02

OPEN ORIENTED

凹 凸 实 验 室

canvas 图像旋转与翻转

张燕婷

2017.05.08

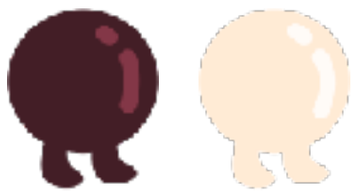
需求背景

02

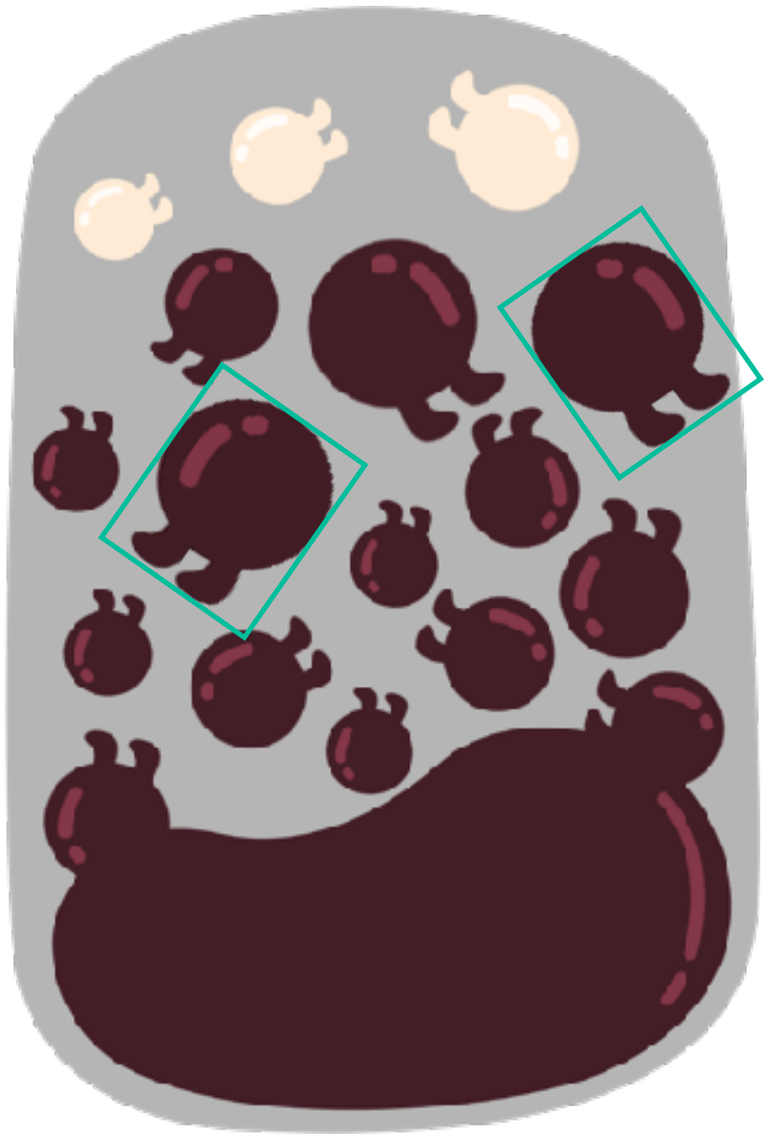
需求背景

02

VS



需求背景

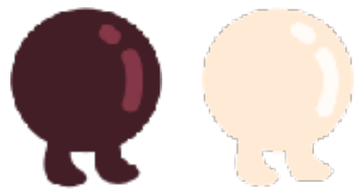


A. 多个动画元素：canvas

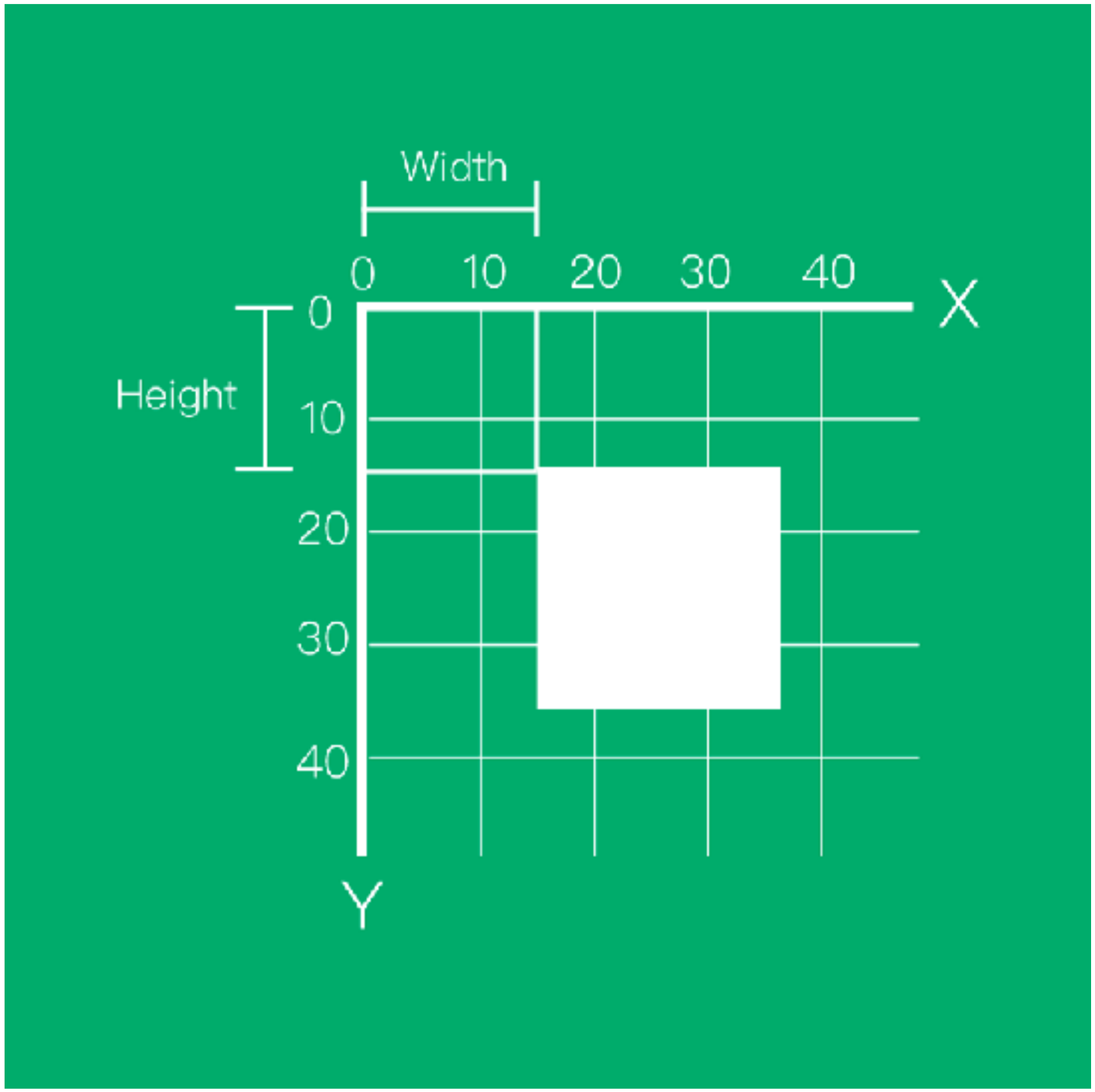
B. 相同的不规则图像：canvas 图像

C. 多个角度/方向的图像：canvas 图像旋转与翻转

VS



canvas 坐标系



A. 原点在左上角

B. X坐标向右方增长

C. Y坐标像下方延伸

C. 向左下方延伸

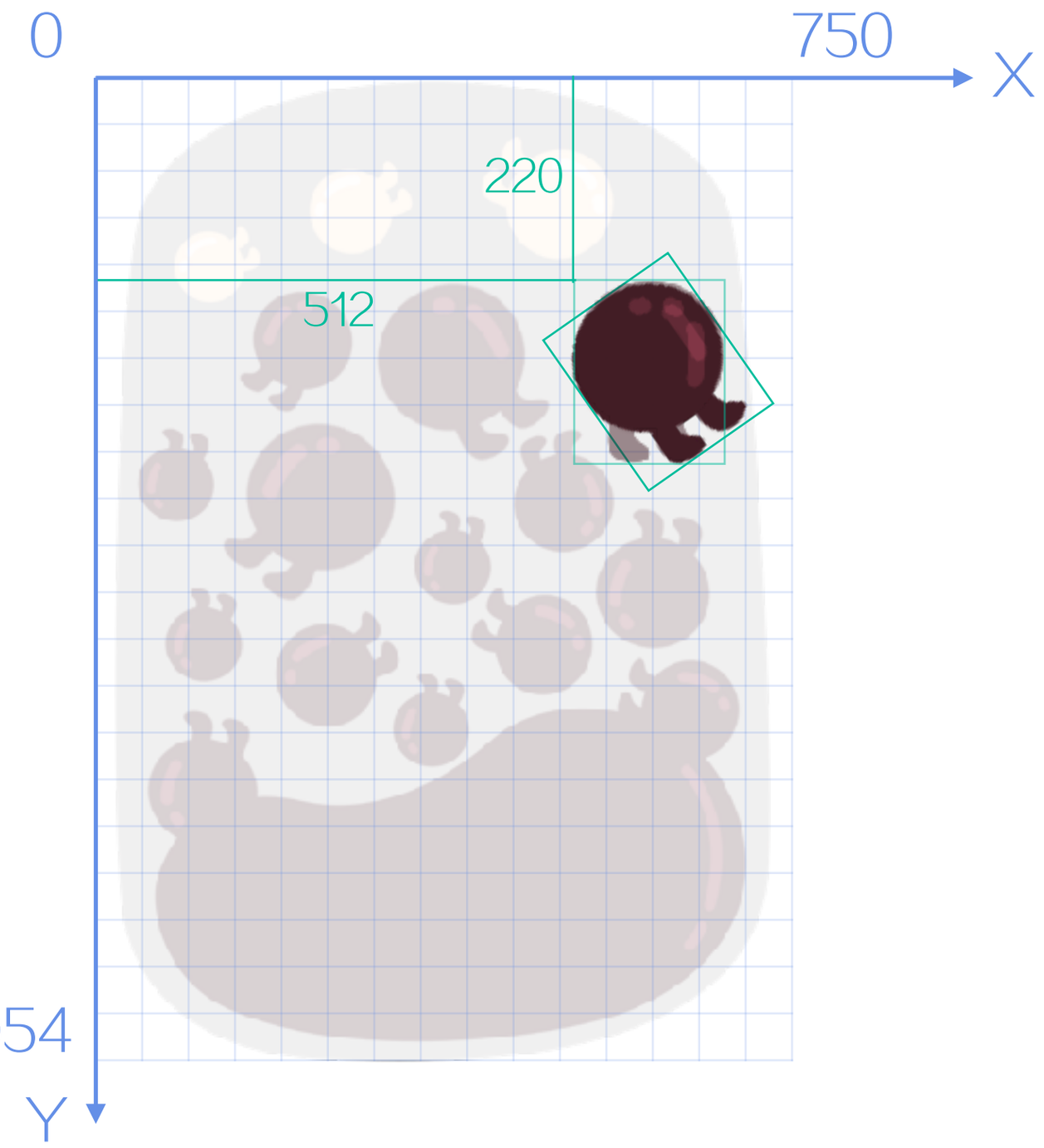
canvas 坐标系

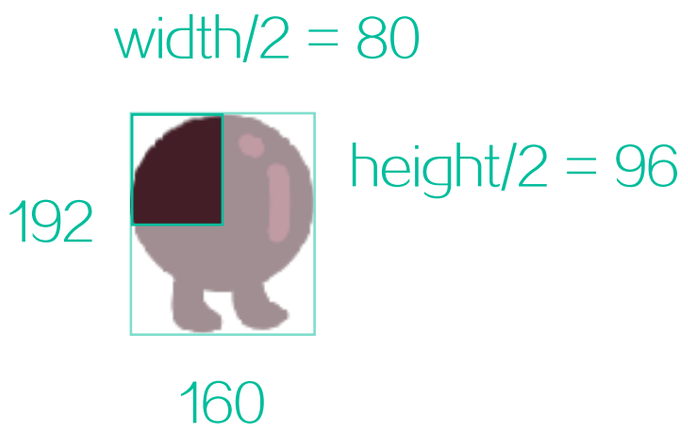

```
var canvas = document.
      getElementById( ‘myCanvas’ )
canvas.width = 750
canvas.height = 1054
var ctx = canvas.getContext( ‘2d’ )
```

