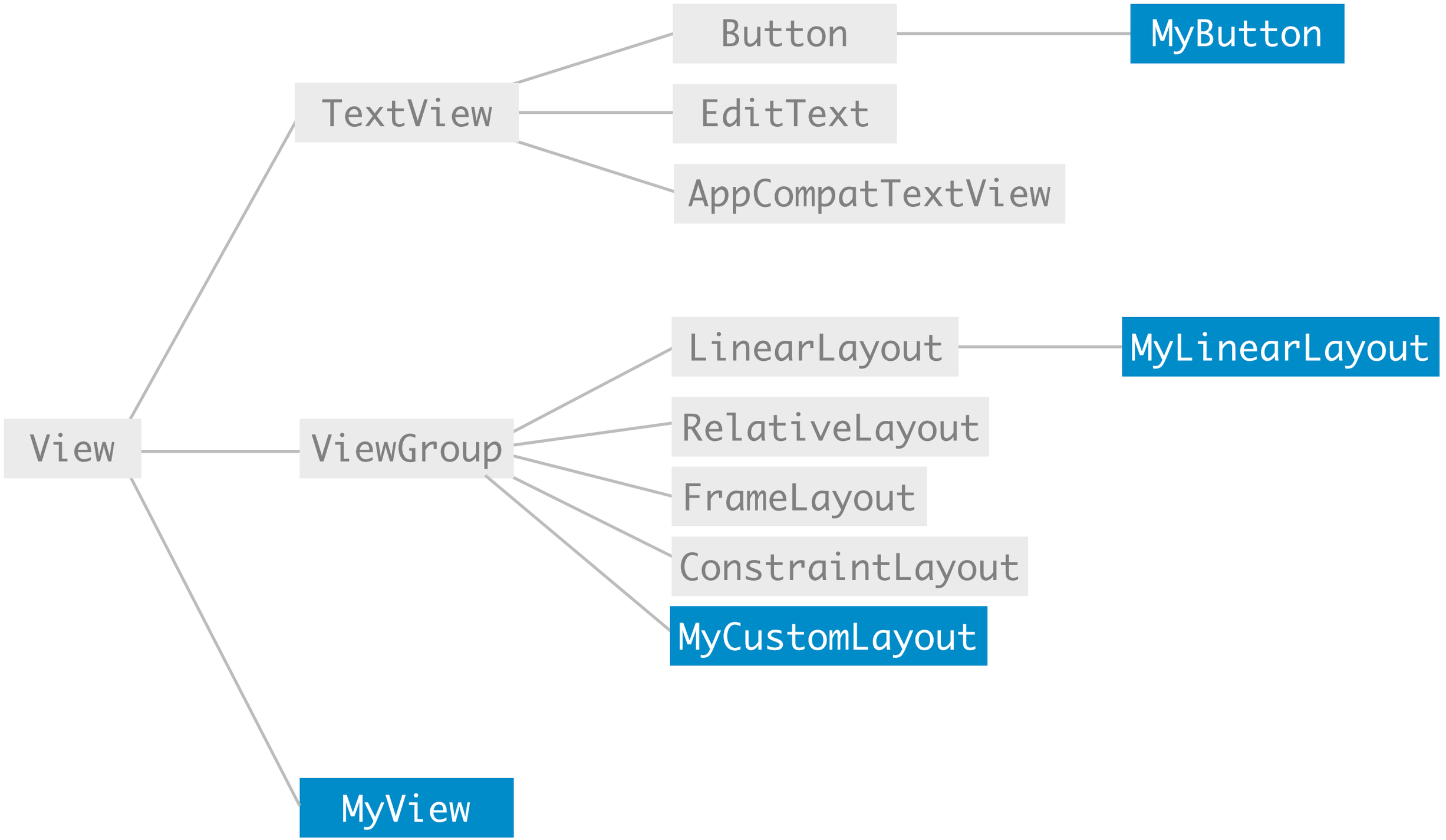
Custom Views

(… & ViewGroups?)

Bases


View
TextView
Button
MyButton
EditText
AppCompatTextView
ViewGroup
LinearLayout
MyLinearLayout
RelativeLayout
FrameLayout
ConstraintLayout
MyCustomLayout
MyView

Bases

	Views	ViewGroups
Extensiones	`MyButton`	`MyLinearLayout`
Extensiones Base	`MyView`	`MyCustomLayout`

Custom Views

- Apariencia / comportamiento personalizado
- Es reusable
- Rendimiento

Custom Views

- Toma tiempo realizarlos
- Complicadas de implementar
- Perdemos las funcionalidades básicas de los componentes nativos

Custom Views

- onDraw

al heredar de "View"

- onMeasure

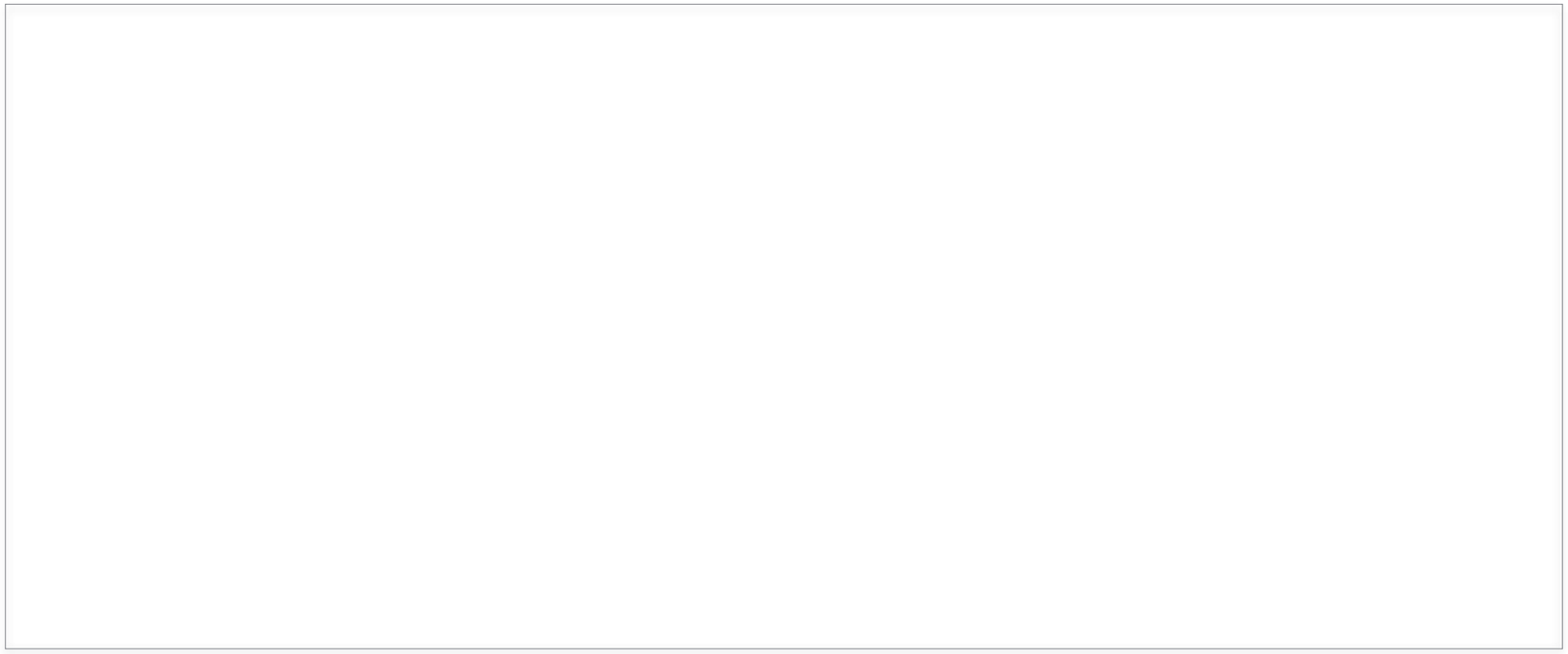
- onLayout

al heredar de "ViewGroup"

