Área visible de la escena/Volumen de vista versus Puerto de vista

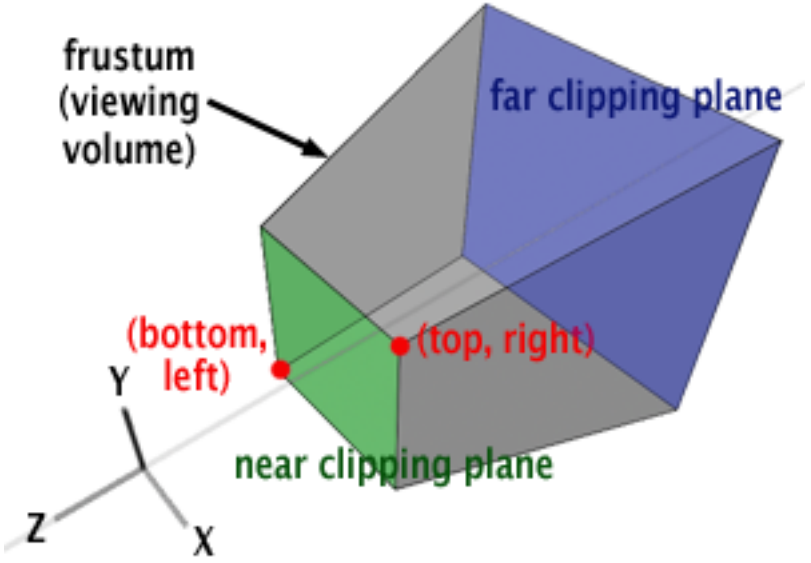
P. J. Martín, A. Gavilanes
Departamento de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid

Área visible de la escena

- El **área visible de la escena** es un rectángulo alineado con los ejes que corresponde a la región de la escena 2D (plano z=0) que se muestra en la ventana.
- Podemos colocar primitivas fuera del área visible de la escena, aunque no se verán por completo (se eliminan o recortan).
- Para delimitar el área visible de la escena usaremos 4 variables de tipo double (se miden en unidades abstractas de modelado): `xLeft`, `xRight`, `yBot`, `yTop`. Las x crecen hacia la derecha, y las y hacia arriba.

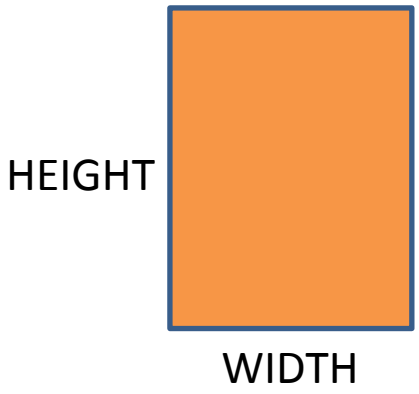
Volumen de vista

- Lo mismo vale para 3D. En este caso se habla de volumen de vista.
- El volumen de vista viene determinado por 6 variables de tipo double: `xLeft`, `xRight`, `yBot`, `yTop`,`Near`,`Far`. Las cuatro primeras tienen el mismo sentido que en 2D. Las dos últimas expresan la distancia, desde la cámara, al plano cercano y al plano lejano.



Puerto de vista

- El **puerto de vista** es un rectángulo alineado con los ejes sobre el área cliente de la ventana en la que OpenGL dibuja.
- Para delimitar el puerto de vista usaremos 4 variables de tipo integer (se miden en píxeles): `xInit`, `yInit` , `WIDTH`, y `HEIGHT`. Se miden desde la esquina inferior izquierda del área cliente. Las x crecen hacia la derecha, y las y hacia arriba.
- Generalmente tendremos que el puerto de vista ocupa todo el área cliente de la ventana, así que tendremos `xInit=yInit=0`, y `WIDTH` y `HEIGHT` corresponderán al tamaño del área cliente.