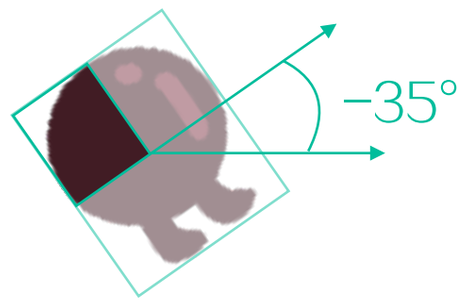
坐标变换

图像旋转

```
var img = new Image()
img.src = 'xxxxxxx.svg'
img.onload = function(){…}
```







坐标变换

02

```
ctx.translate(x, y)
```

translate() 方法接受两个参数。x是左右偏移量，y是上下偏移量。

```
ctx.rotate(angle)
```

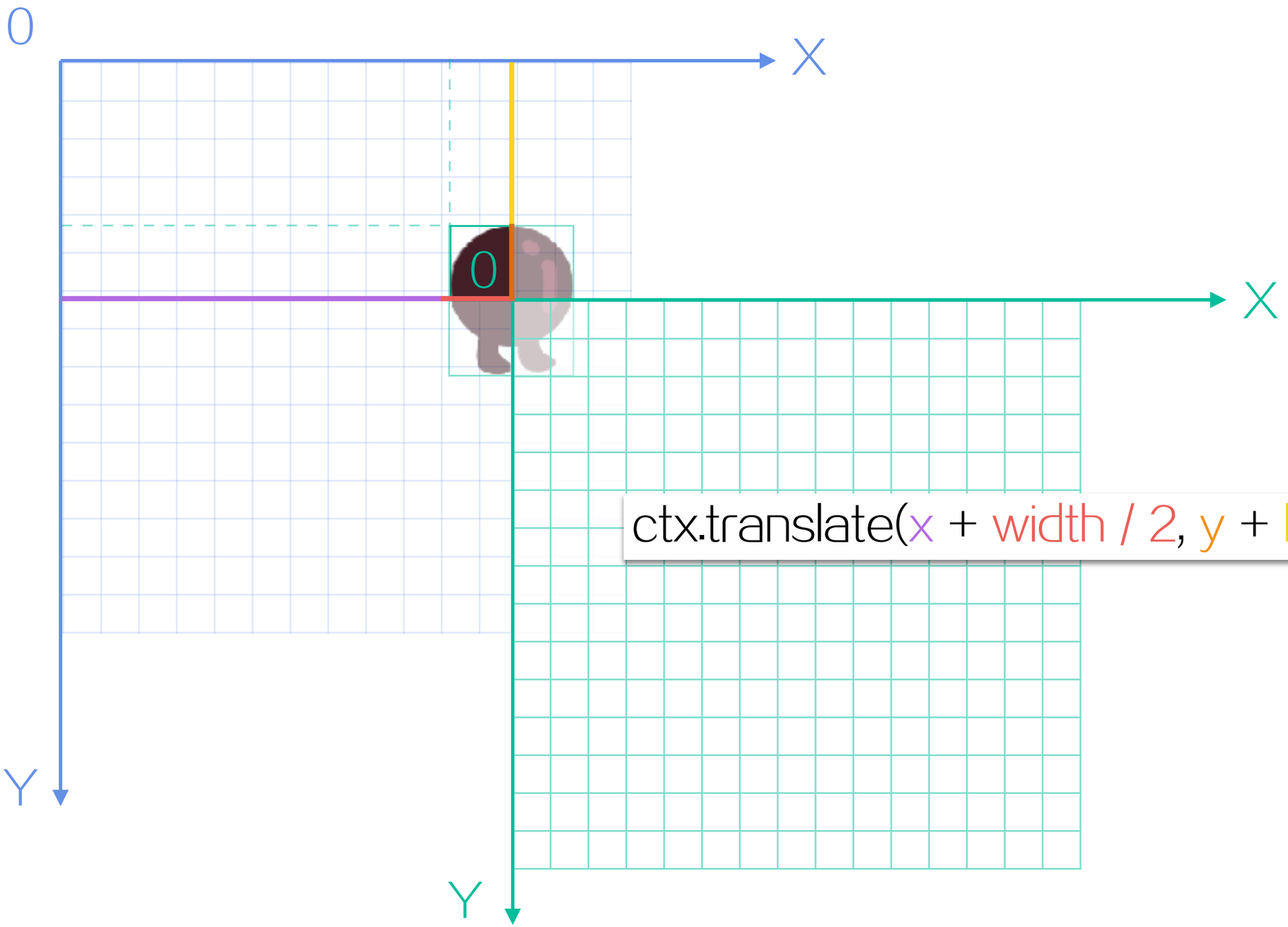
rotate() 方法只接受一个参数。旋转的角度angle，它是顺时针方向的，以弧度为单位的值。

```
ctx.scale(x, y)
```

scale() 方法接受两个参数。x和y分别是横轴和纵轴的缩放因子。其缩放因子默认是1，如果比1小是缩小，如果比1大则放大。

坐标变换

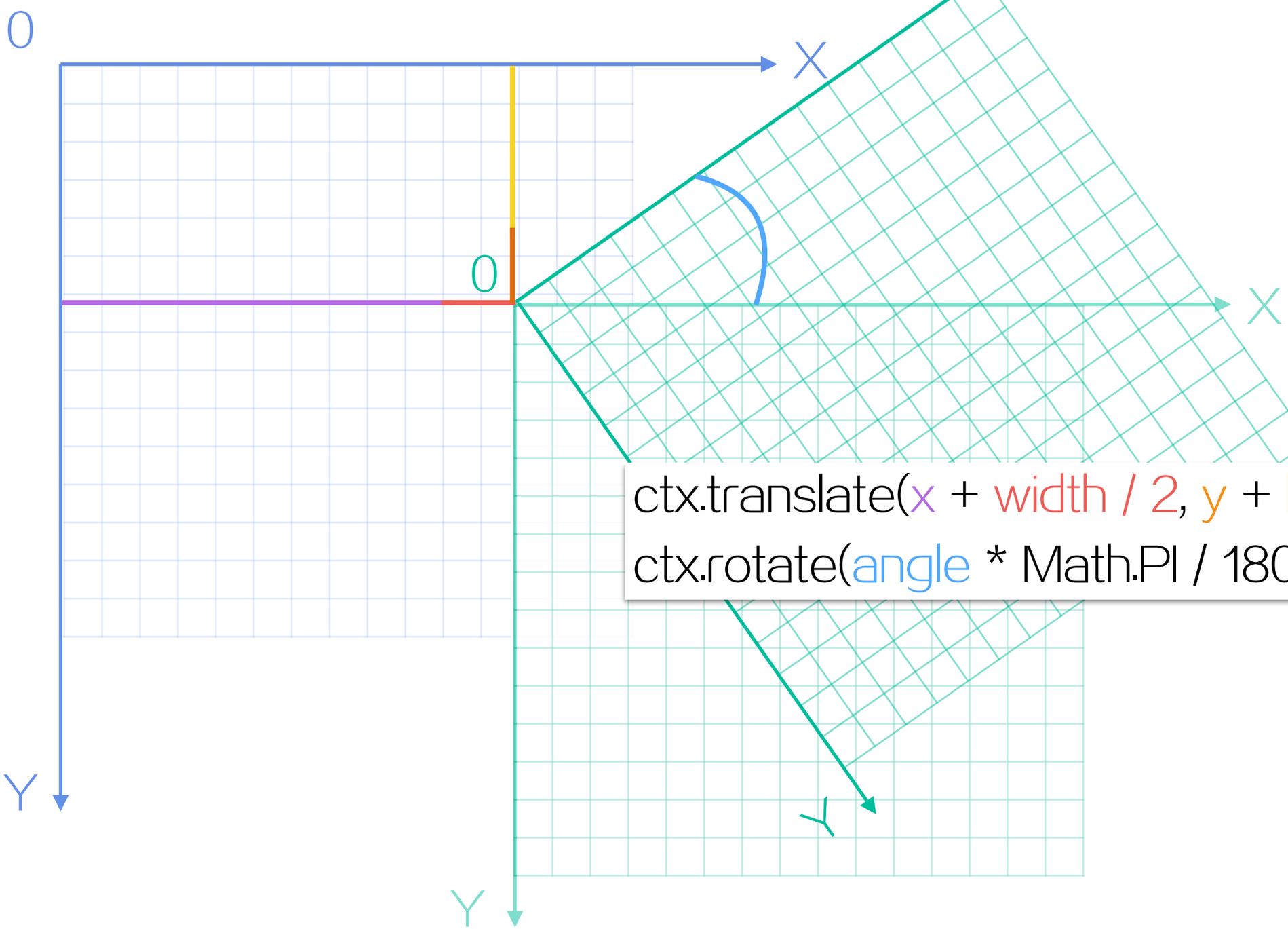
图像旋转




```
ctx.translate(x + width / 2, y + height / 2)
```

坐标变换

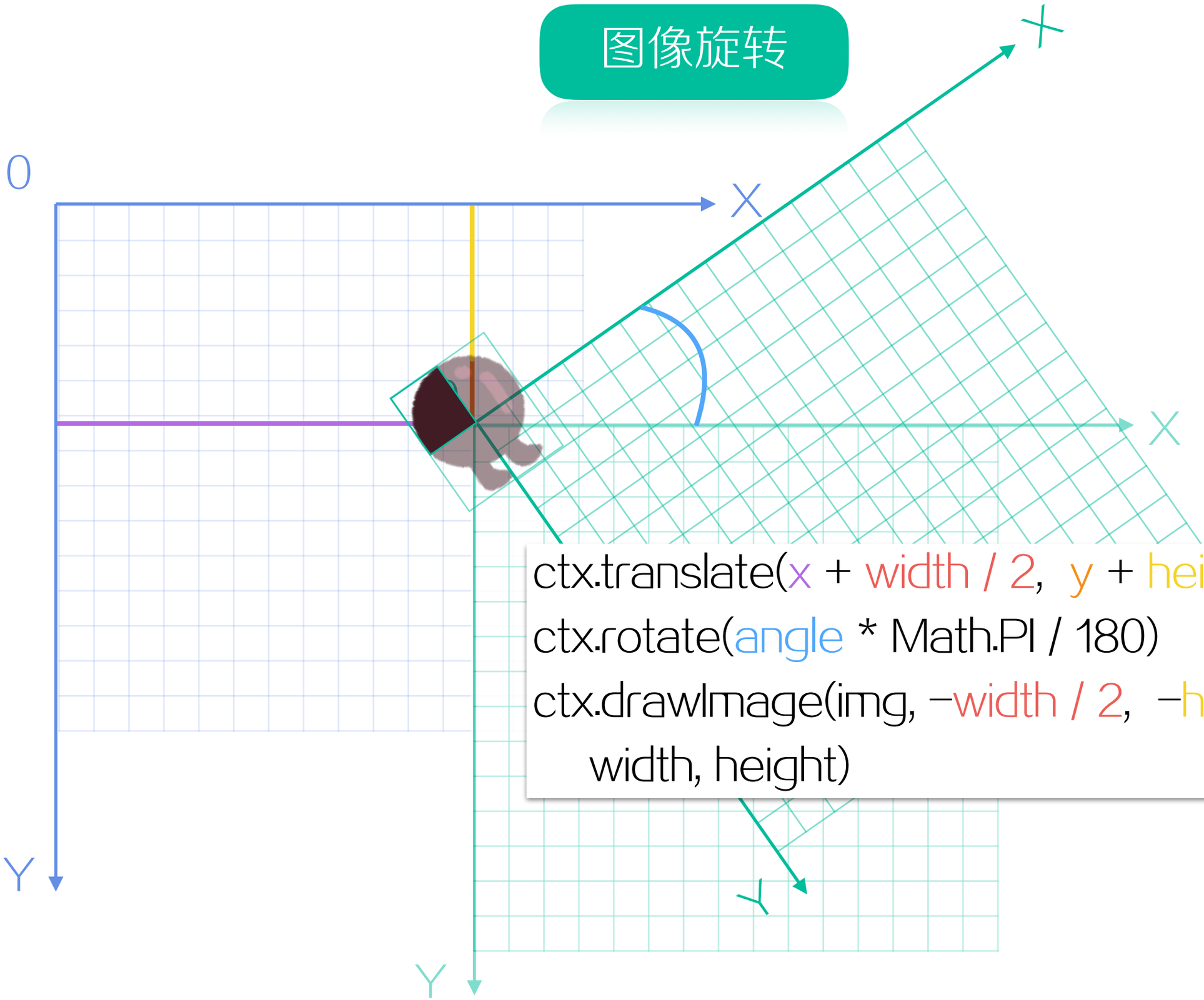
图像旋转




```
ctx.translate(x + width / 2, y + height / 2)
ctx.rotate(angle * Math.PI / 180)
```

