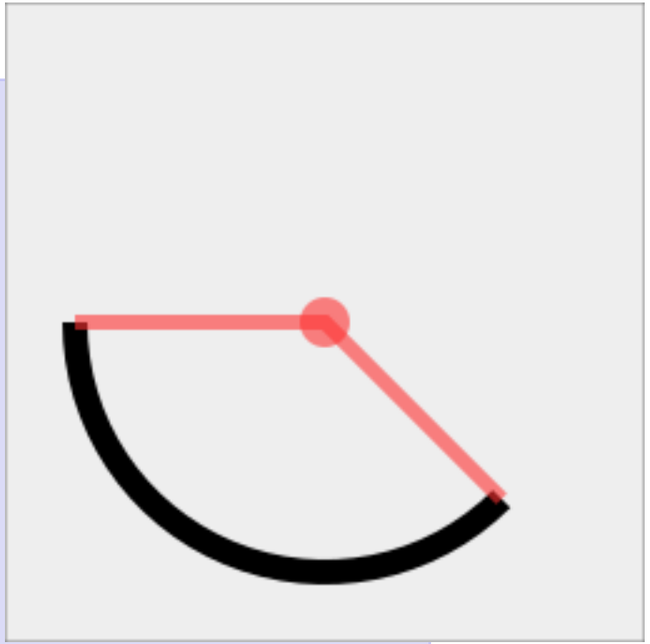
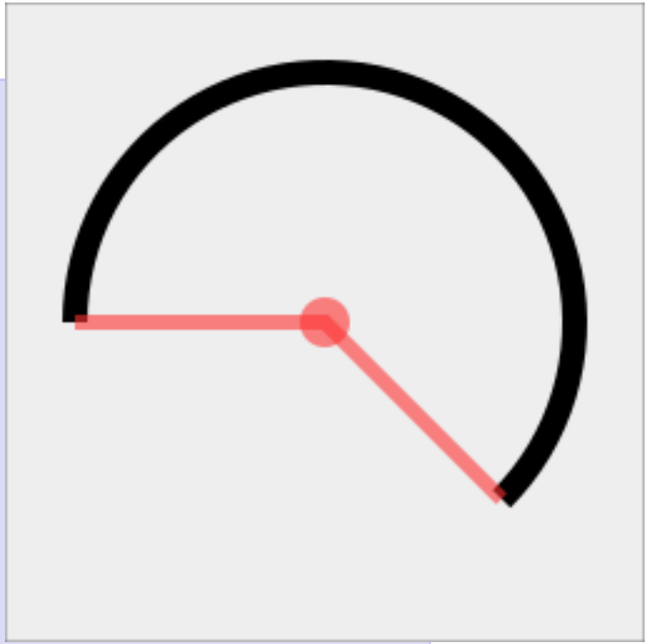
arc

```
centre = snippet_bounds.middle
radius = 100
angle1 = 45 * qah.deg
angle2 = 180 * qah.deg
gx.line_width = 10
gx.arc(centre, radius, angle1, angle2, False)
gx.stroke()
# draw helping lines
gx.source_colour = (1, 0.2, 0.2, 0.6)
gx.line_width = 6
gx.circle(centre, 10)
gx.fill()
gx.arc(centre, radius, angle1, angle1, False)
gx.line_to(centre)
gx.arc(centre, radius, angle2, angle2, False)
gx.line_to(centre)
gx.stroke()
```



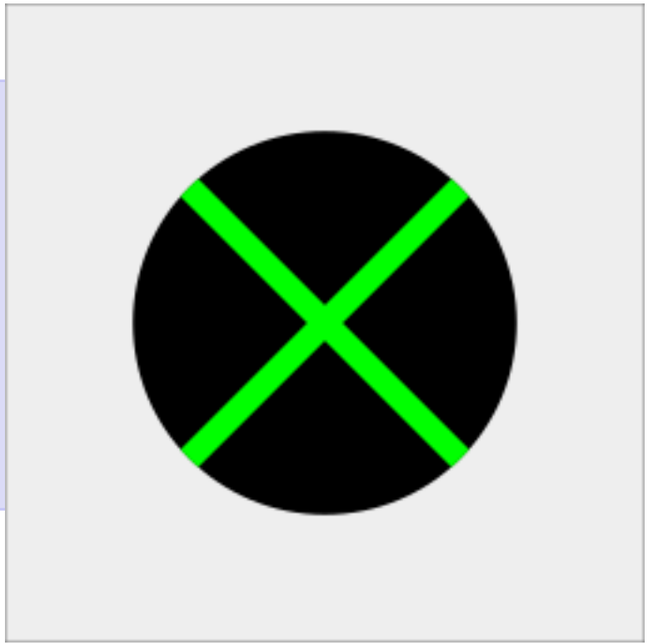
arc negative

```
centre = snippet_bounds.middle
radius = 100
angle1 = 45 * qah.deg
angle2 = 180 * qah.deg
gx.line_width = 10
gx.arc(centre, radius, angle1, angle2, True)
gx.stroke()
# draw helping lines
gx.source_colour = (1, 0.2, 0.2, 0.6)
gx.line_width = 6
gx.circle(centre, 10)
gx.fill()
gx.arc(centre, radius, angle1, angle1, False)
gx.line_to(centre)
gx.arc(centre, radius, angle2, angle2, False)
gx.line_to(centre)
gx.stroke()
```



