - (Optativo) Agregar sombreado de Phong.



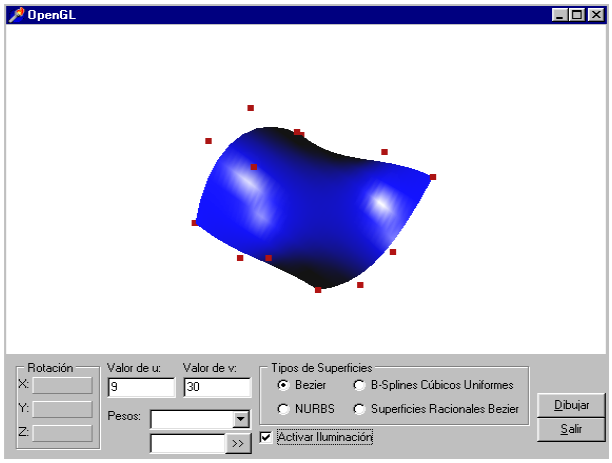
7. OpenGL.



- Realizar una pequeña *GUI* interactiva con una ventana OpenGL que permita manipular una escena (rotar, trasladar, etc.)
- En dicha aplicación repetir los ejercicios 1 b), 1 c), 2 b), 5 a), 5 b), 6 a) y 6 b).
- (optativo) Proponer una forma de poder acercarse al sombreado de Phong en OpenGL.

Observación: del práctico 8 en adelante utilizar OpenGL cuando resulte adecuado.

8. Aproximación de superficies.



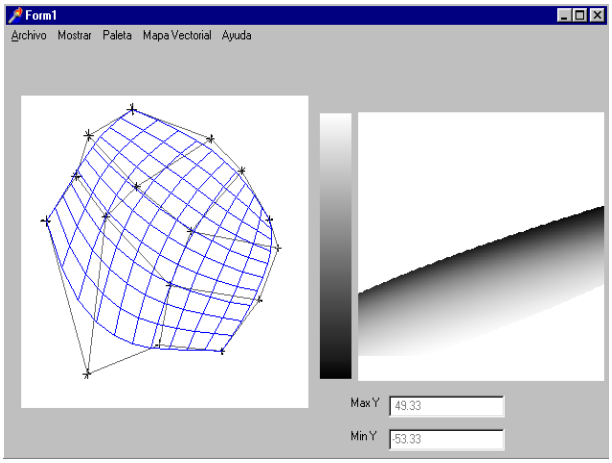
- Implementar el modelado de superficies con puntos de control con los métodos de Bézier y B-Splines cúbicos uniformes.
- Aplicar los algoritmos en un sistema que permita modificar interactivamente las superficies para resolver algún problema práctico.
- (optativo) Implementar superficies racionales de Bézier y NURBS.

9. Modelos *scan-line* avanzados.

- Agregar al ejercicio 8 el mapeo de texturas 2D.
- Utilizar el modelo de iluminación de OpenGL para simular efectos atmosféricos (atenuación, niebla, etc.).
- (optativo) Proponer una forma de utilizar los buffers adicionales para proyectar sombras de objetos sobre otros objetos.

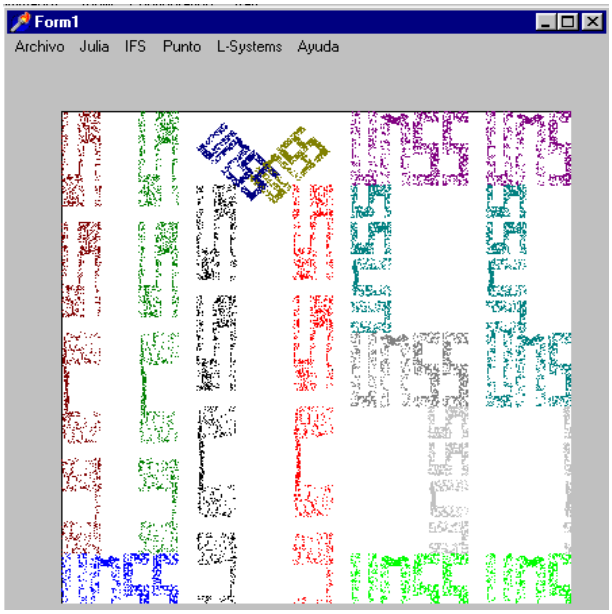
10. Visualización científica.

    - Proponer paletas de pseudocoloring para representar mapas escalares 2D como altitudes, temperaturas, esfuerzo (con umbral), nivel de contaminación, etc.
    - Representar mapas vectoriales 2D y proponer un mecanismo para representar mapas 2D de más de 2 valores.
    - (optativo) Implementar el algoritmo de *marching cubes* para representar volúmenes por isosuperficies.



