
# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

## MAESTRÍA EN ARTES VISUALES

## LA VISUALIZACIÓN DE LAS FÓRMULAS MATEMÁTICAS

**TESIS**
**QUE PARA OPTAR POR EL GRADO DE:**
MAESTRO EN ARTES VISUALES

**PRESENTA:**
LIC. VICTOR HUGO HERNÁNDEZ ALCÁNTARA

**MIEMBROS DEL COMITÉ TUTOR**

MTRO. FRANCISCO ESTRADA RODRÍGUEZ

MTRO. SALVADOR JUÁREZ HERNÁNDEZ

MTRO. MANUEL ELÍAS LÓPEZ MONROY

MTRO. JUAN CARLOS MIRANDA ROMERO

DR. EUGENIO GARBUNO AVIÑA

MÉXICO D.F. SEPTIEMBRE DE 2014

**AGRADECIMIENTOS**

Agradezco a mi tutor el Maestro Francisco Estrada Rodríguez por ser mi mentor y guía durante todo este proceso; por compartir conmigo su conocimiento y experiencia. Por su paciencia, por sus sabias palabras. Por introducirme en ésta apasionante línea de investigación y por su legado a nuestra comunidad de comunicadores y artistas visuales.

Agradezco el apoyo de todos mis Maestros, a quienes siempre recuerdo con mucho cariño y admiración por compartir sus conocimientos, y por contribuir a mi formación profesional.

Agradezco al programa de becas para estudio de posgrado de la UNAM, pues hacen posible la realización de estos trabajos de investigación.

Agradezco a nuestra máxima casa de estudios la UNAM. A la ENAP ahora Facultad de Artes y Diseño, y también, a la Academia de San Carlos, por ser espacios únicos para la reflexión, la formación, el conocimiento y la creación, en las artes y el diseño.

Agradezco a mi Madre Ana María Alcántara Montes por brindarme todo su amor y cuidados desde mi nacimiento, dándome todo lo necesario con tantos sacrificios, tiempo, dinero y esfuerzo, para mi crianza y formación. Te dedico este trabajo Madre, te amo y te admiro mucho.

Agradezco a mis hermanos, Alfredo Hernández Alcántara y Jorge Alberto Hernández Alcántara por ser un ejemplo de esfuerzo, trabajo y lucha constante para salir adelante en la vida y en lo profesional. Estoy muy orgulloso de ustedes.

Agradezco a mi esposa Mónica Castellanos Hernández por su apoyo y comprensión durante este proceso y a mi hijo Fernando Mateo Hernández Castellanos por ser una fuente de motivación para ser un mejor hombre y un mejor padre.

Agradezco a Dios y a la vida por permitirme consumar este logro, y también a mis familiares ausentes. A la memoria de mis abuelos: José Alcántara Parra y Rosa Montes Andrade.

Agradezco a todas aquellas personas que de una u otra manera, me brindaron su ayuda y consejo. Amigos, profesores, compañeros de universidad y familiares. Muchas gracias a todos.

# *Dedicatorias:*

A mi madre quien con su amor, esfuerzo y ejemplo nos hizo hombres de bien a mi y a mis hermanos.

## *Dedicatorias:*

A Mónica mi esposa y a mi hijo Fernándo Mateo, a quienes amo con todo mi corazón.

## *Dedicatorias:*

A mis hermanos Alfredo y Jorge Alberto; por ser mis amigos y compañeros de vida.

## *Dedicatorias:*

A mis maestros, quienes han compartido conmigo su sabiduria y conocimientos.

# LA VISUALIZACIÓN DE LAS FÓRMULAS MATEMÁTICAS.

ÍNDICE:

# Introducción

Este proyecto de investigación es básicamente una propuesta que nace de una necesidad, un interés por mejorar mi propia producción multimedia. A raíz de que elaboré el sitio web de la biblioteca de Universum (museo de las ciencias) y sus contenidos, tuve la inquietud de elaborar interactivos didácticos que explicaran ciertos modelos matemáticos o fenómenos de la física para enriquecer el sitio web y hacerlo más interesante, y por falta de conocimientos relacionados a la ciencia y la programación, no pude realizarlos. Por ello, desde el comienzo él propósito de ésta investigación, fue proponer a los diseñadores de la comunicación visual con especialidad en multimedia, que involucren éstas disciplinas en su hacer profesional, pues considero basado en mi propia experiencia, recopilación de ejemplos similares y trabajo de investigación, que la aplicación de las matemáticas, y otras ciencias en su conjunto con los lenguajes de programación, en definitiva, ayudan al diseñador multimedia a mejorar la producción de sus trabajos interactivos.

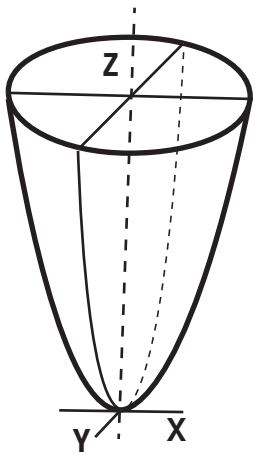
En resumen el tema es: Las matemáticas aplicadas al diseño interactivo y trata de matemáticas interactivas, la documentación y producción de tres ejemplos los cuales, he desarrollado con dedicación y paciencia, y una investigación documental de casos éxito con este tipo de aplicaciones, publicadas en sitios web y otros documentos digitales.

Otros temas que se abordan en este trabajo de investigación son: La visualización de lo abstracto, tomando como marco teórico, las ideas de algunos de los principales teóricos de la visualización, y también, el diseño de interacción en arte y diseño. Agradezco la ayuda del Maestro Marco Antonio Esquivel profesor de matemáticas de la Facultad de Ciencias de la UNAM por aclarar mis dudas y alentar mí proceso, y también, la ayuda de mi tutor y mis sinodales, por su consejo y asesoría. Con esta breve introducción, debo agregar que el único objetivo de éste trabajo, con toda humildad, es ser un material de utilidad para mis colegas de diseño, específicamente aquellos especializados en multimedia.

# I. LA VISUALIZACIÓN DE LO ABSTRACTO

# I. LA VISUALIZACIÓN
## DE LO ABSTRACTO.

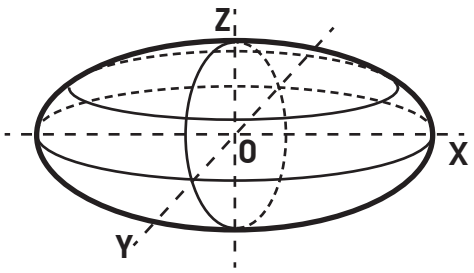
Visualizar de manera abstracta, es una habilidad que todo diseñador de la comunicación visual debe desarrollar, es una capacidad que nos ayudará a comprender y explicar a través de imágenes; ideas y conceptos complejos que de otro modo, son abstracciones puras expresadas en fórmulas matemáticas, códigos o símbolos, muchos de ellos incomprensibles para la mayoría de los seres humanos. Fig 1.1. Representación de un paraboloide elíptico.




$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = cz$$

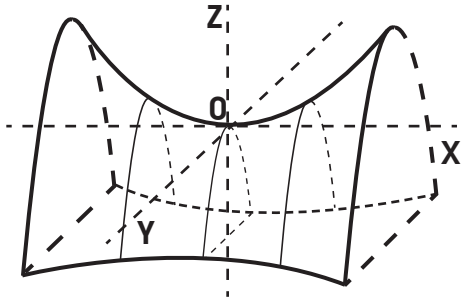
*Fig. 1.1*

Si buscamos en un diccionario el significado de la palabra visualizar, encontraremos definiciones como ésta: *"Visualizar es hacer visible lo que no puede verse a simple vista"*. O bien, *"Visualización: Es la acción y/o efecto de visualizar y representar mediante imágenes ópticas, fenómenos de otro carácter"; (Diccionario de la lengua española. Consulta realizada en mayo de 2012).* Ambas, son definiciones de diccionario, sin embargo, se hace referencia de manera directa o indirecta, al uso de la imagen, por lo tanto, la importancia de las imágenes para visualizar, va a radicar, en su valor como medio de comunicación, por su capacidad para transmitir ideas, conceptos, abstracciones, fórmulas, leyes, etc. Además, encontramos también este término en otras disciplinas como la psicología. Para los psicólogos, visualización es una técnica que pretende la reestructuración profunda de ciertos aspectos del subconsciente ligados a estados mentales y emocionales de la psique, uno de los principales pioneros de éste conocimiento fue Eric Berne en los años 50's. Fig 1.3. Representación de un paraboloide hiperbólico.




$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$$

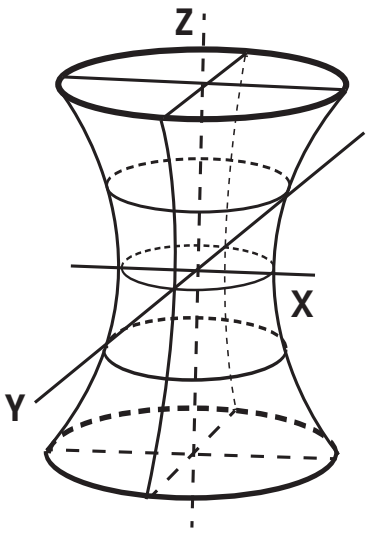
*Fig. 1.2*




$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = cz$$

*Fig. 1.3*

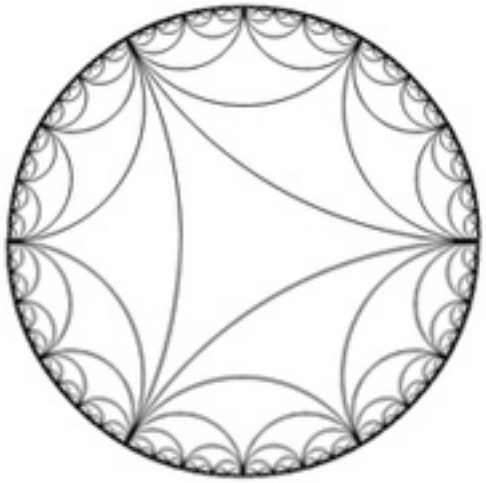
En esta misma línea de investigación encontramos a Miguel de Guzmán, profesor de psicología en la Universidad Complutense de Madrid. Miguel de Guzmán afirma que hoy en día es inseparable la relación que existe entre razonamiento humano y visualización, y ya que hablamos de psicología evolutiva y psicología del desarrollo, tenemos que mencionar las ideas de otro importante teórico de la visualización: Vygotsky. Para Vygotsky; "*El pensamiento en tanto abstracción pura, necesita de un medio material para concretarse y hacer posible su comunicación*. *Y el medio material por el cual puede ser expresado el pensamiento, es básicamente: El lenguaje."* Lenguaje en sus diversas formas: lenguaje gestual, sonidos, signos, el lenguaje visual (conformado de imágenes, diagramas, gráficos o figuras), el lenguaje simbólico, y el de las palabras. (**Vygotsky, Lev.** (1999), *Pensamiento y Lenguaje.* México: Quinto sol.) Si comparamos las ideas de ambos autores, podemos entender que visualización es básicamente, una forma de lenguaje visual que nos ayuda a revelar de manera más clara, otras formas de pensamiento abstracto para facilitar procesos del conocimiento en disciplinas complejas como la física o las matemáticas. Fig 1.4. Hiperboloide.




$$\frac{x^2}{a^2}+\frac{y^2}{b^2}-\frac{z^2}{c^2}=1$$

***Fig. 1.4***

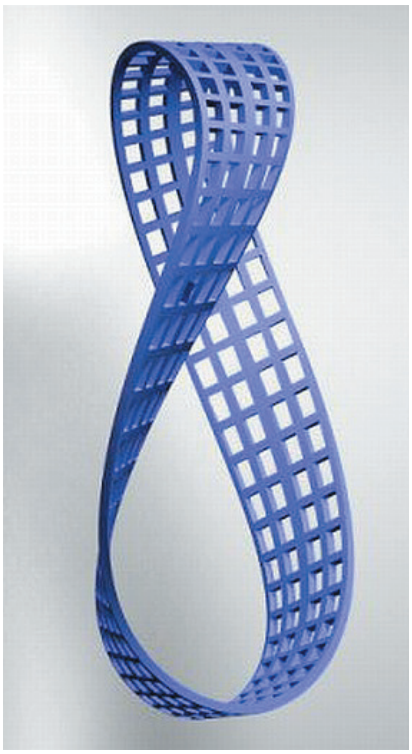
Por ello, diversos trabajos de investigación reportan el uso de la visualización como un recurso para la comprensión de conceptos complejos por medio de cinética y cinemática que simulan procesos a través de medios mecánicos o bien medios computacionales. Respecto a lo anterior, Cantoral nos dice: *"La visualización en ciencias duras como las matemáticas, la física o la química, facilita la comprensión de conceptos abstractos, razonamientos y teorías"* (**Cantoral, R. & Montiel, G.** (2001). *Visualización y pensamiento matemático*. México: Thomson Editores). Fig 1.5. Disco de Poincairé.



$$ds^2=\frac{dx^2+dy^2}{\left(1-x^2-y^2\right)^2}.$$

***Fig. 1.5***

Surge así la siguiente reflexión: Toda visualización es, una representación construida a partir de los datos generados en un proceso de investigación, un documento que busca capturar y comunicar los aspectos más críticos e importantes de un tema en cuestión adaptado al contexto o necesidad, de modo que pueden ser herramientas que facilitan la comprensión de las ideas, favorecen la toma de decisiones para el desarrollo de las capacidades intelectuales. Ese debería ser el objetivo de la mayoría de los trabajos de investigación que se producen en las universidades, sin embargo, muchas veces, no es el caso. Por ello: toda capacidad de visualizar conceptos o problemas, requiere también, la habilidad para interpretar y entender la información, manipularla mentalmente, y expresarla sobre algún soporte material o virtual. (Papel, pantalla, pizarrón, monitor de computadora o proyección) o bien, un proceso interno de la mente humana que aún no ha sido expresado en ninguna forma, pero es importante, no se quede sólo en el mundo de las ideas. Al respecto Cantoral dice: *"La visualización siempre estará asociada con: 1) Las representaciones visuales. 2) La habilidad para interpretar, transformar y comprender esas representaciones, 3) El desarrollo del pensamiento y el lenguaje en el individuo, para comunicar las ideas o conceptos intangibles, y 4) La imagen, el gráfico, o el interactivo como medio físico de una visualización, a través de representaciones externas (soportes tradicionales o digitales)".* Por ello es necesario que el estudiante en diseño y comunicación visual, desarrolle ésta capacidad, una habilidad perfeccionada por nuestros científicos, quienes se han esforzado por comprender el universo y sus fenómenos, dando a nuestra sociedad, las aportaciones más importantes del conocimiento en investigación, ciencias aplicadas y tecnología. Fig. 1.6. La transformación de Möbius y Fig. 1.7. Ilustración de Escher basada en la transformación de Möbius.



$$f(z)=\frac{az+b}{cz+d}$$

*Fig. 1.6*

*Fig. 1.7*

Para enriquecer aún más todo este argumento y relacionarlo a nuestra profesión, diremos que otros conceptos inseparables a la visualización: son el concepto de lo gráfico y el concepto diseño. Según Alejandro Tapia no existe un concepto único de diseño: *"El problema del diseño comienza desde su propia definición: los campos de estudio del diseño y sus enfoques son demasiado amplios, el diseño está presente en muchas disciplinas humanas, las definiciones más básicas de diseño son; dibujo, trazo, objeto, proceso o disciplina. La planeación para llegar al objetivo, el paso intermedio para obtener un producto final*" (**Tapia, Alejandro.** *El diseño gráfico en el espacio social.* pags. 17-19). Fig. 1.8. Esculturas matemáticas de Bathsheba Grossman. En cuanto al tema, Victor Margolín ha dicho: *"La carencia de teorías y metodologías para hacer diseño y la visión de diseño como un mero quehacer artesanal separado de otros procesos y pensamientos abstractos, ha provocado que el diseño no se consolidara como disciplina".* Luego nos dice: *"para superar esta debilidad conceptual del diseño, el diseñador deberá proyectarse tanto hacia dentro como hacia fuera de la profesión".* (**Tapia, Alejandro.** *El diseño gráfico en el espacio social.* Pag 22). Fig. 1.9. La partitura como representación visual de la música.

***Fig. 1.8***

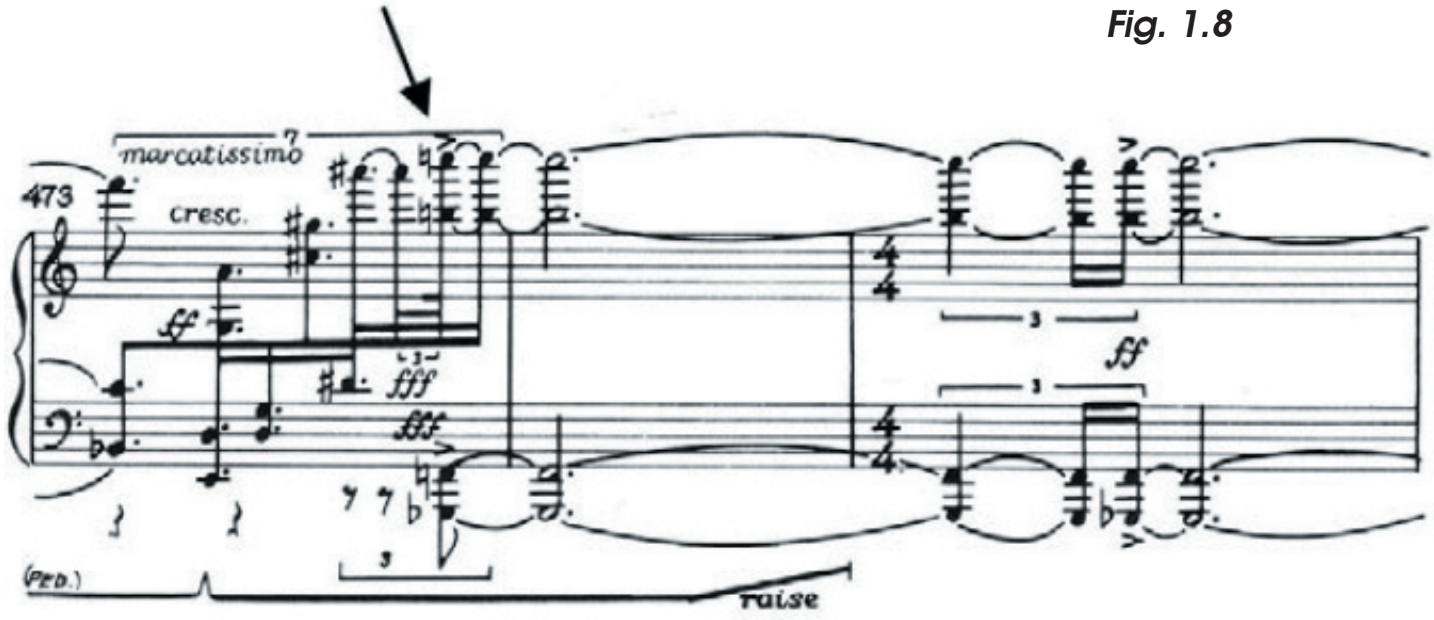



***Fig. 1.9***

Otro concepto importante de diseño es el de John Hesket: *“El diseño consiste en diseñar un diseño para producir un diseño”*. (**Hesket, John.** *Diseño en la vida cotidiana* pag 5). Así encontramos las siguientes definiciones: Diseño como dibujo, diseño como concepto, diseño como proceso, diseño como proyecto, diseño como planeación, diseño de modas, diseño publicitario, diseño de la comunicación visual, diseño arquitectónico, diseño industrial, diseño de ingeniería, diseño multimedia, diseño como visualización de lo abstracto, etc. A esto debo agregar las ideas de un último autor para concluir la ultima reflexión respecto a diseño. Frascara ha dicho: “*El diseño gráfico ha evolucionado de ser inicialmente un oficio a ser una profesión. Como oficio ha sido enseñado como conjunto fijo de destrezas y conocimientos. Como profesión, sin embargo, es necesario educar a los estudiantes para el futuro, y dado que el futuro resulta imposible de predecir, los estudiantes deben ser guiados básicamente para aprender a pensar, a enfrentar nuevas situaciones, y a construir, consecuentemente, respuestas apropiadas y creativas”.* (**Ricupero, Sergio.** *Diseño gráfico en el aula.* 2007, pág 10).

Ahora bien, revisemos por último el concepto de lo gráfico. Los gráficos, al igual que las imágenes, son recursos o elementos que nos ayudan a trasmitir una idea o una acción. Un gráfico puede ser también la representación gráfica de una operación, demostración o serie de ideas que tiene lugar por medio de imágenes, signos y símbolos. Según Galavis: *“Los gráficos pueden ser de diferente índole, de acuerdo a lo que traten de apoyar, así como de la dinámica que posean, por lo cual tenemos: 1) Los dibujos y esquemas: útiles para trabajar conceptos o ideas, para presentar un contexto y reafirmarlo.*

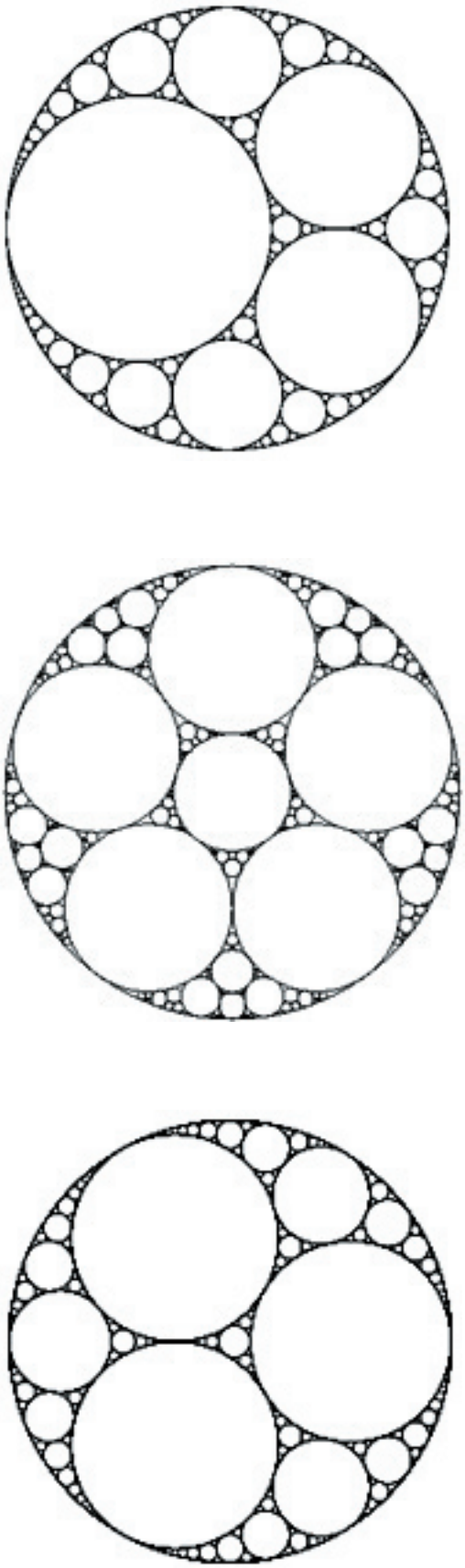


Fig. 1.10. Círculos de Apolonio, muchos de ellos, desarrollados por Gottfried Leibniz en homenaje a su colega matemático griego, Apolonio de Perga.

***Fig. 1.10***

*2) Las animaciones: Sirven para mostrar una secuencia o serie de procesos o como funciona algo.*

*3) Los diagramas: Éstos sirven para ilustrar las fases o relaciones entre partes o estados de un sistema.*

*Los diagramas de flujo indican los pasos y la lógica ligada al logro de una meta. Los llamados gráficos de transición; explican las relaciones entre los diversos estados de un sistema y las condiciones que produce la transición; las redes no cíclicas, muestran precedentes entre sus nodos; los diagramas de barras expresan duración y holgura. El tipo de diagrama que se vaya a utilizar no es arbitrario, depende de lo que se desea especificar.* Fig. 1.11. Superficies de Riemann asociadas al grupo genus.

*Y finalmente, 4) Los gráficos de tratamiento numérico; estos se utilizan cuando nos interesa comprender o manipular cifras, magnitudes o sus relaciones"*. Galavis, A. H. (1992): *Ingeniería del software educativo*, Santa Fé de Bogotá, Ediciones Uniandes, 1992. *Utilización de textos y gráficos en la enseñanza asistida por computador,* Universidad Central de las Villas, Santa Clara, Cuba, 2002. Fig. 1.12. Conjunto de Mandelbroth. Fig. 1.13. Conjuntos de Julia.

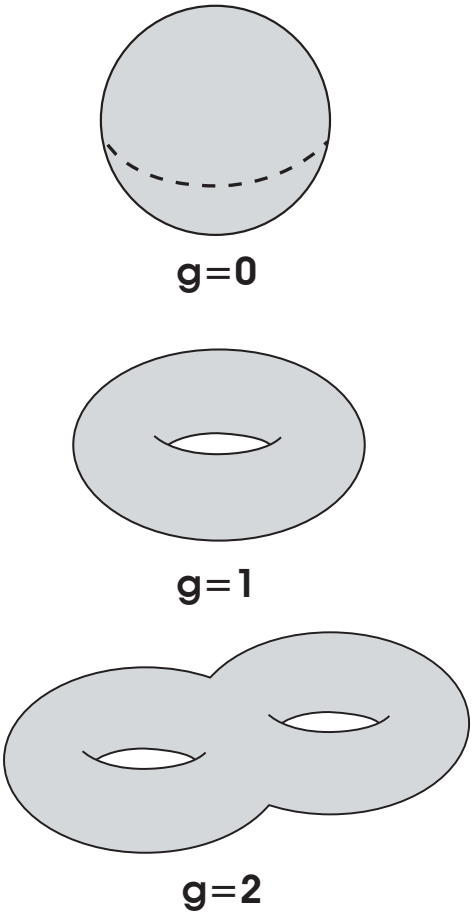



***Fig. 1.11***

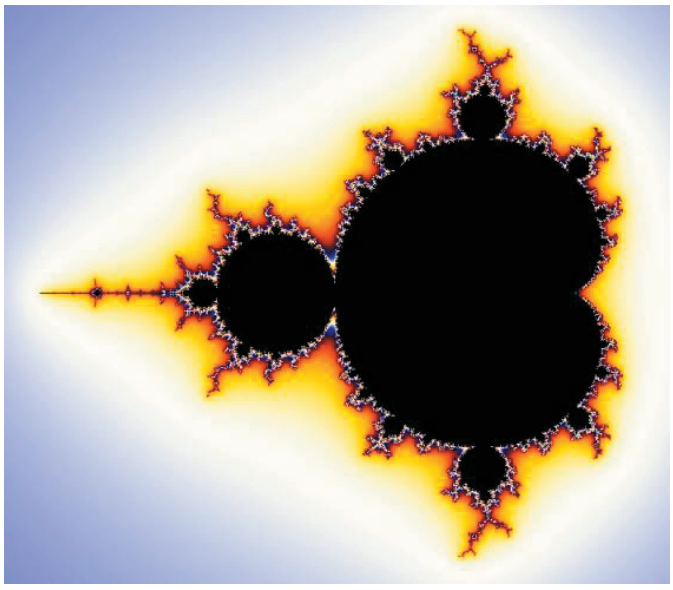


***Fig. 1.12***

$$Z = Z^2 + C$$

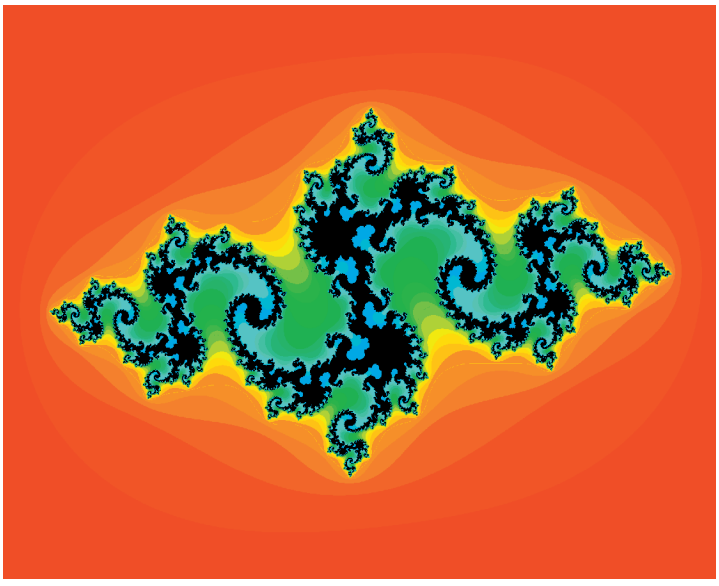


***Fig. 1.13***

$$FC(Z) = Z^2 + C$$

Los gráficos, son elementos indispensables en una visualización, por lo tanto; diseño gráfico, es una disciplina en tanto hace tangible, un concepto abstracto mediante comunicación visual, y cuando hablamos de comunicación visual, hablamos de signo visual, la construcción de una imagen con significado, una preocupación constante de la semiología; disciplina que se ocupa del estudio de los procesos mediante los cuales algo se utiliza como una representación de otra cosa, nada distinto a la visualización; un procedimiento conveniente en comunicación visual, cuando se busca representar también, otra cosa. En este sentido, visualización, como campo de investigación en diseño de interacción y diseño de videojuegos, tiene una creciente importancia en el estudio de ciencias duras y lenguajes de programación. Algo fundamental, si se entiende la naturaleza misma de éstas disciplinas. Fig. 1.14. Transformación de Möbius por Max Bill. Fig. 1.15. En esta figura podemos apreciar algunos diseños de la escuela de ULM, son trabajos de Tomás Maldonado. Fig. 1.16. Diseños de Arquímedes y Da Vinci basados en la espiral.

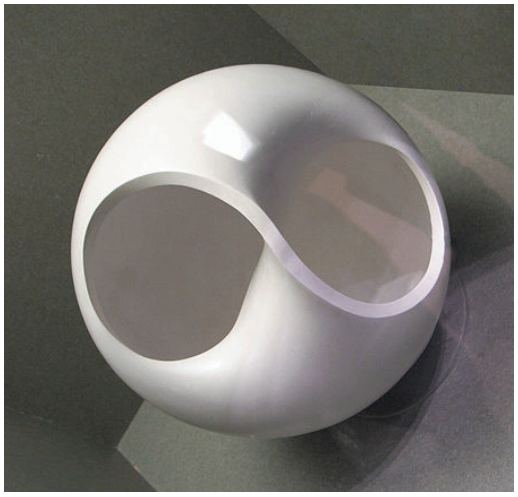
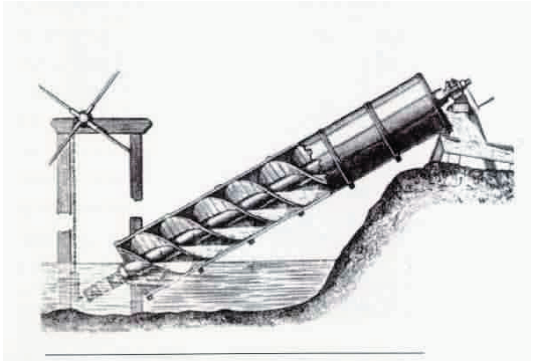


***Fig. 1.15***

***Fig. 1.14***



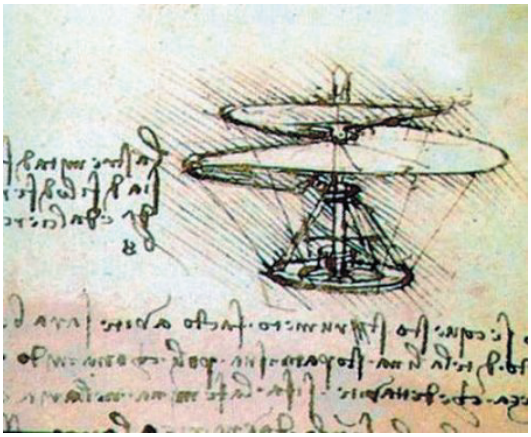


***Fig. 1.16***

En esta búsqueda, el uso de herramientas tecnológicas, se convierte en un medio útil para explorar éstas disciplinas de manera estática o interactiva, para facilitar el estudio de materias como: álgebra y geometría. La primera estudia las relaciones cuantitativas, y la segunda, las formas espaciales, cada una con sus propias construcciones y modelos. Ya por último y antes de finalizar este apartado, haremos una diferenciación entre visión y visualización. La visión es un proceso perceptivo y la visualización, un proceso que va mucho más allá de la percepción, y se ancla en lo racional-abstracto, de la compresión visual. Fig. 1.17. Pintura de Kandinsky. Fig. 1.18. Esta figura muestra una pieza de arquitectura líquida, también conocida como arquitectura de fluidos o del ciberespacio.

*Fig. 1.17*

Raymond Duval, hace una distinción muy importante entre visión y visualización: *"La visión es la percepción directa de un objeto espacial: la percepción visual de un sujeto observador, que aún no logra obtener la aprehensión completa del objeto. La visualización por otra parte, es la representación semiótica del objeto, una organización bidimensional de relaciones entre unidades"* Esto quiere decir que mediante visualización, cualquier organización puede ser sinópticamente comprendida como una configuración, haciendo visible y comprensible, todo lo que no es accesible a la visión, aportando una aprehensión global y total de cualquier organización de relaciones, sin embargo, su idea no queda allí, pues él plantea tres problemas desde el punto de vista del aprendizaje: 1) Discriminación de las características visuales relevantes; 2) El procesamiento figurativo, (descomponer, recomponer una figura; reconfiguración); cambio de perspectiva, y 3) El planteamiento discursivo. (**Duval, R.** *Representations and Mathematics Visualization.* Representation, vision and visualization: cognitive functions in mathematical thinking. Basic issues for learning. 2002). Pags. 311-335. Su metodología propone la comprensión total de un sistema partiendo de una construcción, deconstrucción y reconstrucción de un objeto para entender su esencia y lograr una profunda comprensión que nos permita representarlo, o bien, reinterpretarlo. Fig. 1.19. Entornos basados en arquitectura líquida. Fig. 1.20. Visualización en secuencia de números primos.

*Fig. 1.18*

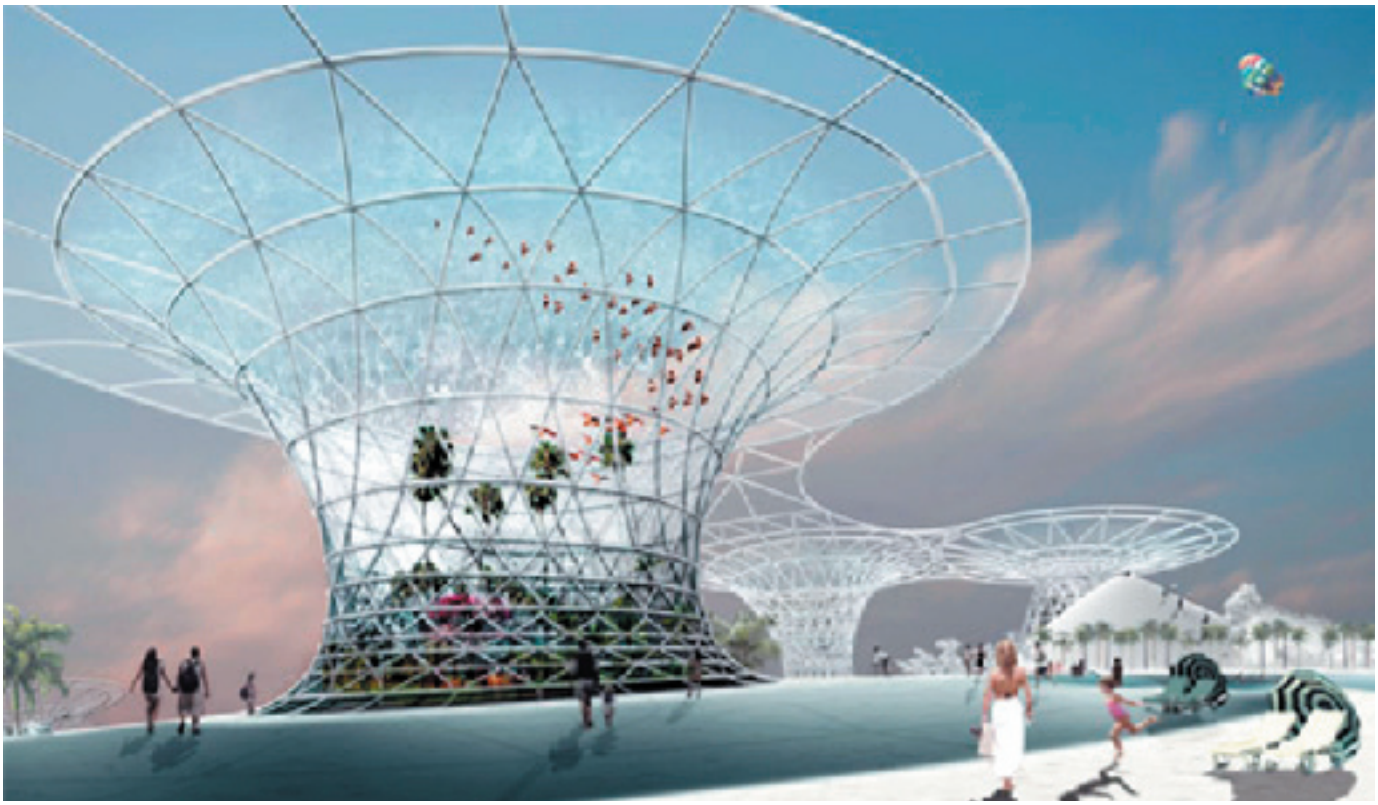


*Fig. 1.19*

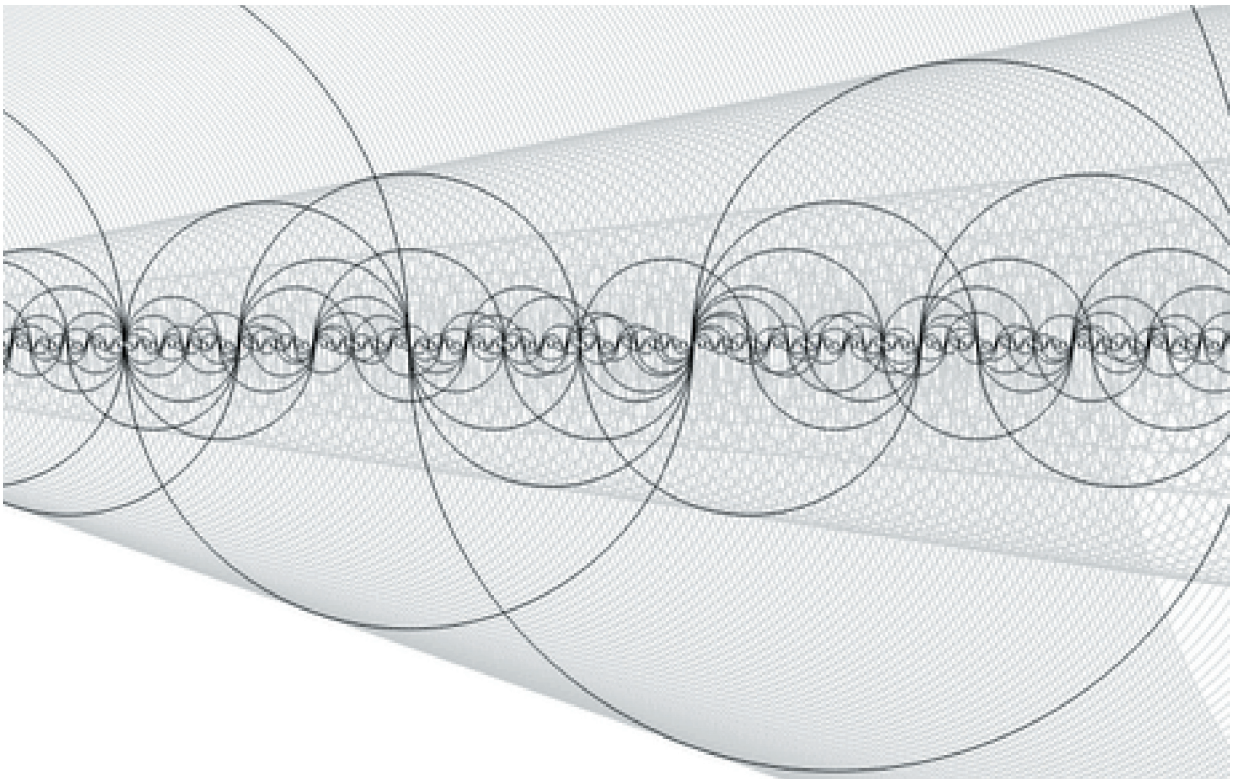


*Fig. 1.20*

Llegado a este punto, despúes de todas y cada una de estas reflexiones, podemos comprender porque es fundamental que el diseñador de la comunicación visual, aprenda a visualizar de manera abstracta, pues ésta capacidad intelectual, le permitirá lograr un equilibrio entre las habilidades técnicas de la producción visual y las teorías, conocimientos y metodologías, necesarias para un adecuado sustento y mejora de su trabajo, para responder a los retos que nos plantea el presente y el futuro. Por todo lo expuesto hasta aquí, considero importante concluir ésta sección y pasar al siguiente capítulo, no sin antes presentar más ejemplos. Fig. 1.21. La visualización de las ondas de sonido. Fig. 1.22. Superficie de Riemman. Fig. 1.23. La casa de las escaleras de Escher. ¿Basada en una superficie de Riemman?. Veamos otras figuras.