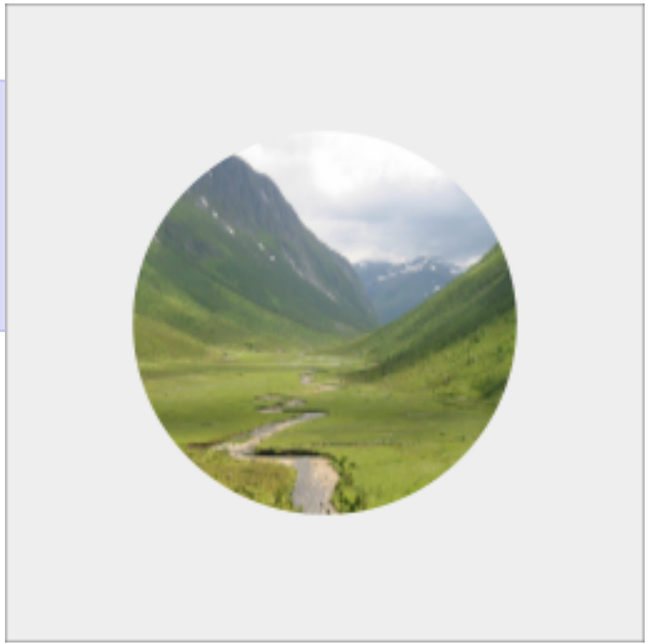
clip

```
gx.circle((128, 128), 76.8)
gx.clip()
# gx.new_path() # yes, current path is consumed by clip()
gx.rectangle(snippet_bounds)
gx.fill()
gx.source_colour = (0, 1, 0)
gx.move_to((0, 0))
gx.line_to((256, 256))
gx.move_to((256, 0))
gx.line_to((0, 256))
gx.line_width = 10
gx.stroke()
```

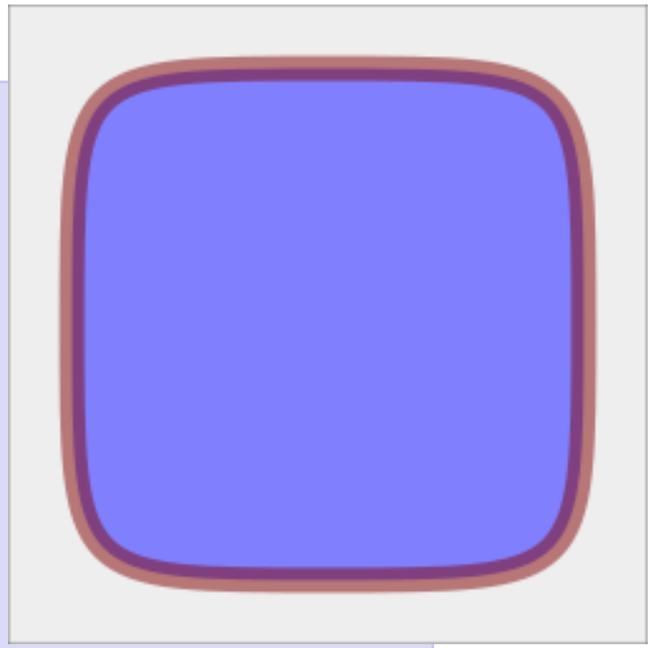


clip image

```
gx.circle((128, 128), 76.8)
gx.clip()
# gx.new_path() # yes, current path is consumed by clip()
image = qah.ImageSurface.create_from_png("data/romedalen.png")
gx.scale(256 * Vector(1, 1) / image.dimensions)
gx.set_source_surface(image, (0, 0))
gx.paint()
```



