Computer Graphics

Prof. Jibum Kim

Department of Computer Science & Engineering

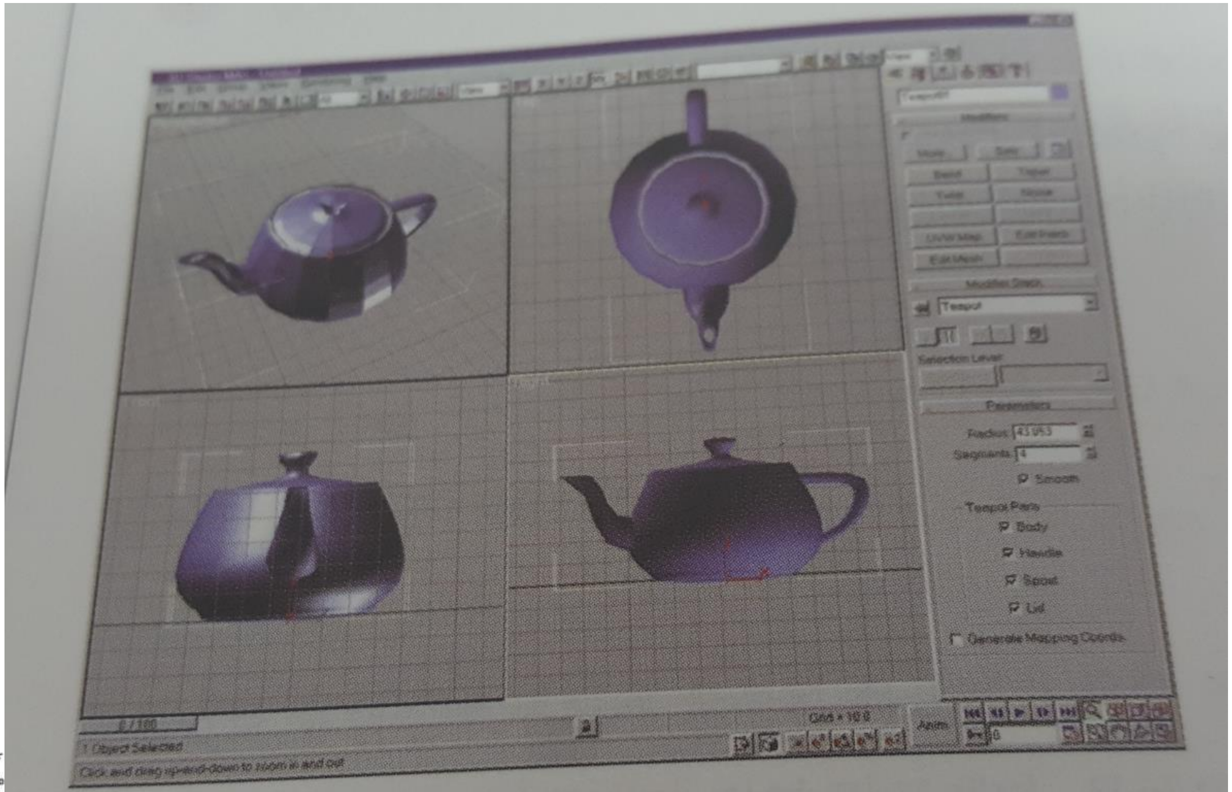
Incheon National University

- **Viewport (뷰포트)**

- 지금까지는 가시공간에서 그린 물체를 **screen window (window)** 전체를 사용하여 나타내었다

- 하지만, 어떤 경우에는 가시 공간에 그린 물체를 **screen window**의 일부만 사용하여 표현하고 싶은 경우도 있을 수 있다

- **Viewport example**
- **3D Studio Max: divide the left window into 4 viewports**



[그림 5-24] 뷰 포트

University of Incheon
인천대학교

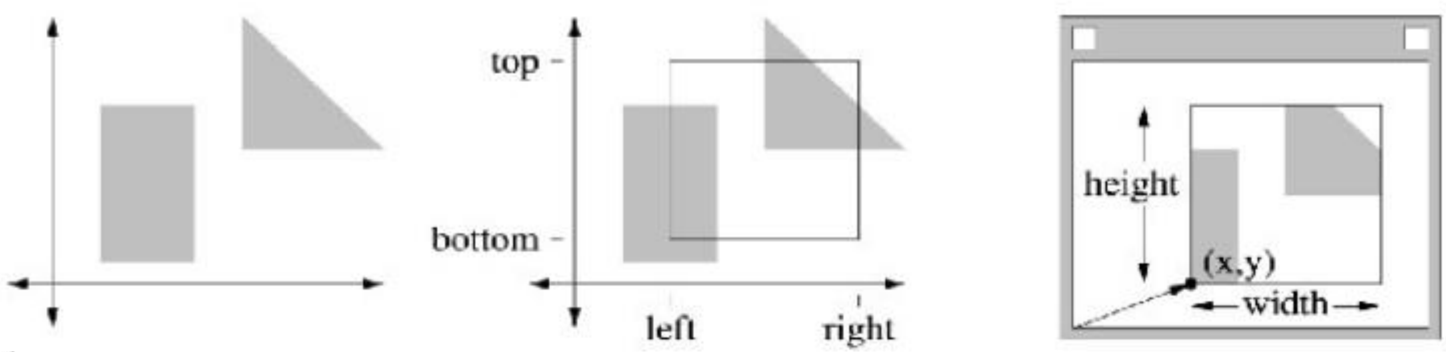
- **The viewport of a scene is that region of the window in which it is drawn**
- **OpenGL**에서는 **glViewPort()**
- 를 사용하여 **viewport**를 설정할 수 있다
- 지금까지와 같이 특별히 **viewport**를 설정하지 않으면 **default viewport**는 전체 **screen window**이다