onDraw

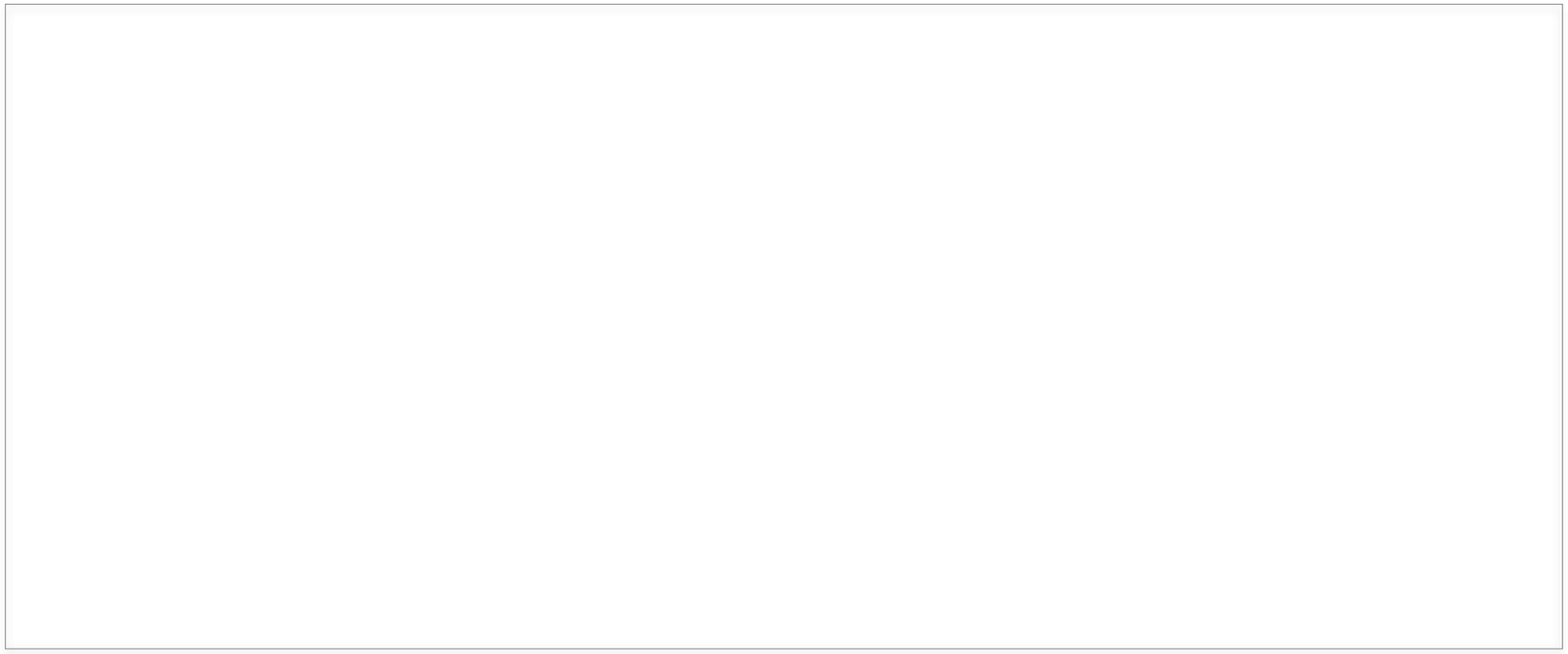
```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
}
```

onDraw



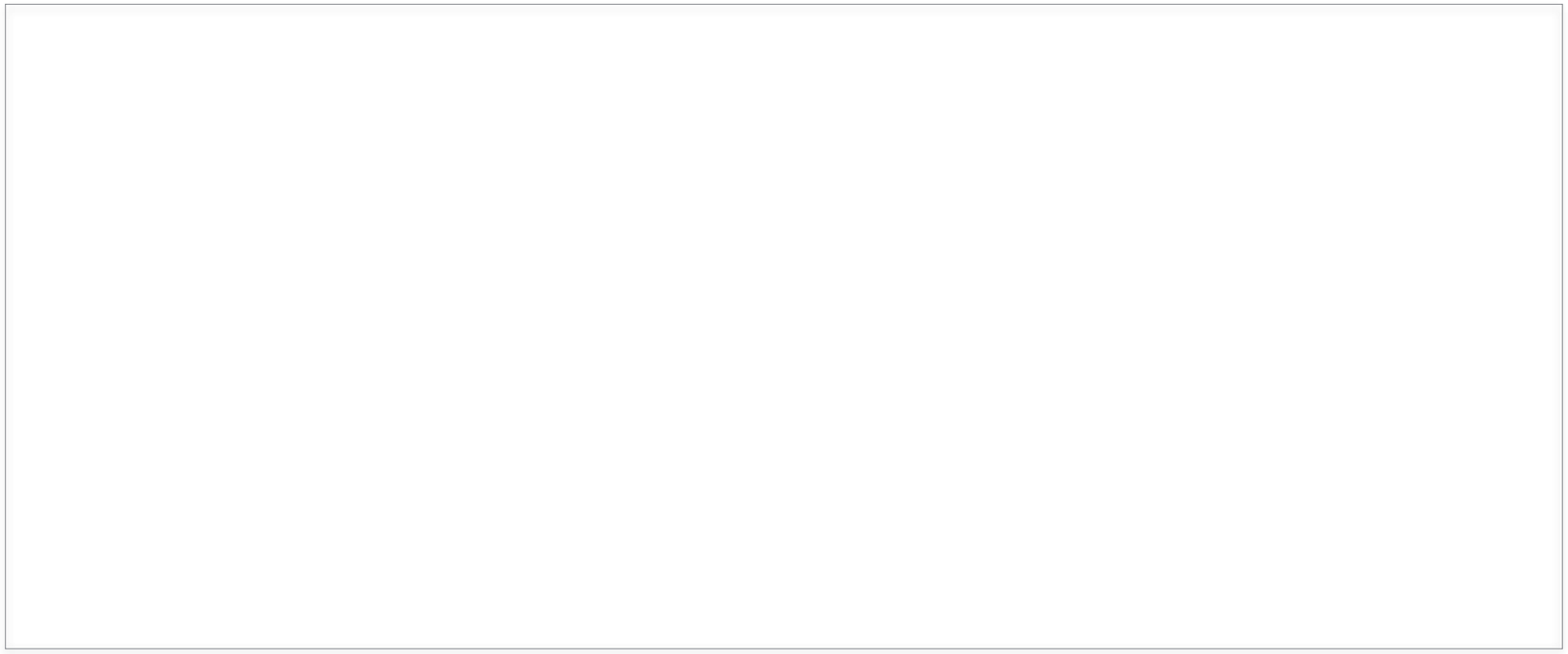
```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
}
```

onDraw



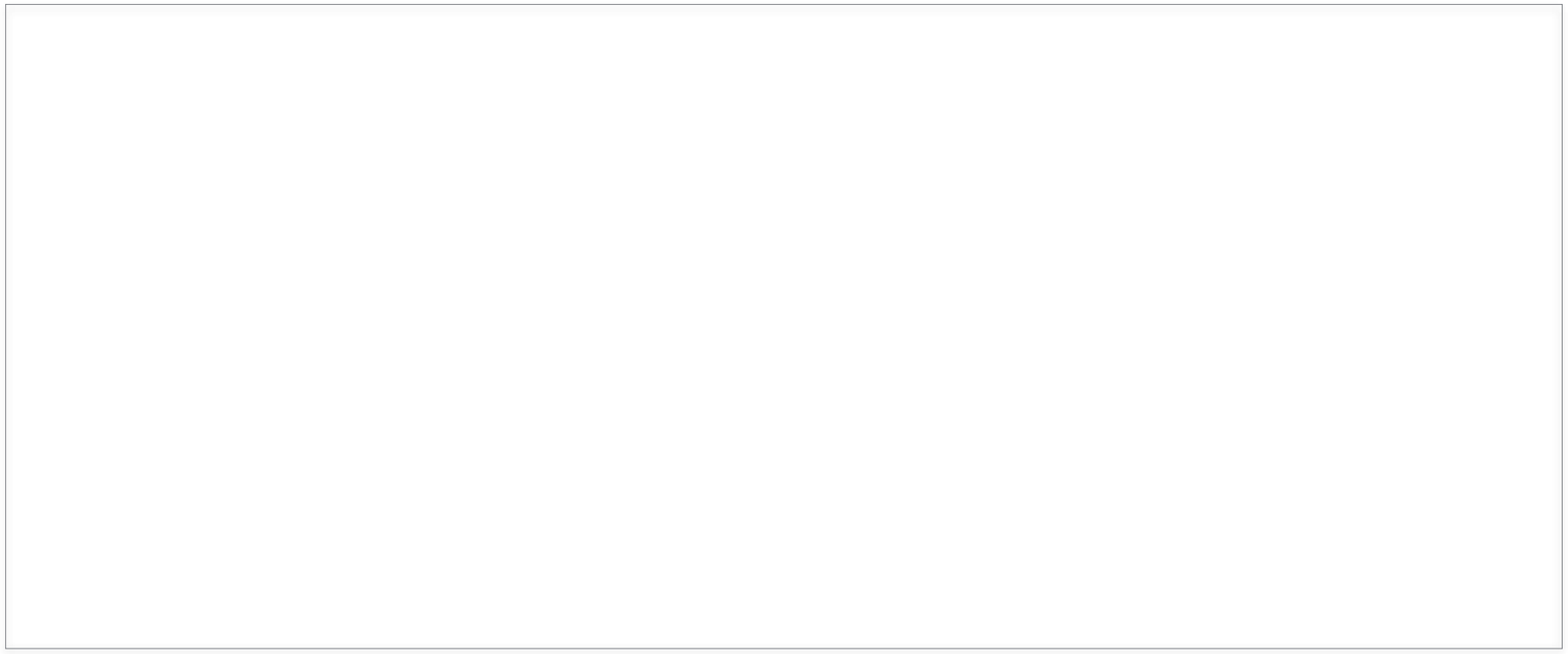
```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
}
```