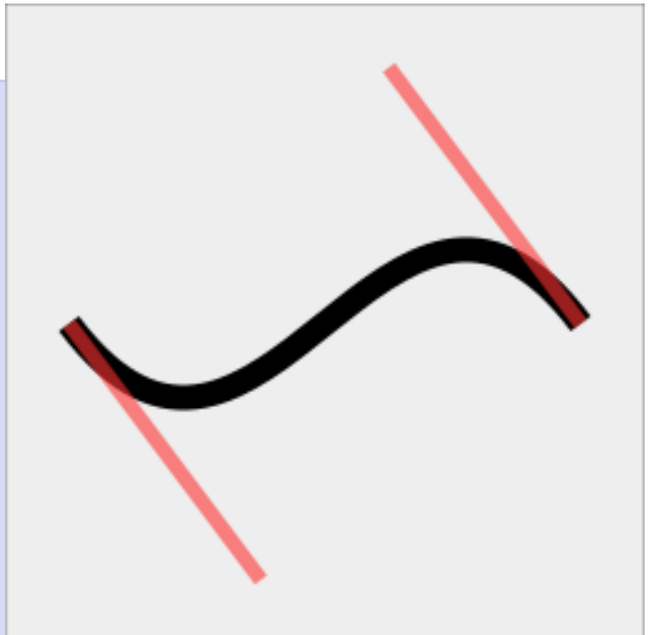
curve rectangle



```
# a custom shape that could be wrapped in a function
rect = Rect(25.6, 25.6, 204.8, 204.8) # parameter like gx.rectangle()
radius = 102.4 # and an appropriate curvature radius

if not rect.is_empty :
    p0 = rect.topleft
    p1 = p0 + rect.dimensions
    pm = (p0 + p1) / 2
    if rect.width / 2 < radius :
        if rect.height / 2 < radius :
            gx.move_to((p0.x, pm.y))
            gx.curve_to(p0, p0, (pm.x, p0.y))
            gx.curve_to((p1.x, p0.y), (p1.x, p0.y), (p1.x, pm.y))
            gx.curve_to(p1, p1, (pm.x, p1.y))
            gx.curve_to((p0.x, p1.y), (p0.x, p1.y), (p0.x, pm.y))
        else :
            gx.move_to(p0 + Vector(0, radius))
            gx.curve_to(p0, p0, (pm.x, p0.y))
            gx.curve_to((p1.x, pm.y), (p1.x, pm.y), (p1.x, p0.y + radius))
            gx.line_to(p1 - Vector(0, radius))
            gx.curve_to(p1, p1, (pm.x, p1.y))
            gx.curve_to((p0.x, p1.y), (p0.x, p1.y), (p0.x, p1.y - radius))
        #end if
    else :
        if rect.height / 2 < radius :
            gx.move_to((p0.x, pm.y))
            gx.curve_to(p0, p0, p0 + Vector(radius, 0))
            gx.line_to((p1.x - radius, p0.y))
            gx.curve_to((p1.x, p0.y), (p1.x, p0.y), (p1.x, pm.y))
            gx.curve_to(p1, p1, p1 - Vector(radius, 0))
            gx.line_to((p0.x + radius, p1.y))
            gx.curve_to((p0.x, p1.y), (p0.x, p1.y), (p0.x, pm.y))
        else :
            gx.move_to(p0 + Vector(0, radius))
            gx.curve_to(p0, p0, p0 + Vector(radius, 0))
            gx.line_to((p1.x - radius, p0.y))
            gx.curve_to((p1.x, p0.y), (p1.x, p0.y), (p1.x, p0.y + radius))
            gx.line_to(p1 - Vector(0, radius))
            gx.curve_to(p1, p1, p1 - Vector(radius, 0))
            gx.line_to((p0.x + radius, p1.y))
            gx.curve_to((p0.x, p1.y), (p0.x, p1.y), (p0.x, p1.y - radius))
        #end if
    #end if
    gx.close_path()

    gx.source_colour = (0.5, 0.5, 1)
    gx.fill_preserve()
    gx.source_colour = (0.5, 0, 0, 0.5)
    gx.line_width = 10
    gx.stroke()
#end if
```