坐标变换

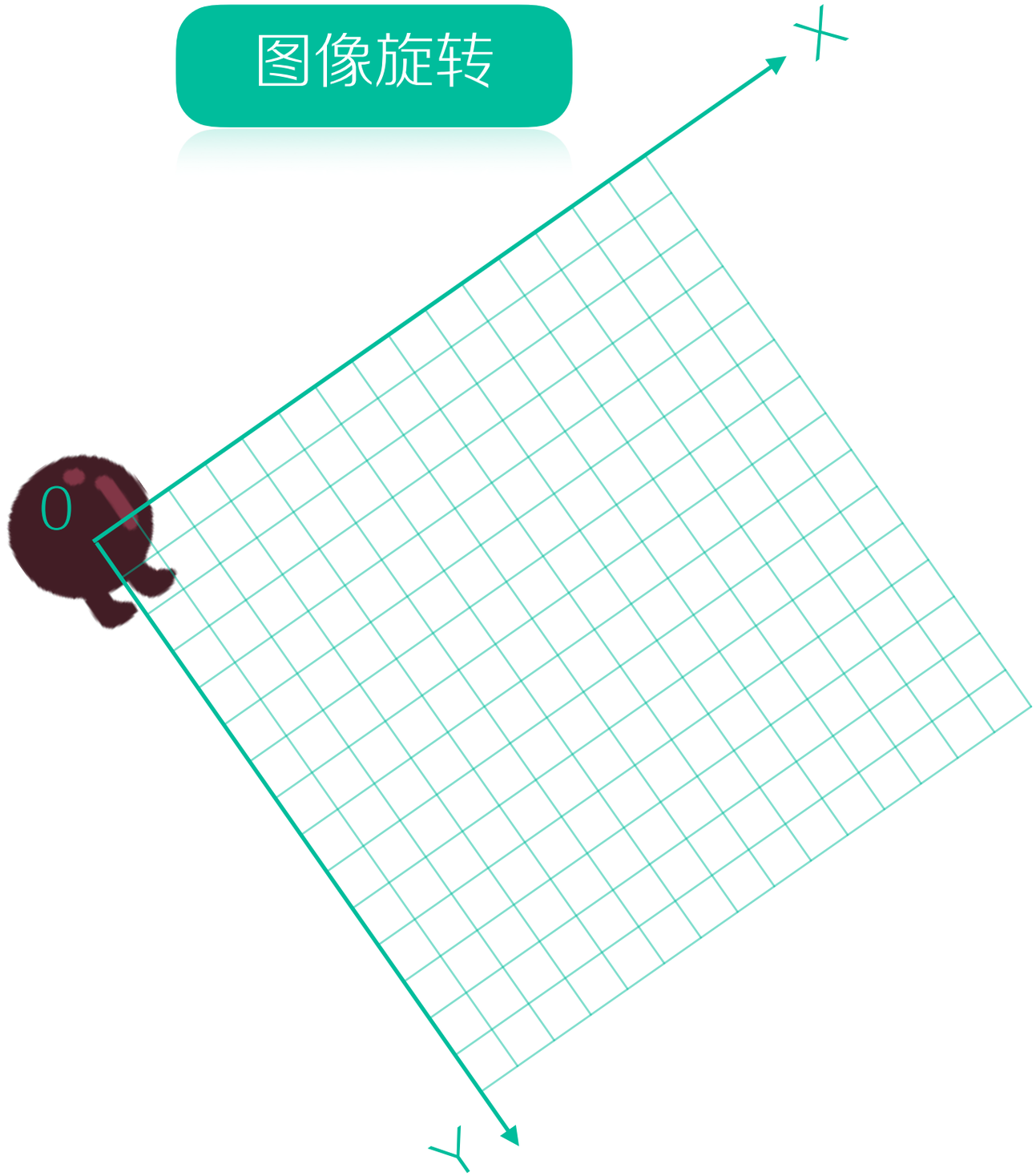
图像旋转




```
ctx.translate(x + width / 2,  y + height / 2)
ctx.rotate(angle * Math.PI / 180)
ctx.drawImage(img, -width / 2,  -height / 2,
    width, height)
```

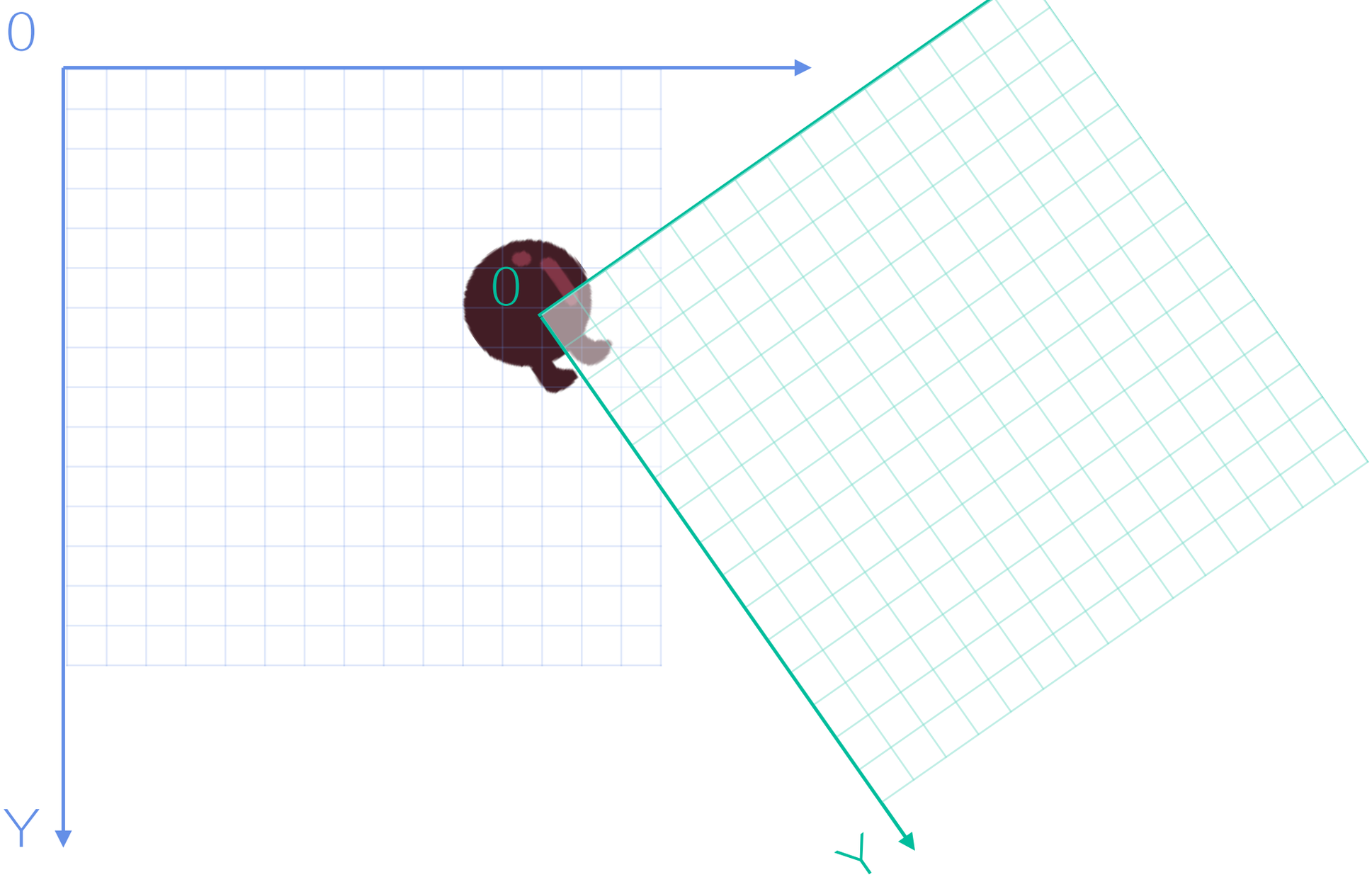
坐标变换

图像旋转



坐标变换

图像旋转



save() 与 restore()

02

save()

save() 方法用来保存 Canvas 状态的，没有参数。
每一次调用 save() 方法，当前的状态就会被推入栈中保存起来。

A. 当前应用的变形（移动/旋转/缩放）

B. strokeStyle, fillStyle, globalAlpha, lineWidth, lineCap, lineJoin, miterLimit, shadowOffsetX, shadowOffsetY, shadowBlur, shadowColor, globalCompositeOperation 的值