onDraw



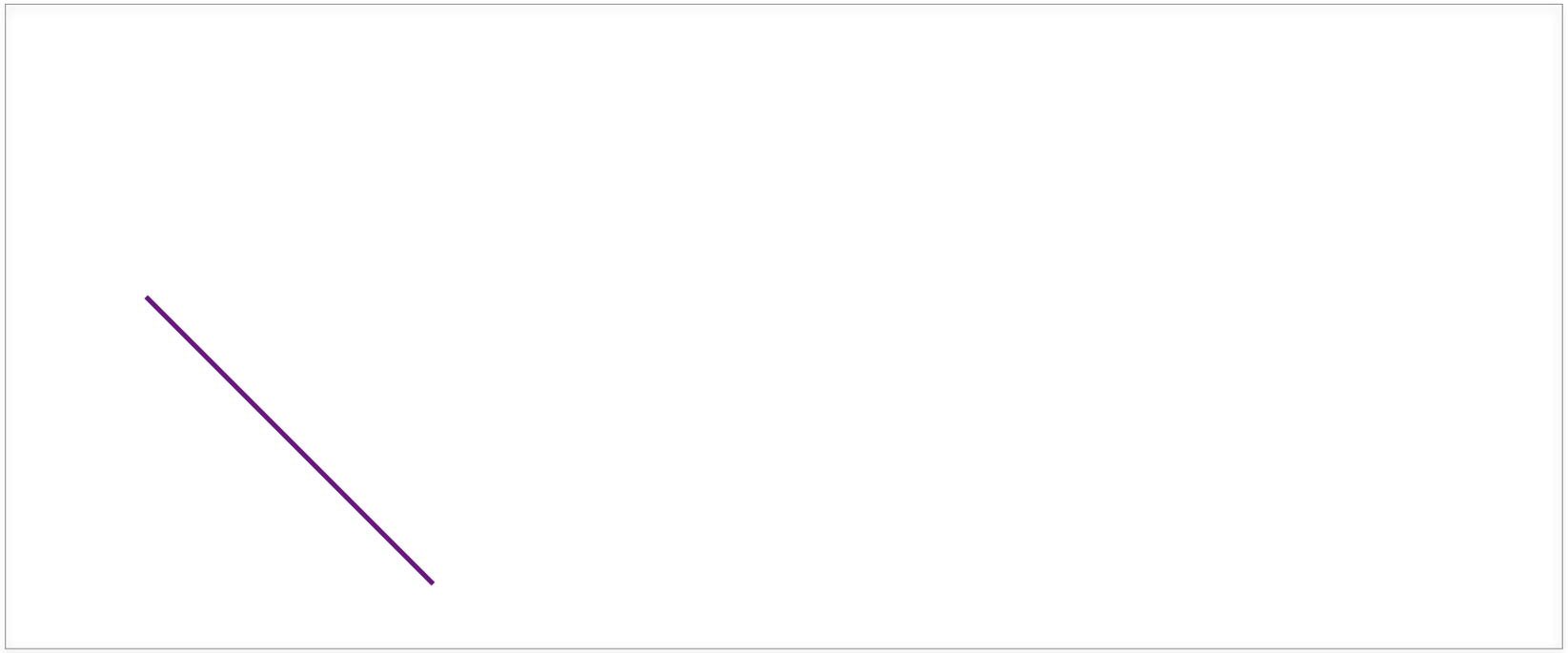
```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
}
```

onDraw



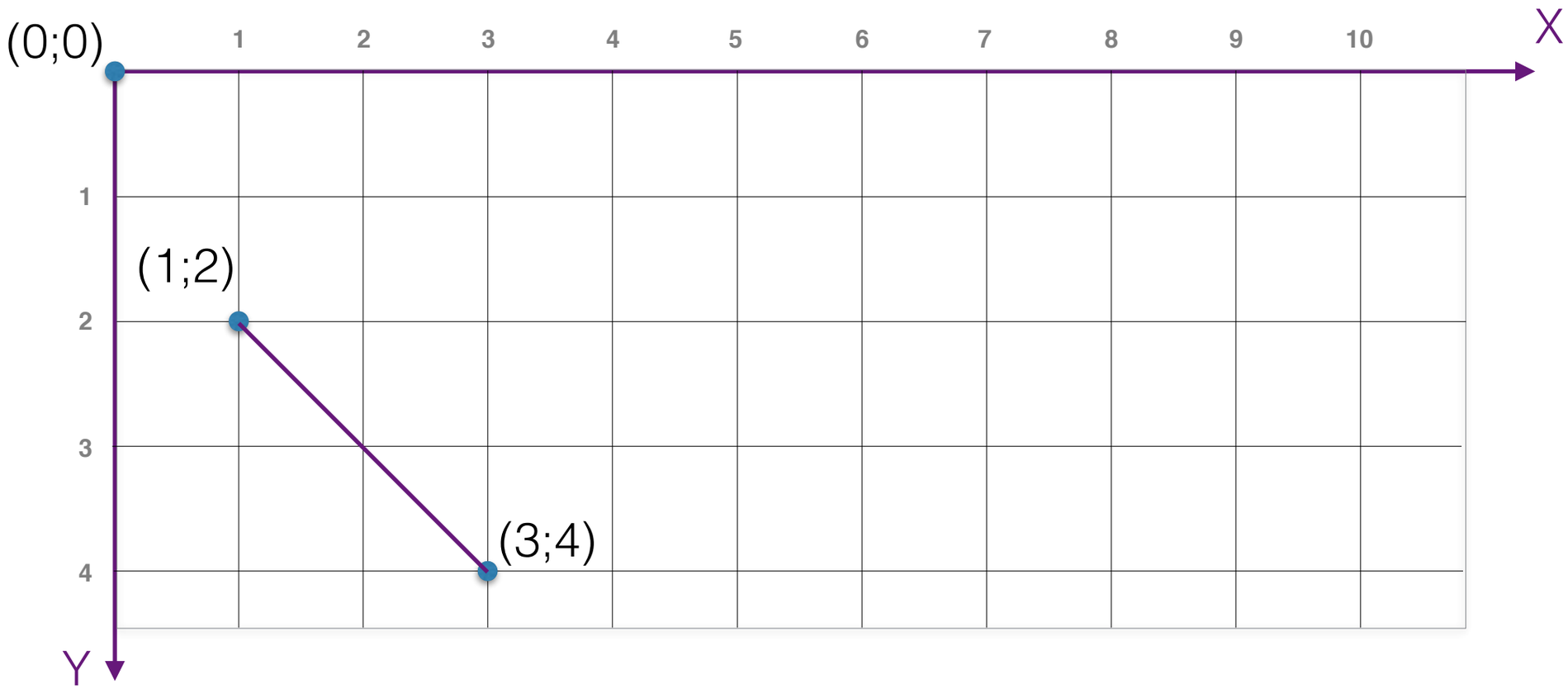
```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
}
private Paint createPaintFromResource(@ColorRes int color){
    Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);
    paint.setColor(ContextCompat.getColor(getContext(),color));
   return paint;
}
```

onDraw



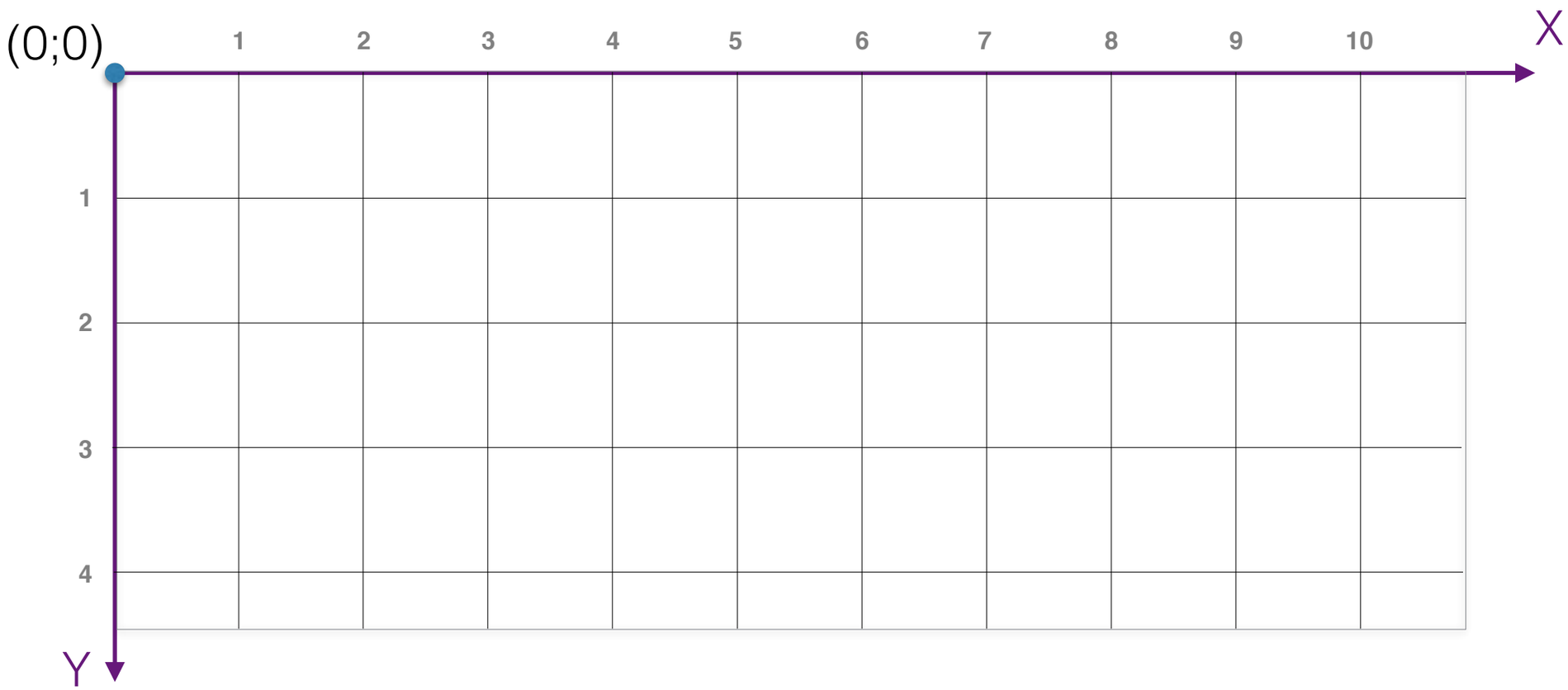
```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
    canvas.drawLine(1,2,3,4,color);
}
```