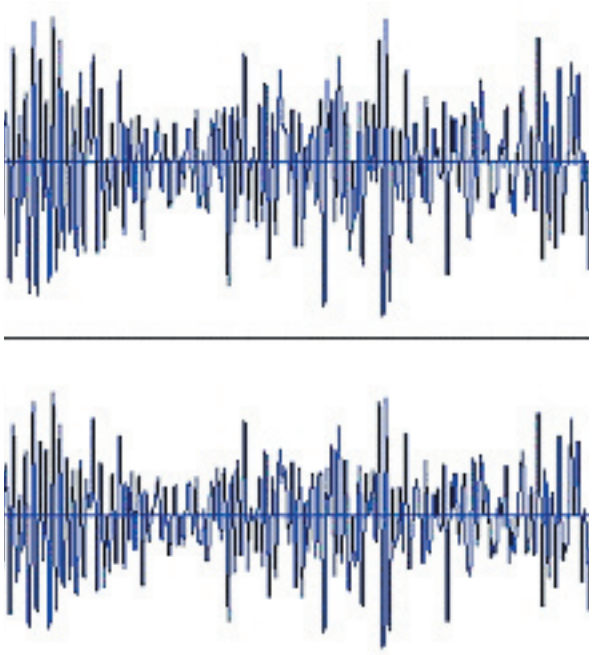
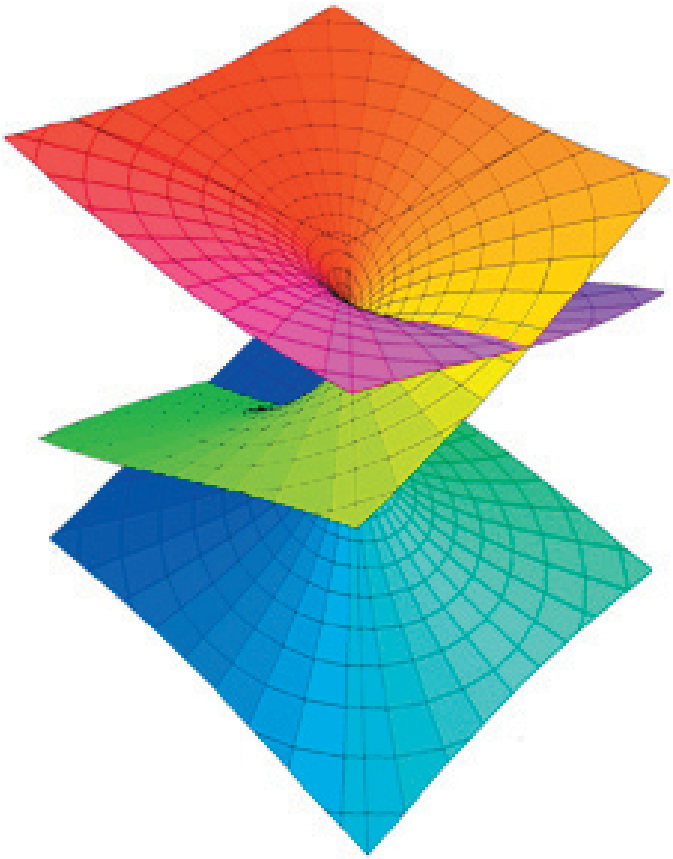


*Fig. 1.21*



$$f(z) = z^{\frac{1}{3}}$$

*Fig. 1.22*

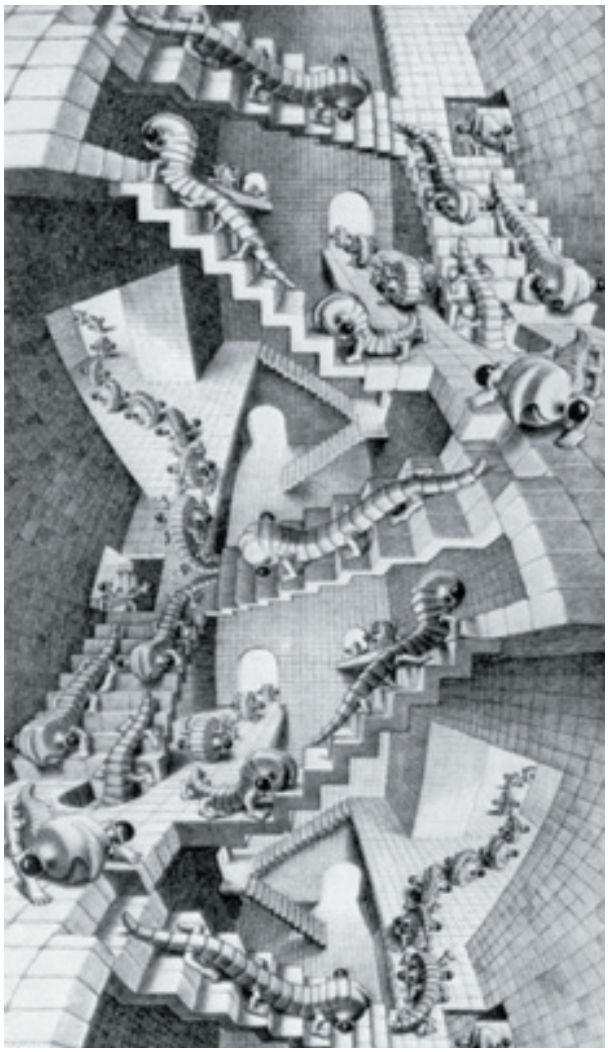


*Fig. 1.23*

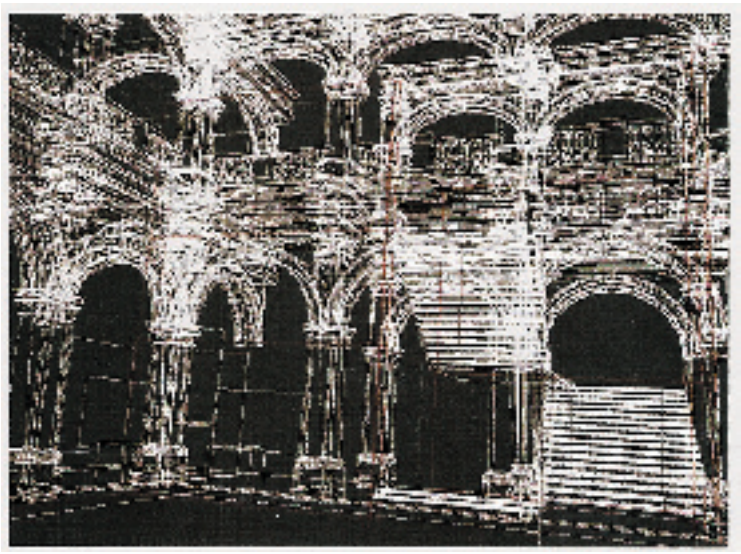


*Fig. 1.24*

**Francisco Estrada, Patio central de la Academia de San Carlos, 2008. Se utilizó Render en Alambre del modelo, VRML.**

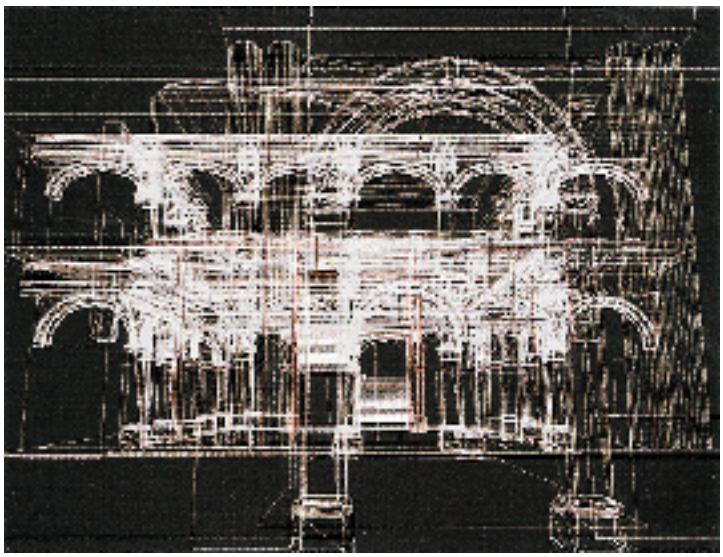
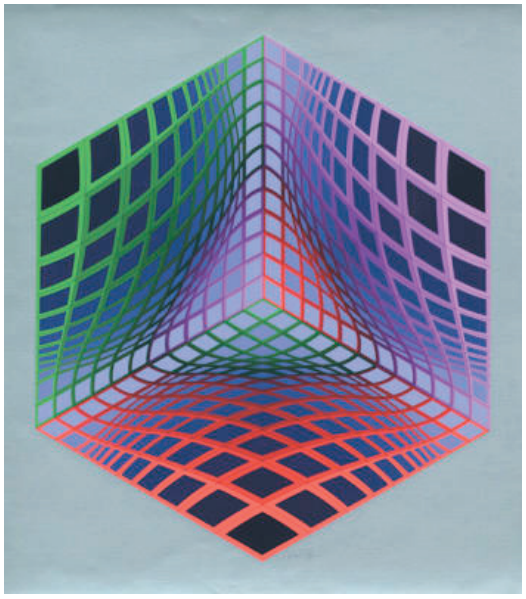
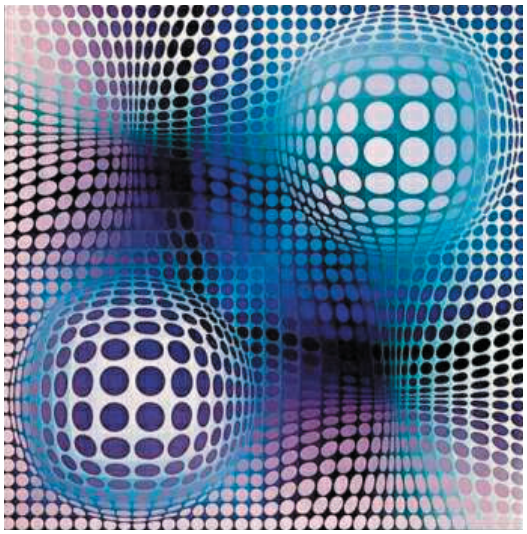


*Fig. 1.25*

**Francisco Estrada, Entrada de la Academia de San Carlos, 2008. Se utilizó Render en Alambre del modelo, VRML. El Maestro Estrada, ha realizado diversos proyectos de animación digital, 3D, realidad virtual y programación de software para diversas galerias de arte interactivo. Su trabajo, ha sido trascendental para la comunidad de la FAD (Facultad de Artes y Diseño) UNAM, en lo que respecta a Multimedia, Arte y Tecnología.**





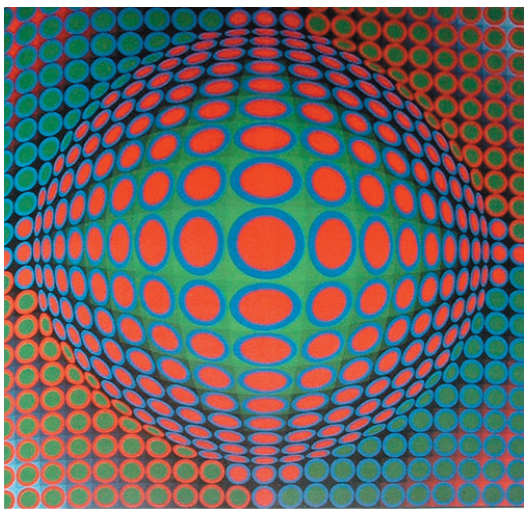


*Fig. 1.26*

**Trabajos de Víctor Vasarely.**

*Fig. 1.27*

La física detrás del arte: Turbulencias en cuadros de Van Gogh.

Un equipo de científicos de México, España y Reyno Unido realizó estudios a algunos cuadros del artista holandés Vincent Van Gogh; En el proyecto participaron dos físicos mexicanos: el Dr. José Luis Aragón Vera del Centro de Física Aplicada y Tecnología Avanzada y el Dr. Gerardo García Naumis, del Instituto de Física de la UNAM. Los resultados se publicaron en diversas revistas científicas, entre ellas la prestigiosa "Nature", que tituló el trabajo como "Escala de Kolmogorov en apasionadas pinturas de Van Gogh".

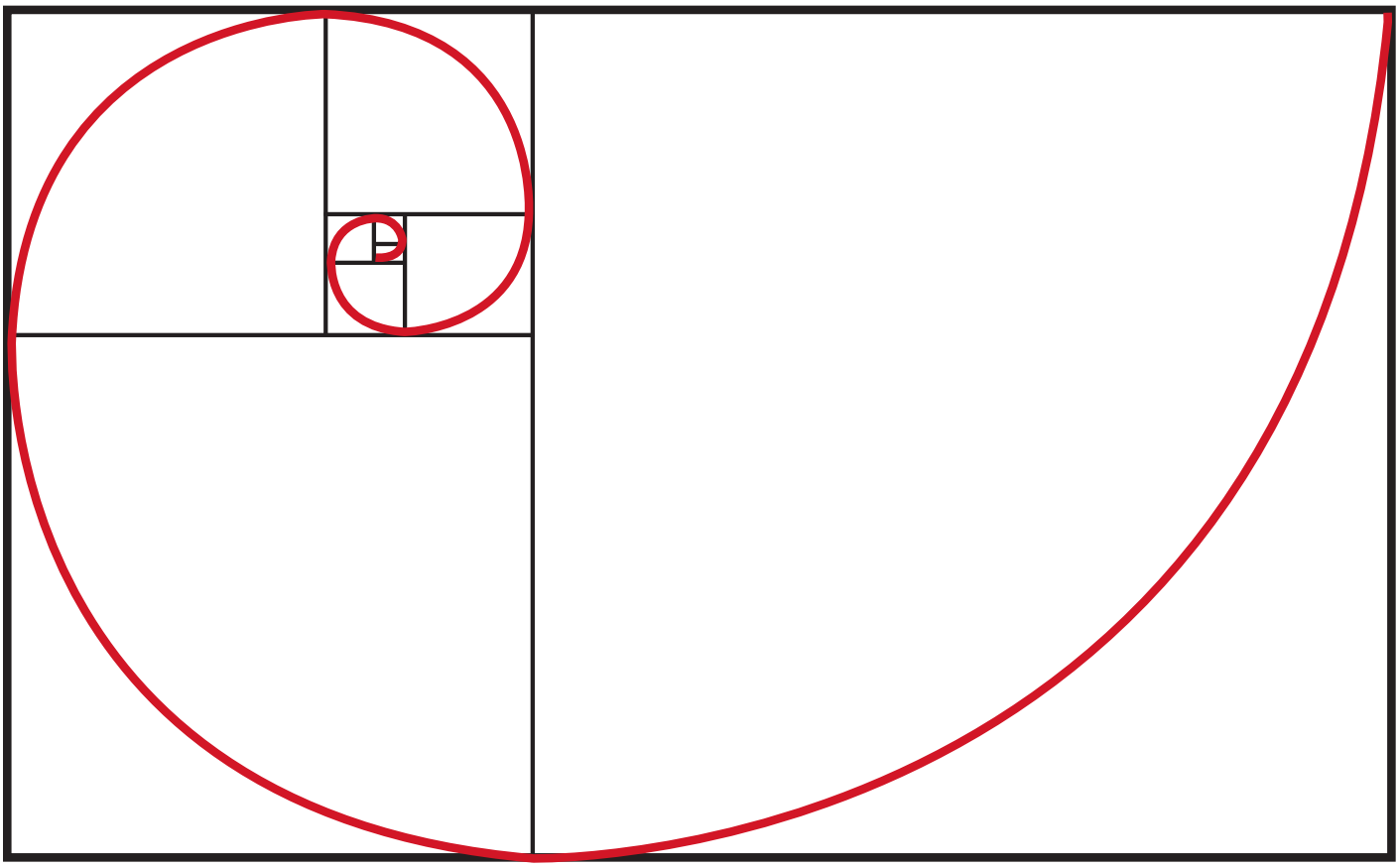


*Fig. 1.28*

La visualización de la espiral áurea: $x_1 = \frac{1+\sqrt{5}}{2} = \varphi \approx 1{,}61803$

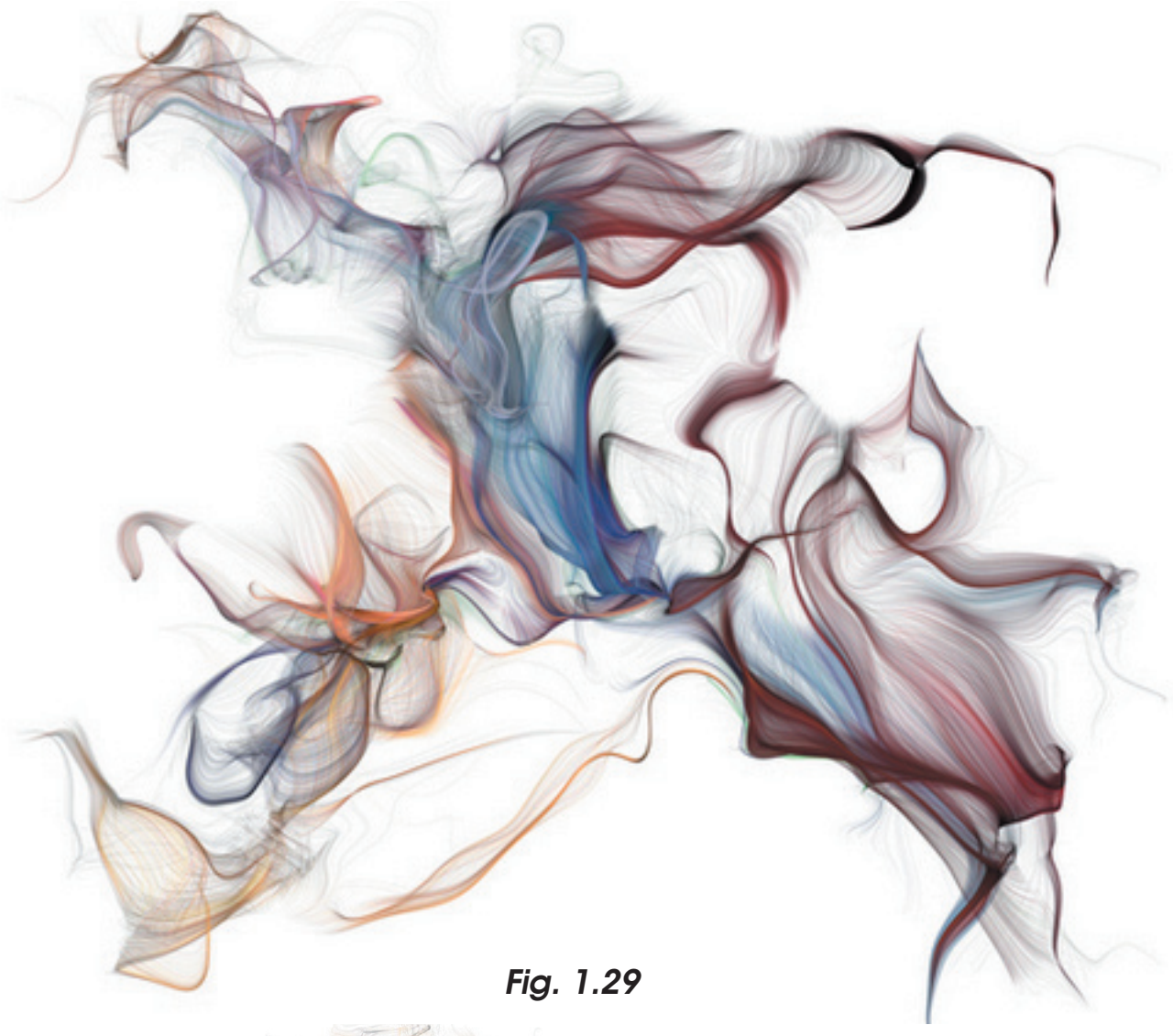


*Fig. 1.29*

*Fig. 1.30*

Fig. 1.29. y Fig. 1.30. Pinturas de Thomas Briggs. ¿Acaso, una emulación de los fluidos de la naturaleza?. Fig. 1.31 Triángulo Sierpinski. Fig. 1.32 Bote de Klein. Fig. 1.33 y 1.34 Floreros basados en formas matemáticas. Fig. 1.35 De las matemáticas al arte. Esta figura nos muestra una obra de Math-art y como esta, hay muchos ejemplos de arte basado en matemáticas.

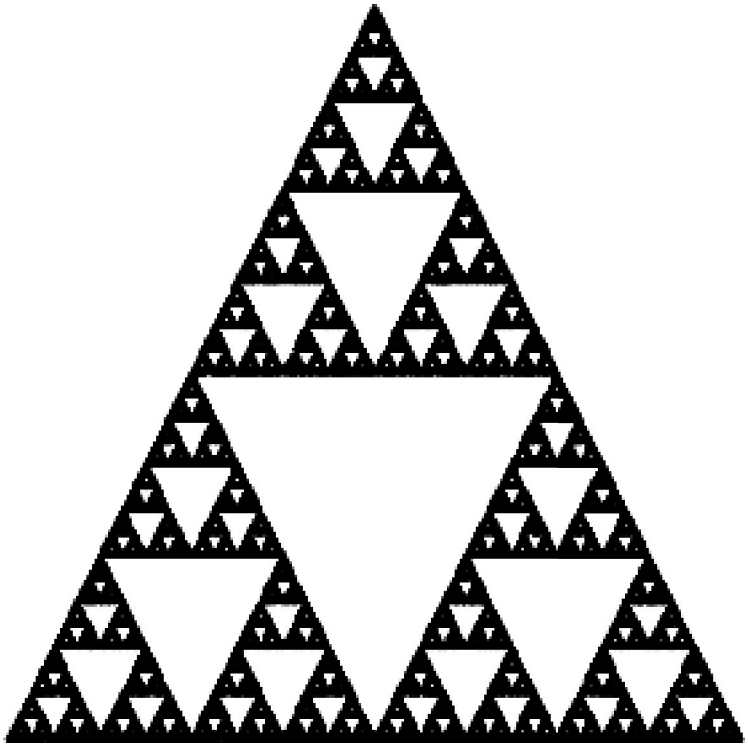


*Fig. 1.31*

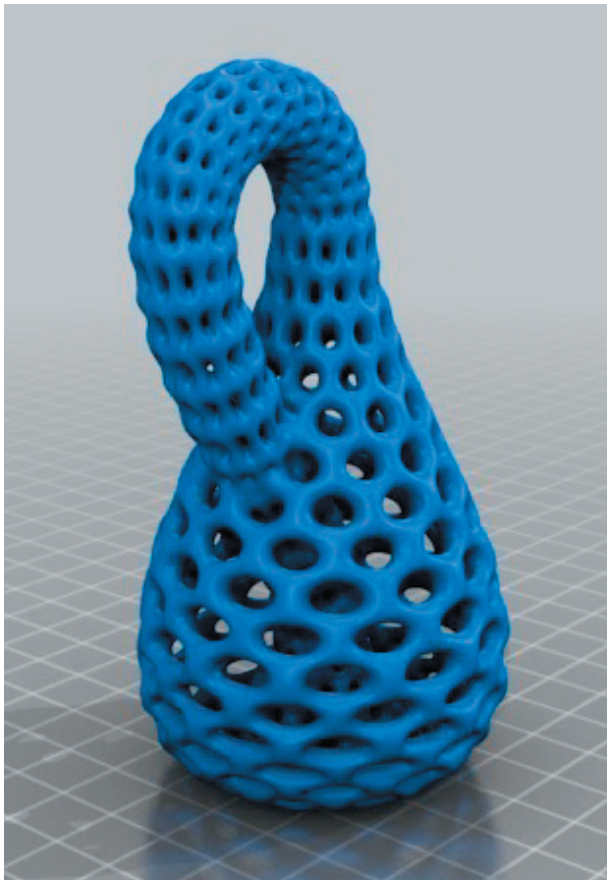


*Fig. 1.32*

*Fig. 1.33*

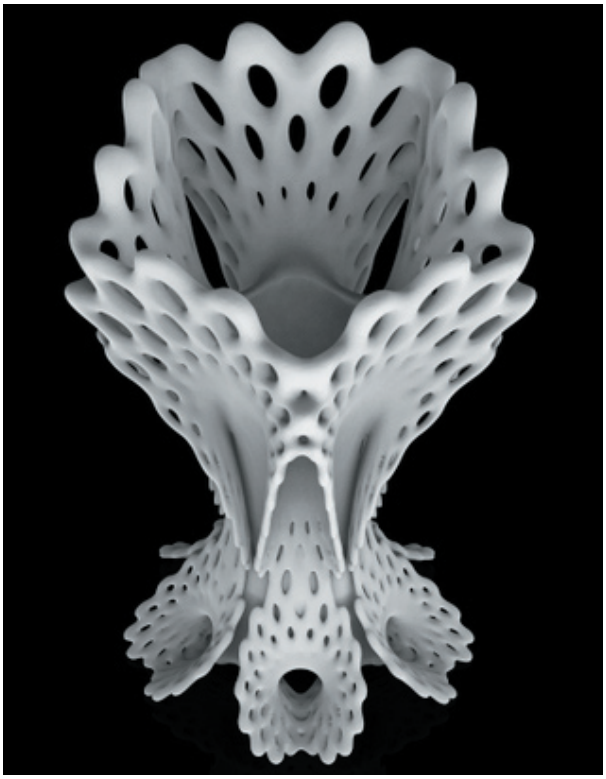


*Fig. 1.34*

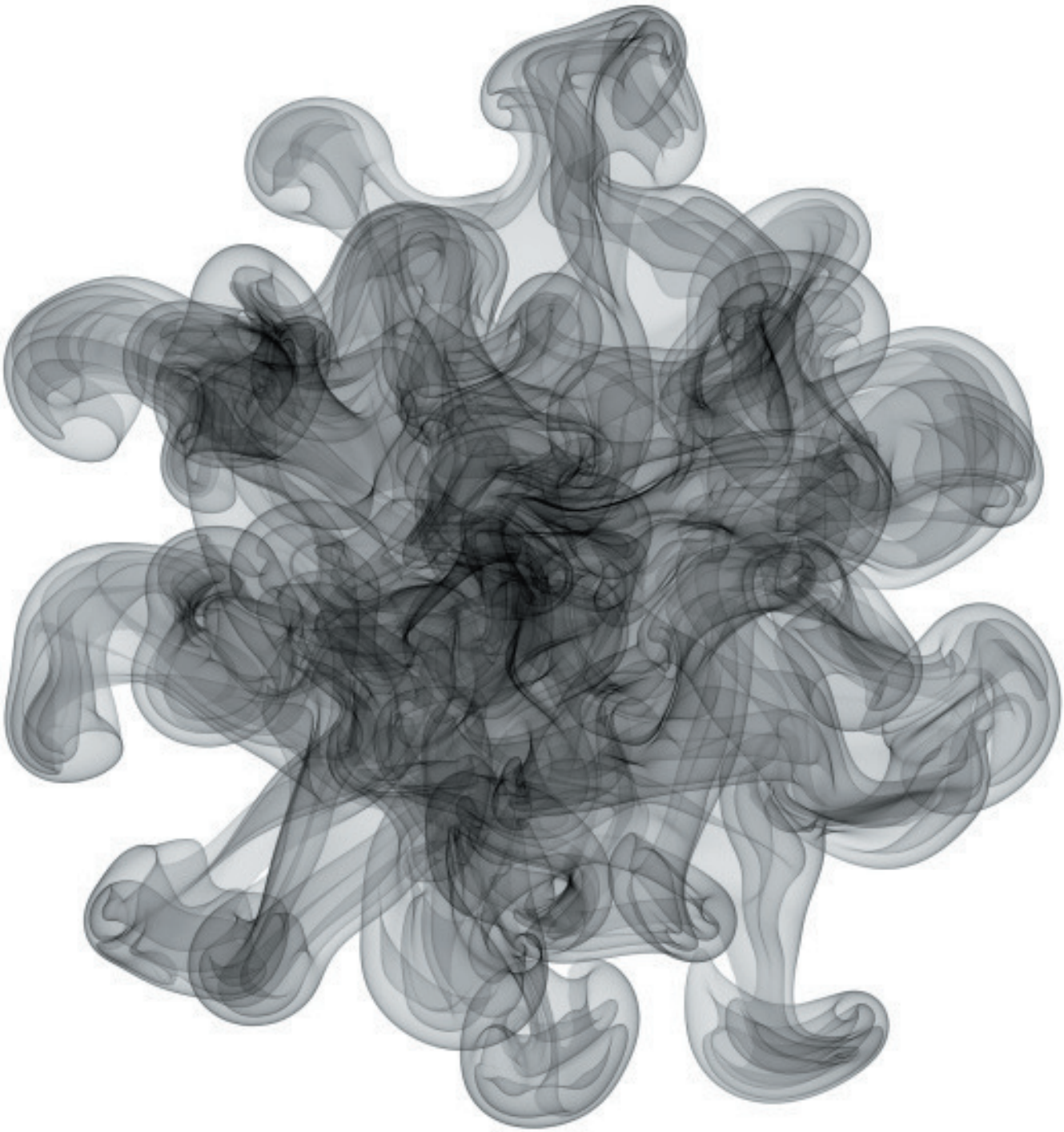


*Fig. 1.35*

# II. LA VISUALIZACIÓN DE LAS FÓRMULAS MATEMÁTICAS

# II. LA VISUALIZACIÓN DE LAS FÓRMULAS MATEMÁTICAS.

Es un hecho que la mayoría de los diseñadores de la comunicación visual, nos involucramos poco o nada en las ciencias duras. Aún así, nos sentimos fuertemente atraídos hacia la tecnología, porque ésta provee maravillosas herramientas para diseñar. En mi caso como diseñador de trabajos multimedia, me encuentro en una posición en medio del diseño, la comunicación visual, las artes y la tecnología de software, ésta última, un producto de la ciencia, la tecnología, las matemáticas y la programación, situación que me invita a reflexionar sobre lo importante que puede ser, pensar científicamente en la profesión del diseño. Fig 2.1. Animación interactiva de una barra que rebota.

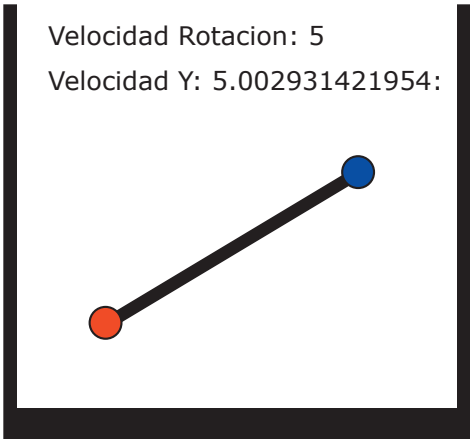



*Fig. 2.1*

En mi experiencia profesional como diseñador multimedia, he observado que la mayoría de los diseñadores, aprendemos algunos principios básicos de programación para realizar web sites o interactivos multimedia, en la mayoría de los casos; estáticos, con pocos recursos y también, con poca programación. Considero que para mejorar esta producción multimedia, puede haber varios caminos, disciplinas tan variadas como posibilidades creativas en un universo de recursos tecnológicos. Éstos caminos a mi modo de ver, pueden ser, el perfeccionamiento de las habilidades en las artes o el estudio de las ciencias duras, dentro de éstos caminos, en el proyecto de investigación en cuestión, se ha elegido la ciencia y dentro de la ciencia, las matemáticas. La razón es simple. La mayoría de los diseñadores no conocemos lo suficiente sobre matemáticas, ni siquiera las más básicas, entonces cuando intentamos producir animaciones interactivas o un videojuego bidimensional, lo hacemos sin estar conscientes de ésta parte; en el caso de los gráficos digitales, los dibujamos usando el software de ilustración, pero muchos diseñadores no entienden que ése gráfico 2D o 3D, tuvo su origen en la matemática y la programación, dos conocimientos que sutilmente se encuentran detrás de la interfaz en la mayoria de los programas que utilizamos para trazar o hacer animación 3D por computadora. Fig. 2.2. Videojuego de Pong. Fig. 2.3. Videojuego 3D Kinect.

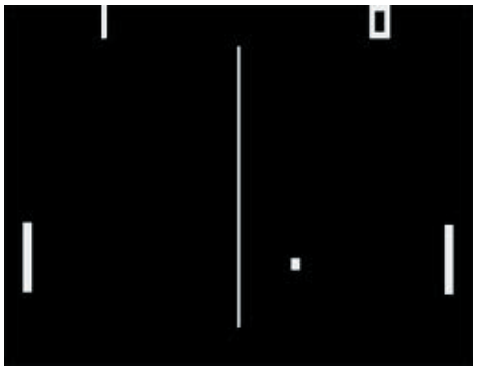


*Fig. 2.2*

*Fig. 2.3*

En el caso de la producción de animaciones interactivas o juegos de video, muchos diseñadores copian los códigos de los tutoriales sin entender que el código nos ayuda a trazar un gráfico, a animarlo y/o programarlo, y en general, hacer funcionar toda la aplicación, por lo tanto, hacen el trabajo sin conciencia de lo que están haciendo y limitan su experimentación. Fig. 2.4. Video Juego bidimensional de Pac-Man. Uno de los primeros juegos de video y se puede programar con código Action Script.

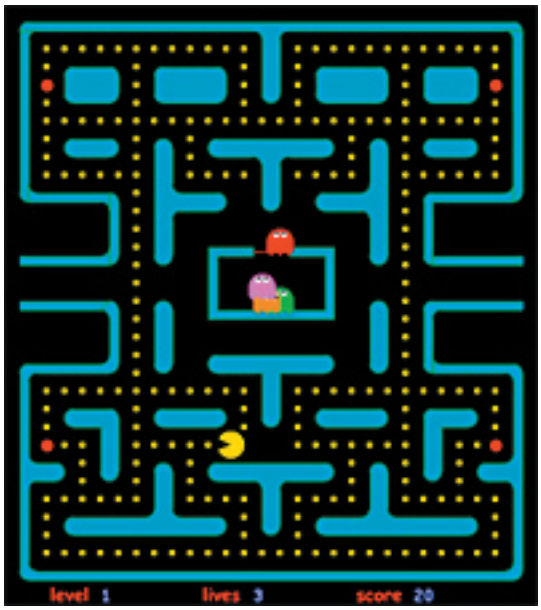



*Fig. 2.4*

Del mismo modo, toda esta cuestión relacionada a la producción de animación interactiva y videojuegos, es la que hace indispensable explicar al diseñador multimedia, que los gráficos tienen un origen matemático basado en programación.

Recordemos que este proyecto se llama la visualización de las fórmulas matemáticas, sin embargo sería imposible desarrollar un escrito que explique todas las fórmulas existentes en esta ciencia, por ello lo más lógico sería hablar primero del punto, la línea y el plano, una base que nos permita avanzar hacia otros niveles y desarrollar, proyectos más complejos como: Las fórmulas matemáticas de las formas bidimensionales, o tridimensionales, notemos que hasta aquí, solo hablamos de objetos sin movimiento. Ahora, si entiendo esto, porque no desarrollar un proyecto titulado, las matemáticas en la animación interactiva didáctica. Éste, un tema con distintos niveles de complejidad, desde una animación simple como el movimiento de un vehículo con aceleración constante, hasta una animación compleja en los que se han de involucrar otras fuerzas físicas como peso, masa, resistencia de los materiales, aspectos que hacen creíble una animación de este tipo. Por consiguiente, posterior a la animación interactiva, tendríamos temas como: Las físicas en los videojuegos, desde el simple juego bidimensional de pong hasta el video juego 3D kinect de estrategia etc., y para llegar a éstos niveles debemos ser modestos y sencillos y hablar de las fórmulas matemáticas básicas, para después generar tres interactivos didácticos con este conocimiento. Espero con toda sinceridad hacer de este texto una lectura útil para el diseñador de la comunicación visual. Fig. 2.5.Videojuego bidimensional de carreras, éste también se puede programar utilizando el código Action Script.

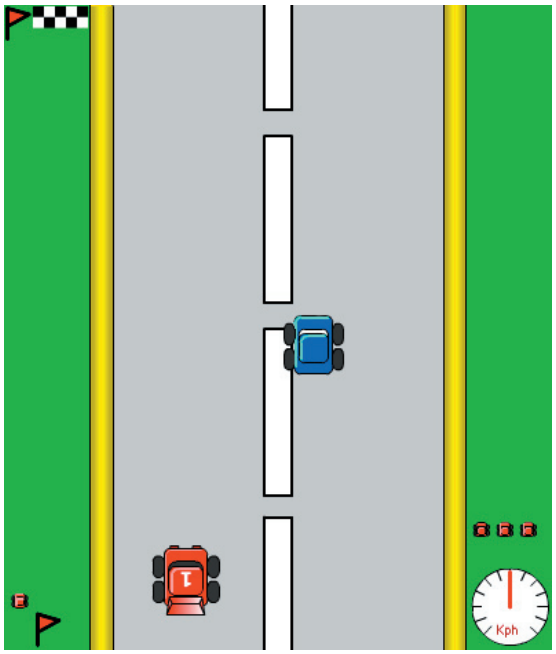



*Fig. 2.5*

**2.1. El plano cartesiano.**

Pongamos nuestra mente en blanco. Ahora pensemos que ese espacio es una hoja infinita de papel. En ella aparecen una tras otra, varias figuras geométricas; una línea, un triángulo, un cuadrado, un pentágono, un hexágono y un círculo. Fig. 2.6 Portada de la Novela de Edwin A. Abbott. Fig. 2.7 Plaskolandia. Película animada. Fig. 2.8 Flatland. La Pelicula.

Estas figuras geométricas cobran vida y comienzan a moverse hacia arriba, hacia abajo, a los lados, sin poder elevarse por encima o sumergirse por debajo. Pueden moverse, porque en ese mundo como en el nuestro, también existe el tiempo y las formas que allí habitan, están atrapadas en su dimensión. Ese mundo lo imaginó Edwin Abbott Abbott a finales del siglo XIX, y lo llamó planilandia. Fig. 2.9  El Doctor Quantum visitando Planilandia. Fig. 2.10  Plaskolandia. Película animada.


*Fig. 2.6*

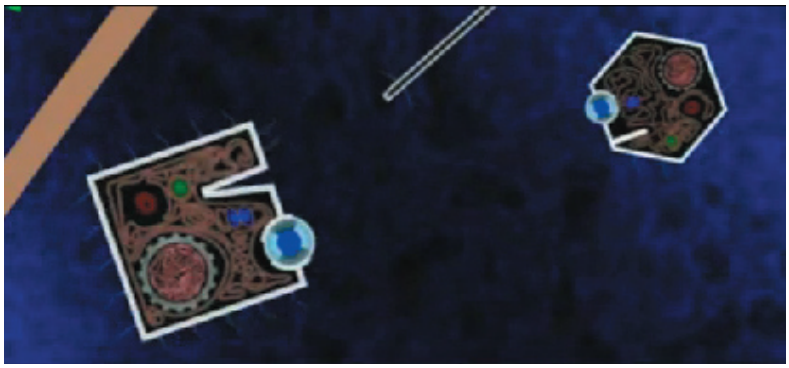


*Fig. 2.7*

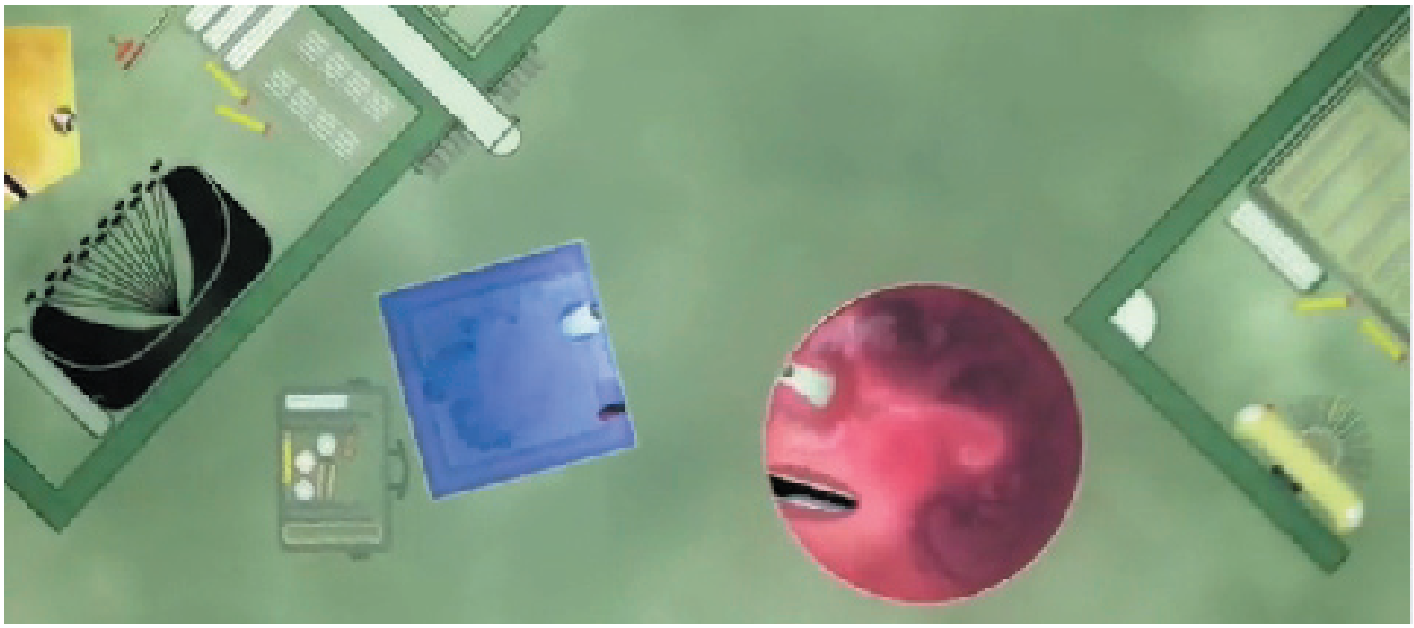


*Fig. 2.8*

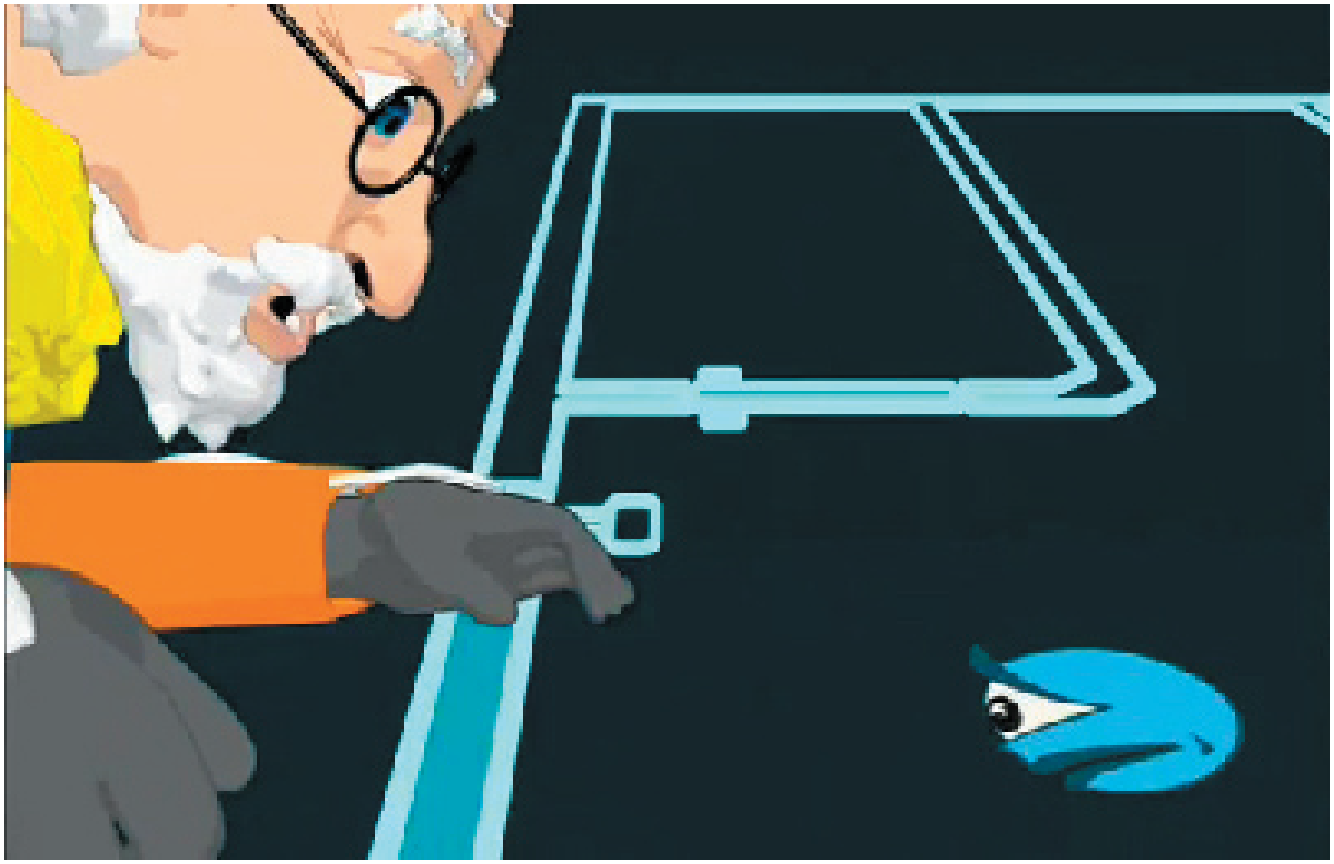


*Fig. 2.9*

*Fig. 2.10*

Planilandia es el plano cartesiano, en él existen también, cuatro puntos cardinales; norte, sur, este y oeste, por supuesto que esas referencias, no están trazadas físicamente, sin embargo, un mapa de planilandia es bidimensional y para trazarlo sólo tendríamos que ubicar donde es el centro de ese espacio.

Para este fin, cualquier lugar que escojamos será ideal, recordemos que es un espacio infinito, así que pongamos un punto origen, y a partir de él, tracemos con nuestra regla dos líneas rectas, una horizontal y otra vertical.

En matemáticas la recta horizontal o X es el eje de las abscisas, y la recta vertical o Y, es el eje de las ordenadas. En nuestra línea vertical con dirección hacia arriba será el NORTE, y con dirección hacia abajo el SUR.

En nuestra línea horizontal con dirección a la derecha será el ÉSTE, y con dirección a la izquierda, el OESTE. Hacia arriba su valor será positivo, hacia abajo, será negativo, hacia la derecha será positivo y hacia la izquierda será negativo. El plano cartesiano será la casa del punto, un ente gráfico primigenio. La primera presencia. Fig. 2.11 El plano cartesiano.

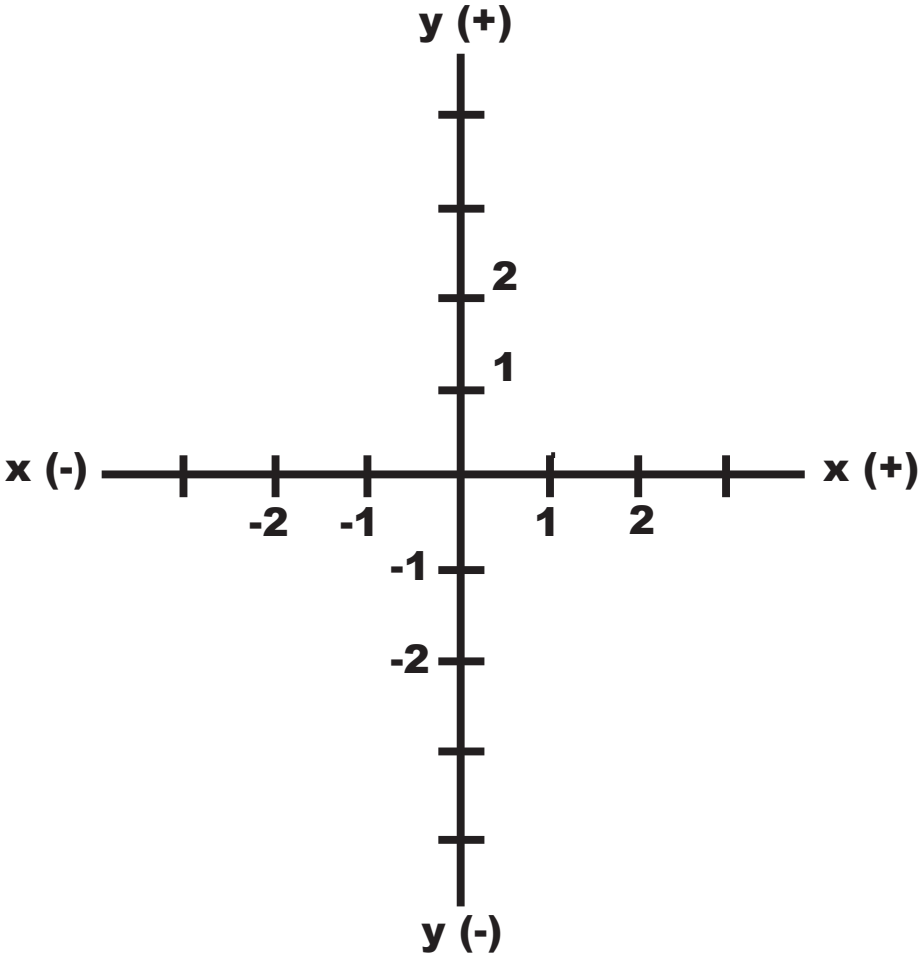



*Fig. 2.11*

**2.2. El punto.**

La mayoría de nosotros cuando pensó en planilandia, la observó con los ojos de la tercera dimensión. Sin embargo para entender como es un universo bidimensional, lo mejor es mirarlo con los ojos de alguno de sus habitantes.

Si soy un pentágono o un cuadrado, tengo la necesidad de moverme, comunicarme y convivir con las demás figuras; allí percibo a todos como líneas; líneas cortas, líneas largas o líneas del mismo tamaño, dependiendo que figura es, como la veo y como se mueve.

Por lo tanto, también hay peligros, engaños e ilusiones, por ejemplo, toparse con un triángulo isósceles de frente creyendo que es una línea, es peligroso, un descuido que podría costarle la vida a cualquiera en planilandia. Por ello ésta forma se prohibió en los edificios.

También en planilandia como en nuestro mundo, los habitantes más interesantes y peligrosos, son las mujeres. Las mujeres en planilandia son líneas cuando las miramos de lado, y puntos, cuando les vemos de frente, prácticamente se vuelven invisibles la mayor parte del tiempo.