인천대학교

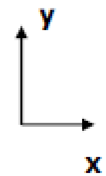
가시 공간과 viewport의 개념

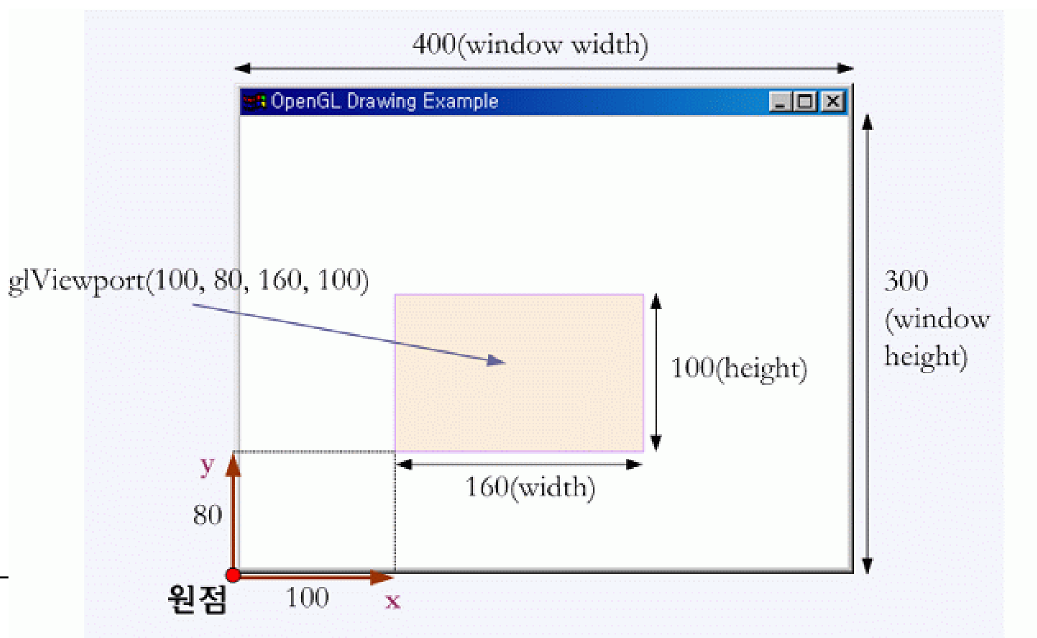
1. 가시 공간 밖에 있는 물체는 잘려서 보이게 된다 **(clipping)**

2. 가시 공간에서 그려진 물체는 **screen window** 전체가 아닌 **viewport**에만 그려지고 **viewport**에 **mapping** 된다



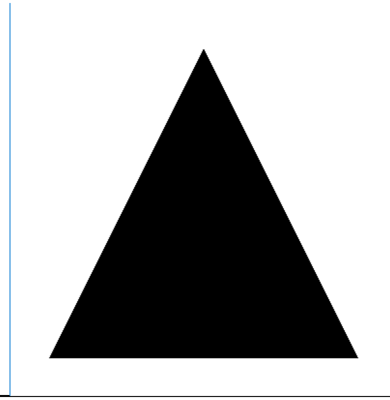
- OpenGL에서의 viewport 설정 예
- window의 왼쪽 하단이 원점 (0,0)
- glutInitWindowSize(400,300);   // window 크기
- glViewport(100, 80, 160, 100);   // 단위 pixel





- 기본적으로 **OpenGL**에서 별도로 **viewport**를 설정하지 않으면 전체 **screen window**를 모두 사용한다
- 만일 앞의 그림과 같이 **viewport**를 설정하여 **screen window**의 일부만 사용하여 그리게 되면 어떤 현상이 생기게 될까**?**
- **https://www.dropbox.com/s/7pe44lu2joxh4fd/viewport.txt?dl=0**
- 이예제는 **viewport**를 따로 설정하지 않고 전체 **screen window**를 사용하는 예제

- **Screen window**의 크기를 가로 **400,** 세로 **400**으로 설정하고
- **glutInitWindowSize(400,400); // window**
- **gluOrtho2D(0.0, 2.0, 0.0, 2.0); //** 가시 공간
- 아래 그림과 같은 삼각형이 보인다
- 가시 공간의 종횡비 **(aspect ratio)**와
- **Screen window**의 종횡비가 일치 한다



- 이번에는 방금 전 코드에서 주석처리 되어있는

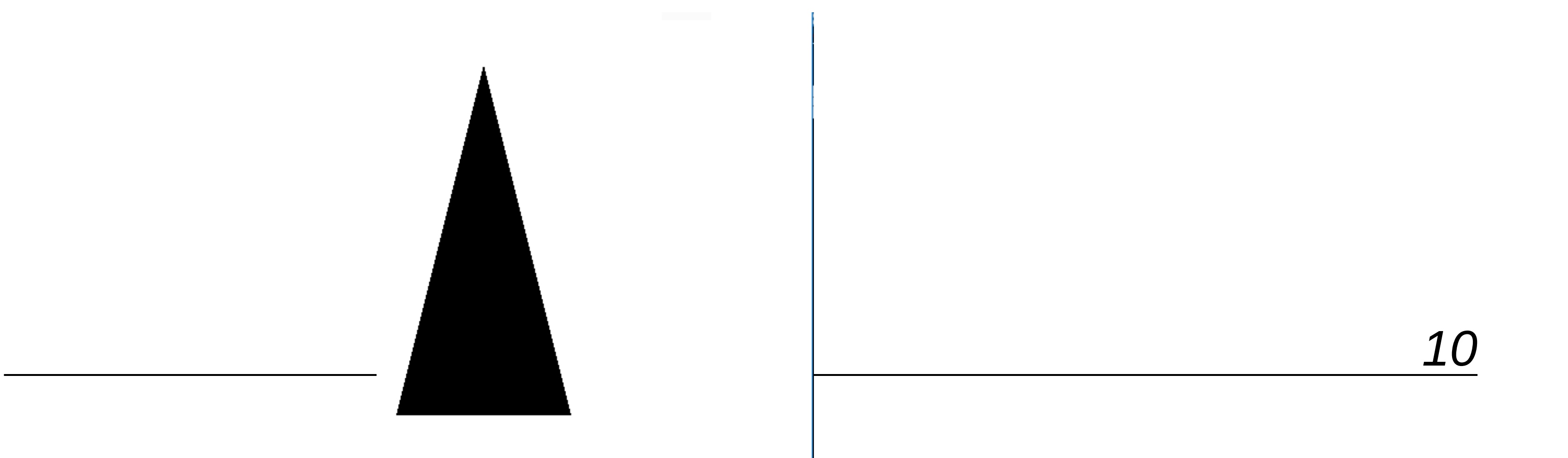
glViewport(0, 0, 200, 400);

