Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXVIII, págs. 1203-1222 (artículo completo)
Cristian García Bauza, Pablo Lotito, Lisandro Parente, Marcelo Vénere (Eds.)
Tandil, Argentina, 3-6 Noviembre 2009

OPERACIONES DE MANIPULACIÓN DE OBJETOS 3D

Alejandro Cosimo[a] **y Nestor Calvo**[a,b]

[a]*Ingeniería en Informática, Facultad de Ingeniería y Ciencias Hídricas, Universidad Nacional del Litoral, Ciudad Universitaria, Paraje "El Pozo", S3000, Santa Fe, Argentina, http://fich.unl.edu.ar*

[b]*Centro Internacional de Métodos Computacionales en Ingeniería - INTEC - CONICET*

Palabras Clave: Herramientas de manipulación, Visualización de datos, Interacción usuario-geometría

Resumen. Todo software de geometría computacional que permita al usuario interactuar con la geometría en tiempo real, requiere implementar un conjunto de operaciones básicas como lo es, por ejemplo, una rotación del modelo geométrico. En la actualidad muchos sistemas CAD y de post procesamiento proveen este conjunto de herramientas pero tienen la particularidad de que no son tan intuitivas desde el punto de vista del usuario final. Es por esto que en este trabajo se propone un conjunto de operaciones de manipulación novedosas que tienen como característica principal ser intuitivas. Primero, se hace una introducción a la temática. Segundo, se propone un conjunto de operaciones de manipulación a implementar. Luego se presenta el desarrollo de cada una de ellas. Y por último se dan a conocer las conclusiones y los trabajos futuros.

1. INTRODUCCIÓN

Toda GUI (Graphic User Interface) que forme parte de un software de geometría computacional debería implementar un conjunto de operaciones o herramientas de manipulación de los objetos geométricos. Sistemas CAD y de post procesamiento[1] incorporan este tipo de herramientas que facilitan al usuario la visualización y manipulación del modelo. Sin embargo, se puede observar que en sistemas CAD tradicionales las operaciones de interacción implementadas no son tan intuitivas.

A lo largo de este trabajo se introducirán un conjunto de operaciones que presenten la característica de ser intuitivas. Esto es, intuitivas desde el punto de vista del usuario; si el usuario piensa "quiero ver ese vértice", él verá ese vértice sin pensar mucho en cómo alcanzar esa vista. Por ejemplo, el usuario rota el cuerpo geométrico pero de una manera que la operación siendo aplicada será muy similar a interactuar con el objeto moviéndolo con sus propias manos, generando la ilusión de que el software entiende lo que el usuario está queriendo ver.

Este tema es bastante importante ya que por medio de la intuición se ahorra tiempo destinado a la capacitación y al entrenamiento del usuario de la aplicación. Además, esta característica permite trabajar eficientemente, ya que el usuario no deberá gastar tiempo en pensar cómo, por ejemplo, rotar el objeto, sino que estará concentrado enteramente en el análisis que está realizando de la geometría por medio de su observación.

Debido a que OpenGL es una de las API gráficas estándar más utilizada para la implementación de GUIs, el trabajo será desarrollado teniendo en cuenta conceptos propios de esa API. A pesar de esta situación, los algoritmos ideados no pierden su generalidad, es decir, la idea que da vida a cada una de las operaciones de manipulación presentadas puede ser tomada para implementarse con otra API gráfica que no sea OpenGL. Para hacer más fácil el entendimiento de lo expuesto a lo largo del trabajo se van introduciendo ciertos conceptos de OpenGL.

2. SISTEMAS DE REFERENCIA

En OpenGL se pueden identificar básicamente dos sistemas coordenados, cada uno de los cuales conforma un espacio de trabajo. Uno es el Model Space o Espacio de Modelo que es utilizado para dibujar los objetos de la escena, y el otro es el Viewing Space o Espacio de Vista que se encuentra determinado por la posición y la orientación de la cámara o el observador.

Se podría pensar también en la existencia de otro espacio que es el Screen Space o Espacio de la Pantalla. A pesar de que existe es posible prescindir del mismo, ya que las coordenadas x, y del espacio de la pantalla son equivalentes a las del espacio de la vista excepto en unidades, escalas y profundidad.

Generalmente para dar soporte a las interacciones del usuario de la aplicación (rotaciones, traslaciones, escalados) con la escena que se ha renderizado, se trabaja en el Espacio de Modelo utilizando rutinas de transformación provistas por OpenGL, como lo son glScale*(), glRotate*() y glTranslate*(). Así de esta manera el implementador de la interfaz simplifica su labor a "sistemas locales" a los cuales aplica las operaciones descriptas. El usuario puede mover los objetos en el sistema del modelo o puede cambiar el punto de vista (respecto al sistema del modelo), dado que aquí se trata con modelos provenientes del CAD, fijos en el espacio, se considerará el espacio del modelo invariante. Al usuario se le hace creer que está manipulando el modelo, para visualizarlo de distintos ángulos, cuando en realidad está cambiando el punto de vista.

En este trabajo se toma como único sistema de referencia al sistema de la vista, las operaciones

[1] De ahora en más se hará referencia a sistemas CAD, pero la aplicabilidad de lo expuesto puede ser destinada de la misma manera a sistemas de post procesamiento.

que se quieran aplicar al objeto de la escena se calculan teniendo en cuenta que las mismas se expresan de manera inversa en el espacio de la vista, por ejemplo si se quiere girar el objeto hacia la derecha para visualizar su lado izquierdo, en realidad el sistema de la vista se gira hacia la izquierda.

3. IMPLEMENTACIÓN DE LAS OPERACIONES DE INTERACCIÓN

Para entender cómo se implementan las operaciones de interacción se necesita hacer una introducción al modelo visual que utiliza OpenGL. El sistema de coordenadas visual estándar es una terna de vectores con origen en el ojo ($\mathbf{eye}$), tiene el eje x horizontal hacia la derecha y el eje y vertical hacia arriba definido mediante un versor $\mathbf{up}$ que establece la inclinación lateral de la cabeza o la cámara. El eje z apunta hacia dentro del ojo y su dirección es la línea que une el ojo con el centro o target visual, esto es la línea eye-target. El target visual tiene una coordenada z arbitraria, solamente importa la dirección eye-target que es la que se mantiene como un versor, denotado por $\mathbf{et}$.
Para que las operaciones desarrollas sean lo más general posible, se las ideó para que soporten trabajar tanto en proyección ortogonal como en perspectiva. En el modelo visual estándar, la perspectiva se define mediante la pirámide que une el ojo virtual con la pantalla virtual. Por razones algorítmicas, para el cálculo de las proyecciones es necesario definir un plano de corte (clipping) cercano al ojo y otro alejado, los planos: near y far, que recortan la pirámide infinita, generando un tronco de pirámide o frustum que define la porción visible del espacio. Dado que ambos planos son perpendiculares a la línea visual, bastan dos distancias (positivas) al ojo: znear y zfar para definirlos.
Entre znear y zfar se mapeará todo el rango numérico de profundidades disponible para cálculos de oclusión, por lo tanto deben contener a los objetos visibles en la forma más ajustada que sea prácticamente posible. La pantalla se suele identificar con la cara del frustum que corresponde al plano near.
Dado que los objetos giran respecto del sistema de coordenadas visual, lo más apropiado parece ser el mantenimiento de una esfera contendora o bounding sphere de los objetos. De ese modo, las distancias znear y zfar se pueden calcular en base al radio de la esfera y la posición del ojo. Para poder tener tolerancias absolutas es necesario un primer escalado de todo el modelo para que quepa en una esfera de radio unitario. De ese modo se puede decir que dos puntos (ej.: ojo y punto fijo) no pueden tener una distancia menor a un determinado ϵ, independiente de las unidades de medida del modelo.