Configuración del AVE/VV

- Para fijar el área visible de la escena:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(xLeft, xRight, yBot, yTop);
```

- Para fijar el volumen de vista:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
```

Y a continuación se opta por una de las siguientes tres posibilidades. Para obtener proyección ortográfica:

```
glOrtho(xLeft, xRight, yBot, yTop, Near, Far);
```

Para proyección perspectiva, con 6 parámetros:

```
glFrustum(xLeft, xRight, yBot, yTop, Near, Far);
```

Para proyección perspectiva, con cuatro parámetros:

```
gluPerspective(fovy, aspect, Near, Far);
```

Configuración del puerto de vista

- Para fijar el puerto de vista:

```
glViewport(xInit, yInit, WIDTH, HEIGHT);
```

- Para fijar el puerto de vista ocupando todo el área cliente de la ventana:

```
glViewport(0, 0, WIDTH, HEIGHT);
```

Transformación desde el área visible de la escena al puerto de vista

- OpenGL representa todas las primitivas que caen dentro del área visible de la escena (recortándolas si fuera preciso) en el puerto de vista. Para ello aplica una transformación basada en el uso de escalas:

$$\mathbf{escala_{ancho}} = \frac{\mathbf{WIDTH}}{\mathbf{xRight - xLeft}}$$

$$\mathbf{escala_{alto}} = \frac{\mathbf{HEIGHT}}{\mathbf{yTop - yBot}}$$

- Para transformar un punto de la escena con coordenadas abstractas (x, y) en un píxel del puerto de vista con coordenadas enteras (x', y'), OpenGL usa:

$$\mathbf{x'} = (\mathbf{GLint})(\ \mathbf{escala_{ancho}} * (\mathbf{x - xLeft})) + \mathbf{xInit}$$

- Esta ecuación puede simplificarse si xInit=0. Para la coordenada y se procede de forma análoga.

Cómo evitar deformaciones

- Las primitivas que se incluyan en el área visible de la escena se representarán sin deformaciones si las escalas de ancho y alto coinciden:

$$\textbf{No deformaciones} \Leftrightarrow \textbf{escala}_{\textbf{ancho}} = \textbf{escala}_{\textbf{alto}}$$

- Esto equivale a que el aspecto (ancho/alto) del área visible de la escena coincida con el aspecto del puerto de vista:

$$\textbf{No deformaciones} \Leftrightarrow \frac{\textbf{xRight} - \textbf{xLeft}}{\textbf{yTop} - \textbf{yBot}} = \frac{\textbf{WIDTH}}{\textbf{HEIGHT}}$$

El método resize

- Al modificar el puerto de vista para que ocupe todo el área cliente de la nueva ventana, se actualizan las escalas. Es posible que se pierda la igualdad entre las escalas de ancho y alto, lo que daría lugar a deformaciones.
- Para evitarlas, modificaremos el área visible de la escena con el objetivo de recuperar la igualdad de escalas.
- Se compara el aspecto del puerto de vista actual con el del área visible antigua:
  - Si es mayor, haremos que el aspecto del área visible crezca:
    - ✓ Aumentando su numerador: `xRight-xLeft` crece
    - ✓ Disminuyendo su denominador: `yTop-yBot` disminuye
  - Si es menor, haremos que el aspecto del área visible decrezca:
    - ✓ Disminuyendo su numerador: `xRight-xLeft` disminuye
    - ✓ Aumentando su denominador: `yTop-yBot` crece
- En el esqueleto se opta por que crezca el área visible de escena.

