



# ШУТИС, Мэдээлэл Холбооны Технологийн Сургууль

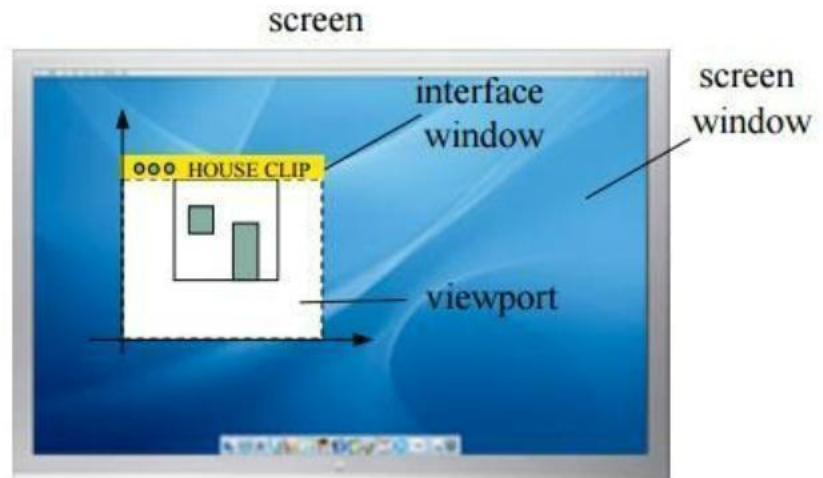
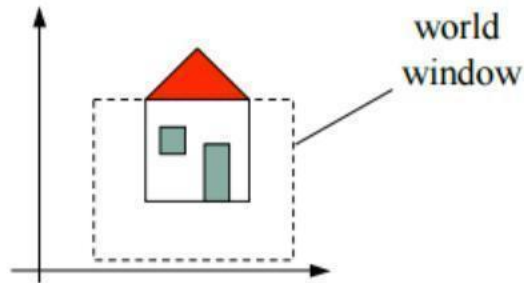
*F.CS209 Компьютерийн график*

Лекц 9 – Clipping 1  
Боловсруулсан багш: Х.Хулан

2022 он

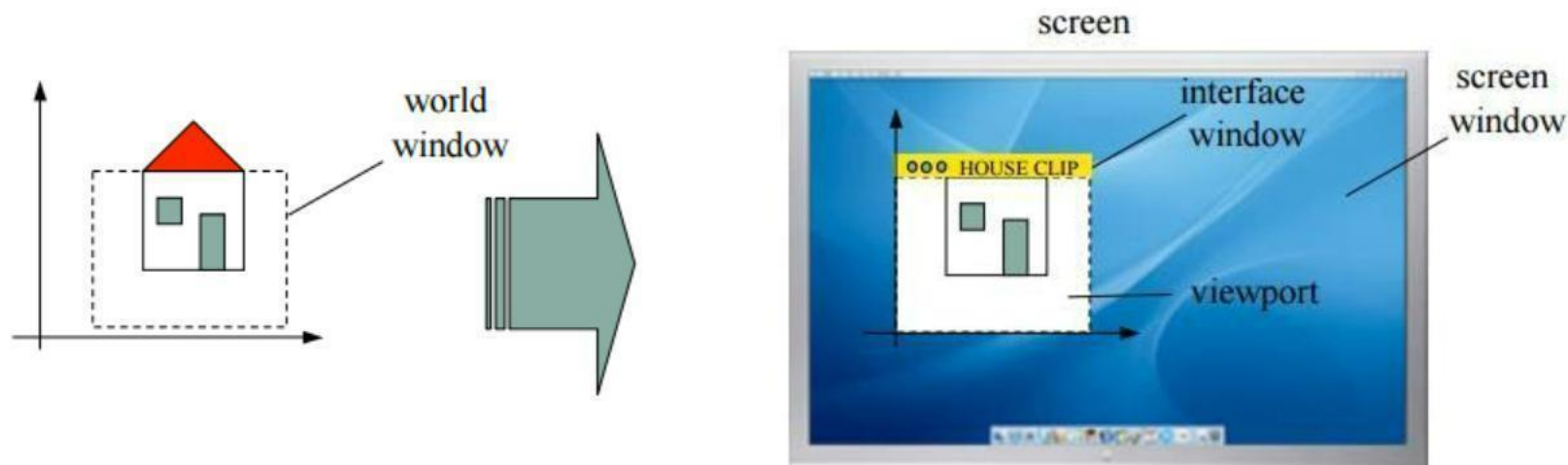
# Нэр томъёо

- ▶ **World Window** (Object Subspace) –дэлгэцэнд харуулах world хэсгийг тодорхойлох тэгш өнцөгт
- ▶ **World Coordinate System** (Object Space) –application model тодорхойлсон орон зай;
- ▶ **Interface Window, Image Coordinate System** (Image Subspace) – дэлгэцэнд харуулах зургийн орон зай
- ▶ **Screen Coordinate System** (Image Space) –дэлгэцэнд харуулсан зургийн орон зай;



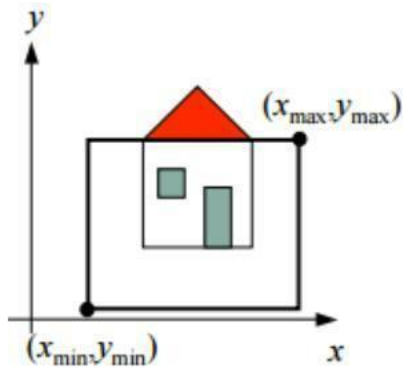
# Нэр томъёо

- ▶ **Viewport (Image Subspace)** цонхноос үзэгдэх зургийн орон зай дахь тэгш өнцөгт тайлбай
- ▶ **Viewing Transformations** –world координат дахь window-с зургийн координат (image coordinates) дахь viewport руу шилжих үйл явц

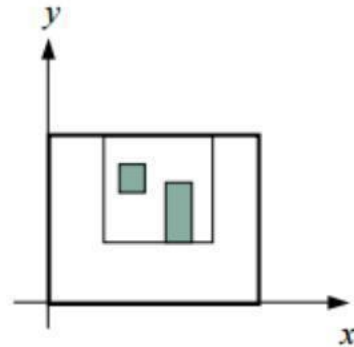


# Window-Viewport Mapping

- ▶ window болон viewport өгөгдсөн бол window-г world координатаас дэлгэцийн координат (screen coordinates) дахь viewport руу хувиргах хувиргалтын матриц ямар байх вэ? Тус матрицыг дараах зургуудын дарааллаар гурван-алхамт хувиргалт хэлбэрээр өгч болно:

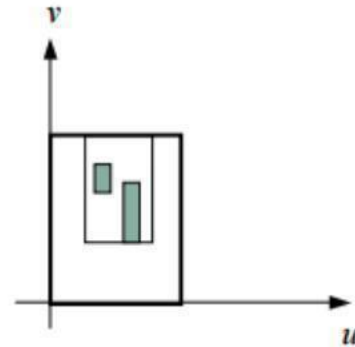


window  
in world  
coordinates



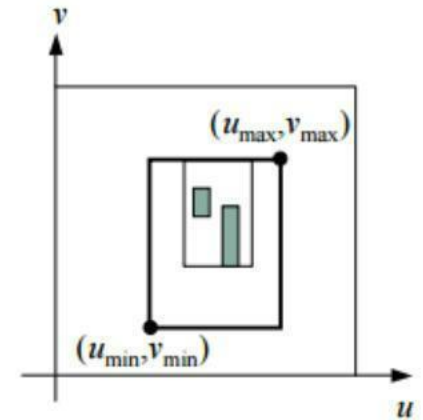
window  
translated  
to origin

$$T(-x_{\min}, -y_{\min})$$



window  
scaled to size  
of viewport

$$S\left(\frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}}, \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}}\right)$$