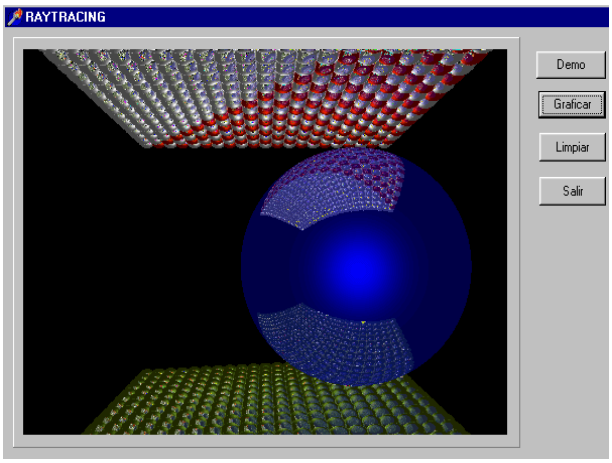
11. Fractales y modelos procedimentales.

    - Implementar los conjuntos de Julia y Mandelbrot. Bajar el FRACTINT de Internet.
    - Implementar superficies fractales con el algoritmo de desplazamiento aleatorio del punto medio.
    - (optativo) Utilizar fractales como texturas para mapear sobre superficies de objetos.
    - Implementar el algoritmo para sistemas de funciones iteradas y generar algún objeto natural.
    - Implementar un pequeño algoritmo creación de fractales con gramáticas y generar algún objeto natural.



12. *Rendering*

    - Implementar un pequeño algoritmo de *ray-tracing* para esferas.
    - Conseguir el POV-RAY en Internet y repetir el ejercicio anterior.
    - (Optativo) Conseguir RADIANCE en Internet y repetir el ejercicio.



Respecto del rendimiento de los alumnos, podemos decir que satisfacieron las mejores expectativas posibles. Al curso en la UMSS se inscribieron 51 personas, muchas de las cuales tenían otras obligaciones académicas y laborales. De ellos 4 obtuvieron categoría A, 15 categoría B, 20 categoría C y 4 categoría D. Los restantes abandonaron generalmente por razones laborales. Es necesario tener en cuenta que los alumnos de la maestría toman alrededor de 8 horas diarias de clase durante dos semanas, y luego entregan los exámenes o prácticos por mail, por lo que normalmente están con tarea pendiente de dos o más cursos (además de sus ocupaciones laborales!).

La experiencia realizada fue lo suficientemente positiva como para servir de base en dos cursos de similares características dictados en la Universidad Mayor San Andrés de La Paz en Diciembre de 2000 y en la Universidad Autónoma Gabriel Moreno de Santa Cruz de la Sierra, durante Enero de 2001. Si bien los alumnos de estos cursos están todavía realizando sus prácticos, es posible anticipar que su rendimiento será similar al de sus colegas de Cochabamba. Es importante destacar en los tres casos que los grupos humanos son excepcionalmente hospitalarios y cordiales, además de estar predispuestos a trabajar y a aceptar las pautas del docente.

5 Conclusiones

En este trabajo se presentaron alternativas para la enseñanza de un curso intensivo de Computación Gráfica para alumnos de Maestría en Ciencias de la Computación en la Universidad Mayor San Simón, Cochabamba, Bolivia. El es similar en estilo a un curso tradicional, pero incorpora el uso de librerias y sistemas abiertos de una manera gradual, en el momento en que los trabajos prácticos realizados ya le han dejado al alumno una clara comprensión formativa de los distintos elementos teóricos y de implementación que constituyen una aplicación gráfica. En la dinámica del curso se utiliza la computadora en clase, de manera de plantear los problemas teóricos como *experimentos*. Esto ayuda a que los alumnos fijen los conceptos y puedan representarse mentalmente en forma conjunta las diferentes partes que constituyen una aplicación gráfica. Los programas desarrollados experimentalmente en clase luego son utilizados como punto de partida para la realización de los trabajos prácticos, lo cual produce cierto impulso positivo. También se cuenta con un apunte electrónico, lo que permite que los alumnos aprovechen mejor la clase para incorporar los conceptos.

Se presentaron los contenidos teóricos, los trabajos prácticos, y se dejó la URL con los materiales del curso. También se mostraron algunas resoluciones destacadas de los trabajos prácticos. Los resultados de aprendizaje y entusiasmo conseguidos fueron mayores a los planteos más optimistas, y la experiencia de Cochabamba fue repetida en La Paz y Santa Cruz con resultados similares, lo que permite concluir que la experiencia reseñada es una buena alternativa para tener en cuenta en la organización general de cursos en todos los niveles.

