Tema 9

Modelos de Representación en 3D

Ahora que sabemos cómo visualizar información tridimensional sobre superficies bidimensionales, vamos a ver cómo podemos emplear un computador para representar, visualizar y manipular en tiempo real reprentaciones tridimensionales de objetos. La restricción de tiempo real es fundamental para poder utilizar estos modelos en sistemas CAD interactivos, que son la mayoría.

9.1 – Modelos de Representación

A la hora de representar objetos tridimensionales, lo más importante antes de definir cómo representarlos es conocer cuál es el propósito de esa representación. Existen varias aproximaciones diferentes para representar objetos tridimensionales, y el uso de una u otra dependerá de para qué se deba emplear esa representación.

Si necesitamos especificar información acerca de cómo es el objeto por dentro, emplearemos un **modelo de representación sólido**. En este modelo, el computador almacena qué partes del espacio tridimensional están ocupadas por el objeto y qué partes no lo están. Esto nos puede permitir definir cortes con otros objetos y asignar propiedades diferentes a cada una de las partes interiores de los objetos.

Si por el contrario, lo único que nos importa es el aspecto exterior del objeto, y no la información acerca del interior, emplearemos un **modelo de representación de superficie**. Los modelos de representación de superficie permiten conocer cuál es el exterior de un objeto, cuál es el interior, y cuál es la superficie del mismo, pero no permiten establecer propiedades sobre el contenido del mismo. Lo único que importa es definir bien el **contorno del objeto**.

El tipo de representación que empleemos también se ve condicionado por el tipo de interacción que queramos establecer con los objetos representados. Si queremos ser capaces de interactuar con el modelo en tiempo real (movernos por la escena virtual, cambiar los objetos de posición, etc.), debemos buscar un sistema de representación que lo permita.

El problema de los modelos de representación sólidos es que requieren una mayor capacidad de cálculo computacional, por lo que normalmente, en informática gráfica en tiempo real se emplean modelos de representación de superficies. Además, dado que los objetos a representar pueden presentar cualquier forma (cubos, esferas, poliedros, superficies cónicas) debemos encontrar una manera unificada de representar cualquier objeto.

Es posible representar los objetos mediante superficies analíticas. Es decir, mediante superficies definidas por ecuaciones más o menos complejas (como cuádricas o ecuaciones paramétricas). El problema de esta representación es que es muy complicado encontrar ecuaciones para representar las formas de los objetos cotidianos, ya que evidentemente éstos no suelen seguir la forma de una ecuación, y para obtener una buena representación habría que disponer de una biblioteca de ecuaciones inmensa.

Por eso, la manera en que habitualmente se representan los objetos tridimensionales dentro de un computador para su visualización y edición en tiempo real es mediante la definición de **superficies poligonales**. De esta forma, todos los objetos se representarán de forma unificada mediante un conjunto de caras planas, que si se hacen de un tamaño suficientemente pequeño serán capaces de representar formas geométricas complejas (incluso superficies curvas) sin que se note la diferencia. A este modelo de representación se le llama **modelo de representación poliédrico o de superficies poligonales**.

9.1.2 – El Modelo de Superficies Poligonales

El modelo de representación de superficies poligonales consiste en representar los objetos como superficies formadas por **caras planas unidas entre sí**. Lo único que se exige para que la representación sea consistente es que la superficie no se intersecte consigo misma. A veces tenemos superficies abiertas, como puede ser un plano finito o una semiesfera. Ello no supone ningún problema, a no ser que intentemos analizar el contenido o el volumen interior, ya que éste no estará definido.

La ventaja de las caras planas es que siempre tienen la misma forma y no necesitamos complejas ecuaciones para representar el objeto puesto que sólo hemos de almacenar la posición de los vértices de las caras planas, y anotar qué caras se forman con esos vértices. La desventaja es que hemos de crear muchas caras planas si queremos aproximar bien la forma de los objetos. Si la representación se hace con pocas caras, la curva no parecerá una curva, pero si usamos demasiadas caras, es posible que el ordenador tenga problemas para dibujar tal cantidad de objetos y el rendimiento se degrade.

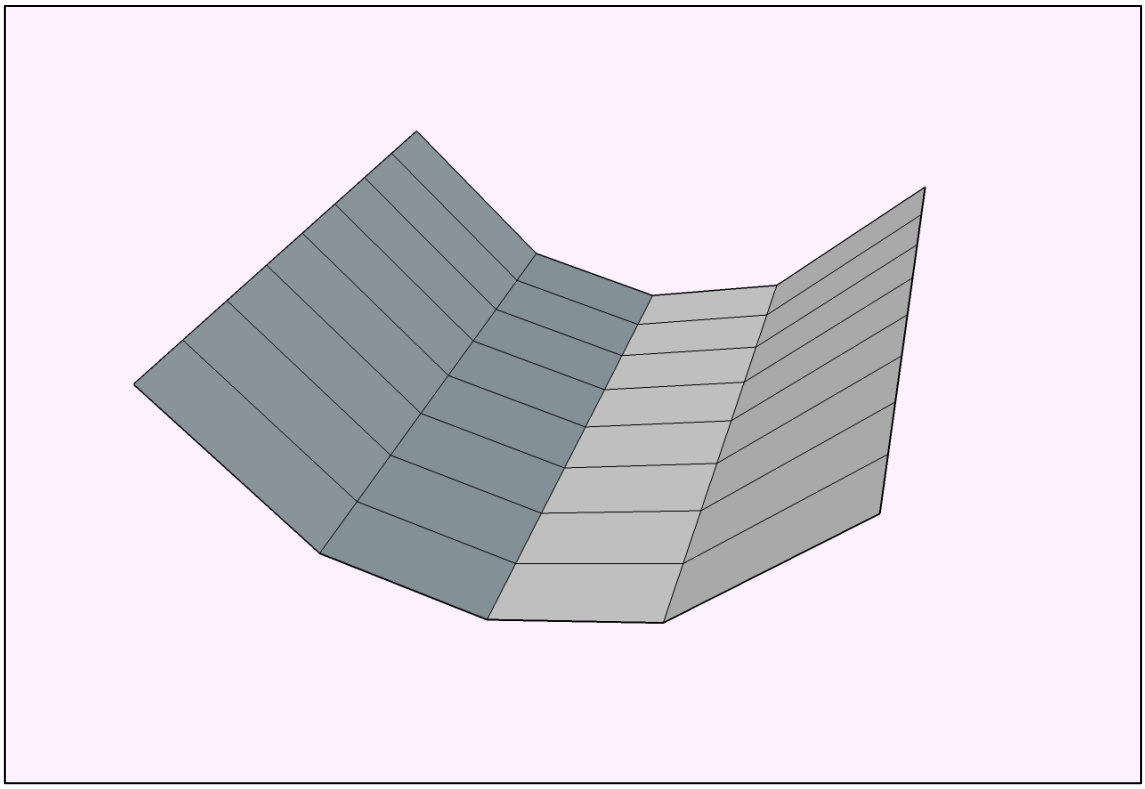

Figura 9.1 – Representación poligonal de una superficie curva.

9.1.2.1 – Primitivas de Poligonales de Dibujado

Las primitivas poligonales son grupos de polígonos que se describen de forma conjunta para ahorrar espacio de almacenamiento y coste de visualización en tiempo real, razón por la cual son ampliamente utilizadas. Si una serie de caras planas comparten vértices, no es lo mismo almacenarlas como si fueran caras aisladas (repitiendo las coordenadas de los vértices compartidos cada vez), que almacenarlas de modo que las coordenadas de los vértices compartidos se guarden una sóla vez en la memoria del ordenador.

Las caras planas empleadas en los sistemas de representación en tiempo real son casi siempre triángulos, aunque se admite también la utilización de cuadriláteros. Es por ello que las primitivas poligonales de dibujado son estructuras formadas en torno a triángulos y cuadriláteros.

Las **primitivas poligonales** de dibujado más utilizadas son:

• **Tira de cuadriláteros** (*quad strip*): los primeros cuatro vértices definen el primer cuadrilátero, y cada nuevo par define otro cuadrilátero más, formado por éste par de vértices y el anterior par. Podemos ver que este tipo de primitiva ahorra, respecto de la especificación de cuadriláteros aislados, casi la mitad de espacio, ya que cada cuadrilátero se forma especificando 2 vértices en lugar de 4. Sin embargo, tiene el inconveniente de que no se garantiza que en cada cuadrilátero los vértices sean coplanarios.

• **Tira de triángulos** (*triangle strip*): con los tres primeros puntos se construye un triángulo y los demás se forman añadiendo sucesivos puntos. Cada nuevo triángulo se forma con los tres últimos vértices añadidos, de tal forma que con N puntos se obtienen $N-2$ triángulos. El ahorro es significativo con respecto a especificar los triángulos aisladamente, ya que se ahorra casi dos terceras partes de espacio en memoria (pues con cada nuevo vértice especificamos un triángulo, que de haber sido especificado aisladamente requeriría de tres vértices). La ventaja de las tiras de triángulos es que un triángulo siempre define un plano, mientras que los vértices de un cuadrilátero pueden no ser coplanarios y por tanto, no siempre definen una cara plana.

• **Abanico de triángulos** (*triangle fan*): se da un primer punto y luego el resto siguiendo un abanico. Aparecen $N-2$ triángulos formados por el primer vértice y los vértices i, $i+1$ (para aquellos valores de i que sean diferentes de 1). Se emplea para conseguir formas imposibles de lograr con la tira de triángulos. Esta primitiva, al igual que la anterior, tiene la ventaja de que no es necesario comprobar la coplanariedad.

• **Malla rectangular** (*rectangular mesh*): se utiliza directamente una matriz de n por m vértices. Se emplea en pocas ocasiones.

• **Malla triangular** (*triangular mesh*): utiliza una malla de vértices en forma de triángulo. Apenas se emplea.

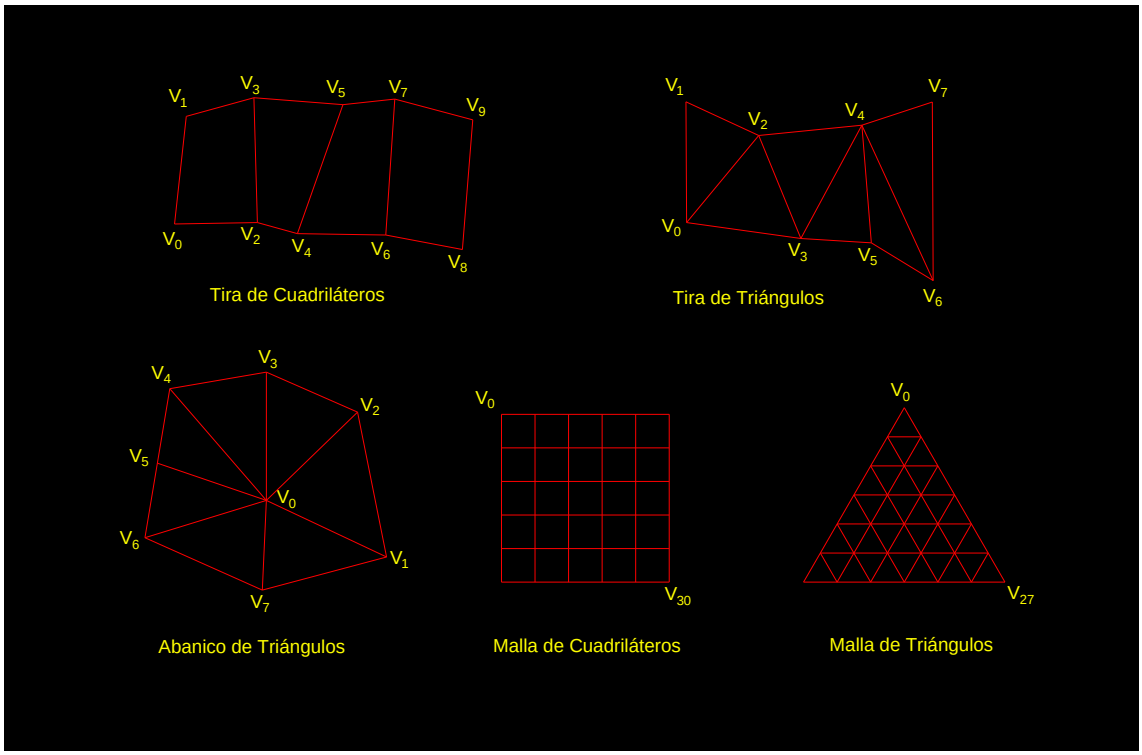

Figura 9.2 – Primitivas de dibujado más habituales.

9.1.2.2 – Iluminación y Texturizado

El modelo de superficies poligonales utiliza las primitivas anteriores para almacenar las posiciones de los vértices y definir las caras de los objetos, de modo que podememos saber qué caras y vértices definen cada objeto. Lo habitual es disponer de una lista de vértices numerados (normalmente de 0 a $n-1$) en la que se guardan las coordenadas (x, y, z) de cada uno de los vértices, y otra lista de caras en la que se especifica qué vértices de la lista de vértices forman cada cara.

Sin embargo, esta no es la única información que debemos guardar para representar la información tridimensional y poder visualizar la figura. Dado que el aspecto visual de los objetos es esencial debemos guardar información acerca del **color de cada cara**. Normalmente el color se almacena como un vector de cuatro componentes con la convención RGBA (rojo, verde, azul, alfa – *red*, *green*, *blue*, *alpha*), donde las tres primeras componentes son números expresando la intensidad de color rojo, verde y azul respectivamente, y el cuarto número indica la transparencia de esa cara. Si la cara es totalmente transparente, el valor de alfa será 0, y si es opaca, tendrá el valor máximo. El

rango de estas componentes puede ser diferente según cada sistema, aunque lo normal es que sean números enteros entre 0 y 255, o números reales entre 0 y 1.

Si bien la representación del color mediante vectores RGBA es prácticamente universal, esta representación en la que se le asigna a la cara un único color uniforme suele ser demasiado sencilla y poco realista. A este tipo de iluminación se le llama **sombreado plano**, y prácticamente no se utiliza nunca puesto que no da un aspecto que evoque sensación de profundidad en los objetos.