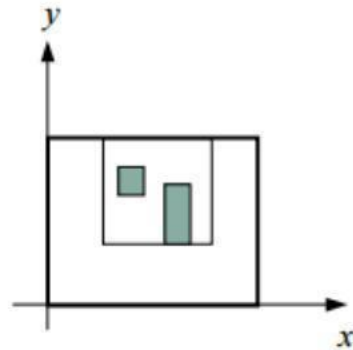
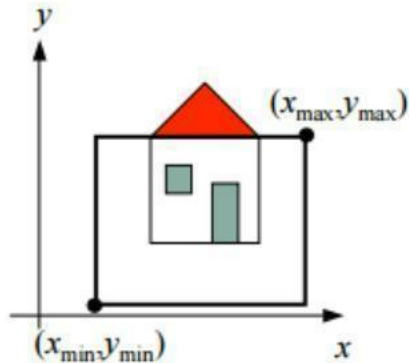
translated by  
( $u_{\min}, v_{\min}$ )  
to final position

$$T(u_{\min}, v_{\min})$$

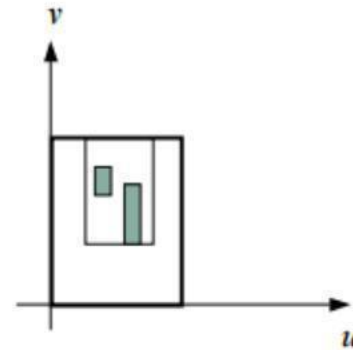


# Window-Viewport Mapping

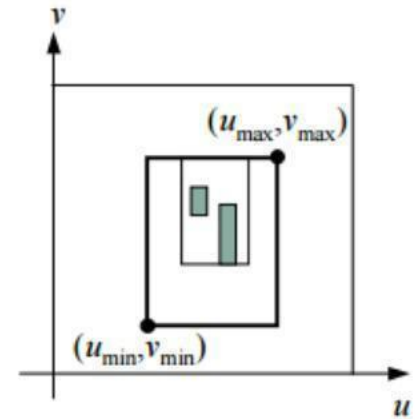
## matrix representation



$T(-x_{\min}, -y_{\min})$



$S\left(\frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}}, \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}}\right)$



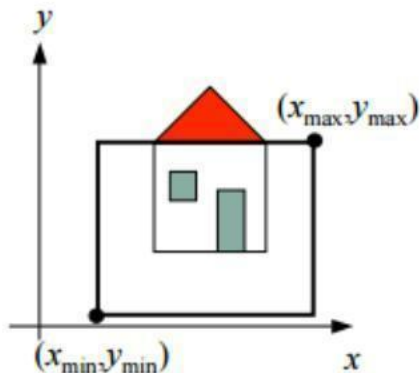
$T(u_{\min}, v_{\min})$

$$M_{wv} = T(u_{\min}, v_{\min}) \cdot S\left(\frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}}, \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}}\right) \cdot T(-x_{\min}, -y_{\min})$$

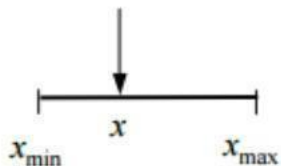
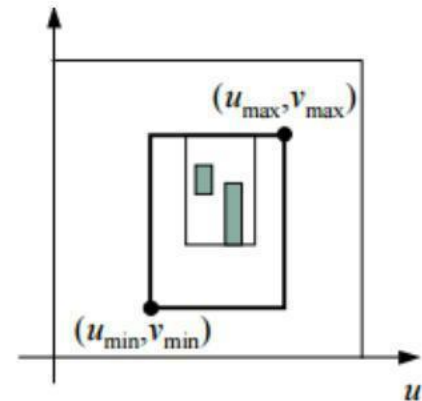
$$= \begin{bmatrix} 1 & 0 & u_{\min} \\ 0 & 1 & v_{\min} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}} & 0 & 0 \\ 0 & \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_{\min} \\ 0 & 1 & -y_{\min} \\ 0 & 0 & 1 \end{bmatrix}$$

# Window-Viewport Mapping

► Үүнийг хэрхэн хийх вэ?

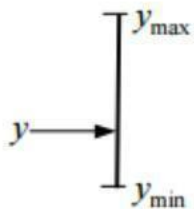
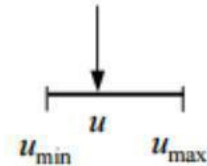


**Keeping proportionality in mapping  $(x,y)$  to  $(u,v)$**

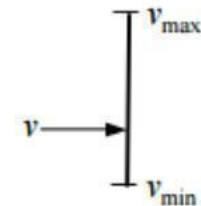


$$\frac{x - x_{\min}}{x_{\max} - x_{\min}} = \frac{u - u_{\min}}{u_{\max} - u_{\min}} \Leftrightarrow u = (x - x_{\min}) \cdot \frac{u_{\max} - u_{\min}}{x_{\max} - x_{\min}} + u_{\min}$$

*translation*      *scaling*      *translation*

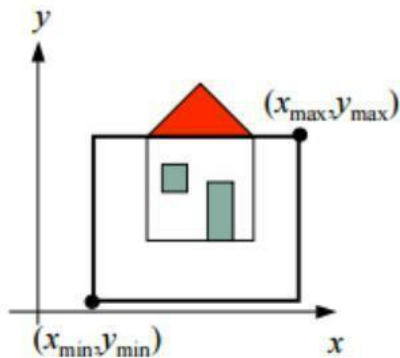


$$\frac{y - y_{\min}}{y_{\max} - y_{\min}} = \frac{v - v_{\min}}{v_{\max} - v_{\min}} \Leftrightarrow v = (y - y_{\min}) \cdot \frac{v_{\max} - v_{\min}}{y_{\max} - y_{\min}} + v_{\min}$$

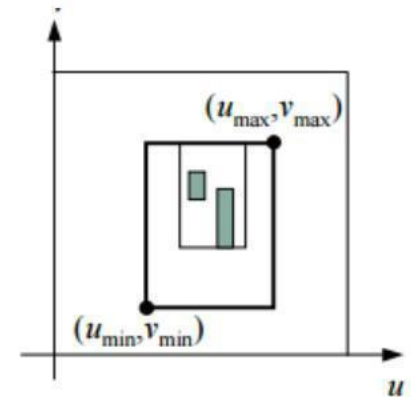


# Window-Viewport Mapping

## ► Жишээ



window(10.0,2.0,40.0,30.0)



viewport(100,50,250,300)

$$u = (x - 10.0) \cdot \frac{250 - 100}{40.0 - 10.0} + 100 \quad \lambda_x = \frac{250 - 100}{40.0 - 10.0} = 5.0$$

$$v = (y - 5.0) \cdot \frac{300 - 50}{30.0 - 5.0} + 50 \quad \lambda_y = \frac{300 - 50}{30.0 - 5.0} = 10.0$$





# Window-Viewport Mapping

---

## in OpenGL

### ■ `gluOrtho2D(left, right, bottom, top)`

- Sets up a 2-D orthographic viewing region or *world window*. Defined by two vertical clipping planes `left` and `right` and two horizontal clipping planes `bottom` and `top`.
- The *world window* by default is `(-1,1,-1,1)`.
- Defines a 2-D orthographic projection matrix.
- Sets up the window-viewport mapping, being the viewport defined by the following function:

### ■ `glViewport(x, y, width, height)`

- Sets up the viewport in the *interface window*, where `x,y` specify the lower left corner, and `width, height` its dimensions.
- By default, it uses the whole graphics area of the interface window.
- There may be various viewports inside the interface window.