C. 当前的裁切路径（clipping path）

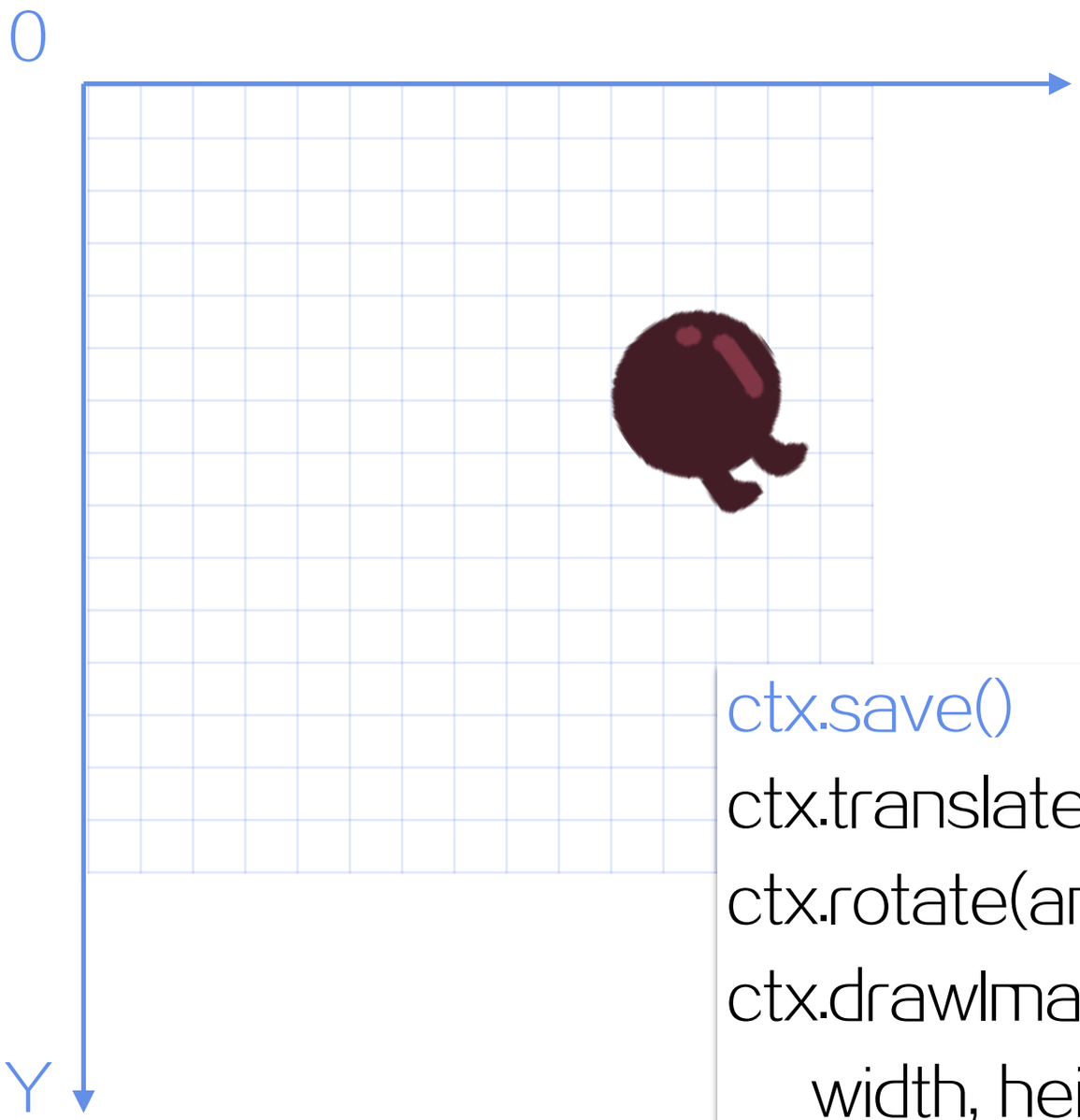
restore()

restore() 方法用来恢复 Canvas 状态，没有参数。
每一次调用 restore() 方法，上一个保存的状态就从栈中弹出，所有设定都恢复。

状态保存在栈：可以嵌套使用 save() 与 restore()

坐标变换

图像旋转



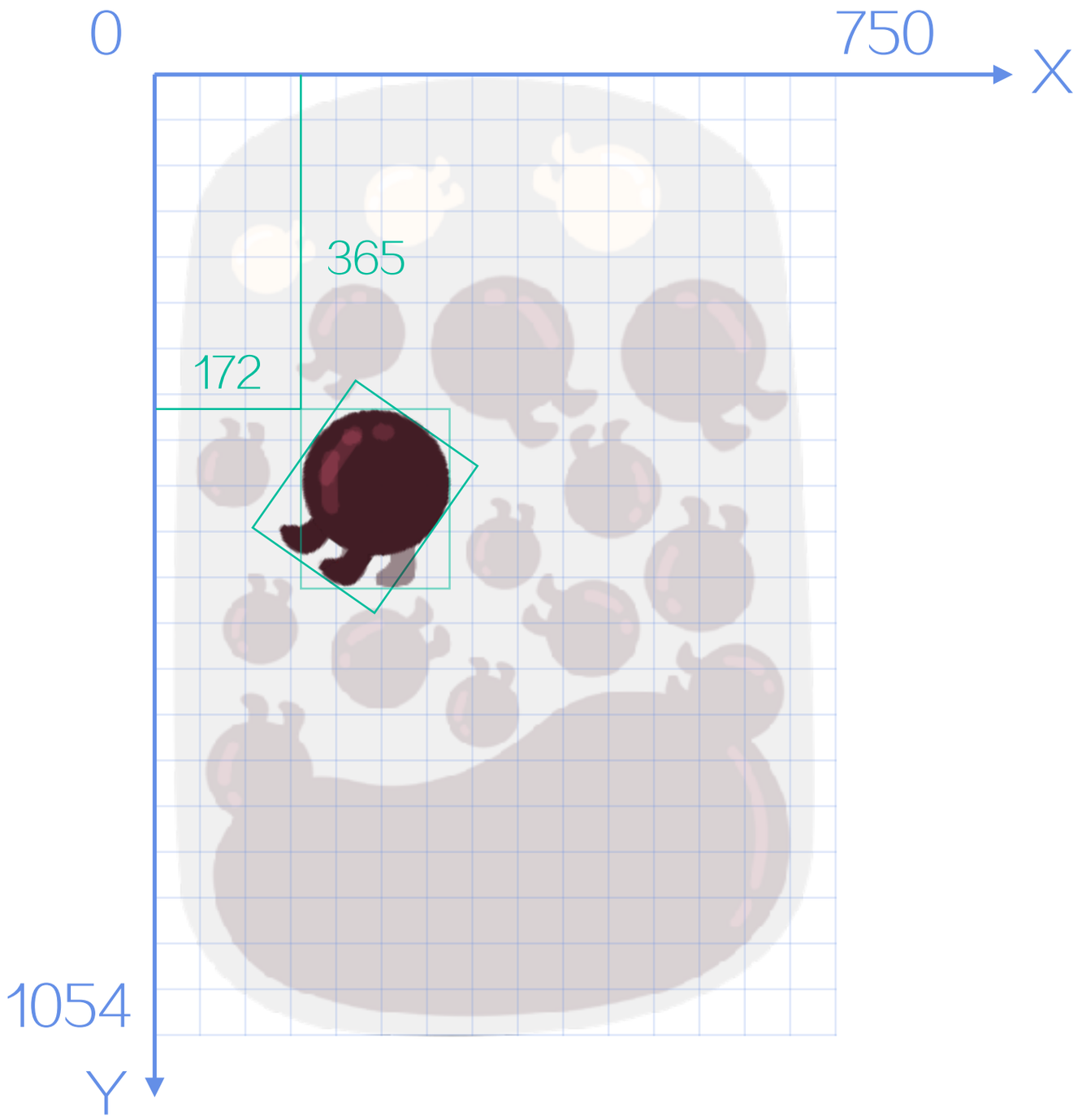

```
ctx.save()
ctx.translate(x + width / 2,  y + height / 2)
ctx.rotate(angle * Math.PI / 180)
ctx.drawImage(img, -width / 2,  -height / 2,
    width, height)
ctx.restore()
```

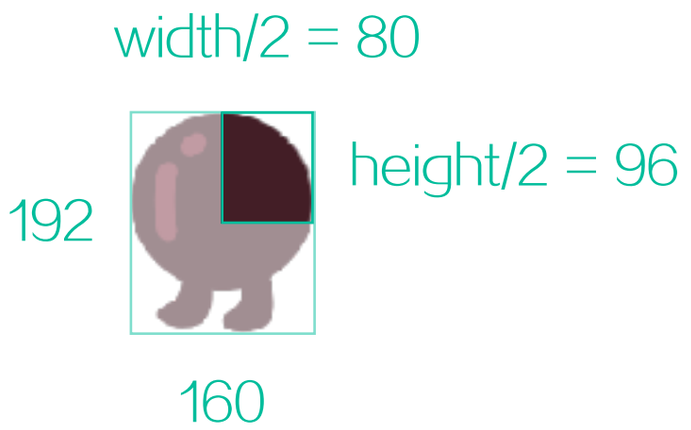
坐标变换

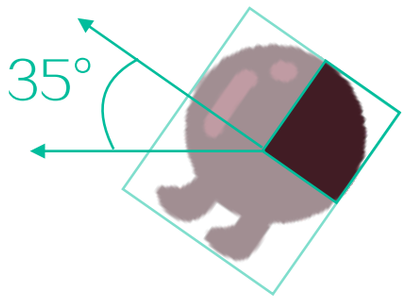
02

图像翻转

```
var img = new Image()
img.src = ‘xxxxxxx.svg’
img.onload = function(){…}
```

