Gráficas

Gráficas

- Objetivo: hacer una aplicación para graficar.

Gráficas

- Extender la clase *View*.
- Sobreponer el método *onDraw*(*Canvas canvas*).
- La clase *Canvas* tiene métodos para dibujar:
  - *drawBitmap*(...) – dibuja una imagen.
  - *drawCircle*(...) – dibuja un círculo.
  - *drawLine*(...) – dibuja una linea.
- La mayoría de estos métodos necesita un objeto *Paint* para especificar el color, ancho de la línea, etc.

Ejemplo

- Hacer un programa para dibujar una línea roja.

Clase *PintaLinea*

```
public class PintaLinea extends View {
    private Paint redPaint;

    public PintaLinea(Context context)
    {
        super(context);

        redPaint = new Paint();
        redPaint.setColor(Color.RED);
        redPaint.setStrokeWidth(2);
        redPaint.setAntiAlias(true);
    }

    public void onDraw(Canvas canvas)
    {
        canvas.drawLine(0, 0, getWidth(), getHeight(), redPaint);
    }
}
```

Actividad

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        PintaLinea p = new PintaLinea(this);
        setContentView(p);
    }
}
```

Ejemplo

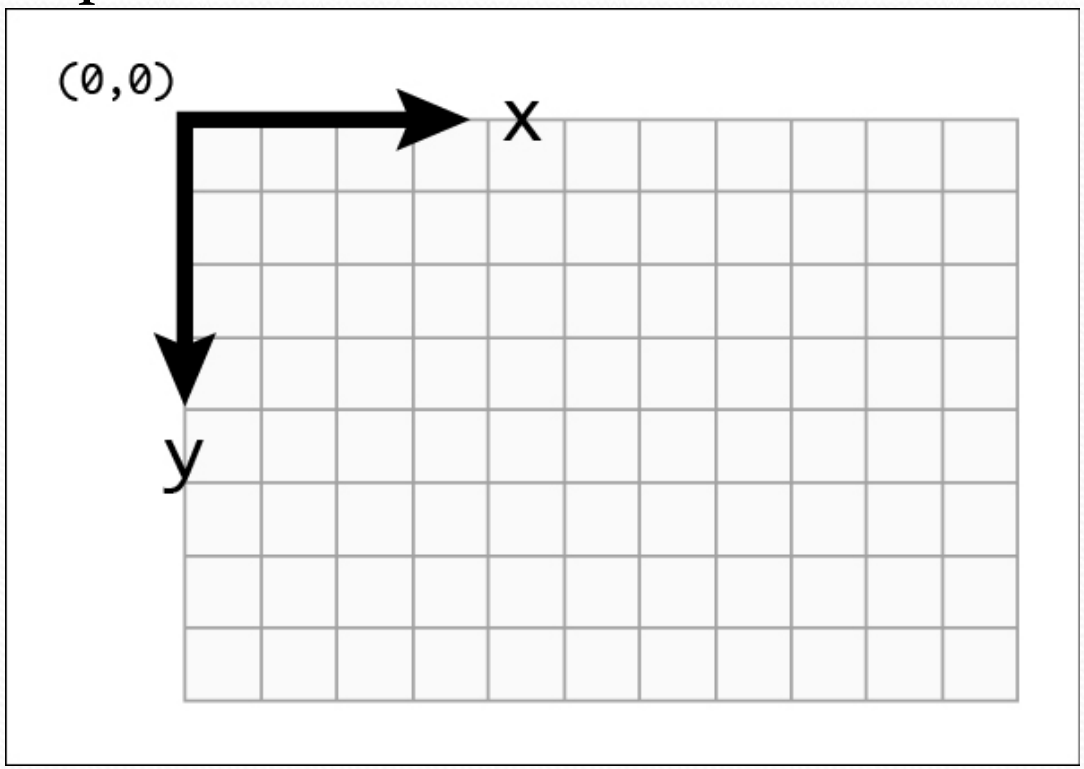
- Graficar la función seno(*x*) para *x* entre 0 y 2 * PI.

Problema con la escala

- Función seno(x):
  - x mínima: 0
  - x máxima: 2 * PI = 6.28…
  - y mínima: -1
  - y máxima: +1
- Canvas en una pantalla de 720 x 1280 pixeles:
  - x mínima: 0
  - x máxima: 720
  - y mínima: 0
  - y máxima: 1280

Problema con la escala

- En el canvas, el origen está en la parte superior izquierda.



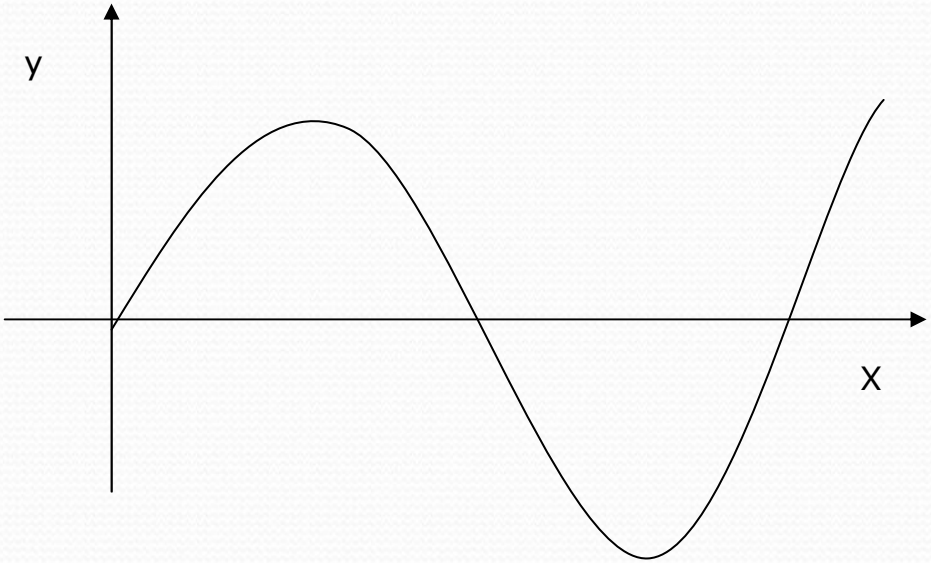
Fuente:
https://cdn.tutsplus.com/net/uploads/legacy/916_canvas1/1.jpg

Proyección window to viewport

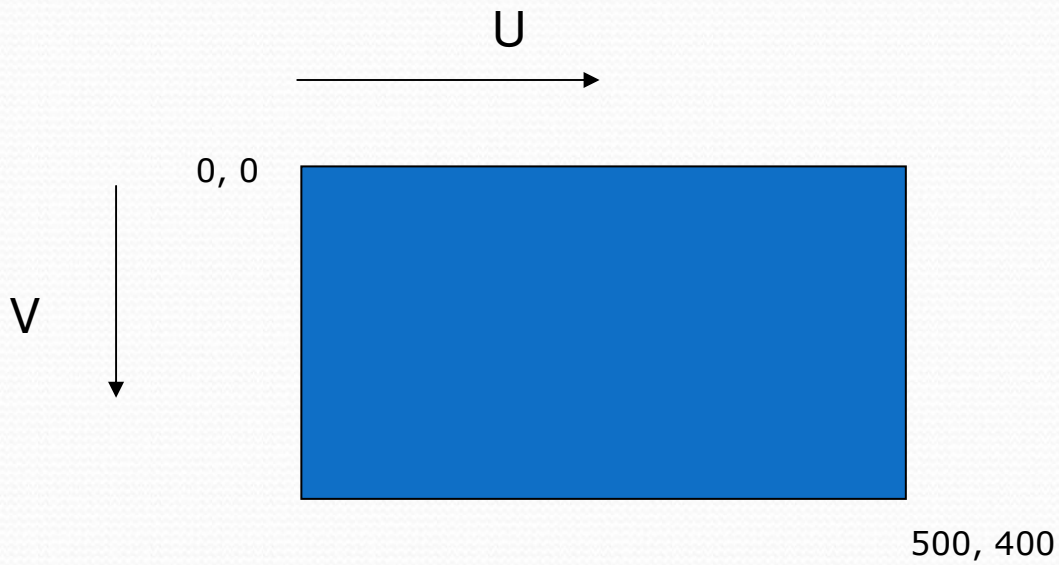
- Ventana (window): región del mundo que se desea pintar en un viewport.
- Viewport: región de la pantalla donde se dibuja, es decir, el canvas.

Ejemplo de ventana

- X: [0...2 * PI]
- Y: [-1...1]

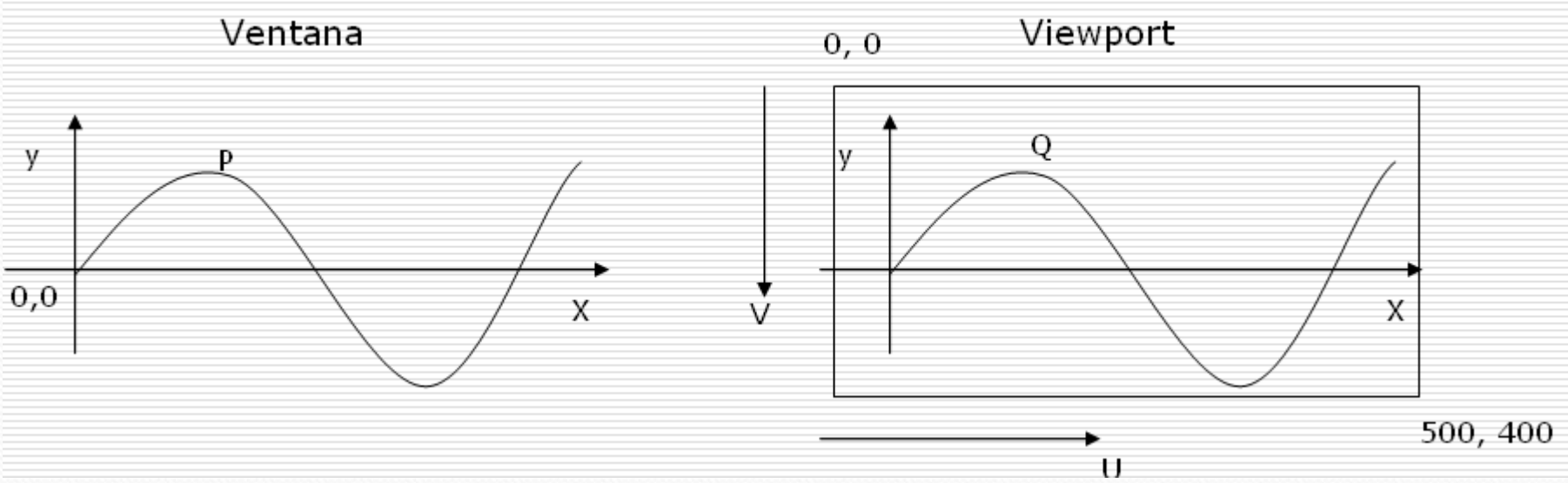


Ejemplo de viewport



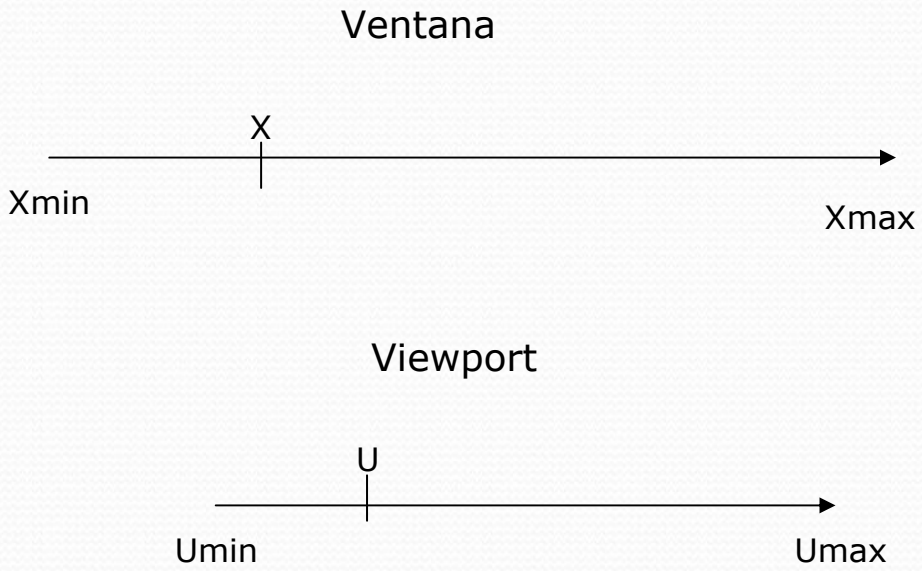
Definición del problema