3.1. Proyección ortogonal, perspectiva y escalas

En perspectiva no tiene mucho sentido hablar de una escala (píxeles/unidad) puesto que depende de la distancia desde el ojo al plano donde se mira. En su lugar se fija el ángulo de la visual (fov = field of view, es la semiapertura). La escala se puede definir en el plano near, de modo que para el fov fijado se vea el máximo cuadrado que cabe en el viewport (ventana de $w \times h$ asignada por la GUI para el dibujo). La escala será la relación entre la menor dimensión de la ventana y la las unidades del modelo en el plano near:

$$escala = \frac{min(w,h)}{2 \cdot znear \cdot tan(fov)} \tag{1}$$

En vista ortogonal, en cambio, fov no tiene sentido pero la escala sí es constante. El ojo podría considerarse en el infinito, pero para poder seguir considerando que la distancia desde el ojo define el tamaño visual de los objetos, se utiliza la misma escala calculada en perspectiva pero aplicada en otro plano que, por razones que más adelante que se explicarán, no es el plano near sino el plano del punto fijo, que también se define más adelante.

$$escala = \frac{min(w,h)}{2 \cdot \langle \mathbf{fijo} - \mathbf{eye}, \mathbf{et} \rangle \cdot tan(fov)} \tag{2}$$

El punto fijo y el plano near deberán estar siempre del mismo lado del ojo.

En la figura 1 se pueden observar los conceptos expuestos hasta el momento que involucran el modelo visual de OpenGL. Se presenta cada uno de los espacios nombrados en 2, con sus correspondientes sistemas coordenados, en azul el sistema del espacio del modelo, en naranja el sistema del espacio de la vista, y en verde el sistema del espacio de la pantalla. También se presentan los planos de corte, near y far, los versores **up** y **et**, y el ángulo fov.

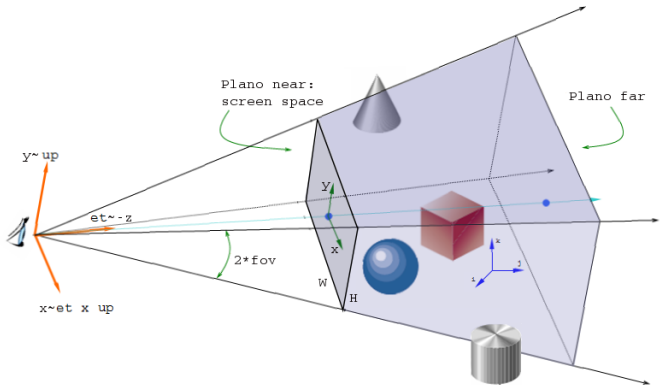

Figura 1: Detalle del modelo visual de OpenGL. La figura representa la pirámide característica de la proyección en perspectiva. Los conceptos son fácilmente extensibles al caso de la proyección ortogonal. Los planos de clipping limitan volúmen del espacio que contine a los objetos (o partes) que serán visualizados.

3.2. Operaciones a implementar

Las operaciones que se implementarán para poder interactuar con la escena son:

- Rotación: rota el objeto en torno a un punto fijo especificado.
- Pan: desplaza la imagen de la escena a través del plano de la pantalla.
- Zoom extents: posiciona el objeto de forma que se vea completo y centrado en la ventana.
- Zoom window: ocupa la ventana con un área especificada.
- Dolly dinámico: el ojo se acerca o se aleja del objeto, manteniéndo un punto fijo.
- Especificación de un punto fijo.

- Centrado del punto fijo: posiciona el punto fijo en el centro de la pantalla. Es un caso especial de la operación pan.

Todo el planteo expuesto con anterioridad tiene las siguientes objetivos:

- Simplificación en el uso y mantenimiento de unas pocas variables globales de estado que definen la vista.
- SelfTracking de las transformaciones aplicadas: la historia de $\mathbf{et}$, $\mathbf{up}$, $\mathbf{eye}$ y $tfov$[2] permite reconstruir cualquier vista anterior.
- Manejo vectorial o algebraico de cualquier operación que se quiera aplicadar a la escena. OpenGL utiliza matrices pero aquí se utilizan vectores que permiten entender inmediatamente las transformaciones de un modo geométrico. El conjunto de datos que se utiliza permite construir inmediantamente las matrices y utilizar las herramientas estándar del álgebra lineal y viceversa, de las matrices se pueden deducir las variables almacenadas trivialmente. El análisis vectorial permite, por ejemplo, identificar rápidamente el objeto que está debajo del mouse.
- Simplificación de cálculos: el cambio de la terna visual se realiza sin utilizar funciones trascendentales como las trigonométricas (implementadas en la mayoría de los casos por medio de series truncadas). En este caso cuando el usuario quiere manipular el objeto en realidad sólo se cambia la terna visual mediante operaciones algebraicas.

3.3. Cambio de sistemas de referencias entre el modelo y el visual

Debido a que la geometría a dibujar se encuentra referenciada con respecto al sistema del espacio del modelo, muchas veces será necesario transformar las coordenadas de un punto que tiene como referencia al sistema visual. La situación inversa también se hace presente en algunas situaciones.
Para efectuar estas operaciones se aplica un cambio de base más un cambio de origen. Por ejemplo: sea $\mathbf{C}$ un punto en el espacio. Suponiendo que se tienen las coordenadas de dicho punto con respecto al sistema del espacio de la vista $\mathbf{V}$, descriptas por el vector $\mathbf{C_V}$, se busca averiguar las coordenadas de $\mathbf{C}$ con respecto al sistema del modelo $\mathbf{M}$, descriptas por el vector $\mathbf{C_M}$.

El espacio del modelo está definido por la base canónica (a cuyos versores se denotará por $\mathbf{m_i}$), mientras que la base del espacio de la vista se halla conformada por los versores $\mathbf{v_0} = \mathbf{v_2} \times \mathbf{v_1}$, $\mathbf{v_1} = \mathbf{up}$ y $\mathbf{v_2} = \mathbf{et}$. Teniéndose en cuenta que el vector $\mathbf{VM}$ localiza el origen del sistema del modelo respecto del sistema de la vista, las coordenadas del punto $\mathbf{C}$ en el sistema del modelo vienen dadas por:

$$\mathbf{C_{M_i}} = \langle \mathbf{m_i}, \mathbf{v_j} \rangle \mathbf{C_{V_j}} - \mathbf{VM} \tag{3}$$

Si se observa atentamente la ecuación anterior, se puede advertir que la matriz de transición $\mathbf{T}$ de $\mathbf{V}$ hacia $\mathbf{M}$ tendrá por vectores columnas a los versores $\mathbf{v_{[i,j,k]}}$. De esta manera se puede expresar matricialmente el cambio de base:

$$\mathbf{C_M} = \mathbf{T}\ \mathbf{C_V} - \mathbf{VM} \tag{4}$$

[2] $tfov = tan(fov)$.

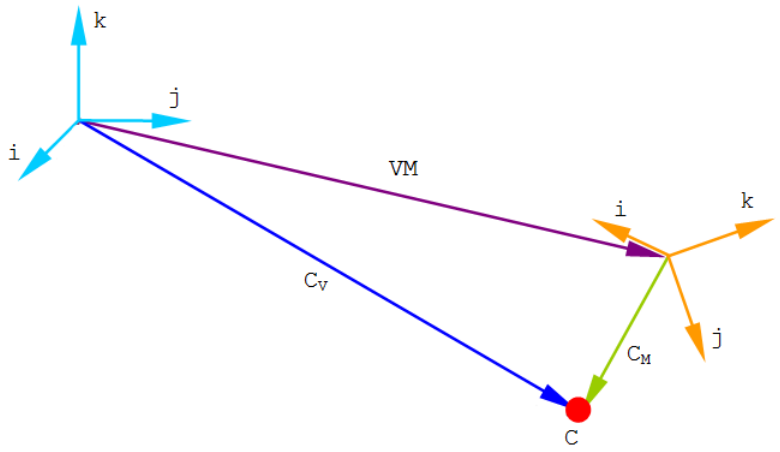

Figura 2: Conversión de coordenadas de un punto. En celeste el sistema coordenado del espacio de la vista. En naranja el sistema coordenado del espacio del modelo.