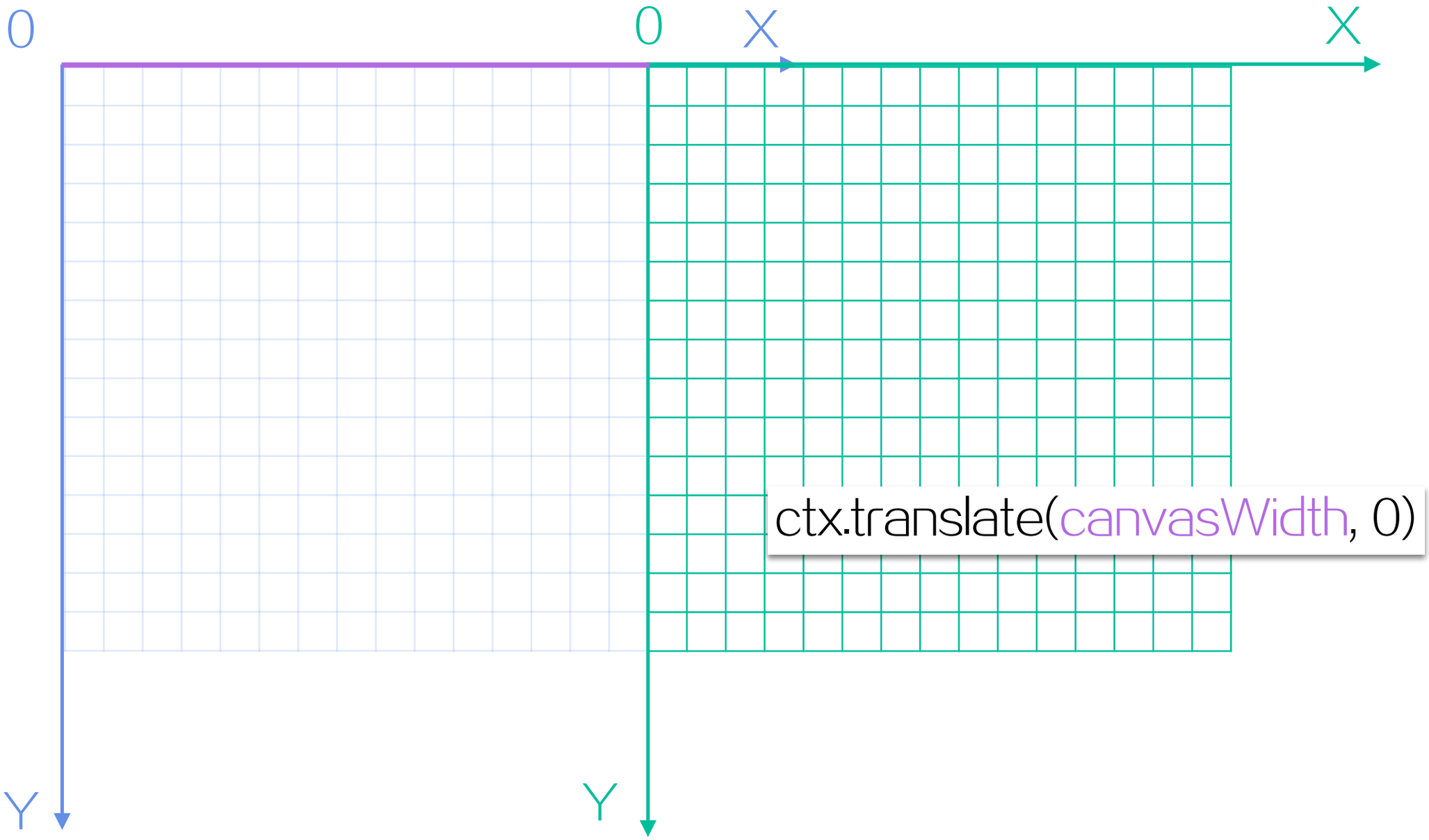






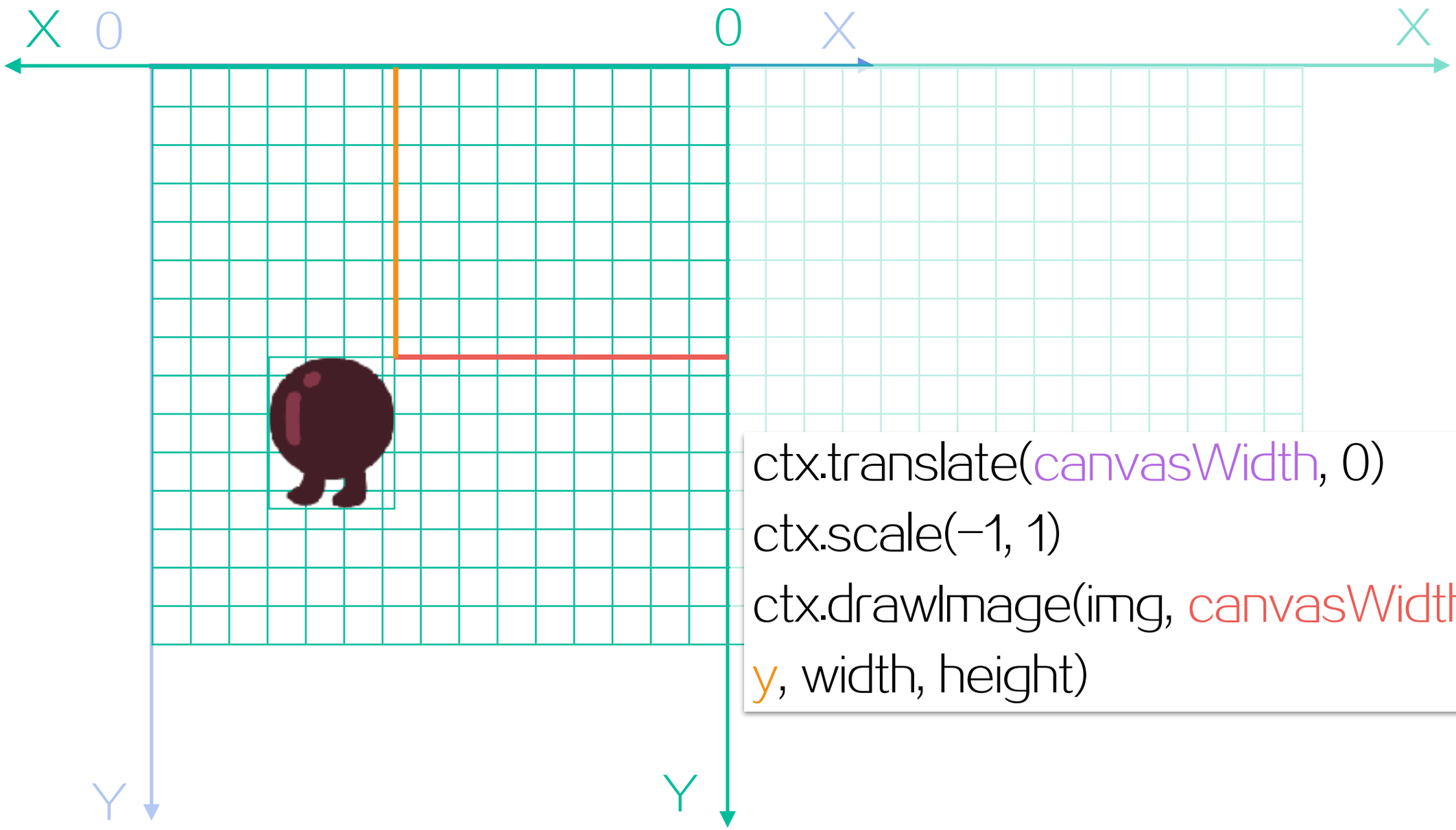
坐标变换

图像翻转



坐标变换

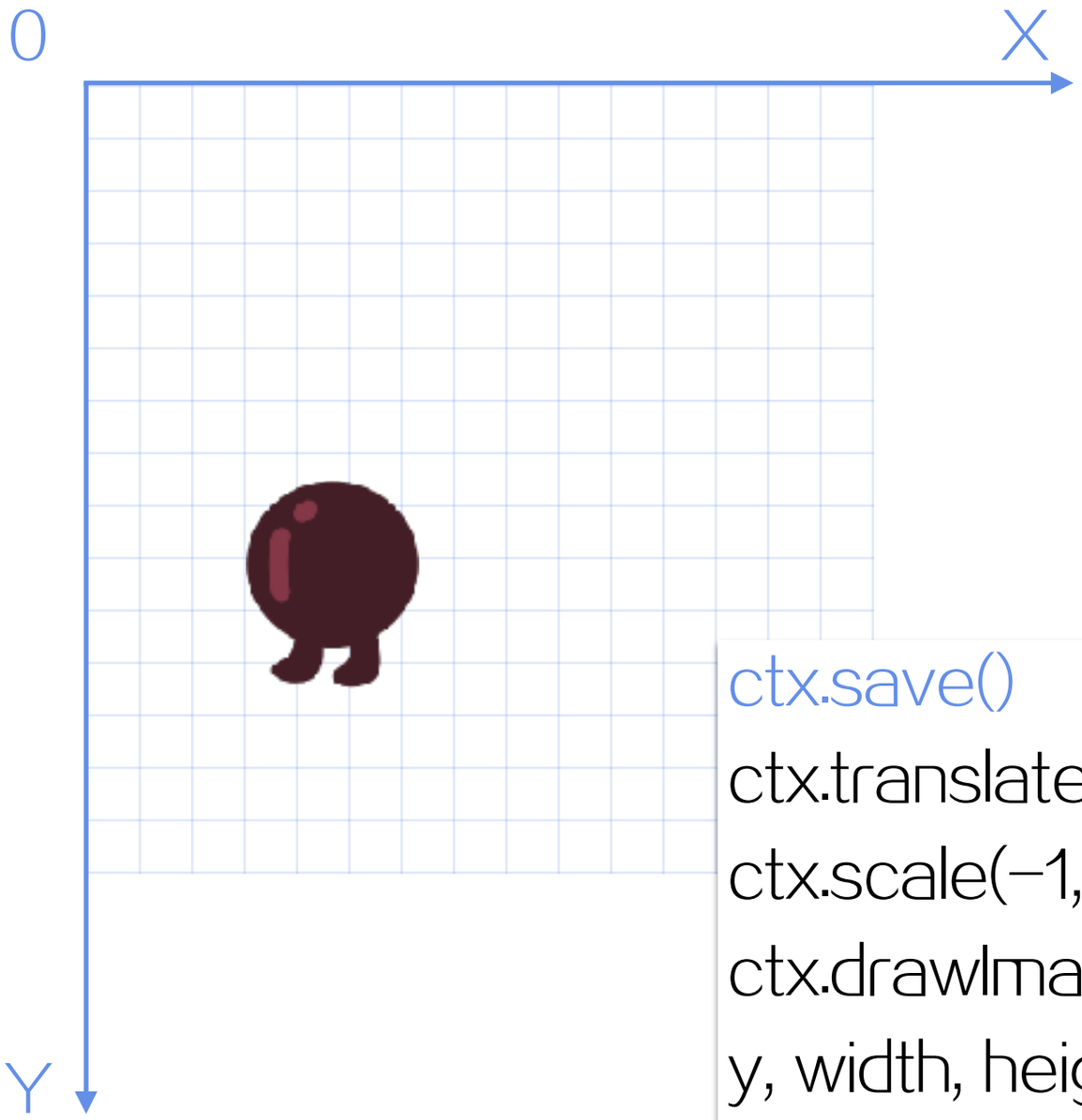
图像翻转




```
ctx.translate(canvasWidth, 0)
ctx.scale(-1, 1)
ctx.drawImage(img, canvasWidth-width-x,
y, width, height)
```

坐标变换

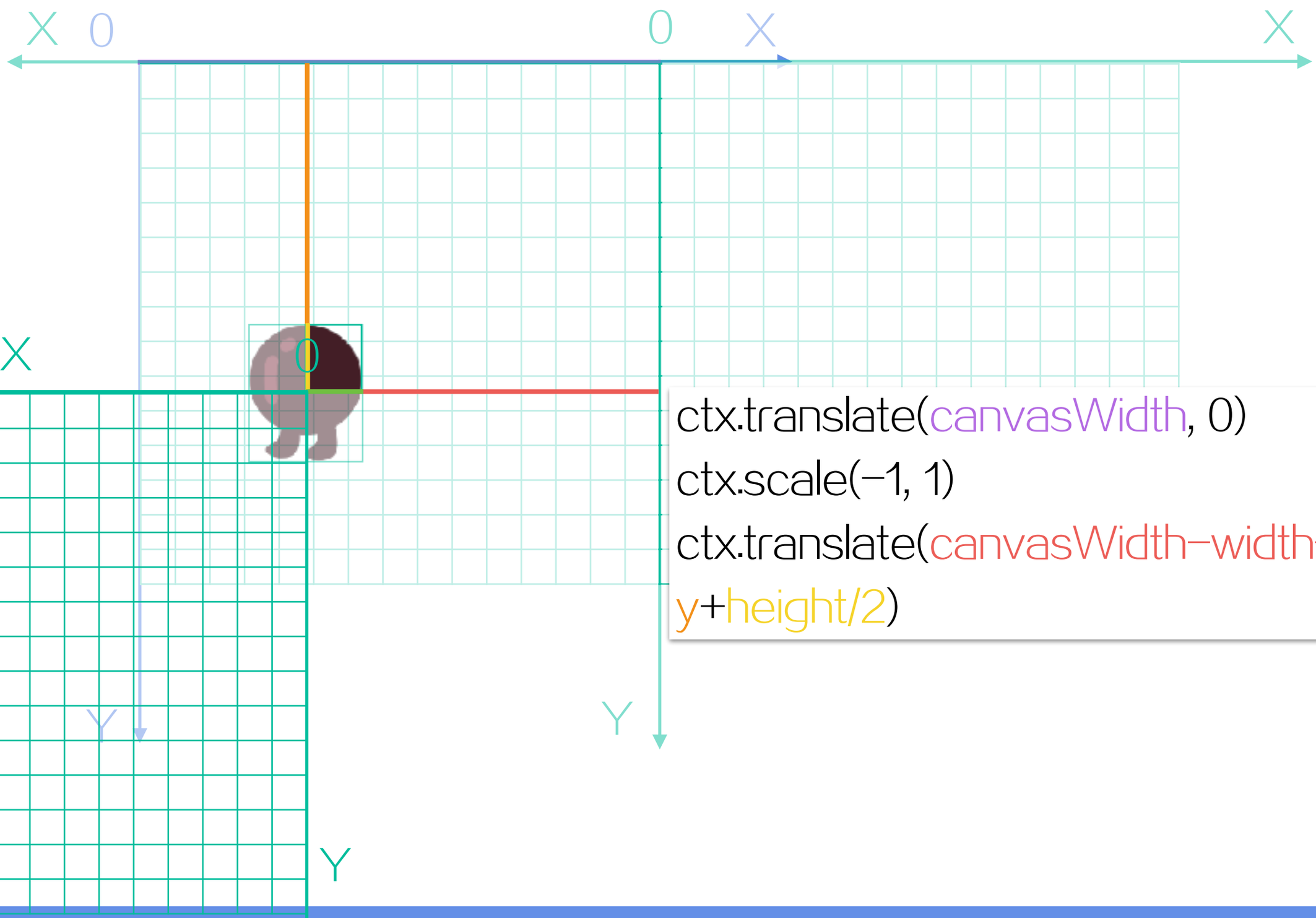
图像翻转




```
ctx.save()
ctx.translate(canvasWidth, 0)
ctx.scale(-1, 1)
ctx.drawImage(img, canvasWidth-width-x,
y, width, height)
ctx.restore()
```