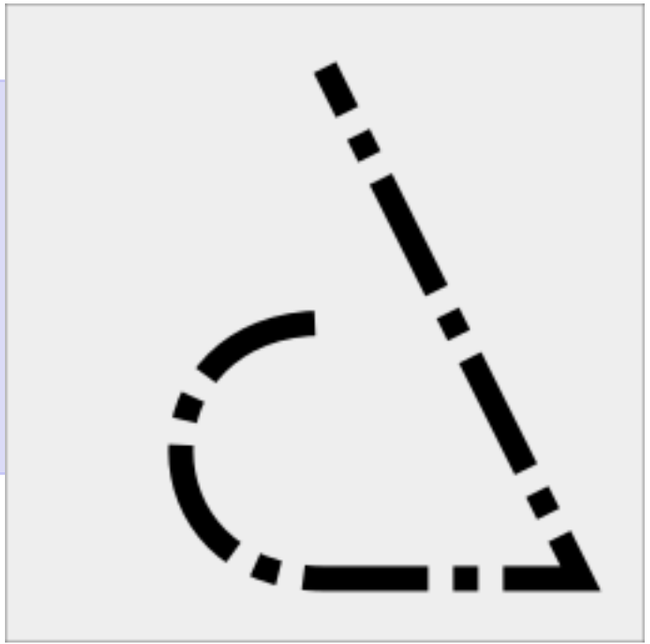
curve to

```
p = Vector(25.6, 128)
p1 = Vector(102.4, 230.4)
p2 = Vector(153.6, 25.6)
p3 = Vector(230.4, 128.0)

gx.move_to(p)
gx.curve_to(p1, p2, p3)
gx.line_width = 10
gx.stroke()

gx.source_colour = (1, 0.2, 0.2, 0.6)
gx.line_width = 6
(gx
    .move_to(p)
    .line_to(p1)
    .move_to(p2)
    .line_to(p3)
)
gx.stroke()
```



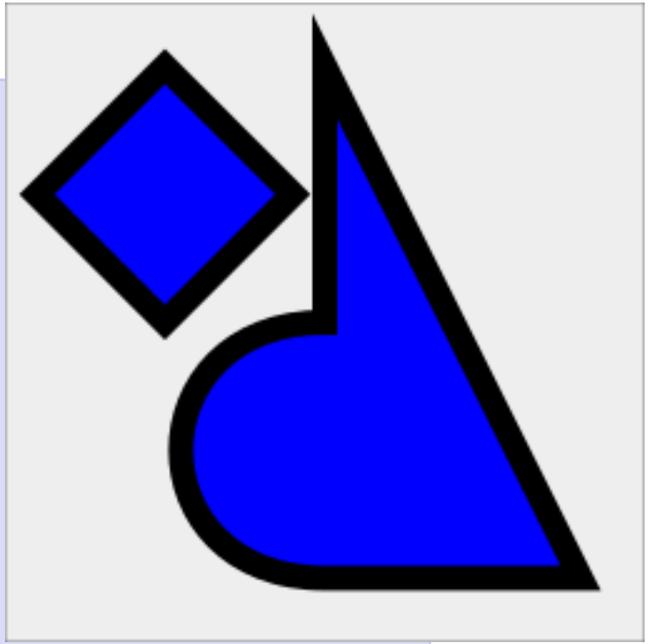
dash

```
dashes = [50, 10, 10, 10]
offset = -50

gx.dash = (dashes, offset)
gx.line_width = 10

gx.move_to((128, 25.6))
gx.line_to((230.4, 230.4))
gx.rel_line_to((-102.4, 0.0))
gx.curve_to((51.2, 230.4), (51.2, 128.0), (128.0, 128.0))
gx.stroke()
```



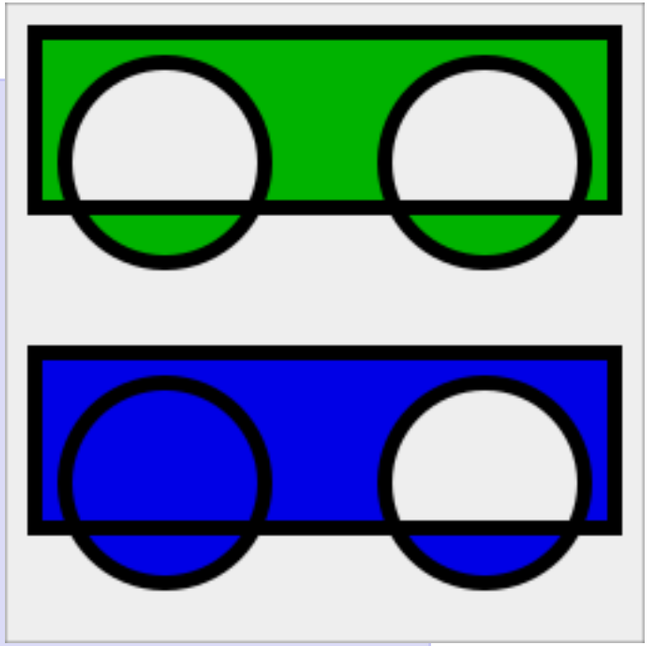
fill and stroke2

```
gx.move_to((128.0, 25.6))
gx.line_to((230.4, 230.4))
gx.rel_line_to((-102.4, 0.0))
gx.curve_to((51.2, 230.4), (51.2, 128.0), (128.0, 128.0))
gx.close_path()

gx.move_to((64.0, 25.6))
gx.rel_line_to((51.2, 51.2))
gx.rel_line_to((-51.2, 51.2))
gx.rel_line_to((-51.2, -51.2))
gx.close_path()

gx.line_width = 10.0
gx.source_colour = (0, 0, 1)
gx.fill_preserve()
gx.source_colour = (0, 0, 0)
gx.stroke()
```