Supóngase ahora que se conocen las coordenadas del punto $\mathbf{C}$ respecto al espacio del modelo y que se quiere hallar las coordenadas del mismo en el espacio de la vista. Como se está trabajando con bases ortonormales, para resolver esta cuestión se puede hacer uso de la ecuación 4 y de la propiedad de que la inversa de una matriz cuyos vectores columna son ortonormales es igual a su transpuesta, que permite asegurar lo siguiente:

$$\mathbf{C_V} = \mathbf{T^{-1}}\ \mathbf{C_M} + \mathbf{VM} \tag{5}$$

Se han simplificado algunos cálculos en la deducción de las ecuaciones 3, 4 y 5, para que el lector pueda captar con rapidez la idea. A pesar de eso, tenga en cuenta que se están aplicando conceptos de espacios vectoriales lineales para llevar a cabo el cambio de base del vector que representa las coordenadas del punto $\mathbf{C}$ respecto a un sistema en particular, y luego se aplica una traslación (operación no aplicable a espacios vectoriales lineales, es decir, en última instacia se está aplicando una transformación afín). Es por esto que para deducir la ecuación 5 no se simplificó el cálculo a un mero pasaje de términos de la ecuación 4, con lo cual se ubiese obtenido el siguiente resultado erróneo: $\mathbf{C_V} = \mathbf{T^{-1}}\ (\mathbf{C_M} + \mathbf{VM})$.

4. SELECCIÓN DE UN PUNTO DE LA GEOMETRÍA

La selección de un punto del cuerpo geométrico por medio del cursor es esencial para muchas de las operaciones de interacción descriptas en este trabajo. Es tarea de la interfaz encontrar las coordenadas del punto en el espacio del modelo por medio de algún método.
Sea un rayo definido, en el caso de proyección ortogonal, por el cursor y el versor $\mathbf{et}$ o, en el caso de perspectiva, por el cursor y la posición del ojo. Una de las posibilidades para encontrar este punto es calcular la intersección de dicho rayo con el cuerpo geométrico, para lo cual debería implementarse una "especie" de ray tracing de superficies.
Esta última opción representa un costo algorítmico y computacional elevado, por lo que se pensó en una lo más barata posible. Utilizar la información dispuesta por el *z-buffer* mantenido por OpenGL para renderizar la escena es una excelente alternativa. Para realizar oclusiones, OpenGL mantiene, en paralelo con el buffer de color (que es el que se muestra en la pantalla) otro array de $w \times h$ números de punto flotante, llamado z-buffer o depth-buffer, en el cual co-

difica la profundidad del último pixel efectivamente dibujado. Cuando el usuario selecciona un punto se leen las coordenadas del cursor para acceder al *z-buffer*. Si la profundidad codificada $\tilde{z}$ leída es distinta de $1{,}0$ quiere decir que se ha picado sobre la geometría, caso contrario se ha picado en la "nada"(fondo).
Suponiéndo que el caso es aquel en donde el usuario picó sobre la geometría, el valor $\tilde{z}$ (comprendido entre 0 y 1) que se haya leído, más las coordenadas planares *x,y* del cursor, determinarán la posición del punto sobre el cuerpo.
El valor $\tilde{z}$, leído del *z-buffer*, tendrá que ser tratado de distinta manera dependiendo de si se está trabajando en perspectiva o en proyección ortogonal.
En los desarrollos que siguen se supone que las coordenadas *x,y* del cursor ya fueron trasladadas desde el sistema de la pantalla al sistema de la vista, es decir, por ejemplo sea x_t la coordenada trasladada que se calcula como $x_t = x_s - w/2$, donde w es el ancho de la pantalla y x_s la coordenada sin trasladar.
A continuación se va a denotar con zn al valor *znear*, con zf al valor *zfar* y con $\mathbf{pp}$ el punto picado sobre la geometría, es decir el punto que se busca calcular.

4.1. El caso de la proyección ortogonal

La transformación ortogonal de la escena acarrea una proyección paralela del modelo 3D en un plano, preservándose los tamaños relativos y los ángulos en el plano perpendicular. Así la coordenada *z*, que tiene como referencia al sistema de la vista, se mapeará linealmente dentro del volumen de visualización que queda determinado por los planos *znear* y *zfar*. Este mapeo consiste en desplazar el volumen de visualización al origen del sistema visual, y normalizar el valor resultante por la distancia existente entre el *znear* y *zfar*. Luego la coordenada *z* sufre la siguiente transformación:

$$\tilde{z} = \frac{z - zn}{zf - zn} \tag{6}$$

Por lo tanto, para averiguar el valor *z* se utiliza la ecuación 6, y se transforma la coordenada resultante al espacio del modelo, esto es:

$$z = zn + (zf - zn) \cdot \tilde{z} \tag{7}$$

$$\mathbf{pp} = \mathbf{eye} + \mathbf{et} \cdot z + (\mathbf{et} \times \mathbf{up}) \cdot (x/escala) + \mathbf{up} \cdot (y/escala) \tag{8}$$

4.2. El caso de la proyección en perspectiva

Las transformaciones en perspectiva son usadas para crear vistas que se caracterizan por tener la cámara o la posición del ojo localizada a una distancia finita de la escena. El uso de perspectiva significa que un objeto parecerá más grande a medida que se acerca al observador. En este caso no se puede mapear linealmente el valor de *z* como se hizo en el caso anterior, ya que se presentará el problema de que líneas serán mapeadas a curvas. Para eludir este problema se busca una seudo-distancia que mapee líneas en líneas y aporte información acerca de la posición relativa de los objetos en cuanto a su profundidad[3]. Una manera de conseguir esto consiste en primero desplazar el volumen de visualización al origen del sistema visual, y normalizar el valor resultante por la distancia existente entre el *znear* y *zfar*, como se hizo en el caso

[3]Se puede encontrar un desarrollo más detallado del tema en (Buss, 2005).

de la proyección ortogonal. Luego se debe multiplicar este término por un valor que exprese una seudo-distancia, que en este caso es zf/z. Como puede apreciarse este término concederá mayor influencia a objetos que se encuentren más cercanos al observador, y menor influencia a aquellos que se encuentren más alejados. Por lo tanto la coordenada z sufre la siguiente transformación:

$$\tilde{z} = \frac{z - zn}{zf - zn}\left(\frac{zf}{z}\right) \tag{9}$$

Luego, para averiguar el valor z se utiliza la ecuación 9, y se transforma la coordenada resultante al espacio del modelo, esto es:

$$f = \frac{zf}{zf - zn} \tag{10}$$

$$z = f \cdot \frac{zn}{f - \tilde{z}} \tag{11}$$

$$\mathbf{pp} = \mathbf{eye} + \mathbf{et} \cdot z + (\mathbf{et} \times \mathbf{up}) \cdot (x \cdot z)/(escala \cdot zn) + \mathbf{up} \cdot (y \cdot z)/(escala \cdot zn) \tag{12}$$

4.3. Cambio entre vista ortogonal y perspectiva

Debido a que en vista ortogonal se utiliza la misma escala[4] que en perspectiva pero aplicada en el plano del punto fijo, al cambiar entre vista ortogonal y perspectiva podría haber un movimiento aparente del punto fijo. Por eso, cuando se selecciona un nuevo punto fijo, si se está trabajando en proyección ortogonal, se debe mover el **eye** de manera que la distancia del ojo al plano que contiene el punto fijo no se vea alterada, como se observa en la figura 3.