坐标变换

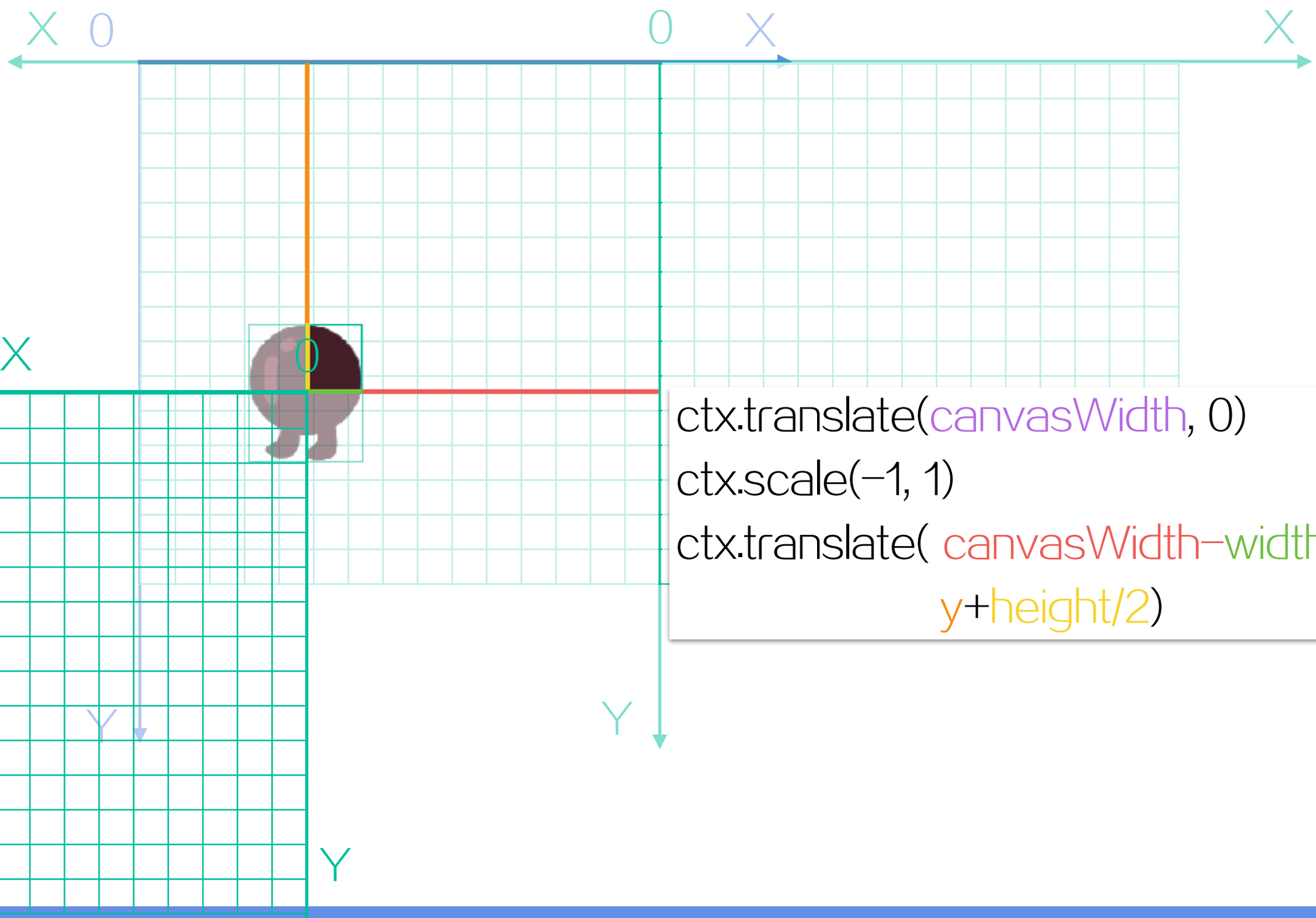
图像翻转




```
ctx.translate(canvasWidth, 0)
ctx.scale(-1, 1)
ctx.translate(canvasWidth-width-x+width/2,
y+height/2)
```

坐标变换

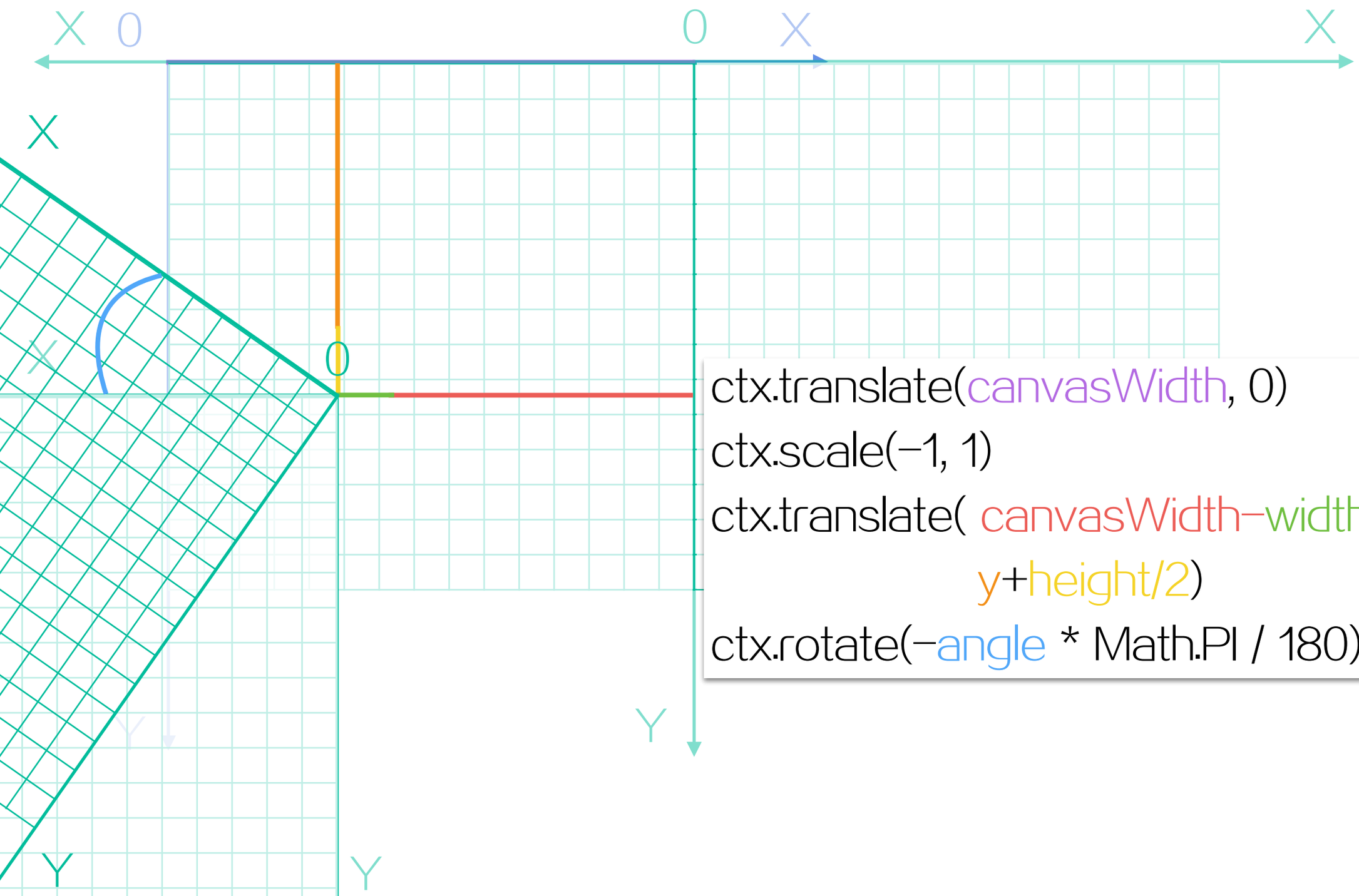
图像翻转




```
ctx.translate(canvasWidth, 0)
ctx.scale(-1, 1)
ctx.translate( canvasWidth-width/2-x,
               y+height/2)
```

坐标变换

图像翻转

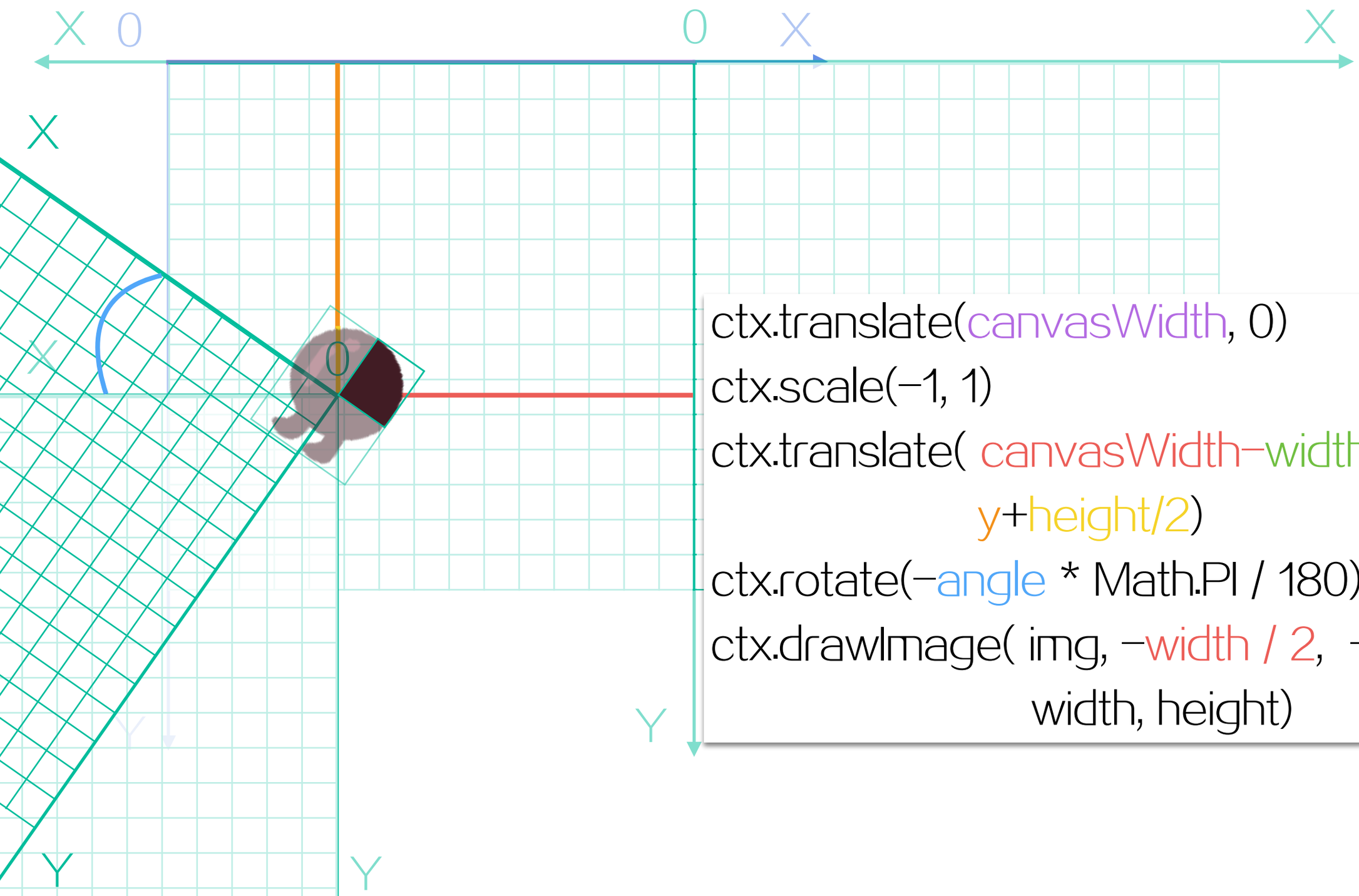



```
ctx.translate(canvasWidth, 0)
ctx.scale(-1, 1)
ctx.translate( canvasWidth-width/2-x,
                y+height/2)
ctx.rotate(-angle * Math.PI / 180);
```