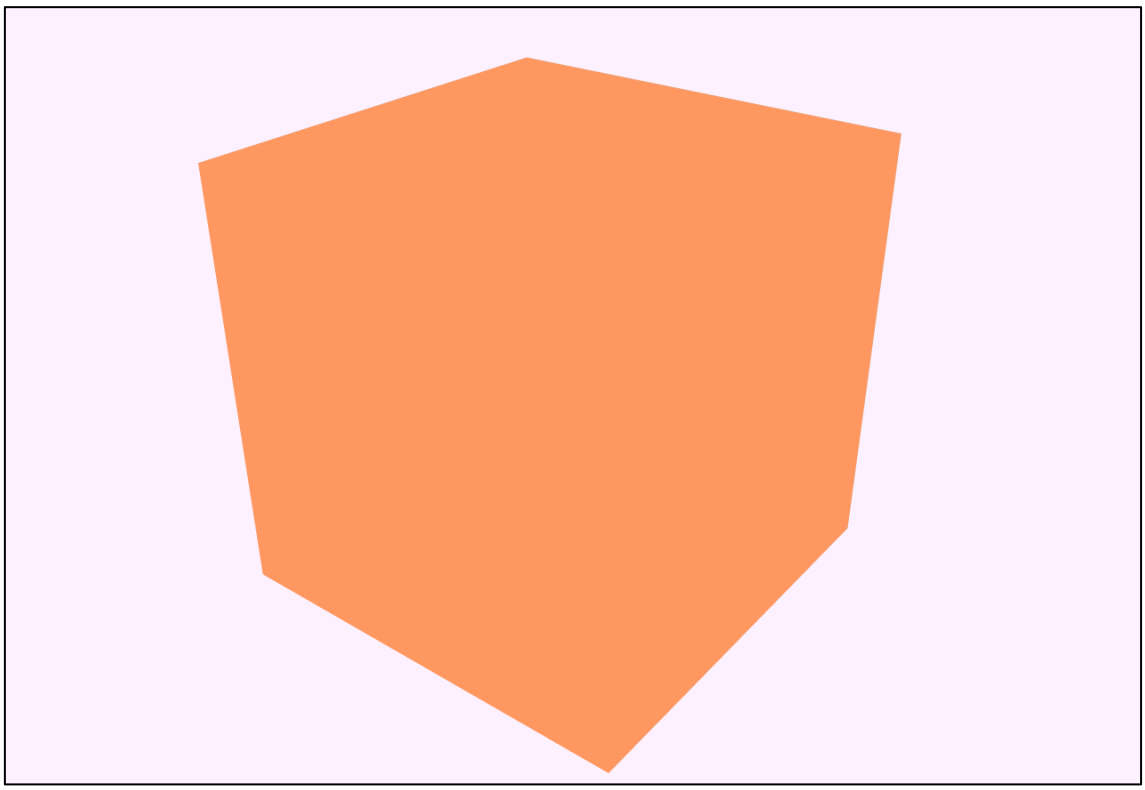

Figura 9.3 – Sombreado plano.

Lo que se hace habitualmente para iluminar los objetos es almacenar en cada vértice (en el vértice, no en la cara) una serie de propiedades acerca de cómo se refleja en ese punto la luz incidente, lo cual permitirá luego calcular de qué color se ve cada pixel de la cara (que contenga a ese vértice) en función del color que se calcule para cada uno de sus vértices. Dicho cálculo se realiza mediante una interpolación lineal, y al proceso se le llama **sombreado suave**.

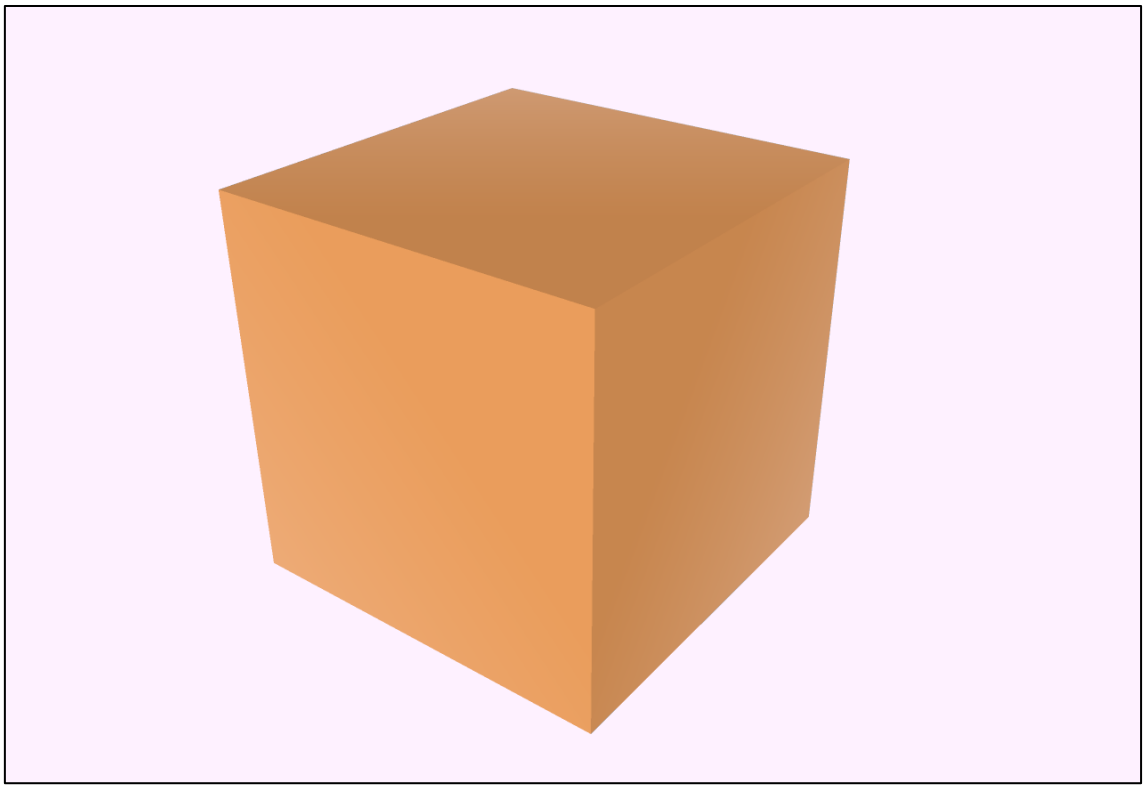

Figura 9.4 – Sombreado suave.

El cálculo del color de cada vértice se realiza mediante una serie de ecuaciones que tienen en cuenta cómo está orientada la cara, cuánta luz incide sobre ella, qué dirección tiene la luz incidente, dónde esté la cámara, y sobre todo qué propiedades físicas tiene ese vértice a nivel de reflexión de la luz.

Hay objetos que son capaces de reflejar la luz de modo casi perfecto como los metales, o los espejos, de modo que toda la luz que incide rebota con un cierto ángulo en la superficie del objeto. Esto se conoce como **material especular**.

Hay otros objetos, sin embargo, que absorben la luz o que la dispersan enormemente cuando ésta incide sobre ellos, de modo que la luz incidente rebota en múltiples direcciones. A esto se le llama **material difuso**.

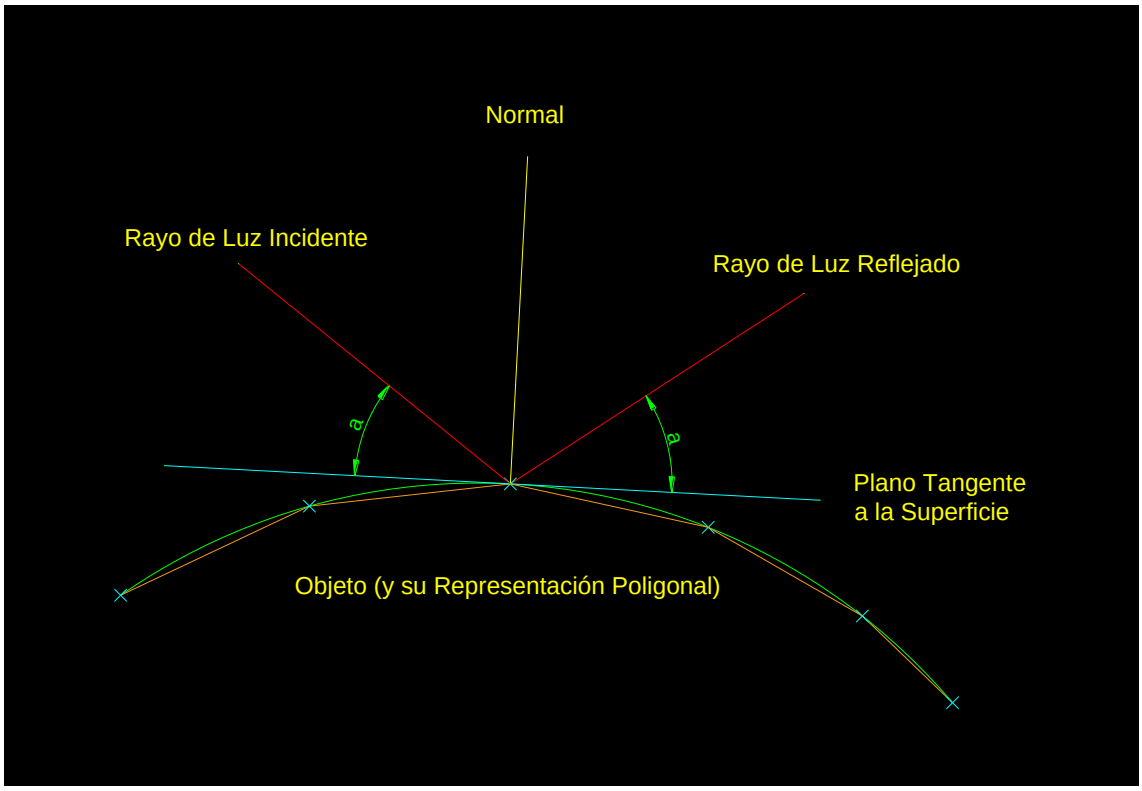

Figura 9.5 – Material especular.

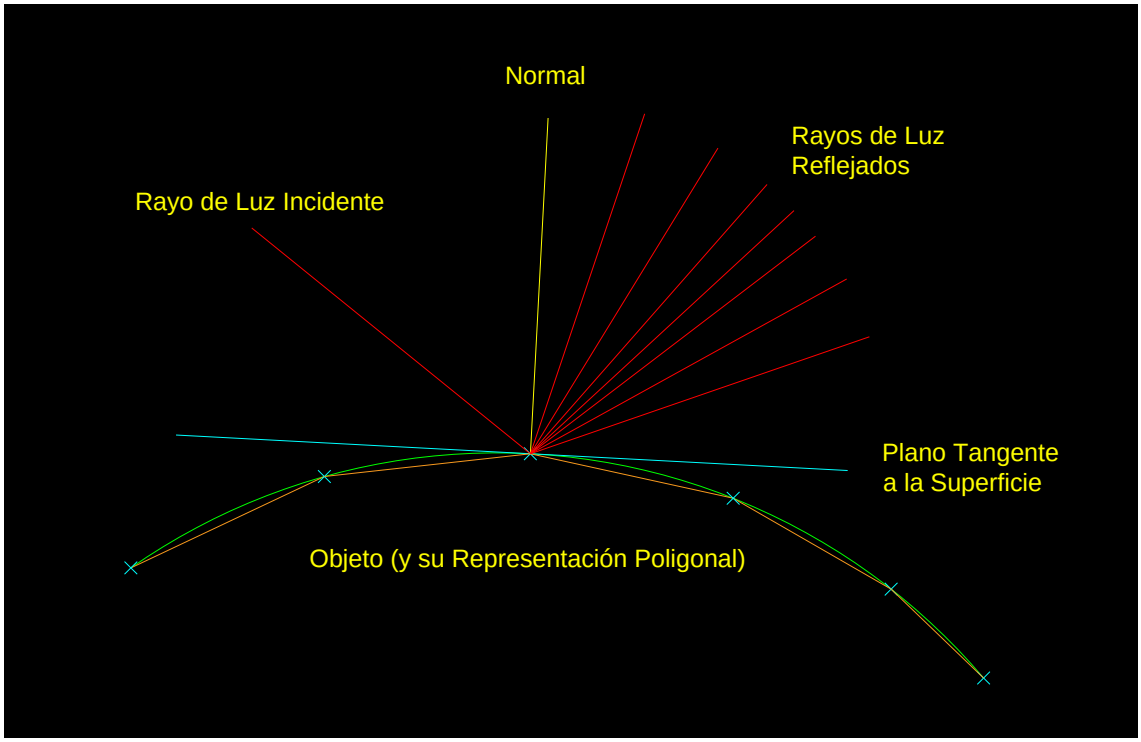

Figura 9.6 – Material difuso.

Además, cada objeto es capaz de reflejar unas longitudes de onda diferentes, y absorber otras. Dependiendo de qué longitudes de onda (gamas de colores) refleje o absorba, y de cómo lo haga (dispersándola o reflejándola especularmente), el color resultante será diferente. Los materiales no son siempre o difusos o especulares puros, suelen combinar propiedades de ambos tipos, y hacerlo de forma diferente para cada color.

A toda esta información se le llama **material**, y suele especificarse en forma de vectores RGBA que definen cómo refleja cada vértice de cada cara cada uno de los colores primarios (rojo, verde, azul) de la luz. Dependiendo de lo complejo que queramos hacer este proceso, dispondremos de un tipo de iluminación más o menos realista. Normalmente se realizan simplificaciones para poder realizar el cálculo en tiempo real.

Además, se debe guardar, para cada vértice, cual es la dirección **normal** (la dirección que representa la perpendicular a la superficie en ese punto). Esa información es de vital importancia para poder realizar el cálculo de iluminación, puesto que la dirección en la que rebota la luz depende del vector normal a la superficie en cada punto.

A este proceso que permite asignar colores a los pixeles de cada cara, se le llama *shading*, habitualmente traducido como **sombreado**. Este nombre da lugar a confusión, puesto que no es el proceso empleado para calcular la sombra que produce de un objeto, si no el proceso para calcular el color de cada pixel del objeto.

La iluminación se suele combinar con un proceso denominado **texturizado**, que consiste en combinar la información de color calculada en el proceso anterior con una imagen (suele ser una fotografía) que represente el aspecto detallado de la cara del objeto. La información de la textura nos da el aspecto visual, y la información de iluminación nos proporciona sensación de profundidad y realismo. La manera de combinar ambos tipos de información puede variar, desde emplear sólo la información de la textura hasta ponderarlas o multiplicarlas entre sí.

El proceso de texturizado es, en en realidad, la proyección de una imagen (normalmente estática como una fotografía) sobre una cara plana, por lo que para especificar qué parte de la imagen proyectada, llamada **textura**, se proyecta sobre cada parte de la cara plana, debemos almacenar para cada vértice unos números, llamados coordenadas de textura, que nos indican cómo hacer ese mapeo o proyección.

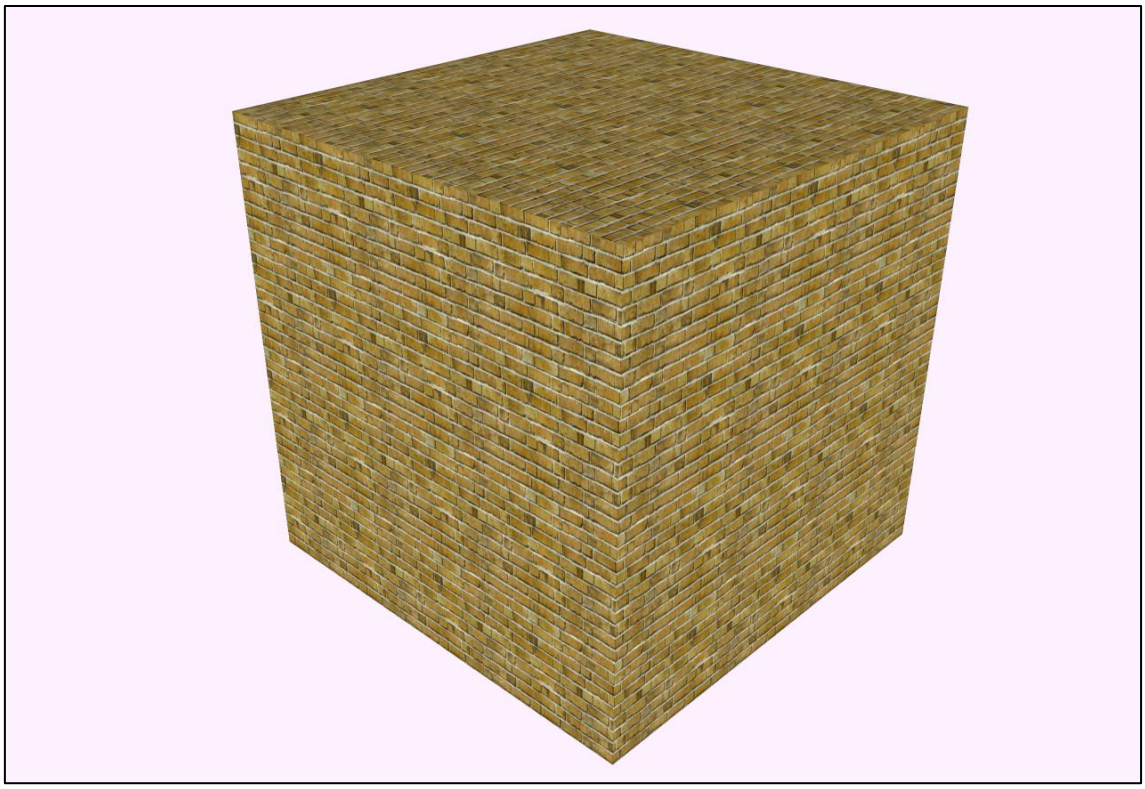

Figura 9.7 – Texturizado.