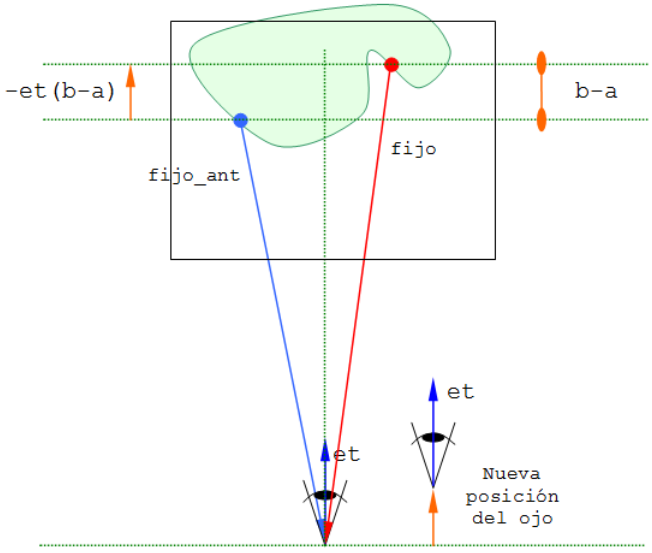

Figura 3: Para eludir un movimiento aparente del punto fijo, cuando se trabaja en proyección ortogonal y se selecciona un nuevo punto fijo, se recurre a cambiar la posición del **eye** para mantener la distancia del ojo al plano que contiene el punto fijo.

[4]Recordar el concepto de escala expuesto en 3.1.

Es decir, se recurre a los siguientes cálculos:

$$a = \langle(\mathbf{eye} - \mathbf{fijo\_ant}), \mathbf{et}\rangle \tag{13}$$

$$b = \langle(\mathbf{eye} - \mathbf{fijo}), \mathbf{et}\rangle \tag{14}$$

$$\mathbf{eye} = \mathbf{eye} - (b - a) \cdot \mathbf{et} \tag{15}$$

donde $\mathbf{fijo\_ant}$ es el punto fijo que se había seleccionado anteriormente.

5. ROTACIONES

Para la rotación se busca una forma intuitiva que mantenga fija la imagen del punto fijo. El modelo ya estándar de la esfera virtual o trackball[5] centrada en el punto fijo es un buen punto de partida.
En el modelo estándar se fija una esfera virtual de modo tal que la posición del cursor sobre la pantalla se proyecta sobre el casquete de la esfera más cercano al ojo. La esfera mantiene su centro invariante mientras gira siguiendo al cursor y al girar arrastra todo el espacio del modelo. En la figura 4 se puede apreciar la idea propuesta por el modelo del trackball.
El radio de la esfera es arbitrario y hay varias elecciones posibles. En la forma básica, el diámetro es la diagonal de la pantalla, pero eso genera movimientos muy lentos.
Una elección más útil es una fracción del mínimo entre el alto y el ancho de la pantalla. En este caso se agrega otra funcionalidad: picando fuera de la esfera se puede considerar que el punto seleccionado de la esfera es el punto que se obtiene interceptando el ecuador con la línea que une el centro y el cursor. De ese modo se facilita la definición de rotaciones de eje z: rotaciones de la imagen invariante en su plano. Mientras que como antes, picando dentro de la esfera el movimiento es el mismo que tenía el trackball, pero más rápido debido a que el diámetro es menor.
Para poder realizar estos movimientos es necesario indicar al usuario el ecuador de modo que sepa cuando gira la imagen y cuando gira en tres dimensiones.
Otra alternativa consiste en interpolar un giro 3D con uno horizontal dependiendo de la distancia al punto fijo.
Una posibilidad totalmente distinta y que es la que efectivamente se implementa, consiste en averiguar el punto del modelo debajo del cursor y mover ese punto (móvil o handle) de modo que se mantenga siempre bajo el cursor o al menos en la línea que une el cursor y el punto fijo. Las rotaciones de la imagen son sencillas e intuitivas. La rotación 3D requiere una virtualización intuitiva, puede pensarse en una esfera de radio igual a la distancia entre el punto fijo y el móvil, el punto móvil se acerca en proyección al fijo junto con el cursor y se aleja al alejarse.
La asignación de movimientos dependerá de si el punto móvil está delante o detrás del fijo. Supóngase el punto móvil más cerca del ojo que el punto fijo, cuando el cursor sobrepasa el radio máximo el punto móvil permanece en la línea que une el cursor y el punto fijo. Cuando el cursor vuelve a entrar en la esfera, el punto lo sigue, pero ahora en la dirección opuesta, por detrás, más alejado del ojo que el fijo.
Es decir, en el caso inicial, en el que el punto móvil está más cerca que el ojo, se podrá observar cómo el punto seleccionado sigue al cursor proyectado sobre la geometría. Cuando el caso es el contrario, esto es, se salió y se volvió a entrar de la esfera, ya no se podrá observar el punto

[5]Para un mejor entendimiento del modelo trackball consultar (Sotil, 2007), aunque a continuación se dará una breve descripción.

móvil, ya que el mismo sigue en proyección al cursor pero ahora desde la "cara oculta" de la geometría.
Un problema de este esquema es cuando el usuario pica sobre un punto sin modelo debajo, pero puede hacerse que en tal caso no se dé curso a la rotación o que se considere el punto móvil inicialmente en el plano determinado por la coordenada z del punto fijo. Si pica sobre el punto fijo se puede interpretar como una interacción de paneo.
La rotación se calcula mediante la intersección de la esfera con un rayo que pasa por el ojo y el cursor (el cursor se mueve en el plano near). La dirección y sentido del rayo se denotan a través del versor $\mathbf{cc}$. En vista ortogonal el ojo se considera arriba del cursor en la dirección dada por el versor $\mathbf{et}$, por lo tanto, en este caso, $\mathbf{cc} = \mathbf{et}$. Si el rayo no intercepta la esfera se considera que el punto móvil está en la intersección de la esfera con la perpendicular al rayo, en el caso de la proyección ortogonal. En perspectiva, la posición del punto móvil se calcula mediante la intersección del rayo y del plano determinado por el punto fijo y el vector $\mathbf{et}$.
El $\mathbf{eye}$ y los versores $\mathbf{et}$ y $\mathbf{up}$ giran en consonancia con el giro de la esfera, pero en forma opuesta, para que parezca que gira el modelo. El punto fijo permanece fijo y el handle se posiciona bajo el cursor.

5.1. Implementación de las rotaciones

El desarrollo que sigue a continuación tendrá en cuenta la implementación de las rotaciones tanto en perspectiva como en proyección ortogonal.
Se necesita encontrar en todo momento la posición del punto móvil o handle (denotado $\mathbf{pm}$) sobre la esfera cuyo radio va del punto fijo al punto picado al iniciar la rotación (denotado $\mathbf{pp}$). En perspectiva se define como centro $\mathbf{c}$ a la posición de la cámara o eye, mientras que en ortogonal el centro está determinado por un punto arriba del cursor. En la figura 5, se pueden observar las variables intervinientes cuando el caso considerado es la proyección ortogonal, en la figura 6 se puede observar el caso de la proyección en perspectiva.

A continuación se desarrollará solamente el caso en perspectiva, ya que, como se observará, el caso en proyección ortogonal es muy similar. Dos situaciones son las que se tienen que analizar en la rotación dependiendo de si el rayo intersecta la esfera o no, en cuyo caso determinará si es una rotación 3D o si es una rotación con respecto al eje z.