Abbott las describe así (**Abbot. A. Edwin.** *Planilandia. Una novela de muchas dimensiones*. Torre de viento. pags. 16-17): *Una mujer es una aguja, ya que es, como si dijéramos, toda punta, por lo menos en las dos extremidades. Añádase a esto el poder de hacerse prácticamente invisible a voluntad, y comprenderéis que una mujer es, en Planilandia, una criatura con la que no se puede jugar.*

*Es posible, sin embargo, que algunos de mis lectores más jóvenes se pregunten cómo puede hacerse invisible una mujer en Planilandia.*

*Esto debería resultar evidente para todos, creo yo, sin ninguna necesidad de explicación.*

*Añadiré, no obstante, unas palabras aclaratorias para los menos reflexivos.*

*Poned una aguja en una mesa. Luego, con la vista al nivel de la mesa, miradla de lado, y veréis toda su longitud; pero miradla por los extremos y no veréis más que un punto, se ha hecho prácticamente invisible. Lo mismo sucede con una de nuestras mujeres.*

*Cuando tiene un lado vuelto hacia nosotros, la vemos como una línea recta; cuando el extremo contiene su ojo o boca (pues entre nosotros esos dos órganos son idénticos) esa es la parte que encuentra nuestra vista, con lo que no vemos nada más que un punto sumamente lustroso; pero cuando se nos ofrece a la vista la espalda, entonces (al ser casi tan mate, en realidad, es como un objeto inanimado) su extremidad posterior le sirve como una especie de tope invisible.*

*¿Qué puede significar tropezar con una mujer, salvo destrucción absoluta e inmediata?*

*Y cuando una mujer resulta invisible, o visible sólo como un punto mate sublustroso, ¡qué difícil es siempre, hasta para el más cauto, evitar la colisión!*

Recordemos que en diseño, el punto es el primer elemento que aparece en un soporte material vacio.

En matemáticas, ocurre exactamente lo mismo, el punto aparece en el plano cartesiano, cuando conocemos de manera muy precisa su ubicación. En el caso de un espacio vacío, la contradicción mas obvia de éste, es que nunca está totalmente vació ya que de él, surgen las demás posibilidades, y un punto por si mismo, indica que ya no existe el vacío.

En artes plásticas y diseño, el punto es la unidad más simple y elemental de los elementos básicos visuales. Un elemento que se hace presente sobre la superficie de un plano, cuando nuestra visión se encuentra con él en el espacio. Su sola existencia como unidad orgánica mínima, nos habla de la presencia de un elemento, de algo que simplemente está ahí.

Dos puntos en el espacio nos dan una referencia de la medida que existe entre ellos y si los unimos se convierten en una línea, así mismo, varios puntos organizados, nos sugieren la idea de una figura.

El mismo Kandinsky escribió casi todo un libro hablando sobre el punto y lo hizo, porque pudo ver la importancia que tiene ésta unidad mínima casi viviente en matemáticas, arte y diseño.

Leamos un poco esa concepción del punto en palabras de Kandinsky. (**Kandinsky.** *Punto y línea sobre el plano. Contribución al análisis de elementos pictóricos.* Paidós Estética*. pags.* 21-22): *El punto geométrico es invisible. De modo que lo debemos definir como un ente abstracto. Si pensamos en él materialmente, el punto se asemeja a un cero. Cero que, sin embargo, oculta diversas propiedades «humanas». Para nuestra percepción este cero —el punto geométrico—está ligado a la mayor concisión. Habla, sin duda, pero con la mayor reserva.*

*En nuestra percepción el punto es el puente esencial, único, entre palabra y silencio.*

*El punto geométrico encuentra su forma material en la escritura: pertenece al lenguaje y significa silencio.*

*En la conversación corriente, el punto es símbolo de interrupción, de no existencia (componente negativo) y al mismo tiempo es un puente de unidad a otra (componente positivo).*

*Tal es en la escritura su significado intrínseco.*

*El punto es, además, en su exterioridad, simplemente el elemento práctico, utilitario, que desde niños hemos conocido.*

*El signo exterior se vuelve costumbre y oscurece el sonido interior del símbolo.*

*El punto se instala sobre la superficie y se afirma indefinidamente. Por consiguiente, tanto en sentido externo como interno, el punto es el elemento primario de la pintura y en especial de la obra «gráfica».*

*El punto es la mínima forma temporal.*

*Desde un punto de vista puramente teórico, si el punto es:*

*a) un complejo (de tamaño y forma) y*

*b) una unidad claramente determinada, su relación con el plano básico ha de constituir, en ciertos casos, un medio de expresión suficiente. Considerado en forma esquemática, el punto puede constituir por sí mismo una obra de arte.*

Con esta descripción impregnada de sentimiento, casi poética, Kandinsky nos hace reflexionar sobre la importancia del punto y nos captura con las palabras, de la misma manera que lo hace Abbott con la imaginación, pero volvamos una vez más a las matemáticas y al plano cartesiano, volvamos una vez más al punto en cuestión: el punto. Fig. 2.12. Ubicaciones del punto en el plano cartesiano: A(3x, 2y), B(–2x, 0), C(–1x, –3y), D(4x,-1y). Fig. 2.13. Los habitantes de planilandia.

Para poder obtener esa entidad llamada punto en el plano cartesiano. P (X,Y) vamos a colocarlo ya sea hacia la derecha o la izquierda tanto como lo indique el número de su ubicación en la línea de las abscisas (línea horizontal), y lo mismo hacia arriba o hacia abajo indicando su posición en la línea de las ordenadas (línea vertical) como en la siguiente imagen que muestra la ubicación: (x1, y1). Fig. 2.14. El punto en x1,y1. Fig. 2.15. Formas puntuales.

Ubicación que también puede ser (x2, y2), u otra, pero que al final, nos dará como resultado visual: el punto, en distintos lugares, una unidad mínima en el diseño, el arte, las matemáticas y la música. Los números romanos en la imagen de ese plano cartesiano representan sus cuatro regiones llamadas “cuadrantes”. Veamos otros ejemplos de ubicaciones del punto en matemáticas para dejar totalmente clara esta idea y pasar, a la aplicación práctica de éste conocimiento. Fig. 2.16. La duración del punto en la música.



*Fig. 2.13*

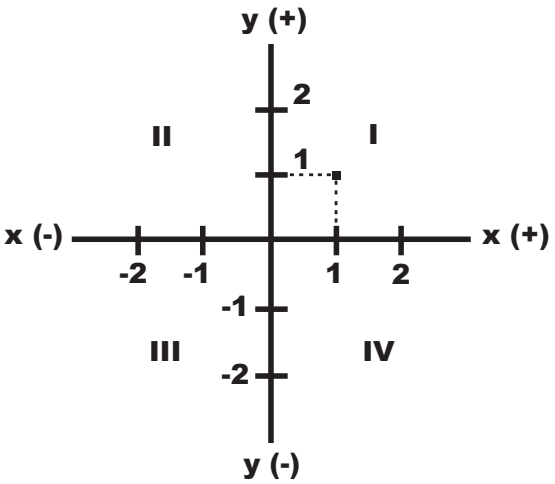



*Fig. 2.14*

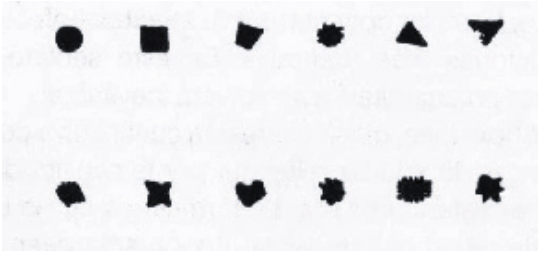


*Fig. 2.15*

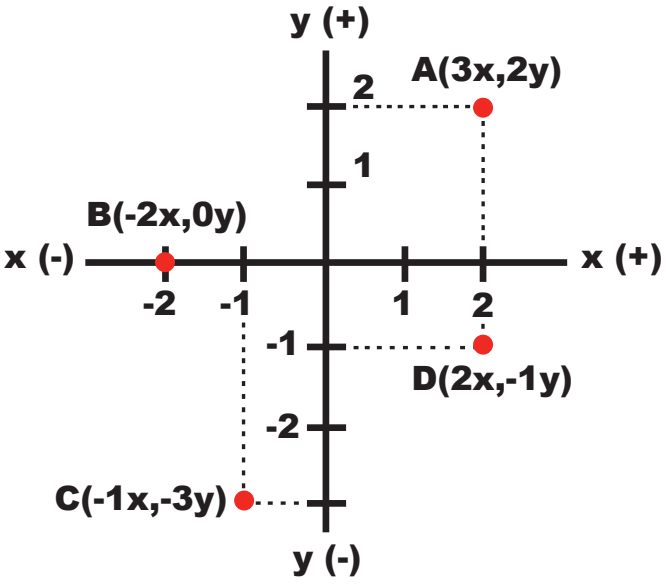



*Fig. 2.12*

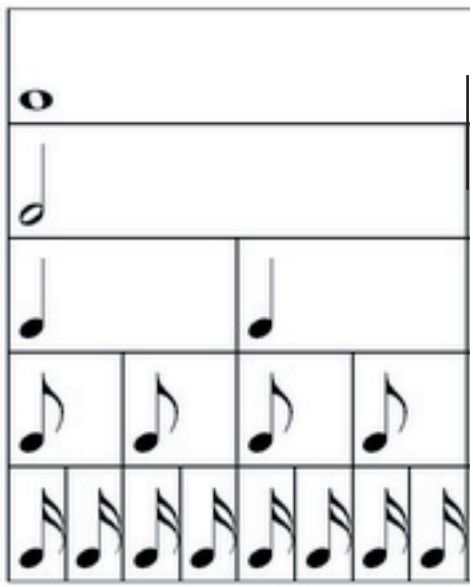


*Fig. 2.16*

### 2.2.1. Interactivo del punto.

La teoría siempre es importante, pero cuando se pone en práctica, adquiere verdaderamente su valor, y esto será; el caso de los interactivos, porque los interactivos, enriquecerán nuestro conocimiento sobre el tema y también, sobre los temas posteriores. Por lo tanto: Nuestro primer interactivo será el interactivo del punto. Éste básicamente, será una aplicación web que nos ofrecerá la posibilidad de mover un punto en un plano cartesiano virtual a través de un navegador web como Google Chrome, Internet Explorer o Mozilla Firefox, dicha aplicación nos mostrará la posición del punto en las coordenadas X,Y. Fig. 2.17. Interactivo del punto.

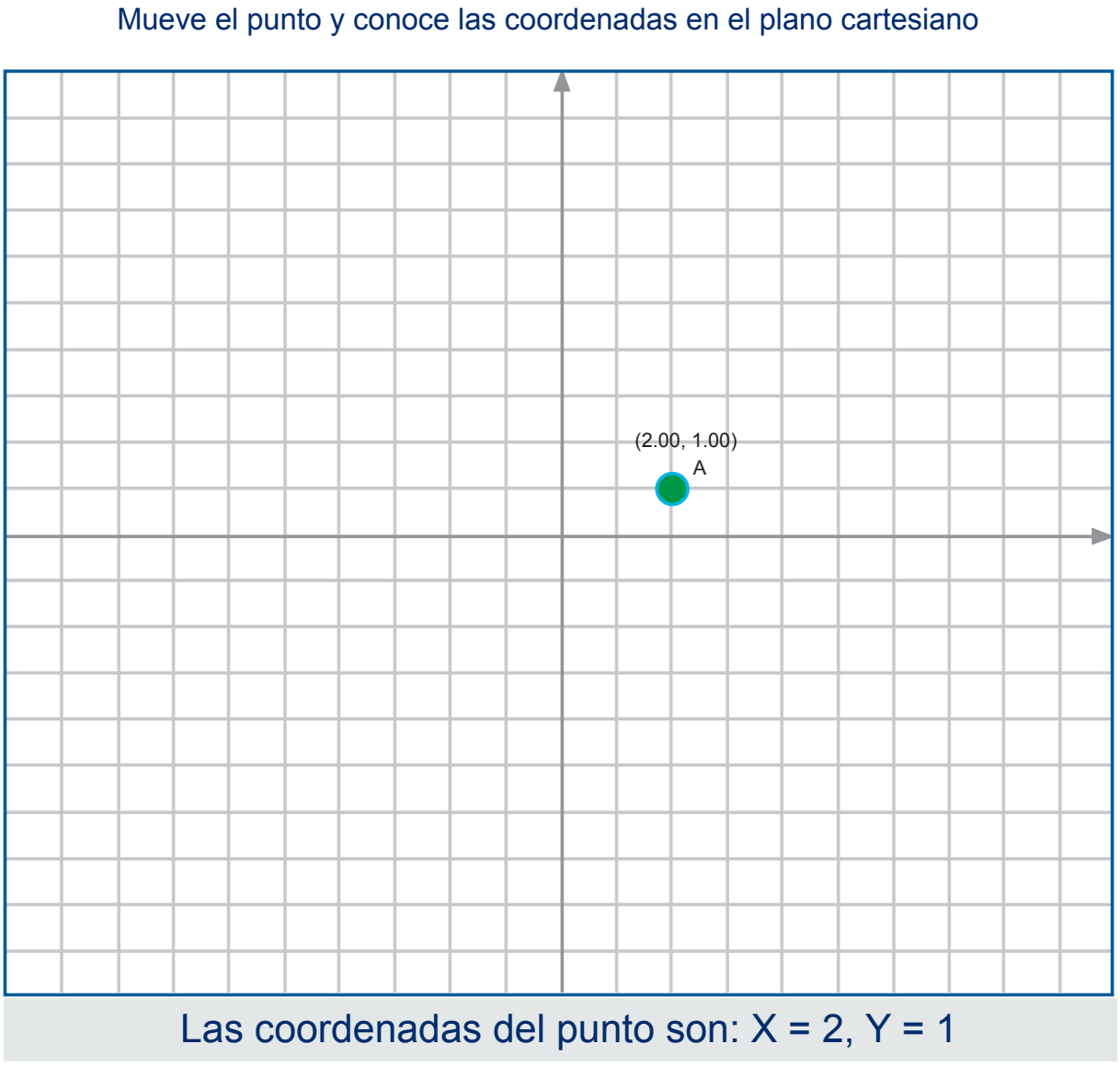



***Fig. 2.17***

La pregunta ahora es ¿porque utilizar éstos lenguajes y no otros? Por tres simples razones:

1) Para cuestiones educativas el código HTML5 es más fácil de comprender para un usuario que navega en internet, pues se puede familiarizar más con el código HTML (el lenguaje de la web), que con otros códigos como C++ o Java.

2) La tecnología de código libre JSXGraph al ser una biblioteca JavaScript, nos proporciona los elementos que necesitamos para trazar y programar nuestros gráficos, y en sí, desarrollar toda la aplicación.

3) La tercera razón, la divulgación de los interactivos. Para poder visualizarlos, sólo bastará abrirlos en cualquier explorador web y esto facilitará su uso, distribución y publicación. Ahora bien, hablemos un poco a cerca de cada uno de estos códigos.

**HTML5:** (HyperText Markup Language, versión 5) según varias fuentes, es la quinta versión del lenguaje básico de la World Wide Web, HTML. John Freddy Vega fundador de CristalLab y Christian Van Der Henst, iniciador del proyecto Maestros de la Web, nos han dicho varias cosas de HTML5, básicamente: *"HTML5, es la actualización de HTML, el lenguaje en el que es creada la web.*

*También es un termino de marketing para agrupar las nuevas tecnologías de desarrollo de lenguajes para web: HTML5, CSS3 y nuevas capacidades de Javascript". "Hoy en día, todas las empresas gigantes de la web: Microsoft, Google, Apple, Adobe, Facebook, Yahoo, Mozilla y miles de proyectos tecnológicos independientes, respiran HTML5". "Los nuevos teléfonos móviles, tabletas, eBooks, netbooks, computadores y otra gama de dispositivos actualmente utilizan HTML5".*

**JavaScript:** JavaScript es un lenguaje de programación multiplataforma orientado a eventos con manejo de objetos, cuyo código se incluye directamente en el mismo documento, éste código pertenece al grupo de los lenguajes interpretados. En estos lenguajes, el ordenador lee cada instrucción dada, la traduce y la ejecuta en el programa para ser visualizado a través de navegadores web como: Internet Explorer, Netscape o Google Chrome.

Para utilizar JavaScript, es necesario conocer sus reglas y vocabulario. Un lenguaje como éste, utiliza una serie de instrucciones conocidas como scripts o guiones, éstos scripts, sólo funcionan en un entorno web, ya que la única razón de ser de JavaScript, son las páginas de internet.

Ahora bien, si ya hemos estudiado HTML, ¿por que tenemos que estudiar JavaScript?, la respuesta es: Porque nos brinda otras posibilidades en la mejora de la interfaz gráfica, efectos visuales, animación interactiva, control y soporte sobre el explorador web, y otras cualidades en aplicaciones externas como: relojes de pantalla, juegos, calculadoras, calendarios, agendas, etc.

**JSXGraph:** Es otro valioso recurso para graficar todo tipo de datos matemáticos. Básicamente es una biblioteca JavaScript que permite crear una amplia variedad de gráficos y aplicaciones a partir de código libre cuando logramos comprender su lenguaje. Entre los gráficos que podemos generar tenemos:

- Gráficos usados en geometría como: puntos, líneas, círculos, intersecciones, etc.
- Graficar curvas, curvas paramétricas.
- Gráficos interactivos por medio de sliders.
- Tangentes, animaciones, polinomios, etc.

Es una aplicación que podemos descargar gratuitamente de internet, un soporte de trabajo en código libre que utilizan algunos programadores para desarrollar aplicaciones, geometría interactiva, trazado de funciones, cartografía y visualización de datos en navegadores web. Ésta aplicación, se desarrolla completamente en JavaScript y tiene diferentes instrucciones, utiliza gráficos vectoriales de tipo SVG, VML, o canvas de gran calidad, y gráficos bidimensionales, tanto estáticos como animados en XML.

JSXGraph se integra fácilmente en JavaScript y tiene un tamaño reducido: pesa menos de 100 kb cuando es incrustado en una página web. Otra de sus ventajas es que no necesita plug-ins adicionales para su funcionamiento como es el caso de Flash o Java y es compatible con otros dispositivos táctiles de la plataforma Apple y dispositivos como Android.

Ahora sin más preámbulos, revisemos el código fuente de nuestro interactivo.

La aplicación llamada interactivo del punto, consta de tres archivos: un archivo principal llamado punto.html y otros dos documentos: jsxgraph.css y jsxgraphcore.js, ahora hablemos brevemente de cada uno, comencemos por nuestro documento principal: punto.html

Punto.html, es nuestro interactivo y documento principal, es el primer archivo para visualizar la aplicación. Por otro lado, el archivo jsxgraph.css es otro documento que contiene estilos tipográficos, colores y tamaños de letra básicamente. Y jsxgraphcore.js, es la biblioteca de código que nos servirá para construir los elementos visuales de nuestro interactivo.

## El Código fuente de nuestro interactivo: punto.html

```
<!DOCTYPE html>
<html lang=”en”>
<head>
<meta charset=”UTF-8”/>
<title>El punto</title>
<script src=”jsxgraphcore.js”></script>
<link rel=”stylesheet” href=”jsxgraph.css” />

<style>
 .instrucciones {
            font-family:Arial, Helvetica, sans-serif;
            font-size:14px;
            color:#003366;
            font-weight:bold;
            text-align:center;
            width:600px;
 }
 #textoCoordenadas {
            font-family:Arial, Helvetica, sans-serif;
            font-size:20px;
            color:#003366;
            font-weight:bold;
            padding:6px;
            background:#EAEAEA;
            width:600px;
            text-align:center;
 }
</style>
</head>
<body>
<div class=”instrucciones”>Mueve el punto y conoce las coordenadas en el plano cartesiano </div><br>
<div id=”plano” class=”jxgbox” style=”width:600px; height:500px;”></div>
<script>
 var planoCartesiano = JXG.JSXGraph.initBoard(‘plano’,
 {
            boundingbox: [-10, 10, 10, -10],
            axis:true,
            grid:true,
            showCopyright:false,
            showNavigation:false
 });
 var punto = planoCartesiano.create(‘point’, [0,0],
 {

name:’A’,
            snapToGrid: true,
     size: 7,
     fillColor: ‘#E6413A’,
     highlightFillColor : ‘#4BB549’,
});
punto.on(‘mousedrag’, function() {
            planoCartesiano.infobox.setProperty({strokeColor: ‘#000000’});
            console.log(punto);
            var lasCoordenadas=”X = “+punto.X()+” , Y = “+punto.Y();
            document.getElementById(‘textoCoordenadas’).innerHTML = “Las coordenadas del punto son: “+lasCoordenadas;
});
</script>
<div id=”textoCoordenadas”> </div>
</body>
</html>
```

Lo siguiente que haremos ahora, será fragmentar el código en partes o bloques para tratar de comprenderlo:

Bloque 1.

```
<!DOCTYPE html>
<html lang=”en”>
<head>
<meta charset=”UTF-8”/>
<title>El punto</title>
```

En este primer bloque, el código nos indica que tenemos un documento tipo HTML. El atributo <lang=”en”> nos indica que el idioma utilizado será en ingles. La etiqueta <head> define la cabeza de nuestro documento. <meta charset=”UTF-8”/> Ésta etiqueta le está indicando al navegador, la manera en que se visualizarán los caracteres de la página en cuestión. En este caso, los caracteres o tipografías están en formato UTF-8, un formato de codificación Unicode. En los formatos Unicode se utilizan alfabetos latinos. Por último, aparece la etiqueta <title>El punto</title>. Nos indica el nombre de nuestro documento.

Bloque 2.

```
<script src=”jsxgraphcore.js”></script>
```

Aquí lo que nos está indicando, es que vamos a insertar a nuestro documento HTML, un script externo, éste script es la biblioteca JavaScript: JSXGrap (Graficador de JavaScript), el cual nos va a proporcionar todos los componentes necesarios en la construcción visual de nuestro interactivo, éstos componentes serán: el plano cartesiano, la cuadricula del plano, y sus divisiones, el punto, y en general; todo el código que necesitamos para el funcionamiento de la aplicación. En pocas palabras el archivo jsxgraphcore.js es el documento que contiene todas y cada una de las funciones necesarias para desarrollar la aplicación. Las posibilidades que ésta nos brinda en el desarrollo de aplicaciones interactivas, son muchísimas.

Bloque 3.

```
<link rel=”stylesheet” href=”jsxgraph.css” />
```

En esta etiqueta se está insertando otro documento en archivo CSS (hoja de estilo en cascada) éste archivo también fue generado en JSXGraph y contiene los estilos predefinidos para el interactivo en: colores, tipos de letra, tamaños, etc.

Bloque 4.

```
<style>
.instrucciones {
        font-family:Arial, Helvetica, sans-serif;
        font-size:14px;
        color:#003366;
        font-weight:bold;
        text-align:center;
        width:600px;
}
```

Este bloque describe los estilos propios del HTML. El estilo ‘instrucciones’ le da formato: (tamaño, color, tipo de letra) al texto que va a indicar las instrucciones al usuario.

Bloque 5.

```
#textoCoordenadas {
        font-family:Arial, Helvetica, sans-serif;
        font-size:20px;
        color:#003366;
        font-weight:bold;
        padding:6px;
        background:#EAEAEA;
        width:600px;
        text-align:center;
 }
</style>
</head>
<body>
```

En este bloque se muestra el estilo ‘textoCoordenadas’ que le da formato (tamaño, color, tipo de letra) al cuadro de texto que muestra las coordenadas del punto. Se cierran las etiquetas estilo y cabeza, y se abre la etiqueta; cuerpo del documento, para incluir el siguiente bloque de código.

Bloque 6.

```
<div class=”instrucciones”>Mueve el punto y conoce las coordenadas en el plano cartesiano </div><br>
```

En este bloque, se incluye el elemento HTML de tipo  con los estilos de tipografía, tamaño, color, etc, dadas en el bloque 4 de la explicación. La etiqueta **<div class>** se emplea para definir un bloque de contenido o sección dentro de una página, en este caso el texto afectado con estas instrucciones es: *“Mueve el punto y conoce las coordenadas en plano cartesiano”.*

Bloque 7.

```
<div id=”plano” class=”jxgbox” style=”width:600px; height:500px;”></div>
```

En este bloque se incluye otro elemento HTML de tipo div, éste elemento será el que se transformará en el plano cartesiano. Las propiedades de éste elemento son:

id=”plano”: Es el identificador del elemento llamado “plano”. Si se edita este valor, debe editarse la línea 57 del código pero es recomendable no modificar este valor.

class=”jxgbox”: Indica que se deben utilizar los estilos (colores, tamaños, etc.) definidos por la biblioteca JavaScript: JSXGraph. Este valor no se debe modificar.

style=”width:600px; height:500px;”: Especifica el tamaño en pixeles de nuestro plano cartesiano. Este valor se puede modificar si se desea cambiar el tamaño del plano.

Bloque 8.

```
<script>
var planoCartesiano = JXG.JSXGraph.initBoard(‘plano’,
{
        boundingbox: [-10, 10, 10, -10],
        axis:true,
        grid:true,
        showCopyright:false,
        showNavigation:false
});
```

En este bloque comienza la programación del código principal de la aplicación. Aquí se define el comportamiento del interactivo. Lo primero que hace es construir un boundingbox, un plano o cuadro de contenido que se denominará: ‘planoCartesiano’. En este se especifican las dimensiones.
El eje X inicia en -10 y termina en 10, y el eje Y inicia en 10 y termina en -10.

Este valor se puede modificar según el tamaño con el que se desee construir el plano cartesiano. Finalmente axis:true : nos indica que se deben mostrar los ejes X,Y, y grid:true, que mostrará la retícula o cuadricula del plano.

showCopyright:false : Oculta el texto de copyright de los creadores de JSXGraph

showNavigation:false : Oculta las barras de navegación del plano que permiten hacer zoom o desplazarse por los cuadrantes.

Bloque 9.

```
var punto = planoCartesiano.create('point', [0,0],
{
        name:'A',
        snapToGrid: true,
        size: 7,
  fillColor: '#E6413A',
  highlightFillColor : '#4BB549',
});
```

En este bloque se indica que debe crearse un punto sobre nuestro plano cartesiano. Cuyos valores iniciales en texto dinámico serán (0,0).

name:'A' : Es el nombre que nos servirá para identificar a ese punto. Este valor se puede modificar.

snapToGrid: true : Indica que el punto debe ajustarse a las intersecciones de las líneas, así, el punto siempre mostrará coordenadas enteras y no decimales. No es recomendable modificar este valor.

size: 7 : Indica el tamaño del punto. Este valor se puede modificar según se desee para hacer el punto más grande o más pequeño.

fillColor: '#E6413A' : Es el código hexadecimal del color que tendrá el punto, en este caso es un tono rojo. Este valor se puede modificar.

highlightFillColor : '#4BB549': Es el código hexadecimal del color que tendrá el punto cuando el cursor del mouse este posicionado sobre él, en este caso es un tono verde. Este valor también se puede modificar.

Bloque 10.

```
punto.on('mousedrag', function() {
```

Esta parte del código, nos indica que cuando se arrastre el punto por el plano cartesiano se deberán obtener las coordenadas del mismo para ser mostradas como texto dinámico en el elemento  que aparece prácticamente en las últimas líneas del código.

Bloque 11.

```
planoCartesiano.infobox.setProperty({strokeColor: '#000000'});
```

Esta parte del código, nos indica que el color de las letras que mostrarán las coordenadas del punto en el plano cartesiano serán: color negro. Éste valor se puede modificar si se desea.

Bloque 12.

```
        var lasCoordenadas="X = "+punto.X()+" , Y = "+punto.Y();
```

En este bloque se registra una variable denominada 'lasCoordenadas' que almacenará las coordenadas del punto en el plano cartesiano. Dichas coordenadas son obtenidas a través de las funciones X() y Y(), con los valores que nos proporciona la construcción del plano cartesiano con los ejes y las guías.

punto.X(): Devuelve el valor de X del punto.
punto.Y(): Devuelve el valor de Y del punto.

Bloque 13.

```
        document.getElementById('textoCoordenadas').innerHTML = "Las coordenadas del punto son: "+lasCoordenadas;
});
</script>
```

En este bloque, se esta indicando que en el elemento HTML de tipo  se escriban de manera automática y se muestren en pantalla; las coordenadas del punto que fueron extraídas del bloque anterior.

Bloque 14.

```
<div id=”textoCoordenadas”> </div>
</body>
</html>
```

Aquí finalmente se muestra el elemento HTML de tipo  que servirá para mostrar en pantalla las coordenadas del punto. Las propiedades de éste elemento son: id=”textoCoordenadas”: Que es, el identificador del elemento.

Y ésta es básicamente la construcción del interactivo punto.html con el uso de éstas maravillosas herramientas tecnológicas, prosigamos ahora con el siguiente tema: La línea.  Fig. 2.18. Interactivo del punto.


***Fig. 2.18***

## 2.3. La línea.

En planilandia, las mujeres son líneas. Líneas que de frente son puntos y de lado, líneas.

Si en planilandia observamos una línea rotar sobre si misma, veremos al principio un punto, punto que se vera como una línea que empieza a crecer hasta apreciarla completamente, y decrecer hasta convertirse nuevamente en un punto.

En matemáticas, esa es precisamente la manera como nace una línea de manera gráfica cuando conocemos un punto y su pendiente; ello lo veremos más adelante cuando estudiemos la ecuación de la recta, por ahora veamos lo que nos dice Kandinsky sobre la línea, (**Kandinsky.** *Punto y línea sobre el plano. Contribución al análisis de elementos pictóricos. Paidós Estética. pags.* 49-51). *La línea geométrica es un ente invisible. Es la traza que deja el punto al moverse y por lo tanto su producto. Surge del movimiento al destruirse el reposo total del punto. Hemos dado un salto de lo estático a lo dinámico.*

*La línea es la absoluta antítesis del elemento pictórico primario: el punto. Es un elemento derivado o secundario. Las fuerzas que provienen del exterior y que transforman el Origen punto en línea varían: la diversidad de las líneas depende del número de estas fuerzas y de sus combinaciones.*

*Sin embargo, todas las fuerzas productoras de líneas pueden reducirse en definitiva a dos:*

*a) fuerza única y*

*b) dos fuerzas:*

*a') con efecto único o continuado de ambas fuerzas alternantes y*

*b') con efecto simultáneo de ambas fuerzas.*

*Cuando una fuerza procedente del exterior desplaza el punto en cualquier dirección, se genera el primer tipo de línea; la dirección permanece invariable y la línea tiende a prolongarse indefinidamente.*

*Se trata de la recta, que en su tensión constituye la forma más simple de la infinita posibilidad de movimiento. He decidido sustituir la palabra «movimiento», de uso corriente, por «tensión».*

*El concepto corriente es demasiado vago y lleva a conclusiones incorrectas, las que a su vez provocan otros malentendidos terminológicos. La «tensión» es la fuerza presente en el interior del elemento, que aporta tan sólo una parte del «movimiento» activo; la otra parte está constituida por la «dirección», que a su vez está determinada también por el «movimiento».*

*Los elementos en la pintura son las huellas materiales del movimiento, que se hace presente bajo el aspecto de:*

*a) tensión,*

*b) dirección.*

*Esta clasificación crea, además, una base para distinguir elementos de tipo diverso, como por ejemplo el punto y la línea.*

*El punto está constituido exclusivamente por tensión, ya que carece de dirección alguna. La línea combina, al contrario, tensión y dirección. Si con respecto a la recta tomáramos solamente en cuenta su tensión, no podríamos diferenciar una horizontal de una vertical.*

*Del mismo modo, al analizar el color veremos que algunos de ellos sólo se diferencian entre sí por la dirección de las tensiones.*

*Hay tres tipos de rectas, de las que derivan otras variantes:*

*1. La forma más simple de recta es la horizontal. En la percepción humana corresponde a la línea o al plano sobre el cual el hombre se yergue o se desplaza.*

*La horizontal es por tanto la base protectora, fría, susceptible de ser continuada en distintas direcciones sobre el plano. Su frialdad y achatamiento constituyen el tono básico de esta línea, a la que podemos definir como la forma más limpia de la infinita y fría posibilidad de movimiento.*

*La horizontal es por tanto la base protectora, fría, susceptible de ser continuada en distintas direcciones sobre el plano. Su frialdad y achatamiento constituyen el tono básico de esta línea, a la que podemos definir como la forma más limpia de la infinita y fría posibilidad de movimiento.*

*2. El perfecto opuesto de esta línea es la vertical, que forma con ella ángulo recto; la altura se opone a la chatedad, el calor sustituye al frío: es lo contrario en un sentido tanto externo como interno. La vertical es, por tanto, la forma más limpia de la infinita y cálida posibilidad de movimiento.*

*3. El tercer tipo de recta es la diagonal, que, esquemáticamente, se separa en ángulos iguales de las anteriores. Su tendencia hacia ambas es equivalente, lo cual determina su tono interior: - reunión equivalente de frío y calidez. O sea, la forma más limpia del movimiento infinito y templado.*

Una vez que hemos revisado la descripción de la línea que hace Kandinsky, es necesario regresar nuevamente a las matemáticas. Para ello contaremos una pequeña historia a manera de ejemplo, para comprender, cómo nace una línea matemáticamente.

La historia plantea un problema básico de matemáticas en secundaria. Para ello utilicemos de nuevo el plano cartesiano y substituyamos los 4 ejes por puntos cardinales. El problema es el siguiente:

José viaja 4 kilómetros al este y 3 kilómetros al norte con respecto a su casa para ir al trabajo. Su casa es 0, el origen. ¿Cual sería la distancia mínima de su casa al trabajo?

Como sabemos la distancia mínima entre dos puntos es una recta. Tracemos esa recta tomando en cuenta la unión de esos puntos con sus valores. Fig. 2.19 Trazo de A(x0,y0) y B(x4,y3).

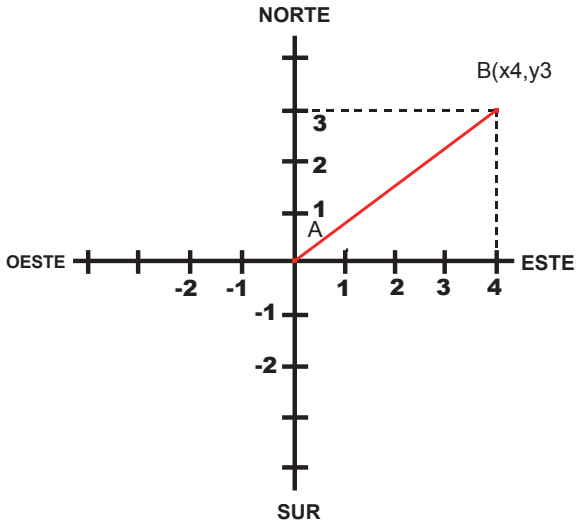



*Fig. 2.19*

Si unimos los 3 puntos involucrados se forma un triángulo-rectángulo, figura de la que sólo conocemos 2 valores: Los 4 kilómetros al ESTE y los 3 kilómetros al NORTE, pero queremos conocer la distancia recorrida entre ambos valores, a esa distancia pongamos una c.

Fig. 2.20. Triángulo rectángulo de Pitágoras.

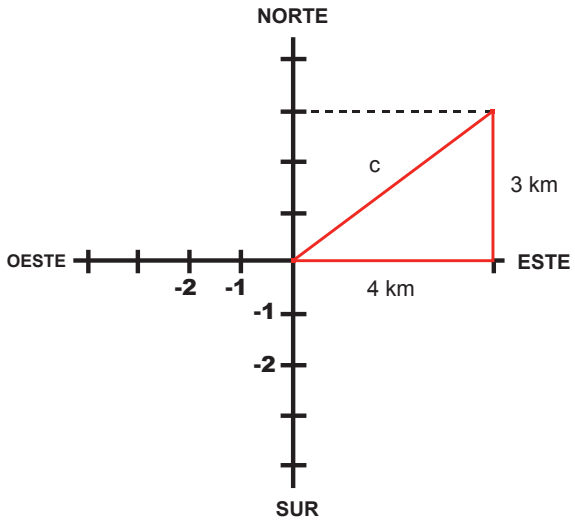



*Fig. 2.20*

La manera ideal de resolver éste problema, será aplicando el famoso teorema de Pitágoras.

Comencemos por la hipotenusa. La hipotenusa es el lado más largo del triángulo rectángulo, y es también, la distancia de la que queremos saber la medida.

A partir de ella, se desarrolla la fórmula como sigue:

$$c^2 = a^2 + b^2$$

Los otros dos lados opuestos a la hipotenusa son los catetos. Éstos tienen los valores: $4^2+3^2$. Pongamos en ambos catetos, dichos valores.

Fig. 2.21. $c^2= 4^2+3^2$

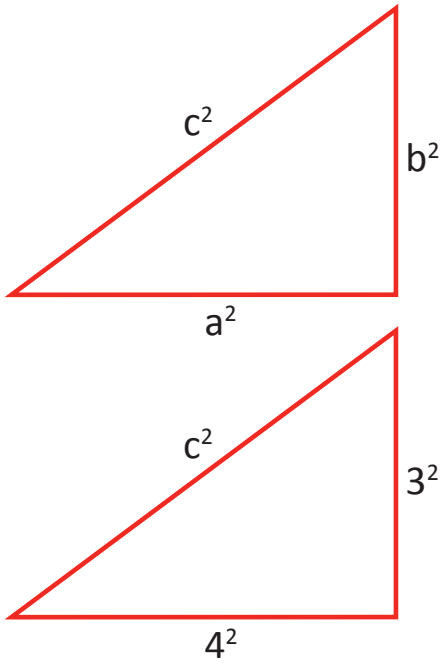



***Fig. 2.21***

Los valores se duplican por estar al cuadrado.

4X4=16. 3X3=9.

$$c^2= 16+9$$

Se suman.

$$c^2= 25$$

Finalmente se saca la raíz cuadrada para obtener el valor de la distancia.

$$c^2=\sqrt{25}$$

Por lo tanto, 5 km es la distancia que separa a José de su trabajo, y como podemos apreciar, el teorema de Pitágoras nos ayuda perfectamente a calcular la distancia entre 2 puntos:

Fig. 2.22. $a^2+b^2=c^2$

Expresada gráficamente obtenemos:

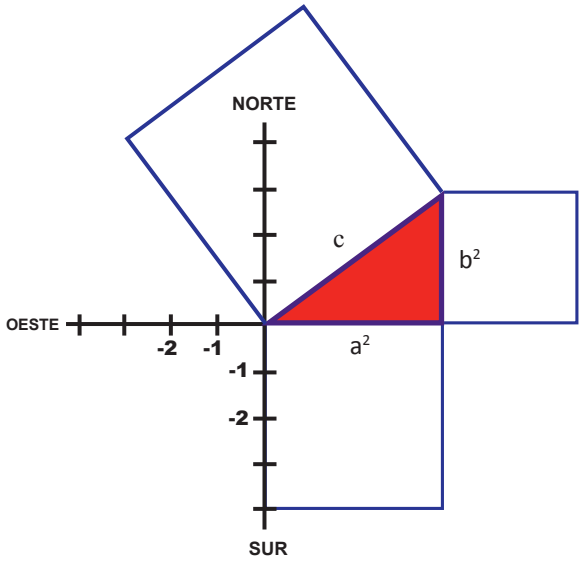



***Fig. 2.22***

Ahora substituyamos esas letras de la fórmula por las letras del plano cartesiano, con las letras X, Y.

Fig. 2.23. $x^2+y^2=c^2$

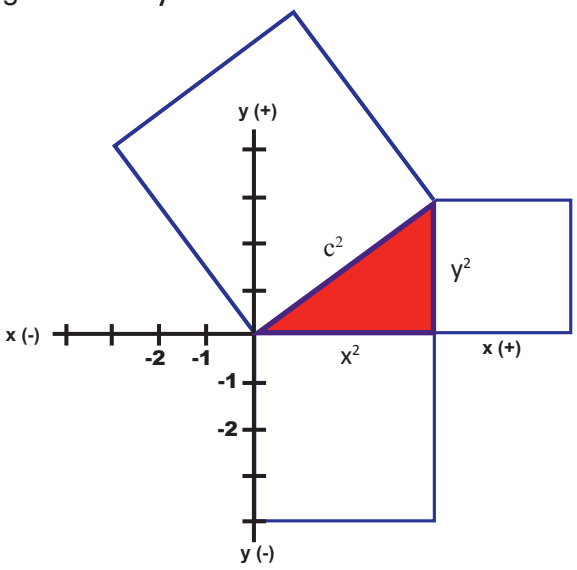



***Fig. 2.23***

Otra manera de expresar la misma formula; es poniendo desde el principio, la raíz cuadrada para que el resultado de c; que es la hipotenusa, pero sin elevarla al cuadrado.

$$\sqrt{a^2+b^2}=c$$

Una vez entendido esto, conviene explicar la ecuación de la recta. Con éste antecedente que refiere el teorema de Pitágoras, es menos complicado entender la ecuación:

$$y=mx+b$$

Ahora expliquemos en palabras, la lógica de la ecuación de la recta.

El valor numérico dado a Y es igual a la pendiente o inclinación expresado con m, multiplicado por el valor numérico dado a X más b, b es el valor numérico de la intersección o punto de cruce de la línea con el eje Y.

Aún no está clara la idea, por lo tanto, veamos algunos ejemplos que clarifiquen la lógica de la ecuación utilizando una recta cuando conocemos su punto y su pendiente. Recordemos que una pendiente tiene una inclinación y una dirección que puede ser hacia arriba o bien hacia abajo, a la derecha o la izquierda.

Pongamos un punto de origen: A(-6x,-5y) con una pendiente:

$$m=\frac{2}{3}$$

Ahora encuentra la ecuación de la recta que nace del punto: A (-6x,-5y) y que tiene la pendiente:

$$m=\frac{2}{3}$$

Para desarrollarlo gráficamente, tenemos que utilizar el plano cartesiano. Fig. 2.24. Plano cartesiano para graficar la recta.

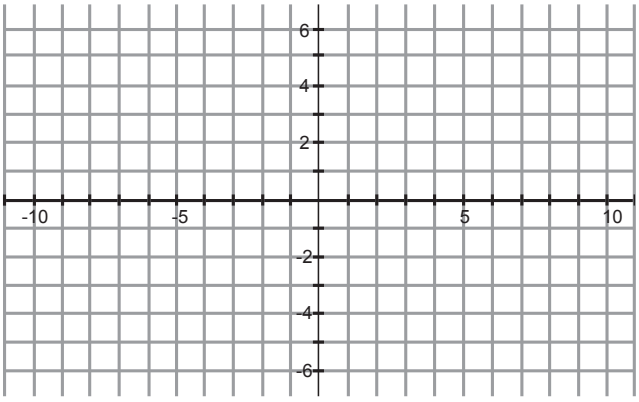



***Fig. 2.24***

El primer paso será graficar el punto. Fig. 2.25. A (-6x,-5y).

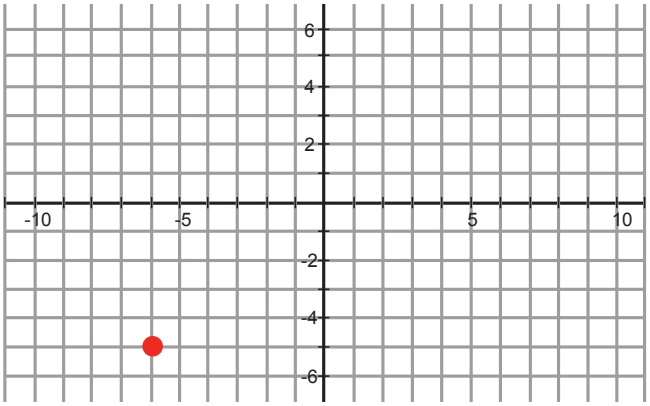



***Fig. 2.25***

nuestra pendiente es:

$$m = \frac{2}{3}$$

y es una pendiente positiva porque no tiene ningún signo de menos en la fracción, ello indica que a partir de ese punto empezará a trazarse hacia arriba una línea siguiendo un patrón: 2 puntos hacia arriba de manera vertical, 3 puntos hacia la derecha de manera horizontal como en las siguientes gráficas. Fig. 2.26. Pendiente.

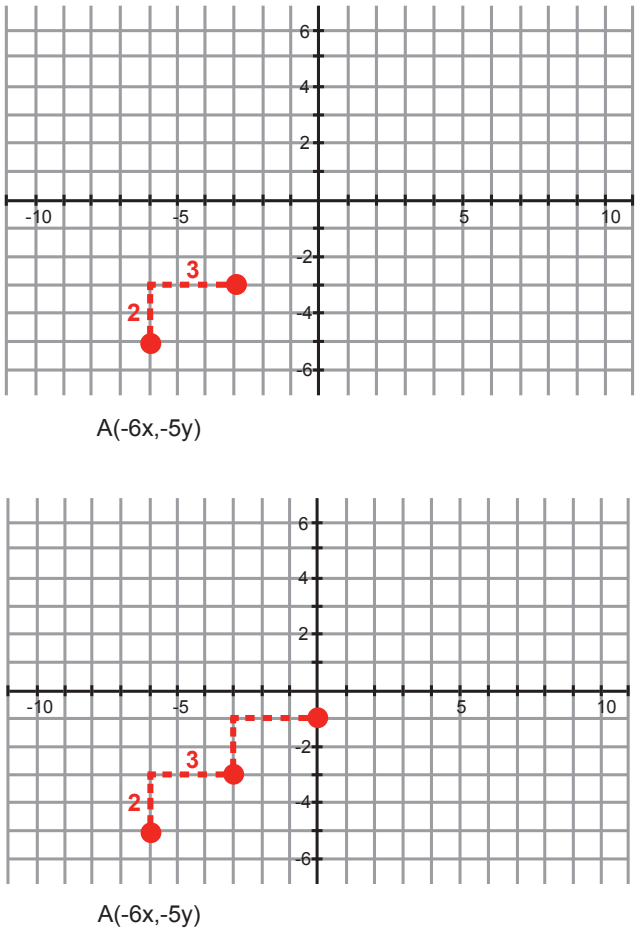
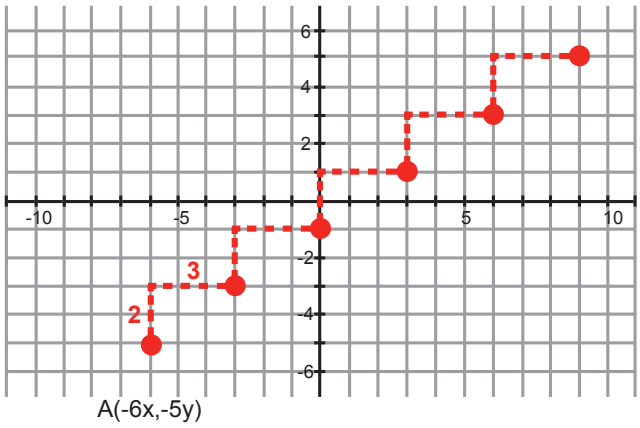






***Fig. 2.26***

Y así sucesivamente hasta poner el último punto para conformar la recta. Fig. 2.27. Pendiente de la recta.