9.2 – La Proyección Perspectiva en Tiempo Real

A la hora de representar objetos tridimensionales en un computador no necesitamos realizar proyecciones, ya que el computador es capaz de almacenar la información de las componentes x, y, z de todos los vértices de todas las caras poligonales del objeto (sea cual sea el tipo de primitiva de dibujado empleado).

Sin embargo, es de poca utilidad almacenar información sin poder visualizarla y manipularla de forma interactiva. Por ello, dado que deberemos representar la información tridimensional en una pantalla bidimensional para que una persona sea capaz de visualizarla y manipularla, deberemos emplear alguna de las proyecciones explicadas con anterioridad.

En este caso, lo que buscamos es una proyección que nos genere una sensación realista del objeto, para que la representación del mismo se parezca a la real. Aunque las proyecciones más semejantes a la visión del ojo humano son las proyecciones esféricas, la proyección perspectiva representa una aproximación suficientemente buena en casi todos

los casos, y además tiene la ventaja de que es computacionalmente más sencilla de implementar de modo algorítmico que una proyección no plana.

Es por ello que, para la visualización en tiempo real de objetos tridimensionales, la proyección más utilizada es la proyección perspectiva.

Sin embargo, la proyección perspectiva que se utiliza en los programas de modelado tridimensional y en los simuladores y sistemas de realidad virtual, es una **proyección perspectiva acotada**. La proyección perspectiva teórica emplea un plano de proyección infinito, sobre el que es proyectado todo el semiespacio que queda al lado contrario del plano con respecto al centro de proyección.

Esto es, evidentemente, poco práctico, por lo que se deben buscar alternativas más eficientes para que podamos realizar la proyección perspectiva en tiempo real.

9.2.1 – La Pirámide de Visión

Por razones de practicidad y eficiencia conviene definir el plano de proyección de modo que no sea infinito, por lo que la proyección se acota a una cierta sección del plano de proyección. Por ello, en lugar de utilizar un plano, utilizaremos una sección rectangular de dicho plano, y todo lo que se proyecte fuera de ese rectángulo, sería descartado.

Pero además, es necesario establecer un límite de distancia, más allá del cual los objetos dejen de proyectarse. Esto permite que los objetos muy lejanos, que aparecerían de un tamaño casi despreciable en la vista proyectada, sean descartados de modo que el cálculo de la proyección sea mucho más eficiente. Hemos de tener en cuenta que, por cada objeto que el computador deba proyectar, éste emplea una cierta cantidad de tiempo. Si evitamos proyectar objetos que prácticamente no se van a ver, podremos realizar la proyección más rápidamente.

Dado que el plano de proyección suele estar a una cierta distancia del centro de poryección, los objetos que se sitúen entre el centro de proyección y el plano de proyección tampoco serán proyectados. Al plano de proyección se le suele llamar **plano cercano**, dado que define una distancia mínima con respecto al centro de proyección, por debajo de la cual no proyectamos los objetos.

Con estas consideraciones, la proyección perspectiva se convierte en una proyección perspectiva acotada por seis planos:

- El **plano lejano** (*far plane*), que define el límite de distancia más allá del cual los objetos no serán visibles.
- El **plano cercano** (*near plane*), que coincide con el plano de proyección (es decir, con la pantalla), y que define el límite de distancia por debajo del cual los objetos no serán visibles.
- El **plano derecho** (*right plane*), que define el límite derecho de la proyección perspectiva.
- El **plano izquierdo** (*left plane*), que define el límite izquierdo de la proyección perspectiva.
- El **plano superior** (*top plane*), que define el límite superior de la proyección perspectiva.
- El **plano inferior** (*bottom plane*), que define el límite inferior de la proyección perspectiva.

El centro de proyección y los seis planos definen una pirámide truncada que se llama **pirámide de proyección**, **pirámide de visión**, o *viewing frustum*. La pirámide de visión es una pirámide truncada de base cuadrada o rectangular (dependiendo de cómo elijamos los planos).

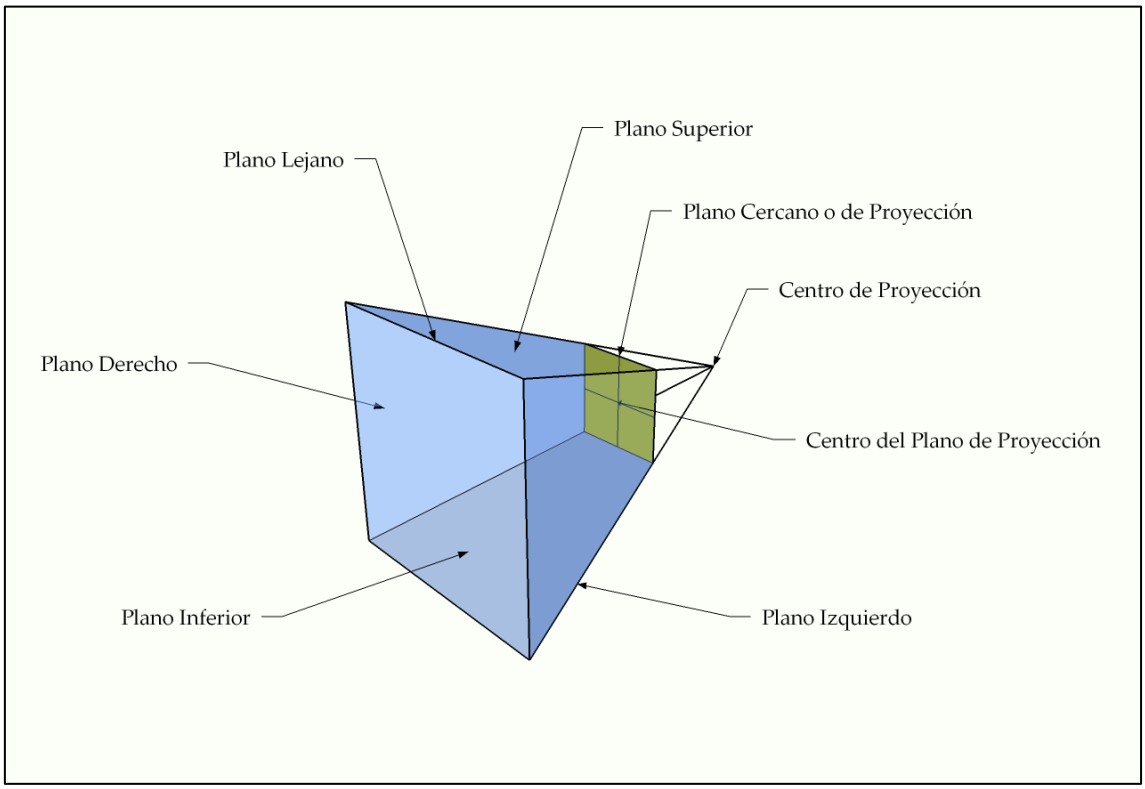

Figura 9.8 – La pirámide truncada de proyección.

Se puede comprobar que este tipo de proyección es el mismo que realiza una cámara de fotografía o una cámara de video. Por ello, al centro de proyección se le llama **posición de la cámara**, porque es el punto desde el que se observa el mundo. La analogía con la fotografía es prácticamente total, por lo que muchos de los conceptos empleados se utilizan también en fotografía.

En realidad, para definir los planos lejano, cercano, derecho, izquierdo, superior e inferior, sólo necesitamos definir seis distancias. Tomando el centro del plano de proyección como referencia, definimos las distancias:

- **lejana** (*far*), que define la distancia entre la posición de la cámara (el centro de proyección) y el plano lejano, pasando por el centro del plano de proyección.
- **cercana** (*near*), que define la distancia entre la posición de la cámara (el centro de proyección) y el centro del plano de proyección.
- **derecha** (*right*), que define la distancia entre el centro del plano de proyección y el límite derecho del plano de proyección.
- **izquierda** (*left*), que define la distancia entre el centro del plano de proyección y el límite izquierdo del plano de proyección.
- **superior** (*top*), que define la distancia entre el centro del plano de proyección y el límite superior del plano de proyección.
- **inferior** (*bottom*), que define la distancia entre el centro del plano de proyección y el límite inferior del plano de proyección.

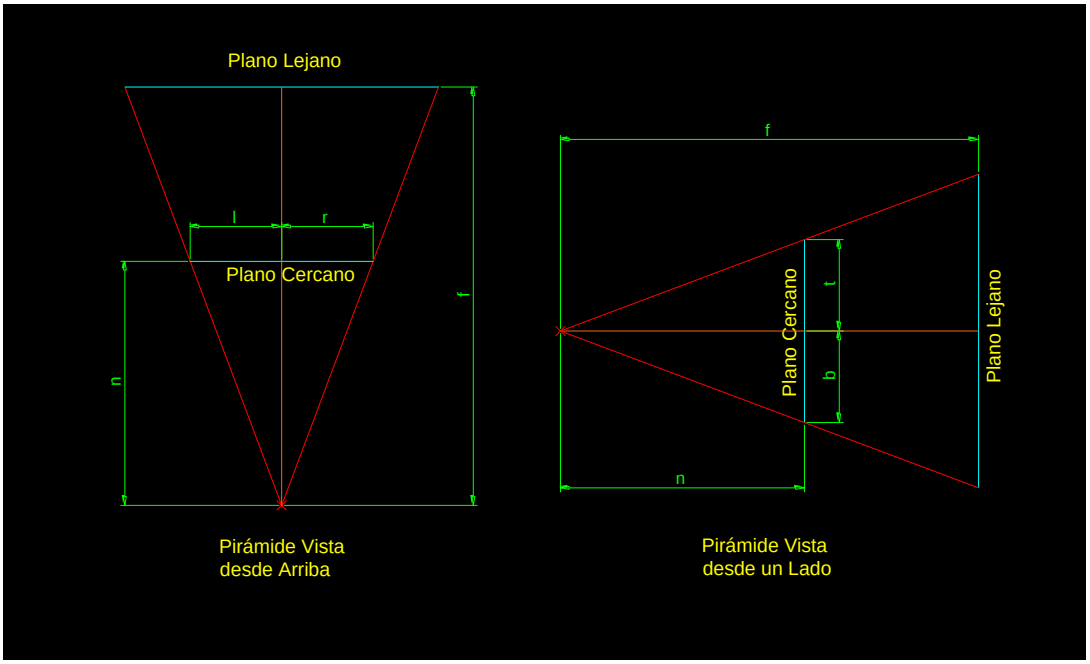

Figura 9.9 – Parámetros de la pirámide de proyección.

En función de cómo elijamos las distancias derecha, izquierda, superior e inferior, la sección del plano de proyección formada tendrá un aspecto rectangular diferente (incluso cuadrado). Si los valores derecho e izquierdo, y superior e inferior son iguales dos a dos, la pirámide es simétrica. Si no lo son, la pirámide es asimétrica. Lo habitual es que el frustum sea simétrico, pero puede haber situaciones en los que deseemos que no lo sea.

A la razón entre el ancho de la sección del plano de proyección y el alto del mismo se le llama **relación de aspecto**, o *aspect ratio*.

Si llamamos n, f, r, l, t, b a las distancias cercana, lejana, derecha, izquierda, superior e inferior, y ar a la relación de aspecto, entonces ésta se puede calcular como:

$$ar = \frac{(r+l)}{(t+b)}$$

ya que $(r+\ l)$ es el ancho de la sección del plano, y $(t+b)$ es el alto de la misma.

La relación de aspecto define la proporción ancho/alto que debe tener la pantalla de nuestro ordenador para coincidir exactamente con la proyección realizada. Los valores típicos son 1:1 (formato cuadrado), 4:3 (proporción habitual de las cámaras de fotos, y de los monitores de hace algunos años), 3:2 (proporción habitual de las cámaras réflex de fotografía) y 16:9 (proporción habitual de los televisores actuales, de la emisión de televisión HDTV, y de la UHDTV 4K).

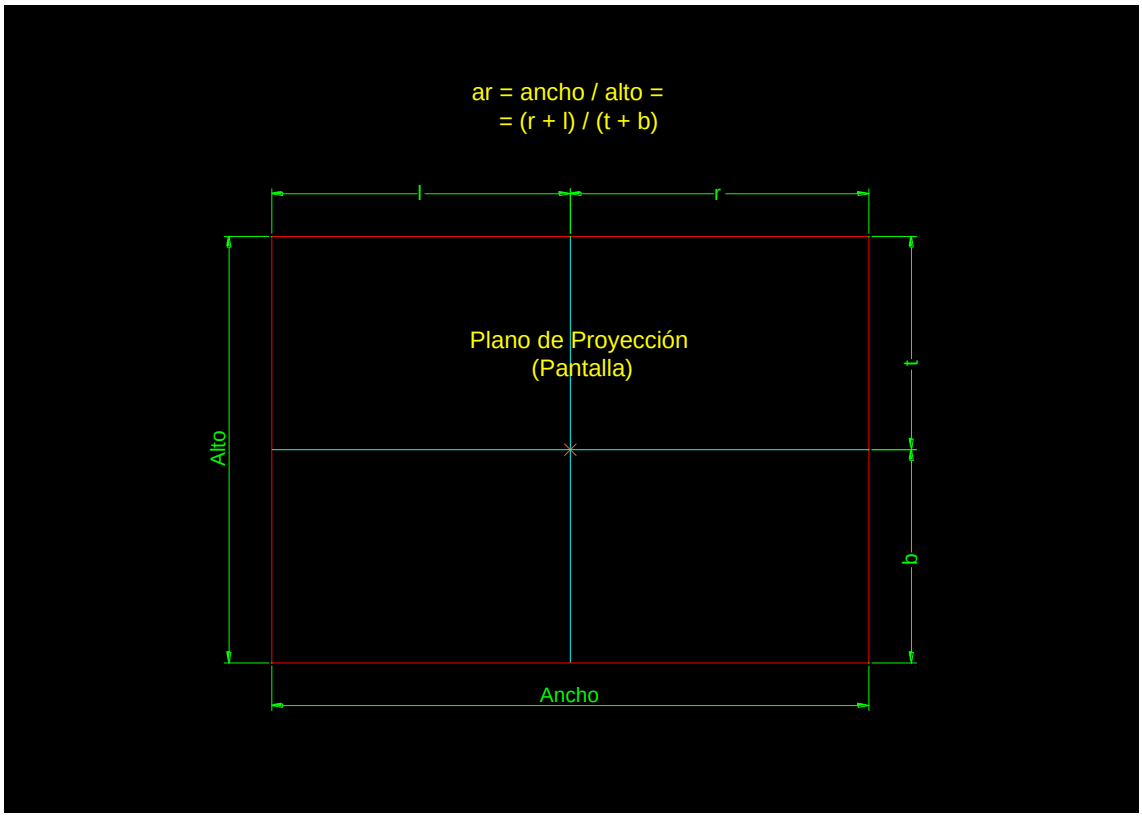

Figura 9.10 – La relación de aspecto.