을 주석을 제거하고 실행하자

전체 **window**가 아닌 **window**의 일부만 사용하였다.

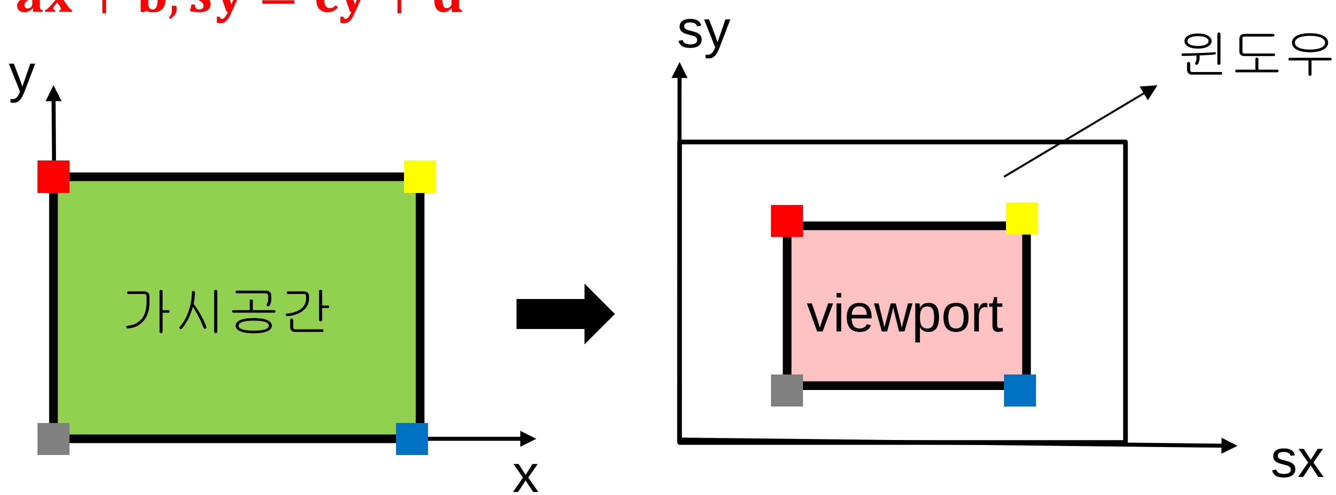
이 경우 가시공간의 종횡비와 **viewport**의 종횡비가 다르다

삼각형의 모양에 변화가 생긴 것을 알 수 있다



- 가시공간 => viewport 변환

- 가시 공간과 **viewport**의 크기 및 모양이 아래와 같다고 하면 가시 공간과 **viewport**는 모두 직사각형이므로 각각의 끝점이 각각의 끝점으로 **mapping** 된다고 하면 이렇게 **mapping** 시키는 선형 함수 **(f)**를 찾으면 된다
- $\boldsymbol{f}:(\boldsymbol{x},\boldsymbol{y}) \rightarrow (\boldsymbol{sx},\boldsymbol{sy})$
- 가로축과 세로축을 구분하여 각각의 **mapping** 함수를 찾고자 한다
- $\boldsymbol{sx = \mathrm{ax} + \mathrm{b}, sy = \mathrm{cy} + \mathrm{d}}$



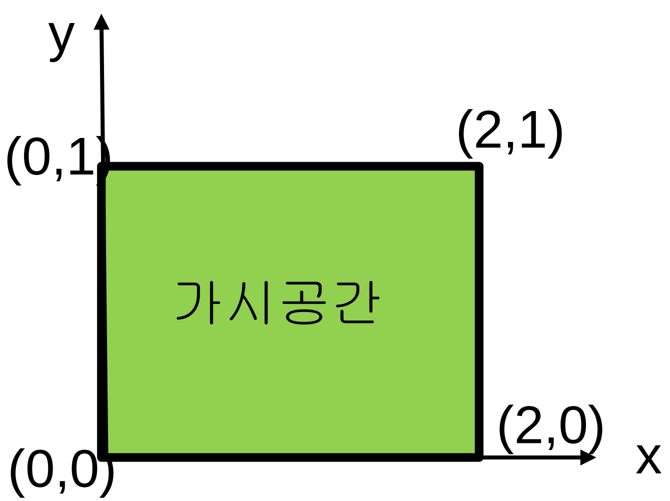
University of Incheon
인천대학교

예) 가시 공간과 window 및 viewport가 주어져 있을 때 가시 공간=>viewport로의 선형 함수를 찾아보자. **가로축 선형 mapping.** $\boldsymbol{Sx = ax + b, a}$**와** $\boldsymbol{b}$ **찾음**