# OpenGL 2D Viewing

---

OpenGL дахь 2D Viewing тодорхойлолт:

- Стандарт загвар, нэр томъёог дагаж мөрддөг

Эхлээд проекц тодорхойлно

Projection Matrix (ModelView matrix оронд):

```
glMatrixMode (GL_PROJECTION) ;
```



# OpenGL 2D Viewing 2

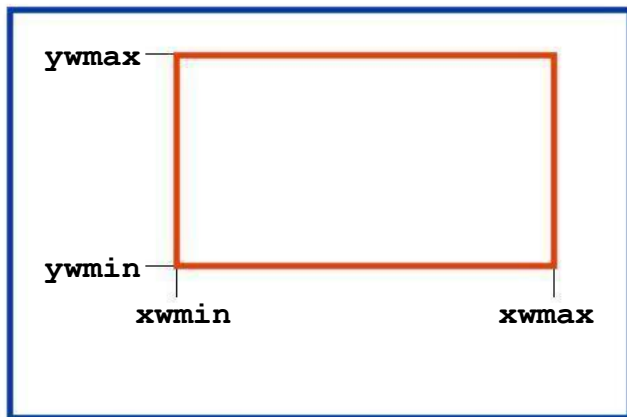
---

Дараа нь 2D clipping window тодорхойлно:

```
gluOrtho2D(xwmin, xwmax, ywmin, ywmax);
```

**xwmin, xwmax:** хэвтээ тэнхлэгийн муж, world coordinates

**ywmin, ywmax:** босоо тэнхлэгийн муж, world coordinates



# OpenGL 2D Viewing 3

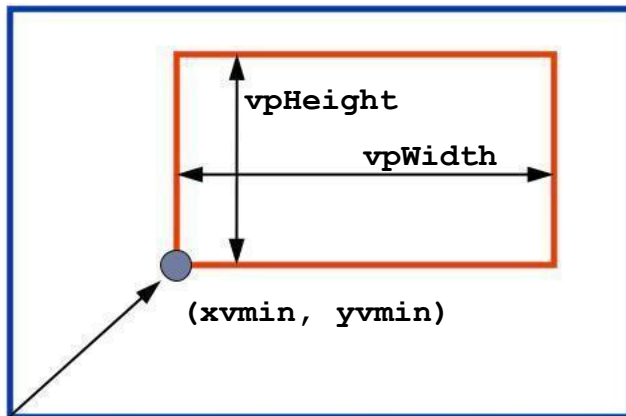
---

Эцэст viewport тодорхойлно:

```
glViewport(xvmin, yvmin, vpWidth, vpHeight);
```

**xvmin, yvmin:** зүүн доод булангийн координат (in pixel coordinates);

**vpWidth, vpHeight:** өргөн ба өндөр (in pixel coordinates);



# OpenGL 2D Viewing 4

---

ТОВЧХОНДОО:

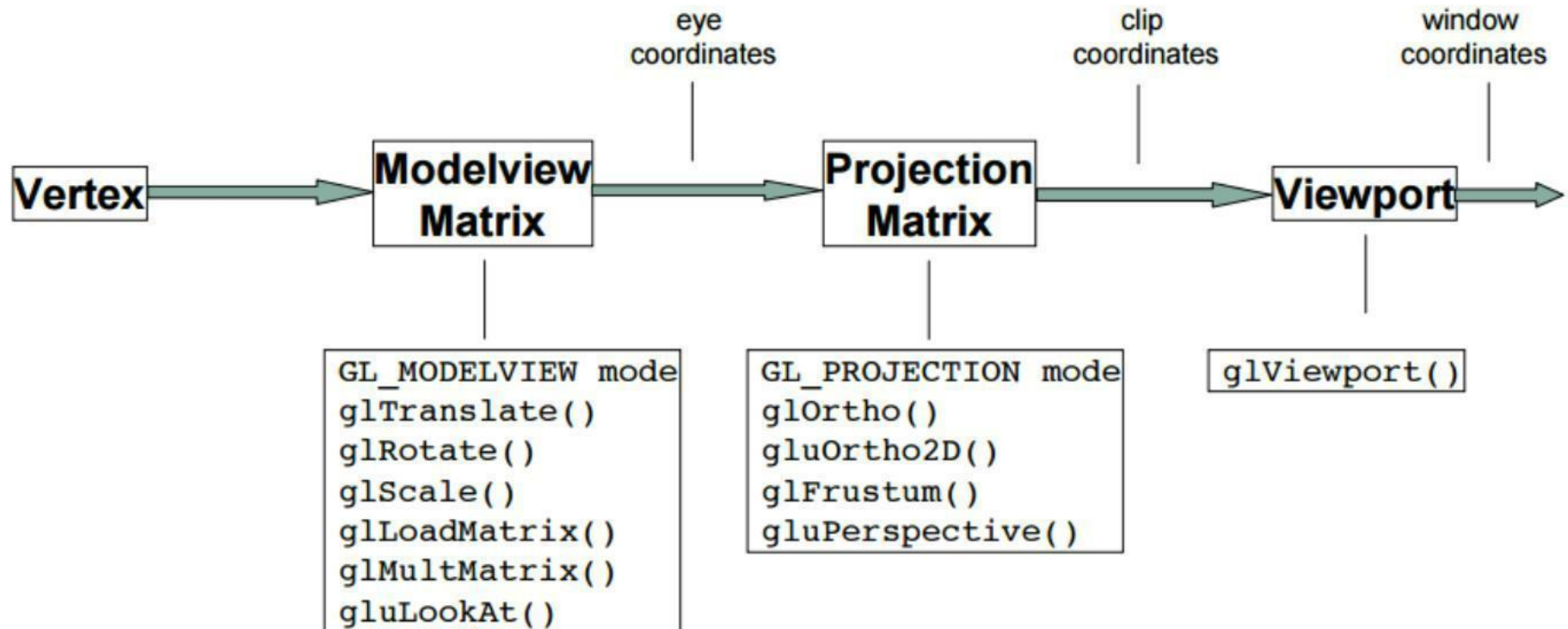
```
glMatrixMode(GL_PROJECTION);  
gluOrtho2D(xwmin, xwmax, ywmin, ywmax);  
glViewport(xvmin, yvmin, vpWidth, vpHeight);
```

Гажуудал үүсгэхгүйн тулд үүнийг шалгаж үзэх хэрэгт  
эй:

$$(ywmax - ywmin) / (xwmax - xwmin) = vpWidth / vpHeight$$



# Pipeline of OpenGL Transformations



# OpenGL-ийн жишээ

## default viewport

```
/* * WV-defaultViewport.cc - Using the default viewport * Abel Gomes */
#include <OpenGL/gl.h>           // Header File For The OpenGL Library
#include <OpenGL/glu.h>          // Header File For The GLu Library
#include <GLUT/glut.h>           // Header File For The GLut Library
#include <stdlib.h>

void draw(){
    // Make background colour yellow
    glClearColor( 100, 100, 0, 0 );
    glClear ( GL_COLOR_BUFFER_BIT );

    // Sets up the PROJECTION matrix
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,50.0,-10.0,40.0); // also sets up world window

    // Draw BLUE rectangle
    glColor3f( 0, 0, 1 );
    glRectf(0.0,0.0,10.0,30.0);

    // display rectangles
    glutSwapBuffers();

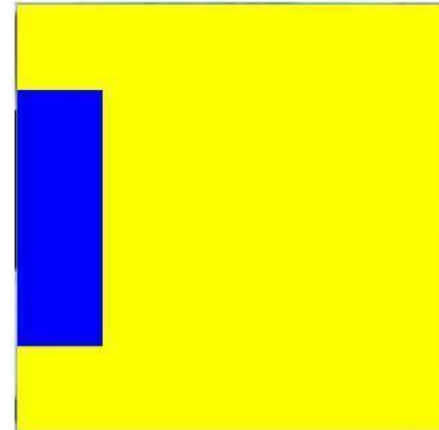
} // end of draw()
```