Además, la medida de las distancias cercana, derecha, izquierda, superior e inferior definen el campo visual de la proyección perspectiva. El campo visual, denominado habitual **FOV** (*field of view*) es el ángulo que abarca la proyección perspectiva acotada. Normalmente se define un campo visual horizontal, y un campo visual vertical.

Cuanto mayor sea el FOV, mayor es la porción del mundo abarcada por la proyección. Por desgracia, a mayor FOV, mayor deformación se produce de los objetos situados en los laterales del campo visual con respecto a una proyección esférica (que sería la más natural para el ojo). Por eso no se suelen emplear FOVs superiores a 90º, siendo habitual que se sitúen en un rango entre 40º y 60º. Los FOVs muy pequeños se corresponden con el zoom de las cámaras de fotos, y los FOVs muy grandes con el gran angular.

El campo visual horizontal (FOVh) y vertical (FOVv) se puede calcular a partir de las distancias que definen la pirámide de visión. Suponiendo que el frustum es simétrico (y por tanto $r = l$, y $t = b$):

$$FOV_h = 2 \cdot \text{atan}\left(\frac{r}{n}\right)$$
$$FOV_v = 2 \cdot \text{atan}\left(\frac{t}{n}\right)$$

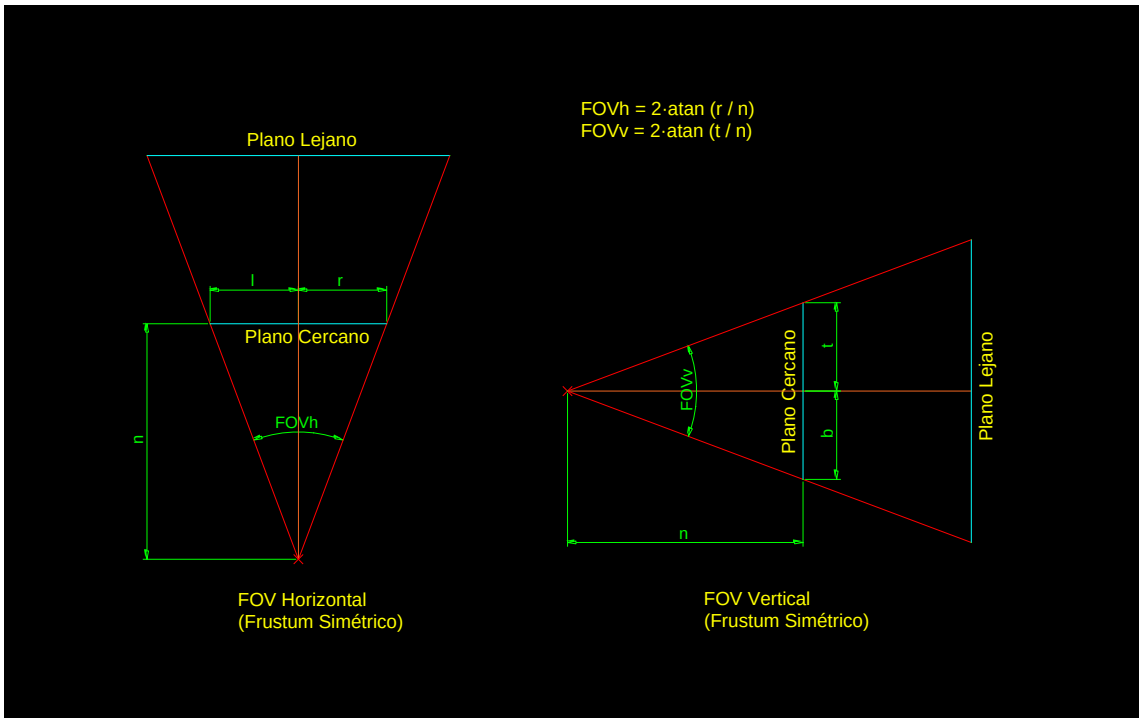

Figura 9.11 – Campo visual con frustum simétrico.

Si la pirámide de visión no es simétrica (o bien en horizontal o bien en vertical) el FOV se calcula de un modo similar, pero sumando los semiángulos que se forman con la línea que une el centro de proyección con el centro del plano de proyección:

$$FOV_h = \text{atan}\left(\frac{r}{n}\right) + \text{atan}\left(\frac{l}{n}\right)$$
$$FOV_v = \text{atan}\left(\frac{t}{n}\right) + \text{atan}\left(\frac{b}{n}\right)$$

Podemos comprobar fácilmente que si $r = l$, y $t = b$, estas fórmulas se convierten en las fórmulas que proporcionan el FOV cuando el frustum es simétrico.

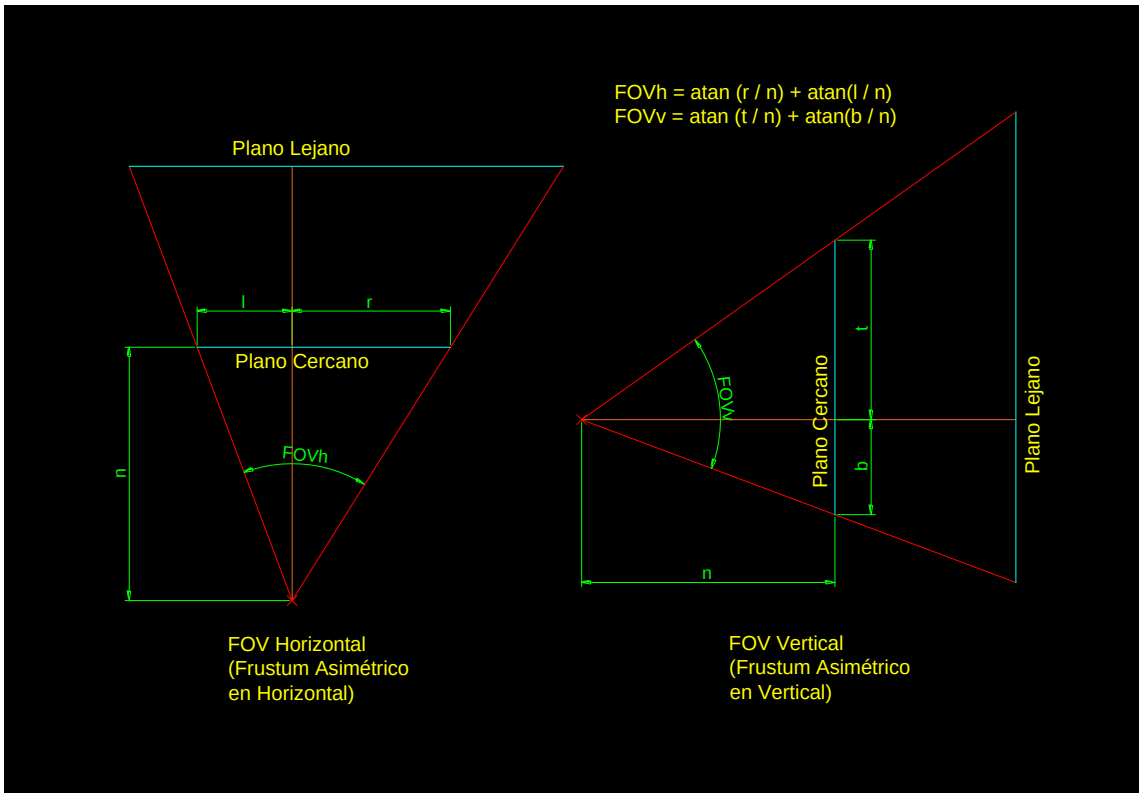

Figura 9.12 – Campo visual con frustum asimétrico.

9.2.1.1 – Sistema del Mundo, Sistema de Objeto y Sistema de la Cámara

Cuando realizamos una representación tridimensional de una serie de objetos, y los visualizamos mediante una proyección, normalmente perspectiva, debemos establecer un sistema de referencia absoluto, al que llamamos **sistema de coordenadas del mundo**, a partir del cuál podemos situar la cámara y los objetos.

Este sistema de coordenadas será inalterable y proporcionará una referencia para poder colocar todos los elementos de nuestro mundo virtual. Sin embargo, lo normal es modelar los diferentes objetos en coordenas relativas al propio objeto. Por ejemplo, si modelamos una esfera, lo normal es definir la geometría de sus caras (su aproximación poligonal) en función del centro de la esfera, ya que, cuando definimos la geometría no sabemos en qué parte del mundo se va a situar ese objeto. Además, esto nos permite reutilizar modelos de una escena virtual a otra o de un computador a otro. Como cada ordenador, o cada escena virtual pueden emplear sistemas de coordenadas del mundo diferentes, lo lógico es definir la geometría de los objetos en coordenadas relativas al centro del objeto, no en coordenadas del mundo, ya que cuando modelamos el objeto no conocemos el sistema del mundo. A este sistema de coordenadas local al objeto se le llama

sistemas de coordenadas del objeto. La elección de cuál es el centro de coordenadas del mismo, y cuáles son las direcciones de los ejes, es arbitraria, aunque se intenta buscar puntos, y direcciones representativas. En el ejemplo de la esfera, lo lógico es que el centro de coordenadas de ese objeto sea el centro de la esfera.

Además, cuando representamos la información tridimensional de una escena virtual, debemos hacerlo desde un punto de vista (el punto de observación). Ese punto de vista es lógicamente el centro de proyección. Al sistema de coordenadas que está situado en el centro de proyección, se le llama **sistema de coordenadas de la cámara** o **sistema de coordenadas del ojo**. Es importante conocer este sistema, puesto que, aunque nosotros situaremos los objetos y la cámara en puntos definidos relativos al sistema del mundo, las ecuaciones de la proyección perspectiva se simplifican cuando se toma como centro el sistema del ojo.

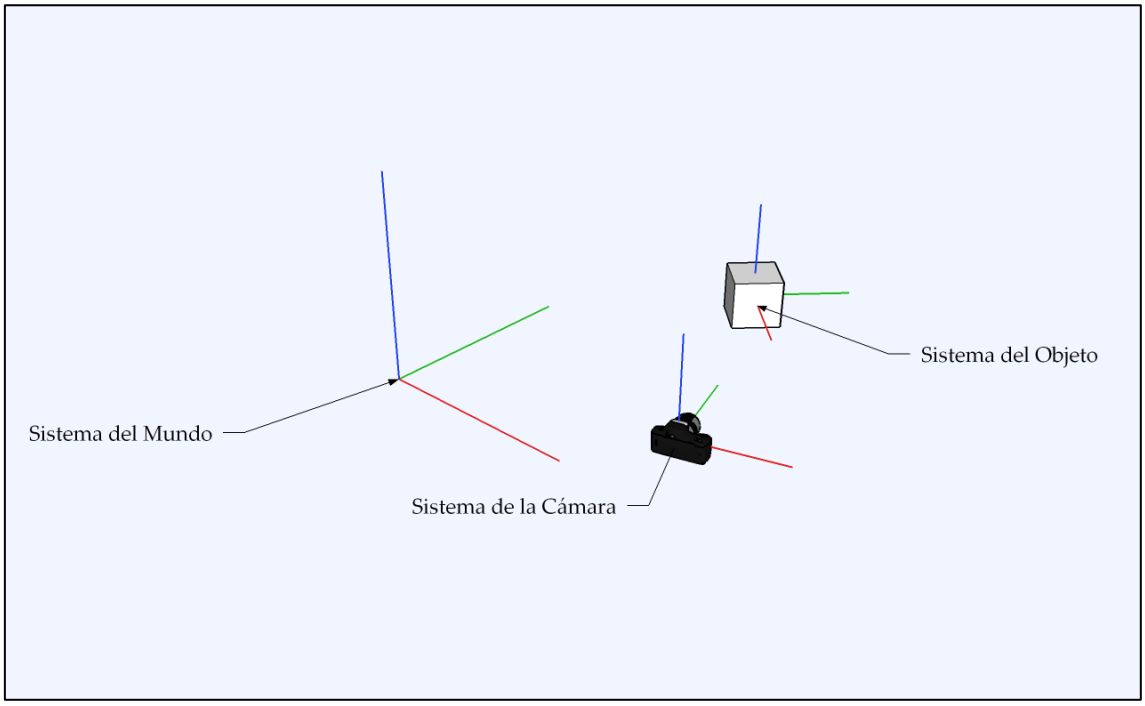

Figura 9.13 – Sistemas de coordenadas de una representación tridimensional.

9.2.1.2 – Posicionamiento de la Cámara

Como el lector probablemente ya haya deducido, no basta con especificar dónde se situa el centro de proyección para definir cómo es la visualización de los objetos de la escena.

Además de especificar en **qué punto se situa la cámara** (que equivale a describir dónde está situado el centro de proyección), debemos especificar dónde está situado el plano de proyección (que equivaldría a especificar **hacia dónde mira la cámara**).

En lugar de dar la orientación del plano de proyección especificando los ángulos que forma con los ejes de coordenadas del mundo, lo que se suele hacer es especificar dos vectores que lo describien de manera unívoca: el vector de orientación de la lente, y el vector vertical que define cuál es la dirección que consideramos hacia arriba en la proyección.

Por tanto, para especificar cómo está situada y orientada la cámara de la proyección perspectiva, necesitamos tres vectores:

- el **vector posición** (*camera center*): es el vector con origen en el sistema de coordenadas del mundo, y con final en el punto en el que se sitúa la cámara (el centro de proyección). Define el centro del sistema de coordenadas de la cámara.
- el **vector de orientación de la lente** (*look-at vector*): es el vector con origen en el centro del sistema de coordenadas del ojo, y que define la dirección a la que mira la cámara. Dicho vector une el centro de proyección con el centro del plano de proyección.
- el **vector vertical** (*up vector*): es el vector con origen en el sistema de coordenadas del ojo, y con una dirección tal, que tomaremos dicha dirección para medir la distancia superior e inferior de la pirámide truncada de proyección.