坐标变换

02

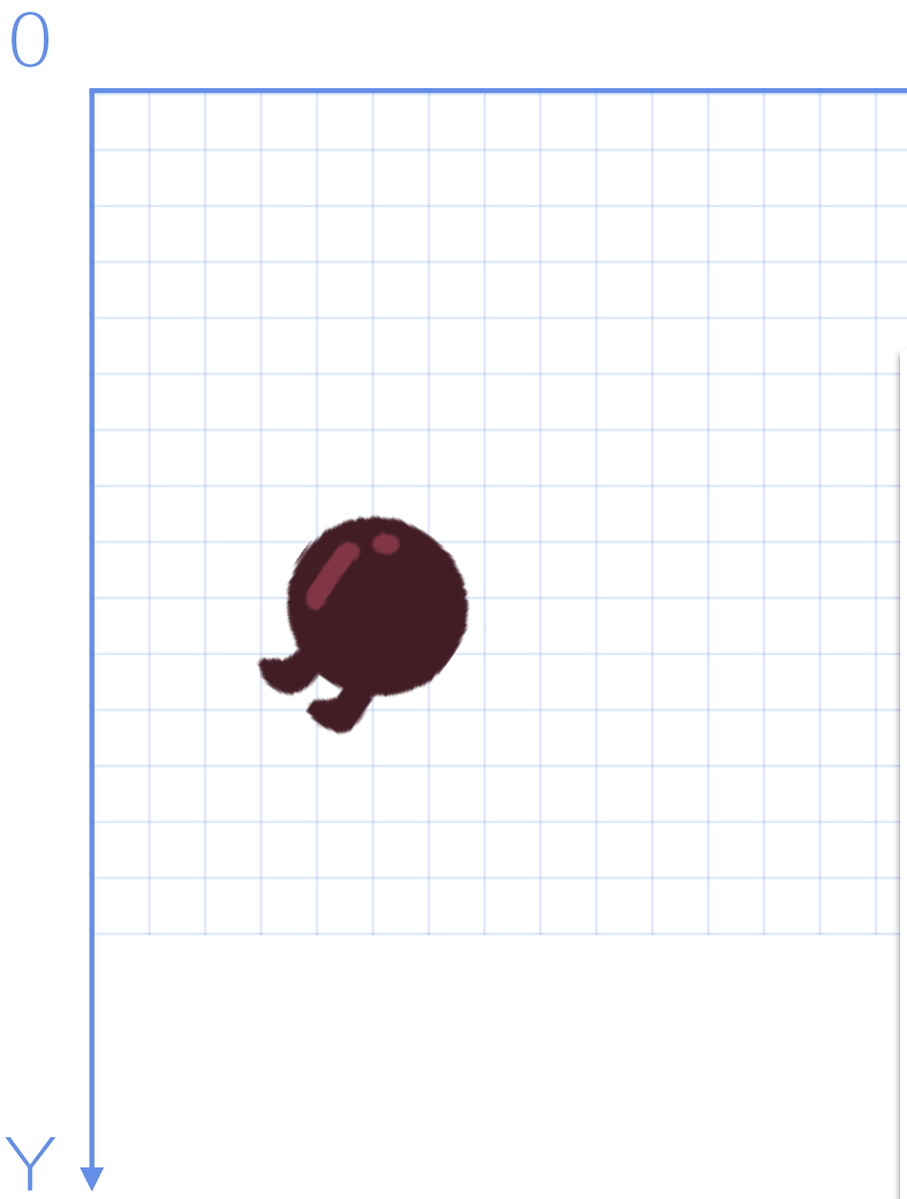
图像翻转




```
ctx.translate(canvasWidth, 0)
ctx.scale(-1, 1)
ctx.translate( canvasWidth-width/2-x,
               y+height/2)
ctx.rotate(-angle * Math.PI / 180)
ctx.drawImage( img, -width / 2,  -height / 2,
               width, height)
```

坐标变换

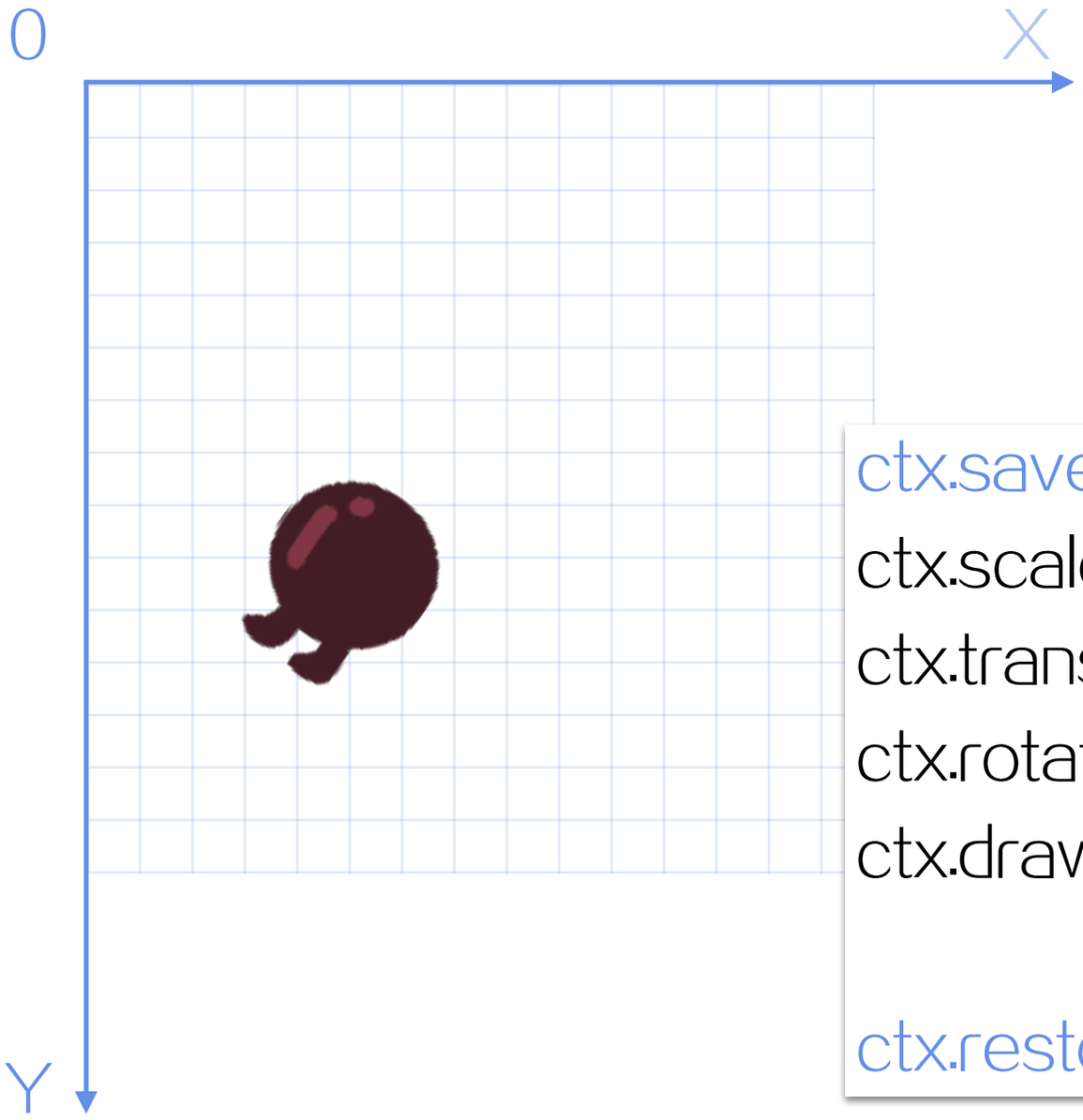
图像翻转




```
ctx.save()
ctx.translate(canvasWidth, 0)
ctx.scale(-1, 1)
ctx.translate( canvasWidth-width/2-x,
                    y+height/2)
ctx.rotate(-angle * Math.PI / 180)
ctx.drawImage( img, -width / 2,  -height / 2,
                    width, height)
ctx.restore()
```

坐标变换

图像翻转




```
ctx.save()
ctx.scale(-1, 1)
ctx.translate(-width/2-x, y+height/2)
ctx.rotate(-angle * Math.PI / 180)
ctx.drawImage(img, -width / 2,  -height / 2,
                        width, height)
ctx.restore()
```

坐标变换

02

ctx.transform (a, b, c, d, e, f)

对当前坐标系进行矩阵变换。

ctx.setTransform (a, b, c, d, e, f)

重置变形矩阵（先将当前的矩阵重置为单位矩阵(默认的坐标系)，再用相同的参数调用 transform 方法设置矩阵）

a	水平缩放绘图
b	水平倾斜绘图
c	垂直倾斜绘图
d	垂直缩放绘图
e	水平移动绘图
f	垂直移动绘图

遵循数学矩阵公式规则：

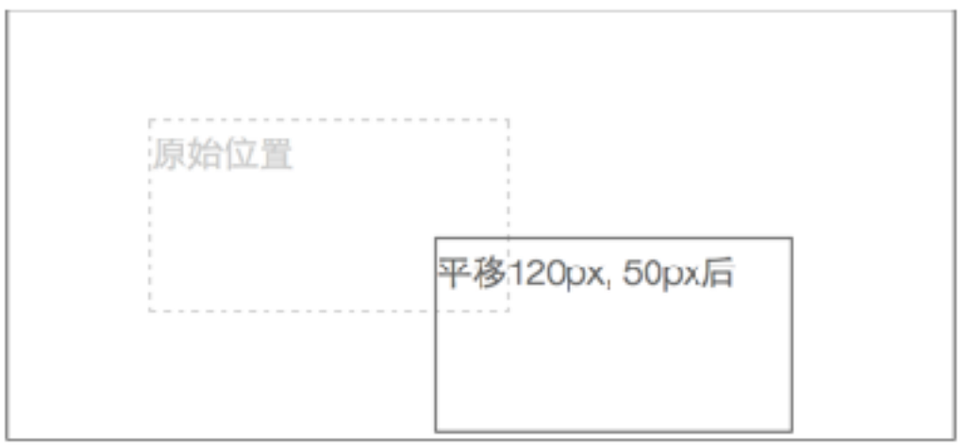
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + cy + e \\ bx + dy + f \\ 1 \end{bmatrix}$$