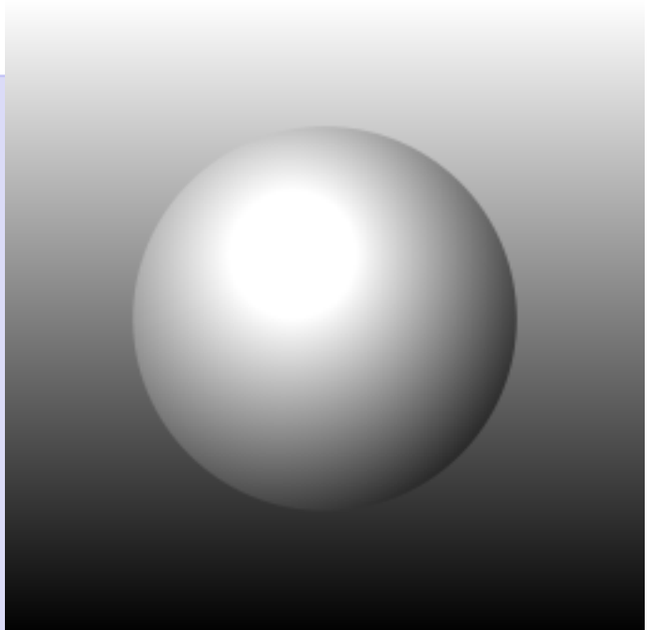
fill style

```
gx.line_width = 6
fig_rect = Rect(12, 12, 232, 70)
gx.rectangle(fig_rect)
(gx
    .new_sub_path()
    .arc((64, 64), 40, 0, qah.circle, False)
    .new_sub_path()
    .arc((192, 64), 40, 0, -qah.circle, True)
)

gx.fill_rule = CAIRO.FILL_RULE_EVEN_ODD
(gx
    .set_source_colour((0, 0.7, 0))
    .fill_preserve()
    .set_source_colour((0, 0, 0))
    .stroke()
)

gx.translate((0, 128))
gx.rectangle(fig_rect)
(gx
    .new_sub_path()
    .arc((64, 64), 40, 0, qah.circle, False)
    .new_sub_path()
    .arc((192, 64), 40, 0, -qah.circle, True)
)

gx.fill_rule = CAIRO.FILL_RULE_WINDING
(gx
    .set_source_colour((0, 0, 0.9))
    .fill_preserve()
    .set_source_colour((0, 0, 0))
    .stroke()
)
```



gradient

```
pat = qah.Pattern.create_linear(p0 = (0, 0), p1 = (0, 256.0))
pat.add_colour_stop(1, (0, 0, 0, 1))
pat.add_colour_stop(0, (1, 1, 1, 1))
gx.rectangle(Rect(0, 0, 256, 256))
gx.source = pat
gx.fill()

pat = qah.Pattern.create_radial \
  (
    c0 = (115.2, 102.4),
    r0 = 25.6,
    c1 = (102.4, 102.4),
    r1 = 128.0
  )
pat.add_colour_stop(0, (1, 1, 1, 1))
pat.add_colour_stop(1, (0, 0, 0, 1))
gx.source = pat
gx.circle((128.0, 128.0), 76.8)
gx.fill()
```