5.1.1. El rayo no intersecta la esfera

Este caso puede ser resuelto determinando la posición del punto móvil por medio de la intersección del rayo y del plano π (que contiene al punto fijo y tiene como normal $\mathbf{et}$). En proyección ortogonal se puede simplificar dicho cálculo considerando que el punto móvil está en la intersección de la esfera con la perpendicular al rayo, ya que el versor director del rayo es siempre perpendicular al plano π. Es en el caso de la proyección en perspectiva cuando se debe utilizar el concepto de la intersección del rayo con el plano π, de otra manera la rotación resultante de este cálculo no sería una rotación puramente respecto al eje z.
A continuación se expone el caso de la proyección en perspectiva. En la figura 7 se pueden observar los vectores que se definen para determinar la intersección. Considérese la siguiente

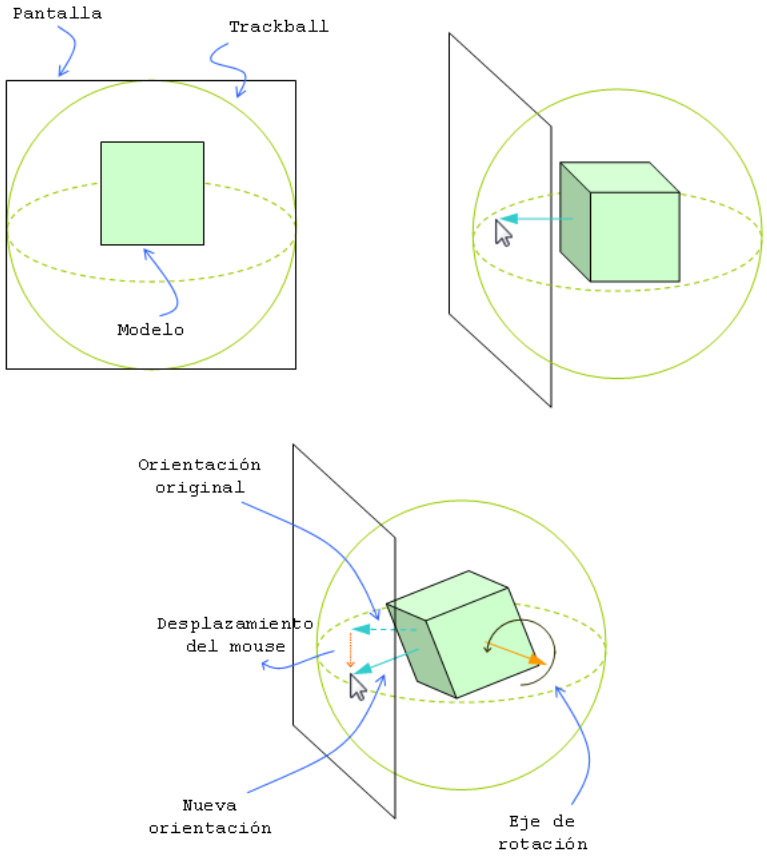

Figura 4: Modelo estándar del trackball.

definición paramétrica del rayo:

$$\mathbf{r}(t) = t \cdot \mathbf{cc} + \mathbf{c} \tag{16}$$

Luego, la intersección $\mathbf{fr}$ del rayo con el plano π se calcula por:

$$\langle \mathbf{et}, \mathbf{r}(t) - \mathbf{fijo} \rangle = 0 \tag{17}$$

$$t\langle \mathbf{et}, \mathbf{cc} \rangle + \langle \mathbf{et}, \mathbf{c} \rangle - \langle \mathbf{et}, \mathbf{fijo} \rangle = 0 \tag{18}$$

$$t = \frac{\langle \mathbf{et}, \mathbf{fijo} \rangle - \langle \mathbf{et}, \mathbf{c} \rangle}{\langle \mathbf{et}, \mathbf{cc} \rangle} \tag{19}$$

$$\mathbf{r} = \mathbf{cc} \cdot \left(\frac{\langle \mathbf{et}, \mathbf{fijo} \rangle - \langle \mathbf{et}, \mathbf{c} \rangle}{\langle \mathbf{et}, \mathbf{cc} \rangle} \right) + \mathbf{c} \tag{20}$$

$$\mathbf{fr} = \mathbf{r} - \mathbf{fijo} \tag{21}$$

En el caso de la proyección ortogonal los cálculos a realizar para determinar $\mathbf{fr}$ son los siguientes:

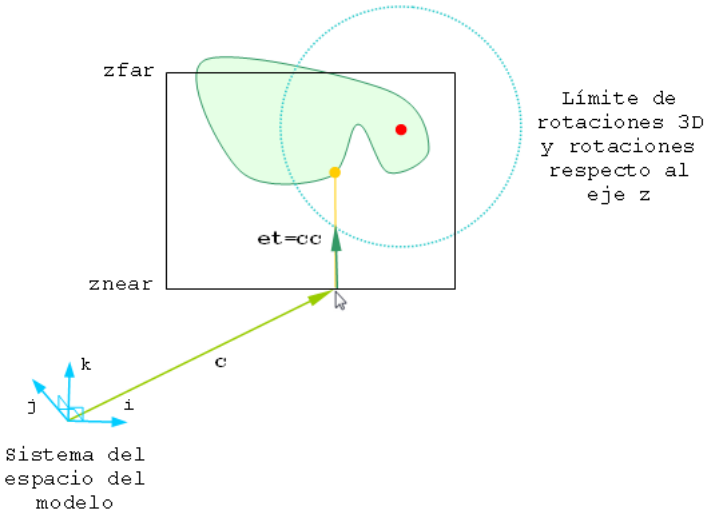

Figura 5: Definición de los vectores **c** y **cc** en proyección ortogonal. El punto central a la circunsferencia representa el punto fijo, el otro punto el handle que el usuario selecciona a través de la interacción por medio del cursor.

$$\mathbf{r1} = \mathbf{fijo} - \mathbf{c} \tag{22}$$

$$\mathbf{r} = \mathbf{c} + \langle \mathbf{r1}, \mathbf{cc} \rangle \mathbf{cc} \tag{23}$$

$$\mathbf{fr} = \mathbf{r} - \mathbf{fijo} \tag{24}$$

El vector **fr** todavía no expresa el dato que se busca (**pm**), para ello se debe encontrar un vector que tenga la dirección de **fr** y la magnitud del radio de la esfera de rotación. En la figura 8 se observa lo dicho anteriormente y a través del vector **fri** se obtiene la posición del punto móvil sobre la esfera, esto es:

$$\mathbf{fri} = \frac{\mathbf{fr}}{\|\mathbf{fr}\|} radio \tag{25}$$

$$\mathbf{pm} = \mathbf{fijo} + \mathbf{fri} \tag{26}$$

5.1.2. El rayo intersecta la esfera

En este caso la rotación se calcula mediante la intersección de la esfera con un rayo que pasa por el ojo y el cursor. En la figura 9 se puede seguir visualmente el proceso para calcular el punto móvil **pm**. Los vectores **r1**, **r** y **fr** se calculan por medio de las fórmulas (22), (23) y (24)[6]. Como paso siguiente se utiliza el vector **fr** para hallar al sistema coordenado que define la circunferencia contenida en el plano que determinan los puntos fijo y móvil sobre el cuerpo. La dirección del eje de las abscisas vendrá dado por la dirección del vector **fr**, el radio de dicha circunferencia es el de la esfera de rotación. Es necesario para los cálculos determinar la

[6]Se utilizan las fórmulas mencionadas independientemente de si se está trabajando en proyección ortogonal o en perspectiva, ya que en las rotaciones 3D el vector **fr** puede no ser paralelo al plano π. Si se utiliza el vector **fr** resultante de la intersección del rayo con el plano π, dicho vector estaría limitado a ser paralelo a dicho plano.