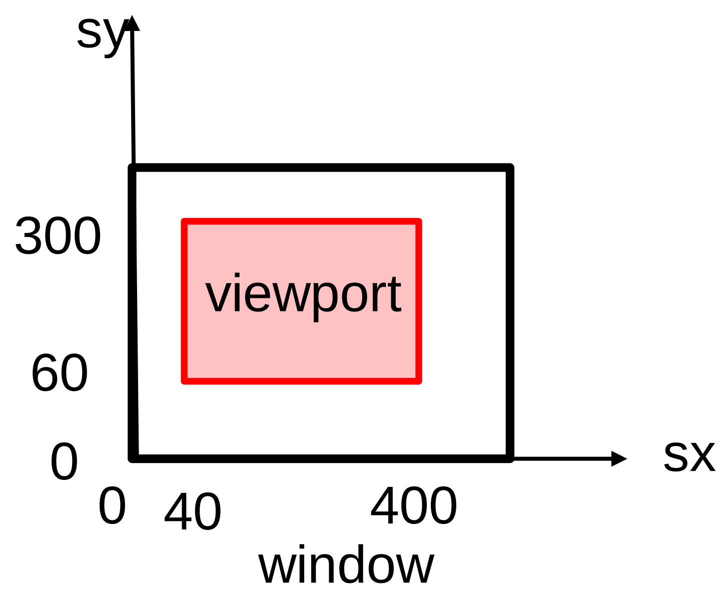
gluOrtho2D(0.0, 2.0, 0.0, 1.0);

glutInitWindowSize(500,400);

glViewport(40, 60, 360, 240);

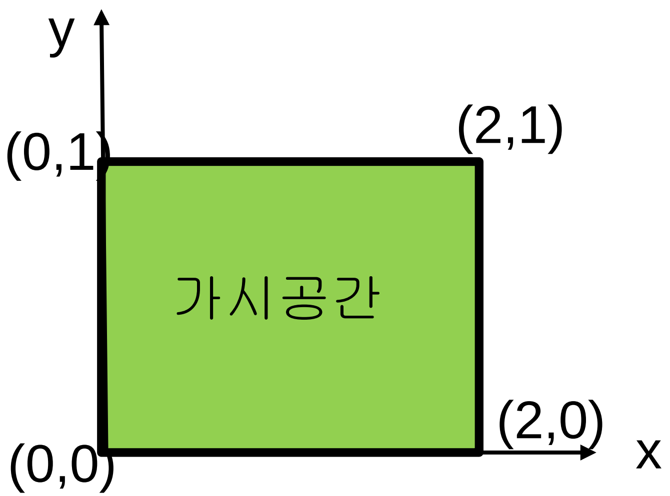


가시 공간 (세계 좌표계)

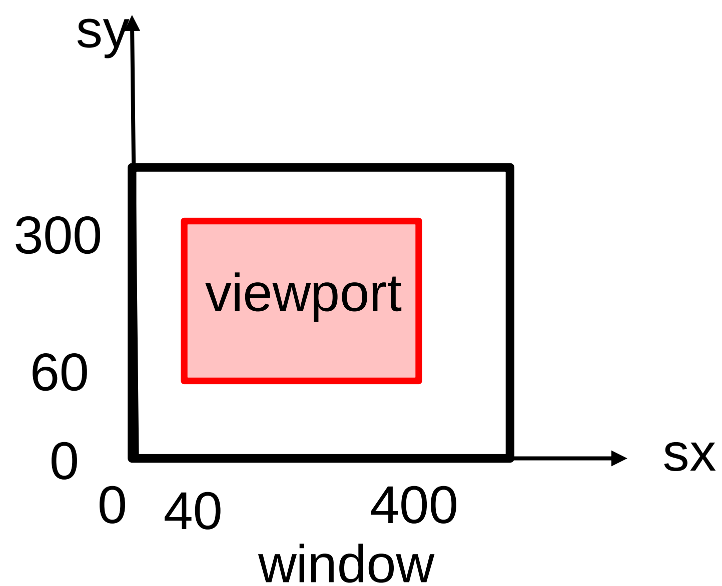


예) 가시 공간과 **window** 및 **viewport**가 주어져 있을 때 가시 공간=>**viewport**로의 선형 함수를 찾아보자. **세로축 선형 mapping.** $\boldsymbol{Sy = cy + d, c}$와 $\boldsymbol{d}$ 찾음

```
gluOrtho2D(0.0, 2.0, 0.0, 1.0);
glutInitWindowSize(500,400);
glViewport(40, 60, 360, 240);
```



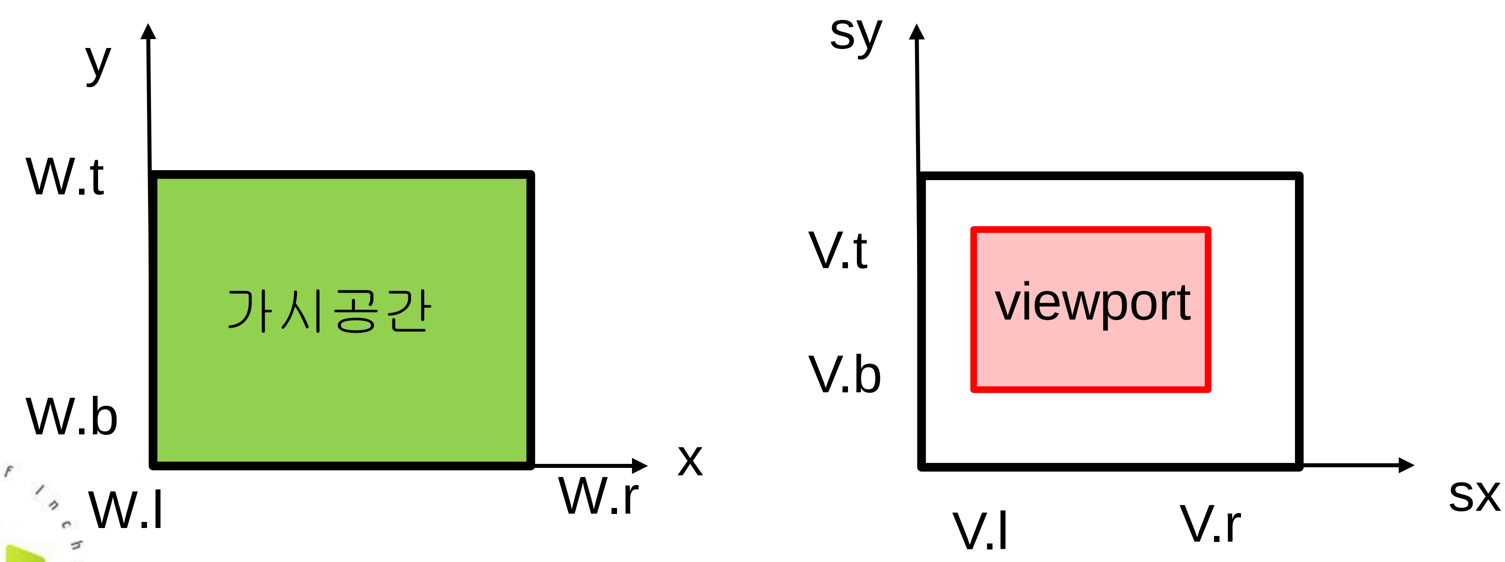
가시 공간 (세계 좌표계)