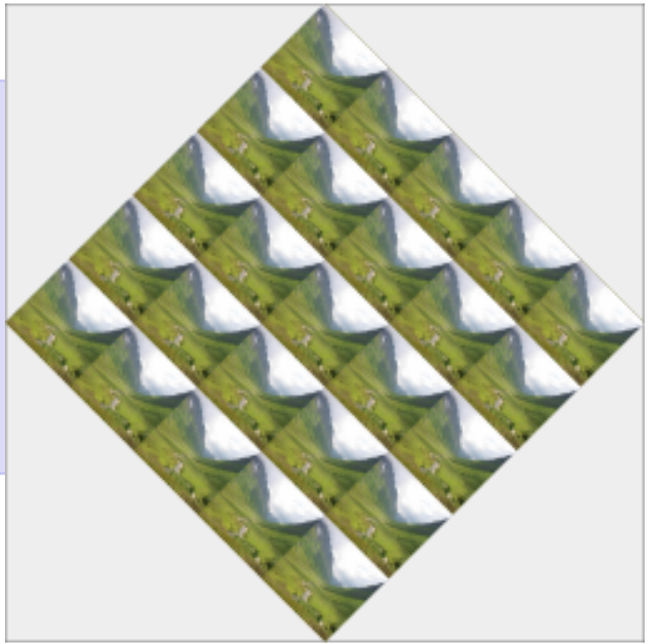


image

```
image = qah.ImageSurface.create_from_png("data/romedalen.png")
gx.translate(snippet_bounds.dimensions / 2)
gx.rotate(45 * qah.deg)
gx.scale(Vector(256, 256) / image.dimensions)
gx.translate(- image.dimensions / 2)
gx.set_source_surface(image, (0, 0))
gx.paint()
```

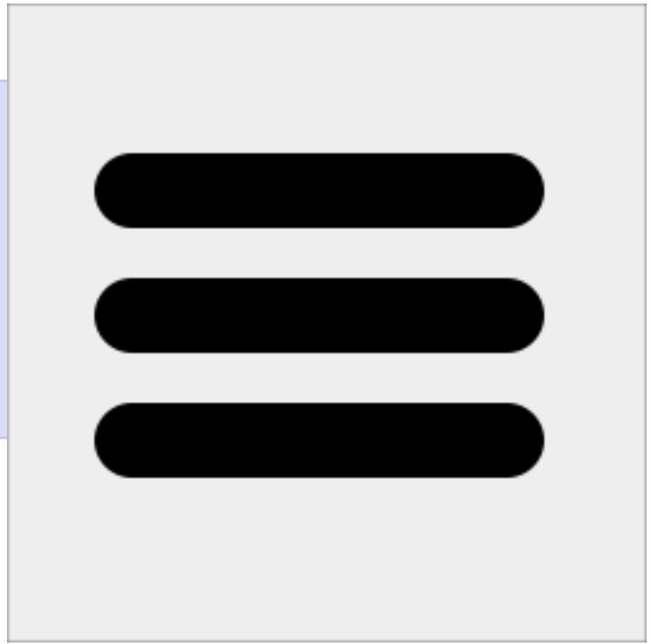
imagepattern

```
image = qah.ImageSurface.create_from_png("data/romedalen.png")
pattern = qah.Pattern.create_for_surface(image)
pattern.extend = CAIRO.EXTEND_REPEAT
gx.translate(snippet_bounds.dimensions / 2)
gx.rotate(math.pi / 4)
gx.scale(1 / math.sqrt(2))
gx.translate(- snippet_bounds.dimensions / 2)
pattern.matrix = Matrix.scale(image.dimensions / snippet_bounds.dimensions * 5.0)
gx.source = pattern
gx.rectangle(snippet_bounds)
gx.fill()
```



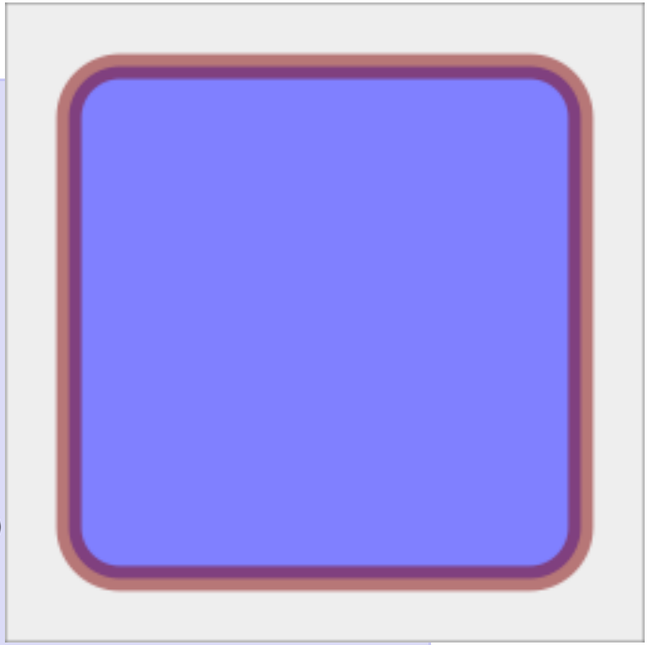
multi segment caps

```
gx.move_to((50.0, 75.0))
gx.line_to((200.0, 75.0))
gx.move_to((50.0, 125.0))
gx.line_to((200.0, 125.0))
gx.move_to((50.0, 175.0))
gx.line_to((200.0, 175.0))

gx.line_width = 30
gx.line_cap = CAIRO.LINE_CAP_ROUND
gx.stroke()
```