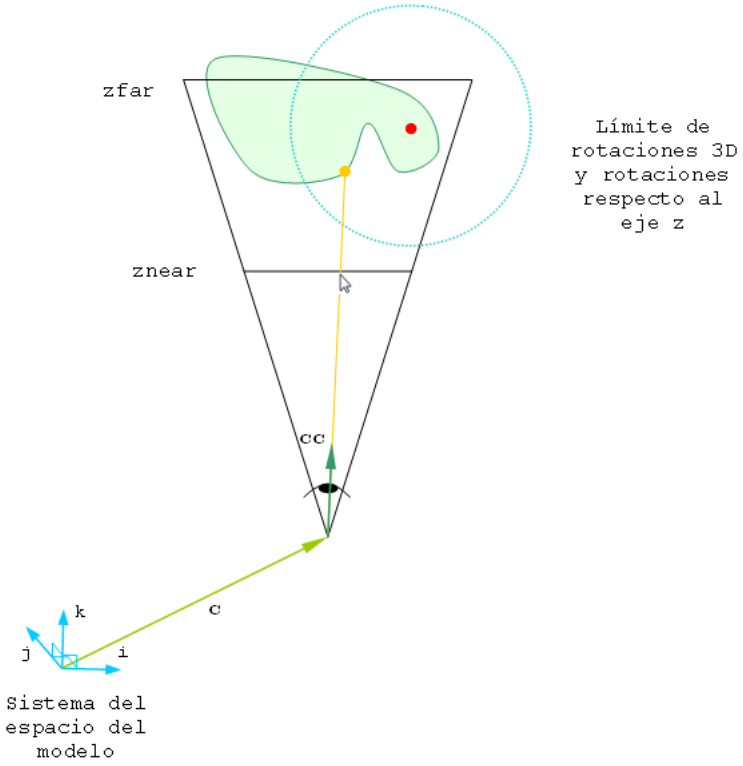

Figura 6: Definición de los vectores **c** y **cc** en perspectiva. El punto central a la circunsferencia representa el punto fijo, el otro punto el handle que el usuario selecciona a través de la interacción por medio del cursor.

ordenada ri del punto que tiene por coordenada abscisa la magnitud del vector **fr**, con lo cual se tiene:

$$ri = \pm\sqrt{\|\mathbf{fr}\|^2 - radio^2} \tag{27}$$

donde el signo de ri dependerá de si el punto móvil está delante o detrás del fijo.

A continuación se busca un vector **v** que tenga por magnitud ri, por sentido al signo de ri y por dirección la del versor **cc**. Paso siguiente se procede a calcular **pm**. Expresándo matemáticamente lo dicho, se tiene:

$$\mathbf{v} = \mathbf{cc} \cdot ri \tag{28}$$

$$\mathbf{pm} = \mathbf{r} + \mathbf{v} \tag{29}$$

5.1.3. Cálculo de la rotación propiamente dicha

Habiendo calculado la nueva posición del handle, se debe llevar **pp** a dicha posición. En este paso se giran el **eye** y los versores **et** y **up** en consonancia con el giro de la esfera, pero en forma opuesta, para que parezca que gira el modelo. El punto fijo permanece fijo y el handle se posiciona bajo el cursor. En la figura 10 se puede apreciar cuál es la situación, se tienen los vectores **pp** y **pm**, y la esfera de rotación (en este caso se representa sólo la circunferencia que

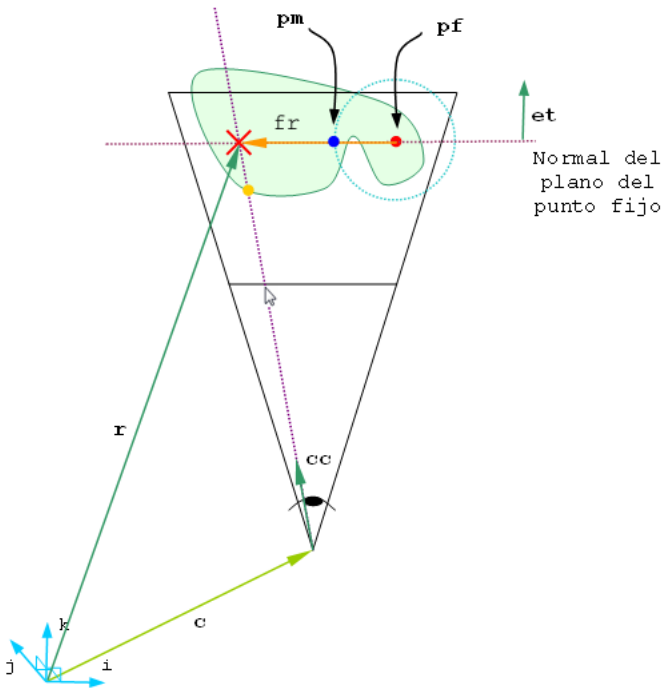

Figura 7: El rayo no intersecta la esfera en el caso de la proyección en perspectiva. Se utiliza la intersección del rayo con el plano π para calcular la posición del punto móvil en la esfera.

contiene los vectores mencionados).

Para efectuar la rotación descripta se toma un enfoque vectorial. Para ello se definen los siguientes vectores:

$$\mathbf{fp} = \frac{\mathbf{pp}}{\|\mathbf{pp}\|} \tag{30}$$

$$\mathbf{fm} = \frac{\mathbf{pm}}{\|\mathbf{pm}\|} \tag{31}$$

$$\mathbf{ng} = \mathbf{fp} \times \mathbf{fm} \tag{32}$$

$$\mathbf{bm} = \mathbf{fm} \times \mathbf{ng} \tag{33}$$

$$\mathbf{bp} = \mathbf{fp} \times \mathbf{ng} \tag{34}$$

donde los vectores $\mathbf{bm}$ y $\mathbf{bp}$ son denominados binormales.
Sean las bases $\mathbf{Vm} = \{\mathbf{ng}, \mathbf{fm}, \mathbf{bm}\}$ y $\mathbf{Vp} = \{\mathbf{ng}, \mathbf{fp}, \mathbf{bp}\}$. Luego para rotar el vector $\mathbf{pm}$ hacia $\mathbf{pp}$[7], se realiza un *cambio de base* de los vectores $\mathbf{eye}$, $\mathbf{et}$ y $\mathbf{up}$ hacia la base $\mathbf{Vm}$. Como paso siguiente se toman las coordenadas resultantes del cálculo anterior para multiplicar los vectores correspondientes de la base $\mathbf{Vp}$, así de esta manera se efectúa una *cambio de base más un cambio de sistema*[8], que se traduce en una rotación del vector $\mathbf{pm}$. Formalizando lo expuesto:

[7]Rotación que se implementará en sentido inverso, ya que se giran el ojo y los versores et y up
[8]En definitiva, se está en presencia de una transformación afín.

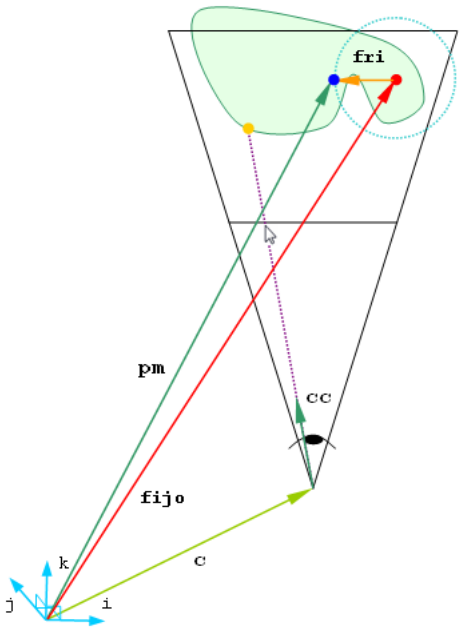

Figura 8: El rayo no intersecta la esfera en el caso de la proyección en perspectiva. Como paso final se obtiene la posición buscada $\mathbf{pm}$, que se representa por un punto verde en la figura.

$$\mathbf{et} = \langle \mathbf{et}, \mathbf{ng} \rangle \mathbf{ng} + \langle \mathbf{et}, \mathbf{fm} \rangle \mathbf{fp} + \langle \mathbf{et}, \mathbf{bm} \rangle \mathbf{bp} \tag{35}$$

$$\mathbf{up} = \langle \mathbf{up}, \mathbf{ng} \rangle \mathbf{ng} + \langle \mathbf{up}, \mathbf{fm} \rangle \mathbf{fp} + \langle \mathbf{up}, \mathbf{bm} \rangle \mathbf{bp} \tag{36}$$

