Antes de plantear la solución del problema es necesario conocer diversos conceptos que serán la base sobre la cual se diseñara el programa:

El flujo de trabajo en el proceso de grafica por computadora involucra 4 elementos importantes:

-ESPACIO DE OBJETO: Dominio en el cual los objetos son modelados y representados (ensamblados)

-ESPACIO MUNDIAL: Dominio en el cual los objetos son posicionados y animados a través de procedimientos matemáticos, incluye a su vez una cámara virtual u observador

-ESPACIO DE CAMARA: Una transformación del espacio mundial relativo a la cámara

-ESPACIO DE IMAGEN: Proyección del objeto del espacio de cámara en una imagen del plano

ESPACIO DE OBJETO

El segmento de datos del programa que involucra la utilización de triadas de doublewords para la representación de un punto en el espacio en números reales y triadas de words para representar un punto en plano de la pantalla

ESPACIO MUNDIAL

Eje mundial

Eje base canónico, sobre el cual se crean los puntos, ya sea en el espacio o en el plano, se define como el punto (0,0,0) en el espacio y (0,0) en el plano

Coordenadas mundiales

Coordenadas absolutas de un punto respecto al eje mundial

Coordenadas relativas

Coordenadas relativas de un punto respecto a un eje definido sobre un punto arbitrario

imagen EJEA

Vectores direccionales unitarios

Es posible representar un eje (tridimensional o bidimensional) mediante un vector unitario (i,j,k), y siguiendo una convención adecuada definir las posiciones ortogonales de dichos vectores

(imagen CEJE)

Punto

Un elemento finito en el espacio de coordenadas representado como un vector de 2 0 3 elementos. P=<px,py,pz>

Rotacion y desplazamiento de un punto en el espacio

La rotación y desplazamiento de un punto en el espacio es llevada a cabo mediante 2 metodos:

-Uso de matrices de transformación:Involucra representar un vector punto de 3 elementos en una matriz de 3x1 y multiplicarlo por una matriz de transformación predefinida (desplazamiento,rotación,etc)

Dicha matriz de transformación es una matriz de 3x3 cuyos elementos son determinados por el angulo de rotación o desplazamiento. Es decir que para computar una rotación son necesarios 3 elementos de un punto más 9 elementos de una matriz de transformación, una matriz de rotación representa un eje de rotación sobre el cual el punto debe rotar

imagen MROT

Desafortunadamente existen numerosas desventajas al momento de implementar dio método:

- Costo computacional inmenso:
  Un numero menor de operaciones de multiplicación y suma/resta es requerido
  -Espacio en memoria:
  Tanto los puntos como los vectores de rotación pueden ser representados por una única forma general q(w,x,y,z) que permitirá definir un objeto tipo quat de 4 elementos de tipo dd
  Considerando lo expuesto anteriormente se determina que se utiliza un gran número de variables (constantes,puntos, resultados intermedios, etc) y que cada numero debe ser real (para su uso en el coprocesador) ocupando 4 bytes por numero, la cantidad de datos a procesar seria demasiada
  - Ortogonalizacion
  Un requisito indispensable a la hora de computar rotaciones es que la matriz de rotación sea ortonormal, asegurar que se cumpla esta condición puede requerir mayor precisión a la disponible a la hora de calcular sus elementos
  -Inestabilidad numérica
  Como consecuencia de lo expuesto anteriormente es posible que tras un numero indeterminado de rotaciones (y procesos de redondeo) se acumulen errores numéricos que pueden llevar a distorsionar el punto
  -Gimbal Lock:
  Ocurre cuando en un proceso de rotación un eje se alinea respecto a otro dando como resultado la pérdida de un grado de libertad, tras cualquier número de rotaciones un simple giro de 90 grados sobre cualquier eje puede llevar a un gimbal lock

Una solucion alternativa es el uso de quaterniones para la rotación de un punto:

Los quaterniones son una extensión de los números reales, similar a la de los números complejos. Mientras que los números complejos son una extensión de los reales por la adición de la unidad imaginaria i, tal que $i^2 = -1$, los quaterniones son una extensión generada de manera análoga añadiendo las unidades imaginarias i, j y k a los números reales, tal que:

$$i^2 = j^2 = k^2 = ijk = -1$$

Los elementos 1, i, j y k son los componentes de la base de los quaterniones considerado como un $\mathbb{R}$-espacio vectorial de dimensión 4.

un quaternion esta compuesto de 4 elementos 1 real y 3 imaginarios del formato

q=(w,x,y,z) donde w es real y x,y,z corresponden a coordenadas en el plano

un punto P(a,b,c) puede transformarse en un quaternion de la siguiente manera

qp=(0,a,b,c)

un quaternion de rotacion sobre el eje [rx,ry,rz]se define la siguiente manera,donde donde r es 0 o 1,indicando si el giro se realiza respecto al eje correspondiente

qr=(cos[ang/2],sin[ang/2]*rx,sin[ang/2]*ry,sin[ang/2]*rz])=(qw,qx,qy,qz)

Para una rotacion de 5 grados

qk=(1,0.0436*rx,0.0436*ry,0.0436*rz)