# OpenGL-ийн жишээ

## default viewport (cont.)

```
// Keyboard method to allow ESC key to quit
void keyboard(unsigned char key,int x,int y)
{
    if(key==27) exit(0);
}

int main(int argc, char ** argv)
{
    glutInit(&argc, argv);
    // Double Buffered RGB display
    glutInitDisplayMode( GLUT_RGB | GLUT_DOUBLE);
    // Set window size
    glutInitWindowSize( 500,500 );
    glutCreateWindow("Default viewport spans the whole interface window");
    // Declare the display and keyboard functions
    glutDisplayFunc(draw);
    glutKeyboardFunc(keyboard);
    // Start the Main Loop
    glutMainLoop();
    return 0;
}
```





# OpenGL-ийн жишээ

## single viewport

```
/* * WV-singleViewport.cc - Using a single viewport * Abel Gomes */
#include <OpenGL/gl.h>           // Header File For The OpenGL Library
#include <OpenGL/glu.h>          // Header File For The GLu Library
#include <GLUT/glut.h>           // Header File For The Glut Library
#include <stdlib.h>

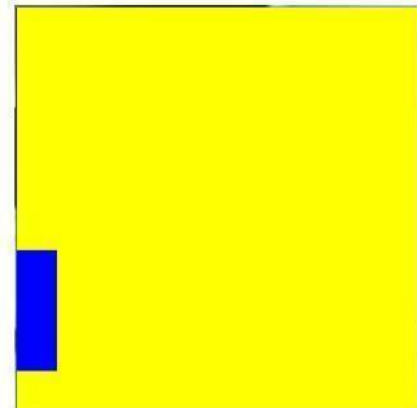
void draw(){
    // Make background colour yellow
    glClearColor( 100, 100, 0, 0 );
    glClear ( GL_COLOR_BUFFER_BIT );

    // Sets up viewport spanning the left-bottom quarter of the interface window
    glViewport(0,0,250,250);
    // Sets up the PROJECTION matrix
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,50.0,-10.0,40.0); // also sets up world window

    // Draw BLUE rectangle
    glColor3f( 0, 0, 1 );
    glRectf(0.0,0.0,10.0,30.0);

    // display rectangles
    glutSwapBuffers();
}
```

// end of draw()



# OpenGL-ийн жишээ

## two viewports

```
/* * WV-twoViewports.cc - Using two viewports * Abel Gomes */
#include <OpenGL/gl.h>           // Header File For The OpenGL Library
#include <OpenGL/glu.h>          // Header File For The GLu Library
#include <GLUT/glut.h>           // Header File For The GLut Library
#include <stdlib.h>

void draw(){
    // Make background colour yellow
    glClearColor( 100, 100, 0, 0 );
    glClear ( GL_COLOR_BUFFER_BIT );

    // Sets up FIRST viewport spanning the left-bottom quarter of the interface window
    glViewport(0,0,250,250);
    // Sets up the PROJECTION matrix
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,50.0,-10.0,40.0); // also sets up world window

    // Draw BLUE rectangle
    glColor3f( 0, 0, 1 );
    glRectf(0.0,0.0,10.0,30.0);
```



# OpenGL-ийн жишээ

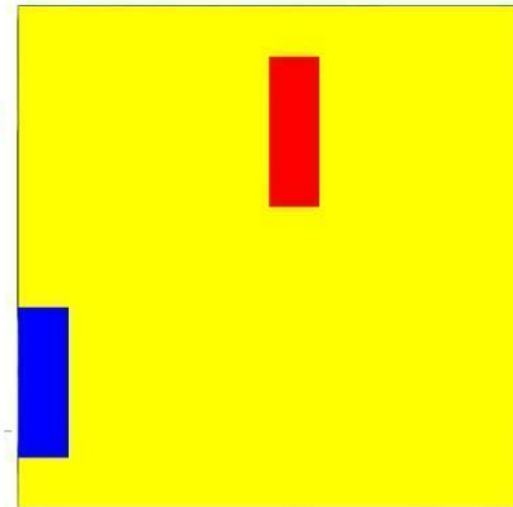
## two viewports (cont.)

```
/* rest of the function draw() */

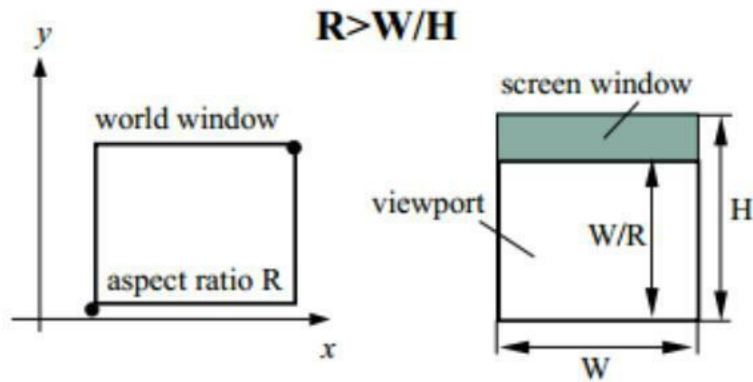
    // Sets up SECOND viewport spanning the right-top quarter of the interface window
    glViewport(250,250,250,250);
    // Sets up the PROJECTION matrix
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,50.0,-10.0,40.0); // also sets up world window

    // Draw RED rectangle
    glColor3f( 1, 0, 0 );
    glRectf(0.0,0.0,10.0,30.0);

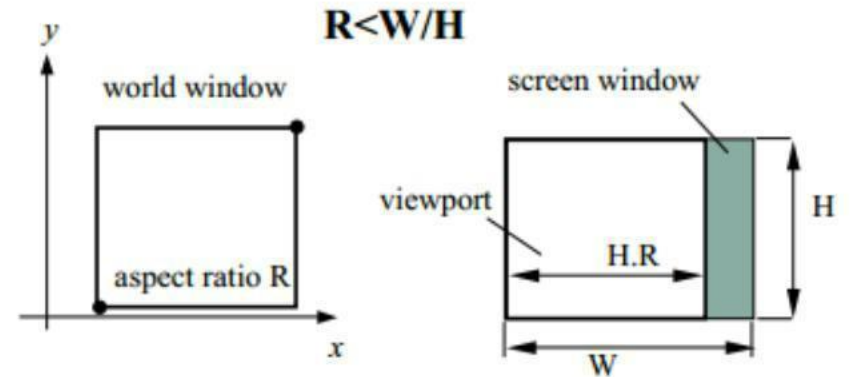
    // display rectangles
    glutSwapBuffers();
}                                     // end of draw()
```



# Viewport автоматаар тохируулах



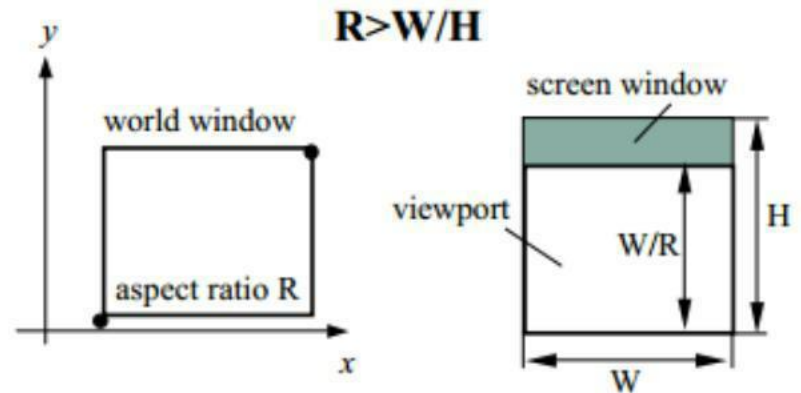
```
glViewport(0,0,W,W/R);
```



```
glViewport(0,0,H*R,H);
```

# Example: short window

- If the world window has  $R=2.0$  and the screen has  $H=200$  and  $W=360$ , then  $W/H=1.8$ .
- Therefore, we fall in first case, and the viewport is set to 180 pixels high and 360 pixels wide.