Agradecimientos: La posibilidad de llevar adelante la experiencia reseñada en este trabajo representó para nosotros una valiosa vivencia, tanto docente como humana. Esto no hubiese sido posible sin la intervención de Pablo Azero, y de todo el grupo de personas en Cochabamba, a quien también agradecemos infinitamente. No queremos olvidar aquí a los coordinadores de las Maestrias en La Paz y Santa Cruz, Fátima Dolz y Julio Zeballos, por confiarnos la posibilidad de repetir la experiencia de Cochabamba, así como a Teresa Ledezma, Ximena García y Claudia Ibarra, por el apoyo brindado durante nuestra estadía. Queremos también agradecer a Gustavo Ramoscelli por su invalorable ayuda tanto con el desarrollo de la programación de OpenGL desde Delphi y C++, como con la actualización de los drivers VESA para modos gráficos avanzados.

Referencias

[1] Edward Angel. *Interactive Computer Graphics: A Top-Down Approach with OpenGL.* Addison-Wesley, 1997.

[2] Edward Angel. Teaching a Three-Dimensional Computer Graphics Class Using OpenGL. *ACM Computer Graphics*, 31(3):54–55, 1997.

[3] S. Castro, C. Delrieux, y A. Silvetti. Computación Gráfica en la Universidad Nacional del Sur. En *VII Ateneo de Profesores Universitarios de Computación*, Tandil, Argentina, 1999.

[4] T. A. Defanti, M. D. Brown, y B. H. McCormick. Visualization: Expanding Scientific and Engineering Research Opportunities. In G. M. Nielson y B. D. Shriver, editores, *Visualization in Scientific Computing*, pages 32–47. IEEE Computer Society Press, 1990.

[5] Claudio Delrieux, Silvia Castro, y Andrea Silvetti. Enseñanza Introductoria y Avanzada en Computación Gráfica. In *Congreso Argentino de Ciencias de la Computación*, páginas 347–358, Bahía Blanca, Argentina, 1995. I CACiC.

[6] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, New York, 1988.

[7] J. Foley, A. Van Dam, S. Feiner, y J. Hughes. *Computer Graphics. Principles and Practice.* Addison-Wesley, Reading, Massachussetts, segunda edición, 1990.

[8] W. K. Giloi. *Interactive Computer Graphics - Data Structures, Algorithms, Languages.* Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

[9] Scott Grissom, Jack Bressenham, Bill Kubitz, G. Scott Owen, y Dino Schweitzer. Approaches to Teaching Computer Graphics. En *SIGCSE '95 Proceedings*, páginas 382–383, Nashville, TN, 1995. ACM SIGCSE, ACM Press.

[10] Jon Jacobs. *Delphi Developer's Guide to OpenGL.* Wordware Publishing, USA, 1999.

[11] Peter Keller y Mary Keller. *Visual Cues: Practical Data Visualization.* IEEE Computer Society Press, Los Vaqueritos, CA, 1990.

[12] M. J. Kilgard. The OpenGL Utility Toolkit Programming Interface. Silicon Graphics, 1995.

[13] John Lowther y Ching-Kuang Shene. Rendering + Modeling + Animation + Postprocessing = Computer Graphics. *ACM Computer Graphics*, 34(5):15–20, 2000.

[14] W. Newman y R. Sproull. *Principles of Interactive Computer Graphics.* McGraw-Hill, 1979.

[15] G. M. Nielson y B. D. Shriver. *Visualization in Scientific Computing.* IEEE Computer Society Press, Los Alamitos, CA, 1990.

[16] G. Scott Owen. A Workshop on Computer Graphics for Undergraduate Faculty. *Computer Graphics*, 25(3):179–185, 1991.

[17] G. Scott Owen. ACM SIGGRAPH Education Committee Activities for Computer Graphics Educators. *Computer Graphics*, 28(3):179–182, 1994.

[18] G. Scott Owen, Maria Larrondo-Petrie, y Cary Laxer. Computer Graphics Curriculum: Time for a Change? *Computer Graphics*, 28(3):183–185, 1994.

[19] L. Rosenblum. Scientific Visualization at Research Laboratories. *IEEE Computer*, 22(8):68–100, 1989.

[20] Karen Sullivan. A Second Generation Computer Graphics Course for Mathematics and Computer Science. *Computer Graphics*, 28(3), 1994.

[21] Lao Tzú. *Tao Te Ching*, el Libro del Recto Camino. Editorial Oasis, Navarra, 1995.

[22] Alan Watt y Mark Watt. *Advanced Animation and Rendering Techniques.* Addison-Wesley, London, 1992.

[23] Zhigang Xiang. A Nontraditional Computer Graphics Course for Computer Science Students. *Computer Graphics*, 28(3):186–188, 1994.