Podemos especificar un cuarto vector (**vector horizontal derecho**) que indique la dirección derecha (*right vector*). Este vector indicaría la dirección que tomaríamos para medir las distancias derecha e izquierda de la pirámide de visión. Sin embargo, no es necesario especificar este vector porque se puede calcular a partir del producto vectorial de los vectores vertical ($\vec{u}$) y de orientación de la lente ($\vec{l}$):

$$\vec{r} = -\vec{l} \times \vec{u}$$

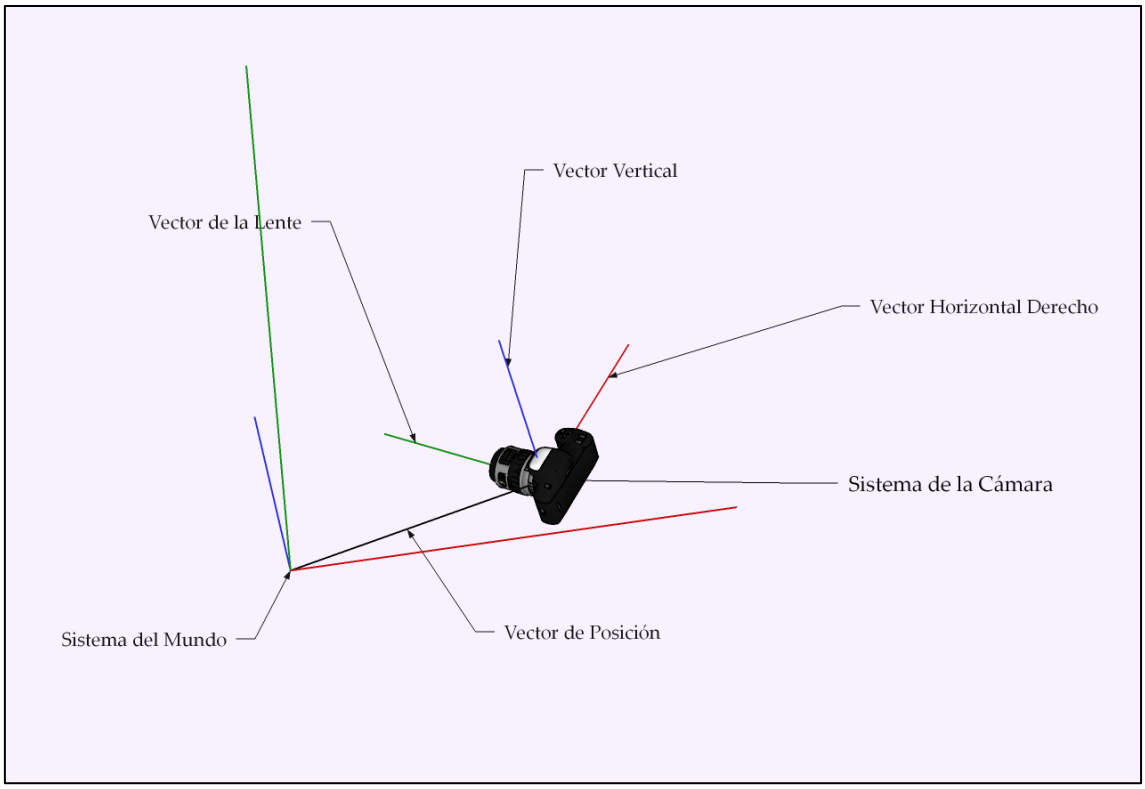

Figura 9.14 – Especificicación de la cámara

Una vez definidos estos vectores tenemos claramente definida la posición y orientación de la cámara. Con ello, conocemos la posición del centro de proyección y la orientación del plano de proyección. Si definimos la distancia cercana, ya podemos situar el plano de proyección de manera precisa. Si además definimos las distancias derecha, izquierda, superior e inferior, queda definido completamente el sistema de proyección de la representación tridimensional.

9.3 – Transformaciones Afines en 3D

A la hora de representar objetos tridimensionales mediante un computador, éstos suelen estar definidos en coordenadas locales del objeto. Por ello, para situarlos en la posición y orientación deseada con el tamaño deseado, hemos de ser capaces de transformar ese objeto hasta que éste tenga la posición, orientación y tamaño deseado.

Para ello podemos aplicar tres de las transformaciones que vimos con anterioridad: la traslación, el giro, y el escalado, pero sobre un espacio tridimensional y no sobre un plano. A estas transformaciones se les conoce como transformaciones afines.

9.3.1 – Traslación

La traslación es la transformación isométrica que nos permite mover/situar el objeto a/en la posición que le corresponde. Si cargamos un objeto definido en coordenadas del objeto en nuestra escena virtual, tendremos el objeto situado sobre el centro de coordenadas del mundo, cosa que no es siempre lo que queremos.

La traslación es la transformación geométrica que permite desplazar el objeto de un punto de la escena a otro. Su definición es completamente análoga a la que vimos para dos dimensiones.

La expresión en forma vectorial de la traslación es:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

donde (x, y, z) es el punto original, (x', y', z') es el punto transformado por la traslación, y T_x, T_y, T_z representan las tres componentes del vector de traslación.

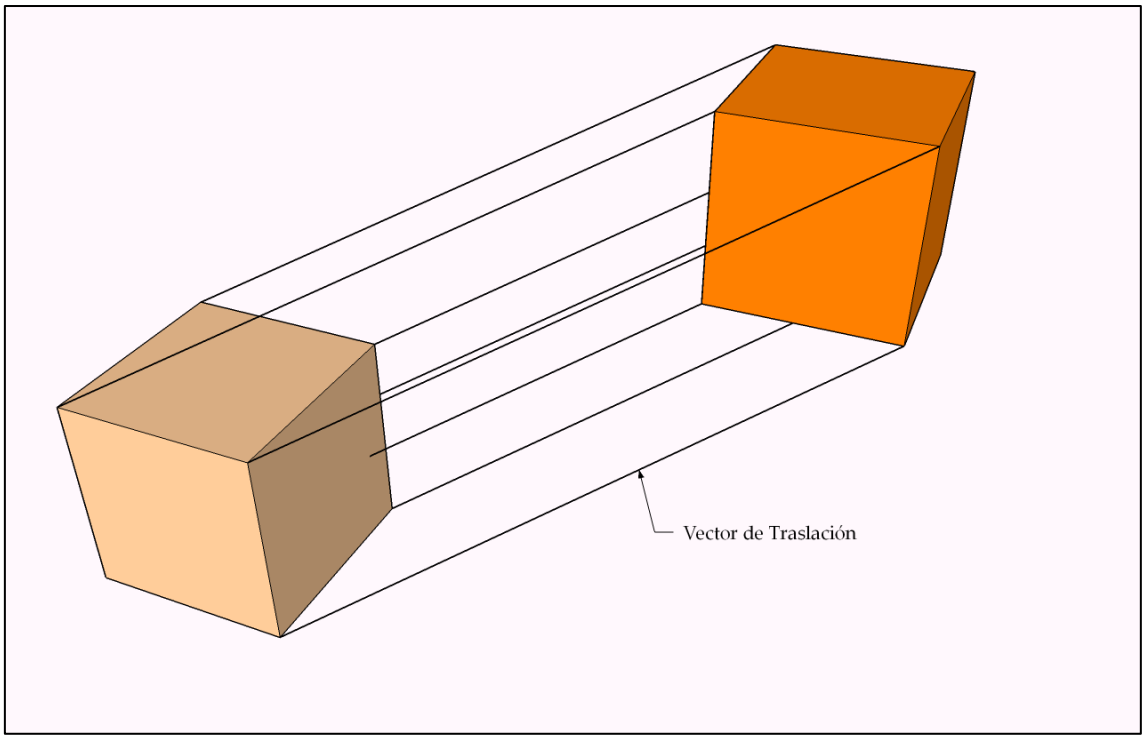

Figura 9.15 – Traslación en 3D.

9.3.2 – Giro/Rotación

Una vez tenemos el objeto situado en la posición deseada, es posible que deseemos cambiar también su orientación. Para ello, debemos aplicar la transformación isométrica llamada giro.

La definición del giro es diferente a la que vimos en dos dimensiones, puesto que necesitamos especificar un centro de giro, un plano de giro, un ángulo y un sentido de giro, en lugar de sólo un centro y un ángulo. El centro de giro y el plano se suelen especificar mediante un eje de giro, que es el vector normal a ese plano que pasa por ese punto. El sentido de giro se suele tomar como positivo según la regla de la mano derecha y negativo, en sentido opuesto.

Además, lo habitual es realizar el giro del objeto con respecto a un eje que pase por su centro, ya que éste es el punto de origen del sistema de coordenadas del objeto. Cuando un giro se realiza con respecto a uno de los ejes del sistema de coordenadas de una pieza u objeto, se denomina **rotación**.

La rotación es, lógicamente, diferente para cada eje, por lo que se suele expresar en forma matricial, y dado que trabajamos en un sistema de coordenadas de tres componentes, las rotaciones en 3D suelen descomponerse en sus componentes ortogonales (x, y, z).

Para una rotación de ángulo α_x alrededor del eje x, la expresión matricial es:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_x) & -\sin(\alpha_x) \\ 0 & \sin(\alpha_x) & \cos(\alpha_x) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Para una rotación de ángulo α_y alrededor del eje y, la expresión matricial es:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\alpha_y) & 0 & \sin(\alpha_y) \\ 0 & 1 & 0 \\ -\sin(\alpha_y) & 0 & \cos(\alpha_y) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Para una rotación de ángulo α_z alrededor del eje z, la expresión matricial es:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\alpha_z) & -\sin(\alpha_z) & 0 \\ \sin(\alpha_z) & \cos(\alpha_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Si realizamos una rotación alrededor de un eje arbitario, la ecuación se obtiene de combinar las tres matrices anteriores con tres ángulos sobre los tres ejes de coordenadas ortogonales x, y, z. La expresión es:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

de forma que R_{11}, R_{12}, … R_{33} son coeficientes que indican cómo se transforma cada extremo del eje de coordenadas del sistema original al aplicar la rotación. De manera que R_{i1} indica cuál es la posición final del punto $(1, 0, 0)$ (el extremo del eje x) al aplicar la rotación; R_{i2} indica cuál es la posición del extremo del eje y al rotar; y R_{i3} indica cuál es la posición del extremo del eje z al rotar.

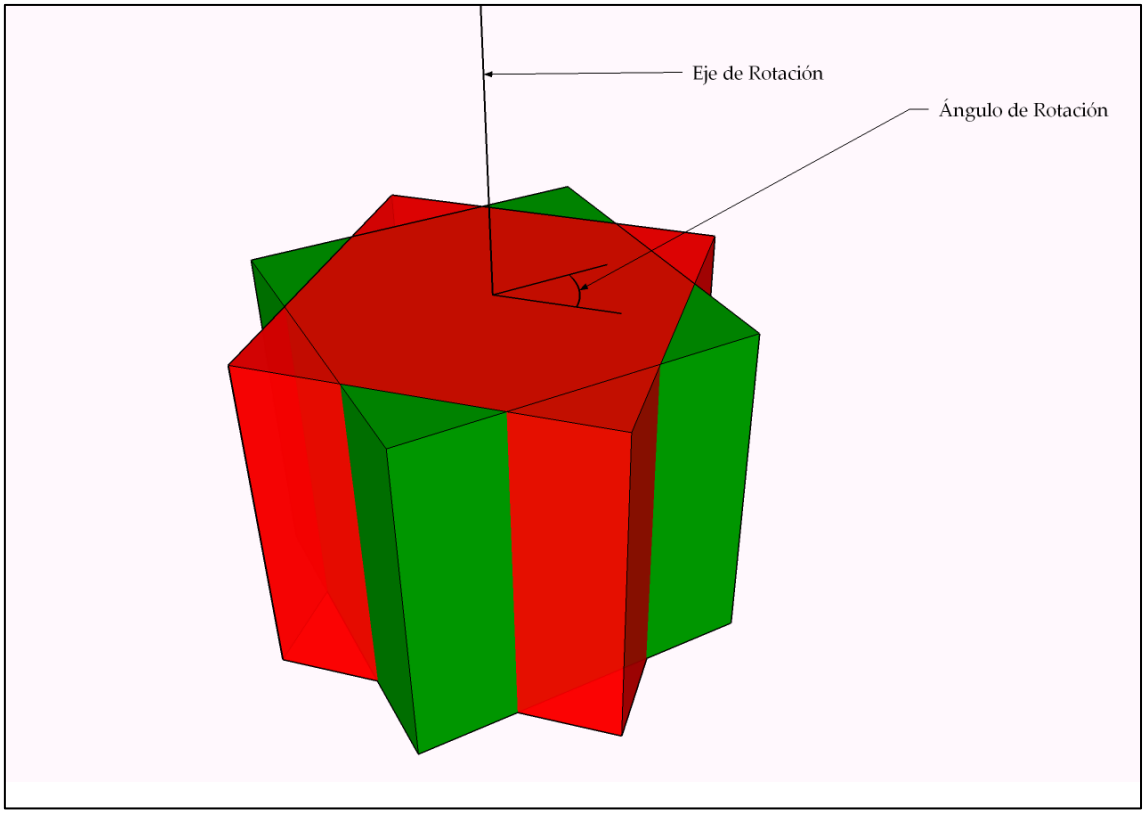

Figura 9.16 – Rotación en 3D.

9.3.3 – Escalado

Una vez situado y orientado el objeto, es posible que queramos cambiar el tamaño del objeto, ya que muchas veces se realizan modelos de piezas que son plantillas de objetos, y una vez introducidos en la escena virtual, es necesario darles el tamaño real.

Para conseguir este objetivo podemos realizar un escalado del objeto. El escalado se hace prácticamente siempre con respecto al centro de coordenadas del sistema del objeto, de modo que se cambia el tamaño del objeto realizando una homotecia con centro en ese punto. La razón de la homotecia dependerá de cuánto queramos ampliar o reducir el tamaño del objeto original.

Visto así, la extensión del concepto de escalado a tres dimensiones es trivial, ya que su especificación es análoga a la de dos dimensiones.

Sin embargo, es habitual realizar escalados diferentes para cada una de las direcciones de los ejes de coordenadas. En este caso ya no sería una homotecia de razón K, sino que se trataría de una transformación isomórfica con tres razones S_x, S_y, S_z que en forma matricial quedaría expresada de la siguiente forma:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

donde S_x, S_y, S_z representan las razones o factores de escalado aplicados para cada eje. Nótese que no se especifica un centro de escalado, ya que se asume que es el centro del sistema de coordenadas.

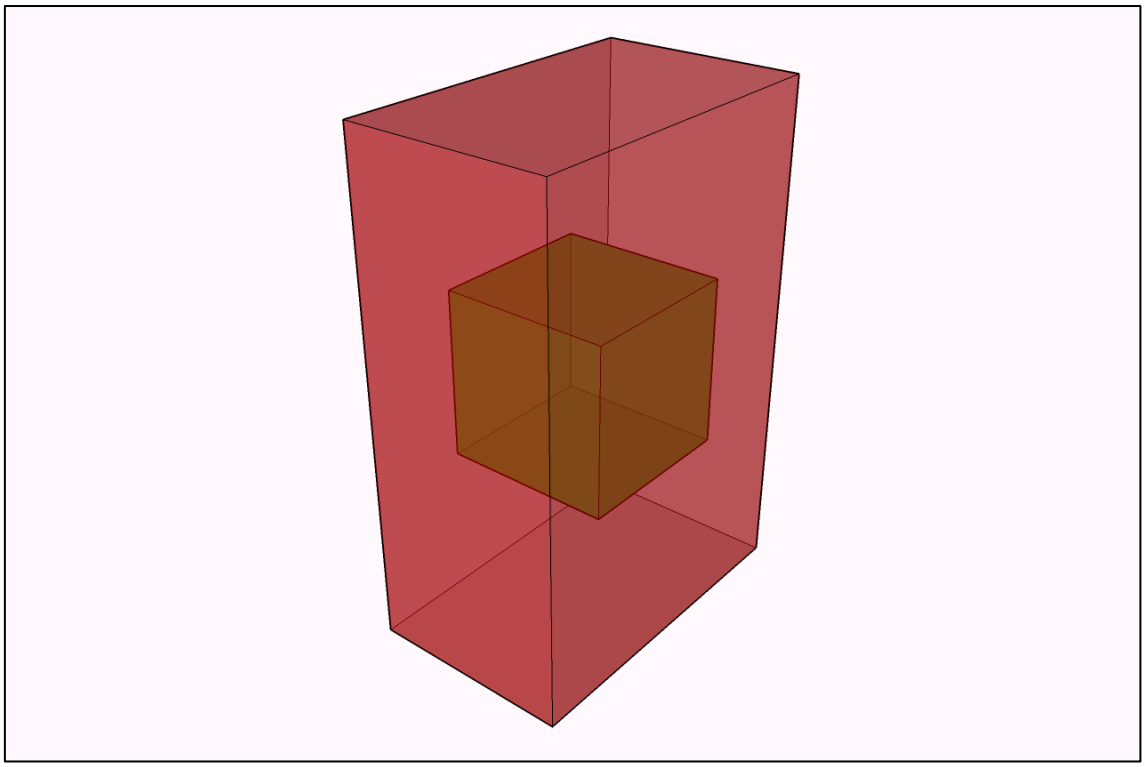

Figura 9.17 – Escalado en 3D.