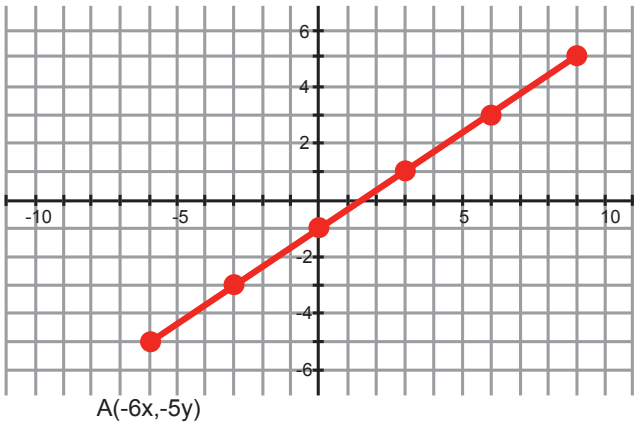




***Fig. 2.27***

Ahora vamos a localizar la ordenada de la recta que está determinada por el punto de intercepción con el eje Y, ese punto de cruce se representa con la letra b y es otra parte importante de la ecuación en cuestión, para apreciarla, la encerraremos en un círculo: Fig. 2.28. Ordenada de la recta.

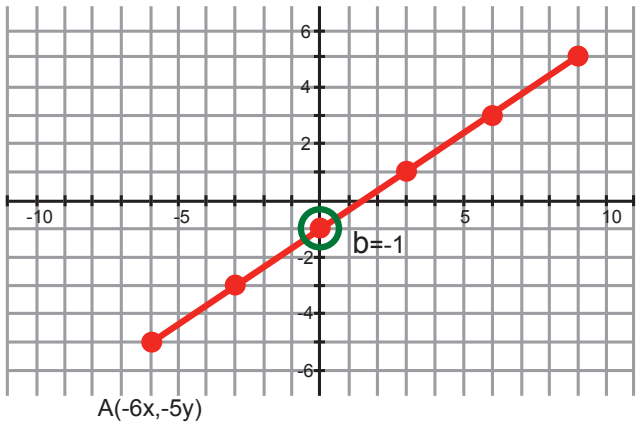



***Fig. 2.28***

Esta ordenada o punto de intercepción encerrado en el circulo de nombre b, es b =-1 y describe gráficamente cómo a partir de un punto origen, nace una recta progresivamente cuando se conoce el valor de la pendiente.

Ahora veamos su desarrollo numérico utilizando la famosa ecuación de la recta: **y=mx+b**

Tomando en cuenta esta información, tomemos el punto:

A (-6x,-5y)

La pendiente:

$$m = \frac{2}{3}$$

y la ordenada de origen: b= -1.

Sustituyamos nuestra ecuación de la recta con esta información para obtener:

$$y = \frac{2}{3}x-1$$

Sigamos con la comprobación. Punto:

A (-6x,-5y), Pendiente:

$$m = \frac{2}{3}$$

Ecuación: **y=mx+b**

El punto de origen con el que iniciamos es A (-6x,-5y) esta información ya la tenemos en el plano cartesiano: X=-6, Y=-5, ahora pondré estos valores en la ecuación de la recta tomando en cuenta la pendiente para comprobar que el valor de b es efectivamente -1.

-5 es igual a la pendiente dos tercios. Ésta pendiente se multiplica por X cuyo valor es -6 y se suma con b, así obtenemos:

$$-5 = \frac{2}{3}(-6)+b$$

Luego tenemos la pendiente dos tercios y -6, es decir, una fracción que se va a multiplicar con un entero. Para multiplicarlos, tengo que convertir al -6 también en una fracción, para hacerlo, agrego al -6 un -1:

$$-5 = \frac{2}{3}\left(\frac{-6}{-1}\right)+b$$

Multipliquémos las fracciones para obtener:

$$-5 = \frac{-12}{3}+b$$

Divido el -12 entre 3 para obtener -4.

$$-5 = -4+b$$

Ahora para finalizar esta comprobación es necesario convertir al -4 en su opuesto +4 un número positivo y hago la resta:

$$+4 - 5 = b \qquad +4 - 5 = -1$$

Esto nos da como resultado -1. b igual a -1. Así podemos decir que: Y cuyo valor es -5 es igual a la pendiente:

$$m = \frac{2}{3}$$

Multiplicado por X cuyo valor es -6 más b, cuyo valor es -1.

De esta manera se comprueba que la ecuación de la recta coincide con sus gráficas, y también se comprende como nace una recta a partir de un punto, siempre y cuando exista una pendiente.

Veamos otro ejemplo: Fig. 2.29. P (-8x, 5y).

La pendiente será:

$$m = -\frac{1}{4}$$

En este ejemplo la pendiente será negativa.

Lo primero que debemos hacer, es situar el punto:

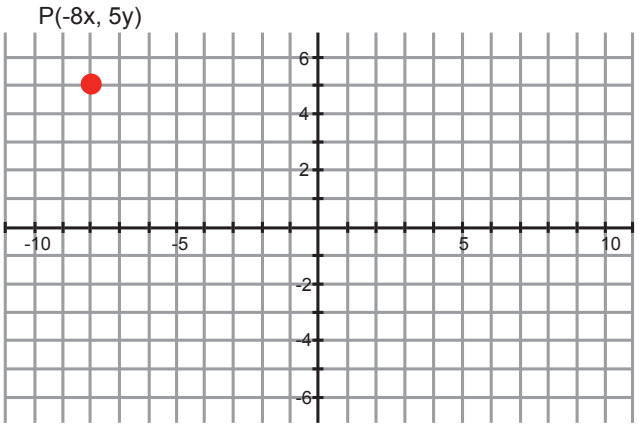



***Fig. 2.29***

En este ejemplo, el patrón de la pendiente es:

$$m = -\frac{1}{4}$$

Por lo tanto la secuencia es: 1 hacia abajo vertical y 4 hacia la derecha horizontal. Fig. 2.30. Pendiente.

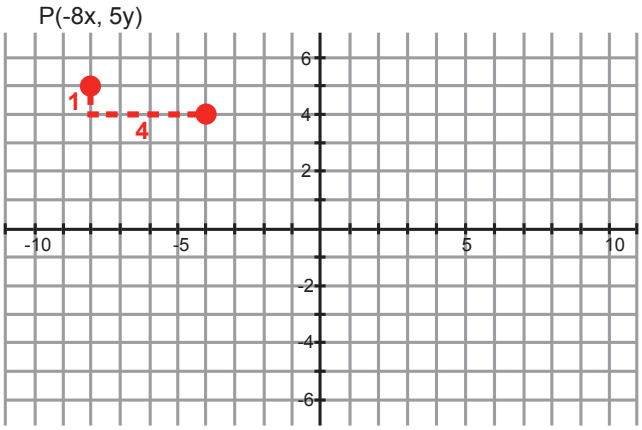



***Fig. 2.30***

Coloquemos todos los puntos siguiendo ésta secuencia y luego sobre ellos, tracemos la recta como en la gráfica anterior. Fig. 2.31. Pendiente de la recta.

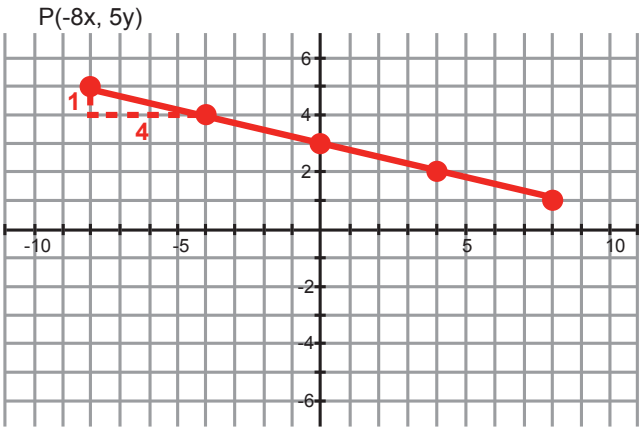



*Fig. 2.31*

Luego, como en el ejemplo anterior, localicemos el punto de intersección de la recta sobre el eje Y para determinar su valor. Su valor es b=3. Fig. 2.32. Ordenada de la recta.

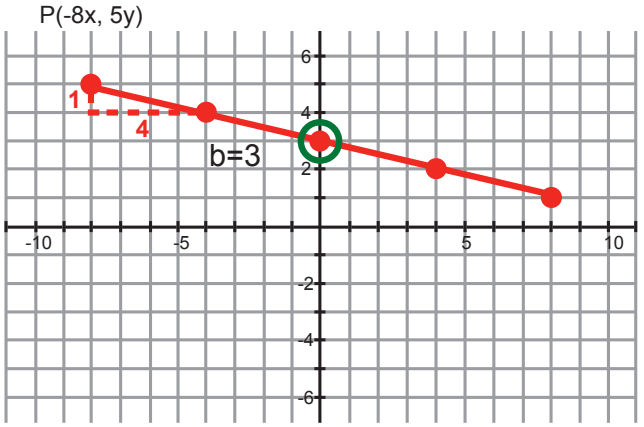



*Fig. 2.32*

Con esta información determino como será mi ecuación.

La pendiente es:

$$m = -\frac{1}{4}$$

La ordenada de origen es b = 3 y la ecuación es:

$$y = -\frac{1}{4}x+3$$

Hemos obtenido la ecuación del ejemplo por medio de gráficas, ahora comprobemos que efectivamente b es igual a 3.

Tenemos otra vez el punto

P (-8x, 5y).

La pendiente: $m = -\frac{1}{4}$

Y la ecuación: **y=mx+b**

El punto de origen con el que iniciamos es:

P (-8x, 5y).

Que en el plano cartesiano es:

X=-8, Y=5, reemplazamos los valores para la comprobación.

Y cuyo valor es 5 es igual a la pendiente menos un cuarto, que se multiplica por X cuyo valor es -8, más b.

$$5=-\frac{1}{4}(-8)+b$$

Las fracciones se multiplican, por ello se agregará un 1 debajo de -8 para obtener:

$$5=-\frac{1}{4}\left(\frac{-8}{-1}\right)+b$$

Se multiplican los numeradores y los denominadores, y también, los signos negativos:

$$5 = +\frac{8}{4} + b$$

Se hace la división de ocho entre cuatro:

$$5 = 2 + b$$

Para finalizar esta comprobación, convertimos el 2 en su opuesto: - 2. Se suma con 5 y obtenemos 3:

$$-2 + 5 = 3$$

b es igual a 3 y la lógica de la ecuación queda así: Y cuyo valor es -8 es igual a la pendiente:

$$m = -\frac{1}{4}$$

Multiplicado por X cuyo valor es 5, más b cuyo valor es -1. Ésta es la lógica de la ecuación de la recta y su visualización.

Ahora veamos el desarrollo de la aplicación interactiva. Para mayores referencias podemos consultar el tema de geometría analítica en el libro: CONAMAT. Matemáticas Simplificadas. Autores varios. Prentice Hall. 2009.

### 2.3.1. Interactivo ecuación de la línea.

Revisemos ahora el código fuente del interactivo denominado “Ecuación de la Línea”. Pero antes, describamos en que consiste:

El interactivo “Ecuación de la línea” consiste en una aplicación web que nos mostrará también, un plano cartesiano, pero a diferencia del primer interactivo, éste nos ofrecerá la posibilidad de mover en él dos puntos, y en base a su posición, obtener: la ecuación de la línea. Dicha aplicación, también podrá ser visualizada a través de los navegadores web.

Los lenguajes de programación que utilizaremos para elaborar dicho interactivo, serán los mismos que en el ejemplo del punto: HTML5, JavaScript y JSXGraph.

Las razones, ya las comentamos anteriormente, por lo tanto, pasemos directo a la explicación de la construcción de dicho interactivo que consta también de 3 archivos, uno principal llamado: linea.html, y otros dos, los archivos: jsxgraph.css, y jsxgraphcore.js, la hoja de estilos, y la biblioteca de códigos JavaScript.

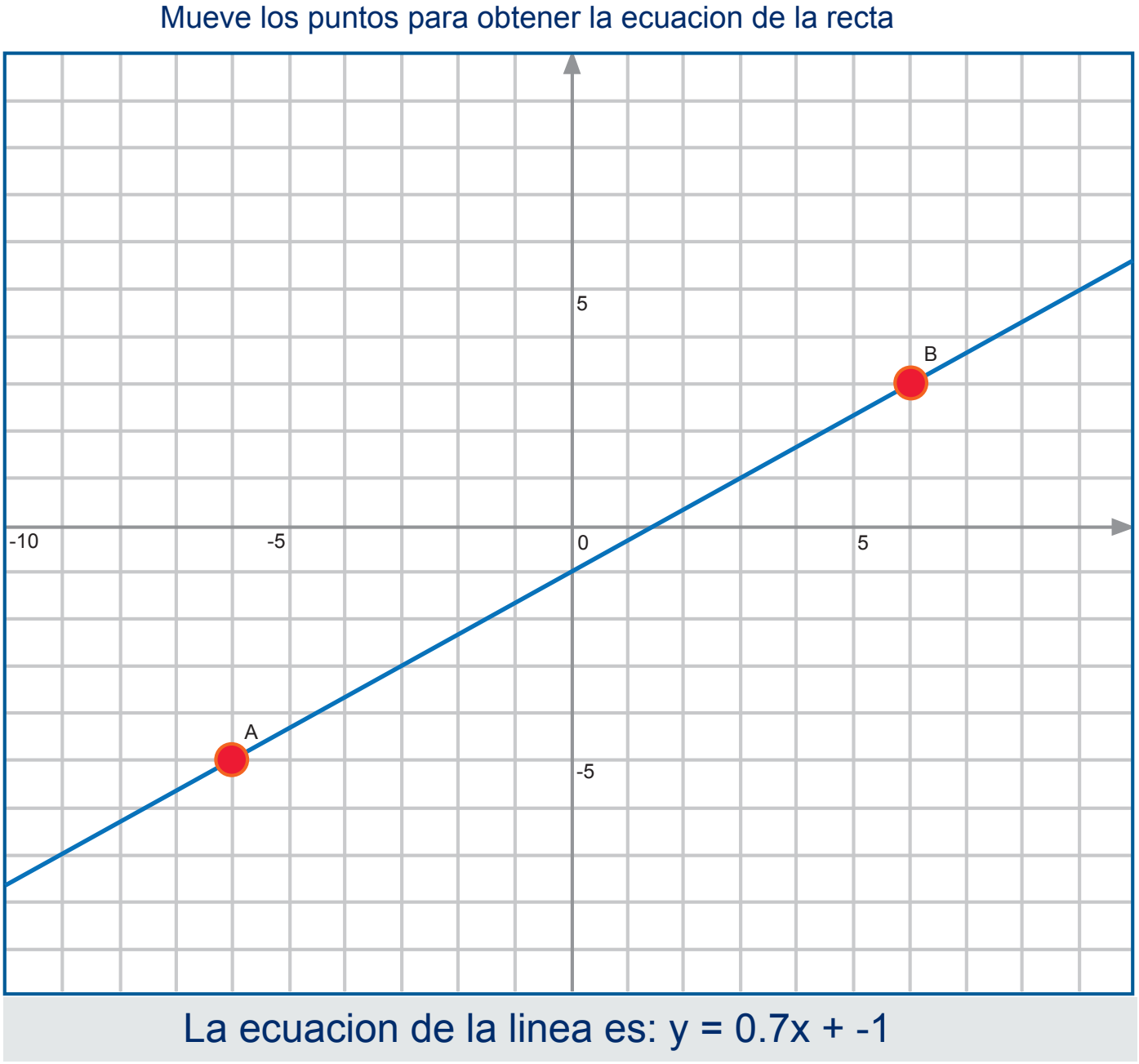



***Fig. 2.33***

Fig. 2.33. La imagen, pertenece a nuestro interactivo titulado la ecuación de la línea. En él, los valores numéricos de la pendiente, se muestran en números decimales en vez de fracciones. Por cuestiones de cálculos y programación, es mejor presentar este valor en sistema decimal, ya más adelante les diremos porqué, conforme analizamos el código de nuestro interactivo.

En el primer ejemplo de la explicación, la ecuación era: $y = \frac{2}{3}x+-1$

Y en este ejemplo, la ecuación es: **y=0.7+-1**

Pues bien, ambas son la misma ecuación, lo que aquí ocurrió, fue que la pendiente se dividió dándonos: 0.6666666677, así el valor se redondea a 0.7 gracias a una función de código que nos ayuda a evitar éstas enormes cantidades decimales.

## El Código fuente de nuestro interactivo: linea.html

```
<!DOCTYPE html>
<html lang=”en”>
<head>
<meta charset=”UTF-8”/>
<title>Ecuacion de la linea</title>
 <script src=”jsxgraphcore.js”></script>
 <link rel=”stylesheet” href=”jsxgraph.css” />
 <style>

.instrucciones {
            font-family:Arial, Helvetica, sans-serif;
            font-size:14px;
            color:#003366;
            font-weight:bold;
            text-align:center;
            width:600px;
 }
#textoEcuacion {
            font-family:Arial, Helvetica, sans-serif;
            font-size:20px;
            color:#003366;
            font-weight:bold;
            padding:6px;
            background:#EAEAEA;
            width:600px;
            text-align:center;
 }
 </style>
</head>

<body>
<div class=”instrucciones”>Mueve los puntos para obtener la ecuacion de la recta </div><br>
<div id=”plano” class=”jxgbox” style=”width:600px; height:500px;”></div>
<script>

var planoCartesiano = JXG.JSXGraph.initBoard(‘plano’,
 {
            boundingbox: [-10, 10, 10, -10],
            axis:true,
            grid:true,
            showCopyright:false,
            showNavigation:false
 });

var puntoA = planoCartesiano.create(‘point’, [0,1],
 {
            name:’A’,
            snapToGrid: true,
            size: 7,
   fillColor: ‘#E6413A’,
   highlightFillColor : ‘#4BB549’,
 });

var puntoB = planoCartesiano.create(‘point’, [4,-1],
 {
            name:’B’,
            snapToGrid: true,
            size: 7,
   fillColor: ‘#E6413A’,
   highlightFillColor : ‘#4BB549’,
 });
```

```
var linea = planoCartesiano.createElement('line',[puntoA,puntoB]);
puntoA.on('mousedrag', function() {
	obtenerEcuacionLinea()
});

puntoB.on('mousedrag', function() {
	obtenerEcuacionLinea()
});

function obtenerEcuacionLinea() {
	planoCartesiano.infobox.setProperty({strokeColor: '#000000'});
	var x1=puntoA.X();
	var y1=puntoA.Y();
	var x2=puntoB.X();
	var y2=puntoB.Y();
	if(x1==x2 || y1==y2) {
		if(x1==x2)
			document.getElementById('textoEcuacion').innerHTML = "x = "	+x1;
		if(y1==y2)
			document.getElementById('textoEcuacion').innerHTML = "y = "	+y1;
	} else {
		var parteUnoEcuacion =  ( y2 - y1 ) / ( x2 - x1 ) ;
		var parteDosEcuacion = y1 - x1 * ( y2 - y1 ) / ( x2 - x1 );
		var laEcuacion= "y = " +redondearNumero(parteUnoEcuacion)+"x + " +  redondearNumero(parteDosEcuacio
n);
		document.getElementById('textoEcuacion').innerHTML = "La ecuacion de la linea es: " + laEcuacion
	}
}

function redondearNumero(numeroARedondear) {
	return Math.round(numeroARedondear*10) / 10;
}
</script>
<div id="textoEcuacion"> </div>
</body>
</html>
```

Una vez que hemos dado un vistazo al código fuente, lo siguiente que haremos será fragmentar el código en partes o bloques como ya lo hemos hecho, para tratar de comprenderlo:

Bloque 1.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8"/>
<title>Ecuacion de la linea</title>
```

Al igual que en el ejemplo anterior, lo que este primer bloque de código nos dice es que tenemos un documento tipo HTML en idioma ingles. El formato de texto es UTF-8 para alfabetos latinos, cuyo nombre es: Ecuación de la línea.

Bloque 2.

```
<script src=”jsxgraphcore.js”></script>
```

Aquí nos está indicando, que vamos a insertar la biblioteca JavaScript: JSXGrap (Graficador de JavaScript), el cual nos va a proporcionar los componentes necesarios en la construcción visual de nuestro interactivo, estos componentes serán: el plano cartesiano, la cuadricula del plano, y sus divisiones, los dos puntos, la recta, y en general; todo el código que necesitamos para el funcionamiento de la aplicación.

Bloque 3.

```
<link rel=”stylesheet” href=”jsxgraph.css” />
```

En esta etiqueta, se está insertando el otro documento CSS (hoja de estilo en cascada) éste archivo también fue generado en JSXGraph y contiene los estilos predefinidos para el interactivo en: colores, tipos de letra, tamaños, etc.

Bloque 4.

```
<style>
.instrucciones {
        font-family:Arial, Helvetica, sans-serif;
        font-size:14px;
        color:#003366;
        font-weight:bold;
        text-align:center;
        width:600px;
}
```

Este bloque describe los estilos propios del HTML. El estilo ‘instrucciones’ le da formato: (tamaño, color, tipo de letra) al texto que va a indicar las instrucciones al usuario.

Bloque 5.

```
#textoEcuacion {
        font-family:Arial, Helvetica, sans-serif;
        font-size:20px;
        color:#003366;
        font-weight:bold;
        padding:6px;
        background:#EAEAEA;
        width:600px;
        text-align:center;
 }
 </style>
</head>
<body>
```

Aquí se describen los estilos de ‘textoEcuacion’ que le da formato (tamaño, color, tipo de letra) al cuadro de texto que muestra: la ecuacion de la linea.

Bloque 6.

```
<div class=”instrucciones”>Mueve los puntos para obtener la ecuacion de la recta </div><br>
```

Aquí se incluye el elemento HTML de tipo  con las instrucciones del interactivo. Ésta línea se puede editar según se desee, para cambiar los estilos en éste texto.

Bloque 7.

```
<div id=”plano” class=”jxgbox” style=”width:600px; height:500px;”></div>
```

En este bloque, se incluye otro elemento HTML de tipo div, éste será el que se transformará en el plano cartesiano. Sus propiedades son:

id=”plano”: Es el identificador del elemento. Si se edita este valor, debe editarse la linea 57, es recomendable no modificar este valor.

class=”jxgbox”: Indica que se debe utilizar el estilo (colores, tamaños, etc.) definido por el framework JSXGraph. Este valor no se debe modificar.

style=”width:600px; height:500px;”: Especifica el tamaño en pixeles del plano cartesiano. Este valor se puede modificar si se desea cambiar el tamaño del plano.

Bloque 8.

```
var planoCartesiano = JXG.JSXGraph.initBoard(‘plano’,
{
        boundingbox: [-10, 10, 10, -10],
        axis:true,
        grid:true,
        showCopyright:false,
        showNavigation:false
});
```

En este bloque comienza la programación del código principal de la aplicación, que define el comportamiento del interactivo.  Genera un plano cartesiano que se denominará ‘planoCartesiano’ con las medidas específicas.

boundingbox: Especifica las dimensiones del plano. Para el eje X iniciar en -10 y termina en 10. Para el eje Y inicia en 10 y termina en -10. Este valor se puede modificar según el tamaño con el que se desee construir el plano

axis:true : Indica que se deben mostrar los ejes X y Y.

grid:true : Indica que se debe mostrar la cuadricula (líneas del plano)

Bloque 9.

```
var puntoA = planoCartesiano.create('point', [0,1],
{
        name:'A',
        snapToGrid: true,
        size: 7,
  fillColor: '#E6413A',
  highlightFillColor : '#4BB549',
});
```

A continuación se indica que debe crearse un punto sobre el plano cartesiano y name:'A' será el nombre que utilizarémos para identificar ese punto. Este valor se puede modificar.

snapToGrid:true : Indica que el punto debe ajustarse a las intersecciones de las líneas, así el punto siempre mostrara coordenadas enteras y no decimales. No se recomienda modificar este valor.

size: 7 : Indica el tamaño del punto. Este valor se puede modificar según se desee.

fillColor: '#E6413A' : Es el código hexadecimal del color que tendrá el punto, en este caso es un tono rojo. Este valor se puede modificar.

highlightFillColor : '#4BB549': Es el código hexadecimal del color que tendrá el punto cuando el cursor del mouse este posicionado sobre él, en este caso es un tono verde. Este valor también se puede modificar.

Bloque 10.

```
var puntoB = planoCartesiano.create('point', [4,-1],
{
        name:'B',
        snapToGrid: true,
        size: 7,
  fillColor: '#E6413A',
  highlightFillColor : '#4BB549',
});
```

A continuación se indica que debe crearse otro punto sobre el plano cartesiano. name:'B' : Será el nombre que utilizarémos para identificar este otro punto. Su ubicacón será en (4,-1). Este valor se puede modificar.

snapToGrid: true : Indica que también debe ajustarse a las intersecciones de las líneas, así, el punto siempre mostrara coordenadas enteras y no decimales. No se recomienda modificar este valor.

size: 7 : Indica el tamaño del punto. Este valor se puede modificar según se desee.
fillColor: '#E6413A' : Es el código hexadecimal del color que tendrá el punto, en este caso es un tono rojo. Este valor se puede modificar.

highlightFillColor : '#4BB549': Es el código hexadecimal del color que tendrá el punto cuando el cursor del mouse este posicionado sobre él, en este caso es un tono verde. Este valor se puede modificar.

Bloque 11.

```
var linea = planoCartesiano.createElement('line',[puntoA,puntoB]);
```

En esta línea de código, se indica que debe crearse una linea desde el punto A al punto B.

Bloque 12.

```
puntoA.on('mousedrag', function() {
        obtenerEcuacionLinea()
});
```

Aquí se indica que cuando se arrastre el punto A por el plano, se deberá ejecutar una función que se llama obtenerEcuacionLinea (esta última función se describe líneas abajo).

Bloque 13.

```
puntoB.on('mousedrag', function() {
        obtenerEcuacionLinea()
});
```

Al igual que el código anterior, se indica que cuando se arrastre el punto B por el plano, se debera ejecutar la misma función llamada obtenerEcuacionLinea (hablaremos de ésta más adelante).

Bloque 14

```
function obtenerEcuacionLinea() {
        planoCartesiano.infobox.setProperty({strokeColor: '#000000'});
```

Esta función es la responsable de generar la ecuación de la recta apoyándose de los siguientes códigos en los bloques 15 y 16.

Bloque 15

```
        var x1=puntoA.X();
        var y1=puntoA.Y();
        var x2=puntoB.X();
        var y2=puntoB.Y();
```

Aquí obtenemos las variables que se obtienen de las coordenadas X, Y, de los puntos A, B. Que se cargaran al momento de mover los puntos en el plano cartesiano.

Bloque 16

```
y = ( y2 - y1 / x2 - x1 ) x + [ y1 - x1  * ( y2 - y1 / x2 - x1 ) ]
```

Éste bloque es precisamente la fórmula de la ecuación de la recta en su forma: dos puntos; una recta, donde el resultado de la ecuación se calcula a partir de 2 puntos. Veremos también una imagen de esto.

Bloque 17

```
if(x1==x2 || y1==y2) {
                    if(x1==x2)
                              document.getElementById('textoEcuacion').innerHTML = "x = "          +x1;
                    if(y1==y2)
                              document.getElementById('textoEcuacion').innerHTML = "y = "          +y1;
```

La función de ésta condicionante, es sólo para verificar la alineación de los puntos en ambos ejes: X,Y.

```
if(x1==x2 || y1==y2) {
```

Y nos dice: Si se cumple esta validación, significa que los 2 puntos están alineados en alguno de los ejes, por lo tanto, no podremos obtener la ecuación. Luego tenemos:

```
if(x1==x2)
    document.getElementById('textoEcuacion').innerHTML = "x = "      +x1;
```

Esta nos dice: Si los 2 puntos están alineados en el eje X, entonces no es posible obtener la ecuación, por lo tanto el valor que nos dará, sólo será su posición en la coordenada "X" en donde se encuentre en ese momento. por ejemplo: X=4 o X= 7 etc. Después tenemos:

```
if(y1==y2)
    document.getElementById('textoEcuacion').innerHTML = "y = "      +y1;
```

Esta parte de código nos dice lo mismo que la linea anterior, pero su aplicación es ahora al eje Y: Si los puntos están alineados en el eje Y, entonces no es posible obtener la ecuación, por lo tanto el valor que nos dará, será sólo su posición en el número de la coordenada "Y" en donde se ubique en ese momento.

Bloque 18

```
} else {
                    var parteUnoEcuacion =  ( y2 - y1 ) / ( x2 - x1 ) ;
                    var parteDosEcuacion = y1 - x1 * ( y2 - y1 ) / ( x2 - x1 );
```

Este bloque de código nos está diciendo: Si los puntos no están alineados horizontalmente en X o verticalmente en Y, entonces SI podemos obtener la ecuación de la recta según la fórmula: ecuación de la recta a partir de dos puntos.

Para fines de claridad en el código, se dividió esta fórmula en dos partes:

```
var parteUnoEcuacion =  ( y2 - y1 ) / ( x2 - x1 ) ;
var parteDosEcuacion = y1 - x1 * ( y2 - y1 ) / ( x2 - x1 );
```

Presentado de otra manera es: Parte uno = ( y2 - y1 / x2 - x1 ) x
Parte dos = y1 - x1  * ( y2 - y1 / x2 - x1 )

Ahora hagamos una pausa para estudiar la forma de ésta ecuación y su lógica, pues tal vez haya dudas al respecto y para aclararlas, veamos lo siguiente. La ecuación, es exactamente la misma que y=mx+b.

Ya mencionamos que para elaborar éste interactivo, utilizamos una variante de la ecuación: y = mx + b, esta variante es la ecuación de la recta a partir de dos puntos dados. Se utilizó ésta forma por dos razones; la primera porque para el usuario, es más claro entender la lógica de la fórmula de la recta cuando simplemente mueve dos puntos en un plano cartesiano, y la segunda, por una cuestion de claridad en la programación, pues ésta forma, se adecúa al objetivo de hacer más entendible el código. Pues bien, ahora expliquemos ésta forma variante de y = mx + b. Que es: y-y1 = (y2-y1) / (x2 - x1) (x - x1), ésta se logra cuando obtenemos los valores de las 4 coordenadas de 2 puntos dados. (x1,y1) y (x2,y2). Veamos esta expresión:

$$\frac{y-y^1}{y^2-y^1} = \frac{x-x^1}{x^2-x^1}$$ O bien: $$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

Ahora pongamos un ejemplo:

Supongamos que tengo dos puntos. Punto 1 está en: (x1=1 , y1=2) y punto 2 en: (x2=2, y2=4).

Reemplazo estos valores en la fórmula:

$$y - 2 = \frac{4 - 2}{2 - 1}(x - 1)$$

Ahora desarrollo mi ecuación restando los valores de la fracción: $y - 2 = 2(x - 1)$

Aplicamos una propiedad distributiva: $y - 2 = 2x - 2$

Para finalmente obtener. Fig. 2.34. $y = 2x$

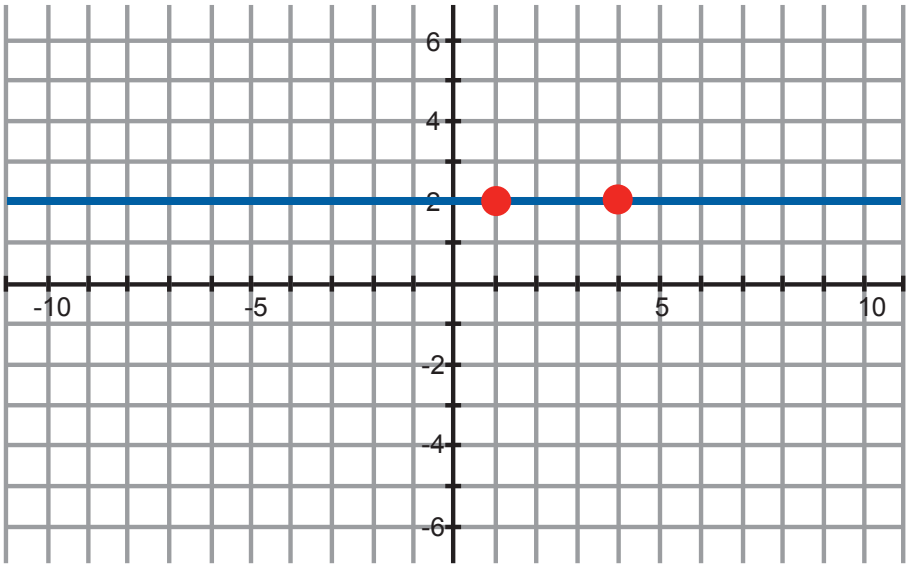



***Fig. 2.34***

Continuemos revisando el código.

Bloque 19

```
var laEcuacion= “y = “ +redondearNumero(parteUnoEcuacion)+”x + “ + redondearNumero(parteDosEcuacion);
```

En esta parte del código, se define otra de nuestras variables, la variable: ‘laEcuacion’ la cual contendrá el texto de la ecuación de la recta. Ésta debe quedar de la siguiente manera: y = parteUnoEcuacion x + parteDosEcuacion.

En algunos casos, debido a los cálculos matemáticos, se obtienen números con decimales al infinito, por eso se creó otra función llamada redondearNumero(), que como su nombre lo indica, redondea un número a su valor numérico superior como lo veremos en el bloque 21. Ésta función se activa cuando obtenemos la ecuación a partir de sus dos partes, esto lo vimos en el bloque anterior.

Bloque 20

```
                document.getElementById(‘textoEcuacion’).innerHTML = “La ecuacion de la linea es: “ + laEcuacion
        }
}
```

Una vez que se han realizado todos estos cálculos, basta con aplicar color y estilo en HTML para mostrarlos en pantalla.

Bloque 21

```
function redondearNumero(numeroARedondear) {
        return Math.round(numeroARedondear*10) / 10;
}
</script>
```

Aquí aparece de nuevo la función llamada: redondearNumero, que recibe como parámetro un número con decimales y lo redondea a su número superior. Por ejemplo, si recibe 4.6566666666666666, el resultado final será: 4.66. De esta manera el cálculo de la pendiente de la recta es mucho más aproximado a la forma de la ecuación que buscamos representar.

Bloque 22