El quaternion resultante tras la rotacion sera:

qt=(wt,xt,yt,zt)=qk*qp*qk^-1,donde qk^-1 es el inverso del quaternion qk^-1=(wk,-wx,-wy,-wz)

Donde el resultado de la multiplicacion de quaterniones q1(a,b,c,d) y q2=(i,j,k,l)es qr(n,m,o,p):

n=a*i-b*j-c*k-d*l

m=a*j+b*i+c*l-d*k

o=a*k+b*l+c*i-d*j

p=a*l+b*k+c*j-d*i

Ventajas

- Costo computacional minimo
- Estabilidad numérica:Al no tener que cumplir una restriccion como la ortogonalidad, ni tener la necesidad de ejecutar un gran numero de operaciones y utilizar un gran numero de términos, es numéricamente mas estable
- Uso intuitivo, la rotación de un vector esta claramente representada en sus elementos
  -Gimbal Lock:
  Al utilizar un enfoque de 4 dimensiones se evita la posibilidad de que se presente
- Es posible representar cada rotación única alrededor del eje con un solo quaternion denominado roll,pitch y yaw (ejes x,y,z respectivamente)

ESPACIO DE CAMARA

El problema de representar un objeto tridimensional como un objeto bidimensional ha sido estudiado por artistas muchos años antes de la invención de las computadoras, y hallaron que la proyeccion mas simple es la proyección paralela, en la cual los puntos 3D son mapeados a 2D mediante la translación de cada punto sobre una dirección de proyección (rayo) hasta que se encuentran la imagen del plano.

La vista producida es determinada por la elección de la dirección de proyección y el plano de imagen. Si el plano de imagen es perpendicular a la dirección de vista, la dirección es denominada ortográfica, de otra manera se denomina oblicua

imagen PORTHO

Para una vista ortográfica (espacio de cámara definido vectores direccionales unitarios <u,v,w> con origen e), todos los rayos tienen una dirección de –w, e inician en un plano

definido por (u,v) y el origen e, la única información faltante es donde se deben ubicar en plano de imagen.

Es posible definir el plano de imagen con 4 numeros para cada lado del plano de imagen:

l y r son las posiciones de los bordes izquierdo y derecho respectivamente, medidos desde e en la dirección de u

b y t son las posiciones de los bordes superior e inferior respectivamente, medidos desde e en la dirección de v

Como convención es posible asumir que l = – r y b = – t, por lo que solo es necesario conocer el alto y el ancho.

Para ajustar una imagen de nx × ny pixeles a un rectángulo de tamaño (r − l) × (t – b), los pixeles son espaciados una distancia (r − l)/nx horizontalmente y (t − b)/ny verticalmente, con medio pixel alrededor del borde para centrar la rejilla de pixeles dentro del rectángulo de imagen.

Esto significa que un pixel en la posición (i,j) en el espacio de objeto tiene una posición

u = l + (r − l)(i + 0.5)/nx

v = b + (t − b)(j + 0.5)/ny

Don de (u,v) son las coordenadas del pixel en plano imagen medidos con respecto al origen e y la base <u,v>

En la vista ortográfica podemos simplemente utilizar la posición del pixel en el plano imagen como el punto de origen del rayo dado que ya sabemos la dirección (-w)

El procedimiento de calculo es entonces:

-Calcular u,v con las ecuaciones de arriba

- dirección del rayo ← −w

-origen del rayo ← e + u< u> + v< v>

Notese que en la proyección ortográfica la posición en el eje z es desacartada puesto que todos los puntos tienen la misma profundidad

imagen PARPRO

ESPACIO DE IMAGEN

Lugar donde el objeto es representado en un plano 2D, en presente caso es la pantalla en modo de video

Implementacion

-ESPACIO DE OBJETO:

Conjunto de triadas datos de formato dd que representan un punto en el espacio, conjunto de diadas de datos de formato dw que representan un punto en el plano

Cuartetos de números de formato dd que representan un quaternion de rotacion

-ESPACIO MUNDIAL:

Un cubo de 2 unidades de lado, ubicado en el centro (0,0,0) , los limites son -1,1 por eje

imagen CEJE

-ESPACIO DE CAMARA:

La cámara esta ubicada sobre el eje z,en el centro del espacio mundial (0,0,0)

-ESPACIO DE IMAGEN:

La pantalla de 640*480 pixeles (640 de ancho y 480 de alto) con centro en (320,240)

Con estos datos y la ecuación de proyeccion expuesta anteriormente, tenemos:

u=640/2*(1+i)

v=480/2*(1-j)

Donde (u,v) son coordenadas en el plano imagen y (i,j) son las coodenadas (x,y) del punto en el espacio mundial

El programa esta basado en 3 algoritmos principales:

-Rotacion de un punto utilizando quaterniones

-Proyeccion de un punto del espacio mundial al espacio de imagen

-Despliegue de los puntos en pantalla

Rotacion de un punto alrededor de un eje utilizando quaterniones

Utilizando el hecho de que tanto el numero como el quaternion de rotación son tratados como un vector de 4 elementos y recordando que al tratar con elementos tipo dd es necesario escalar el numero de elemento 2 veces,el proceso es bastante directo