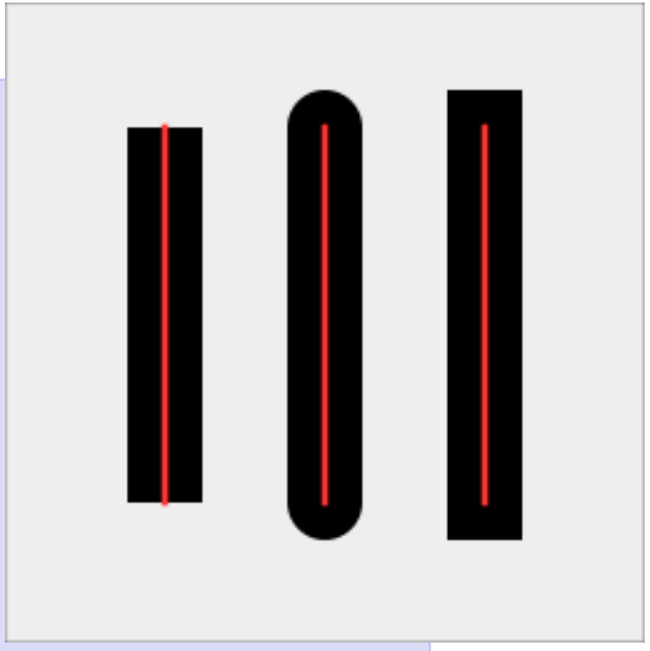
rounded rectangle

```
# a custom shape that could be wrapped in a function
rect = Rect(25.6, 25.6, 204.8, 204.8) # parameter like gx.rectangle()
aspect = 1.0 # aspect ratio
corner_radius = rect.height / 10.0 # and corner curvature radius

radius = corner_radius / aspect
p1 = rect.topleft
p2 = rect.botright
gx.new_sub_path()
gx.arc((p2.x - radius, p1.y + radius), radius, - 90 * qah.deg, 0, False)
gx.arc((p2.x - radius, p2.y - radius), radius, 0, 90 * qah.deg, False)
gx.arc((p1.x + radius, p2.y - radius), radius, 90 * qah.deg, 180 * qah.deg, False)
gx.arc((p1.x + radius, p1.y + radius), radius, 180 * qah.deg, 270 * qah.deg, False)
gx.close_path()

gx.source_colour = (0.5, 0.5, 1)
gx.fill_preserve()
gx.source_colour = (0.5, 0, 0, 0.5)
gx.line_width = 10.0
gx.stroke()
```



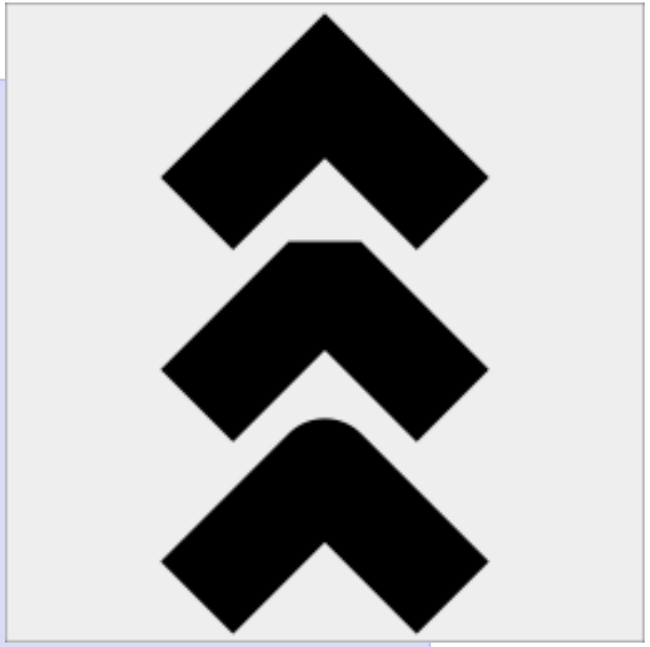
set line cap

```
gx.line_width = 30
gx.line_cap = CAIRO.LINE_CAP_BUTT # default
gx.move_to((64.0, 50.0)).line_to((64.0, 200.0))
gx.stroke()
gx.line_cap = CAIRO.LINE_CAP_ROUND
gx.move_to((128.0, 50.0)).line_to((128.0, 200.0))
gx.stroke()
gx.line_cap = CAIRO.LINE_CAP_SQUARE
gx.move_to((192.0, 50.0)).line_to((192.0, 200.0))
gx.stroke()

# draw helping lines
gx.source_colour = (1, 0.2, 0.2)
gx.line_width = 2.56
gx.move_to((64.0, 50.0)).line_to((64.0, 200.0))
gx.move_to((128.0, 50.0)).line_to((128.0, 200.0))
gx.move_to((192.0, 50.0)).line_to((192.0, 200.0))
gx.stroke()
```