9.3.4 – Representación Matricial Combinada

Dado que habitualmente emplearemos simultáneamente los tres tipos de transformaciones, es muy común realizar las tres operaciones de manera combinada mediante una única matriz que incorpora los tres tipos de transformaciones.

Para utilizar este tipo de representación matricial, se suele añadir una cuarta componente a los vectores que representan los puntos (tridimensionales) o vértices de las figuras. Esta cuarta componente, llamada **componente homgénea**, es redundante, y su introducción se realiza porque simplifica y compacta la notación matricial.

Para cada vector de cuatro componentes, su vector equivalente de tres componentes se calcula dividiendo cada una de las tres primeras por la componente homogénea w:

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x/w \\ y/w \\ z/w \end{bmatrix}$$

De esta manera, si queremos representar un vector (x, y, z), con cuatro coordenadas, sólo hemos de añadir un 1 para dar valor a w:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Con este tipo de representación homogénea, la representación matricial del proceso de traslación, rotación y escalado, es la siguiente:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

donde:

- el vector de traslación será (M_{14}, M_{24}, M_{34}).

- el vector columna (M_{11}, M_{21}, M_{31}) indica cómo se transforma el extremo del eje x (estos coeficientes aglutinan la rotación y el escalado sobre el eje x).

- el vector columna (M_{12}, M_{22}, M_{32}) indica cómo se transforma el extremo del eje y (estos coeficientes aglutinan la rotación y el escalado sobre el eje y).

- el vector columna (M_{13}, M_{23}, M_{33}) indica cómo se transforma el extremo del eje z (estos coeficientes aglutinan la rotación y el escalado sobre el eje z).

9.4 – La Tubería Gráfica

Ahora que ya conocemos cómo se representa la información (mediante primitivas poligonales) en la mayoría de los sistemas de representación 3D interactivos, cómo se visualizan estos objetos (con la proyección perspectiva acotada) y cómo se sitúan, orientan y escalan los objetos, ya estamos en condiciones de comprender cómo es el proceso completo que tiene lugar en un computador para visualizar objetos tridimensionales en tiempo real.

Este proceso tiene lugar en forma de cadena de montaje, de manera que el proceso se divide en una serie de pasos. La entrada de cada paso es la salida del anterior, formándose una estructura que se llama **tubería gráfica** o *pipeline*, puesto que el proceso se asemeja a una tubería.

La tubería gráfica está implementada en algunas librerías de visualización gráfica interactiva como *OpenGL* o *DirectX* y se descompone en una serie de fases. Actualmente la mayoría de estas fases son ejecutadas por el procesador de la tarjeta gráfica, y algunas de ellas son programables de modo que se puede modificar su comportamiento a voluntad del programador.

A grandes rasgos, las fases de la *pipeline* gráfica son las siguientes:

Conversión a polígonos

En gráficos interactivos toda representación de objetos 3D que no sea poligonal se debe previamente transformar en una representación de superficies poligonales para poder hacer uso de la *pipeline* gráfica. Aunque lo normal es que los objetos ya vengan en este tipo de representación, si esto no fuera así, primero se deben convertir a este tipo de representación. Como esto puede ser costoso, si ha de realizarse esta conversión, se hace previamente y no se realiza la conversación en cada iteración del dibujado, por lo que es discutible considerar este proceso como parte de la tubería gráfica.

Selección de objetos.

En la fase de selección se intenta optimizar la calidad y cantidad de objetos que se van a utilizar. La selección de objetos se basa en dos técnicas principales:

• Selección del detalle. Un mismo objeto puede estar representado con más de un nivel de detalle (con más o con menos caras poligonales), y por tanto, podemos escoger el que sea más adecuado (según su calidad y coste computacional) en cada momento. Es habitual elegir modelos menos detallados cuando los objetos están lejos de la cámara, y modelos más detallados cuando están cerca.

• Selección por visibilidad. Si queremos disminuir el coste computacional de la tubería gráfica, podemos dejar de enviar los objetos que estén ocultos o fuera del campo de visión.

Transformaciones de los modelos.

En esta fase transformamos los modelos de coordenadas de los objetos a coordenadas del mundo. Para ello utilizamos los tres tipos de transformaciones afines que hemos visto. De esta manera los situamos en las posiciones deseadas, con las orientaciones y tamaños deseados.

Transformación de la vista.

Una vez seleccionados y situados los objetos, el siguiente paso es transformar los vértices de todos los objetos (que ahora ya están especificados en coordenadas del mundo) a coordenadas de la cámara. Esto nos permite hacer "la fotografía" de la escena suponiendo que el centro de proyección es el centro de nuestro sistema de coordenadas.

Cálculo de iluminación.

Cuando ya tenemos todos los objetos en coordenadas de la cámara, procedemos a analizar qué color va a tener cada cara en función de las propiedades que tengan sus vértices (en función de los materiales y de los vectores normales en cada vértice) y de la posición de éstos respecto a la cámara, ya que esto puede modificar su aspecto.

Proyección y recorte.

Ahora ya sabemos, para cada vértice, su posición en coordenadas del ojo, y el color que debe tener. Toda esta información sigue siendo tridimensional, por lo que para mostarla debemos hacer una proyección. Al aplicar la proyección perspectiva acotada sobre los objetos obtendremos la representación bidimensional de los mismos. A este proceso se le conoce como transformaciones de proyección.

Dado que usamos una proyección acotada, aprovechamos para recotar la escena y eliminar todo aquello que quede fuera de la pirámide de visión. A este proceso se le conoce como **recorte** o *culling*.

La salida de esta fase son vértices proyectados bidimensionales. Muchas veces se obtiene una tercera componente que nos indica la distancia a la que estaba cada vértice del plano de proyección. Esta componente (z) nos permite ordenar los objetos proyectados de manera que los objetos lejanos queden detrás de los más cercanos, mediante un proceso que se llama **z-buffering.**

Rasterizado y transformaciones de ventana.

A partir de las coordenadas bidimensionales obtenidas en el punto anterior (que representan puntos en el plano de proyección), y del cálculo de iluminación que se ha hecho sobre cada vértice, debemos obtener una representación de cada cara en pixels discretos. A este proceso se le denomina **rasterización**. En este paso se realiza el

sombreado, y se combina la información de iluminación y de texturización para dar lugar a un conjunto de pixeles con diferentes colores.

Además, debemos ajustar los pixeles a coordenadas relativas a la pantalla o ventana en la que dibujemos esa escena. A este proceso (que transforma coordenadas (x, y) reales a pixels) se le llama **transformaciones de ventana (o del dispositivo)**. El proceso debe tener en cuenta el tamaño (en pixels) de la ventana o pantalla sobre la que se dibuja y la relación de aspecto del plano de proyección.

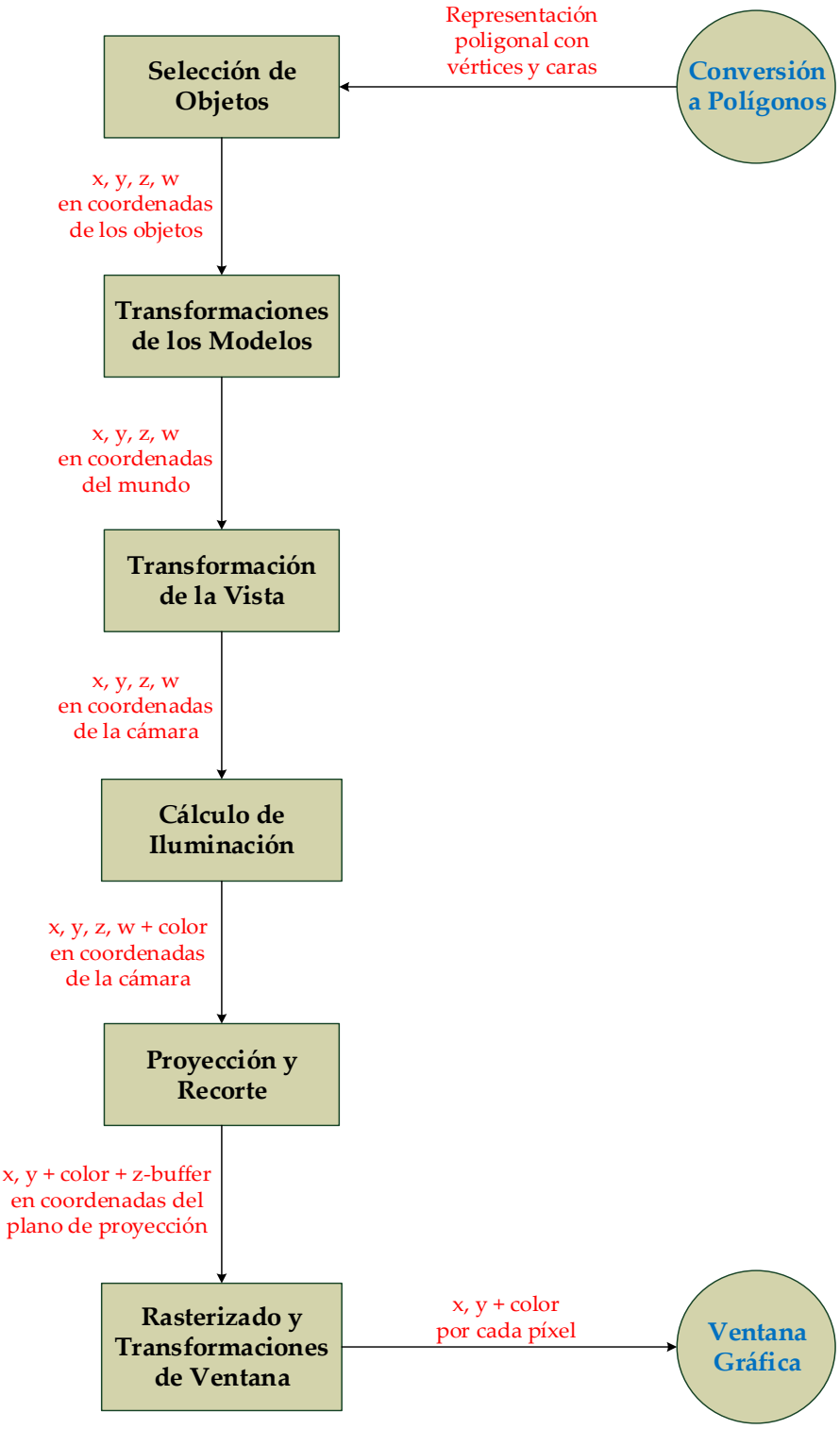

Figura 9.18 – La tubería gráfica.

9.5 – Ejemplo Práctico de Modelado 3D – SketchUp

SketchUp es programa que permite crear y visualizar de manera interactiva objetos tridimensionales. Fue creado en 2000 por una compañía desconocida hasta entonces llamada Last Software, y empezó a ser popular cuando fue adquirido por el gigante tecnológico Google en 2005. Google fue generando sucesivas versiones del programa desde la versión 5 hasta la última llamada "Google SketchUp 8". En 2012 Google vendió SketchUp a la empresa Trimble, por lo que desde entonces el programa se llama Trimble SketchUp, y la última versión es la versión 2015.

Sin entrar en las diferencias entre las distintas versiones, SketchUp está basado en una representación de superficies mediante caras poligonales, exactamente cómo hemos explicado antes, aunque a diferencia de otros programas, habitualmente no muestra cómo está organizada la malla de caras planas, especialmente en superficies curvas.

SketchUp emplea un mundo tridimensional virtual con un sistema de coordenadas dextrógiro de tres coordenadas: x, y, z, de modo que dibuja una línea roja para indicar el eje x, una línea verde para indicar el eje y, y una línea azul para representar el eje z.

Emplea dos tipos de proyecciones para mostrar la información visual: la proyección plana perspectiva acotada que hemos explicado, y una proyección plana paralela ortogonal también acotada. El programa permite alternar entre ambas, permitiendo además, seleccionar la orientación de la cámara en las direcciones de planta, perfil, alzado e incluso en vista isométrica.

En la proyección perspectiva se puede modificar el campo visual, indicando el número exacto de grados del FOV, y se puede hacer zoom, de manera que el programa modifica automáticamente el campo visual en función del zoom deseado.

9.5.1 – Interfaz y Comandos Básicos

SketchUp funciona básicamente por **inferencia**. El funcionamiento habitual es realizar líneas y superficies 2D, partiendo de puntos previamente dibujados (el origen del sistema de coordenadas, puntos sobre los ejes, vértices y aristas de otras figuras, etc.). El programa intenta inferir qué es lo que quiere hacer el usuario de modo que cuando el ratón se acerca a ciertas partes de la escena, SketchUp selecciona por inferencia ciertos puntos para comenzar el trazado desde ahí. Las inferencias dependen de lo que el usuario realice y de cómo esté orientada la cámara. Las dimensiones de las líneas, los radios de las circunferencias, y las dimensiones de los rectágulos, se pueden especificar por teclado.

El funcionamiento por inferencia lo hace especialmente sencillo de usar para realizar objetos sencillos, pero puede llegar a ser un estorbo cuando se trata de dibujar objetos más complejos. La inferencia es por tanto, al mismo tiempo, la mayor ventaja y el peor inconveniente de SketchUp.