Luego, para mantener la imagen del punto fijo, se mueve el vector $\mathbf{eye}$:

$$\mathbf{ef} = \mathbf{eye} - \mathbf{fijo} \tag{37}$$

$$\mathbf{ef} = \langle \mathbf{ef}, \mathbf{ng} \rangle \mathbf{ng} + \langle \mathbf{ef}, \mathbf{fm} \rangle \mathbf{fp} + \langle \mathbf{ef}, \mathbf{bm} \rangle \mathbf{bp} \tag{38}$$

$$\mathbf{eye} = \mathbf{fijo} + \mathbf{ef} \tag{39}$$

6. DOLLY

El concepto de “Dolly” es extraído de la cinematografía, en donde la cámara se acerca o se aleja de la escena sin modificar el factor de zoom[9]. En este trabajo se utiliza esta idea para implementar una operación con la cual el usuario pueda observar en más detalle ciertos aspectos de la escena o, más específicamente, de la geometría.
En la interfaz propuesta, el ojo se acerca o aleja de los objetos moviéndose por la línea que lo une con el punto fijo. De ese modo el punto fijo queda siempre en el mismo punto de la imagen. Los versores $\mathbf{up}$ y $\mathbf{et}$, y el ángulo fov permanecen inalterados. El ojo se mueve en forma conjunta con la pirámide, pero acercando o alejando los planos near y far que deben permanecer

[9]En dos dimensiones el zoom y el dolly producen el mismo efecto, en un editor de texto aumentando el factor de zoom se agrandan las letras y esto ha creado cierta confunción al trasladar el término zoom a 3D. El zoom es un cambio del águlo de la visual, fov, mientras que el acercamiento o alejamiento al objeto observado es un dolly, una traslación de la cámara sin cambiar las lentes.

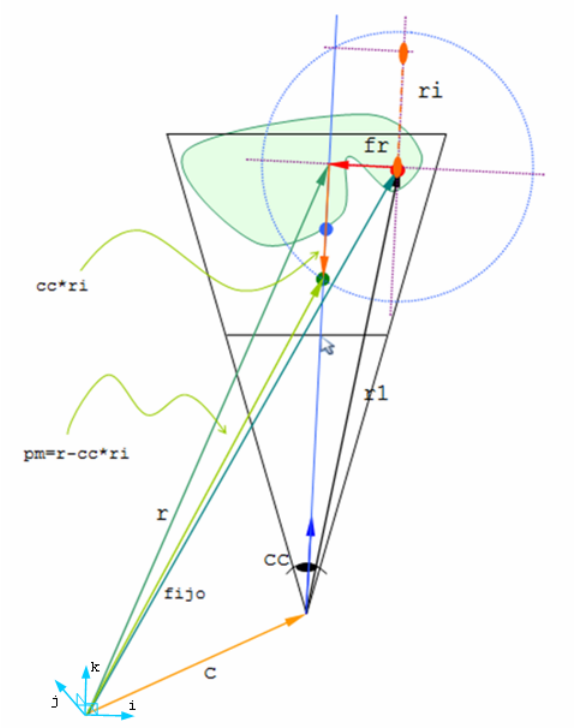

Figura 9: El rayo intersecta la esfera. Se presentan los vectores intervinientes en el cálculo del punto móvil, $\mathbf{pm}$, sobre la esfera, que se representa por un punto verde en la figura. El caso representado es aquel en que el punto móvil está más cerca del eye que el punto fijo, es por eso que se utiliza el signo negativo para calcular $\mathbf{pm}$.

ajustados siempre al modelo. En la figura 11 se puede observar la idea propuesta.

El vector $\mathbf{ef} = \mathbf{fijo} - \mathbf{eye}$ cambia de longitud, pudiendo reducirse hasta un mínimo tal que $\langle \mathbf{ef}, \mathbf{et} \rangle = \epsilon$, que es el mínimo valor admisible para znear.
Al acercarse demasiado el ojo al punto fijo algunas partes del modelo quedarán detrás del ojo y deben clippearse. Si la distancia es suficiente el plano near se mantiene tangente a la enclosing ball, pero cuando el ojo atraviesa ese plano se fija la distancia znear en el mínimo ϵ y se mantiene hasta la aproximación máxima. En todo momento el plano far se mantiene tangente a la enclosing ball por detrás.
Cuando se cambia el punto fijo por un punto visible no hay ningún problema, pero si se cambia por un centro u otro punto no visible hay que prever que puede estar detrás del ojo o a distancia menor que ϵ. En tal caso se impide el cambio. La otra opción sería admitirlo y retraer el ojo, pero la vista cambiaría en forma abrupta.

6.1. Implementación de la idea propuesta

Sea δ el dolly en píxeles por unidad, λ el factor multiplicativo a aplicar para aumentar o disminuir δ y sean w y h el ancho y el alto del viewport respectivamente. Para encontrar la nueva posición del ojo se debe modificar el módulo del vector definido por el $\mathbf{eye}$ y el punto fijo acorde a un factor de amplificación. Paso seguido se mueve el eye en consonancia al nuevo vector para que la posición del punto fijo quede inalterada. Traduciendo las palabras en vectores, se tiene:

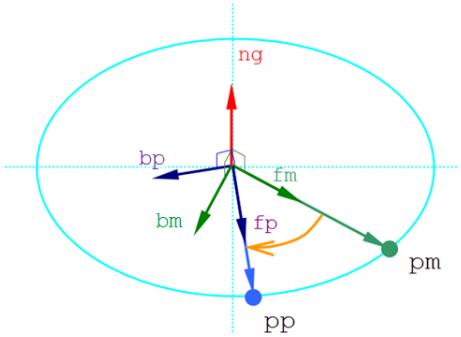

Figura 10: Cálculo vectorial de la rotación. El punto móvil se traslada a la posición del punto picado y el punto fijo permanece en su posición sin modificarse.

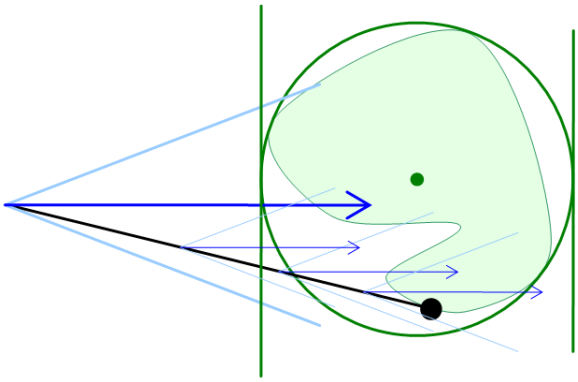

Figura 11: Implementación del dolly. En el caso que describe la figura el puntero del mouse estaría en posición coincidente con el punto fijo, si se considera que el punto negro es el mismo.

$$\mathbf{ef} = \mathbf{fijo} - \mathbf{eye} \tag{40}$$

$$efz = \langle \mathbf{ef}, \mathbf{et} \rangle \tag{41}$$

$$\mathbf{ef} = \mathbf{ef}/efz \tag{42}$$

$$efz = max(efz \cdot e^{\frac{\lambda \cdot \delta}{w+h}}, \epsilon) \tag{43}$$

$$\mathbf{eye} = \mathbf{fijo} - \mathbf{ef} \cdot efz \tag{44}$$

La exponencial se utliza para cambiar geométricamente el factor de dolly, con λ positivo.
Una vez encontrada la nueva posición del ojo, se procede a calcular, como se dijera con anterioridad, los clipping planes. Sea $\mathbf{ebc}$ el centro de la enclosing ball, luego:

$$ecz = \langle \mathbf{ebc} - \mathbf{eye}, \mathbf{et} \rangle \tag{45}$$

$$znear = max(ecz - 1, \epsilon) \tag{46}$$

$$zfar = ecz + 1 \tag{47}$$

7. PAN

El desplazamiento de los objetos o paneo se logra siguiendo al cursor mediante el movimiento de todo el sistema visual en un plano paralelo al de la pantalla. El sistema visual se mueve en forma opuesta al cursor para que parezcan movimientos del modelo que, en el sistema global, permanece fijo.
En perspectiva, para lograr que un punto del modelo, el handle, permanezca debajo del cursor durante el movimiento, habrá que corregir el desplazamiento (medido en el plano near) con el cociente entre el z del handle y znear. La coordenada z se calcula mediante la lectura del z-buffer en el punto picado; si no se picó sobre un punto del modelo el z-buffer devuelve zfar, pero se usa znear, equivale a no corregir.
Para entender lo expuesto, sean dy y dx los desplazamientos a llevar a cabo en cada una de las direcciones, y sea zp la coordenada z del punto picado. En el caso de que se esté trabajando en perspectiva, la posición del ojo se calcula de la siguiente manera:

$$\mathbf{eye} = \mathbf{eye} + (\mathbf{et} \times \mathbf{up}) \cdot (dx \cdot zc/(escala \cdot znear)) + \mathbf{up} \cdot (dy \cdot zc/(escala \cdot znear)) \quad (48)$$