```
<div id=”textoEcuacion”> </div>
</body>
</html>
```

Finalmente, se incluye el elemento HTML de tipo  que servirá para mostrar en pantalla la ecuación de la recta. Las propiedades de este elemento son: id=”textoEcuacion”: el elemento donde se cargará la información calculada en los bloques anteriores.

Básicamente, este es todo el código fuente del interactivo. Para mayor referencia, revisen el CD-Rom con los archivos de la aplicación, éste material les permitirá estudiar su construcción y funcionamiento. Ahora pasemos a nuestro último tema: El plano.

Nuestro último tema a desarrollar es el plano, en este apartado comprenderemos como nace el espacio bidimensional donde habitan los puntos y las líneas.

**2.4. El plano.**

En planilandia, los puntos, las líneas y las figuras geométricas se mueven en un plano.

El plano es el escenario donde viven los seres bidimensionales, o más bien; el soporte donde han sido trazados.

Para Kandisky, el plano es la superficie material donde será plasmado el contenido de la obra pictórica. Veamos su definición: "*Por plano básico se entiende la superficie material llamada a recibir el contenido de la obra. Será denominado con las letras PB.*

*El PB esquemático está limitado por 2 líneas horizontales y 2 verticales y adquiere así, en relación al ambiente que lo rodea, una entidad independiente".* (**Kandinsky.** *Punto y línea sobre el plano. Contribución al análisis de elementos pictóricos. Paidós Estética. pag.* 103).

Por lo tanto, como entidad independiente, ese soporte, bien podría ser el lienzo de un cuadro donde Kandinsky pintó su arte abstracto. Un arte muy cercano a los entes matemáticos.

Reflexionando lo anterior podriamos decir:
Un punto por si solo es un ente estático, sin dimensión. El desplazamiento de un punto genera una línea. El desplazamiento de una línea, genera un plano. Y finalmente, el desplazamiento de un plano, genera un volumen.

Como bien sabemos, el plano crece en X y Y. Ambos ejes cartesianos, son los únicos que conoce planilandia. Ahora la siguiente pregunta es. Si el plano es bidimensional. ¿Por qué es necesaria una representación 3D para visualizar un plano? La respuesta es, para poder visualizar los planos inclinados. El plano inclinado es el siguiente nivel de representación que hace necesaria la aparición del eje Z. Fig. 2.35. El plano inclinado.

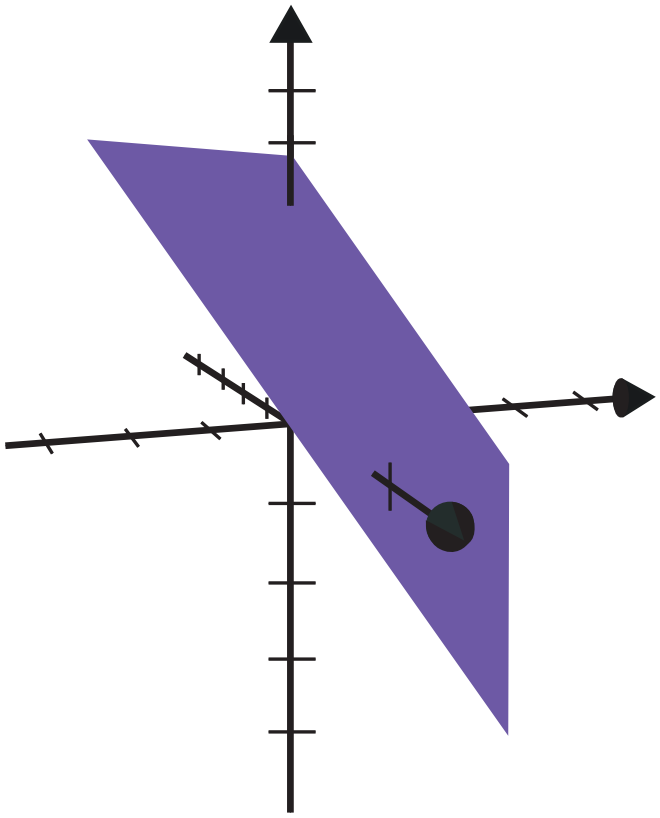


***Fig. 2.35***

La siguiente cuestión es: ¿Cómo podríamos conocer su inclinación, posición, y en general, su descripción de manera abstracta? Con la ecuación del plano y su representación gráfica.

Para entender la lógica de esta ecuación, es necesario ver algunas imágenes. La representación del plano matemático se vuelve mucho más compleja que la representación del punto y la línea cuando intentamos comprender su ecuación, sin embargo, no es una tarea imposible, más adelante desarrollaremos algunos ejemplos, pero antes visualicemos gráficas de planos como ejemplos introductorios.

El primer paso para entender el plano, es visualizar el plano cartesiano tridimensional, imaginando algunas de sus posiciones. Cada plano que nace es un universo y pueden coexistir en él, tantos planos como universos paralelos en nuestro mundo tridimensional.

Reflexionando sobre esto, nuestros físicos hablan de la cuarta dimensión haciendo referencia al tiempo.

El presente dura alrededor de 3 a 4 segundos perceptibles por el cerebro humano. 5 a 6 segundos después, es el pasado inmediato, en ese lapso pueden ocurrir circunstancias irremediables como romper un vaso o impactarse al conducir un vehículo. En 2 o 3 segundos nuestro destino puede cambiar para siempre. En 4D podrían existir diversos universos paralelos de tercera dimensión, hablamos de mecánica cuántica y líneas de espacio-tiempo que se afectan con cada acción o decisión que tomamos en la vida cotidiana. La teoría de cuerdas habla de posibilidades y nos dice; en un espacio-tiempo paralelo, no se extinguieron los dinosaurios, en otro, Alemania ganó la segunda guerra mundial.

Respecto a este tema, existe toda una gama de especulaciones y escritos de metafísica y ciencia ficción sobre como sería un universo de cuarta dimensión y los seres que lo habitan, sin embargo, nosotros como seres 3D, no podemos entender completamente la cuarta dimensión, así como los seres bidimensionales no entienden la tercera dimensión.

Sin embargo, una forma de representar la cuarta dimensión es si registramos el rastro de movimiento o acción de un objeto tridimensional en el espacio-tiempo con sus distintas variables y posiciones. El hipercubo es una figura muy básica de la cuarta dimensión. Si podemos registrar el desplazamiento de cubos o pirámides, o graficar formas basadas en el hipercubo, estaríamos intentando representar objetos de cuatro dimensiones utilizando gráficos por computadora en programas de representación 3D. Con el tema de la cuarta dimensión, podriamos hacer otra tesis.

Fig. 2.36. Reminiscencias de la cuarta dimensión.

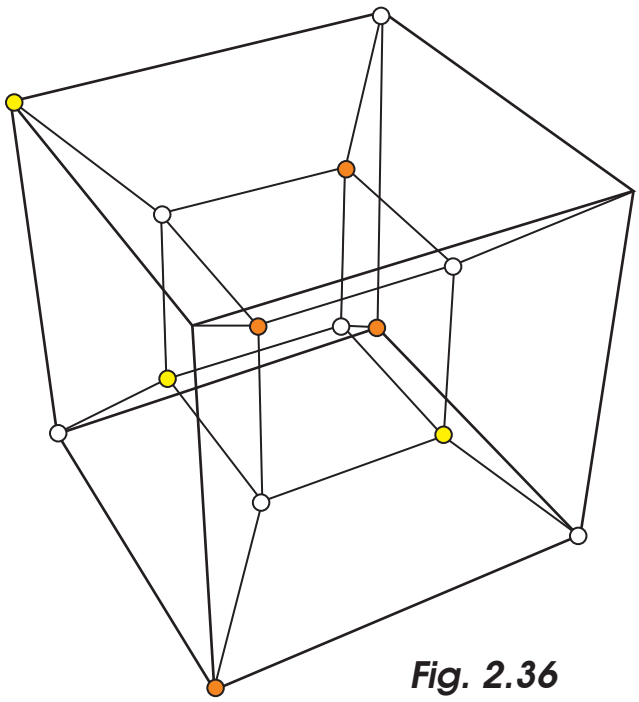


***Fig. 2.36***

Pero regresando al tema en cuestión, es importante que agreguemos al plano cartesiano bidimensional un eje más, el eje **z**. Como en la imagen. Fig. 2.37. El plano cartesiano tridimensional.

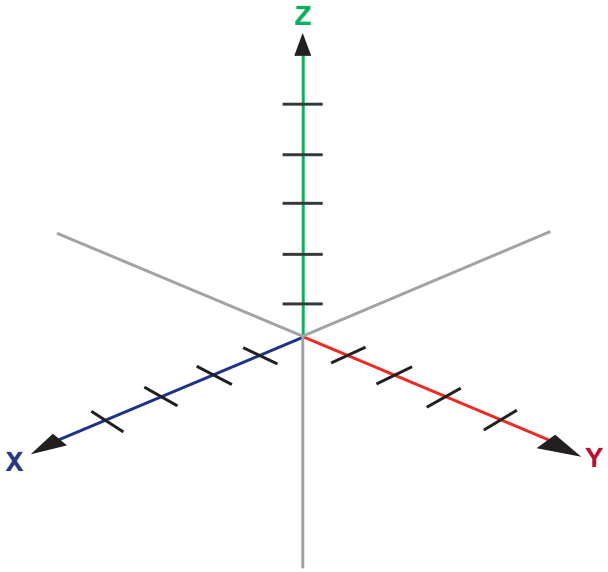



*Fig. 2.37*

Los planos en 3 dimensiones son fáciles de visualizar si los pensamos de manera tridimensional con los ejes **x, y, z**. Fig. 2.38 y Fig. 2.39. Plano Cartesiano 3D. Vista en 2D y 3D.

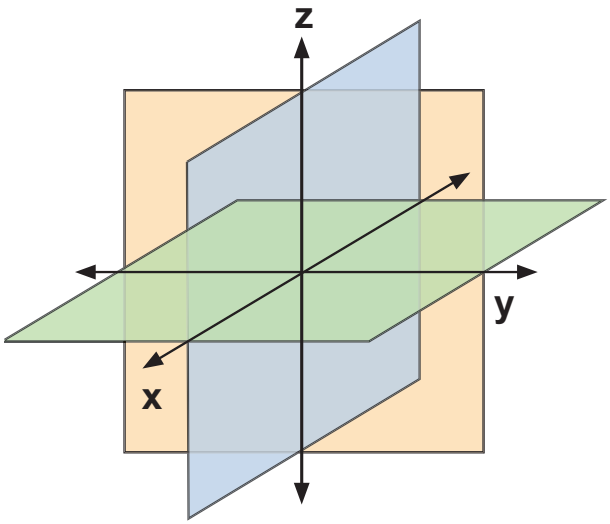



*Fig. 2.38*

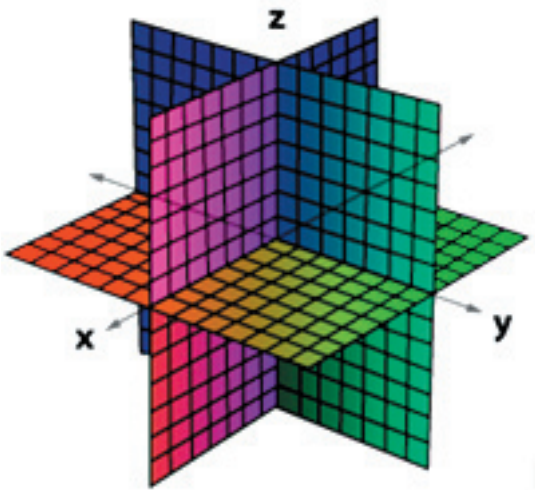



*Fig. 2.39*

En los libros de matemáticas, Z es el eje que representa la altura en vez de Y, ésta representación se obtiene cuando giramos el eje Z en la posición de Y, Pero en los programas de diseño 3D, Z representa la profundidad, no la altura. Fig. 2.40. El eje Z como profundidad.

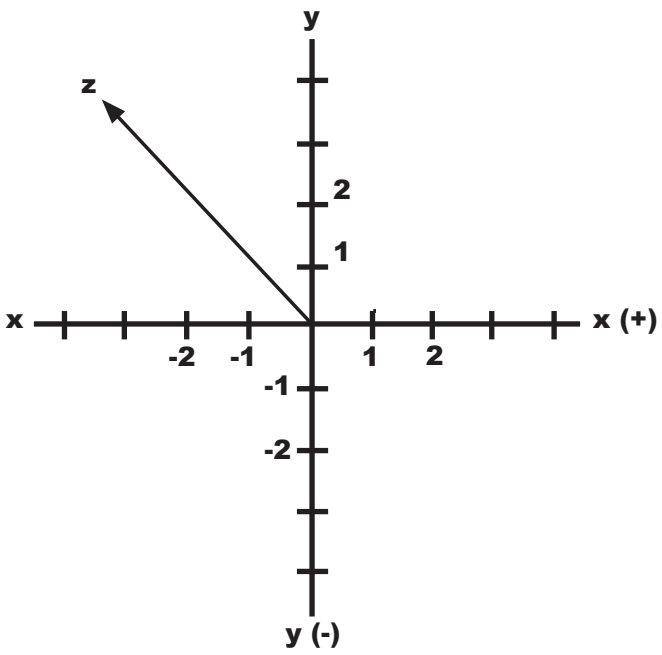



*Fig. 2.40*

Lo siguiente que haremos será visualizar un plano a la vez, generado a través de un valor dado a una coordenada. Fig. 2.41. x=2.  Fig. 2.42. z=2. Fig. 2.43. y=1.

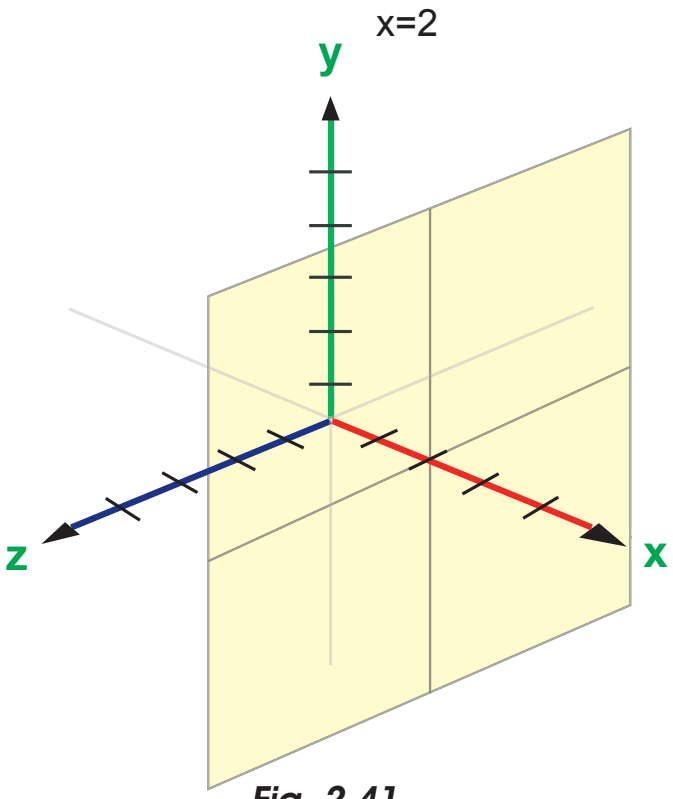



*Fig. 2.41*

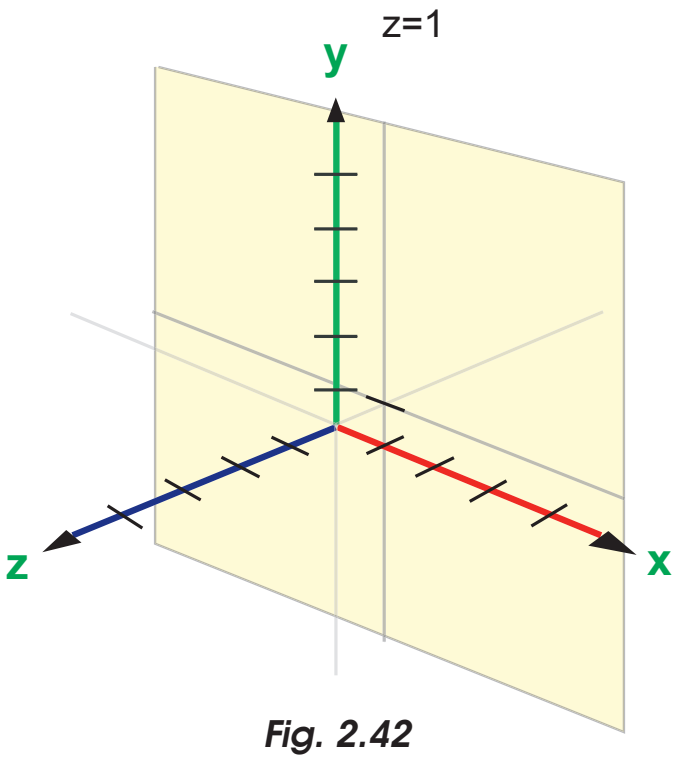



*Fig. 2.42*

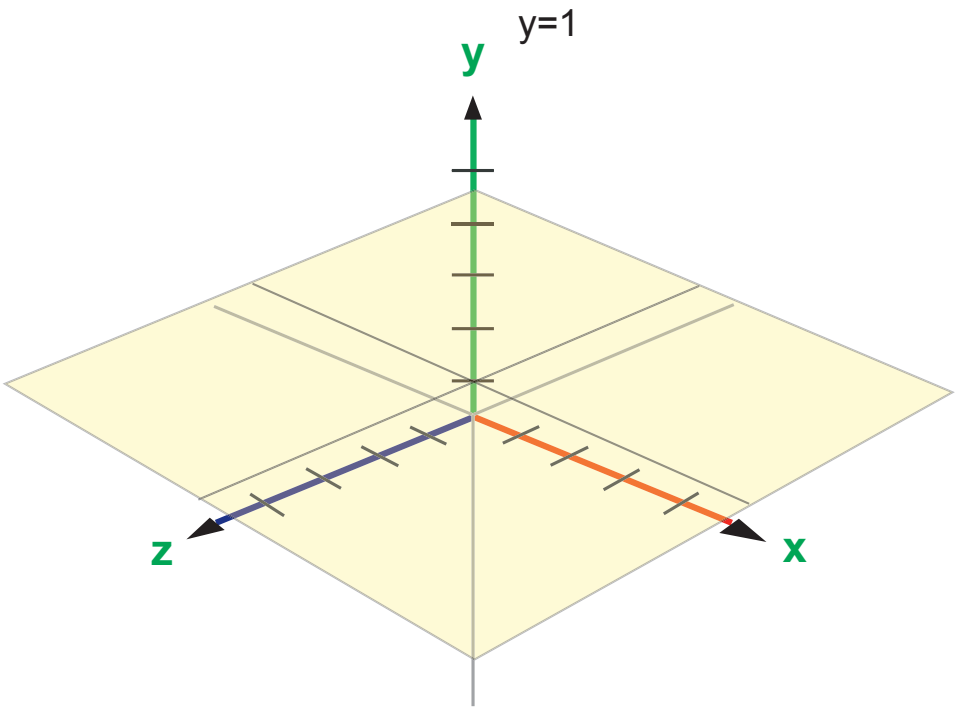



*Fig. 2.43*

Veamos ahora otras posibilidades en gráficos generados por computadora. Fig. 2.44.
El ejemplo **z=y** genera un plano ubicado exactamente entre los ejes **z, y.**

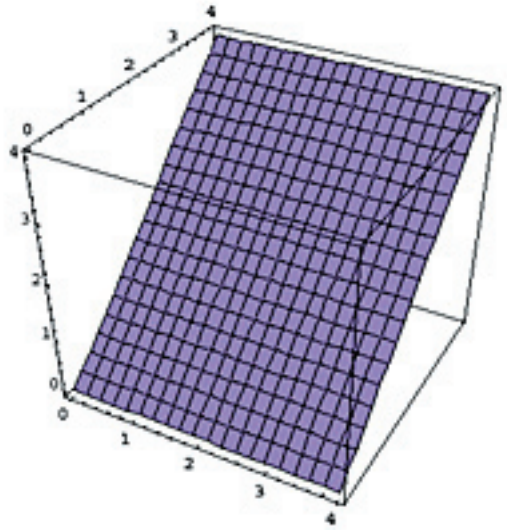


*Fig. 2.44*

Del mismo modo, el ejemplo **z=x** genera un plano ubicado en medio de los ejes **z, x.** Mismo que también puede ser **x=z.** Recordemos que en matemáticas el orden de los factores no altera el producto y también que en 3D, puedo levantar, girar y desplazar mi plano a voluntad. Fig. 2.45. z=x.

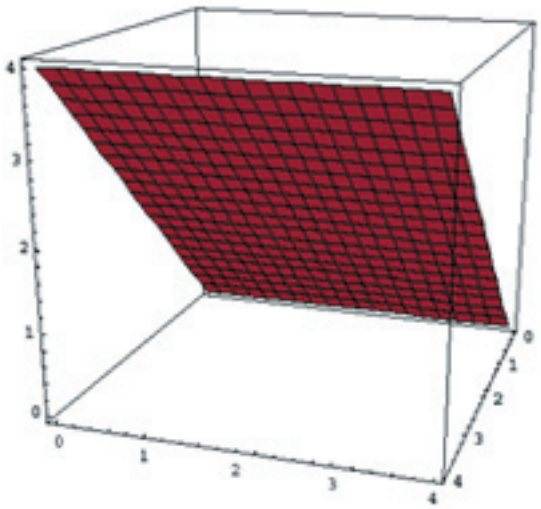


*Fig. 2.45*

Hasta ahora hemos visto algunas ecuaciones introductorias muy simples que generan posiciones de planos básicos. A partir de aquí la explicación se pone mucho más interesante y el nivel de complejidad aumenta, porque vamos a comprender como nace un plano a partir de cuatro factores fundamentales: 1) La generación de un plano a partir de puntos, puntos con una ubicación en nuestro plano cartesiano tridimensional, 2) La unión de esos puntos para conformar rectas con una pendiente, muy similar a lo que hicimos con la ecuación de la recta, 3) La generación del plano a partir de la unión de éstas rectas o puntos, para con esta información, llegar a la ecuación del plano. La ecuación general del plano es: **Ax+By+Cz=D.**

Ahora analicemos cada parte de esa ecuación, las letras A, B y C, se refieren a un punto ubicado en un sistema de tres coordenadas, algo como esto: A(x,y,z), B(x,y,z) y C(x,y,z), que es igual a D. D engloba el valor total de la ecuación. Ahora desarrollemos algunos ejemplos desde cero partiendo de los cuatro factores fundamentales mencionados en el párrafo anterior: 1) Graficación de puntos, 2) Unión de puntos para generar rectas, 3) Conformación de planos a partir de rectas con su descripción matemática, y 4) La conformación final de la fórmula del plano a partir de la fusión de las ecuaciones de recta y su despeje. Hagámoslo paso a paso y con metodología.

**Ejemplo 1:**

**Paso 1.** En un plano cartesiano tridimensional vamos a graficar los siguientes puntos:

Fig. 2.46. **A(x=1, y=0, z=0). B(x=0, y=1, z=0). C(x=0, y=0, z=1).**

Cada punto tiene un valor determinado para cada uno de los ejes: x, y, z.

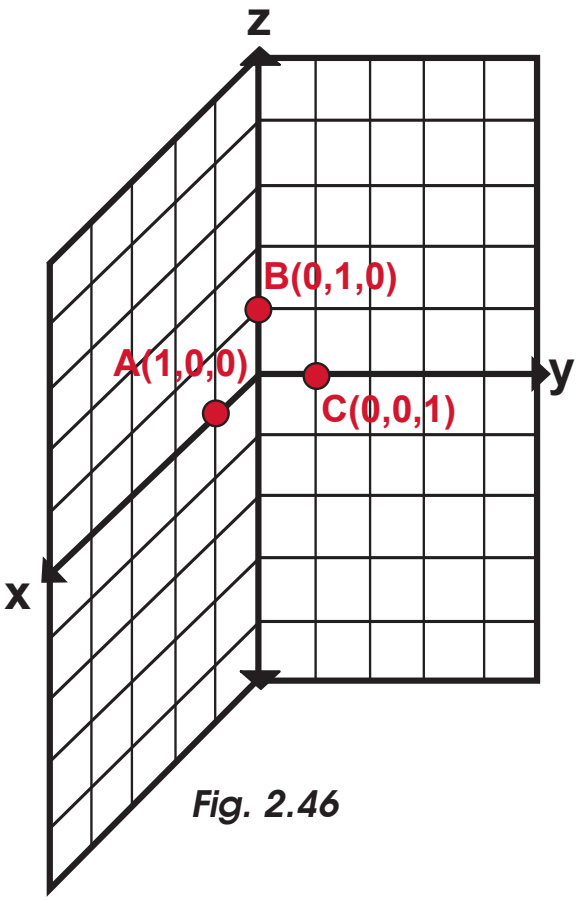



*Fig. 2.46*

**Paso 2.** Vamos a unir los puntos de la gráfica anterior para conformar un triángulo.

Este triángulo es de hecho, un plano.

Ahora en base a él, determinemos su ecuación implícita.

Fig. 2.47. Rectas-trazos de: **x** a **z**, y de **y** a **z**.

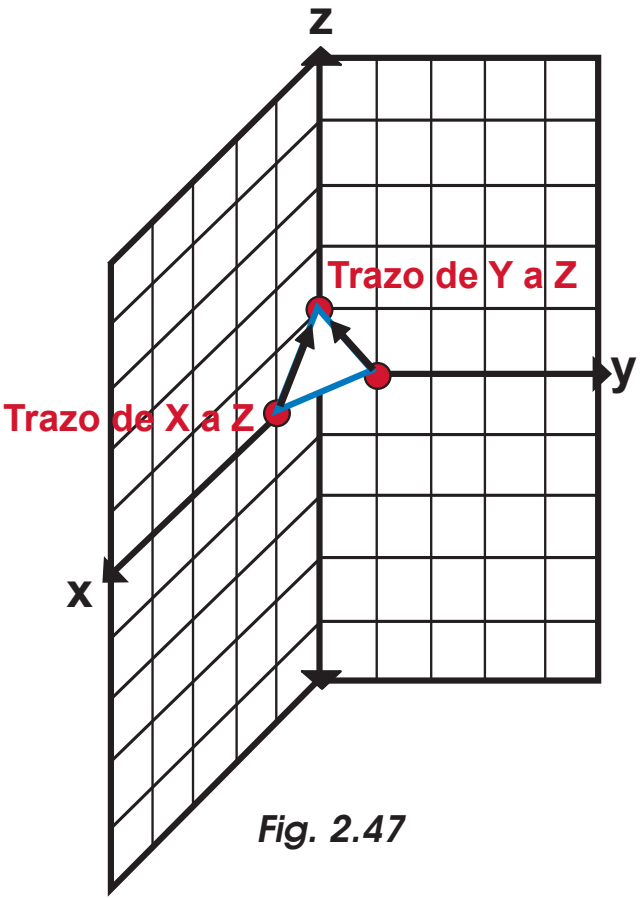



*Fig. 2.47*

**Paso 3.** El punto de intersección de ambas se ubica en la coordenada 1 de **z**. Z=1. El trayecto **z** a **x**, es una pendiente 1/1**.** Esto genera una pendiente de recta de 1 porque 1÷1 es igual a 1, del mismo modo, el trayecto de **z** a **y,** vale 1. La recta **z** a **x** forma la ecuación: **z=1x+1**. La recta **z** a **y** forma la ecuación: **z=1y+1**. La integración de ambas forma la ecuación del plano. Fig. 2.48. **x+y+z=1**.

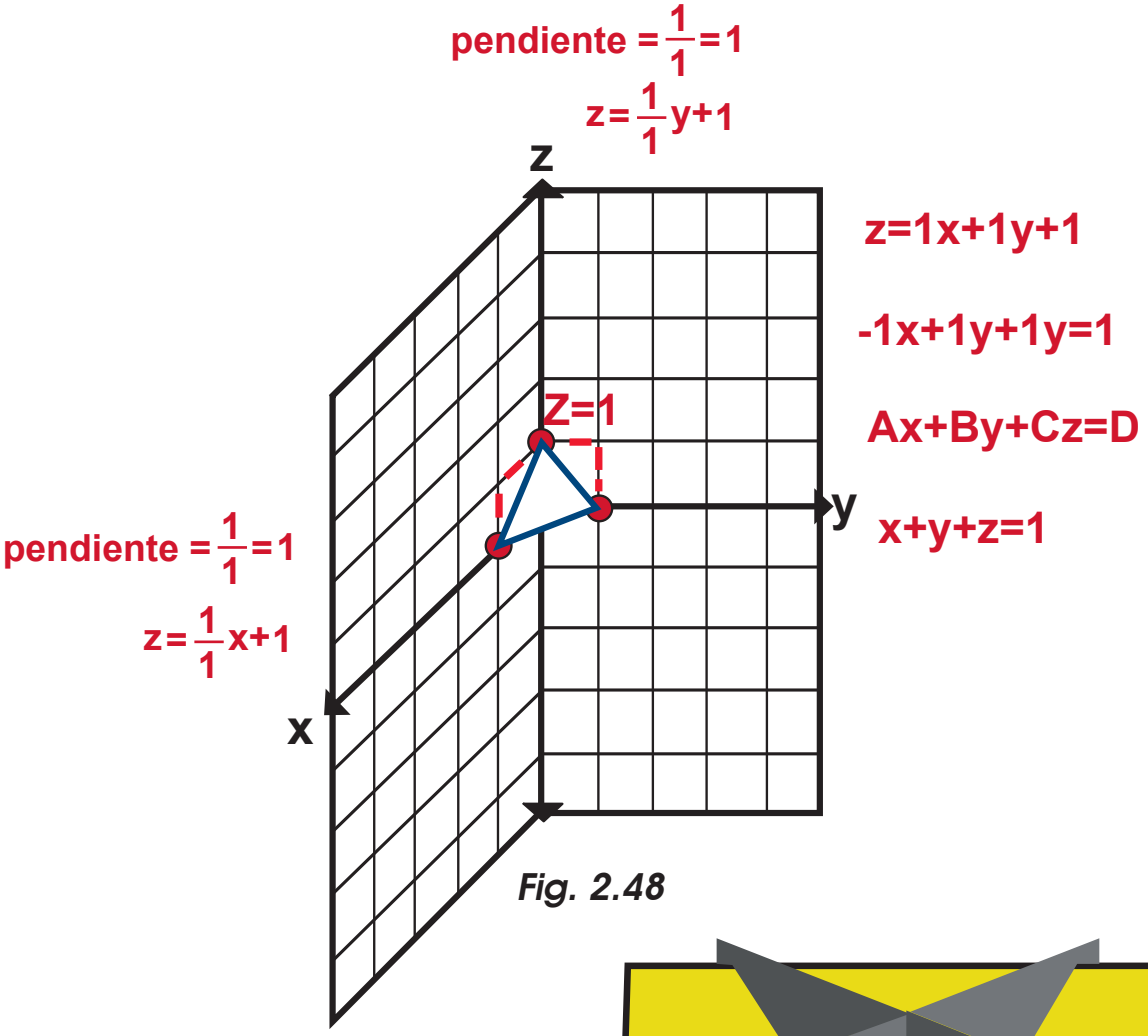



*Fig. 2.48*

Al visualizar la ecuación x+y+z=1 obtenemos un plano como el de ésta imagen.

Recordemos que el plano final siempre se visualizará como aquí, el proceso anterior lo hacemos solamente para llegar a la ecuación. Fig. 2.49. Plano que genera: **x+y+z=1**.

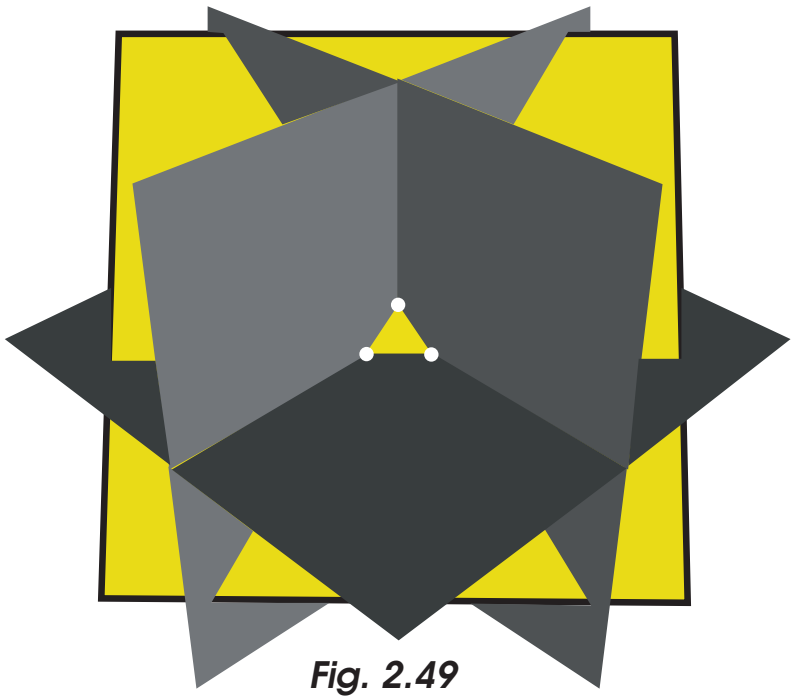


*Fig. 2.49*

Veamos ahora otros ejemplos. Estos ejemplos son un poco más complejos que los anteriores, y nos ayudarán a comprender mejor como obtener la ecuación del plano a partir de su graficación.

**Ejemplo 2:**

**Paso 1.** En un plano cartesiano tridimensional grafiquemos los siguientes puntos:

**Fig. 2.50. A(x=2, y=0, z=5). B(x=0, y=0, z=-3). C(x=0, y=5, z=0). Fig. 2.51. Unión de puntos.**

Podemos colocar los puntos en cualquier otro lugar, ello va a determinar la inclinación del plano y también su ecuación.

**Paso 2.** Determinemos el valor del punto de de intersección de X a Z, y de Y a Z. Fig. 2.52. Punto de intersección Z=-3.

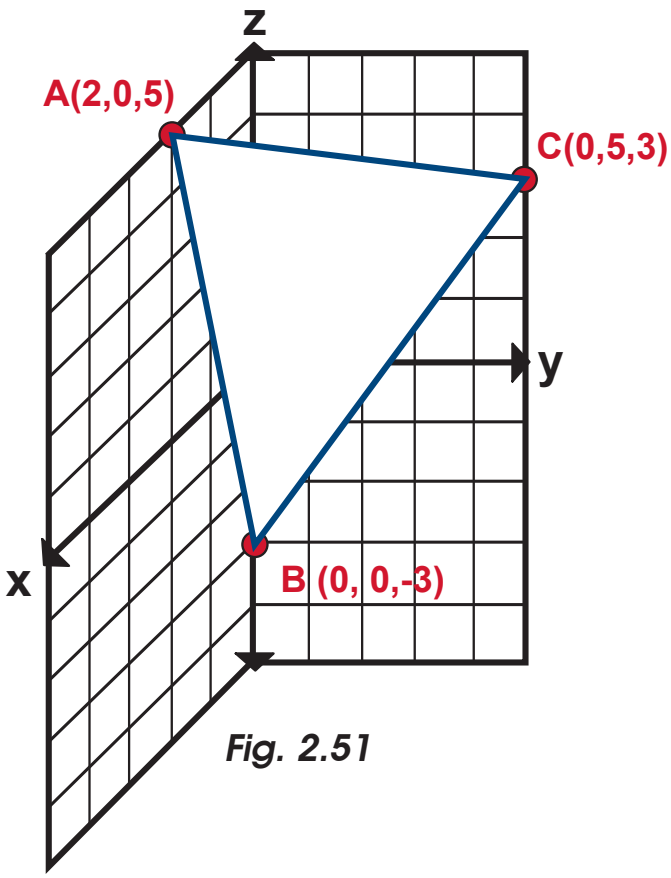



*Fig. 2.51*

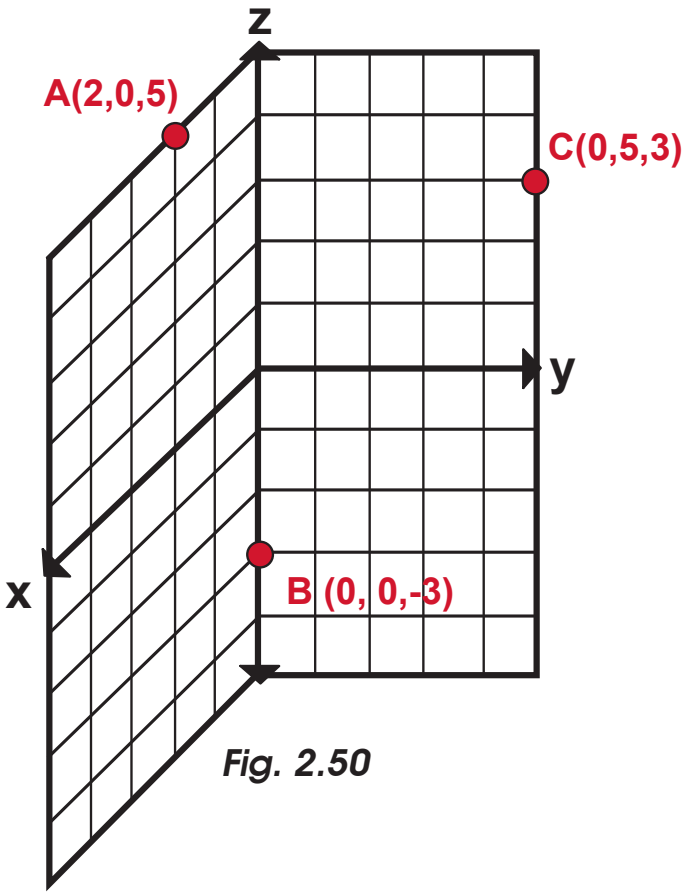



*Fig. 2.50*

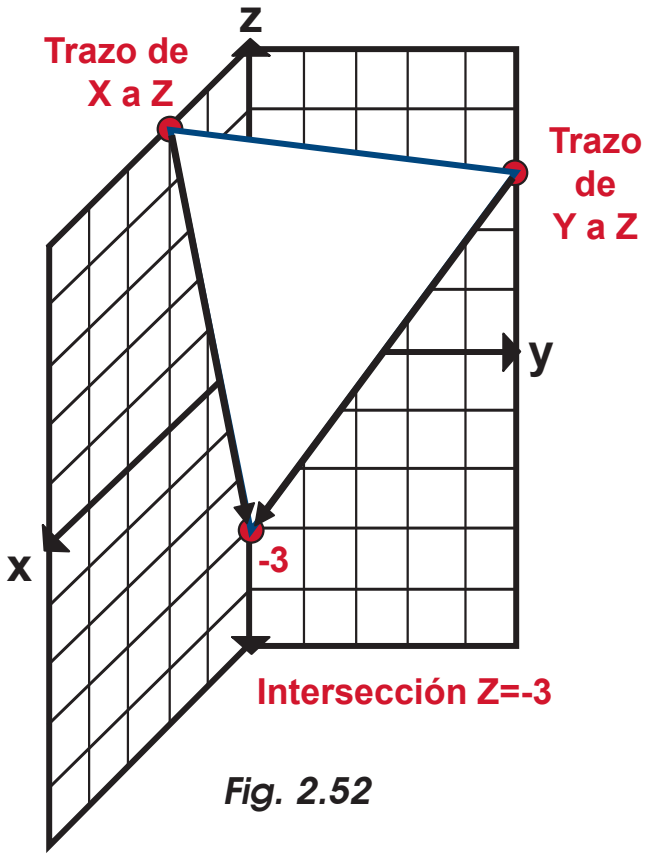



*Fig. 2.52*

**Paso 3.** Partiendo de A(x=2,y=0,z=5) en el trazo de x a z, seguimos una trayectoria de cuatro coordenadas hacia abajo sobre el eje z, y una coordenada a la derecha sobre el eje x. Ello determinará la pendiente.

$$p = \frac{4}{1}$$

Fig. 2.53. Pendiente de recta cuyo valor final es 4.

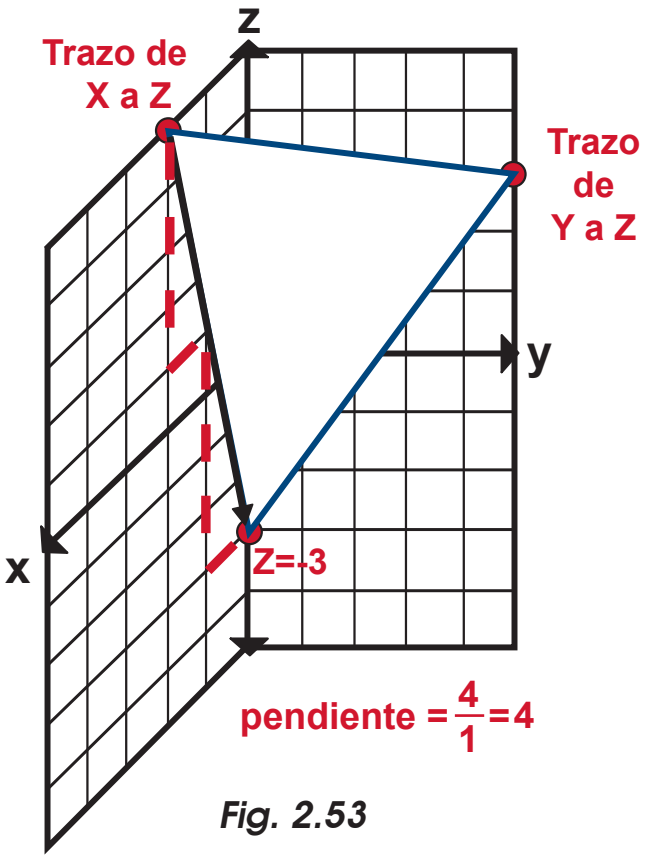



*Fig. 2.53*

**Paso 4.** 4x es la pendiente de la recta del trazo x a z, -3 es la intersección de la recta con el eje z. Fig. 2.54. De esta información obtenemos la ecuación aislada: z = 4x- 3 si partimos ahora de z a x.

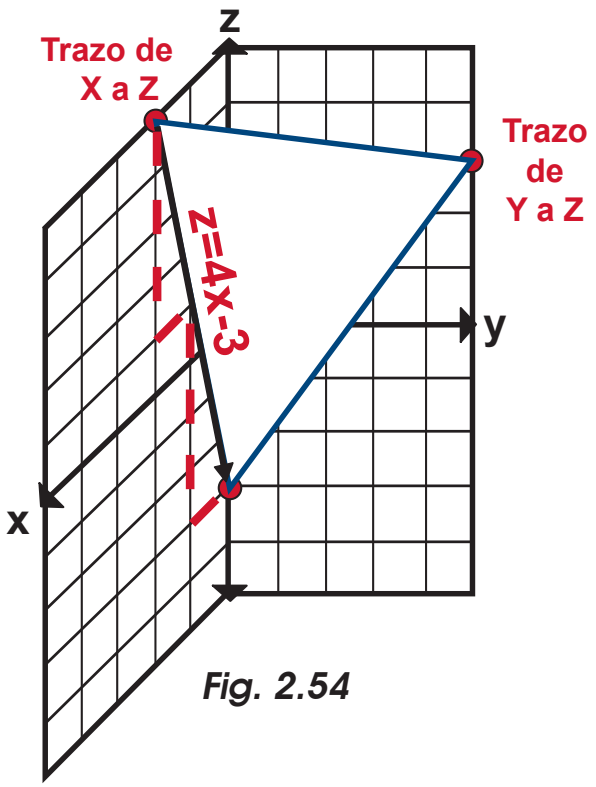



*Fig. 2.54*

Fig. 2.55. Por otra parte de y a z, tenemos la pendiente:

$$p = \frac{6}{5}$$

Haciendo el proceso anterior obtenemos la ecuación:

$$z = +\frac{6}{5}y - 3$$

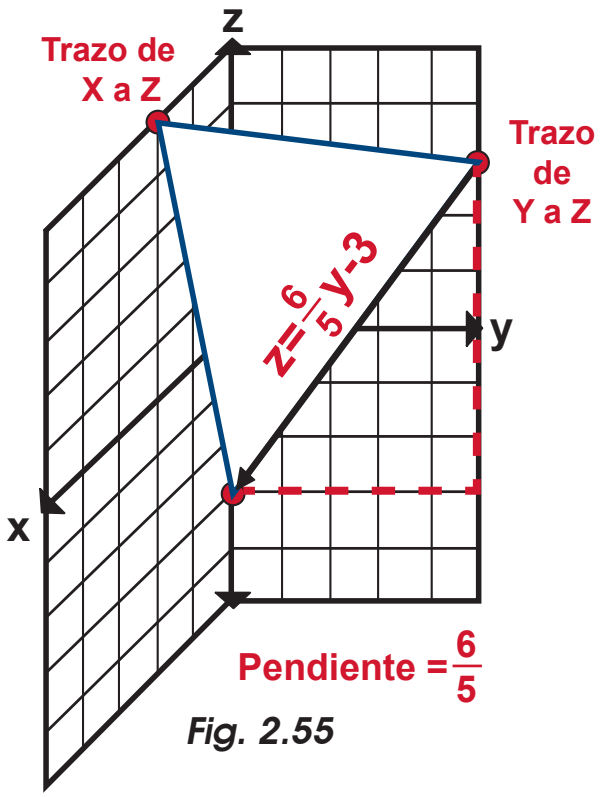



*Fig. 2.55*

**Paso 5.** Una vez que hemos definido las ecuaciones de las dos rectas principales de éste plano, el siguiente paso es fusionarlas para generar una tercera ecuación. Esta ecuación resume ambas.

Así tenemos: z = 4x - 3 y también:

$z = +\frac{6}{5}y - 3$ Que nos da:

$z = 4x + \frac{6}{5}y - 3$

**Paso 6.** La ecuación anterior todavía no tiene la forma de la ecuación del plano **Ax+By+Cz=D**. Lo que debemos hacer es continuar despejando hasta obtenerla.

**Paso 7.** Sigamos desarrollando la ecuación:

$$z = 4x + \frac{6}{5}y - 3$$

para buscar la forma: x+y+z=D. Ahora, vamos a despejar y convertir los números positivos en negativos. 4x se convertirá en

-4x, y $\frac{6}{5}y$ será ahora: $-\frac{6}{5}y$

Z en este caso pasa del lado derecho de la ecuación como en la ecuación:

$$-4x - \frac{6}{5}y + z = -3$$

**Paso 8.** Ya nos estamos acercando a la forma final de la ecuación del plano.

Para terminarla debemos multiplicar cada parte de la ecuación por nuestro denominador común. El denominador común en este caso es: 5. Recordemos que la fracción de nuestra ecuación es:

$$\frac{6}{5}$$

Tomemos el denominador común de ésta fracción y con él, multipliquemos cada parte de la ecuación de la siguiente manera.

Partiendo de:

$$-4x - \frac{6}{5}y + z = -3$$

Conformamos la ecuación:

$$5\ (-4x)\ 5\left(-\frac{6}{5}y\right)5\ (z) = -3$$

Luego multiplicamos:

$5\ (-4x) = -20x$ $5 \times -4 = -20$

Recordemos esa regla:

Menos por más = menos y

Más por menos = menos.

$5\left(-\frac{6}{5}y\right) = 6y$ $5 \times -6 = -30$,

-30 entre 5 = -6 o bien: $-\frac{30}{5} = -6y$

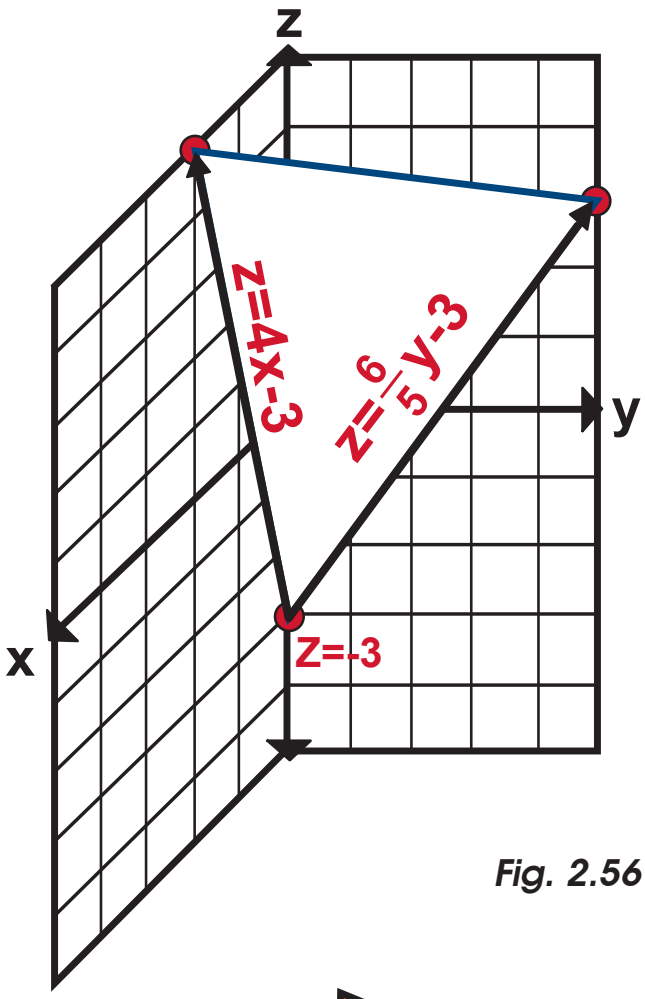
y finalmente:

$$5\ (z) = 5z$$

Conjuntando los resultados obtenemos nuestra ecuación del plano.

Fig. 2.56. -20x-6y+5z=-15.

Fig. 2.57. Visualización del plano inclinado de la ecuación: -20x-6y+5z=-15.




$$z = 4x - 3$$

$$z = +\frac{6}{5}y - 3$$

$$z = 4x + \frac{6}{5}y - 3$$

$$-4x - \frac{6}{5}y + z = -3$$

$$5\,(-4x)\;5\left(-\frac{6}{5}y\right)5\,(z) = -3$$

$$-20x - 6y + 5z = -15$$

$$Ax + By + Cz = D$$

*Fig. 2.56*

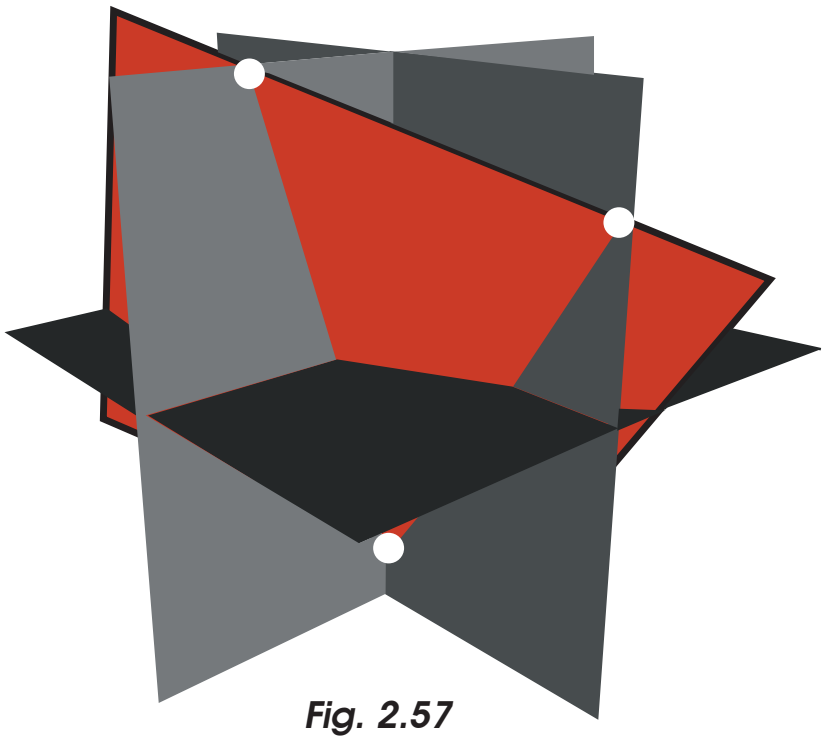


*Fig. 2.57*

**Ejemplo 3:**

**Paso 1.** En un plano cartesiano tridimensional grafiquemos los siguientes puntos:

**Fig. 2.58. A(x=5, y=0, z=4). B(x=0, y=0, z=2). C(x=0, y=3, z=-5).**

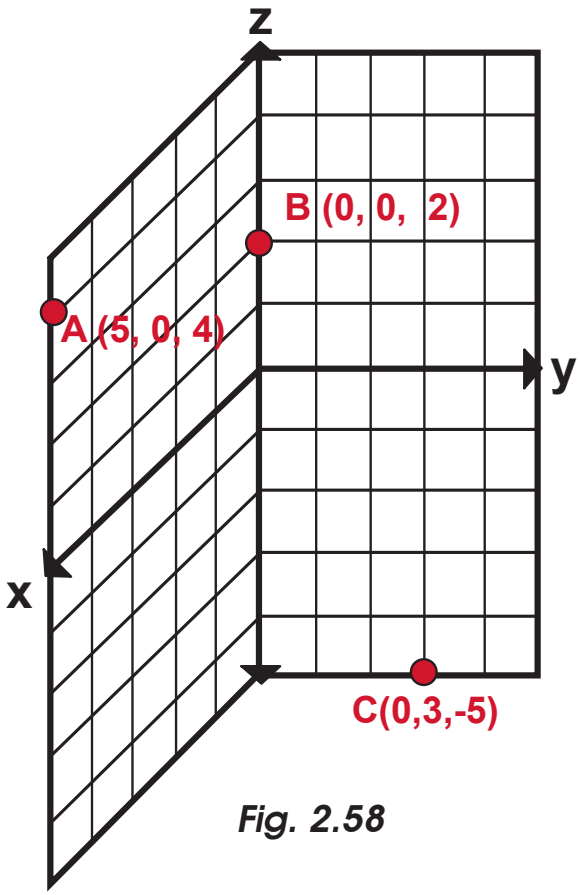



*Fig. 2.58*

**Paso 2.** Igual que en los dos ejemplos anteriores vamos a unir los puntos de la gráfica anterior. De esa unión, vamos a obtener también un plano. Fig. 2.59. Unión de puntos.

Ahora en base a él, determinemos su ecuación, como lo hemos hecho hasta ahora.

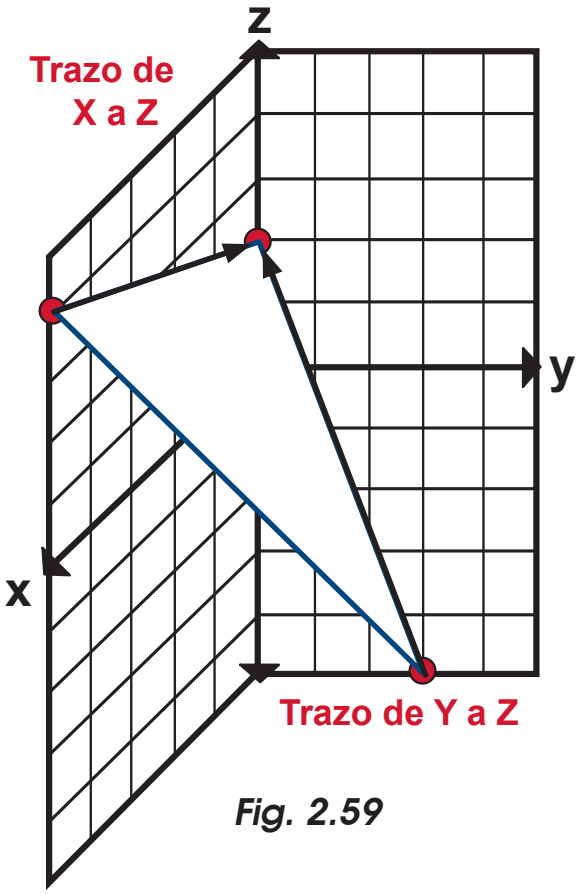



*Fig. 2.59*

**Paso 3.** El punto donde ambos trazos se unen es Z=2. Fig. 2.60. Z=2

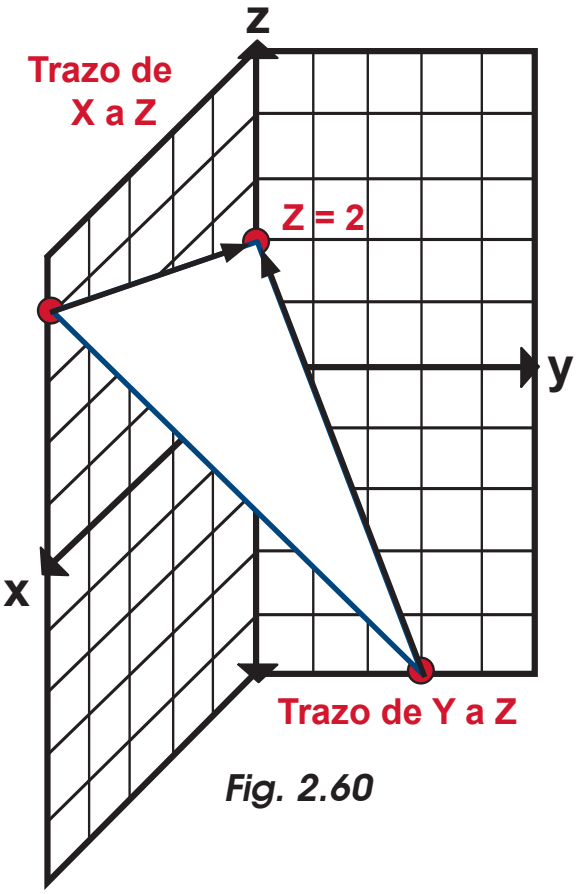



*Fig. 2.60*

**Paso 4.** Partiendo de Z=2 en el trazo de z a x, seguimos una trayectoria de cinco coordenadas hacia la izquierda sobre el eje z, y dos coordenadas hacia arriba sobre el eje x, que determina las pendiente:

$$p = \frac{2}{5}$$

Del mismo modo partiendo de Z=2 en el trazo de z a y, seguimos una trayectoria de 3 coordenadas hacia la derecha sobre el eje y, y 7 coordenadas hacia abajo sobre el eje z, para obtener la pendiente. Fig. 2.61. Pendientes del plano.

$$p = -\frac{-7}{3}$$

De las que obtenemos las 2 ecuaciones de la recta:

$$z=\frac{2}{5}x+2 \qquad z=-\frac{-7}{3}y+2$$

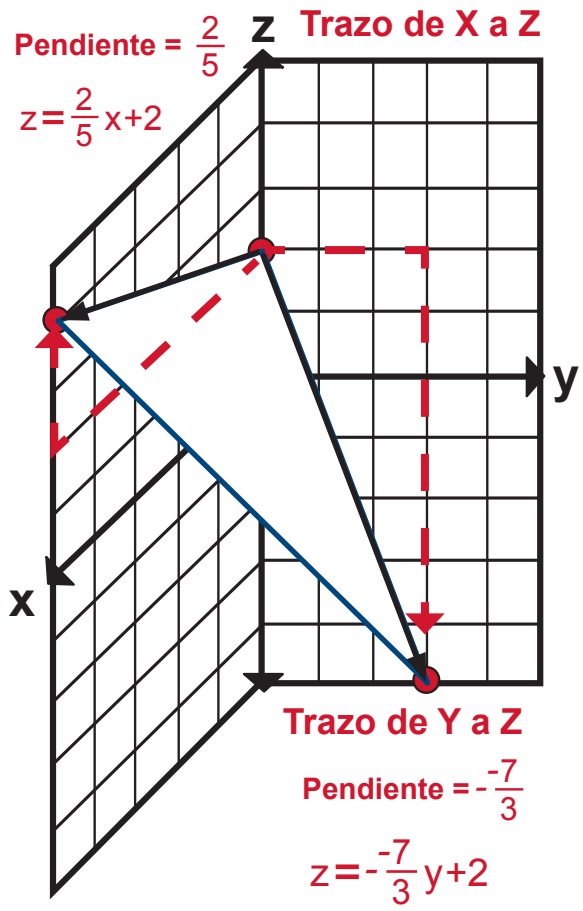



*Fig. 2.61*

Al fusionar ambas obtenemos:

$$z=\frac{2}{5}x-\frac{-7}{3}y+2$$

**Paso 5.** La ecuación: $z=\frac{2}{5}x-\frac{-7}{3}y+2$

No tiene todavía la forma de la ecuación del plano, para obtener esa forma debemos continuar despejándola, convirtiendo las fracciones en sus valores negativo y positivo. Por lo tanto:

$$\frac{2}{5}x \text{ será } -\frac{2}{5}x \quad y$$

$$-\frac{-7}{3}y \text{ cambiará a: } \frac{7}{3}y$$

Z en este caso, se mueve también al lado derecho de la ecuación para obtener:

Fig. 2.62 $-\frac{2}{5}x+\frac{7}{3}y+z=2$

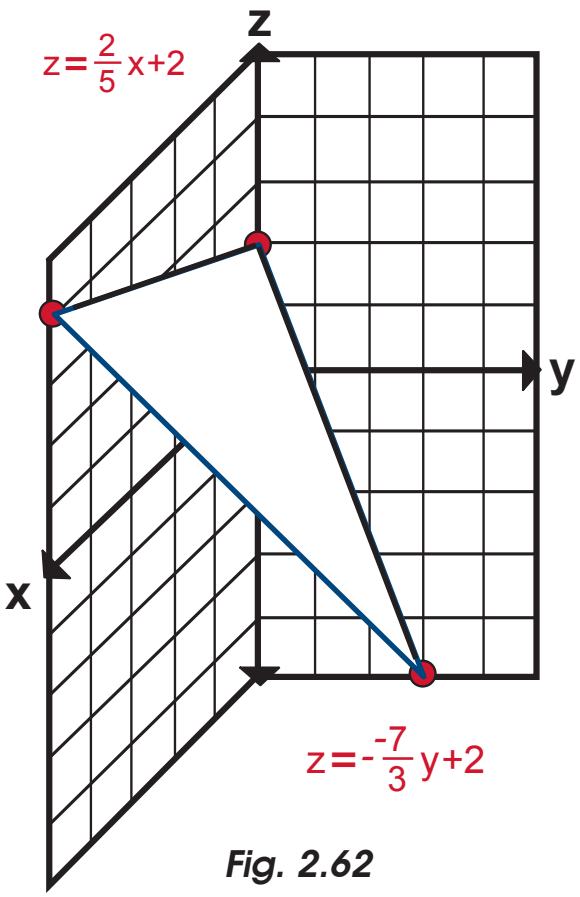



*Fig. 2.62*

**Paso 6.** Nos estamos acercando de nuevo a la forma final de nuestra ecuación del plano. Para terminarla debemos multiplicar cada parte de la ecuación por nuestro denominador común. El denominador común en este caso, se obtiene de multiplicar ambos denominadores de la ecuación, en este caso: 5x3 = 15. Nuestro denominador común es 15. Con este número, multipliquemos cada parte de la ecuación de la siguiente manera:

$$15\left(-\frac{2}{5}x\right)15\left(\frac{7}{3}y\right)15(2)=z$$

Aquí lo que hicimos en la parte final de la ecuación fue multiplicar el 2 por el 15 para continuar despejando.

Otra manera de multiplicar esta ecuación utilizando lógica-matemática, sería colocar un 1 como denominador para trabajar con fracciones. El resultado, es exactamente el mismo:

$$\frac{15}{1}\left(-\frac{2}{5}x\right)\frac{15}{1}\left(\frac{7}{3}y\right)15(2)=z$$

Así obtenemos:

$$-\frac{30}{5}x+\frac{105}{3}y+15z = 30$$

30 entre 5 = 6

105 entre 3 = 35

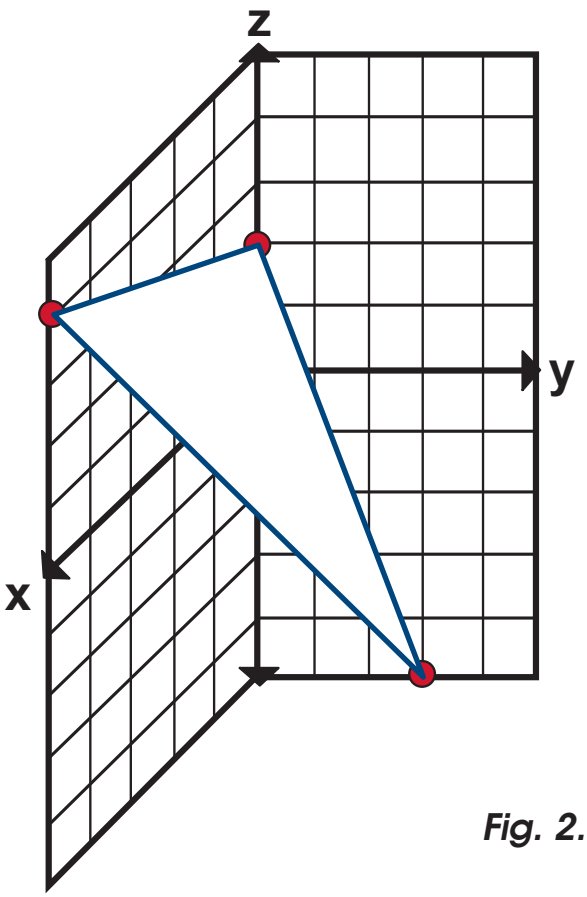
Con esta información conformamos la ecuación del plano:

Fig. 2.63. -6x+35y+15z=30.

Ya hemos terminado de desarrollar los ejemplos, estos ejemplos son sólo una aproximación al conocimiento de la ecuación del plano. Sin embargo antes de finalizar este apartado haré un último comentario respecto a los ejemplos. En ellos, solo visualizamos una parte del plano cartesiano 3D, si nosotros cambiamos nuestro punto de vista de ese objeto, veremos los otros cuadrantes, por lo tanto, tendremos una vista distinta del plano inclinado y su plano cartesiano, en este caso considero que ya no tiene caso indagar más en ello, mejor continuemos con el tema que sigue.

Hay que recordar que el interés de éste proyecto de investigación, es que el diseñador comprenda éstos conceptos para que después los aplique en el desarrollo de sus proyectos interactivos, como los que hemos desarrollado hasta ahora, y como el que presentaré en el siguiente apartado.

Espero que estos ejemplos sean suficientes para comprender la lógica de la ecuación del plano y que los diseñadores desarrollen otros ejemplos similares para la comprensión de conceptos matemáticos aplicados a proyectos de diseño multimedia.  Fig. 2.63. -6x+35y+15z=30 y Fig. 2.64. Visualización del plano inclinado que genera la ecuación: -6x+35y+15z=30




$$z=\frac{2}{5}x+2$$

$$z=-\frac{7}{3}y+2$$

$$z=\frac{2}{5}x-\frac{7}{3}y+2$$

$$-\frac{2}{5}x+\frac{7}{3}y+z=2$$

$$15\left(-\frac{2}{5}x\right)15\left(\frac{7}{3}y\right)15(2)=z$$

$$-\frac{30}{5}x+\frac{105}{3}y+15z=30$$

$$-6x+35y+15z=30$$

$$Ax+By+Cz=D$$

*Fig. 2.63*

*Fig. 2.64*

### 2.3.1. Interactivo ecuación del plano.

Para elaborar el último interactivo de nuestro proyecto, utilizaremos el software Adobe Flash CS6 con el conocido lenguaje de programación: Action Script 2.0, pero para quienes no conocen ésta famosa aplicación, diremos lo siguiente: Flash CS6 es un poderoso software, o aplicación, útil para desarrollar impactantes trabajos multimedia tales como: animación interactiva, aplicaciones portátiles o web, y videojuegos.

Asi también, las posibilidades para producir trabajos interactivos en Flash, se incrementan de manera exponencial, cuando conocemos el lenguaje que éste utiliza: Action Script, un poderoso código de programación orientado a objetos.

La siguiente pregunta será: ¿Porque se utilizó esta aplicación y este código, y no otro? La respuesta es, porque yo como diseñador de aplicaciones interactivas básicas, estoy de alguna manera un poco más familiarizado con éste código, que con otros, aunque debo ser honesto y decir que aún me falta mucho por aprender de éste.

Ahora bien, ¿este interactivo se puede desarrollar en otros lenguajes o plataformas? La respuesta es, definitivamente: Si. Actualmente existen muchas plataformas de trabajo para desarrollar un interactivo como éste, sin embargo, cada una de éstas, requiere un tiempo de estudio, dedicación y practica para su uso e implementación en proyectos interactivos.

Por lo anterior, mencionaré algunas otras plataformas que podrian haberse utilizado para la elaboración de éste interactivo: three.js que es una aplicación JavaScript para 3D con la cual podemos dibujar literalmente con código: objetos, vistas, materiales etc, este se lleva de maravilla con HTML5.

Otra plataforma o código que se puede utilizar para desarrollar una aplicación como ésta, es CSS3, que es la versión 3 del lenguaje utilizado para las hojas de estilo en cascada para definir estilos en; contenidos, formatos, textos y ahora, hasta figuras, éste código también se lleva de maravilla con HTML5.

Sin embargo, al final me decidí por Flash Action Script, por las posibilidades que brinda para desarrollar interactivos didácticos con matemáticas, creo que en lo personal me ha impactado el trabajo de la Doctora Barbara Kaskosz quien ha desarrollado complejos e impresionantes interactivos en el sitio web: www.flashandmath.com, y cuyo trabajo, me ha motivado a seguir utilizando la plataforma Flash. Por ello, en el último capitulo de ésta tesis titulado: Las matemáticas aplicadas al diseño interactivo, se entenderá porque es tan importante esta plataforma de trabajo, y las posibilidades que nos brinda, además, mostraré algunos ejemplos de interactivos didácticos con matemáticas publicados en este sitio web. Dicho lo anterior, hablemos del interactivo. Fig. 2.65. Interactivo ecuación del plano.

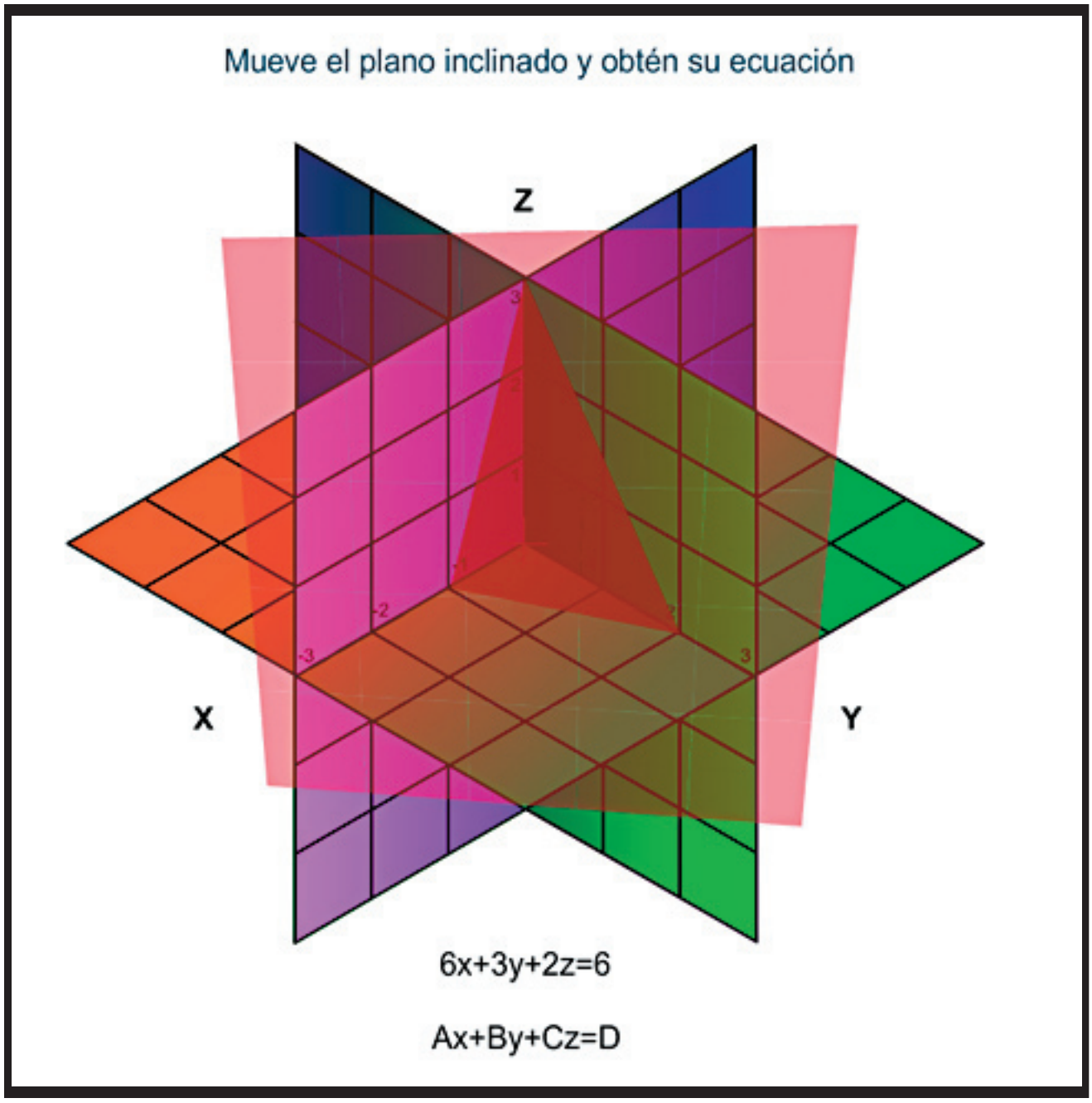



*Fig. 2.65*

Para obtener la ecuación en este interactivo, basta que el usuario mueva el plano con el ratón. Éste representa un plano inclinado (colocado o incrustado) en un plano cartesiano tridimensional en vista isométrica. cuya posición inicial es: x+y+z=1. ¿Como se realizó este interactivo?. Éste interactivo a grandes rasgos se elaboró utilizando Adobe Flash CS6 con el código de programación; Action Script 2.0.

Action Script, es un producto para el desarrollo de trabajos multimedia que requieren animación, desarrollo de vectores, video juegos y aplicaciones web o portátiles. Puede ser visto en acción mediante Adobe Flash Player, un software desarrollado para visualizar multimedia, ejecutar aplicaciones, así como transmisión de video y audio.

Mediante el uso del lenguaje ActionScript 2.0 utilizado en la programación orientada a objetos, se desarrolló la aplicación del plano 3D. Un interactivo que para funcionar, requiere de clases y operaciones.

Cuando un programador desarrolla una aplicación de cómputo, éste debe ligar los datos o los contenidos con ciertas operaciones. En este caso, la programación orientada a objetos, nos aporta entidades compuestas de datos. Para poder entender esta parte, llamemos a esas entidades: objetos, por un lado tenemos los objetos y por otro, las clases. Las clases es todo aquello que hace funcionar a los objetos, y lo podemos definir como un conjunto de operaciones o métodos que se aplican a nuestros objetos.

Si aún no queda clara esta idea, usemos una metáfora, utilicemos como ejemplo, un objeto de la vida real. Todos los objetos, tienen ciertos atributos (propiedades) como su forma, color o tamaño, pero también tienen una función (un método), ésta función o método nos dice para que sirve ese objeto, una utilidad que al final, determina su razón de ser. Una impresora por ejemplo, en dicho objeto, los atributos son: tiene la forma de un prisma rectangular, es color gris, tiene papel en la bandeja etc. Por otra parte, la impresora tiene una función principal: imprimir y también otras funciones o métodos como apagar, prender, indicar niveles tinta etc. Esto es precisamente el ejemplo de un objeto; es un elemento (que puede ser considerado, una variable) misma que nosotros programamos, asignándole propiedades y métodos, a éstos métodos se les llama Clases.

Ahora bien, tenemos tres archivos principales en nuestra aplicación, por una parte tenemos el archivo plano.swf, éste archivo es la película final y surge de *plano.fla, plano.fla* contiene básicamente nuestros objetos: el plano cartesiano en isometrico, los botones con instancias y los estados o etiquetas de cada triángulo (los triángulos van a indicar la posición e inclinación del plano), etc. Básicamente plano.fla, integra todos nuestros objetos, y no contiene código fuente, solamente movieclips con instancias vinculadas a clases. Por otra parte, tenemos el archivo NavegacionMagazine.as, este archivo contiene todas las clases o funciones que cargan la animación de nuestro plano, éstas clases o funciones las va a tomar de otras bibliotecas de código, los nombres de éstas bibliotecas son: greensock, papervision 3D, y Caurina. Más adelante hablaré de cada una de ellas, ésto sólo es una visión general de como esta construida la aplicación, pues todo, esta integrado en una sola carpeta. Ahora hablemos nuevamente de nuestro archivo: *plano.fla,* este tiene en el escenario un MovieClip llamado navMagazine, navMagazine se encuentra en la librería y está linkeado a una clase. El nombre de ésta clase es: NavegacionMagazine.as y como lo mencioné en el párrafo anterior, éste archivo va a tomar elementos de otras librerias, una de ellas es: Papervision 3D, que podemos definir como un grupo de clases que nos ayudarán a crear un Plano 3D debido a su renderizado en tiempo real. Papervision, al igual que otros programas 3D, maneja conceptos 3D, y utiliza términos como Viewport, Scene, Camera etc, etc. Para entender como participa Papervision, debemos entender primero éstos conceptos, ello nos ayudará a comprender un poco más, el código fuente de nuestra aplicación.

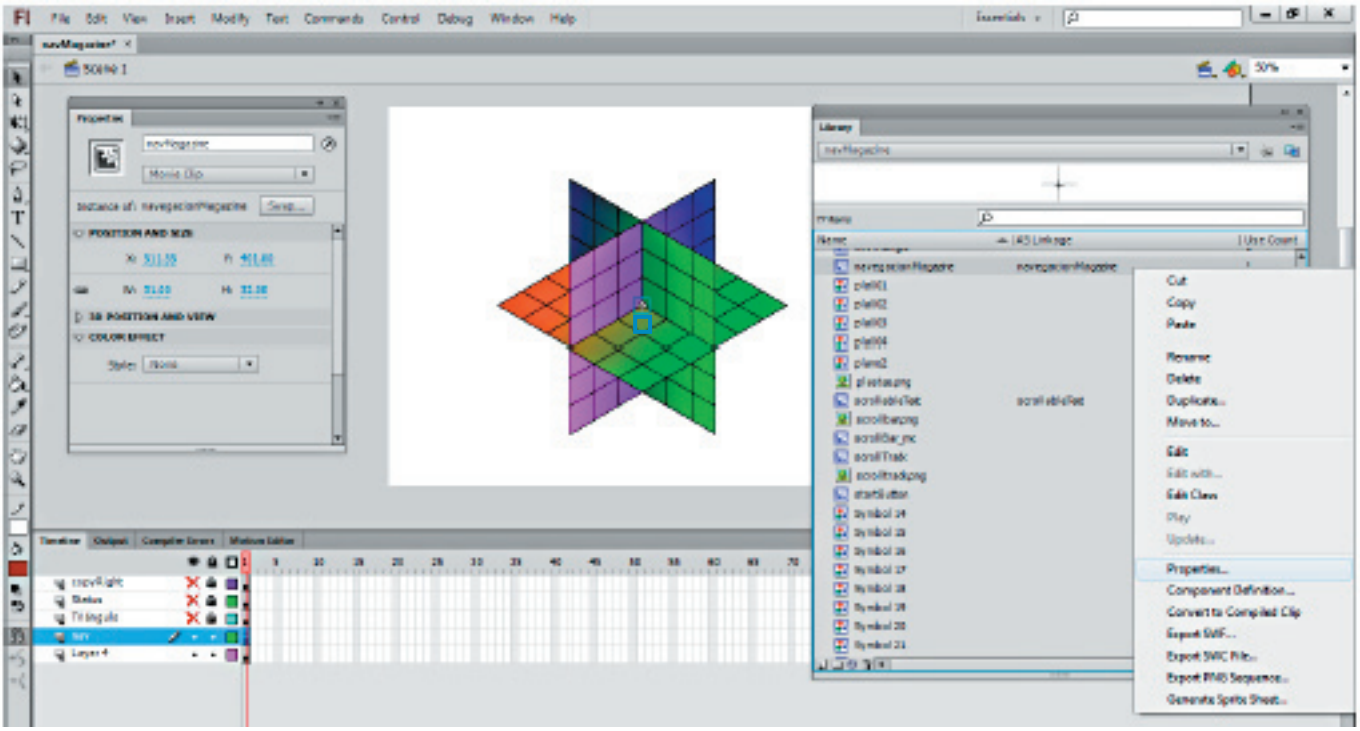


*Fig. 2.66*

Fig. 2.66. En la imagen se muestra seleccionado en el centro del plano cartesiano isométrico, el MovieClip *navMagazine* y también se muestra en la librería el mismo MovieClip, si dámos click al botón derecho, podemos observar el menú de opciones que se pueden realizar con *navMagazine*. Y si damos click a "Properties" observaremos que tiene un hipervínculo a la clase *NavegacionMagazine.as,* el archivo intermediario con las otras bibliotecas de código; greensock, courina y papervision.

Ya sabemos qué es Action Script, (el lenguaje de programación que se utiliza en Flash), ahora expliquemos nuevamente que es Papervisión: Papervisión es una aplicación que nos sirve para generar ambientes 3D en Flash, una biblioteca de código que genera recursos 3D como luces, materiales, cámaras, escenas etc, pero antes de entrar de lleno al análisis del código fuente utilizado para la producción del interactivo del plano, me gustaria explicar a manera de breve glosario, algunos de los conceptos utilizados en la aplicación, pienso que con este antecedente podremos comprender mejor el código de nuestro interactivo, cuya explicación y producción, denota cierta complejidad.

**Viewport:** es básicamente el área visualizable o renderizable de la aplicación, en este caso seria como un rectángulo bidimensional con un alto y un ancho de pantalla donde se va a proyectar la escena tridimensional.

**Scene:** se refiere a la escena tridimensional que vamos a mostrar y sobre la cual vamos a trabajar, es el lugar donde se encuentran por decirlo de alguna manera, los objetos 3D y contiene todo el entorno tridimensional, en ella podemos añadir y quitar objetos, moverlos, transformarlos o ver la escena desde otro punto de vista.

**Luces (LightObject3D, PointLight3D)**
Para iluminar el entorno tridimensional tenemos varias opciones como son las luces direccionales y las luces focales o de punto.

**DisplayObject3D:** Representa una instancia de cualquier objeto que se encuentra en la escena tridimensional. Todos los objetos 3D que vemos en una escena descienden de esta clase.

**Camera:** Sirve para indicar un punto de vista desde el cual se puede visualizar la escena. Existen diferentes tipos: cámara libre, cámara objetivo (siempre apunta a un objeto 3D).

**Materiales:** Para que los objetos tridimensionales se vean, estos tienen que estar dotados de un material. Es el que tiene los datos sobre la apariencia de los objetos. Dice de que color son y como éstos se renderizan. Existen materiales de muchos tipos. De hecho no es raro ver que un mismo objeto contenga varios materiales diferentes para cada una de sus caras. Los tipos de materiales que podemos generar son: ColorMaterial, BitmapMaterial, MovieMaterial, VideoStreamMaterial.

**Motor de Render:** Lo que hace que todo funcione y se vea. Su función básica es traducir todos los datos tridimensionales que hemos generado en la estructura de la escena al visor bidimensional. En este caso en nuestro interactivo se usará otro grupo de motores de física o bibliotecas de código que elevarán el nivel de interacción y animación a nuestra aplicación: Greensock y Courina, otro tipo de clases usadas para animaciones y transiciones vía ActionScript. En cuanto a Greensock y Courina también diremos que son bibliotecas de código complementarias a este lenguaje.

Por todo lo expuesto con anterioridad y con esta información como terminología introductoria, podemos continuar con la explicación del código de nuestro interactivo, espero que el diseñador multimedia entienda ésta información de manera adecuada y comprenda que existen muchas maneras de hacer éstas aplicaciones interactivas.

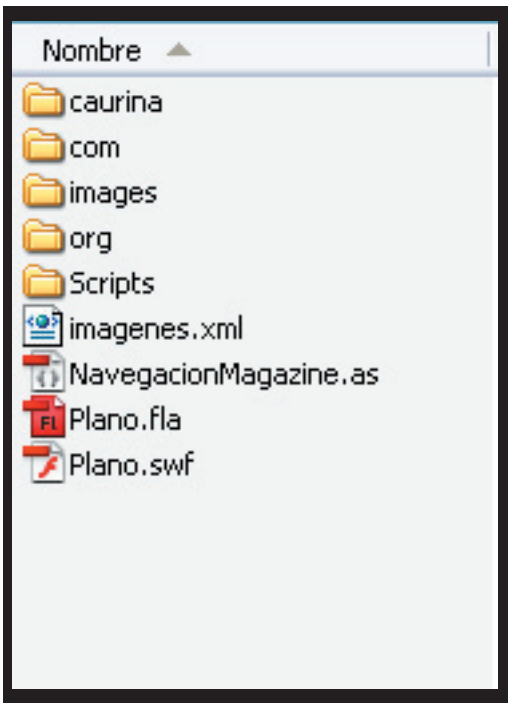



***Fig. 2.67***

Fig. 2.67. Esta imagen muestra la carpeta de contenidos del interactivo ecuación del plano elaborado en Flash CS6. Los archivos que contiene son: Plano.fla, Plano.swf NavegacionMagazine.as, con el código fuente de la aplicación, además de todas las bibliotecas de código que se ocuparon para desarrollar la aplicación. Caurina y Greensock nos van a servir para la animación (con las físicas de movimiento del plano). Dentro de com tenemos la carpeta Greensock, y en org Papervision 3D, las cuales fueron descargadas de sus respectivos sitios web. Además tenemos imagenes.xml y la carpeta images con la foto o material para el plano.

### El Código fuente de nuestro interactivo: plano.swf

Aunque plano.swf nace del archivo plano.fla, el archivo *NavegacionMagazine.as* es el que carga construye, anima y renderiza el plano 3D para poder moverlo y visualizarlo. Veamos el código:

```
import caurina.transitions.Tweener;
import mx.utils.Delegate;
import flash.display.BitmapData;
import org.papervision3d.cameras.*;
import org.papervision3d.materials.BitmapAssetMaterial;
import org.papervision3d.materials.*;
import org.papervision3d.objects.Plane;
import org.papervision3d.scenes.*;
import org.papervision3d.core.proto.DisplayObject3D;
import flash.geom.Rectangle;
import flash.geom.Matrix;
import flash.geom.ColorTransform;
import flash.events.MouseEvent;
import com.greensock.*;

class NavegacionMagazine extends MovieClip{
        var scene:Scene3D;
        private var camera:Camera3D;
        private var container:MovieClip;
        private var group:DisplayObject3D;
        private var objectArray:Array;
        private var theimages:Array;
        private var target:DisplayObject3D;
        private var numFotos:Number;
        private var numFotosLoaded:Number = 0;
        private var PI:Number = 3.1415926;
        private var thexml:XML;

        function NavegacionMagazine(){
                init3D();
                thexml = new XML();
                thexml.ignoreWhite = true;
                thexml.onLoad = Delegate.create(this, xmlLoaded);
                thexml.load("imagenes.xml");
      }
        function init3D()  {
                container = this.createEmptyMovieClip("container", this.getNextHighestDepth());
                container._x = 0;
                container._y = 0;
                scene = new MovieScene3D(container);
                camera = new Camera3D();
                camera.z = -1000;
                camera.zoom = 1;
                camera.focus = 500;
        }
```

```
function layoutFotos(){

	for (var i = 0; i < numFotos; i++){
	var foto = theimages[i];
	var bmpdata:BitmapData = new BitmapData(foto._width, foto._height, false);
	bmpdata.draw(foto,new Matrix(),null,null,null,true);
	var bitmapMaterial:BitmapMaterial = new BitmapMaterial(bmpdata, true);
	bitmapMaterial.smooth = true;
	bitmapMaterial.oneSide = false;
	_root.borde.attachBitmap(bmpdata,2);
	var material4 = new ColorMaterial(0xff0000, 80);
	material4.oneSide = false;
	var ESPACIO:Number = 0;
	var ANCHO:Number = 230;
	var razonX:Number = 1;
	var razonY:Number = 1;

			if (foto._width > foto._height){
				razonY = foto._height / foto._width;
			}else{
				razonX = foto._width / foto._height;
			}

var plane:Plane = new Plane(bitmapMaterial, razonX * ANCHO, razonY * ANCHO, 2, 2);

			var j = i % 5;
			var k = Math.floor(i / 5);
			plane.x = (ANCHO + ESPACIO) * j - (ANCHO + ESPACIO) * 2;
			plane.y = (ANCHO + ESPACIO) * k - ANCHO * 2 + ESPACIO;
			scene.push(plane);
			var cont:MovieClip = plane.container;
			cont._alpha = 50;
		}

		camera.x = 1;
		camera.y = 1;
		scene.renderCamera(camera);
		TweenLite.delayedCall(30,comienza,[],this,true);
		function comienza(){
			this.onEnterFrame = loop3D;
			var botonActual;

			for(var i:Number = 1; i<= 29; i++){
				botonActual = _root.triangle["edo_"+i];
				botonActual.num = i;
				botonActual.onRollOver = function(){
					_root.triangle.gotoAndStop("edo_" + this.num);
				}
			}
		}
	}
```

```
        function loop3D(){
                if(-container._xmouse <= -145 || -container._xmouse >= 145){
                }
                else if (-container._ymouse >= 145 || -container._ymouse <= -145) {
                }
                else {
                        camera.x = -container._xmouse * 2;
                        camera.y = -container._ymouse * 2;
                }
                        scene.renderCamera(camera);
                        _root.ecPlano.text = "";
        }

        private function xmlLoaded(success)
        {
                var children:Array = thexml.firstChild.childNodes;
                theimages = new Array();
                numFotos = children.length;
                for (var i = 0; i < numFotos; i++)
                {
                        var foto = attachMovie("foto", "foto" + i, this.getNextHighestDepth());
                        foto._visible = false;
                        foto.loadFoto(children[i].attributes.src);
                        foto.link = children[i].attributes.link;
                }
        }

      private function onFotoLoaded(e)
        {
                theimages.push(e.target);
                //e.target._visible = false;
                numFotosLoaded++;
                //trace("foto loaded");
                if (numFotosLoaded == numFotos)
                {
                        layoutFotos();
                }
        }
}
```

Básicamente este es el código fuente de nuestro interactivo ecuación del plano. El Lenguaje de programación en el que está escrito, es Action Script 2.0.

Una vez que hemos dado un vistazo al código fuente, lo siguiente que haremos será fragmentarlo en partes o bloques, para tratar de comprenderlo, en verdad siempre es complejo tratar de explicar un código o un lenguaje de programación y requiere analizarlo más de una vez para comprenderlo, sigamos adelante con la explicación del código fuente de nuestro último interactivo.

**Bloque 1**

Este bloque esta compuesto de clases o funciones, veamos para que sirve cada una:

import caurina.transitions.Tweener;
Clase usada para crear animaciones y transiciones vía ActionScript. Caurina nos facilitará la tarea de realizar movimientos de objetos o efectos de transiciones de manera muy fácil.

import mx.utils.Delegate;
Es un método estático que permite ejecutar una función en un ámbito específico.

import flash.display.BitmapData;
Esta clase permite trabajar y visualizar los datos en forma de pixeles (importa a flash objetos Bitmap)

import org.papervision3d.cameras.*;
Esta clase crea una especie de cámara apoyándose de papervision y nos permite ver el objeto desde un punto de vista.

import org.papervision3d.materials.BitmapAssetMaterial;
Clase para crear texturas desde un bitmap externo guardado en la libreria o en los documentos de apoyo.

import org.papervision3d.materials.*;
Clase con una lista de materiales para los objetos, en este caso sólo utilizaremos una imagen bitmap.

import org.papervision3d.objects.Plane;
Clase que permite crear y mostrar objetos en forma de rectángulo. Esta clase nos va ayudar a configurar nuestro plano.

import org.papervision3d.scenes.*;
Clase que permite crear una escena, en ésta escena todos los objetos renderizados se unen para conformarla.

import org.papervision3d.core.proto.DisplayObject3D;
Clase que permite representar instancias de objetos 3D que son contenidos en la escena.

import flash.geom.Rectangle;
Clase usada para crear y modificar objetos, en este caso nuestro rectángulo del que se conforma el plano.

import flash.geom.Matrix;
Clase que representa una transformación de la matriz, ésta define la manera en que vamos a determinar sus puntos o configuración, coordenada a coordenada.

import flash.geom.ColorTransform;
Clase que permite ajustar los valores de color en el objeto display

import flash.events.MouseEvent;
Con esta clase, ocurre un evento cada vez que el usuario mueve el ratón

import com.greensock.*;
Clases para animación de objetos con greensock. Greensock es una libreria para animación compatible con action script.

**Bloque 2.**

class NavegacionMagazine extends MovieClip{

Recordemos el clip de película llamado: NavMagazine. Esta línea de código nos está diciendo que por medio de una instancia, está vinculado el movie clip del archivo flash, a NavegacionMagazine.as

var scene:Scene3D;

Se declara la variable escena con la instancia: Escena 3D.

private var camera:Camera3D;

Se declara una variable privada llamada: cámara, con una instancia que ejecuta la clase Camara 3D. A una variable privada, sólo podemos acceder por medio del script o código a diferencia de una pública, en la cual, el usuario accede libremente.

private var container:MovieClip;

Se declara la variable privada llamada: container, que tiene calidad de MovieClip.

private var group:DisplayObject3D;

Se declara la variable privada: group, que ejecuta la clase DisplayObject3D

private var objectArray:Array;

Declaramos la variable privada: objectArray, la cual es una variable de tipo Arreglo.
Nos dice que se creará una matriz o colección de objetos.

private var theimages:Array;

Se declara la variable privada: theimages, otra variable de tipo Arreglo.
Una matriz o colección de imagenes.

private var target:DisplayObject3D;

Declaramos la variable privada target o destino, que ejecuta la visualización del objeto 3D.

private var numFotos:Number;

Declaramos la variable privada: numFotos, que es de tipo númerica y trabaja con imágenes

private var numFotosLoaded:Number = 0;

Declaramos la variable privada: numFotosLoaded, que es de tipo númerica y su valor es 0

private var PI:Number = 3.1415926;

Declaramos la variable privada: PI, que también es de tipo númerica con valor igual a 3.1415926
Pi es una constante matemática cuyo valor es igual a la proporción existente entre la longitud del perímetro del círculo y la longitud de su diámetro (esta variable nos va a ayudar con el movimiento del plano).

private var thexml:XML;

Declaramos la variable privada: thexml, la cuál es un formato XML. Ésta variable, va a generar una estructura estrictamente jerárquica de nuestros elementos, éstos elementos, van a ser imágenes.

Este bloque básicamente, declara variables privadas y son la base que contiene los elementos que necesitamos para configurar el interactivo. La materia prima en código de nuestra aplicación.

**Bloque 3.**

```
function NavegacionMagazine(){
        init3D();
        thexml = new XML();
        thexml.ignoreWhite = true;
        thexml.onLoad = Delegate.create(this, xmlLoaded);
        thexml.load("imagenes.xml");
}
```

En este bloque de código, se inicia la función init3D. Init3D está vinculado a NavegacionMagazine.as
En init3D se va a cargar la información XML que mencionamos. XML es una estructura de datos que provee de imagenes a cada cuadro del plano, en este caso va a cargar una imagen color rojo a la que se le aplicará un alfa y que al repetirse, generará un plano. Imagenes.xml es un archivo que identifica la direccion url de la foto llamada bg.png.

**Bloque 4.**

```
function init3D()  {
        container = this.createEmptyMovieClip("container", this.getNextHighestDepth());
        container._x = 0;
        container._y = 0;
        scene = new MovieScene3D(container);
        camera = new Camera3D();
        camera.z = -1000;
        camera.zoom = 1;
        camera.focus = 500;
}
```

En este bloque se inicializa una especie de contenedor y se coloca en la profundidad más alta de las posiciones x,y = 0. Así se crea y se centra el MovieClip en la posición 0. De la misma manera en ese contenedor, se crea una escena con un zoom y una cámara con un enfoque en valores numéricos.

**Bloque 5.**

```
function layoutFotos(){
    for (var i = 0; i < numFotos; i++){
```

Iteración para crear un bitmapData y entonces añadir cada nodo del xml en dicho movieclip en forma de imagen.

```
var foto = theimages[i];
```

Declaración de variable foto equivalente al arreglo theimages[i]. Images es una carpeta que contiene una imagen color rojo llamada bg.png, que es la que va a repetirse.

```
var bmpdata:BitmapData = new BitmapData(foto._width, foto._height, false);
```

Declaración de variable bmpdata la cual tiene definidos un ancho y alto relativo a foto o imagen.

```
bmpdata.draw(foto,new Matrix(),null,null,null,true);
```

Dibujar el arreglo o matriz respectivo de la foto o imagen.

```
var bitmapMaterial:BitmapMaterial = new BitmapMaterial(bmpdata, true);
bitmapMaterial.smooth = true;
bitmapMaterial.oneSide = false;
```

Añadir material a bmpdata. En este caso el material es de nuevo mi imagen.

```
_root.borde.attachBitmap(bmpdata,2);
```
Añadir bmpdata al movieclip borde ubicado en la raiz de la escena, nivel de profundidad 2. En otras palabras se va a cargar mi imagen en el escenario.

```
var material4 = new ColorMaterial(0xff0000, 80);
```
Declaración de la variable: material4 equivalante a un nuevo color hexadecimal con 80 grados alpha

```
material4.oneSide = false;
```
Uno de los lados de material4 no se tomará en cuenta, ésto para visualizarlo como queremos.

```
var ESPACIO:Number = 0;
```
Declaración de variable: ESPACIO tipo numérica igual a 0

```
var ANCHO:Number = 230;
```
Declaración de variable: ANCHO tipo numérica igual a 230

```
var razonX:Number = 1;
var razonY:Number = 1;
```
Declaración de variable razonX tipo numérica igual a 1
Declaración de variable razonY tipo numérica igual a 1

```
if (foto._width > foto._height){
razonY = foto._height / foto._width;
}else{
razonX = foto._width / foto._height;
```
Si el ancho de foto es mayor que su altura, entonces la variable razonY, es el cociente de dividir foto._ height/foto._width, si es menor a la altura, entonces razonX es el cociente de dividir foto._width / foto._height. En este bloque se configura el espacio, el ancho y la altura en X y Y.

**Bloque 6.**

```
var plane:Plane = new Plane(bitmapMaterial, razonX * ANCHO, razonY * ANCHO, 2, 2);
```
Se declara una variable llamada plane la cual es un nuevo Plano de la clase Papervision 3D con un ancho igual al producto de razonX por ANCHO, una altura igual al producto de razonY por ANCHO con 2 segmentos de profundidad en lo ancho y lo alto.

```
var j = i % 5;
var k = Math.floor(i / 5);
```
Se declara j equivalente una iteración del valor i=0 sumando 1 cinco veces
Se declara k equivalente al cociente de i/5 redondeando al entero previo más cercano

```
plane.x = (ANCHO + ESPACIO) * j - (ANCHO + ESPACIO) * 2;
plane.y = (ANCHO + ESPACIO) * k - ANCHO * 2 + ESPACIO;
scene.push(plane);
```
El valor del movieclip plane en x es equivalente a: (ANCHO + ESPACIO) * j - (ANCHO + ESPACIO) * 2;
El valor del movieclip plane en y es equivalente a: (ANCHO + ESPACIO) * k - ANCHO * 2 + ESPACIO;
Empujar o introducir las dimensiones de plane a la escena en X y Y.

Otra cosa que aqui está ocurriendo, es que se esta construyendo un plano a partir de un cuadrado más pequeño de 5X5 para generar el plano 3D con un total de 25 cuadros.

**Bloque 7.**

```
var cont:MovieClip = plane.container;
```
Se crea un movieclip contenedor llamado plane.container

```
cont._alpha = 50;
```
Se ajusta una propiedad alpha del 50% de visibilidad al movieClip “cont”

```
camera.x = 1;
camera.y = 1;
```
Valor de X en el movieClip camera; es igual a 1 y Valor de Y en el movieClip camera: es 1

```
scene.renderCamera(camera);
```
Se coloca y renderiza el plano con valores en x,y = 1

```
TweenLite.delayedCall(30,comienza,[],this,true);
function comienza(){
```
Se ejecuta la función, ésta comenzará con un retraso de 30 frames o 3 seg.

```
this.onEnterFrame = loop3D;
```
En cuanto el usuario mueva el ratón sobre el plano, se ejecutará de manera contínua la función loop3D que iniciará la animación del plano.

```
var botonActual;
```
Declaración de botonActual como variable que será usada dentro de la iteración para asignar cada uno de sus valores, en este caso en el documento plano.fla se colocaron 19 botones, cada botón tiene asignado un estado o instancia que carga un triangulo que indica, la posición e inclinación del plano y al mismo tiempo mueve el plano.

```
for(var i:Number = 1; i<= 29; i++){
botonActual = _root.triangle[“edo_”+i];
botonActual.num = i;
```
Iteración de 20 triángulos, para que en cada rollover que se haga sobre el id de triangulo, se proceda a detener el frame en la linea de tiempo relacionada con el “edo_”+i que corresponde a un botón o zona sensible.

```
botonActual.onRollOver = function(){
_root.triangle.gotoAndStop(“edo_” + this.num);
}
```
Al hacer RollOver sobre botonActual se mostrará el frame “edo”+i del movieClip triangle en el escenario. Es decir, se cargará en el escenario el triangulo que corresponde a la zona de botón actual y también a la zona de inclinación del plano.

**Bloque 8.**

```
function loop3D(){
```
Nos permite renderizar el plano siempre y cuando esté dentro del área asignada para ello.

```
if(-container._xmouse <= -145 || -container._xmouse >= 145){
}
```
Si el contendedor en lo horizontal es menor a -145 o mayor a 145, entonces no se realiza renderizado

```
else if (-container._ymouse >= 145 || -container._ymouse <= -145) {
}
```
Si el contendedor en lo vertical es menor a -145 o mayor a 145, entonces no se realiza renderizado

```
else {
camera.x = -container._xmouse * 2;
camera.y = -container._ymouse * 2;
}
```
Si está dentro del área asignada en el contenedor, entonces se realizará el renderizado.

**Bloque 9.**

```
scene.renderCamera(camera);
```
Renderizar escena

```
 private function xmlLoaded(success)
 {
```
Si la información del xml es cargada con éxito, entonces añadir la imágen relacionada con el nodo del xml leído.

```
var children:Array = thexml.firstChild.childNodes;
```
Declaramos la variable children, que equivale a los nodos contenidos en: thexml

```
theimages = new Array();
```
Variable theimages de tipo arreglo. Matriz o colección de imagenes.