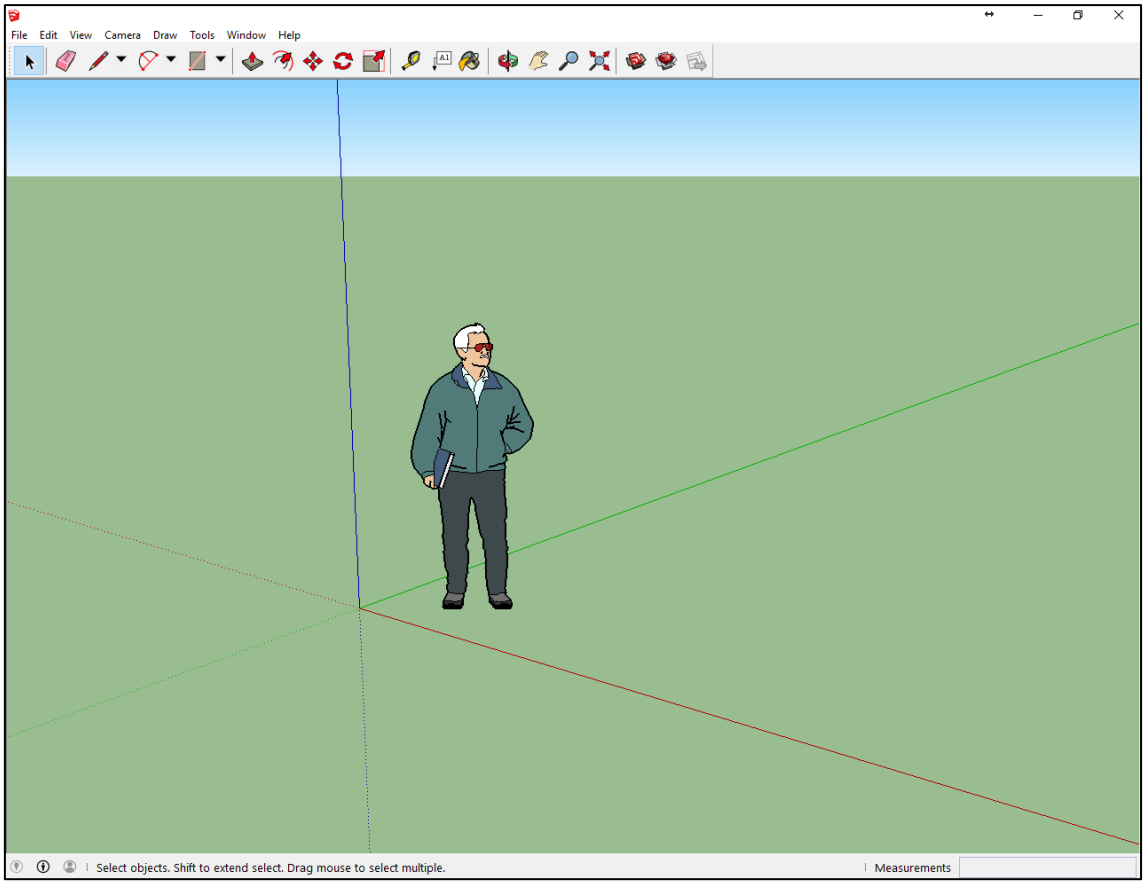

Figura 9.19 – SketchUp 2015.

Para realizar objetos bidimensionales, la herramienta dispone de comandos para realizar líneas rectas, círculos, arcos, rectángulos y polígonos. Es importante entender que, a diferencia de AutoCAD, SketchUp instancia las circunferencias directamente como polígonos de un cierto número de lados. De modo que, cuando se dibuja un círculo o un arco, en realidad se dibuja una cara poligonal o una línea poligonal. El número de lados que emplea SketchUp para aproximar las circunferencias o los arcos se puede modificar por teclado, pero sólo antes de dibujar la figura. No después.

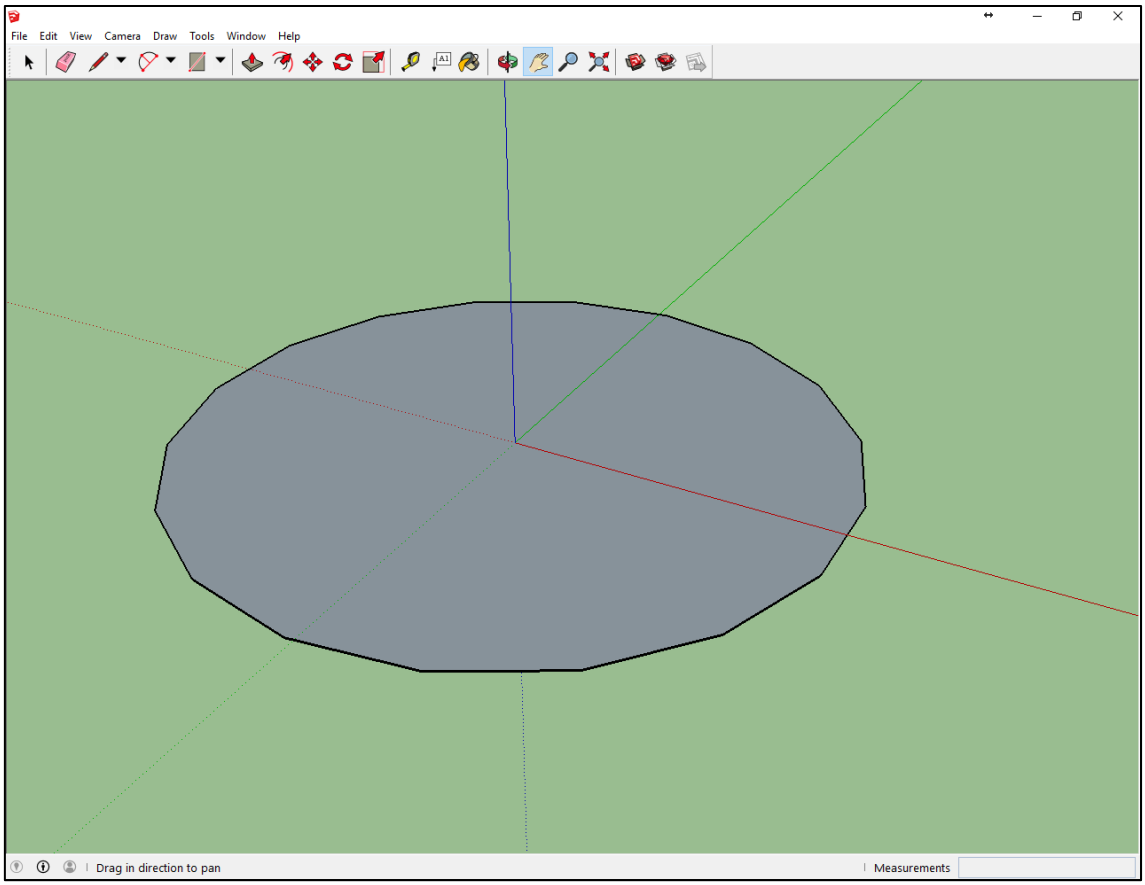

Figura 9.20 – Aproximación poligonal de una superficie curva en SketchUp.

Para realizar objetos tridimensionales, lo habitual es dibujar algún objeto bidimensional y luego extruirlo (levantarlo) o generar una superficie de revolución a partir de él, por lo que no hay herramientas para construir cubos, esferas, etc.

Sin embargo, también pueden dibujarse las aristas del objeto tridimensional, y SketchUp irá creando las caras poligonales correspondientes cuando encuentre coplanariedad geométrica entre las aristas. Siempre, por inferencia.

9.5.1.1 – Uso del Ratón

SketchUp funciona con ratones de tres botones y de un solo botón, pero es preferible utilizar ratones de tres botones, ya que aumentan considerablemente la eficiencia de la herramienta. Antes de empezar a dibujar en SketchUp, se debe aprender las distintas funcionalidades de cada botón del ratón.

El botón izquierdo sirve para seleccionar objetos y para dibujar primitivas (según cuál se haya seleccionado). Pero si se selecciona la herramienta "Desplazar", servirá para mover la cámara lateral o verticalmente (con respecto a la orientación de la cámara) por la escena.

El botón central permite orbitar (rotar) la cámara de modo que se vea una zona distinta de la escena. Además, si se mueve la rueda del botón central, se desplaza la cámara en la dirección en la que ésta mira (no confundir con el zoom porque a veces proporciona un efecto similar, pero es completamente diferente ya que al desplazarse por la escena no se cambia el FOV).

El botón derecho sirve habitualmente para mostrar menús contextuales.

Dado que SketchUp funciona por inferencia, es muy importante entender que las sugerencias de posición que hace el programa dependen de la posición y orientación de la cámara, ya que según situemos el ratón en un punto u otro de la escena, el programa va a inferir que queremos empezar una línea o cara en un punto u otro de la escena 3D.

Cuando SketchUp piensa que estamos buscando un punto sobre el eje x, o sobre una paralela a él, muestra una línea roja. Cuando es sobre el eje y, muestra una línea verde. Y cuando es sobre el eje z, una línea azul. Cuando el programa cree que buscamos hacer una perpendicular o paralela a alguna línea ya dibujada, Skecth Up muestra una línea violeta.

Es posible, sin embargo, bloquear la inferencia una vez ésta ha sido sugerida, para que al mover el ratón, no se pierda. Para ello, se debe mantener presionada la tecla "Mayúsculas" para bloquear la operación sobre ese eje.

También se puede forzar la inferencia en una **dirección fija de inferencia**. Si se mantiene presionado el cursor hacia arriba, Skecth Up sugerirá el eje z o alguna paralela. Con el cursor izquierdo, se forzará la dirección y. Mientras que con el cursor derecho, se fuerza la dirección x.

9.5.2 – Empujar/Tirar y Sígueme

A diferencia de otros programas similares, la construcción de objetos tridimensionales en SketchUp suele realizarse primero dibujando objetos bidimensionales. Por ejemplo, para realizar un cubo, primero dibujaríamos un cuadrado, y después levantaríamos esa cara en la dirección de la normal al plano que define el cuadrado. Ese proceso, llamado **extrusión**, se realiza en SketchUp con la herramienta "**Empujar/Tirar"**.

Dicha herramienta permite levantar o reducir volúmenes empujando caras en una cierta dirección, y una cierta cantidad. La dirección se elige por inferencia, y la cantidad se puede especificar introduciéndola por teclado.

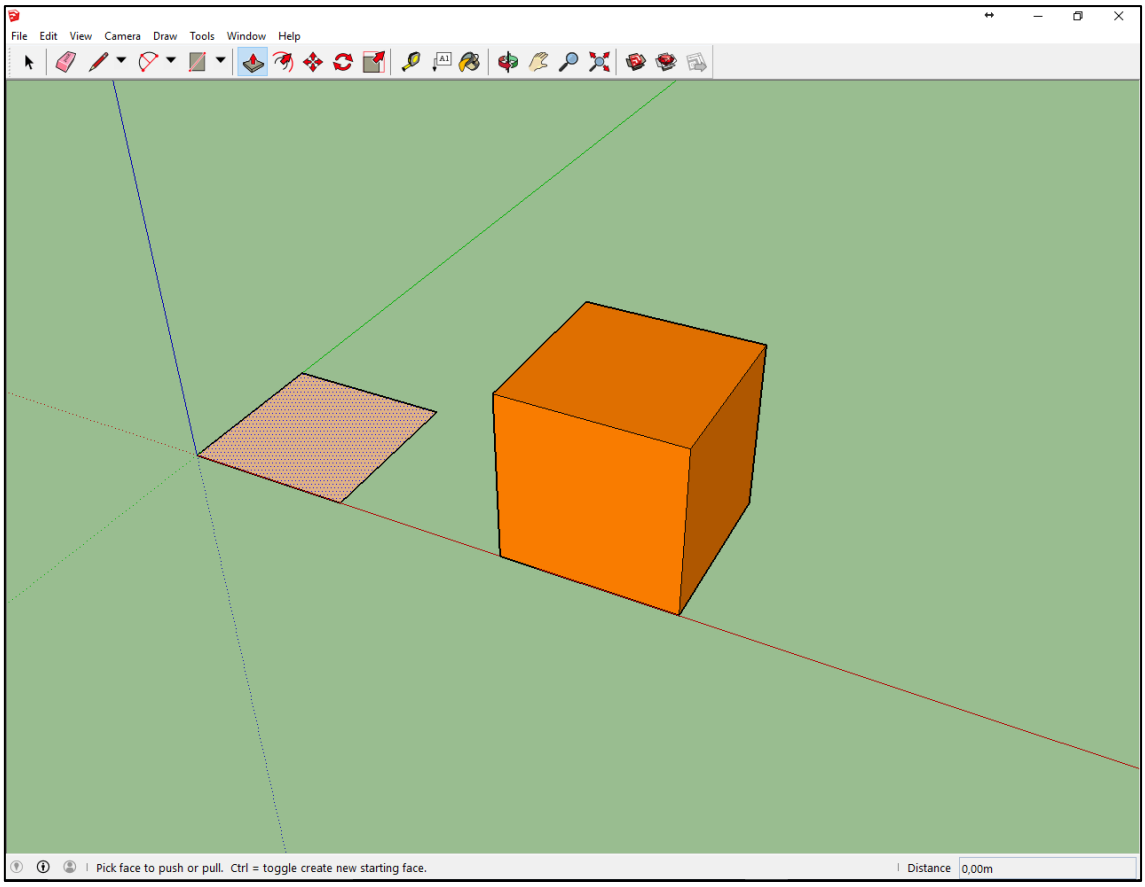

Figura 9.21 – Herramienta "Empujar/Tirar".

Con "Empujar/Tirar" también es posible crear un espacio vacío simplemente dibujando una forma 2D sobre una cara de una geometría 3D y utilizando la herramienta Empujar/Tirar para empujar la cara 2D hasta que alcance la cara posterior de la geometría 3D. Esto crearía un hueco en el volumen.

La otra herramienta que permite crear objetos tridimensionales a partir de objetos planos es la herramienta "**Sígueme"**. Esta herramienta implementa el concepto de **sólido de revolución**, de manera que permite seleccionar una cara y una línea (recta o curva) a seguir, de modo que se forme un objeto tridimensional a base de desplazar dicha cara sobre la línea. La selección del seguimiento de la línea se hace desplazando el cursor sobre la línea a seguir.

Si una vez seleccionada la herramienta "Sígueme" y la cara a extruir, se presiona la tecla "Alt" sobre otra cara diferente, SketchUp buscará el contorno de dicha cara, y empleará ese contorno como línea a seguir para construir el volumen de revolución.

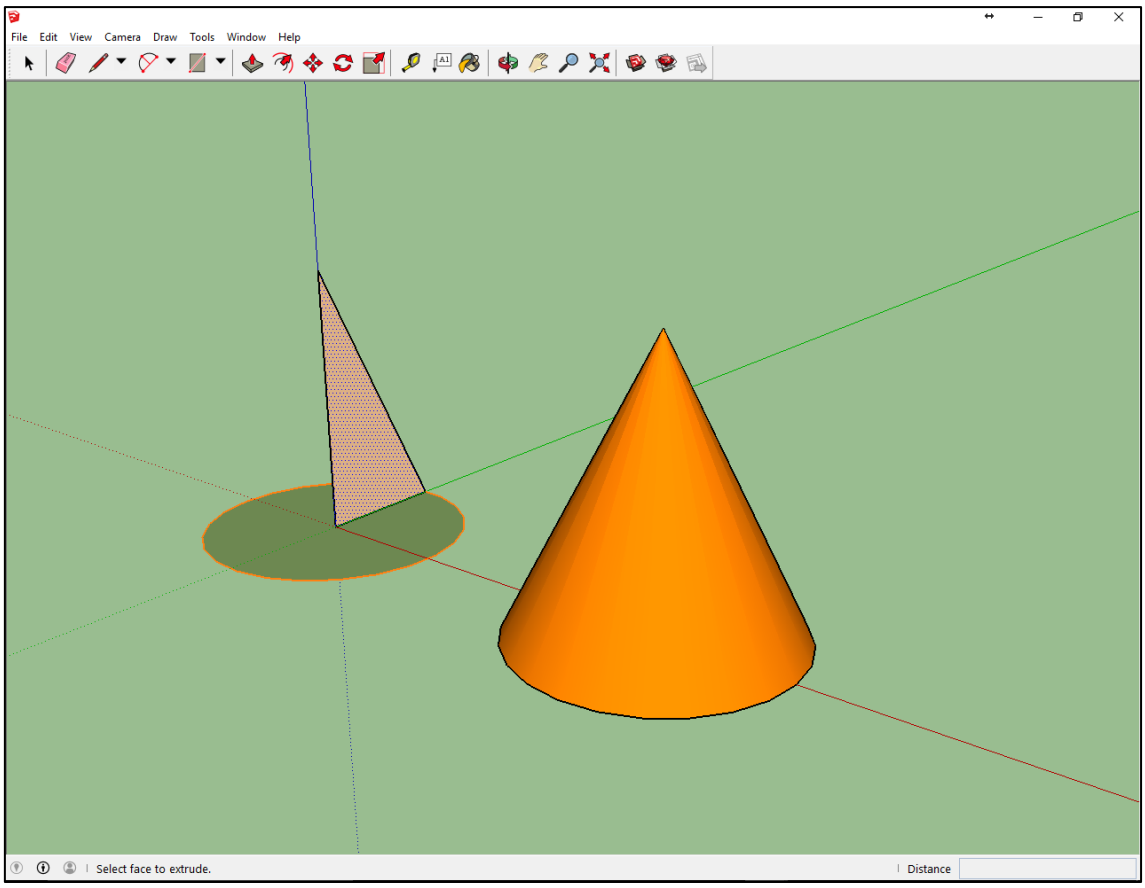

Figura 9.22 – Herramienta "Sígueme".