Si se está trabajando en proyección ortogonal, entonces:

$$\mathbf{eye} = \mathbf{eye} + (\mathbf{et} \times \mathbf{up}) \cdot (dx/escala) + \mathbf{up} \cdot (dy/escala) \quad (49)$$

8. ZOOM WINDOW

Es interesante considerar la posibilidad de modificar el field of view (la semiapertura) o lo que es lo mismo su tangente ($tfov$). El caso en que más sentido tiene considerar poder interactuar con la semiapertura es cuando se está trabajando en perspectiva, ya que posibilitaría cambiar el aspecto de la proyección o, dicho de otro modo, posibilitaría controlar interactivamente la deformación que sufre la geometría bajo la transformación proyectiva.
Otros de los casos en el que tiene sentido modificar la semiapertura, tanto en perspectiva como en proyección ortogonal, es cuando se quiere realizar un Zoom Window a una porción del viewport. Para esta operación sería inaceptable el dolly, por lo cual se realiza cambiando el fov. Se podría pensar en resolver el problema de la misma manera como se hizo en 6, pero esta solución podría resultar en un clipping no deseado de la geometría. Es por eso que se concluye que modificar la $tfov$ es una buena alternativa.
Siguiendo entonces con la idea, se obtienen las dimensiones del área de zoom. Si las dimensiones son muy pequeñas se puede multiplicar el zoom por 2, teniéndo como centro el punto que se picó. Si no es así, se busca cuál es la dimensión (ancho o alto) que controlará el factor de zoom.[10] Luego se modifica la $tfov$ según el factor calculado y por último se hace un pan de la zona ampliada para centrarla en el centro del área seleccionada.
Para ayudar a entender como es que se modifica la $tfov$ se expone un pequeño desarrollo. Sea el ancho del área seleccionada wn la dimensión que controlará el factor de zoom, y sea w el ancho de la pantalla, luego:

$$r = \frac{w}{wn} \quad (50)$$

$$tfov = \frac{tfov}{r} \quad (51)$$

[10]Por ejemplo se podría determinar la menor dimensión como la controladora.

En la figura 12 se puede ver la mecánica propuesta. En la figura 12.(a) se tiene una vista de la geometría sin zoom, en rojo se expresa el área a zoomear. En la figura (b) se modifica la $tfov$ y en la (c) se hace el pan correspondiente para centrar el área de zoom que especificó el usuario.

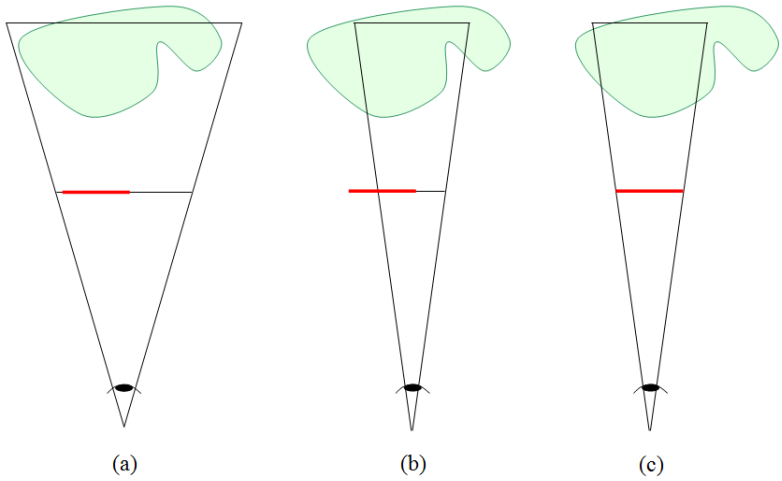

Figura 12: Etapas de un zoom window. En (a) la vista original, en rojo el área a zoomear. En (b) se modifica la $tfov$. En (c) se realiza el pan correspondiente.

9. ZOOM EXTENT

Un Zoom Extent es una tarea compuesta de dos acciones, un pan y un zoom. El pan se basa en averiguar donde está el centro de la enclosing ball y una vez hecho esto moverlo al centro de la pantalla. Para realizar el zoom hay que reestablecer la tangente de la semiapertura o $tfov$ a su valor inicial, y luego posicionar el ojo teniendo en cuenta que se quiere observar en la pantalla solamente la enclosing ball.
Suponiendo que se ha entendido como realizar el pan que se describió al principio, a continuación se expresa matemáticamente el zoom. Sea α el valor inicial de $tfov$, sea ξ un "espesor" que se agrega a la enclosing ball para poder apreciar mejor el objeto y sea ebr el radio de la enclosing ball[11], entonces se tiene:

$$tfov = \alpha \tag{52}$$

$$r = \frac{ebr}{tfov} + \xi \tag{53}$$

$$\mathbf{eye} = \frac{\mathbf{eye} - \mathbf{ebc}}{\|\mathbf{eye} - \mathbf{ebc}\|} \cdot r + \mathbf{ebc} \tag{54}$$

10. CONCLUSIONES

Se desarrollaron un conjunto de operaciones de interacción que facilitan al usuario la visualización y manipulación del modelo mediante metáforas gráficas de significado evidente, como

[11] Valores típicos de α y ξ son 0.25 y 0.2 respectivamente. El valor de ebr puede ser cualquiera, aunque podría ser 0.5 o 1.0, para poder hablar de una bola unitaria o bi-unitaria.

la rotación mediante un punto fijo de un punto móvil. Estas herramientas demostraron ser muy intuitivas desde el punto de vista del usuario y, desde el punto de vista tecnológico, novedosas. Es importante notar que la heurística introducida por las técnicas desarrolladas en este trabajo podrían ser utilizadas en otras aplicaciones que no sean estrictamente sistemas CAD. Por ejemplo, estas herramientas podrían implementarse en sistemas de pronósticos médicos por imágenes: por medio de la introducción de pantallas táctiles y la "metáfora del punto fijo y del punto móvil", el profesional de la salud haciendo uso de sus dedos índice podría interactuar con los modelos 3D de los órganos de una manera muy intuitiva y dar un veredicto lo más posible certero.

11. TRABAJOS FUTUROS

Los trabajos futuros podrían estar dirigidos a extender el concepto de **intuición** a otros tipos de herramientas de geometría computacional. Por ejemplo, herramientas que permitan el diseño de geometrías y la visualización de datos intuitivamente. Se puede mencionar a manera de anécdota, que ultimamente se está poniendo mucho esfuerzo en la visualización de datos, por ejemplo se está investigando en la generación de hologramas 3D de alta fidelidad.

REFERENCIAS

Buss S.R. *3D Computer Graphics: a Mathematical Introduction with OpenGL.* Cambridge, 2005.

Sotil W. y Calvo N. Interfaz gráfica para la visualización de datos tridimensionales restringidos a un plano variable. *ENIEF 2007 - MACI 2007*, 2007.