onDraw



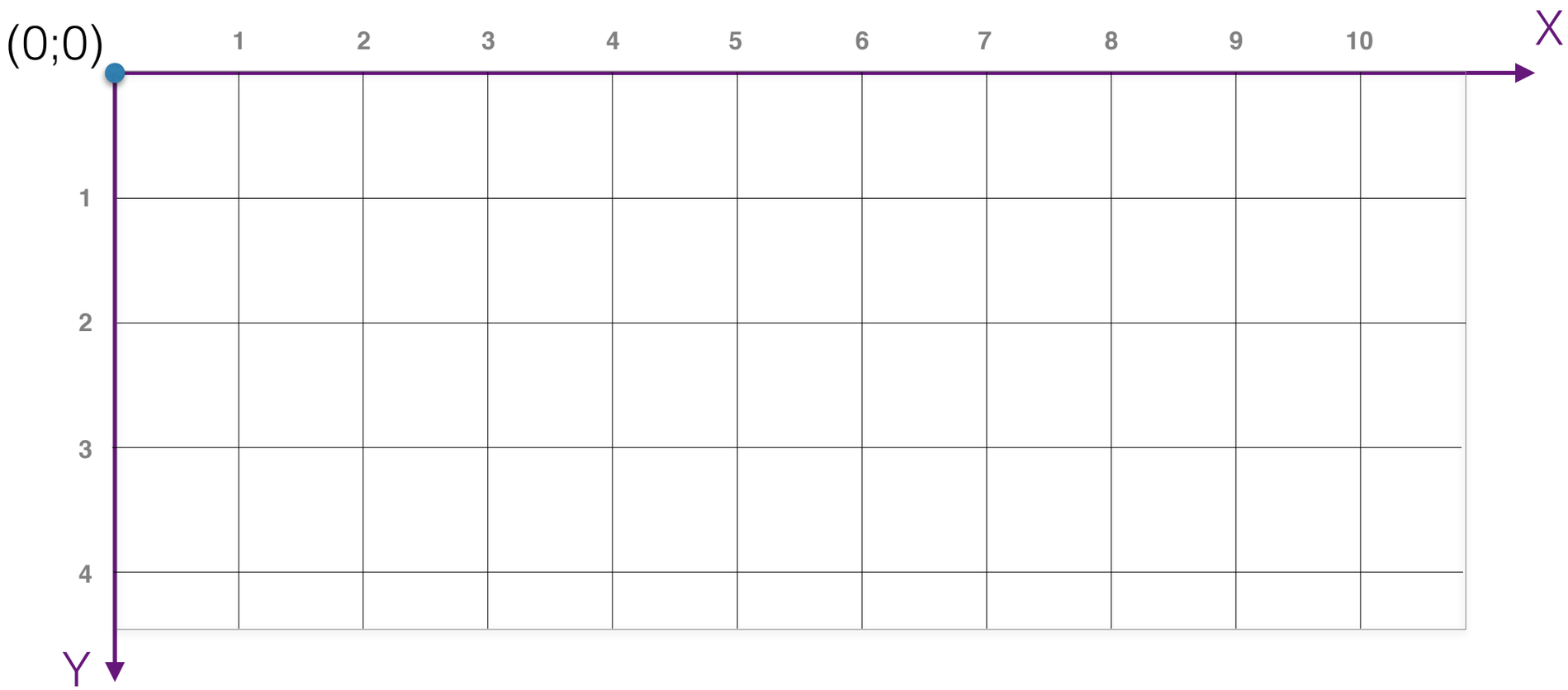

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
    canvas.drawLine(startX=1,startY=2,stopX=3,stopX=4,paint);
}
```

onDraw



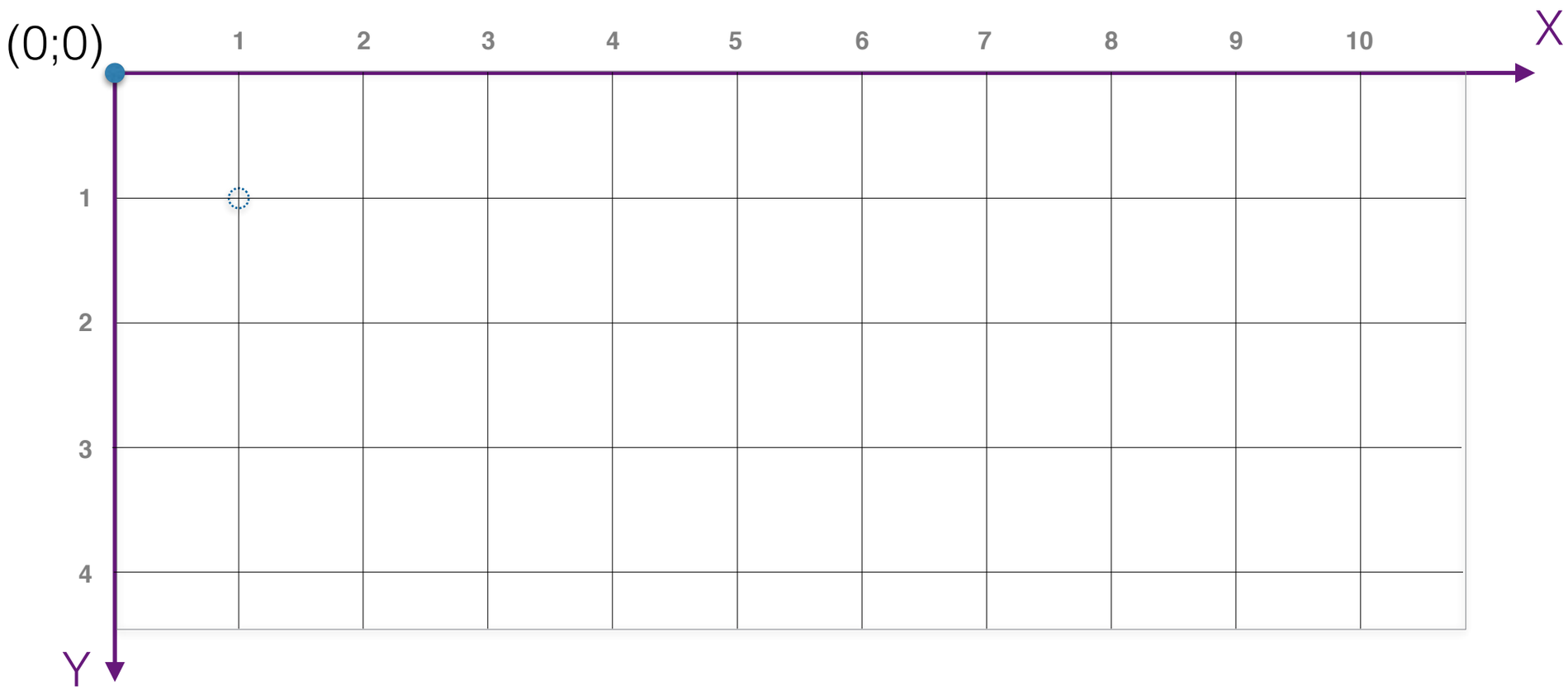

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);

}
```