set line join

```
gx.line_width = 40.96
gx.move_to((76.8, 84.48))
gx.rel_line_to((51.2, -51.2))
gx.rel_line_to((51.2, 51.2))
gx.line_join = CAIRO.LINE_JOIN_MITRE # default
gx.stroke()

gx.move_to((76.8, 161.28))
gx.rel_line_to((51.2, -51.2))
gx.rel_line_to((51.2, 51.2))
gx.line_join = CAIRO.LINE_JOIN_BEVEL
gx.stroke()

gx.move_to((76.8, 238.08))
gx.rel_line_to((51.2, -51.2))
gx.rel_line_to((51.2, 51.2))
gx.line_join = CAIRO.LINE_JOIN_ROUND
gx.stroke()
```

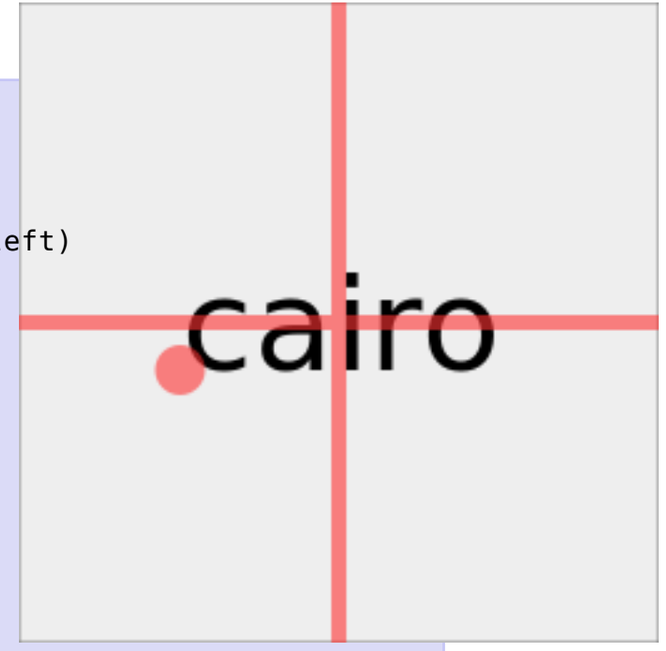


text

```
gx.select_font_face("Sans", CAIRO.FONT_SLANT_NORMAL, CAIRO.FONT_WEIGHT_BOLD)
gx.set_font_size(90.0)

gx.move_to((10.0, 135.0))
gx.show_text("Hello")

gx.move_to((70.0, 165.0))
gx.text_path("void")
gx.source_colour = (0.5, 0.5, 1)
gx.fill_preserve()
gx.source_colour = (0, 0, 0)
gx.line_width = 2.56
gx.stroke()

# draw helping markers
gx.source_colour = (1, 0.2, 0.2, 0.6)
gx.circle((10.0, 135.0), 5.12)
gx.close_path()
gx.circle((70.0, 165.0), 5.12)
gx.fill()
```

text align centre

```
text = "cairo"
gx.select_font_face("Sans", CAIRO.FONT_SLANT_NORMAL, CAIRO.FONT_WEIGHT_NORMAL)
gx.set_font_size(52.0)
extents = gx.text_extents(text)
pos = snippet_bounds.middle - (extents.bounds.dimensions / 2 + extents.bounds.topleft)
gx.move_to(pos)
gx.show_text(text)

# draw helping lines
gx.source_colour = (1, 0.2, 0.2, 0.6)
gx.line_width = 6.0
gx.circle(pos, 10.0)
gx.fill()
gx.move_to((128.0, 0))
gx.rel_line_to((0, 256))
gx.move_to((0, 128.0))
gx.rel_line_to((256, 0))
gx.stroke()
```



text extents

```
text = "cairo"
gx.select_font_face("Sans", CAIRO.FONT_SLANT_NORMAL, CAIRO.FONT_WEIGHT_NORMAL)
gx.set_font_size(100.0)
extents = gx.text_extents(text)

pos = Vector(25.0, 150.0)
gx.move_to(pos)
gx.show_text(text)

# draw helping lines
gx.source_colour = (1, 0.2, 0.2, 0.6)
gx.line_width = 6.0
gx.circle(pos, 10.0)
gx.fill()
gx.move_to(pos)
gx.rel_line_to((0, - extents.height))
gx.rel_line_to((extents.width, 0))
gx.rel_line_to((extents.x_bearing, - extents.y_bearing))
gx.stroke()
```