- Proyección ventana-viewport: dado un punto P en la ventana, encontrar la proyección, Q, de P en el viewport.



Eje horizontal

- Dados X_{min}, X_{max}, X, U_{min} y U_{max}, encontrar U.



Eje horizontal

- Suposición: se respeta la proporción.

$$\frac{X - Xmin}{Xmax - Xmin} = \frac{U - Umin}{Umax - Umin}$$

Eje horizontal

- Despejando U

$$U = \frac{Umax - Umin}{Xmax - Xmin} * (X - Xmin) + Umin$$

- ¡Cuidado con la división entera!
- Definir las variables como reales y redondear U a entero.

Eje vertical

- Para el eje vertical…

- las proporciones son al revés

Eje vertical

$$\frac{Y - Ymin}{Ymax - Ymin} = \frac{Vmax - V}{Vmax - Vmin}$$

- Despejando V

$$V = Vmax - \frac{Vmax - Vmin}{Ymax - Ymin} * (Y - Ymin)$$

Resumen

- Definir la ventana: X_{min}, Y_{min}, $X_{max,}$ Y_{max}.
- Definir el viewport: U_{min}, V_{min}, U_{max}, V_{max}.
- Para cada punto P(X, Y) en la ventana:
  - Aplicar las ecuaciones para encontrar el punto Q(U,V).
  - Graficar Q en el viewport (canvas).

Clase Proyecciones