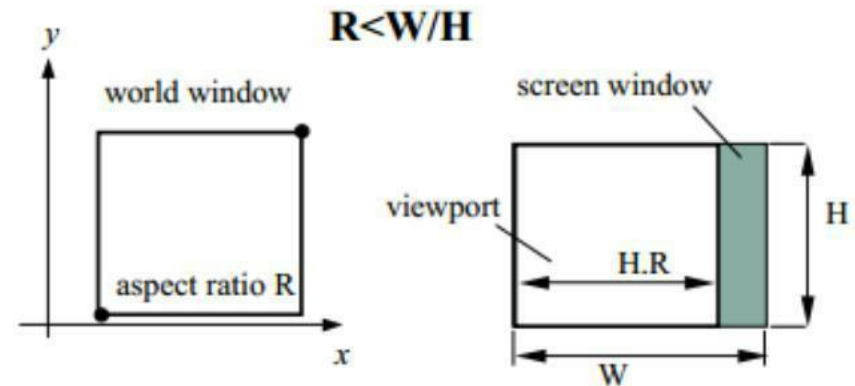
```
glViewport(0,0,W,W/R);
```

```
glViewport(0,0,360,360/2);
```



# Example: tall window

- If the world window has  $R=1.6$  and the screen has  $H=200$  and  $W=360$ , then  $W/H=1.8$ .
- Therefore, we fall in second case, and the viewport is set to 200 pixels high and 320 pixels wide.

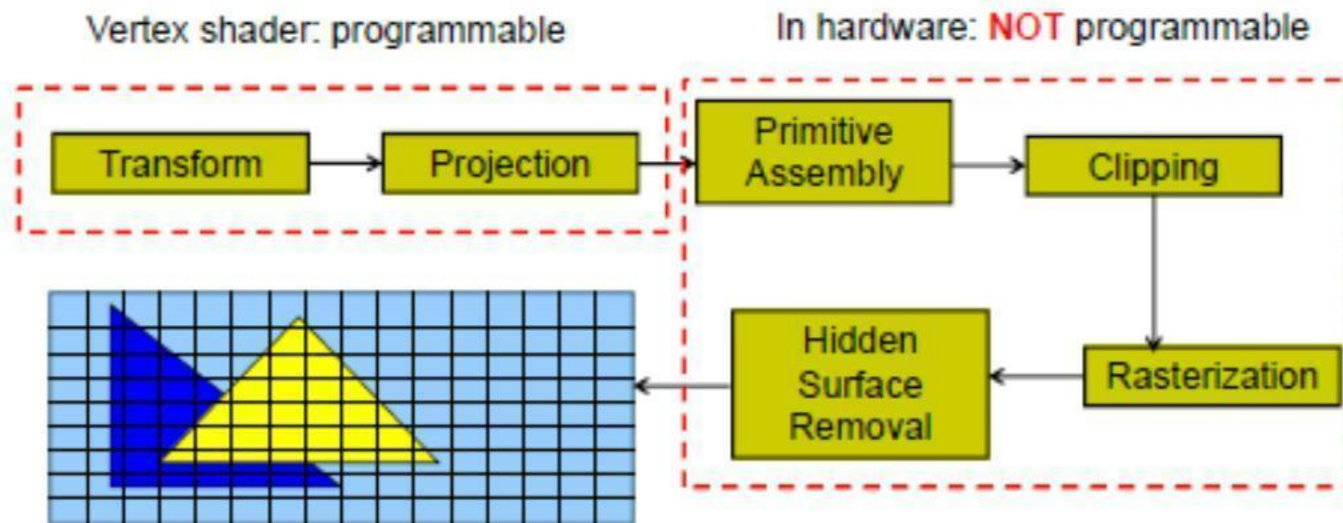


```
glViewport(0, 0, H * R, H);
```

```
glViewport(0, 0, 320, 200);
```

# OpenGL үе шат

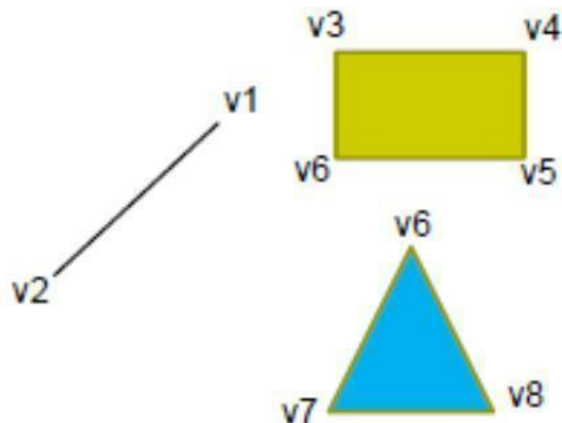
- ▶ Projection хийгдсэний дараа объектийг дэлгэцэнд зур ахын өмнө хэд хэдэн үе шатуудыг дамжина.
- ▶ Эдгээр үе шатууд нь програмчлагдахгүй.





# ТХ -ийн үе шат: Примитив угсрах

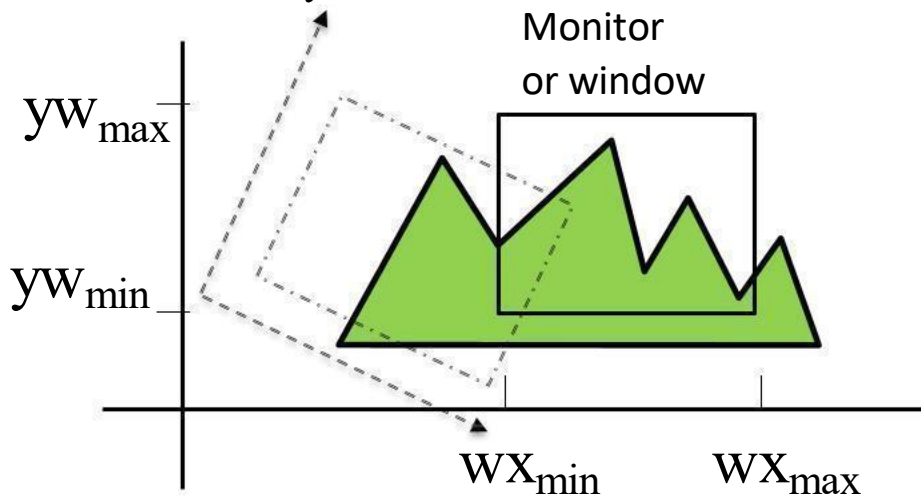
- ▶ Энэ үе хүртэл: Салангид vertex-д transformations, projections –г хэрэгжүүлнэ.
- ▶ Примитив угсрах: transformations, projections –г хэрэгжүүлсэний дараагаар салангид vertex-д примитив рүү групплэгдэнэ.
- ▶ Жнь: v6, v7 болон v8 гурвалжин руу групплэгдсэн



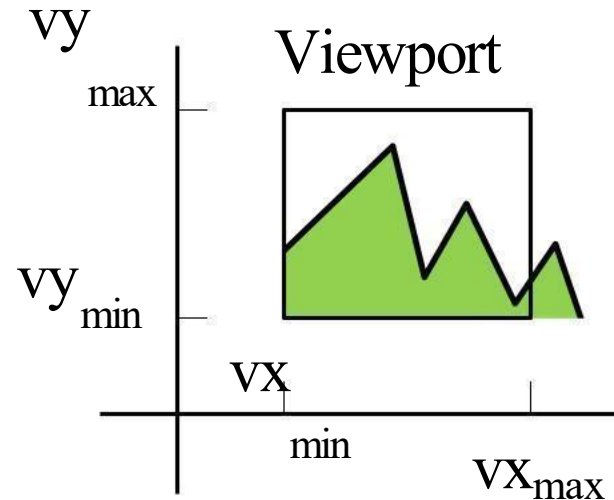
# Viewing transformation

- Мастер координатын систем, нийтлэг нэр нь world coordinate system
  - ✓ Clipping window: Юуг харуулахыг хүсч байна вэ?
  - ✓ Viewport: Бид үүнийг хаанаас харуулахыг хүсч байна вэ?