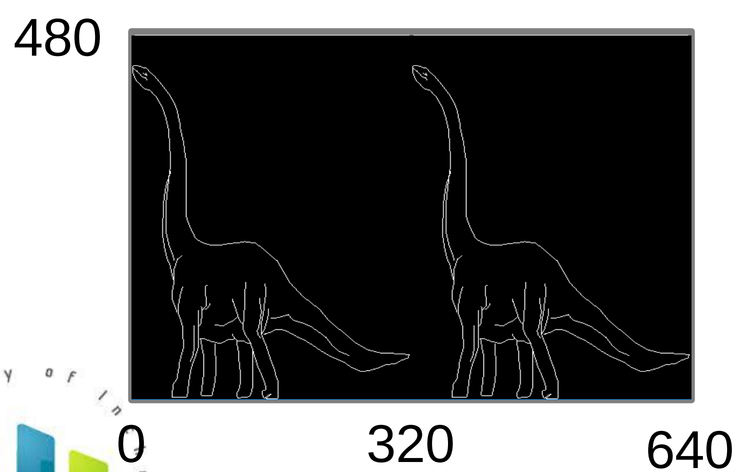
정리하면

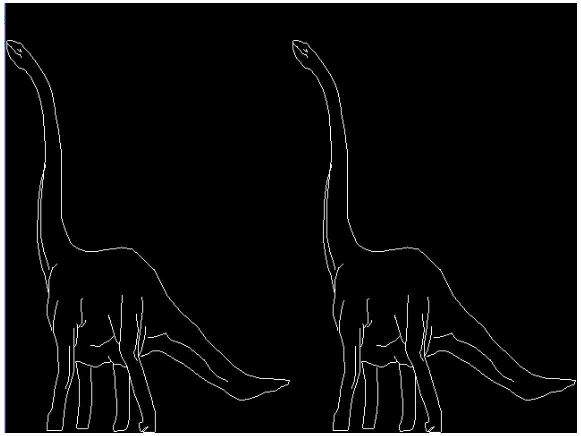
- $Sx = 180x + 40$
- $Sy = 240y + 60$

- 가시공간에서의 viewport mapping
- $sx = Ax + c, sy = By + D$
- $A = \frac{V.r - V.l}{W.r - W.l}, C = V.l - A * W.l$
- $B = \frac{V.t - V.b}{W.t - W.b}, D = V.b - B * W.b$

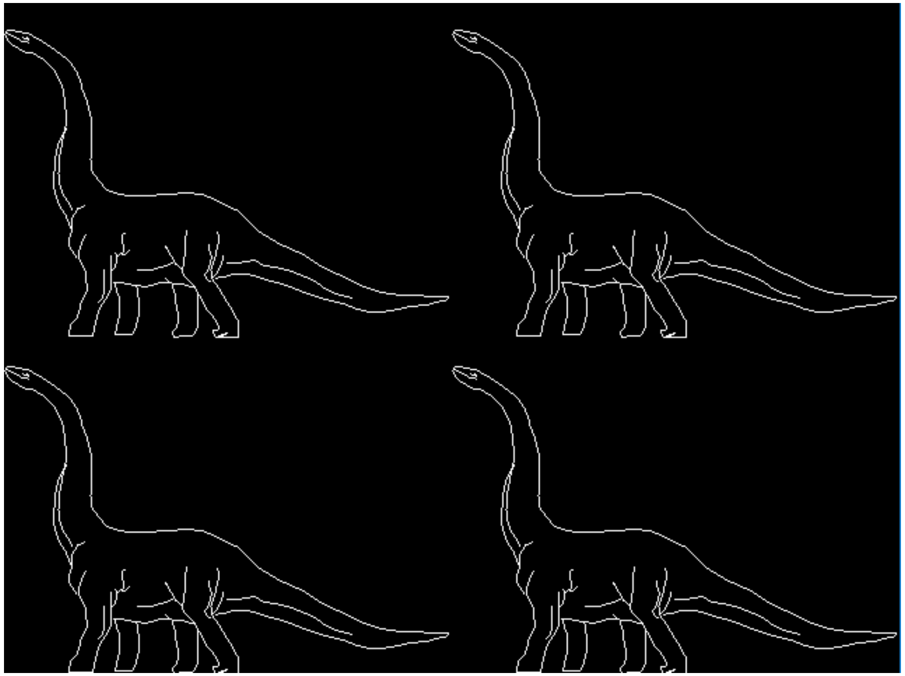


- 예) 앞에서 실습했던 **dinosaur** 예제를 **OpenGL**의 **glViewport** 함수를 이용하여 다음과 같이 만들어 보자
- **Screen window**를 가로 **640 pixel,** 세로 **480pixel**로 정하고아래와 같이 **screen window**를 동일한 크기의 **2**개로 분할한 **viewport**를 설정하였다