onDraw



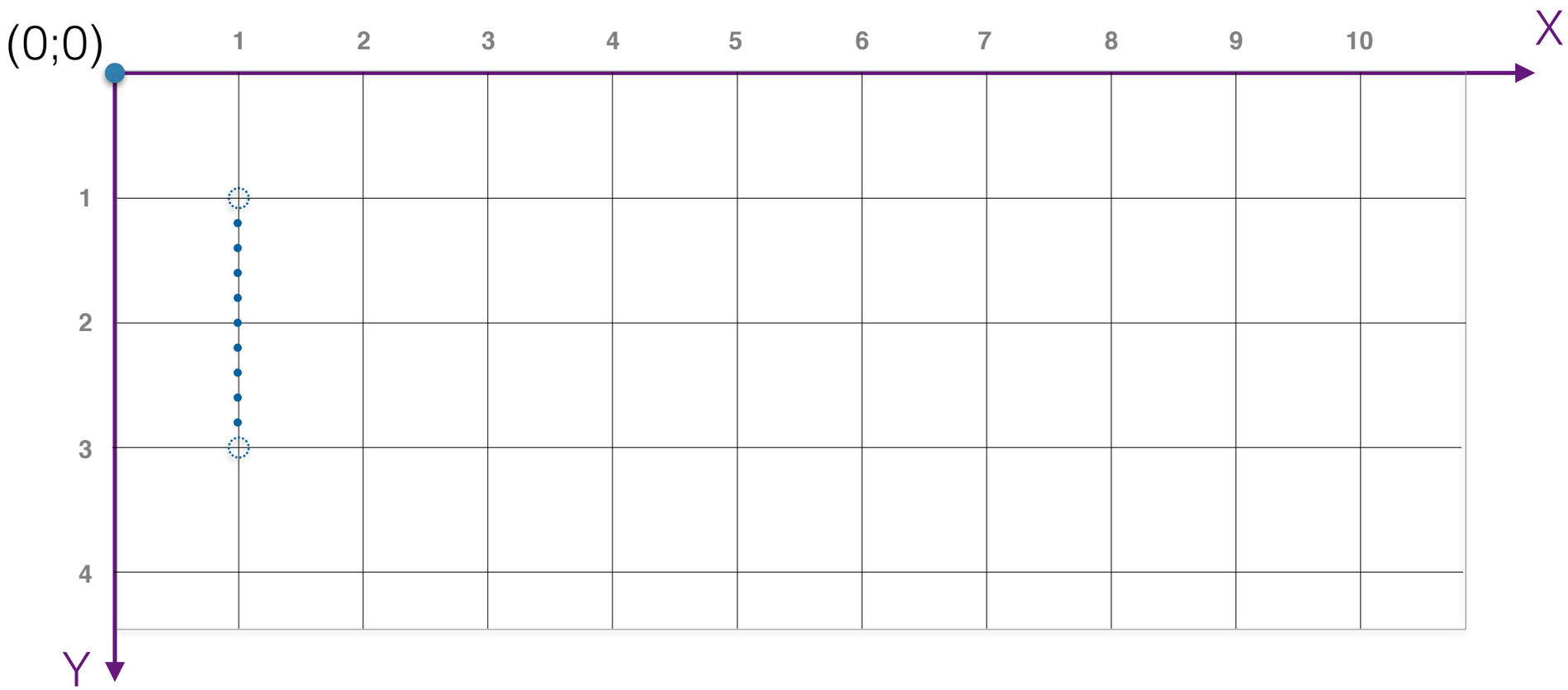

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
    Path path = new Path();



}
```

onDraw



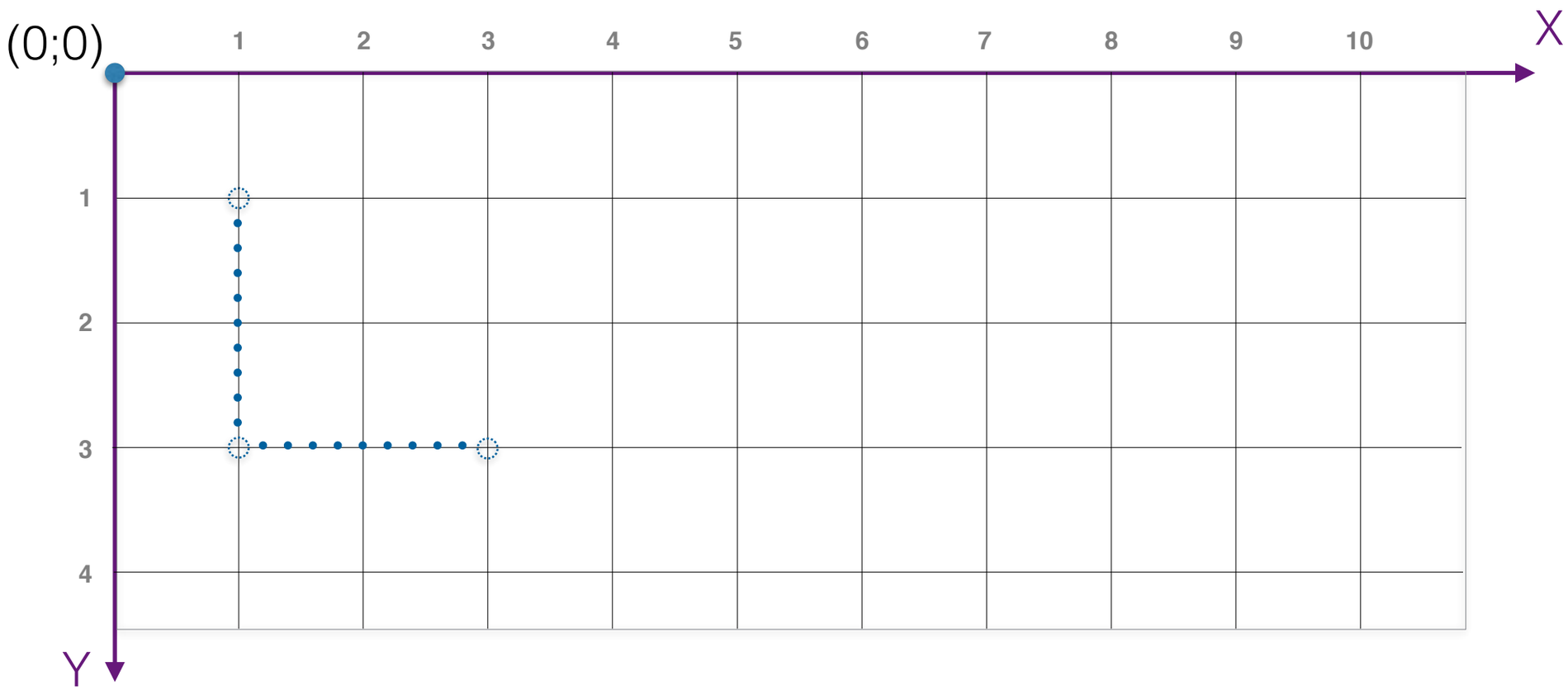

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
    Path path = new Path();
    path.moveTo(1,1);



}
```