x’ = ax + cy + e
y’ = bx + dy + f

坐标变换

02

平移

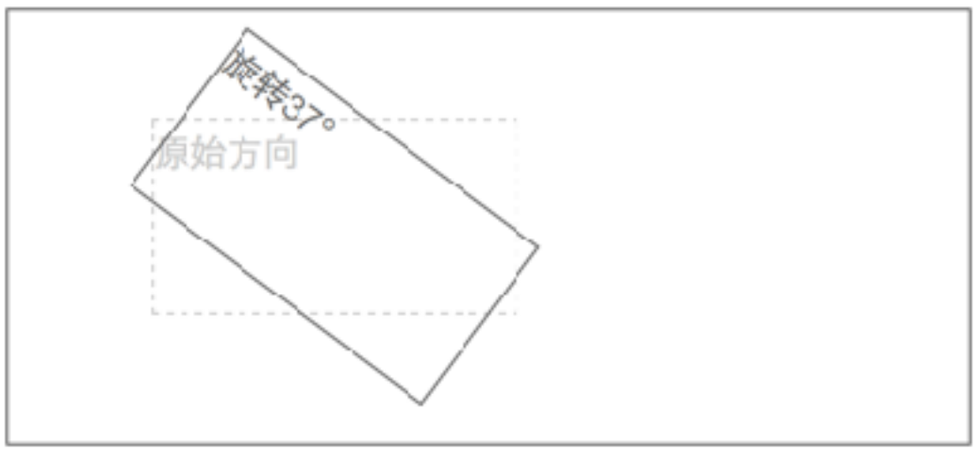


$$x' = 1x+0y+tx = x+tx$$
$$y' = 0x+1y+ty = y+ty$$

缩放和拉伸

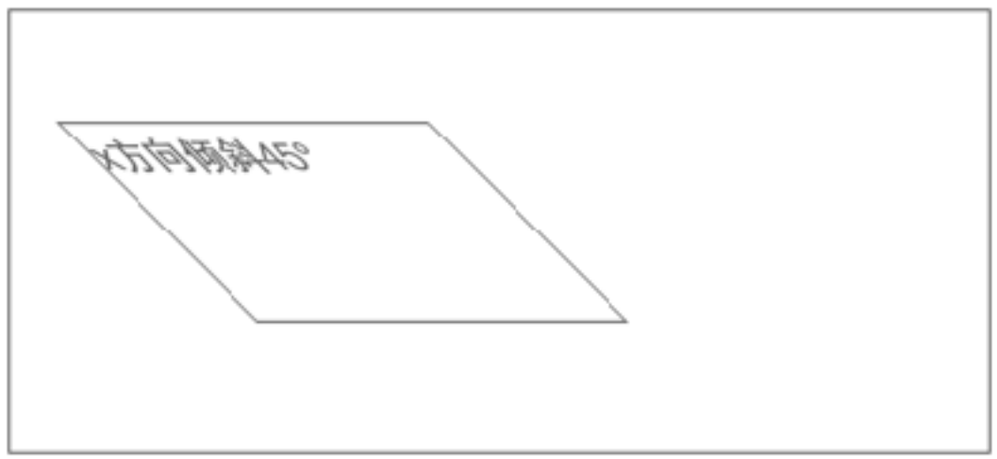
$$x' = Sx*x+0y+0 = Sx*x$$
$$y' = 0x+Sy*y+0 = Sy*y$$

旋转



$$x' = x*\cos\theta - y*\sin\theta + 0 = x*\cos\theta - y*\sin\theta$$
$$y' = x*\sin\theta + y*\cos\theta + 0 = x*\sin\theta + y*\cos\theta$$

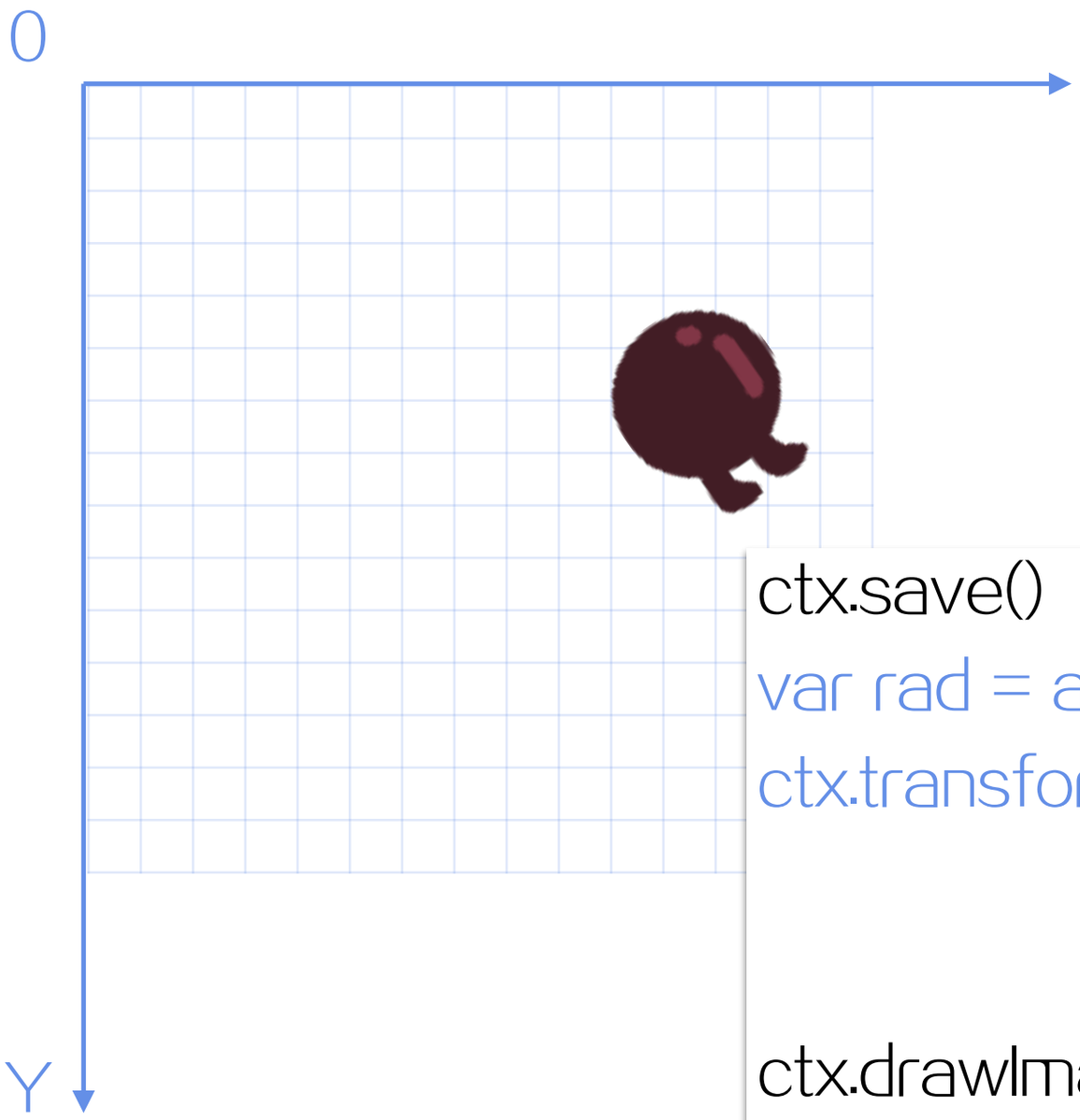
切变



$$x' = x+y*\tan(\theta\, x)+0 = x+y*\tan(\theta\, x)$$
$$y' = x*\tan(\theta\, y)+y+0 = x*\tan(\theta\, y)+y$$

坐标变换

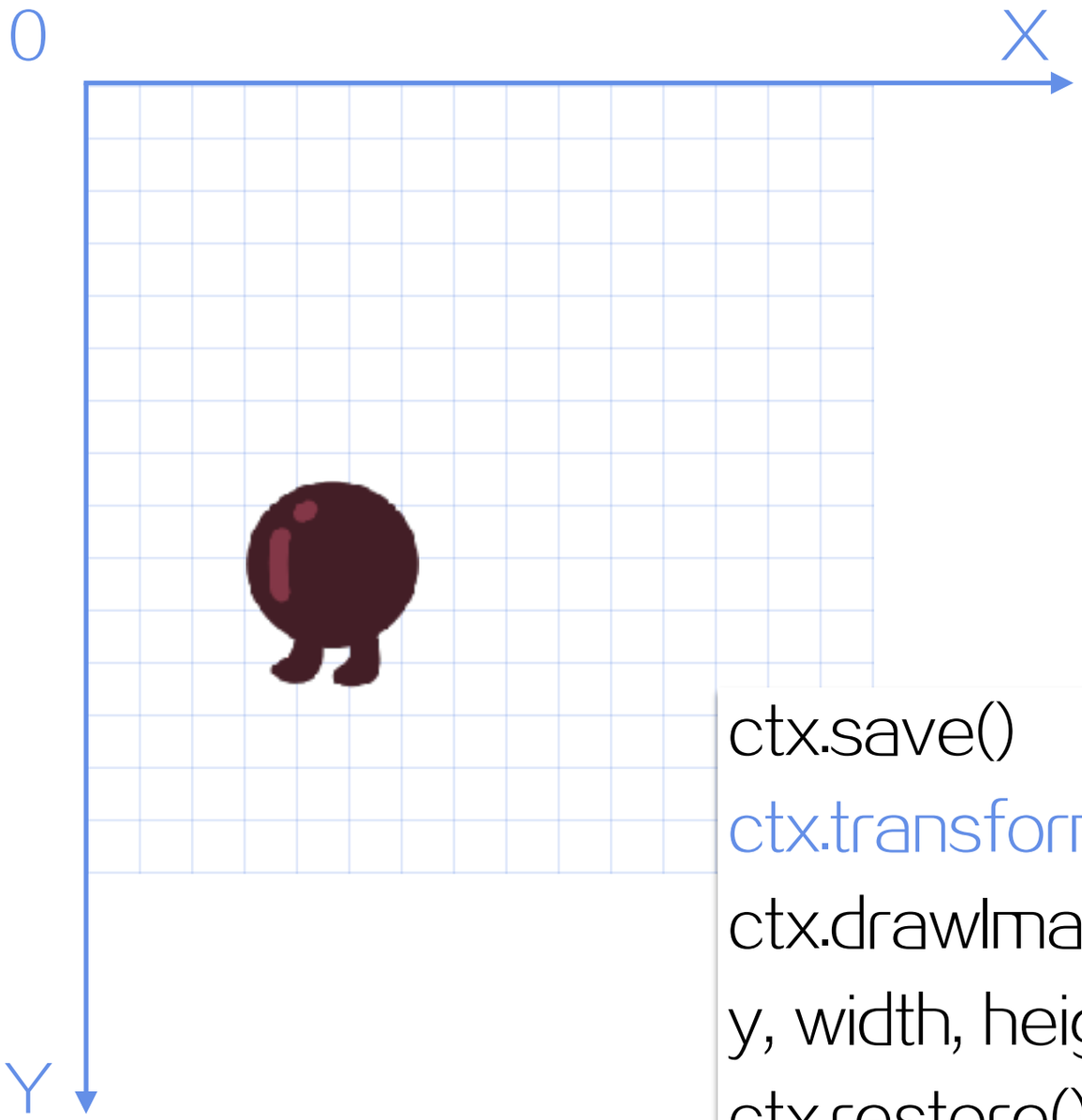
图像旋转




```
ctx.save()
var rad = angle * Math.PI/180
ctx.transform( Math.cos(rad), Math.sin(rad),
               -Math.sin(rad), Math.cos(rad),
               x + width / 2,  y + height / 2)
ctx.drawImage(img, -width / 2,  -height / 2,
                   width, height)
ctx.restore()
```

坐标变换

图像翻转




```
ctx.save()
ctx.transform( -1, 0, 0, 1, canvasWidth, 0)
ctx.drawImage(img, canvasWidth-width-x,
y, width, height)
ctx.restore()
```

参考文章

《html5 canvas.transform[转] 》
http://sumsung753.blog.163.com/blog/static/146364501201281311522752/

《Canvas学习：坐标变换》
https://www.w3cplus.com/canvas/transformation-coordinates.html

《html5 canvas 学习笔记》
https://www.gitbook.com/book/oxcow/h5-canvas-study-notes

《在HTML5中翻转图片》
https://blog.oldj.net/2011/02/09/flip-images-in-html5/