9.5.3 – Transformaciones de Objetos

SketchUp implementa las transformaciones de traslación, rotación y escalado.

La traslación se denomina "Mover", y se aplica seleccionando un punto inicial y un punto final. Ambos puntos definen el vector de traslación.

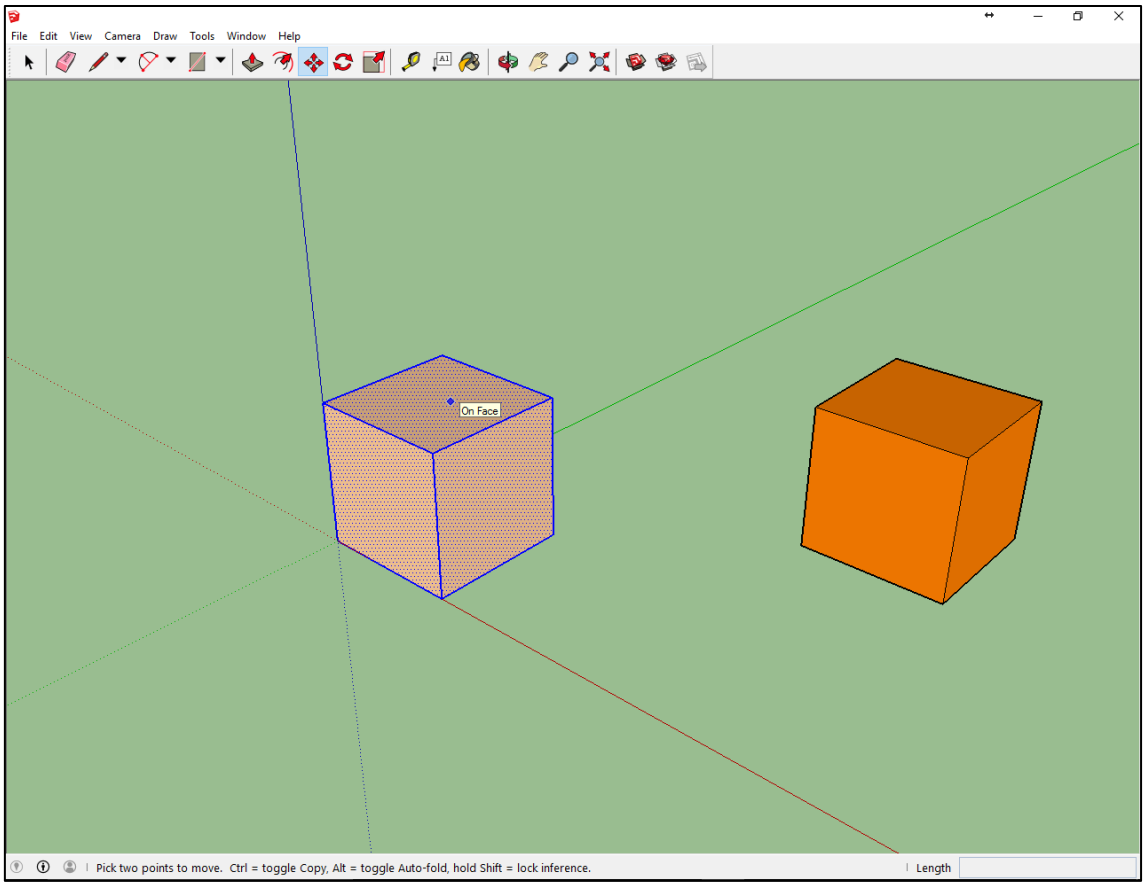

Figura 9.23 – Herramienta "Mover".

El escalado se aplica mediante la herramienta "Escala", que muestra un cubo amarillo que permite escalar el objeto en las diferentes direcciones del espacio. La cantidad a escalar se puede escribir por teclado o mediante el movimiento del ratón.

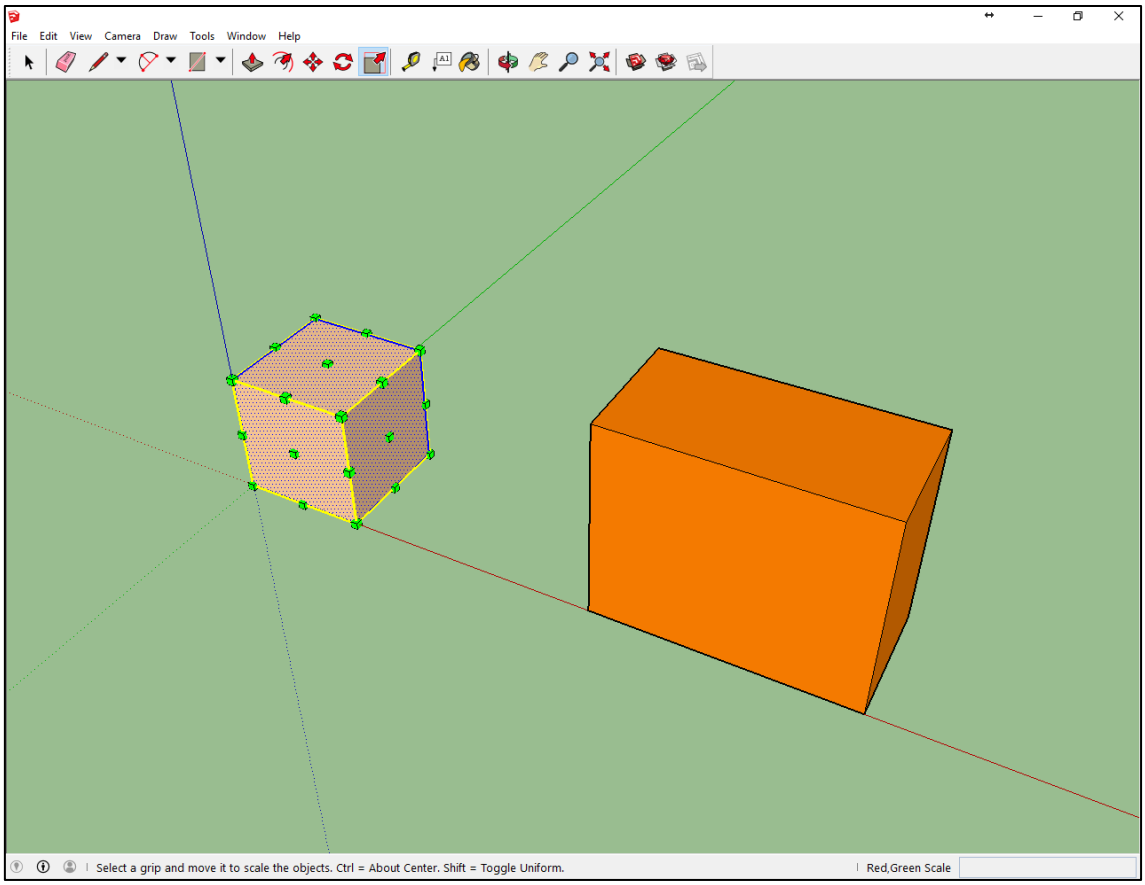

Figura 9.24 – Herramienta "Escala".

La herramienta de rotación se llama "Rotar", y es en realidad un giro, no una rotación, ya que obliga al usuario a elegir un plano (mediante un transportador de ángulos que dibuja la herrmienta sobre la cara que selecciones), un centro de giro, y un ángulo que se especifica seleccionando un punto inicial del giro y un punto final. Si especificamos un punto inicial podemos escribir el ángulo con el teclado sin necesidad de buscar cuál es el punto final.

La herramienta "Rotar" puede llegar a ser difícil de usar si intentamos girar en base a un plano que no se corresponda con una cara ya dibujada. Al funcionar por inferencias, este tipo de operaciones se complica.

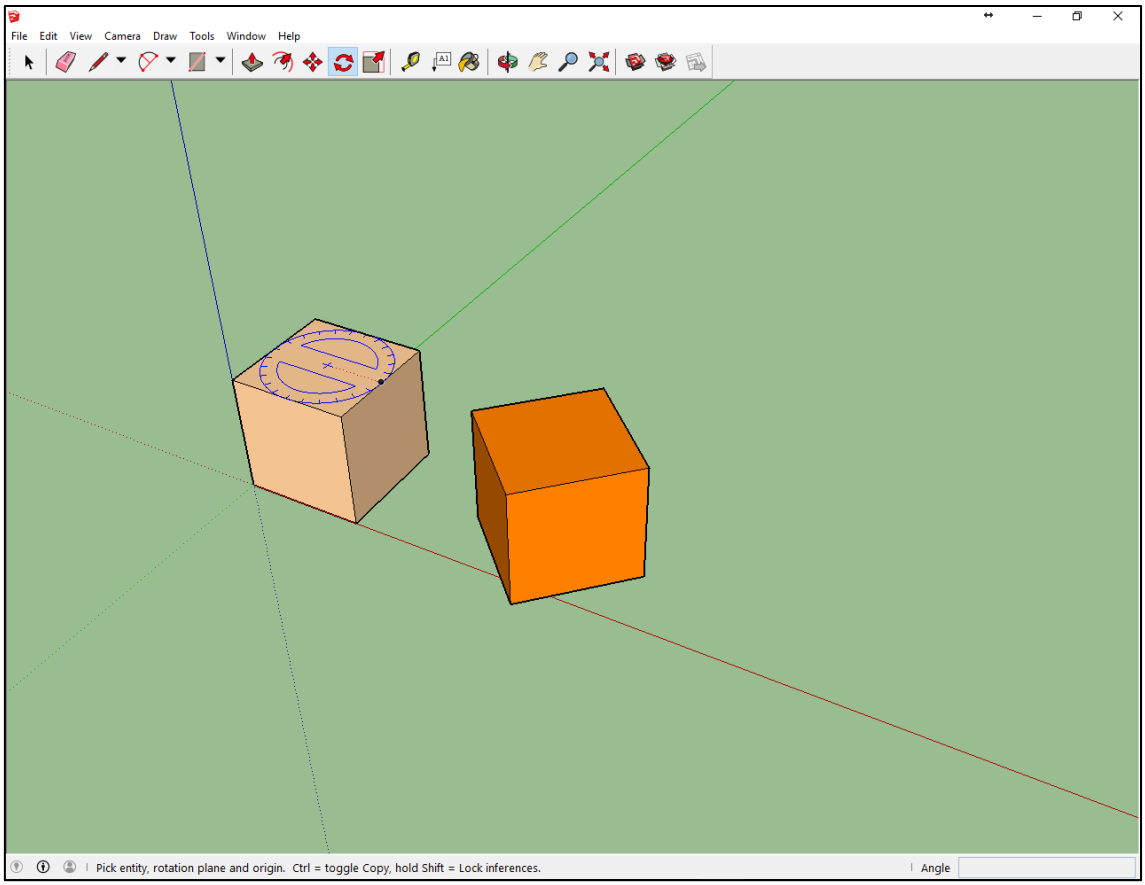

Figura 9.25 – Herramienta "Rotar".

9.5.4 – Detalles de los Objetos

SketchUp también permite modificar los materiales de las caras y texturizarlas, de modo que se visualizan de manera diferente en función de los materiales elegidos. El modelo de sombreado y texturizado es bastante simple, por lo que para realizar visualizaciones realistas se suelen utilizar otros programas como Autodesk 3D Studio Max, ya que éste permite controlar la iluminación y los parámetros de los materiales de una manera más precisa.

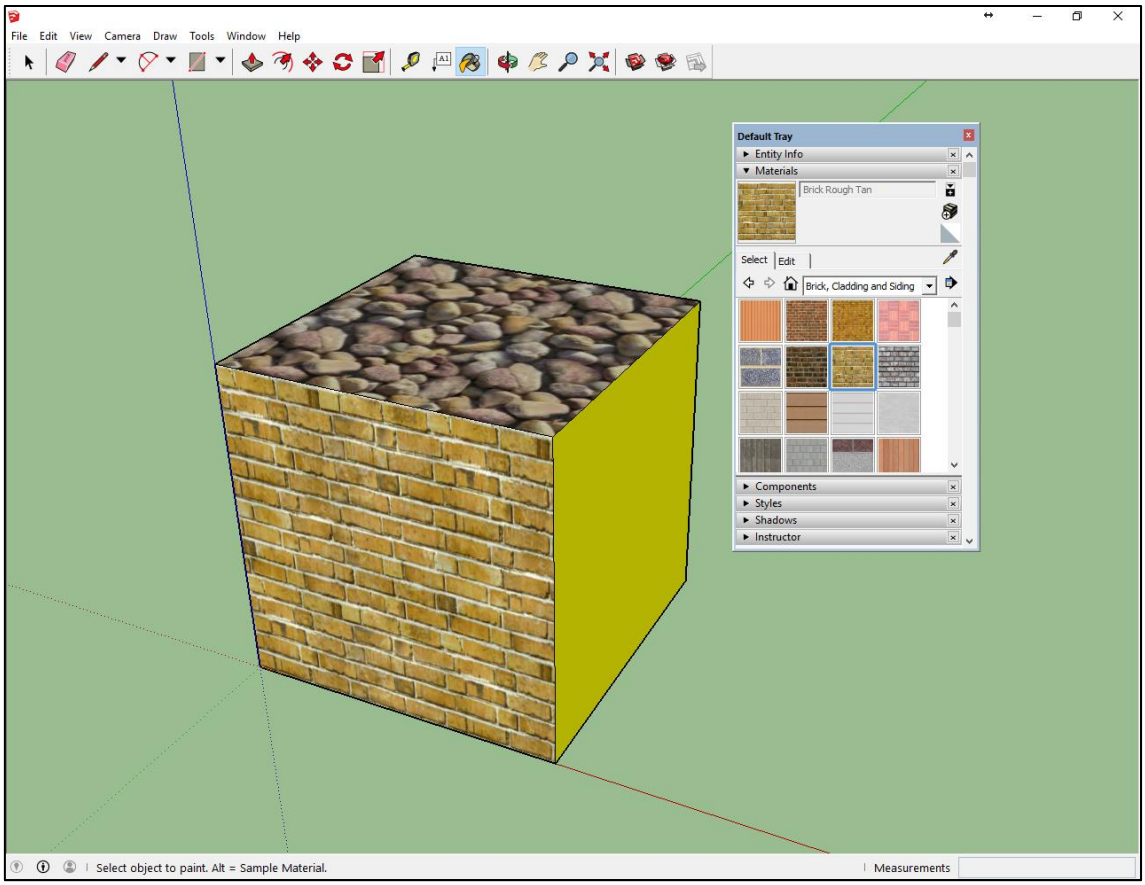

Figura 9.26 – Herramienta "Materiales".