onDraw



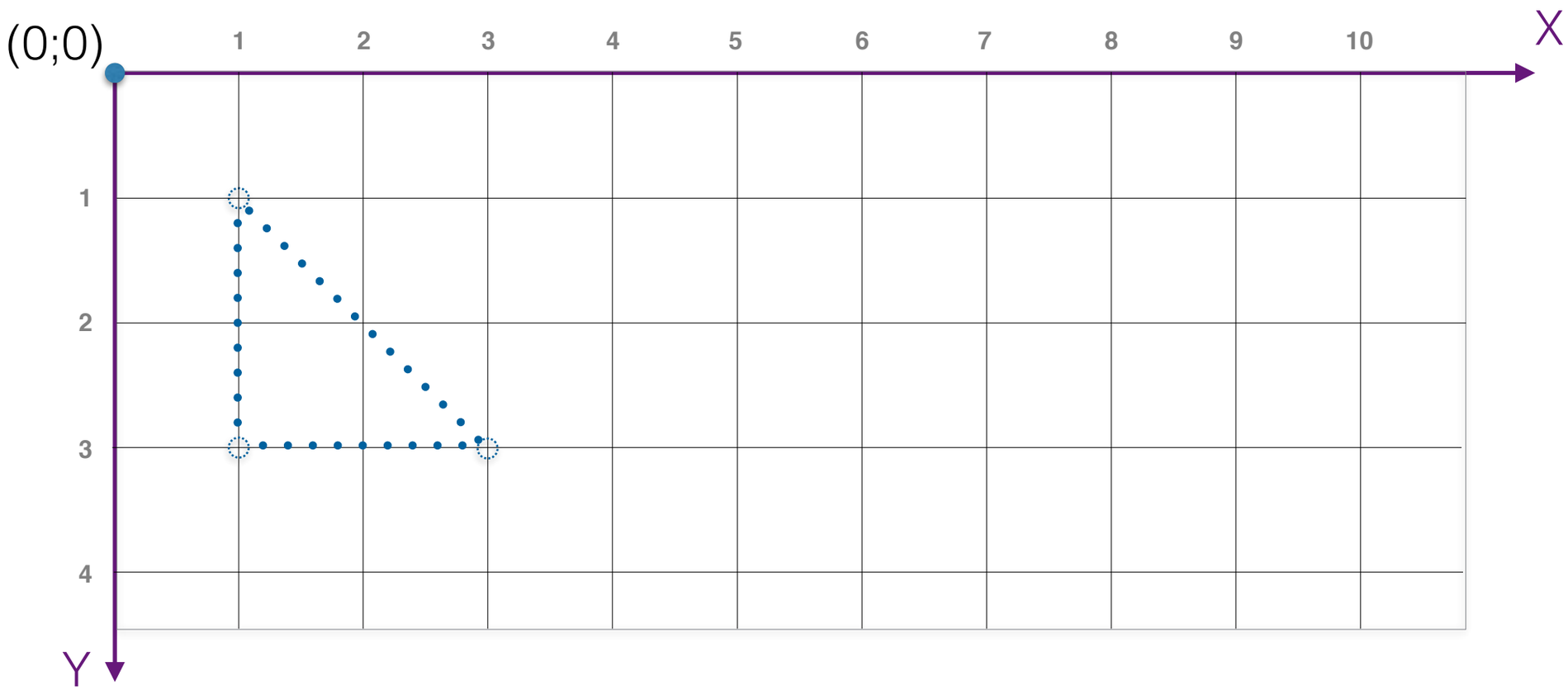

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
    Path path = new Path();
    path.moveTo(1,1); path.moveTo(1,3);



}
```

onDraw



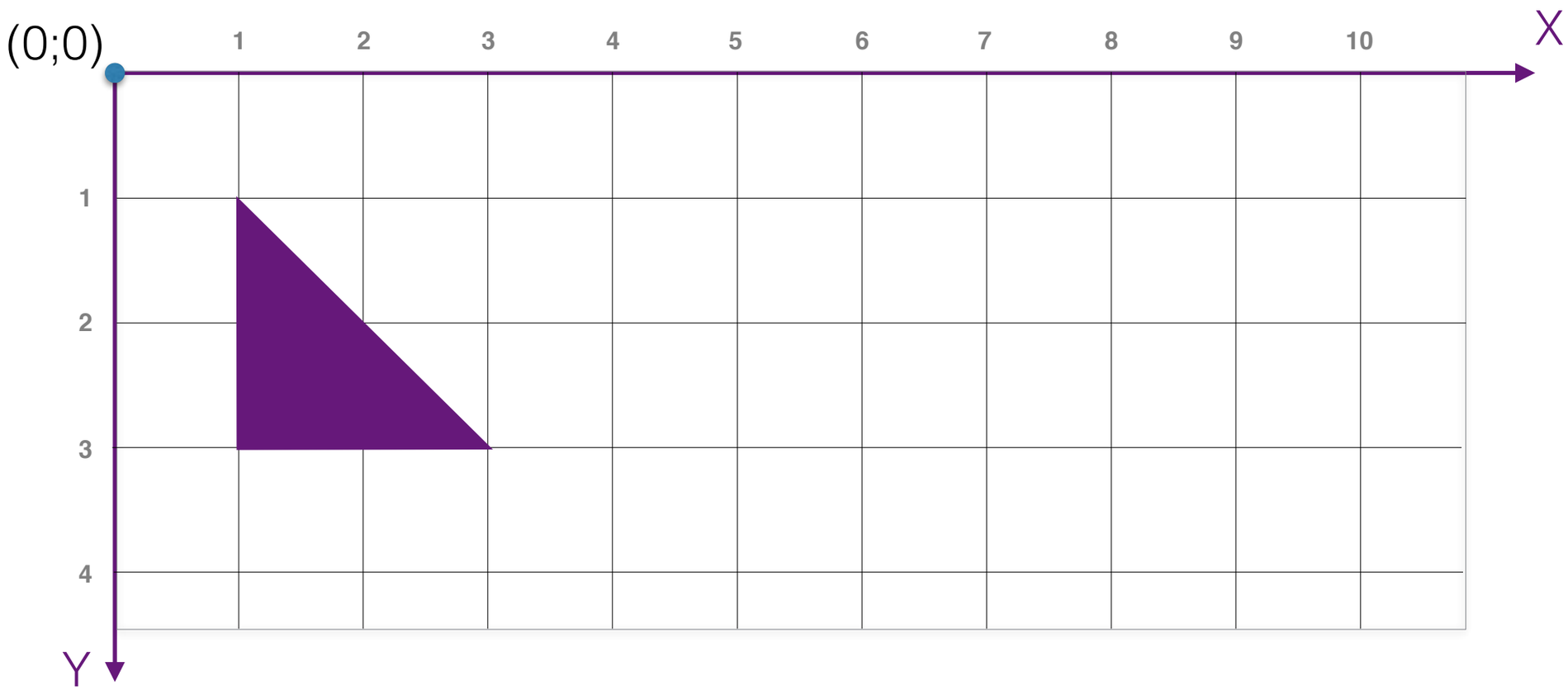

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
    Path path = new Path();
    path.moveTo(1,1); path.moveTo(1,3); path.moveTo(3,3);



}
```

onDraw



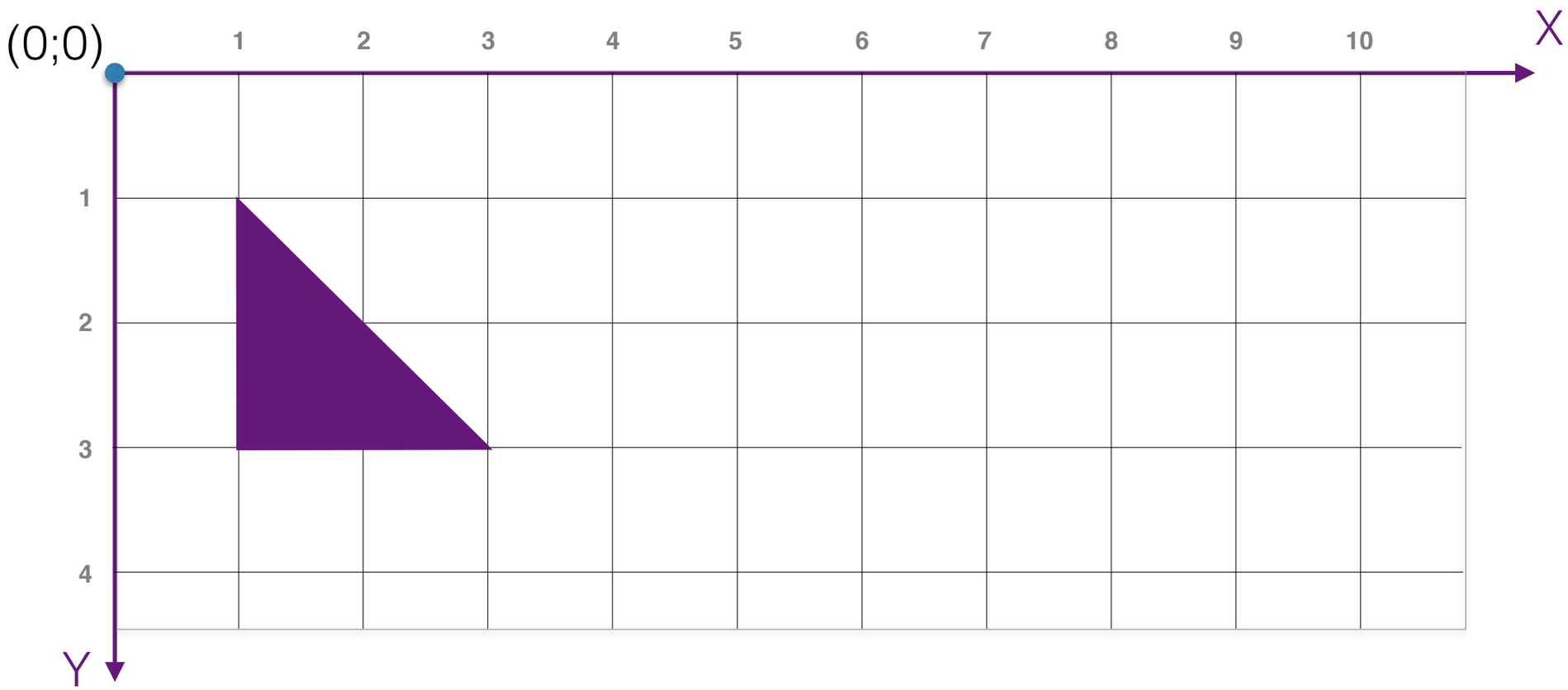

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
    Path path = new Path();
    path.moveTo(1,1); path.moveTo(1,3); path.moveTo(3,3);
    path.moveTo(1,1);