```
numFotos = children.length;
for (var i = 0; i < numFotos; i++)
{
```
Leemos el total de los hijos de la estructura xml y lo asignamos a numFotos

```
var foto = attachMovie("foto", "foto" + i, this.getNextHighestDepth());
foto._visible = false;
foto.loadFoto(children[i].attributes.src);
foto.link = children[i].attributes.link;
}
```
Declaramos fotos equivalente al movieClip "foto"+i puesto en la profundidad más alta

```
private function onFotoLoaded(e)
{
        theimages.push(e.target);
        numFotosLoaded++;
        if (numFotosLoaded == numFotos)
        {
        layoutFotos();
        }
    }
}
```
Si todas las imágenes del xml han sido cargadas correctamente, entonces ejecutar la función layoutFotos

Por fin hemos terminado de revisar la aplicación y la carpeta plano que contiene todos los archivos que hacen posible la elaboración de nuestro último interactivo con su código fuente, y con ello, hemos terminado prácticamente nuestro marco teórico y la parte de producción. Por lo tanto, vamos a adentrarnos en el problema de la investigación: Como y en base a que factores, podemos mejorar nuestra producción multimedia, para ello será importante revisar los dos últimos capítulos de ésta investigación: Diseño de interacción y programación en arte y diseño y las matemáticas aplicadas al diseño de interactivos didácticos.

Para mayor referencia de como se construyeron los interactivos, por favor revisen el CD-Rom con los archivos punto.html, linea.html y plano.fla, con sus respectivas bibliotecas de código de las que se apoyan. Éste material les permitirá estudiar más a detalle su construcción y analizar su contenido. Considero que lo anterior brindará al interesado, más información al respecto y disipará las dudas que puedan surgir sobre el tema, y sobre el funcionamiento de las aplicaciones. Continuemos con el tercer capítulo de éste documento.

# III. DISEÑO DE INTERACCIÓN Y PROGRAMACIÓN EN ARTE Y DISEÑO.

# III. DISEÑO DE INTERACCIÓN Y PROGRAMACIÓN EN ARTE Y DISEÑO.

Actualmente las actividades en diseño se están expandiendo. El diseño tradicional basado en técnicas de representación gráfica y comunicación visual, se está fusionando con la tecnología y también, con las artes visuales. Evidencia de ello por una parte, la encontramos en las nuevas generaciones de ilustradores, pintores y productores audiovisuales que han incorporado conocimientos de programación y diseño interactivo en su quehacer profesional, y por otra, los ingenieros en informática y desarrolladores de sitios web, que están interesados en las artes visuales y el diseño, por la necesidad de mejorar su producción web-multimedia. Fig. 3.1. Sitio web de tatto salón elaborada para artistas del tatuaje.

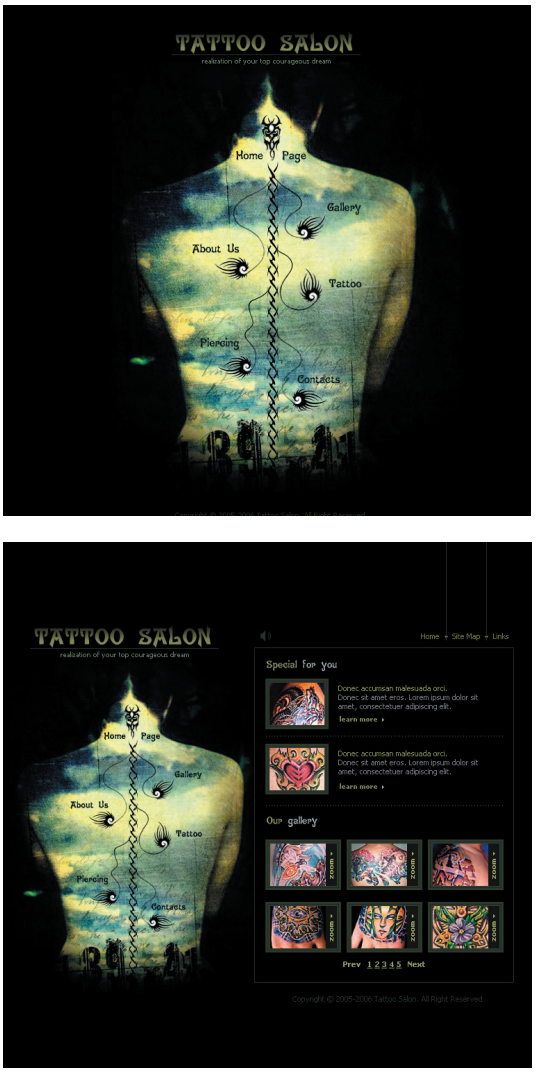



*Fig. 3.1*

Este punto de unión entre las disciplinas relacionadas al arte y la programación ha dado origen a manifestaciones como net-art, game-art y software art, que básicamente desarrolla piezas de arte a partir de código, imágen y código, o bien, código y sonido. José Damián Peralta nos dice: *"El web-art designa los proyectos creados específicamente para el WWW. Sin embargo el término más generalizado, pues es el que ha sido difundido históricamente y ha sido legitimado por el medio artístico, es net-art".* **Peralta Mariñelarena, José Damián.** *Net-art, Software-art, game-art, arte digital, computable y en línea en el periódo 1993 a 2008.* Tésis de Maestria en Artes Visuales. Pag 66. Fig. 3.2. Imagen tomada del sitio web de gameon.com, un sitio de artistas visuales que elaboran gráficos para videojuegos.


*Fig. 3.2*

Definitivamente, el arte es otro de los caminos para mejorar la producción multimedia a nivel audiovisual, revisando diversos materiales, encontré que muchos programadores, diseñadores y artistas visuales, han perfeccionado su trabajo interactivo aplicando principios básicos en arte y diseño, forma, composición, teoría del color, fundamentos del lenguaje audiovisual y arte sonoro. *"El arte es una herramienta que enriquece y consolida sistemas de pensamiento que permiten mejorar la formación integral en los niveles elementales y preparar al estudiante para enfrentar con mejores posibilidades de éxito, las dificultades de la educación profesional en los niveles superiores. Se estará mejor capacitados para enfrentar el estudio de las estructuras curriculares tradicionales gracias al desarrollo de habilidades para el dibujo, el diseño o la creatividad; competencias que las actividades artísticas contribuyen a desarrollar".* **López Monroy, Manuel.** *Las alas del deseo tecnológico. Reflexiones en torno al arte, la educación y la tecnología. Ensayo titulado: Los palacios de la memoria digital. Pag 73.* Fig. 3.3. Gemeovida. Pieza de software-art elaborada por Víctor Hugo Hernández Alcántara.

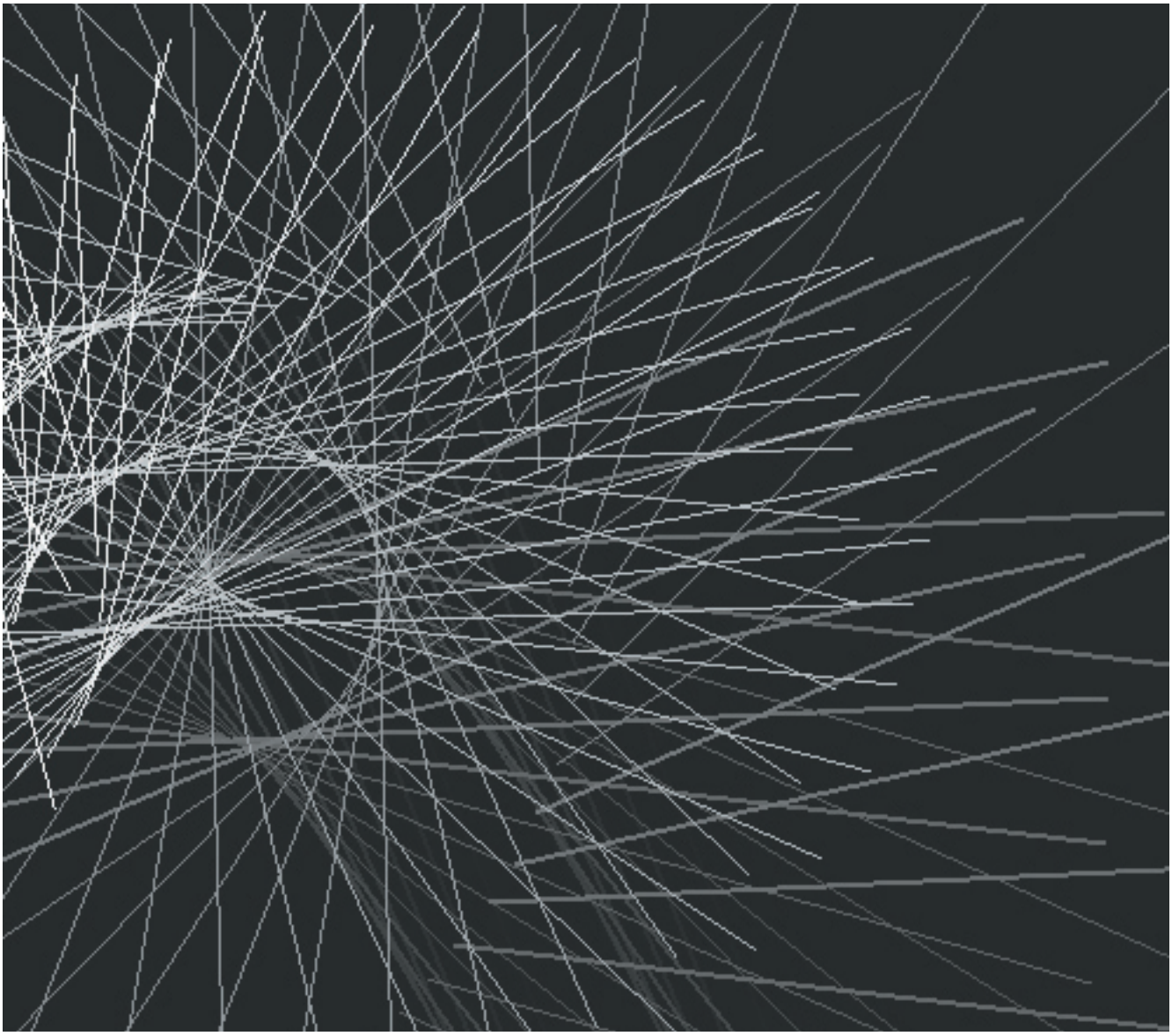


***Fig. 3.3***

Pero regresando a la cuestión del diseño de interacción aplicado al arte y diseño, vemos que notablemente el diseño tradicional ha evolucionado al arte y diseño digital - interactivo. Si analizamos la historia, veremos que dedicarse al diseño hace 10 o 20 años, era ejercer una actividad manual similar a la decoración o el dibujo publicitario, bien sabemos, que gran cantidad de diseñadores tradicionales se formaron en licenciaturas como diseño gráfico o comunicación gráfica. En su momento, estas profesiones se limitaron al desarrollo de las habilidades técnicas en la producción gráfica. Después se habló de diseño y comunicación visual; una fusión de ambas, cuyo objetivo principal era satisfacer la producción de mensajes visuales por medio de distintas especialidades. Con el paso de los años, estas especialidades se han diversificado en otras ramas o actividades de la disciplina. Recordemos algunas de éstas: Audiovisual y Multimedia, Ilustración, Fotografía, Diseño Editorial, Identidad Corporativa y Diseño en soportes Tridimensionales. Fig. 3.4 Imagen tomada del sitio atmosfera; experiencia visual, cuya producción fusiona fotografía, gráfica digital, video, animación y arte sonoro.


*Fig. 3.4*

Sabemos también que éstas distinciones a la fecha, no han podido responder a la resolución de las necesidades más actuales en comunicación y diseño, ello ha detonado el surgimiento de otros campos en la disciplina. Mencionemos algunos de ellos; asignaturas de la especialidad audiovisual y multimedia como dirección de arte o producción audiovisual, son licenciaturas por sí mismas. De las áreas de audiovisual, multimedia e ilustración digital, han emergido otros campos de especialización como la licenciatura en animación digital, la licenciatura en diseño de interacción y la ingeniería en diseño de videojuegos impartidos en instituciones privadas como la universidad panamericana, o la universidad de las artes digitales de Guadalajara, Jalisco, por mencionar algunos ejemplos. Del mismo modo, otras áreas como diseño editorial y multimedia, se han fusionado para generar especialidades en diseño web, diseño de interfaces gráficas o diseño de materiales didácticos interactivos. Como podemos ver, las actividades del diseño están cambiando continuamente los modos de expresión, las prácticas profesionales, las teorías y metodologías, los usos tecnológicos y las áreas de oportunidad.

En la actualidad, las áreas en diseño digital, están promoviendo una fusión entre ingeniería, informática, diseño y arte, volviendo cada vez más necesario para el especialista en multimedia, diseño interactivo y diseño de videojuegos; la incorporación de conocimientos en lenguajes de programación, física y matemática aplicada, así como otras ciencias y artes; ilustración, dibujo y pintura digital, composición musical, sintetizadores y secuenciadores, tecnología MIDI, audio digital, etc. La licenciatura en multimedia y la ingeniería en diseño de videojuegos es un claro ejemplo de ésta fusión. Ésta licenciatura en la mayoría de las universidades que la imparten, tiene casi siempre dos vertientes: la modalidad en arte o diseño digital y la modalidad en programación. Arte digital tiene asignaturas como: dirección de arte, dibujo tradicional y digital, ilustración digital, fotografía y video digital, diseño de gráficos 2D y 3D, animación 2D y 3D, diseño de escenarios, creación de personajes, guión y sonorización.  Programación tiene asignaturas como: matemáticas y física, programación 2D y 3D, sistemas operativos, bases de datos y lenguajes ensambladores, redes y programación web, inteligencia artificial, motores gráficos, mecánicas de juego etc. Todo ello habla de un cambio radical en la manera de enseñanza convencional en artes y diseño y una clara evolución del diseño tradicional al diseño digital-interactivo.

Respecto a esto agregaré mi última reflexión: La actividad de diseñar puede ser aplicada a muchas cosas, y tiene varios niveles de complejidad en la ejecución, como resultados. Tiene tipos de diseño y tipos de diseñadores; diseñadores con grandes habilidades manuales, y dominio técnico del ordenador, pero pocas habilidades conceptuales, y tiene también, diseñadores altamente intelectuales, conocedores de distintos lenguajes simbólicos; símbolos visuales, símbolos matemáticos,  símbolos musicales, lenguajes de programación, y una amplia cultura general obtenida a través de años de lectura y trabajo de investigación, y en mi opinión, ese es el diseñador que necesitamos.

En el planteamiento del problema, decíamos que la mayoría de los diseñadores multimedia carece de conocimientos básicos de matemáticas y programación, y esto ocurre porque los enfoques de pensamiento en las academias de artes y diseño están dirigidos más a las artes aplicadas, que a la ciencia y la tecnología, además de que las matemáticas y la programación no están contempladas en los planes de estudio de la licenciatura en diseño y comunicación visual, por pertenecer al área 4 de artes y humanidades. Si revisamos la historia del diseño observaremos que esto tiene un porque, y se remonta al modelo educativo de la Bauhaus, que provocó que la mayoría de los diseñadores se formara más en artes aplicadas y menos en ciencia y tecnología, éste último, un modelo de la escuela de ULM que planteaba la producción del diseño como un proceso racional.

Esta referencia histórica ha sido fundamental para este proyecto, que desde el comienzo ha buscado integrar pensamiento científico con pensamiento divergente al relacionar las matemáticas con lo visual y lo tecnológico, por la necesidad de racionalizar el proceso de producción visual para su aplicación en distintos soportes digitales - interactivos, una problemática que tocaremos en nuestro siguiente capítulo, que básicamente aborda ideas relacionadas a las matemáticas y la programación, y recopila proyectos multimedia en los cuales se aplican ambos conocimientos.

Fig. 3.5. Galería cilíndrica 3D con fotografías. Elaborado en flash por Barbara Kaskosz para la universidad de Rhode Island. Año 2007. http://www.flashandmath.com/flashcs4/cylingallery/cylin_gallery.html

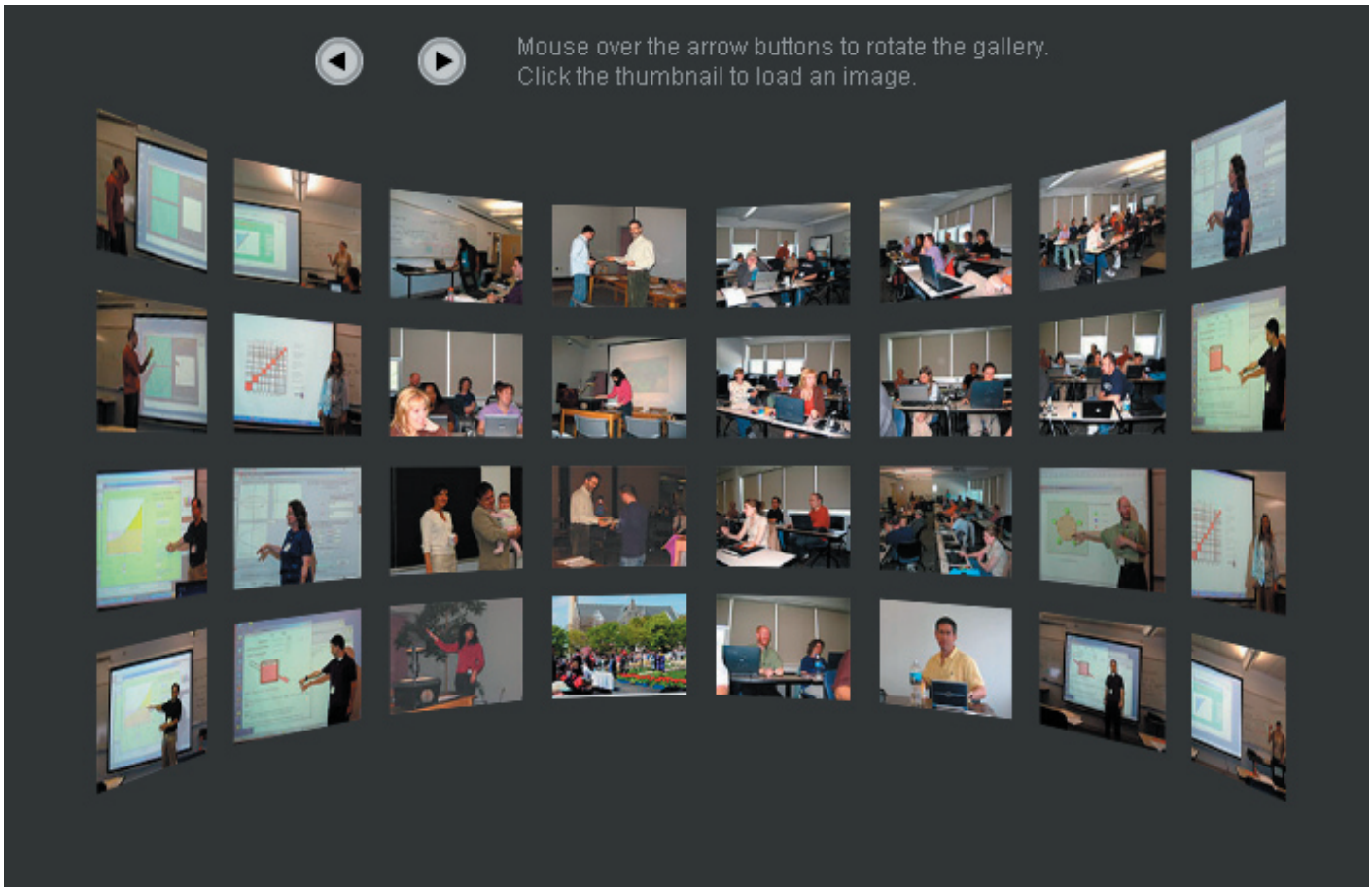



***Fig. 3.5***

# IV. LAS MATEMÁTICAS APLICADAS AL DISEÑO DE INTERACTIVOS DIDÁCTICOS.

# IV. LAS MATEMÁTICAS APLICADAS AL DISEÑO DE INTERACTIVOS DIDÁCTICOS.

Para este capítulo se exponen algunas ideas acerca de la relación entre matemáticas aplicadas en interactivos didácticos y programación. Además se realizó una investigación documental, en la cual se estudiaron 18 casos de éxito en los cuales la producción multimedia mejora mediante la utilización de ambas disciplinas en términos de visualización, interactividad, animación, diseño y funcionalidad. Éstos trabajos actualmente se encuentran publicados en diferentes sitios web. Fig. 4.1. Interactivo del cubo. Rota el cubo con el ratón.http://www.flashandmath.com/advanced/simple3d/index.html  Fig. 4.2. Interactivo del dodecaedro. Rota el dodecaedro con el raton. http://www.flashandmath.com/advanced/dode/index.html Fig. 4.3. Interactivo de la esfera. Rota la esfera con el ratón. http://www.flashandmath.com/advanced/simple3d/sphere.html Los tres interactivos han sido elaborados por Barbara Kaskosz en Action Script 3.0. para el sitio web: www.flashandmath.com

*Fig. 4.2*

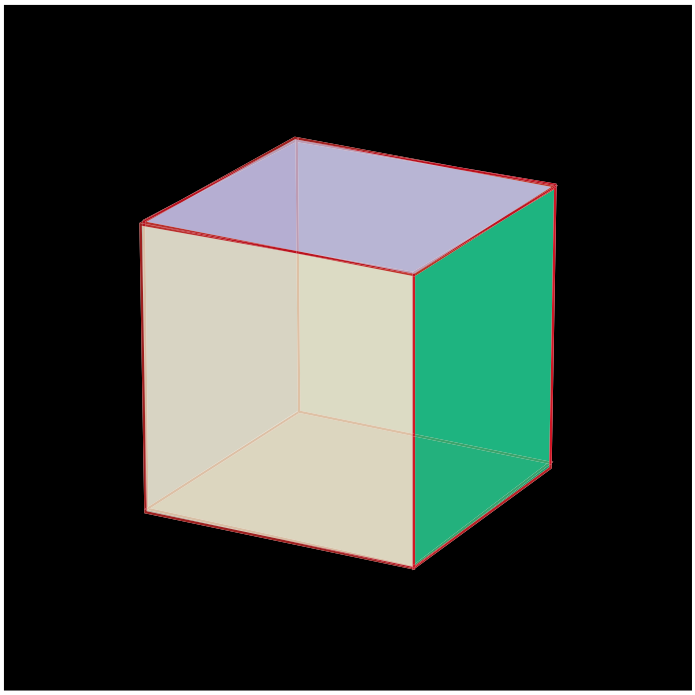


*Fig. 4.1*

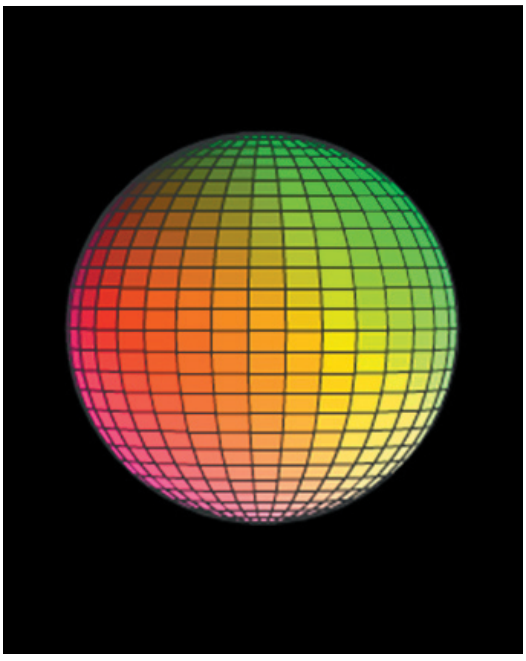


*Fig. 4.3*

Edsger Dijkstra, un importante físico de la computación alguna vez comentó: *"La interacción humana con el computador, es una manipulación mecanizada de símbolos. La programación y la informática esta situada en el área de las matemáticas formales y la lógica aplicada"* **Dijkstra, Edsger,** "On the cruelty of really teaching computer science". Pag. 16-17. Universidad de Texas. https://www.cs.utexas.edu/users/EWD/ewd10xx/EWD1036.PDF.

Tomando esta idea, entendemos que para la ciencia de la computación, la lógica como una rama de las matemáticas es imprescindible para la resolución de problemas informáticos en aspectos de programación. Por lo tanto, un lenguaje de programación, es un idioma artificial diseñado para expresar algoritmos y controlar el comportamiento físico-lógico de una máquina. La palabra programación, se define como un proceso de creación de programas computacionales mediante la aplicación de procesos lógicos traducido a la producción de software o una pieza de animación interactiva con un contenido didáctico. Fig. 4.4. Interactivo del péndulo matemático. Arrastra el péndulo matemático. Ejemplo tomado del sitio web: www.crystalab.com


*Fig. 4.4*

Barbara Kaskosz, doctora en matemáticas y desarrolladora de interactivos didácticos ha expuesto lo siguiente: *"Los conocimientos de matemática aplicada a lenguajes de programación, determinan las físicas de movimiento, las animaciones de particulas y los efectos visuales para la simulación de explosiones estelares, fluidos y electromagnetísmo"*. Cita tomada del vínculo: http: //www.flashandmath.com/flashcs4/nova. Consulta realizada en agosto de 2013. Fig. 4.5. Simulación de una explosión estelar. Interactivo elaborado por Douglas Ensley para el sitio: www.flashandmath.com Fig. 4.6. Generador de fractales elaborado por Dan Gries. Basado en la Geometria Fractal de Benoit Mandelbrot también conocida como geometría de la naturaleza. Este interactivo podemos encontrarlo en el siguiente vínculo: http://www.flashandmath.com/advanced/mandelbrot/MandelbrotPlot.html

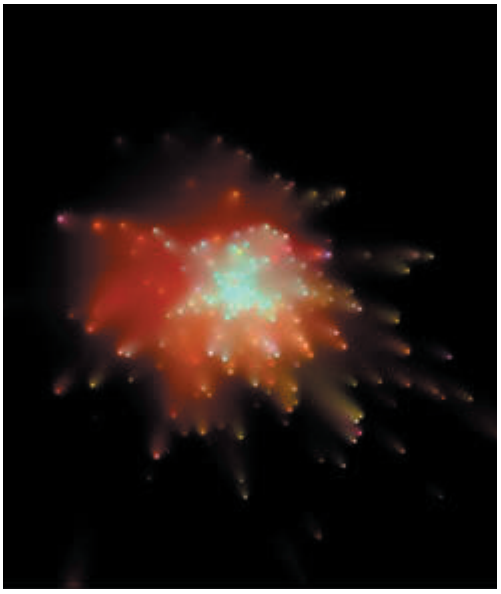


***Fig. 4.5***

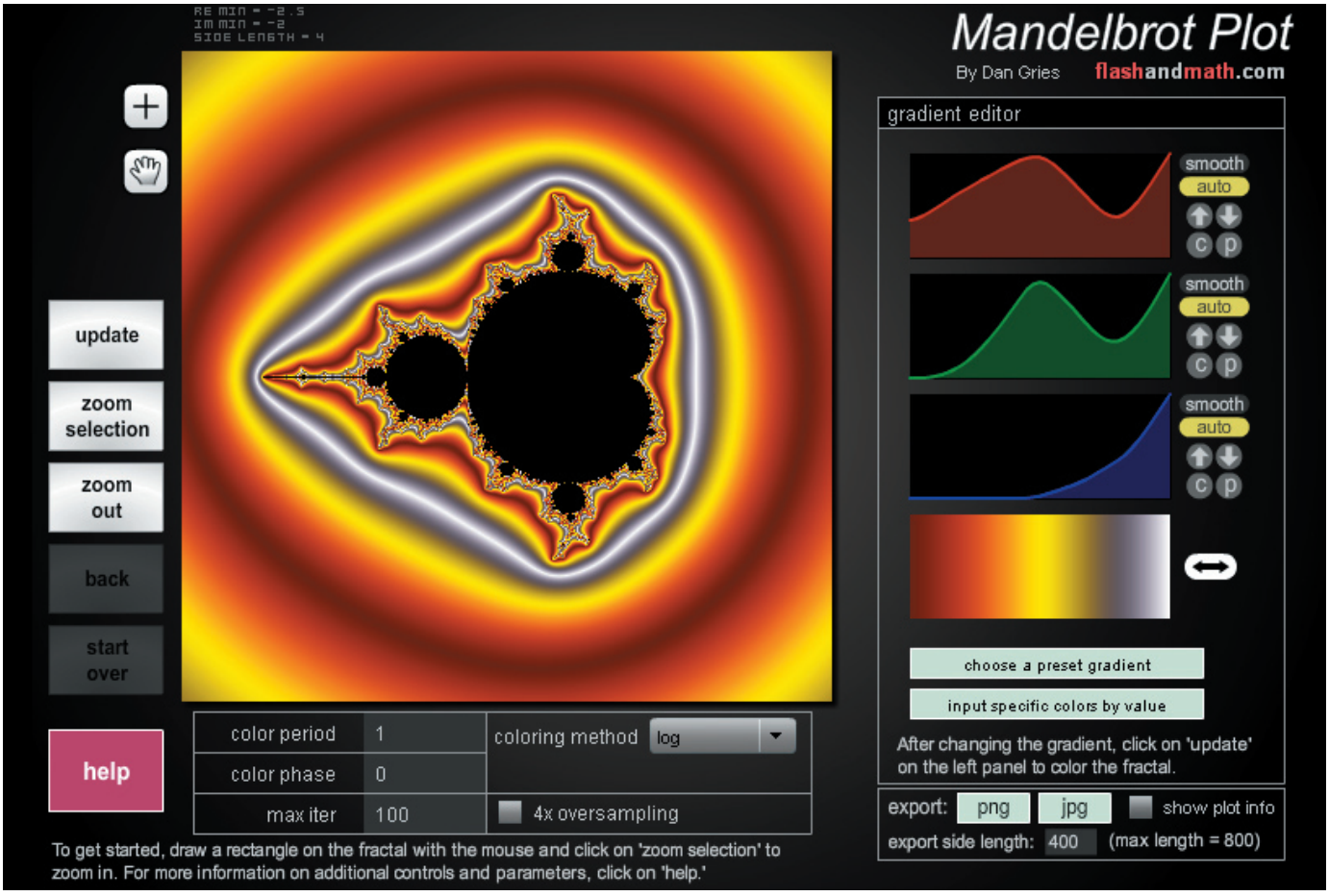



***Fig. 4.6***

En cuanto a matemática y geometría relacionada al diseño, la Doctora Luz del Carmen Vilchis ha compartido la siguiente reflexión: *"Las matemáticas y la geometría integran los conocimientos de magnitudes y cantidades y sus posibles variaciones espacio temporales a la solución de escalas, razones y proporciones, transformación, comparación y propiedades del espacio: fragmentaciones, diagramación y sistemas de retículas, normalizaciones, perspectivas, desarrollo de planos, relieves y volúmenes, etc".* **Vilchis, Luz del Carmen.** *Diseño. Universo de conocimiento. Investigación de proyectos en la comunicación gráfica.* Pag. 98. Centro Juan Acha A.C. Año 2002.

Tomando como referencia los trabajos en diseño interactivo de Dan Gries, Douglas Ensley y Barbara Kaskosz, podemos decir que los interactivos que explican modelos matemáticos, los utilizamos y producimos para explicar las reglas de un teorema. Para demostrar por medio de visualización o simulación, las proposiciones o la construcción de un modelo en química, física o matemáticas, bien sabemos que el proceso enseñanza-aprendizaje de la ciencia es sumamente complejo, y a través del tiempo, el hombre ha desarrollado gran diversidad de metodologías para lograr transmitir las ideas abstractas de manera clara y entendible. Con la llegada de las nuevas tecnologías, en particular las computadoras, se abre un nuevo campo de investigación en cuanto a nuevos ambientes de aprendizaje, producción interactiva y metodologías de enseñanza, aprovechando el enorme potencial que tienen los recursos digitales. En el caso del estudio de las matemáticas aplicadas al diseño de interactivos, existe una mutua relación:

1) El aprendizaje de las matemáticas se hace necesario para el desarrollo de éstas aplicaciones tecnológicas, y

2) El uso de aplicaciones tecnológicas, ayuda en el proceso enseñanza-aprendizaje y en el estudio de los sistemas implicados en su desarrollo (lenguajes de programación, software, etc.).

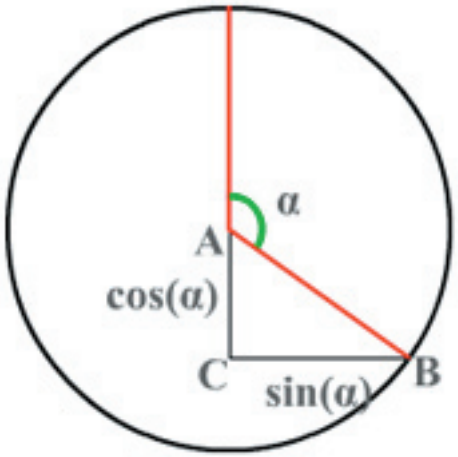
Respecto a matemáticas, sabemos que existe una infinidad de libros, éstos materiales, son de distintos temas y constantemente, son abordados desde diferentes perspectivas. Sin embargo, no es la finalidad de éste documento, elaborar un tratado más sobre ésta importante ciencia, sino más bien, hablar sobre algunos aspectos relacionados a las matemáticas, que están involucrados en el diseño de interactivos, un tema presente en ésta investigación, porque sabemos que la aplicación del conocimiento matemático en éstas tecnologías, se ha ampliado mediante la utilización de novedosos sistemas, técnicas y procesos, que implican el uso de conocimientos como: Teoría de los números, álgebra lineal, geometría, análisis matemático, etc. Recordemos que ésta ciencia es universal y la forma de ponerla en práctica, debe responder a las circunstancias del entorno, por ello, en el caso de la producción de los interactivos del punto, la línea y el plano, fue importante estudiar el tema de geometría analítica además de distintas posibilidades en lenguajes de programación. En cuanto a relaciones interdisciplinarias secundarias al diseño, la Doctora Vilchis habla de la física: *"Contribuye con los estudios especializados acerca de la luz y el color, materias como la óptica y la cromática respectivamente y los correspondientes al movimiento: carácter, posición, velocidad y cinemática, materias de la cinética, la dinámica y la mecánica".***Vilchis, Luz del Carmen.** *Diseño. Universo de conocimiento. Investigación de proyectos en la comunicación gráfica.* Pag. 101. Centro Juan Acha A.C. Año 2002.

Revisando el trabajo de Dan Gries, podemos observar que sus códigos ponen en práctica distintas fórmulas de la física: fuerza de gravedad, resistencia y velocidad, cinemática y dinámica. Fig. 4.7. Modelo matemático de Edward Lorenz aplicado a una animación interactiva. Elaborado por Dan Gries para el sitio web: http://www.flashandmath.com/advanced/bubbles/lorenz_bubbles.html

Respecto al diseño de interactivos en los que intervienen matemáticas aplicadas, podemos decir que su producción requiere tanto del compromiso y estudio de ésta ciencia, como el estudio de cursos, artículos y tutoriales relacionados a lenguajes de programación, ya sea: HTML5, JavaScript, Action Script, CSS3, Processing, C, C++, Phyton, Java, XML o cualquier otro. En cuanto estos lenguajes, podemos encontrar diversos foros y sitios web de profesionales expertos que desarrollan sus propios interactivos y publican diferentes trabajos, los cuales, podemos descargar de sus sitios web y estudiar mediante un proceso de análisis. En pocas palabras, debemos dedicar tiempo para tratar de entender como fueron construidas éstas aplicaciones, e indagar en sus contenidos. Fig. 4.8. Interactivo de las funciones trigonométricas elaborado en Action Script 2.0. Ejemplo descargado del sitio web: www.crystalab.com.


*Fig. 4.7*

Una de las ventajas de ésta forma de estudio, es el alto grado de motivación e interés que podemos generar con la utilización de dichos recursos, además de los distintos manuales y trabajos elaborados que podemos encontrar en internet (ejemplo: maestros de la web, flashandmath o cristalLab) que nos permiten descubrir poco a poco nuevos universos, nuevas cosas, ideas aplicadas que nos permitirán avanzar gradualmente desde los niveles más básicos en animación y programación, hasta los más avanzados en programación de interactivos con matemáticas, o diseño de software ya sea de manera autodidacta, o bien, con la orientación del profesor.




α= 2.19 rad = 125.54°

sin(α)= 0.8137

cos(α)= -0.5812

B(0.8137 , 0.5812)

*Fig. 4.8*

En este caso, siempre lo deseable es tener la asesoría de un maestro que nos ayude en dicho proceso, ya que su conocimiento nos puede aclarar muchas de las dudas que surgen durante el aprendizaje de los códigos de programación. Es así como, con el estudio de matemáticas, tutoriales, plantillas de código, prácticas y asesorías, el alumno interesado en desarrollar trabajos de ésta índole, descubrirá los elementos necesarios que le ayudarán a desarrollar una estructura cognitiva sólida y natural que le permitirá construir un puente entre las ideas intuitivas, la imaginación y los conceptos teóricos-formales que se requieren y que se encuentran detrás de todos éstos trabajos. Todo ello, en favor de enriquecer los proyectos, así como el crecimiento en las capacidades intelectuales y la adquisición de destrezas para visualizar, representar en distintos soportes tradicionales o virtuales, o bien, construir modelos, una habilidad especial para codificar la realidad, en la que interviene el uso de la ciencia y la tecnología.

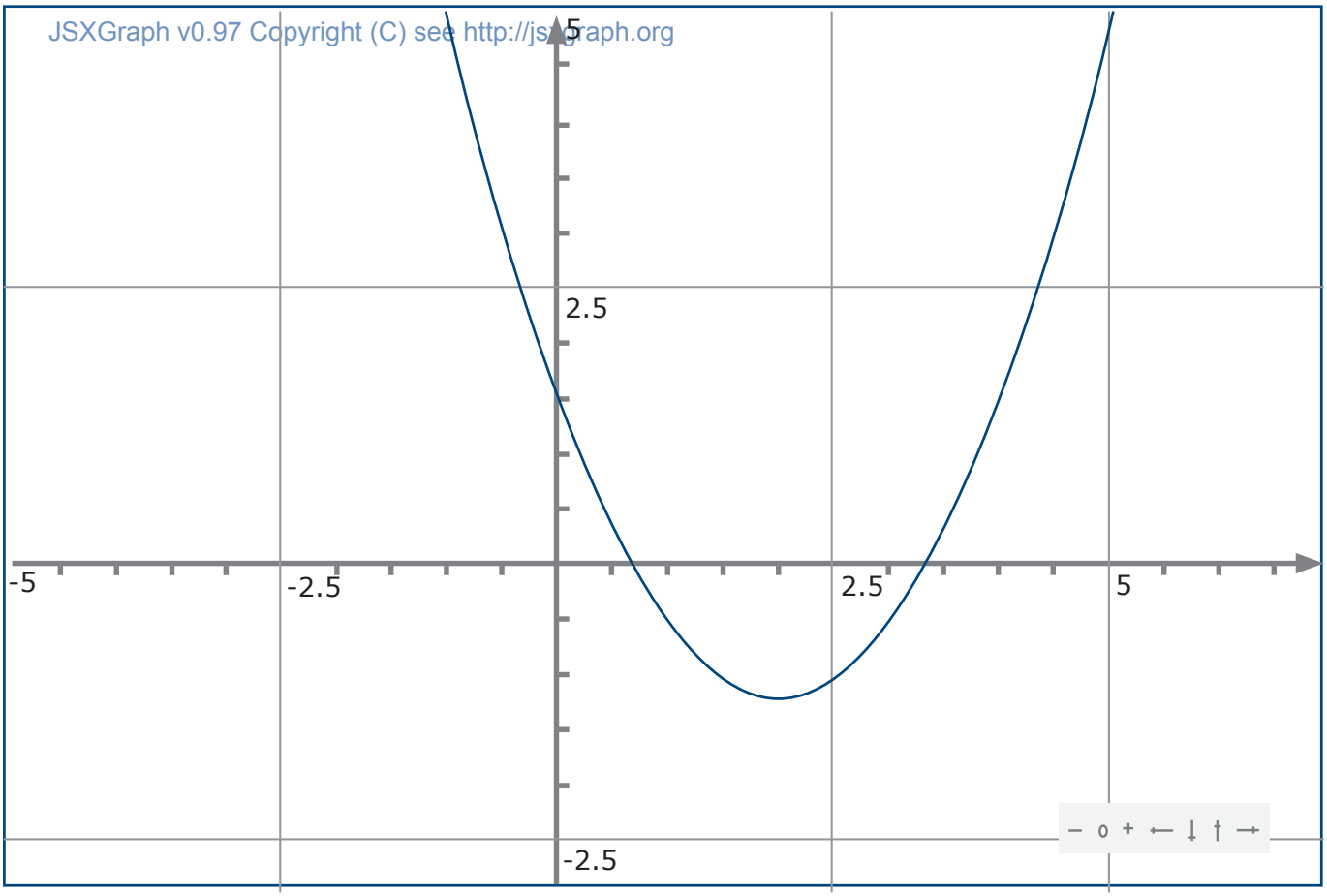



*Fig. 4.9*

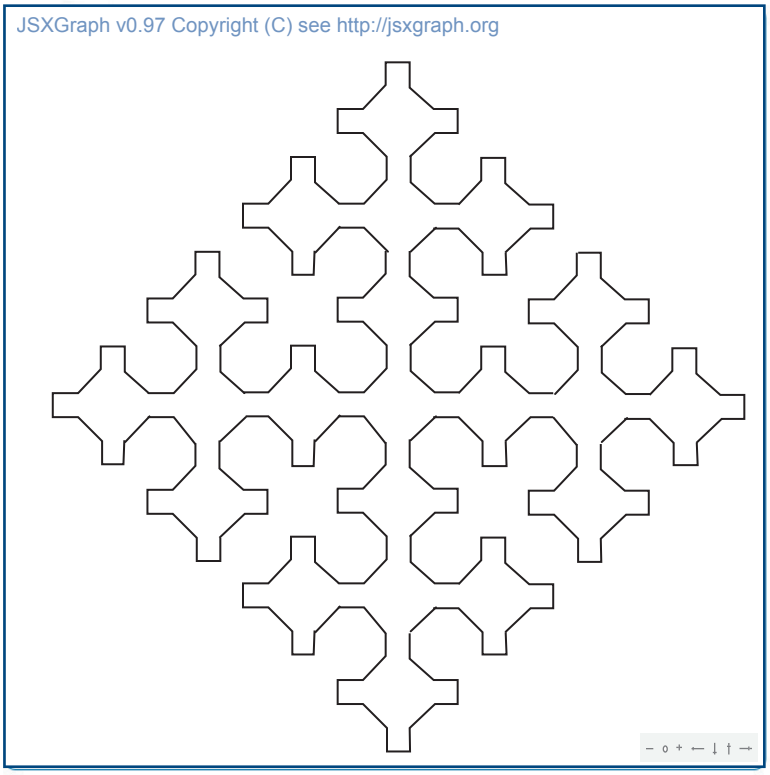



*Fig. 4.10*

Bianca Valentin y Michael Gerhäuser desarrolladores de interactivos en JavaScript han dicho: *"Jsxgraph se diseño para graficar de manera adecuada datos matemáticos en los exploradores de internet"*. **Bianca Valentin y Michael Gerhäuser.** Pag 3. Interactive SVG with JSXGraph. Manual de referencia. 2009.

Fig. 4.9. Interactivo de la parabola en Jsxgraph, el cual lo podemos encontrar en el siguiente link: http://jsxgraph.uni-bayreuth.de/wiki/index.php/Parabola.

Fig. 4.10. Interactivo de la curva sierpinski en Jsxgraph el cual podemos encontrarlo en el siguiente link: http://jsxgraph.uni-bayreuth.de/wiki/index.php/sierpinski_curve

El Maestro Juan José Carreón. Responsable del proyecto: Diseño de aplicaciones distribuidas, interactivas y gráficas, en Android, DGAPA, UNAM. Ha dicho lo siguiente: *“Las matemáticas están presentes en todo código que se genera; al igual que en las relaciones entre los constructos y patrones empleados para producirlo, sin embargo, esas relaciones no son patentes debido a una insuficiente educación de los programadores que les impide formalizarlas en modelos que les permitan comprender tanto las consistencias como las inconsistencias de dichas relaciones”.*