Viewing Coordinate System



a) World Coordinate System

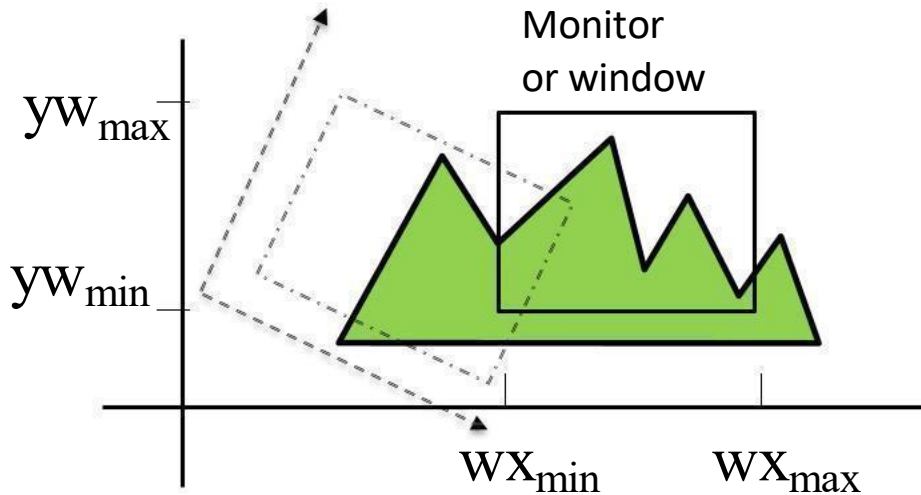


b) Device Coordinate System

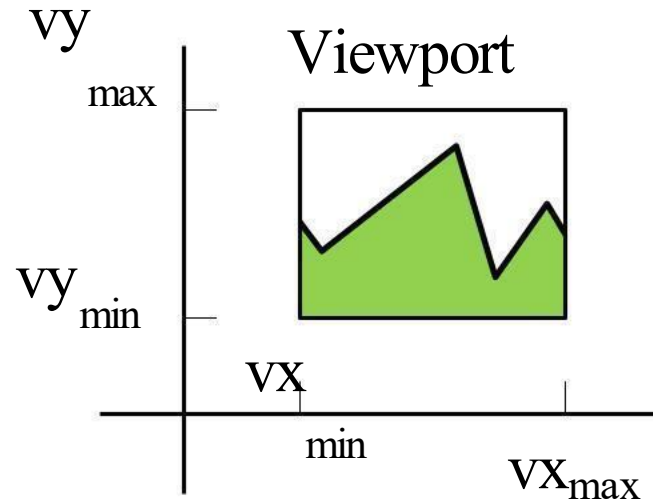
# Viewing transformation

- **Viewing Transformation:** world болон viewing координатын системүүдийн хооронд координатыг буулгах (mapping) үйлдэл юм.  
✓ clipping window -ийг viewport руу буулгана

Viewing Coordinate System



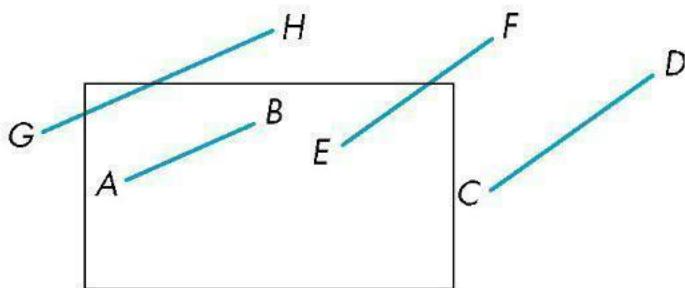
a) World Coordinate System



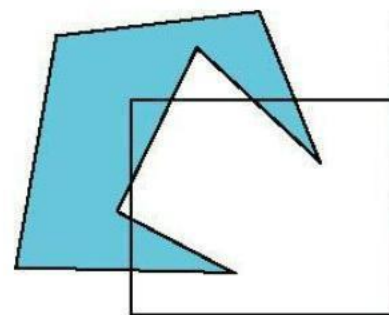
b) Device Coordinate System

# ТХ -ийн үе шат: Clipping

- ▶ Прimitives-ийг угсарсны дараа primitive бүрийн хувьд хийгдэх үйлдэл
- ▶ **Clipping:** frustum-аас гадна орших primitive-ийг устгах (**lines, polygons, text**)
- ▶ lines, polygons-ний хувьд clipping хийхэд хялбар.
- ▶ text-ний хувьд clipping хийхэд хялбар биш.



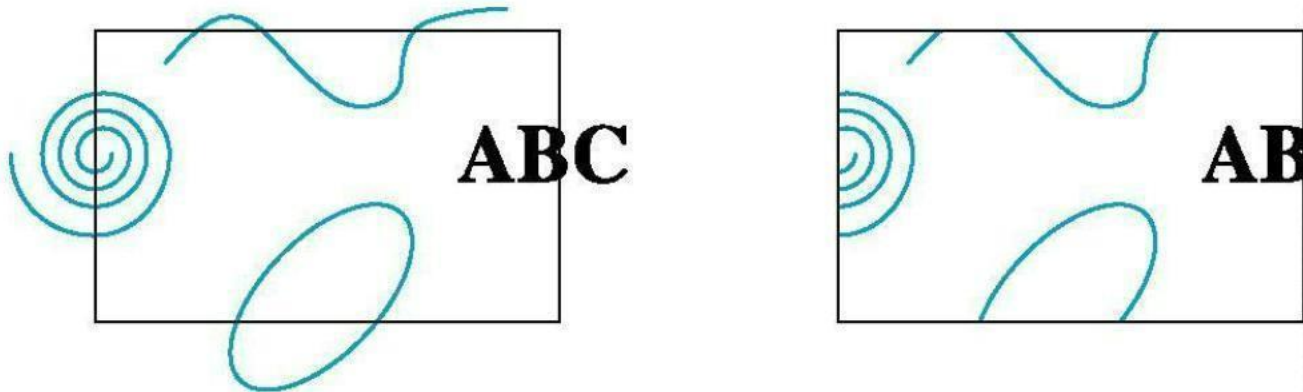
Clipping lines



Clipping polygons

## ► Хоёр хэмжээст (2D) clipping

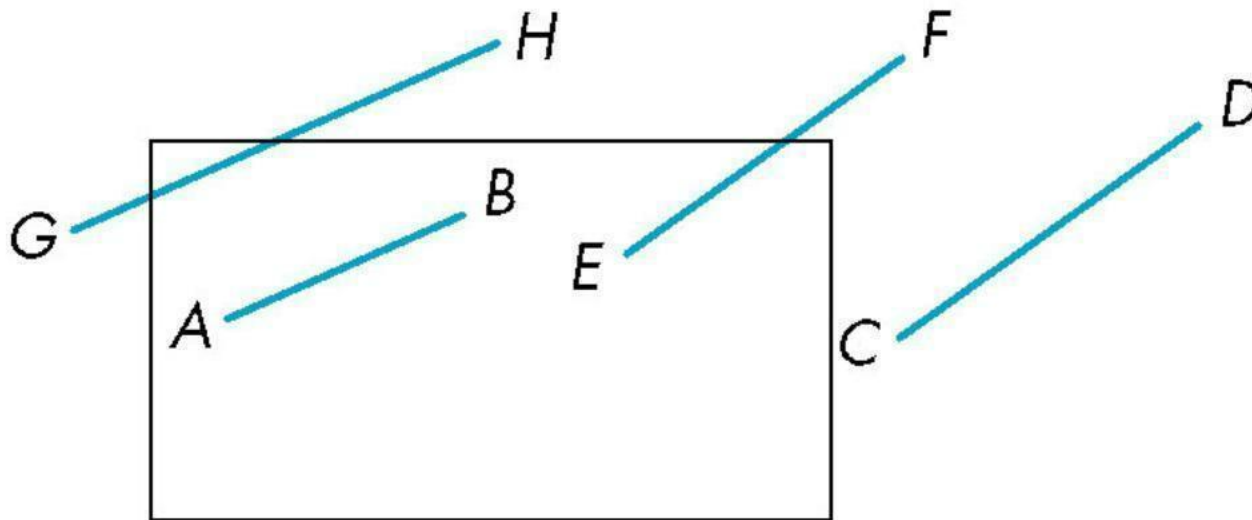
- Lines
- Polygons
- Curves
- Text



## Brute force approach:

---

- ▶ Цонхны тал бүрээс илүү ирмэгийг хасах
- ▶ Огтлолцол бүрт нэг зүсэлт ( үр дүн муу арга)