}
```

onDraw



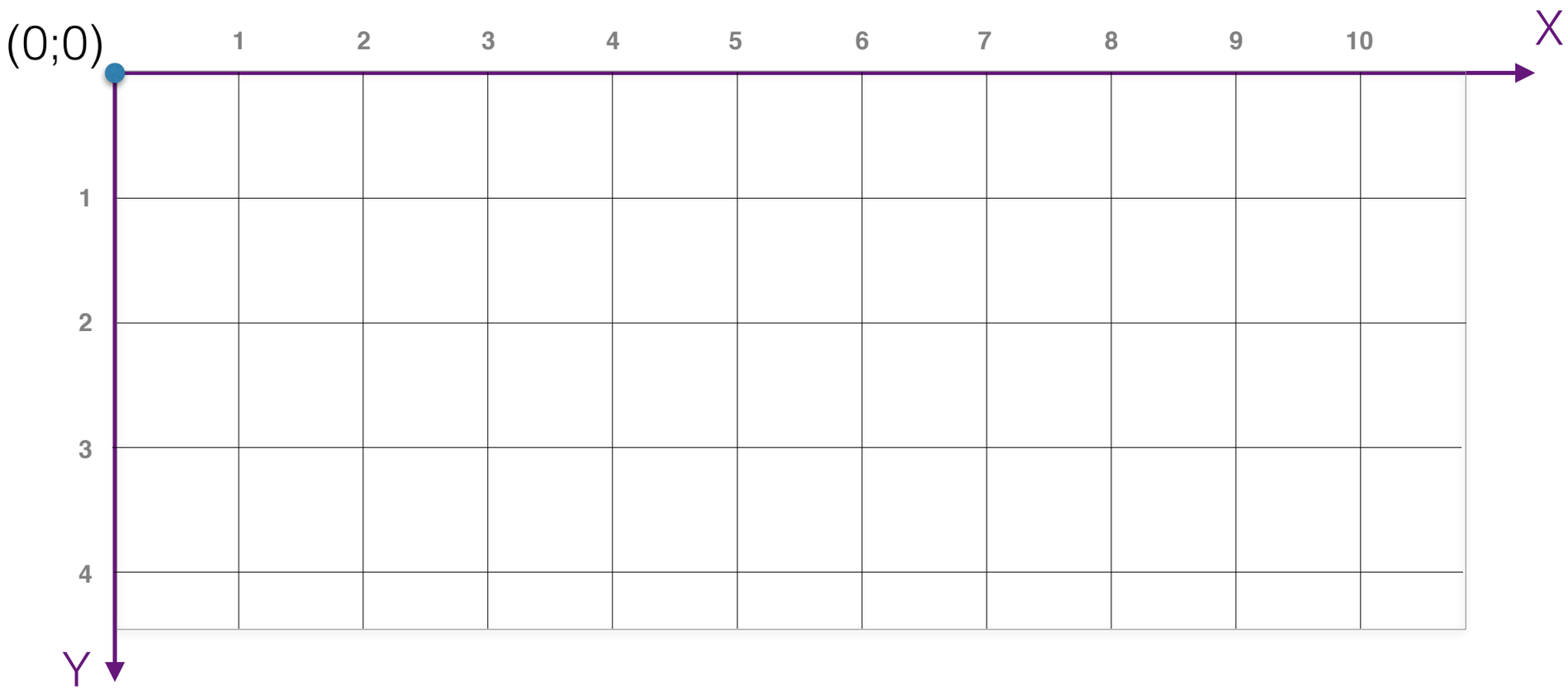

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
    Path path = new Path();
    path.moveTo(1,1); path.moveTo(1,3); path.moveTo(3,3);
    path.moveTo(1,1);
    canvas.drawPath(path,paint);

}
```

onDraw



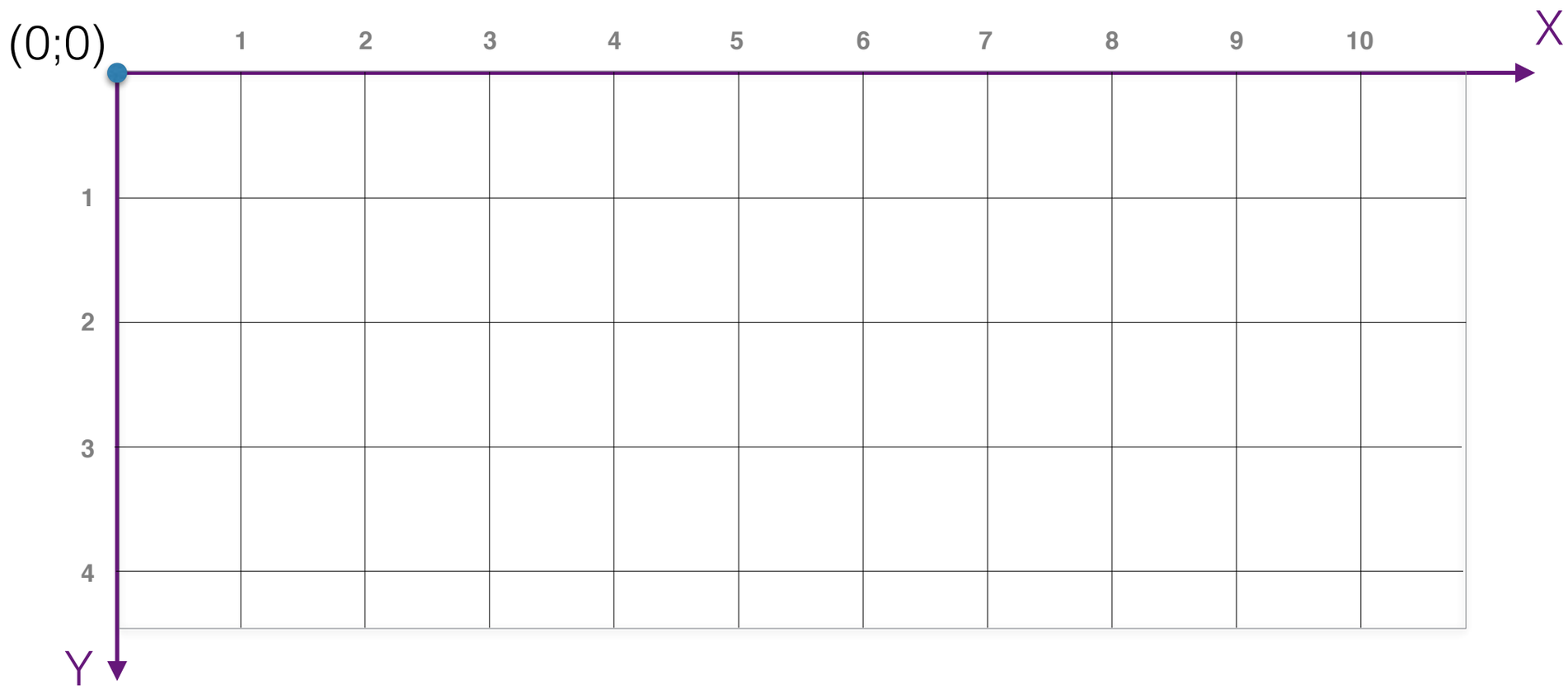

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
    Path path = new Path();
    path.moveTo(1,1); path.moveTo(1,3); path.moveTo(3,3);
    path.moveTo(1,1);
    paint.setStyle(Paint.Style.STROKE);
    canvas.drawPath(path,paint);
}
```

onDraw



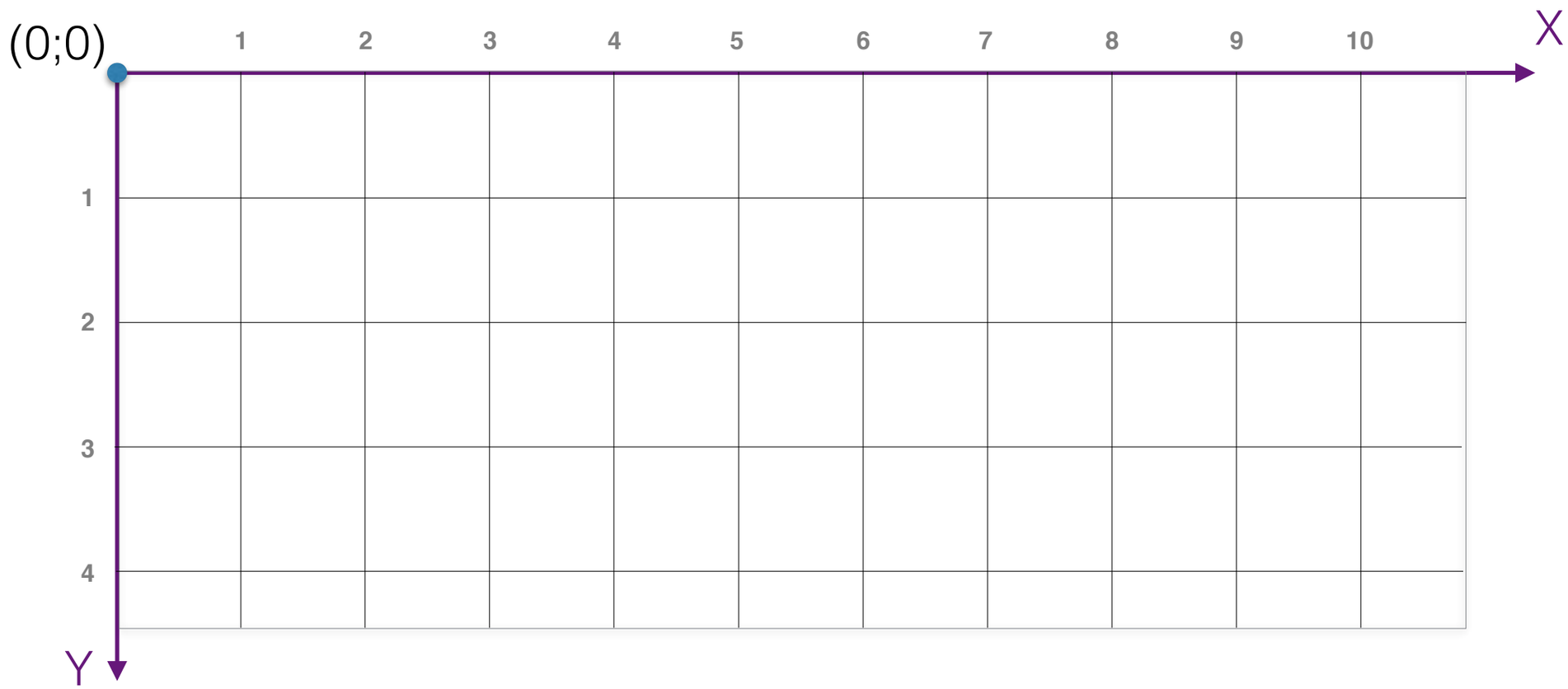

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);

}
```