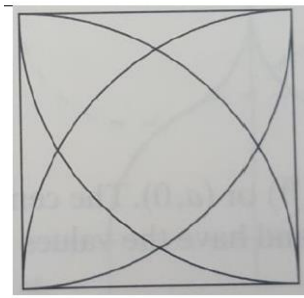
- 예: 이번에는 **viewport**를 여러 번 설정하여 다음과 같이 **dinosaur**가 **4**번 반복되어 보이게 만들어보자

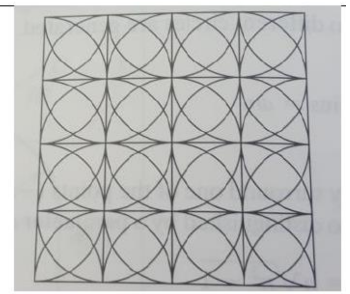


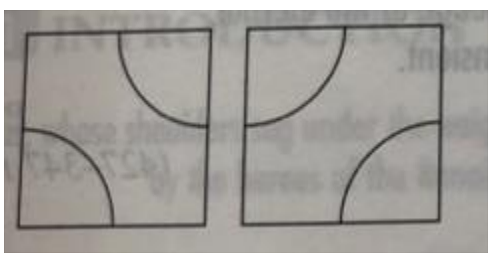
- **Screen window**에 **dinosaur**를 복사해서 여러 개의 카피본을 넣어서 만든 이와 같은 것을 **tiling**이라 한다. 아래 예
- 반복문을 사용하면 손쉽게 **tiling**이 가능하다. 어떻게 가능할까?

```
for (int i=0; i<5; i++)
 for(int j=0; j<5; j++)
   {
    glViewport(i*64, j*64, 64, 44);
   drawPolylineFile("dino.dat");
   }
```

- 멋진 모양의 여러 가지 **tiling** 예







- **Parametric form을 이용한 Curve 근사화 하기**

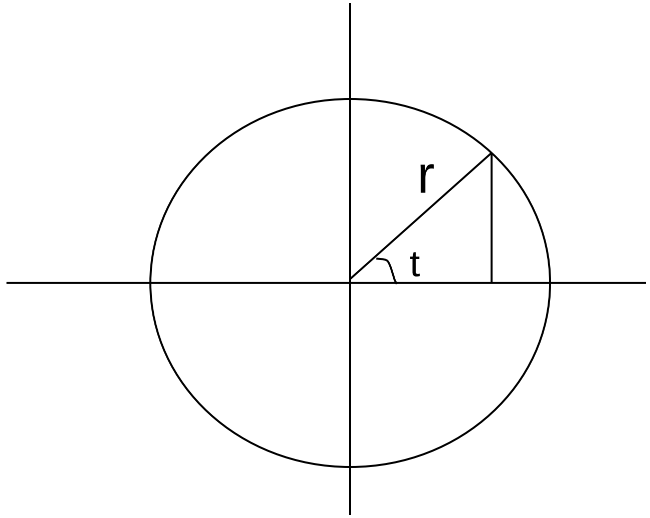
- 원과 같은 **curved line**을 묘사하는 방법은 크게 **2**가지가 있다
- **Implicit form:** 어떠한 **curve**를 함수 **F(x, y)**를 사용하여 **x**와 **y**의 관계식으로 묘사한다**. (x, y)**가 이 **curve**위에 있을 때 $F(x, y)$ =**0**이 되도록 한다

- 예**)** 원의 방정식**:** $x^2 + y^2 = R^2$**,** 중심 **(0, 0),** 반지름 **R**
- **Implicit form:** $F(x, y) = x^2 + y^2 - R^2$

인천대학교

- **Implicit하게 정의된 curve의 다른 예**
- **타원, Ellipse: $x^2/a^2 + y^2/b^2 = 1$**
- $\boldsymbol{F(x,y) = \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1}$

- **Parametric (explicit) form:** 어떠한 **curve**를 매개 변수를 사용하여 정의 한 형태
- 예**)** 중심이 **(0,0)**이고 반지름이 **r**인 원 위의 임의의 점을 각도 **t**를 이용해 아래와 같이 **parametric form**으로 표현해 보자



- $\boldsymbol{x = cos(t), y = sin(t),}$ 단$\boldsymbol{, 0 \leq t \leq 2\pi}$

- 같은 방법으로 타원을 **parametric form**으로 표현 가능하다
- 타원의 **implicit form**
- $F(x, y) = \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1$
- 타원의 **parametric form**
- $x = acos(t), y = bsin(t)$, 단, $0 \leq t \leq 2\pi$