El método resize

- Por ejemplo, para crecer el ancho del área visible de la escena:
  1. Se calcula el nuevo ancho despejando:

$$\mathbf{nuevoAncho} = (\mathbf{yTop} - \mathbf{yBot}) \times \frac{\mathbf{WIDTH}}{\mathbf{HEIGHT}}$$

  2. Se reparte a ambos lados del centro antiguo del área visible de la escena (así mantenemos el centro antiguo):

$$\mathrm{centroX} = \frac{\mathrm{xLeft} + \mathrm{xRight}}{2.0}$$

$$\mathbf{xLeft} = \mathbf{centroX} - \frac{\mathbf{nuevoAncho}}{\mathbf{2.0}}$$

$$\mathbf{xRight} = \mathbf{centroX} + \frac{\mathbf{nuevoAncho}}{\mathbf{2.0}}$$

Embaldosado (Tiling)

- Usamos varios puertos de vista sobre la misma ventana.
- Representamos nuestras primitivas una vez en cada puerto de vista.

```
void embaldosar(int nCol){
  GLdouble SVAratio= (xRight-xLeft) / (yTop-yBot);
  GLdouble w= (GLdouble) WIDTH / (GLdouble) nCol;
  //La altura de cada puerto se calcula proporcionalmente
  GLdouble h= w/SVAratio;

  for(GLint c= 0; c<nCol; c++){
    GLdouble currentH = 0;
    while((currentH + h) <= HEIGHT){
       glViewport((GLint)(c*w), (GLint)currentH, (GLint)w, (GLint)h);
       drawScene(); //dibujar la escena
       currentH += h;
    }//while
  }//for
}
```

Embaldosado (Tiling)

- El método **drawScene()** dibujaría la escena.
- El callback `display` debe poder actuar de dos formas distintas:

```
void display(void){
  glClear( GL_COLOR_BUFFER_BIT );

  if (baldosas) embaldosar(nCol); //modo baldosas
  else drawScene();               //modo normal

  glFlush();
  glutSwapBuffers();
}
```

- El modo `baldosas` se activa desde el callback `key` al pulsar la tecla correspondiente.
- Al desactivar el modo `baldosas` debemos recuperar un único puerto de vista que ocupe todo el área cliente de la ventana. Usaremos el método **desembaldosar()**.

Escalación (Zoom)

- No modifica el puerto de vista.
- Aplica un zoom sobre el área visible de la escena.
- Se modifica el tamaño del área visible de la escena, pero no el centro.
- Se implementa un zoom uniforme modificando las dos escalas (ancho y alto): lo habitual es multiplicar ambas escalas por un mismo factor `f>0`:

```
escalaAncho’ = f * escalaAncho;
escalaAlto’ = f * escalaAlto;
```

Escalación (Zoom)

- Escalaciones (II):
  - Las escalas se modifican cambiando su ancho y alto:

$$\mathbf{nuevoAncho} = (\mathbf{xRight} - \mathbf{xLeft})/f$$

$$\mathbf{nuevoAlto} = (\mathbf{yTop} - \mathbf{yBot})/f$$

  - El nuevo ancho se reparte a ambos lados del antiguo centro del área visible (así nos alejamos o acercamos del centro antiguo). Ídem para el nuevo alto.

Factor	Escala	Área visible	Zoom
$\mathbf{f > 1}$	Aumenta	Disminuye	Acercamiento
$\mathbf{f < 1}$	Disminuye	Aumenta	Alejamiento
$\mathbf{f = 1}$	Se mantiene	Se mantiene	No hay zoom

  - A veces se habla en términos de porcentajes: a un porcentaje $\mathrm{p} = 200\%$ le correspondería un factor $\mathrm{f} = \mathrm{p}/100 = 2$.

Zoom progresivo

- Hacemos variar desde 1 hasta el factor definitivo $\mathbf{factor} > 0$, en pasos discretos (nIter>0):

```
GLdouble fIncr= (factor-1)/(GLdouble)nIter;
for(int i=0;i<=nIter;i++){
   GLdouble fAux= 1 + fIncr*i;
   GLdouble anchoNew= ancho/fAux; GLdouble altoNew= alto/fAux;

   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(centroX-anchoNew/2.0, centroX+anchoNew/2.0,
              centroY-altoNew/2.0, centroY+altoNew/2.0);

   display(); //glutPostRedisplay(); no funciona!
   Sleep(50);
 }
 //Falta actualizar xLeft, xRight, yBot, yTop
```