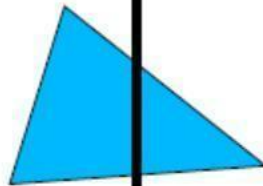
# 2D clipping

---

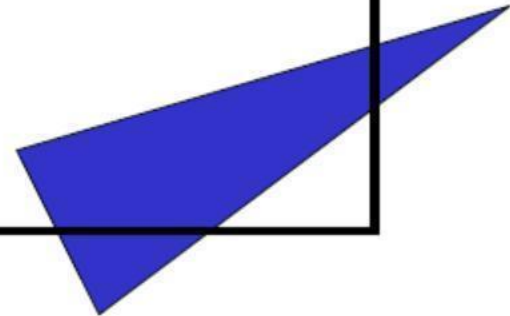
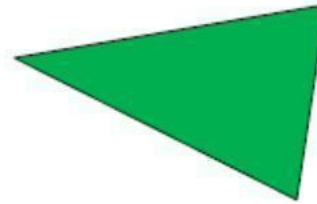
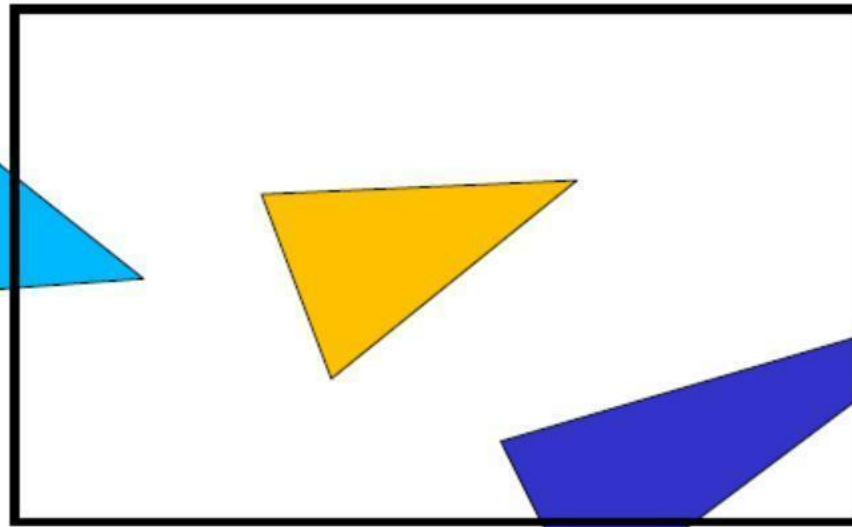
Types of operations

- Accept
- Reject
- Clip

Polygon



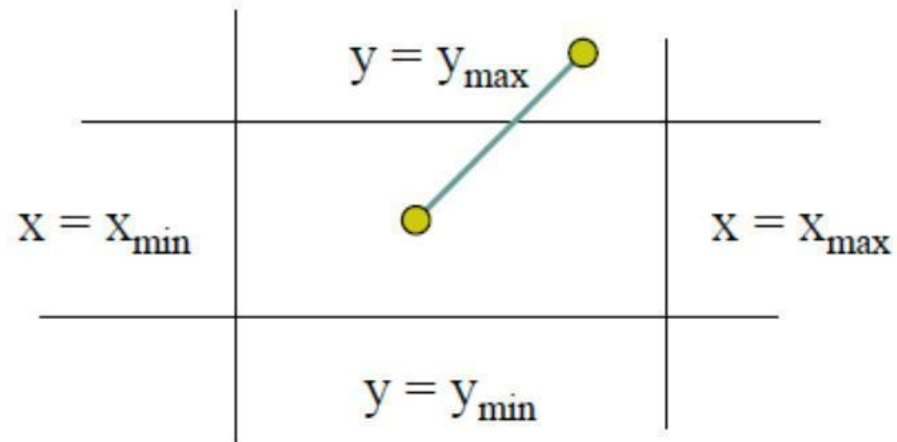
Viewport





# 2D clipping

- ▶ Cohen-Sutherland clipping algorithm



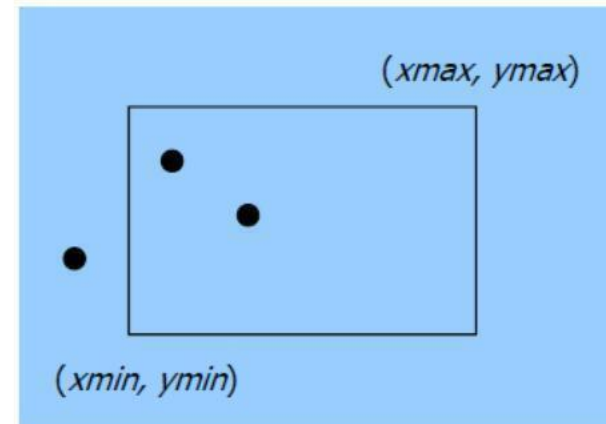
# Clipping points

---

- ▶ Цэгүүдийн байрлалыг тогтооно.
- ▶ Цонхны гадна, дотор байгааг олно.

If  $(x_{min} \leq x \leq x_{max})$   
and  $(y_{min} \leq y \leq y_{max})$

- ▶  $(x, y)$  – дотор талд
- ▶ Эсрэг тохиолдолд – гадна талд



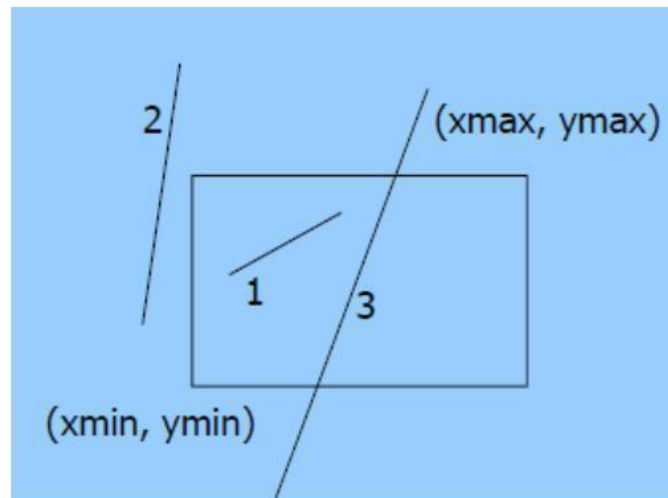
# Clipping Lines

---

Case1:Бүх шулуун багтана - Accept

Case2:Бүх шулуун гадна байна. - Reject

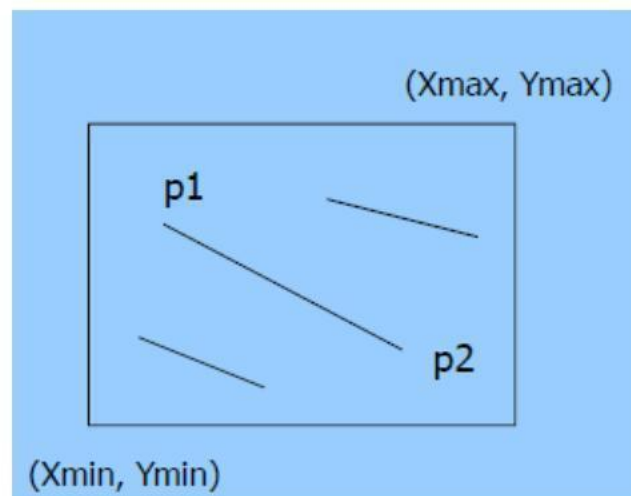
Case3:Зарим нь багтана, зарим нь гадаа - Clip