Calculo del elemento w de la multiplicación de quaterniones b*a donde b(b[0],b[1],b[2],b[3]) es el quaternion de rotación y a(0,a[0],a[1],a[2]) es el punto a rotar,y c (reswp, resxp, resyp, reszp) es resultado intermedio

Recordando que el coprocesador trabaja con resultados parciales (en una operación aritmética genérica los operandos se destruyen)

reswp=b[0]*0-b[1]*a[0]-b[2]*a[1]-b[3]*a[2]= b[0]*0-b[1]*a[0]-(b[2]*a[1]+b[3]*a[2])= b[0]*0-(b[1]*a[0]+(b[2]*a[1]+b[3]*a[2]))

Si consideramos a el primer punto de una cadena si=0

```
mov bx,3
shl bx,two
fld   b[bx]
mov bx,2
shl bx,two
fmul   a[si+bx]; b[3]*a[2]
```

```
fst    reswp


mov bx,2
shl bx,two
fld    b[bx]
mov bx,1
shl bx,two
fmul   a[si+bx]
fadd    reswp
fst     reswp; b[2]*a[1]+b[3]*a[2]


mov bx,1
shl bx,two
fld    b[bx]
mov bx,0
shl bx,two
fmul   a[si+bx]
fadd    reswp
fst     reswp, b[1]*a[0]+(b[2]*a[1]+b[3]*a[2])


mov bx,0
shl bx,two
fld    b[bx]
fmul   zero
fsub   reswp
fst    reswp; b[0]*0-(b[1]*a[0]+(b[2]*a[1]+b[3]*a[2]))
```

Y asi sucesivamente para el calculo de los demas elementos del quaternion intermedio,y el quaternion final

Guardando los resultados de vuelta en la misma dirección del punto

De la misma manera la proyección se realiza de manera directa tal como se establecio anteriormente

Donde a[i] es un numero real en formato dd y b[bx] en un numero entero dw que representa la posición en pantalla del punto a proyectar

```
fld factorx   ;cargando 640/2
        fld a[si]
        fld one
        fadd        ;1+x
        fmul         ;320*(1+x)
  fabs
       frndint                    ; Almacenamos la parte entera en
       fist b[bx]             ; formato entero
        add bx,2
        add si,4
        fld factory   ;cargando 480/2
        fld a[si]
        fld one
        fsub        ;1-y
        fmul        ;240*(1-y)
  fabs
       frndint                    ; Almacenamos la parte entera en
       fist b[bx]          ; formato entero
```

El despliegue de los puntos en pantalla simplemente es aplicar bresenham a los puntos en pantalla según convenga (ya sean puntos consecutivos o una secuencia prededfinida)

El cubo desplegado se rota y proyecta utilizando el eje mundial como referencia

En cambio los elementos del reloj son manejados utilizando coordenadas polares definidas en un eje relativo ubicado inicialmente en el eje mundial

Es decir que para un punto i,j en el plano x',y' del eje relativo:

i=r*cos(angulo)

j=r*sen(angulo)

Donde r es el radio de los elementos del reloj (bordes,posición de los números, las manecillas,etc)y el angulo relativo al eje del reloj

El eje relativo mencionado anteriormente se halla justo en el centro de la cara frontal del cubo de tal manera que se traslada y rota junto a esta

Entonces todos los elementos del reloj rotan de manera relativa a dicho eje y solo cuando es necesario desplegarlos son transformados a coordenadas del eje mundial

Algunos algoritmos necesarios para la construcción del reloj son:

- Calculo de los puntos que conforman un circulo (se requiere un radio y el angulo de separación entre puntos)
- Rotacion relativa de los puntos
- Lectura de la hora
- Bresenham secuencial y contiguo

Calculo de puntos que conforman un circulo

```
finit
  mov bx,0
        mov cx,360;360 puntos
        ccls:
        fild      grados
        fldpi                    ; Convierte angulo G a radianes
        fild   cmv
        fdiv                     ;pi/180
        fmul                     ;grad=g*pi/180
        fsincos                  ; calcula sen (grad) y cos (grad)
        praac rsec,pshand
        add grados,1             ;incrementa el Angulo en 1
        loop ccls
        mov grados,0
praac
fld    a
        fmul   st,st(1)
        fst b[bx]            ;i=r*cos(grad)
        add bx,4
        fld    a
        fmul   st,st(3)    ;j=r*sin(grad)
        fst b[bx]
        add bx,8
```

-Lectura de la hora

Mediante la interrupción adecuada tenemos

```
mov     ah,2ch
int     21h
mov     valbin[0],ch    ;Hora
mov     valbin[1],cl    ;Minutos
mov     valbin[2],dh    ;Segundos
mov     valbin[3],dl    ;Centesimas
```

-Bresenham por secuencia

Los puntos a unir son especificados por una secuencia predefinida en el segmento de datos

-Bresenham contiguo

Los puntos a unir están ya están definidos en el segmento de datos de forma contigua