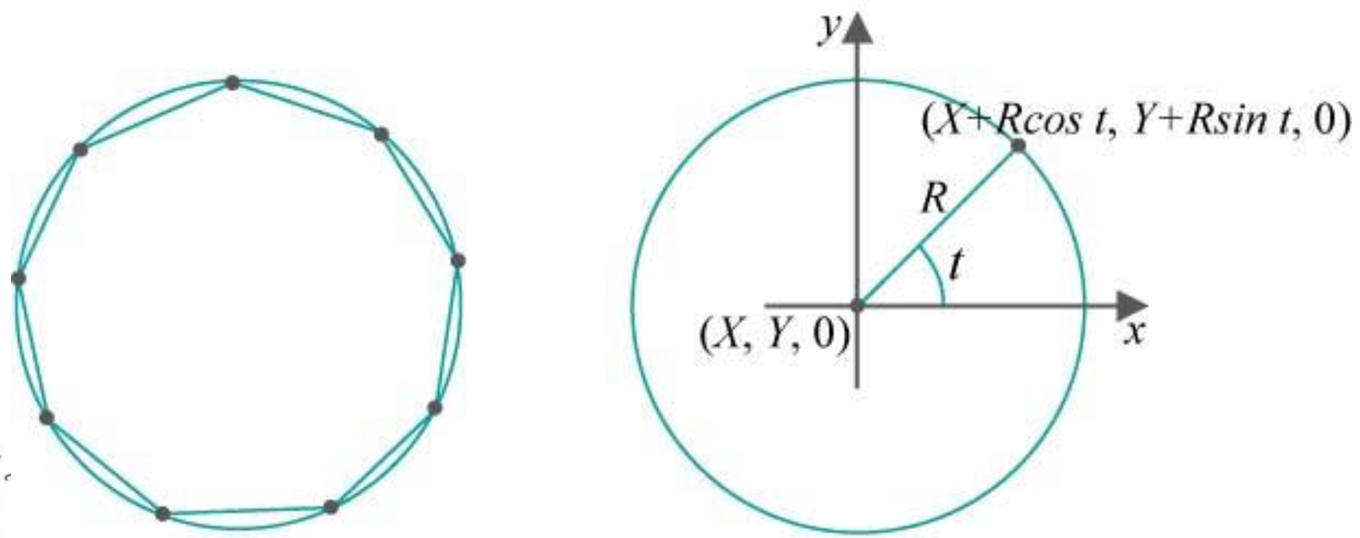
- **Q)** 어떠한 **curve**를 그릴 때 **parametric curve**를 주로 사용한다**.** 그렇다면 어떤 **curve**를 그릴 때 **parametric form**이 **implicit form**에 비해 갖는 장점은 무엇일까**?**

University of Incheon 인천대학교

- 일반적인 원인 경우의 **parametric form**

원의 중점 (X, Y), 반지름 R

$x=X+R\cos t$, $y=Y+R\sin t$, $z=0$, $0 \le t \le 2\pi$



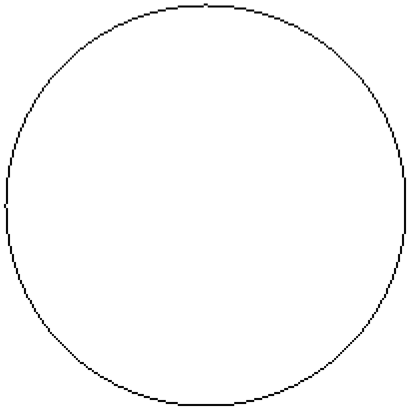
(a) (b)

- **Now, let's draw an ellipse (타원)**
- $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$
- **Implicit equation**
- **F(x,y)=**
- **Parametric equation**
- **x=a*cos(t), y=b*sin(t)**

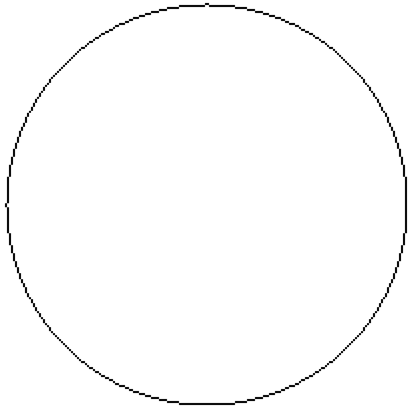
University of Incheon
인천대학교

- $x^2 + y^2 = 1$ 인 원을 앞의 방법으로 그려보자
- $x = cos(t), y = sin(t), 0 \le t \le 2\pi,$ N=100

- #define TWOPI 2*3.141592
- double t=0; // 각도
- int N=100;   // 100개의 sample 사용
- glBegin(GL_LINE_STRIP);
- for(t=0; t<= 2*PI; t+=PI/N)
- {
- glVertex2f(cos(t), sin(t));
- }
- glEnd();



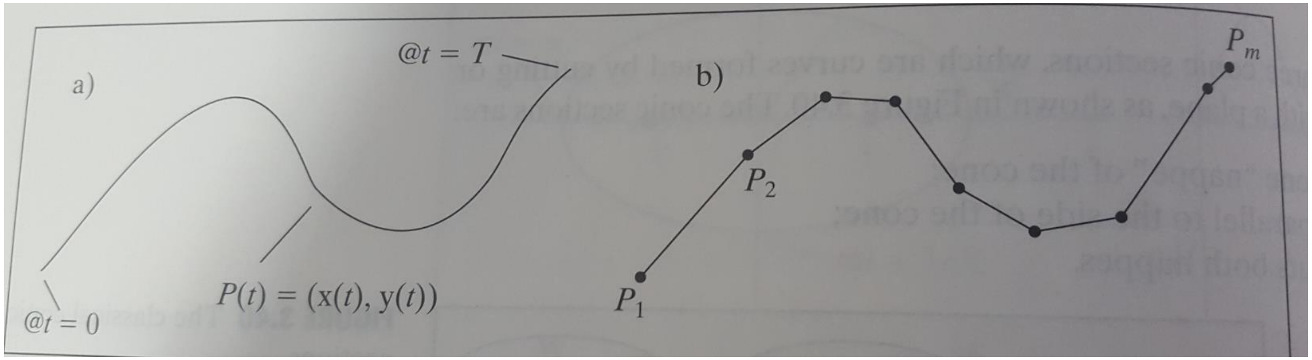
- 예: **LINE_STRIP**을 사용하여 원을 근사화한 예
- **https://www.dropbox.com/s/snvp0ldza01m4dg/circle.txt?dl=0**



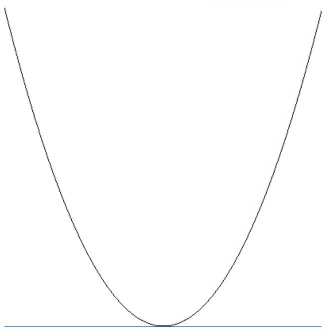
- **Drawing parabola (포물선)**

- **Parabola의 equation: $y = ax^2$**
- **Parabola의 implicit equation: F(x,y)=**
- **Parabola의 parametric equation**
- $\boldsymbol{x = t, y = at^2, t} \in (-\infty, \infty)$