Tomando esta idea podemos decir que parte de éste trabajo pretende proponer a los diseñadores, el uso de las matemáticas como un conocimiento aplicado en el desarrollo de interactivos multimedia, y para lograr ésto, es indispensable lograr una cultura como programadores, que implica la búsqueda, el estudio y la utilización de toda una gama de recursos y herramientas (algunas gratuitas) que nos provee la misma red de internet, código libre y software gratuito para desarrollar nuestros propios trabajos. Pues cada dia, aparecen más y más códigos y lenguajes para desarrollar aplicaciones, herramientas, librerias, etc.

Además de ésto, también es necesario actualizarnos en los temas de ciencia, tecnología y arte, así podremos tener más recursos para diseñar y desarrollar proyectos interactivos, y no sólo en el tema de las matemáticas, sino para todas las cuestiones relacionadas a lo abstracto y lo intangible, temas de difícil comprensión para la gente común. Pues hoy en día no es posible dejar de vincular la educación matemática con la computacional y prescindir de la tecnología en las escuelas y universidades. Los nuevos conocimientos en ciencia y tecnología, hace cada vez más necesario, el uso de aplicaciones de software y herramientas de apoyo en el proceso formativo de docentes y estudiantes, que implica, la utilización de pizarrones electrónicos, hardware, software e interactivos didácticos para la exposición de clases y explicación de conceptos. Fig. 4.11. Generador de imagenes a partir de curvas parametricas con el uso de sliders diseñado por Barbara Kaskosz.http://www.flashandmath.com/flashcs4/curves/curves.html

Jeremy Kun, un importante matemático-programador quien ha dedicado su investigación a las ciencias de la computación y las matemáticas, ha dicho: “*Considero se requiere que los programadores comprendan aspectos como los de que: Las matemáticas son excelentes para captar la esencia de un problema, con respecto al cual la programación es extraordinaria para explorar y formalizar preguntas, así como para automatizar tareas”.* **Jeremy Kun**, “Thoughts after a Year of Math ∩ Programming, June 12, 2012, http://jeremykun.wordpress.com/2012/06/12/thoughts-after-a-year-of-math-programming/

Idea que se complementa con las últimas reflexiones del Maestro Juan José Carreón, quien relaciona de manera contundente ambas disciplinas: *“Las matemáticas se alimentan de la emoción de comprender cómo y por qué se relacionan las cosas; y la programación, de lo que puede lograrse.Un buen conocimiento matemático permite desarrollar programas sencillos y potentes; sin embargo, a veces algunos desarrollos matemáticos no son suficientemente cuidadosos, algo que nunca sucede con un buen programa. Los algoritmos empleados en la programación pueden provenir de cualquier rama de las matemáticas, por lo que a los programadores les conviene conocerlas en alguna medida. Las abstracciones en matemáticas son parecidas a las de la programación; sin embargo, estas últimas pueden ser arduas; en cambio, las primeras instantáneas. De ahí, la importancia, de tener claro tanto por legos, como por programadores tanto aficionados como profesionales, que todo programa divertido, emocionante, o productivo de forma ineludible se apoya en conocimientos matemáticos”.* Estas ideas fueron expuestas en el coloquio compartiendo experiencias de enseñanza basadas en TIC. Tecnologías de la Información y la Comunicación dirigido a estudiantes de la facultad de ingenieria de la UNAM. Fig. 4.12. Generador de superficies tridimensionales diseñado por Barbara Kaskosz http://www.flashandmath.com/advanced/surfaces/index.html

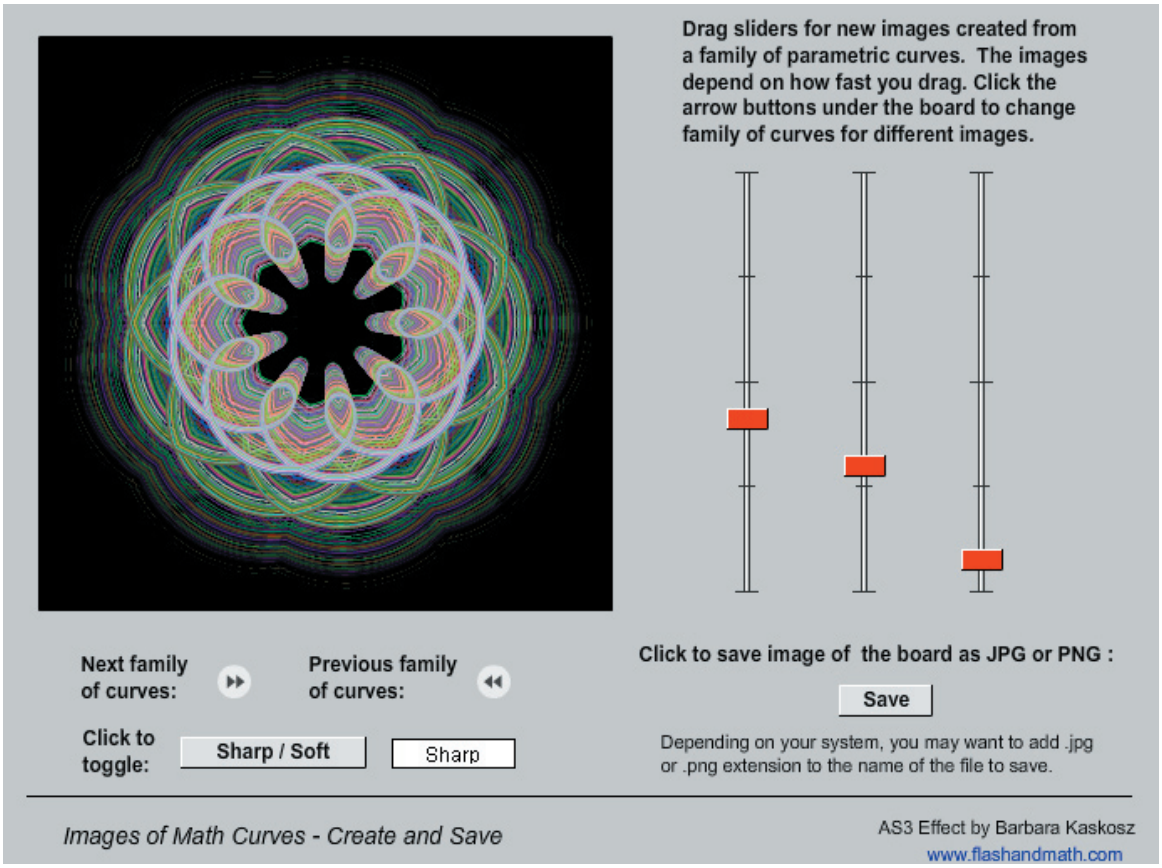
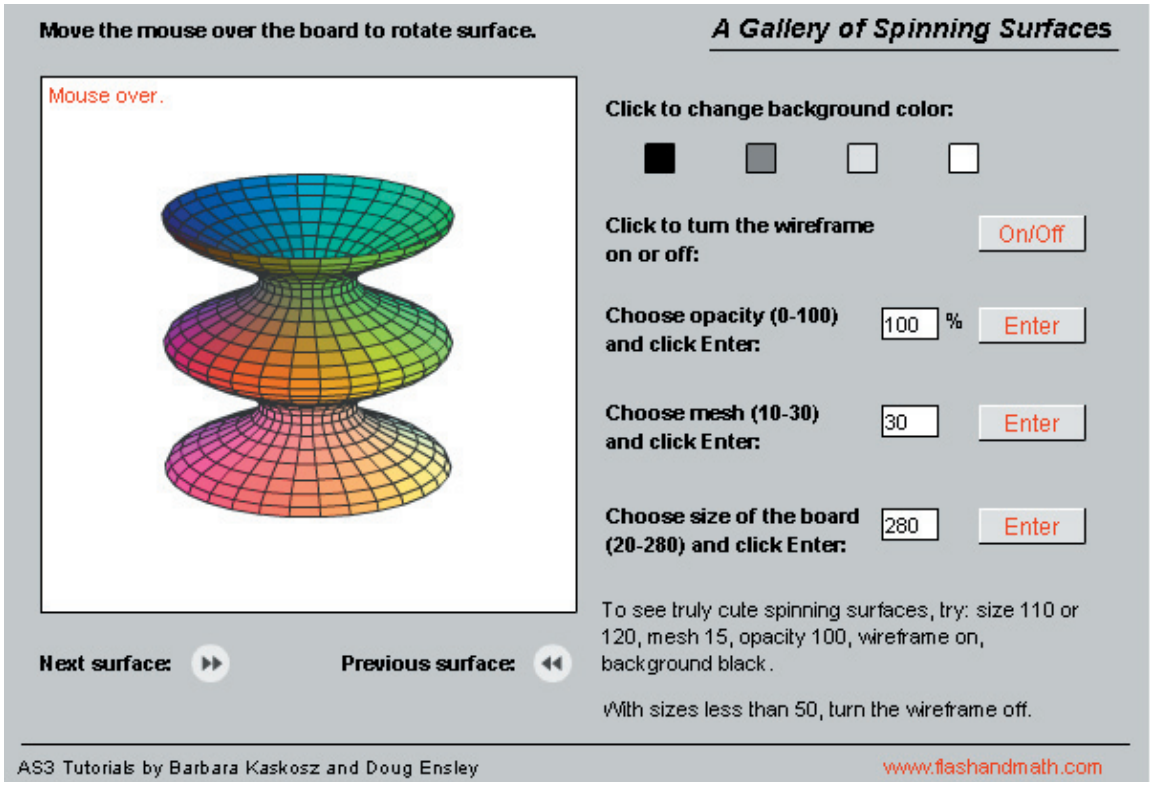
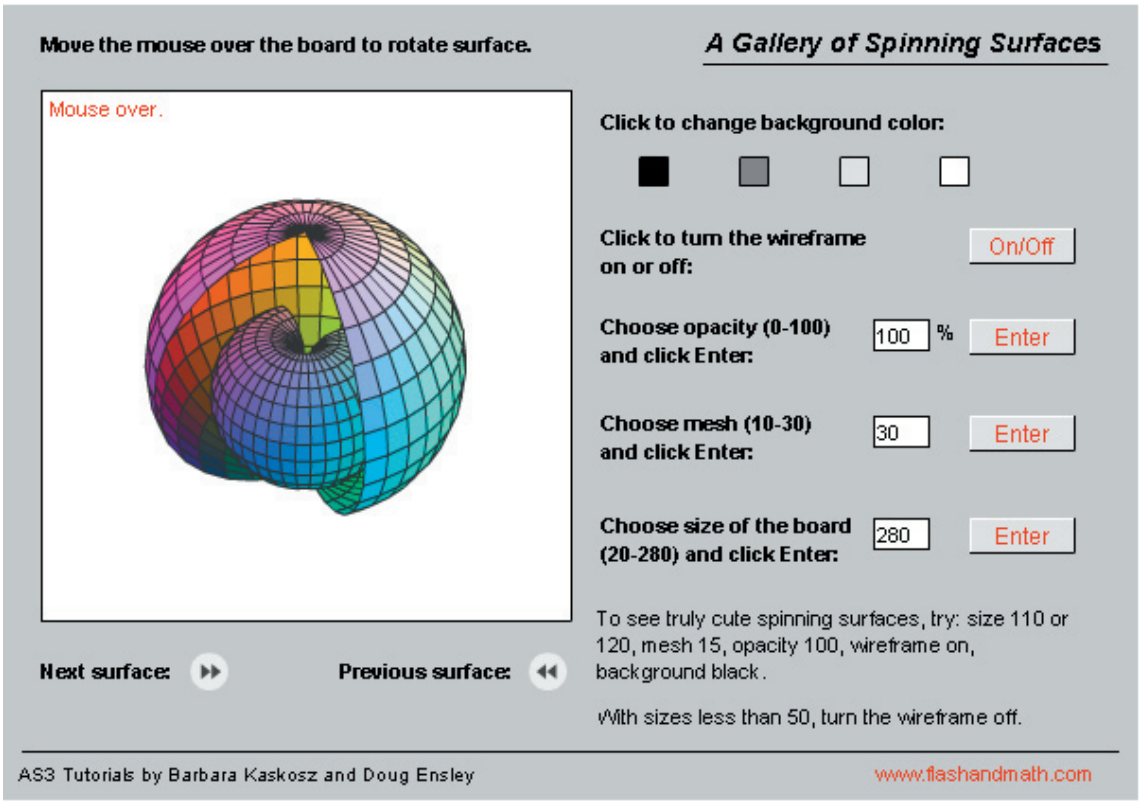

Drag sliders for new images created from a family of parametric curves. The images depend on how fast you drag. Click the arrow buttons under the board to change family of curves for different images.
Next family of curves:
Previous family of curves:
Click to toggle:
Sharp / Soft
Sharp
Click to save image of the board as JPG or PNG :
Save
Depending on your system, you may want to add .jpg or .png extension to the name of the file to save.
Images of Math Curves - Create and Save
AS3 Effect by Barbara Kaskosz
www.flashandmath.com

*Fig. 4.11*


Move the mouse over the board to rotate surface.
A Gallery of Spinning Surfaces
Mouse over.
Click to change background color:
Click to turn the wireframe on or off:
On/Off
Choose opacity (0-100) and click Enter:
100
%
Enter
Choose mesh (10-30) and click Enter:
30
Enter
Choose size of the board (20-280) and click Enter:
280
Enter
To see truly cute spinning surfaces, try: size 110 or 120, mesh 15, opacity 100, wireframe on, background black.
With sizes less than 50, turn the wireframe off.
Next surface:
Previous surface:
AS3 Tutorials by Barbara Kaskosz and Doug Ensley
www.flashandmath.com


Move the mouse over the board to rotate surface.
A Gallery of Spinning Surfaces
Mouse over.
Click to change background color:
Click to turn the wireframe on or off:
On/Off
Choose opacity (0-100) and click Enter:
100
%
Enter
Choose mesh (10-30) and click Enter:
30
Enter
Choose size of the board (20-280) and click Enter:
280
Enter
To see truly cute spinning surfaces, try: size 110 or 120, mesh 15, opacity 100, wireframe on, background black.
With sizes less than 50, turn the wireframe off.
Next surface:
Previous surface:
AS3 Tutorials by Barbara Kaskosz and Doug Ensley
www.flashandmath.com

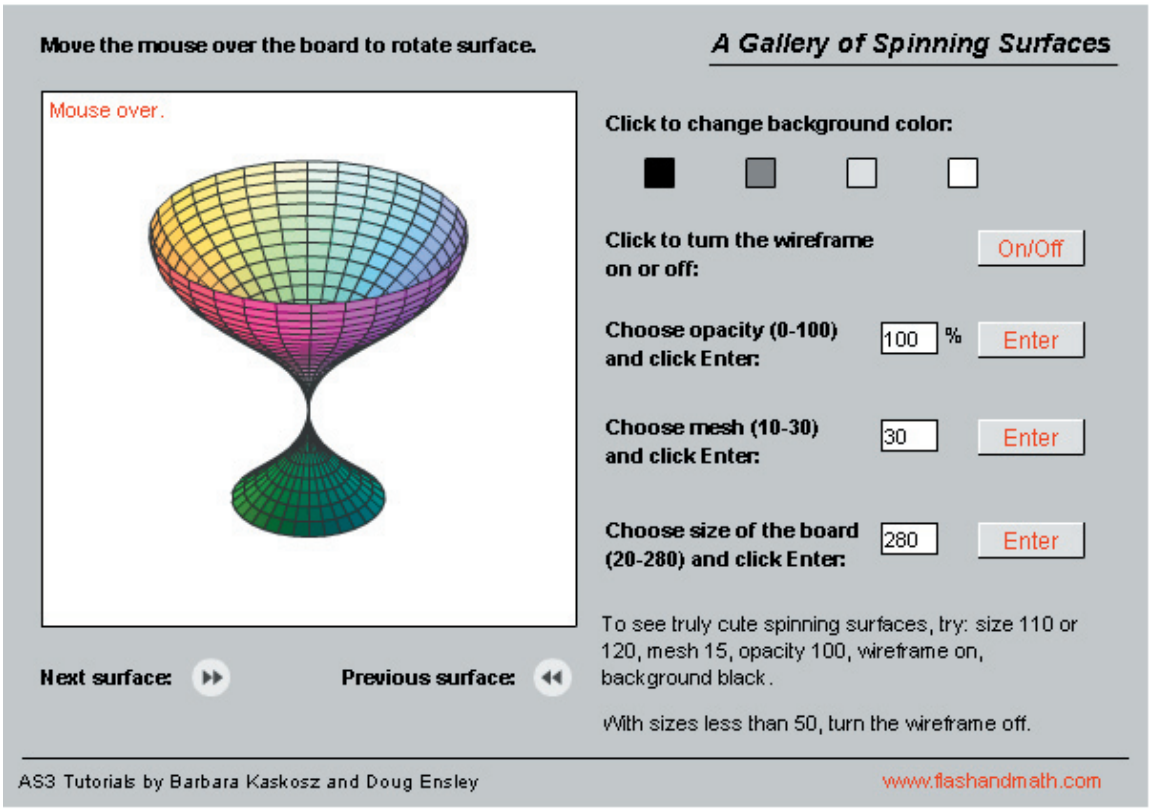

Move the mouse over the board to rotate surface.
A Gallery of Spinning Surfaces
Mouse over.
Click to change background color:
Click to turn the wireframe on or off:
On/Off
Choose opacity (0-100) and click Enter:
100
%
Enter
Choose mesh (10-30) and click Enter:
30
Enter
Choose size of the board (20-280) and click Enter:
280
Enter
To see truly cute spinning surfaces, try: size 110 or 120, mesh 15, opacity 100, wireframe on, background black.
With sizes less than 50, turn the wireframe off.
Next surface:
Previous surface:
AS3 Tutorials by Barbara Kaskosz and Doug Ensley
www.flashandmath.com

*Fig. 4.12*

Respecto a toda la gama de materiales didácticos que podemos encontrar para nuestros desarrollos, existe una gran cantidad de videotutoriales y librerias de programación para distintos usos, muchos gratuitos. En cuanto a código libre se refiere, además de los lenguajes que todos conocemos, también tenemos los llamados frameworks. Los frameworks son herramientas que utilizan la gran mayoría de los programadores actualmente en el desarrollo de aplicaciones para web, teléfonos celulares o para el ordenador. Algunas de las ventajas que éstas herramientas nos brindan, son la agilidad y rapidez para desarrollar trabajos interactivos, interfaces gráficas y otros recursos. Los que utilizamos para desarrollar los primeros dos ejemplos del capítulo II son JsxGraph y HTML5, pero sería bueno revisar otras bibliotecas de código como: CSS3, Three.js, Phyton, Sencha, Facebook api, Google api, twitter api, ruby on rails, phonegap etc.Todos recursos para desarrollar diferentes proyectos.

En este proceso llamado educación, se vuelve necesario el uso de herramientas tecnológicas para las cuestiones relacionadas al estudio de teorías y comprobación de hipótesis mediante simulación de fenómenos y manipulación de variables. Para éstos casos, el uso de la computadora, y el desarrollo de un interactivo o software específico para cada aprendizaje, se convierte en una necesidad fundamental en las cuestiones didácticas, porque el programa o aplicación, refuerza y apoya el aprendizaje por descubrimiento, un universo de entornos interactivos donde existe un amplio rango de eventos en los que ocurren intercambios de recursos y datos entre usuarios. Según Barker, existen 2 modalidades de sistemas interactivos en aprendizaje: los centrados en el hombre y aquellos basados en la tecnología:*"Los sistemas centrados en el hombre pueden diferenciarse por un tipo de interacción en la que los sistemas comunicadores son personas que entran en diálogo con el objeto de facilitar algún proceso de aprendizaje. La interacción entonces puede desarrollarse de tres formas distintas: uno a uno (profesor-alumno), uno a varios (profesor-grupo de alumnos), o bien varios a varios (trabajo en equipo, trabajo en grupo). Por otro lado, en los sistemas basados en tecnologías, el proceso de diálogo se desarrolla entre el alumno o alumnos y el uso de tecnologías para iniciar y mantener los procesos de enseñanza-aprendizaje que se tratan de sostener. El carácter de este tipo de interacción, va a depender en muchos casos del correcto uso que se haga de la tecnología, en este caso, las interacciones controladas y con instrucciones especificas. Y de todas las que pueda haber, las más efectivas son: videoconferencia, CD ROM, video interactivo, web site, hipermedia, simulación, libro electrónico, etc.".* **Barker, P.** *Basic Principles of Human Computer Interface Design.* London, Hutchinson, USA. Fig. 4.13. Interactivo de la fuerza de gravedad. Usa las flechas del teclado para mover, elevar y botar la pelota conforme ciertos principios basados en la fuerza de gravedad. Copyright (c) 2001 Ultrashock, Inc. Interactivo descargado del siguiente vínculo: http://www.alesys.net Fig. 4.14. Curva con rebote. Arrastra la curva y sueltala. La curva de este interactivo se comporta como una liga, basta que el usuario jale la curva para observar su comportamiento como una banda elástica. El código de este trabajo aplica fórmulas de la física relacionadas a aspectos como: Aceleración, desaceleración, velocidad, distancia, fricción, etc. Interactivo descargado del sitio web: www.nomaster.com

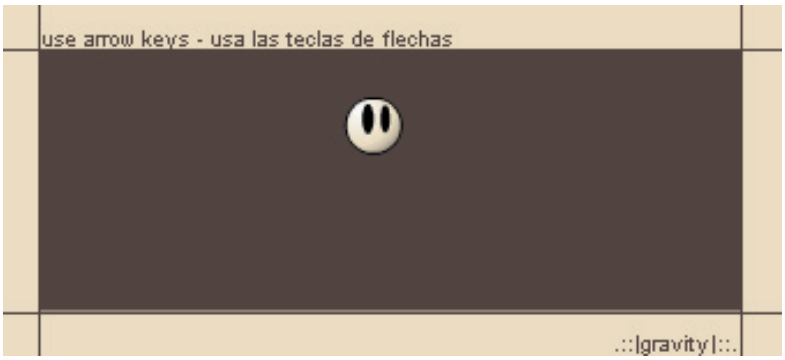



*Fig. 4.13*


*Fig. 4.14*

Por todo lo anterior y ya para concluir éste capitulo, extiendo una cordial invitación a los diseñadores de la comunicación visual interesados en el estudio de la multimedia, a que expandan sus horizontes y vean otras posibilidades en el desarrollo y producción de ésta disciplina, para que involucren a su acervo cultural, conocimientos de física, matemáticas, nuevas tecnologías, y también lenguajes de programación, porque éstos conocimientos, le permitirán mejorar su producción multimedia y desarrollar otras capacidades intelectuales, mismas que le ayudarán a enriquecer, sus proyectos profesionales. Sin nada más que agregar en este apartado, dejaré para el final, más ejemplos. Fig. 4.15. Gas ideal. Modelos de Boyle, Mariotte, Charles y Gay-Lussac. En este interactivo se simulan ciertos aspectos del comportamiento de los gases. Temperatura, volumen, presión. Fig. 4.16. Interactivo: Representación de lo infinito. Este interactivo fue realizado en JSXGraph y podemos verlo en el siguiente vínculo. http://jsxgraph.uni-bayreuth.de/wiki/index.php/Infinity

Fig. 4.17. Interactivo para manipular una espiral hiperbolica. Practicamente en Jsxgraph, podemos generar todo tipo de espirales; la espiral Fibonacci, la espiral de Arquímedes, la espiral logarítmica, la espiral de Fermat, la espiral hiperbólica, la espiral de Durero y manipularlas con sliders para girarlas, acercarse o alejarse de ellas. El vínculo donde se encuentra es el siguiente: http://jsxgraph.uni-bayreuth.de/wiki/index.php/Hyperbolic_spiral

Fig. 4.18. Interactivo para trazar formas florales. En Jsxgraph, podemos generar curvas paramétricas, cardioides y otras formas similares a aquellas que se pueden trazar con el uso de un espirógrafo. Su vínculo es el siguiente: http://jsxgraph.uni-bayreuth.de/wiki/index.php/Rose

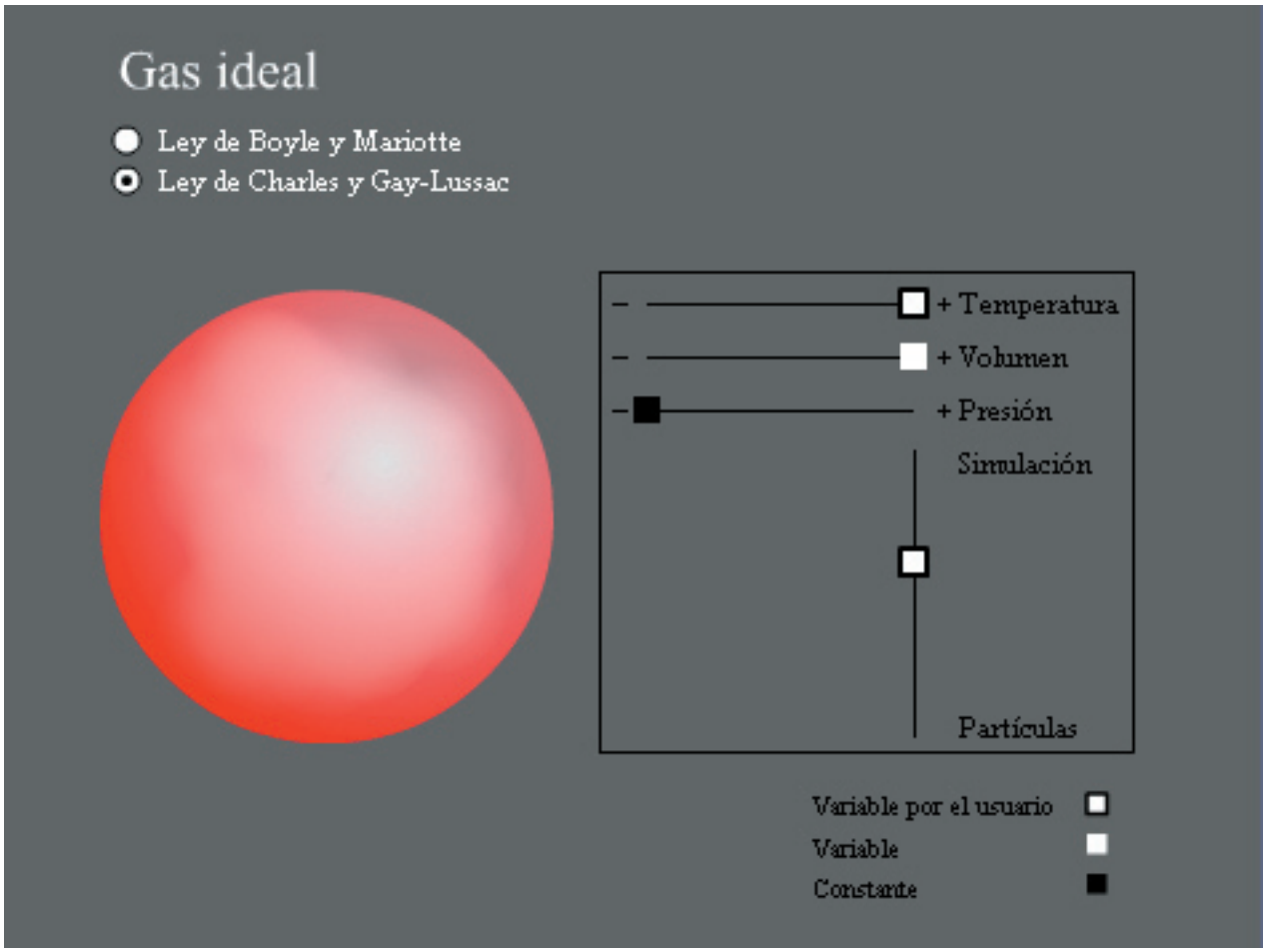



***Fig. 4.15***

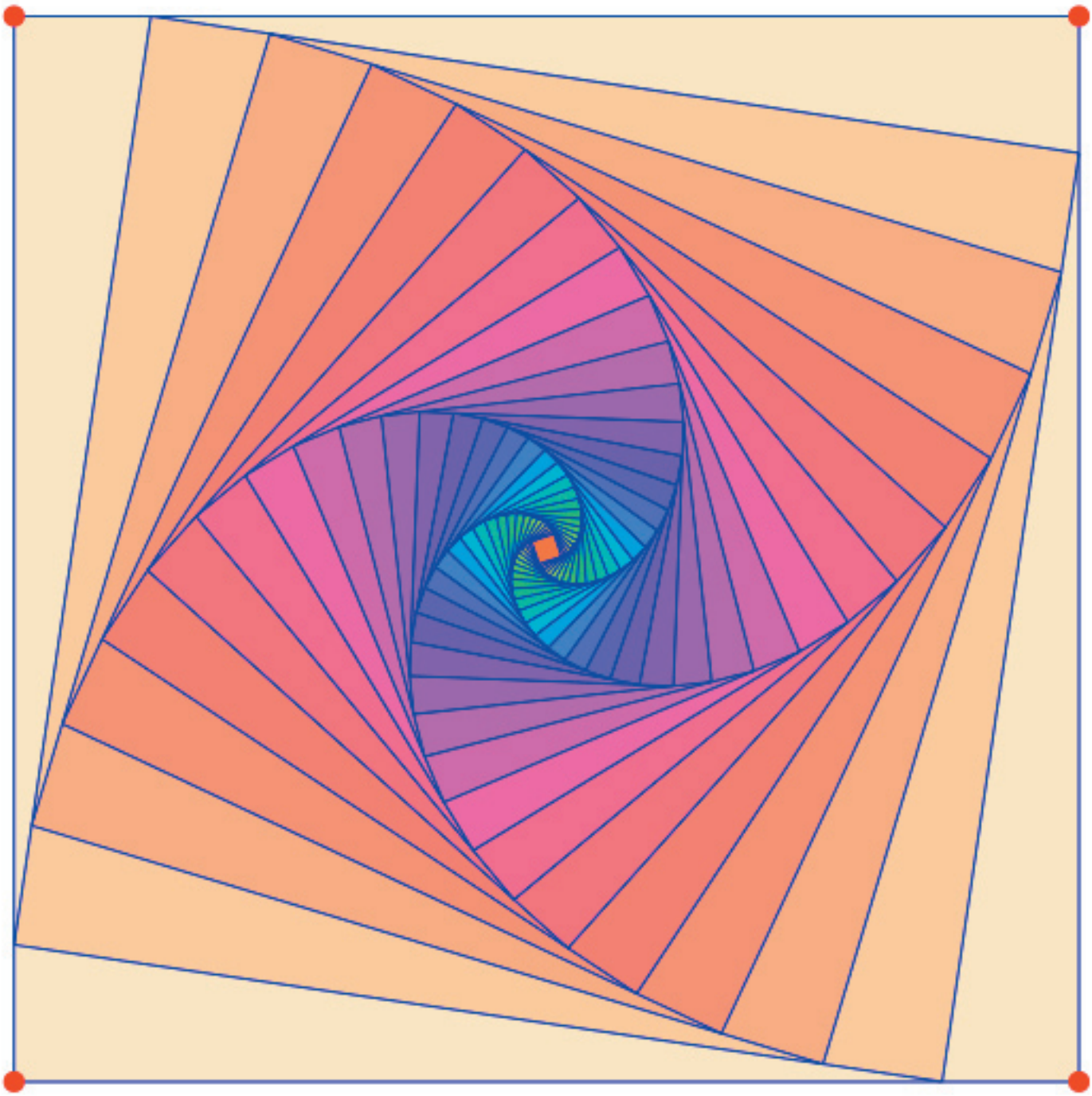


*Fig. 4.16*

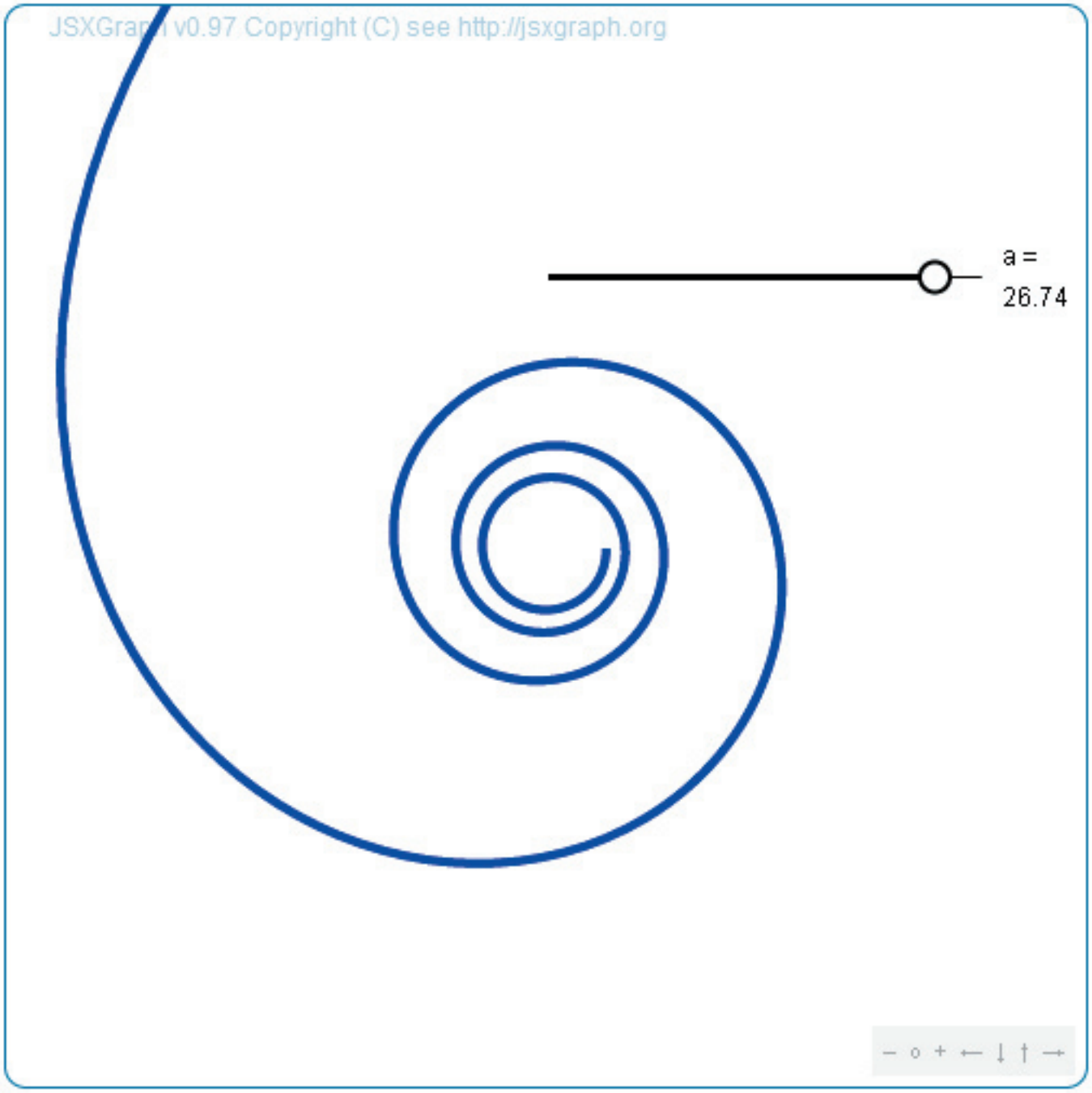



*Fig. 4.17*

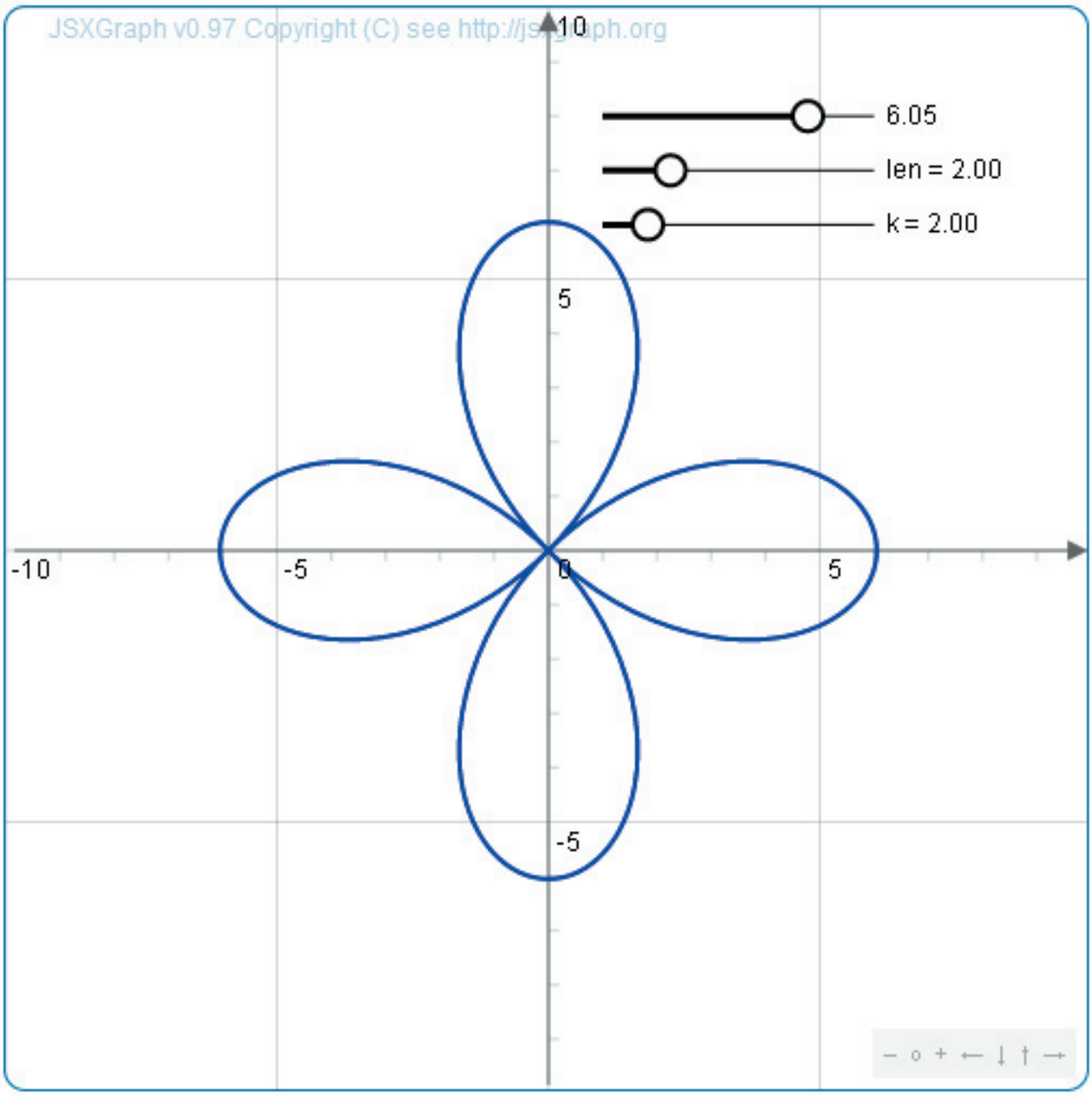

JSXGraph v0.97 Copyright (C) see http://jsxgraph.org
10
6.05
len = 2.00
k = 2.00
5
-10
-5
0
5
-5

*Fig. 4.18*

Conclusiones:

Los conocimientos en matemáticas y programación mejoran la producción multimedia en distintos aspectos. ¿Como y de que manera la mejoran? En el caso de los interactivos didácticos, que explican modelos científicos, el conocimiento matemático es fundamental para la comprensión de conceptos y el desarrollo de las aplicaciones en aspectos compositivos, morfológicos, ópticos, acústicos y cromáticos relacionados a la representación visual, la construcción de interfaces gráficas, funcionalidad, efectos visuales-sonoros y físicas de movimiento en animación estática e interactiva.

En los aspectos compositivos, las matemáticas y la geometría son un conocimiento fundamental para generar sistemas de retículas, diagramación y representación del espacio bidimensional y tridimensional. Ambas relacionadas a la geometría plana y del espacio.

En los aspectos morfológicos, el pensamiento matemático y los lenguajes de programación se pueden convertir en formas visibles: líneas, planos, volúmenes, superficies y relieves. Que a su vez se traducen en vectores, gráficos tridimensionales y gráficos en mapa de bits. En términos cuantitativos se pueden manipular ciertas propiedades de los objetos como: simetría, escala, rotación, proporción, giros, traslaciones, deformaciones, transformaciones y perspectiva.

En los aspectos cromáticos, matemáticamente podemos modificar a nivel de código, propiedades como: Color, tinte, intensidad, brillo, contraste, saturación y luminosidad en la gráfica digital. De la misma manera, podemos también simular ciertos aspectos ópticos: reflexión y refracción de la luz, filtros de imagen como si utilizáramos distintos tipos de lentes; alphas, sharpen, blur, motion blur, en objetos con o sin movimiento. Y también, aspectos relacionados a la acústica: efectos sonoros, timbre, resonancia, intensidad, frecuencias, etc.

Finalmente, a nivel matemáticas y programación, podemos enriquecer nuestra producción multimedia con la aplicación de efectos visuales tales como: efectos con partículas, electricidad y magnetismo, movimientos vibratorios y mecánica de fluidos para generar: explosiones, líquidos, gases, etc. Así también, todo lo relacionado a las físicas de movimiento: Posición, aceleración, desaceleración, fuerza, velocidad, distancia, fricción, resistencia, peso, masa y fuerza de gravedad, aplicando conocimientos en mecánica racional, estática, cinemática (Inversa y directa), cinética, dinámica y mecánica.

En el caso de interactivos multimedia que no explican modelos matemáticos, también es posible aplicar dicho conocimiento para expandir el rango de posibilidades gráficas – interactivas, un control absoluto de la construcción visual, la representación 2D o 3D, efectos visuales y físicas de movimiento que básicamente, enriquecen cualquier animación interactiva, presentación multimedia, carpeta de trabajos, cuento interactivo o sitio web.

Ahora bien, no en todos los casos la producción multimedia emplea matemáticas y programación. Durante este proceso de investigación encontré programadores que desarrollan impactantes proyectos multimedia que no utilizan conscientemente esta ciencia, y conocí matemáticos que no saben nada de programación, pero entienden las posibilidades de su disciplina. Muchos programadores han mejorado su producción multimedia aplicando principios básicos de diseño y composición, arte sonoro, teoría del color, dibujo y conocimientos de animación basados en técnicas tradicionales. En general, una sensibilización basada en las artes y el diseño. Para estos casos específicos, es necesario dedicar más tiempo al estudio de las artes y la programación, que al estudio de las matemáticas.

Por otra parte, los matemáticos tienen una visión más general del problema, para ellos su ciencia prácticamente se puede aplicar a cualquier objeto: piezas arquitectónicas, objetos de uso, diseño industrial, diseño textil, envases, aplicaciones de software, etc. Parte del desarrollo de esta tesis, fue recopilar imágenes que fueran prueba de ello en objetos como obras de arte, jarrones, utensilios o inventos de Arquímedes o Leonardo Da Vinci.

En cuanto a diseño interactivo, los trabajos de la doctora Barbara Kaskosz, Dan Gries, Douglas Ensley, Bianca Valentín y Michael Gerhäuser, dan evidencia de como se aplica la matemática al diseño interactivo en JavaScript y Action Script, la mayoría de sus trabajos, combinan objetos y entornos bidimensionales o tridimensionales, animación interactiva con físicas de movimiento, efectos visuales y otras variables numéricas tomando en cuenta, los conocimientos mencionados en los párrafos anteriores. Recordemos que las matemáticas describen universos armónicos y proporcionados, tienen un lado estético-funcional y un lado científico. Galileo alguna vez dijo frases como: *“El libro de la naturaleza está escrito en símbolos matemáticos”. “Las matemáticas son el alfabeto con el cual Dios ha escrito el universo”.*

Los programadores por otro lado, han creado sus propios universos virtuales a partir de código, recuerdo aquella frase de Abelson y Sussman que nos dice: *“Los programas deben ser escritos para que la gente los lea y sólo incidentalmente, para que las máquinas los ejecuten”,* pues en mi humilde opinión, la programación, es la poesía en código más oculta y sutil, que hace funcionar un objeto interactivo. En el caso de los primeros tres interactivos, fue necesario comprender las fórmulas matemáticas del punto, la línea y el plano por medio de gráficas, antes de producirlos, en otras palabras, sin este conocimiento como antecedente, no habría sido posible desarrollar dichas aplicaciones.

Por lo tanto, en este proceso de investigación y producción, se descubrió que hacer consciente la parte de las matemáticas en el diseño, ayuda al diseñador a racionalizar su producción visual en diferentes aspectos, los cuales ya hemos mencionado. La combinación de ambos conocimientos: matemáticas y programación, brinda herramientas muy poderosas al diseñador multimedia, el cual, prácticamente puede producir cualquier tipo de interactivo por muy complejo que éste sea.

Sin más que agregar en este documento, extiendo una cordial invitación al diseñador de la comunicación visual con esta especialidad, a que involucre estos conocimientos en su producción, para la mejora de su trabajo a nivel profesional.

Gracias.

**Referencias.**


**Autores Varios.** *Conamat. Matemáticas Simplificadas. Aritmética, Álgebra, Geometría y Trigonometría, Geometría Analítica, Calculo Diferencial, Calculo Integral.* Segunda Edición, 2009. **Prentice Hall.**

**Abbot. Abbot. Edwin.** *Planilandia. Una novela de muchas dimensiones*. Torre de viento.

**Arcavi, Abraham.** *The role of visual* representations in the learning of mathematics. *Educational Studies in Mathematics*, 2003 p. 216).

**Barker, P.** *Basic Principles of Human Computer Interface Design.* London, Hutchinson, USA, citado por MORUETA TIRADO, R. (2002)

**Braunstein Roger.** *Action Script 3.0. Bible.* Second Edition. Wiley. 2010.

**Cantoral, R. & Montiel, G.** (2001). *Visualización y pensamiento matemático*. México: Thomson Editores).

**Cordero Benítez, Andrés**. *Aprenda el lenguaje Action Script de Macromedia Flash MX 2004 y Flash 8.* Alfaomega. 2006.

**De Guzmán, Miguel.** (1996): *El papel de la visualización,* Universidad Complutense, Madrid, España.

**Duval, R.** *Representations and Mathematics Visualization.* Representation, vision and visualization: cognitive functions in mathematical thinking. Basic issues for learning. 2002). pags. 311-335.

**Frascara, Jorge.** Diseño gráfico para la gente. Comunicaciones de masa y cambio social. Cita de Jan Van Toorn. Tema: La educación en el diseño. Buenos Aires, Infinito, 2004, pág 239.

**Referencias.**

**Galavis, A. H.** (1992): *Ingeniería del software educativo*, Santa Fé de Bogotá, Ediciones Uniandes, 1992. *Utilización de textos y gráficos en la enseñanza asistida por computador,* Universidad Central de las Villas, Santa Clara, Cuba, 2002.

**Goldstein, Herbert.** *Classical Mechanics.* Pearson Education. 3ra. Edición. 2001.

**Kandinsky.** *Punto y línea sobre el plano. Contribución al análisis de elementos pictóricos.* Paidos Estética.

**López Monroy, Manuel.** *Las alas del deseo tecnológico. Reflexiones en torno al arte, la educación y la tecnología. Ensayo titulado: Los palacios de la memoria digital. Pag 73.*

**Moreno, Rodríguez C.** El diseño gráfico en materiales didácticos. Una investigación sobre el fortalecimiento del aprendizaje educativo. Centro de estudios sociales latinoamericanos, 2009, pág 9 y pág 11.

**Peralta Mariñelarena, José Damián.** *Net-art, Software-art, game-art, arte digital, computable y en línea en el periódo 1993 a 2008.* Tésis de Maestria en Artes Visuales. Pag 66.

**Ramirez, Galarza Ana Irene, Loera, Sienra, Guillermo.** *Invitación a las geometrías no-euclidianas. Facultad de ciencia*s UNAM. 2010. Tema del arte a las matemáticas. Pag 32.

**Riaño, Moncada. Torres, Tovar Alberto. Viviescas, Monsalve Fernándo. Chaparro, Fredy. Ballestas, Rincón Luz Elena.** *El Diseño gráfico. Tema: Diseño Digital.* Universidad Nacional de Colombia. 2004, pág 12.

**Referencias.**

**Ricupero, Sergio.** *Diseño gráfico en el aula. Introducción de Jorge Frascara.* Buenos Aires, Nobuko, 2007, pág 10.

**Roanes Macias Eugenio.** *Nuevas Tecnologías en Geometría. Geometría, Innovaciones, Tecnologías,* ciencias, matemáticas. 1994.

**Thomas, George; Finney, Ross L.,** *Cálculo infinitesimal y geometría analítica*, Editorial Reverté S.A. Barcelona, España. 1975.

**Valentin, Bianca. Gerhäuser, Michael.** Interactive SVG with JSXGraph. Manual de referencia. Pag 3. 2009.

**Vilchis, Luz del Carmen.** *Diseño. Universo de conocimiento. Investigación de proyectos en lacomunicación gráfica.* Pag. 98-101. Centro Juan Acha A.C. Año 2002.

**Vygotsky, Lev.** (1999), *Pensamiento y Lenguaje.* México: Quinto sol.

**Web Sites:**

http://www.alesys.net

http://www.nomaster.com

http://www.flashandmath.com

http://www.crystalab.com

http://jsxgraph.uni-bayreuth