```
public class Proyecciones {
    private float xMin, yMin, xMax, yMax;   // Ventana
    private float uMin, vMin, uMax, vMax;   // Viewport

    public Proyecciones(float x1, float y1, float x2, float y2,
                        float u1, float v1, float u2, float v2)
    {
        xMin = x1; yMin = y1; xMax = x2; yMax = y2;
        uMin = u1; vMin = v1; uMax = u2; vMax = v2;
    }

    /*
        Recibe p en coordenadas de la ventana
        regresa su proyeccion en coordenadas del viewport
     */
    public PointF ventana2Viewport(PointF p)
    {
        float u = ((uMax - uMin) / (xMax - xMin)) * (p.x - xMin) + uMin;
        float v = vMax - ((vMax - vMin) / (yMax - yMin)) * (p.y - yMin);

        return new PointF(u, v);
    }
}
```

Clase PintaSeno

```
public class PintaSeno extends View {
    private ArrayList<PointF> datos;
    private Paint paint;

    public PintaSeno(Context context)
    {
        super(context);
        datos = new ArrayList<>();
        for (float x = 0; x < 2 * Math.PI; x += 0.05) {
            datos.add(new PointF(x, (float) Math.sin(x)));
        }
        paint = new Paint();
        paint.setColor(Color.RED);
        paint.setStrokeWidth(2);
        paint.setAntiAlias(true);
    }
```

Clase PintaSeno

```
    public void onDraw(Canvas canvas)
    {
        Proyecciones proy = new Proyecciones(0, -1.2f, 7, 1.2f,
                                             0, 0, getWidth(), getHeight());
        for (int i = 0; i < datos.size() - 1; i++) {
            PointF p1 = datos.get(i);
            PointF p2 = datos.get(i + 1);
            PointF q1 = proy.ventana2Viewport(p1);
            PointF q2 = proy.ventana2Viewport(p2);
            canvas.drawLine(q1.x, q1.y, q2.x, q2.y, paint);
        }
    }
}
```

Actividad principal

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        PintaSeno p = new PintaSeno(this);
        setContentView(p);
    }
}
```