- **Parametric equation**이 주어진 **Curve**를 그리는 법
- 아래와 같이 곡선의 **sample point (샘플 점)**을 뽑은 후 그 **sample point**끼리 연결한 형태의 **polyline**으로 근사화가 가능하다



- 예)y=x²을 x를 -10에서 10까지 사이에서 **N개의 sample point**를 이용하여 polyline으로 그린다고 할 경우
- $x = t, y = t^2, t \in [-10, 10]$
- 가시 공간: gluOrtho2D(-10.0, 10.0, 0.0, 100.0);



- double t;
- int N=100;   //100개의 sample point 사용
- glBegin(GL_LINE_STRIP);
- for(t=-10; t<= 10; t+=20.0/N)
- {
- glVertex2f(t, t*t);
- }
- glEnd();

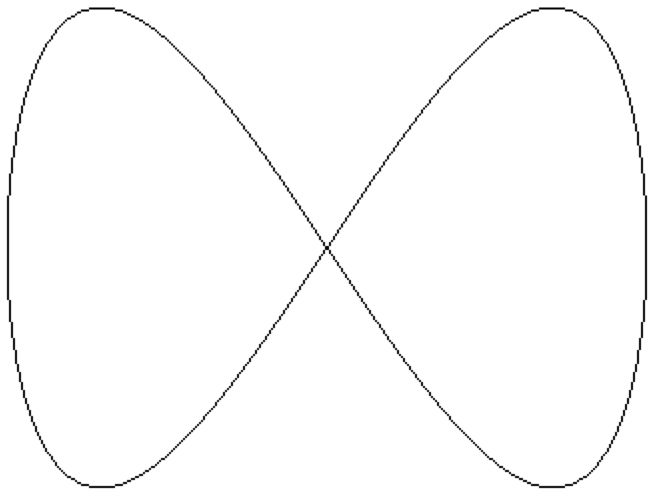
인천대학교

- **Lemniscate of Gerono**
- **https://en.wikipedia.org/wiki/Lemniscate_of_Gerono**

- 다음과 같이 표현된 **parametric equation (lemniscate of Gerono)** 을 적절한 가시공간을 주로 **OpenGL**로 그래프를 그려보자

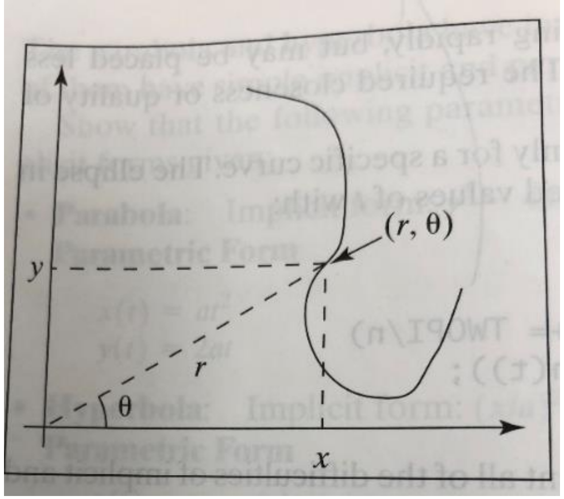
$\boldsymbol{x = cost, y = cost \ * sint}$**, 단,** $\boldsymbol{t \in [-\pi, \pi]}$

인천대학교

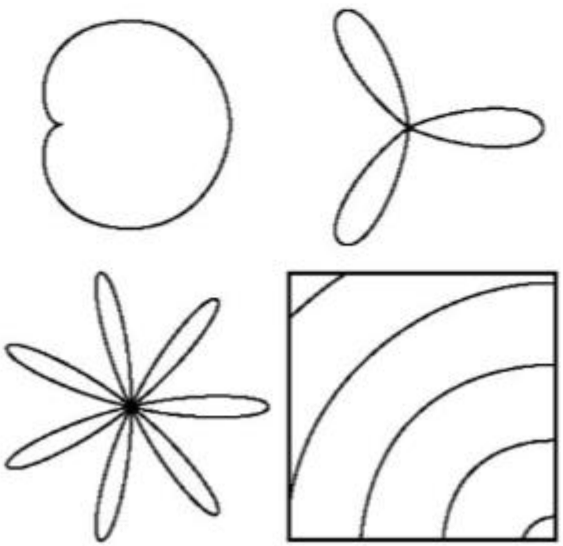


- **Polar coordinate system (극좌표계)**

- 다양한 curve 모양을 표현하는데 **polar coordinates (극좌표)**가 사용될 수 있다
- **Each point on the curve is represented by an angle θ and a radial distance r**
- 이를 파라미터 **t**와 직교 좌표계 **(Cartesian coordinates)**로 표현한다면
- $x(t) = r(t)cos(\theta(t))$
- $y(t) = r(t)sin(\theta(t))$

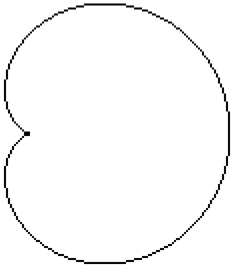


- 극좌표를 이용 하나의 파라미터로 표현된 다양한 커브
- $x = f(\theta)cos(\theta), y = f(\theta)sin(\theta)$
- 여기서 K는 **curve**의 전반적인 크기 결정
- **Cardioid:** $f(\theta) = K(1 + cos(\theta))$
- **Rose curve:** $f(\theta) = Kcos(n\theta)$**,** n은 **rose**의
- **Archimedean spiral:** $f(\theta) = \theta$



gluOrtho2D(-2.0, 2.0, -2.0, 2.0);

```
glBegin(GL_LINE_STRIP);
double K = 0.5;
for (t = 0; t <= 2*PI; t += 2 * PI / N)
{
    glVertex2f(K*(1+cos(t))*cos(t), K*(1+cos(t))*sin(t));
}
glEnd();
```



- **Archimedean spiral**