# Clipping Lines: Trivial Accept

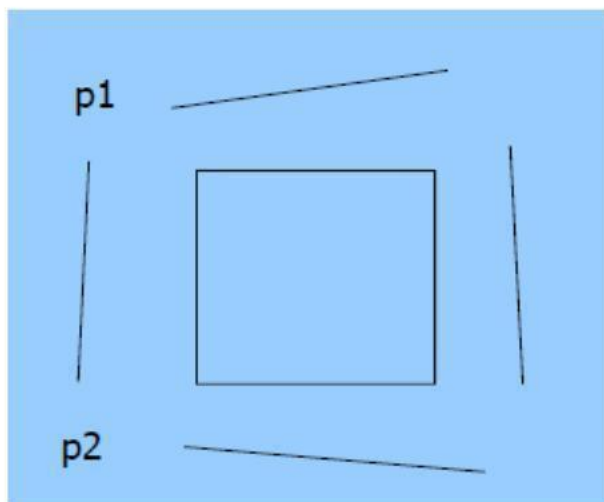
- ▶ Case I: Бүх шулуун багтана
- ▶  $x, y$  утгуудыг тэгш өнцөгтийн  $x, y$  утгуудтай харьцуулна.
- ▶ **Үр дүн:** ердийн зөвшөөрөл(ассерт). Шулуунийг бүхэлд нь зурна.

$Xmin \leq P1.x, P2.x \leq Xmax$  and  
 $Ymin \leq P1.y, P2.y \leq Ymax$



# Clipping Lines: Trivial Reject

- ▶ Case 2: Бүх шулуун гадна байна.
- ▶ Reject.
- ▶ Don't draw line in



- $p1.x, p2.x \leq Xmin$  OR
- $p1.x, p2.x \geq Xmax$  OR
- $p1.y, p2.y \leq ymin$  OR
- $p1.y, p2.y \geq ymax$

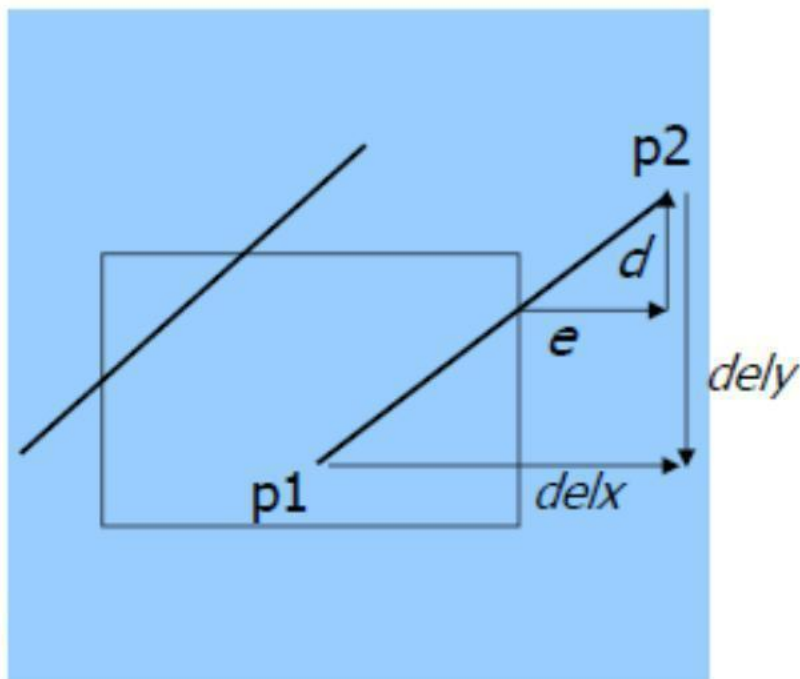
# Clipping Lines: Non-Trivial Cases

---

- ▶ Case 3: Part in, part out

Дотор агуулагдаж буй хэсгийг олох шаардлагатай.

- ▶ Гурвалжин ашиглан уртыг тооцож болдог.

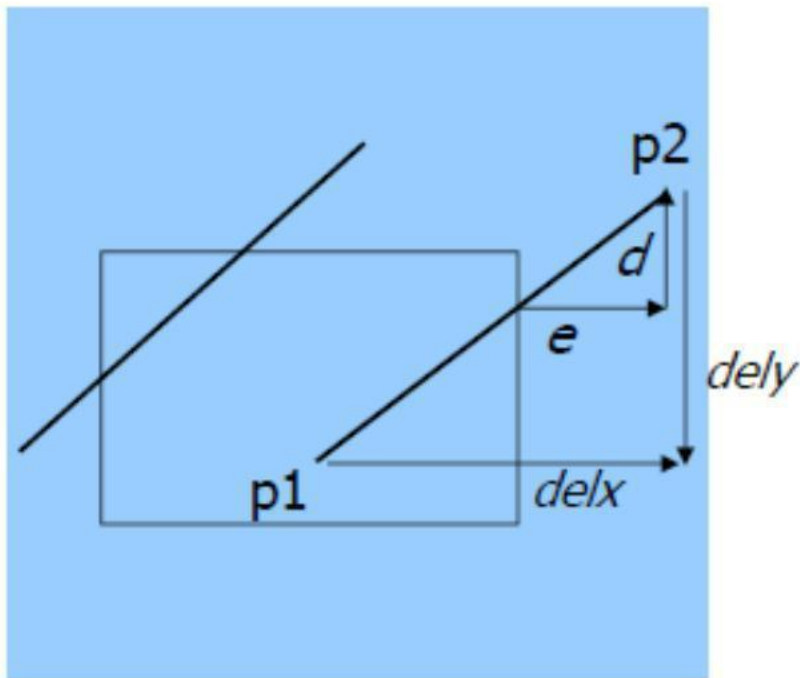


$$\frac{d}{dely} = \frac{e}{delx}$$

---

# Тооцоолол хийх

- ▶ (left, right, bottom, top) = (30, 220, 50, 240)



(a)  $p1 = (40, 140), p2 = (100, 200)$

(b)  $p1 = (20, 10), p2 = (20, 200)$

(c)  $p1 = (100, 180), p2 = (200, 250)$

▶  $\frac{d}{dely} = \frac{e}{delx}$



# Cohen-Sutherland pseudocode

---

```
int clipSegment(Point2& p1, Point2& p2, RealRect W)
{
do{
if(trivial accept) return 1; // whole line survives
if(trivial reject) return 0; // no portion survives
// chop
if(p1 is outside)
// find surviving segment
{
if(p1 is to the left) chop against left edge
else if(p1 is to the right) chop against right edge
else if(p1 is below) chop against the bottom edge
else if(p1 is above) chop against the top edge
}
```



---

```
else // p2 is outside
// find surviving segment
{
if(p2 is to the left) chop against left edge
else if(p2 is to right) chop against right edge
else if(p2 is below) chop against the bottom edge
else if(p2 is above) chop against the top edge
}
}while(1);
}
```



# Үр дүнг ашиглан харьцуулатыг хурдасгах

- Endpoint бүрийг кодчилон доорх үр дүнгээс харьцуулна  
а. (9 талбар, 4 бит код)

$b_0b_1b_2b_3$

$b_0 = 1$  if  $y > y_{\max}$ , 0 otherwise

$b_1 = 1$  if  $y < y_{\min}$ , 0 otherwise

$b_2 = 1$  if  $x > x_{\max}$ , 0 otherwise

$b_3 = 1$  if  $x < x_{\min}$ , 0 otherwise

1001	1000	1010	$y = y_{\max}$
0001	0000	0010	
0101	0100	0110	$y = y_{\min}$
$x = x_{\min}$		$x = x_{\max}$	