onDraw



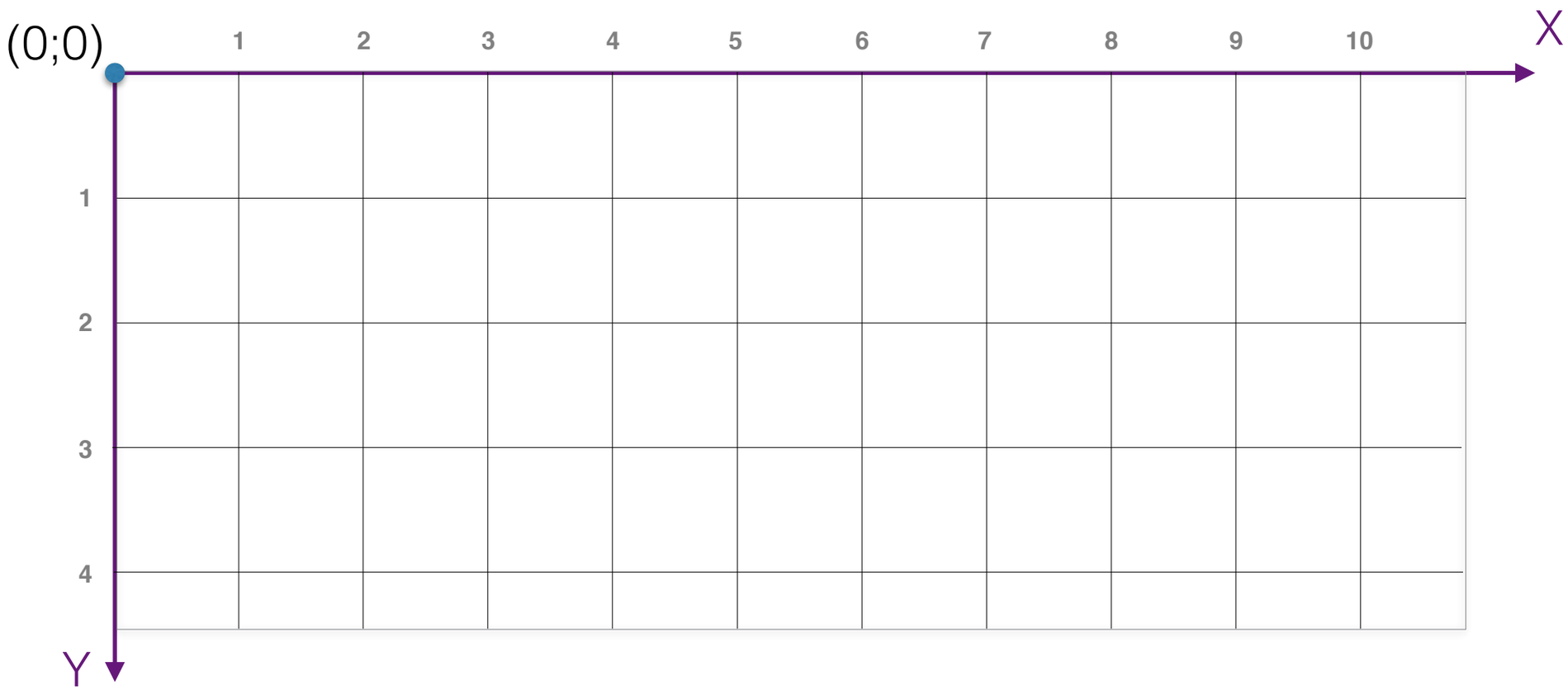

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
    paint.setTypeface(createTypeface("myfont.ttf"));
    canvas.drawText("Hola",x=1,y=2,paint);
}
```

onDraw



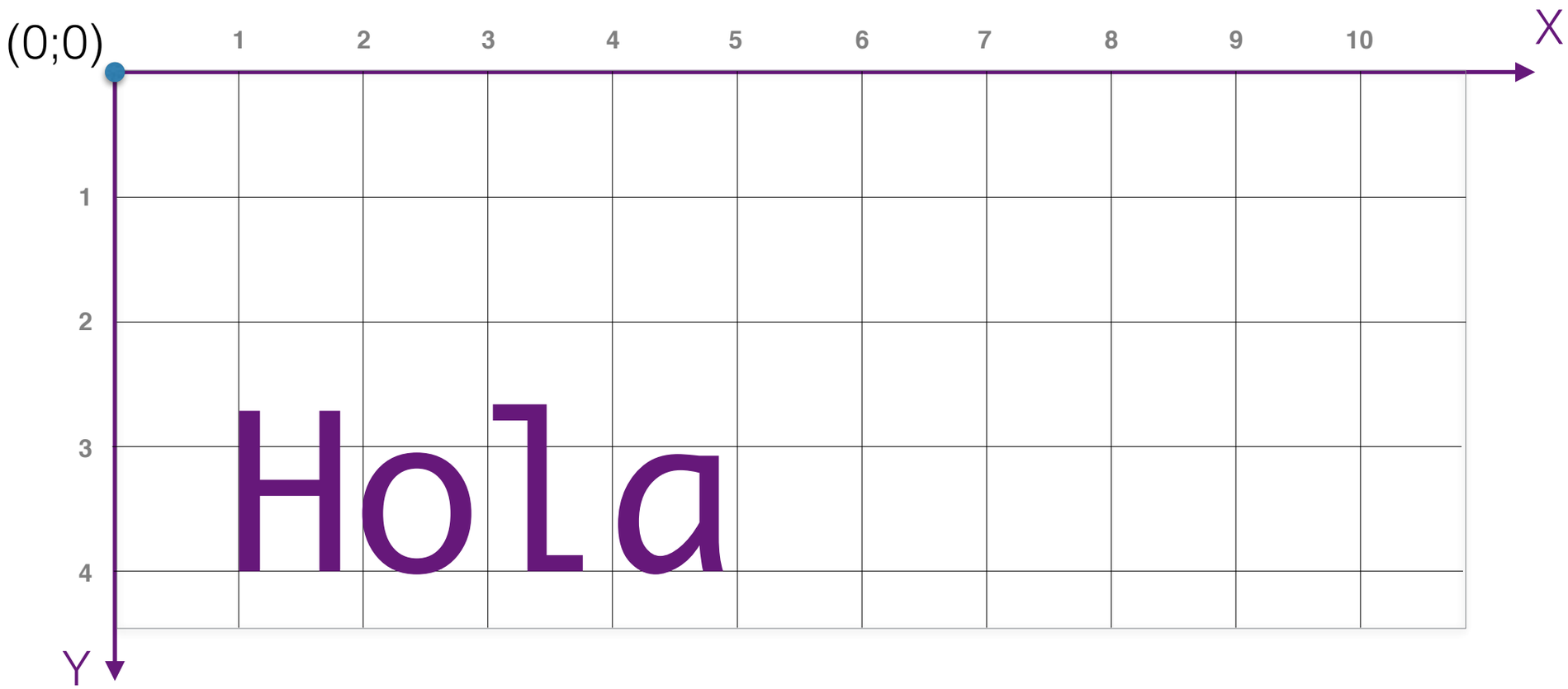

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
    paint.setTypeface(createTypeface("myfont.ttf"));
    canvas.drawText("Hola",x=1,y=2,paint);
}
```

onDraw



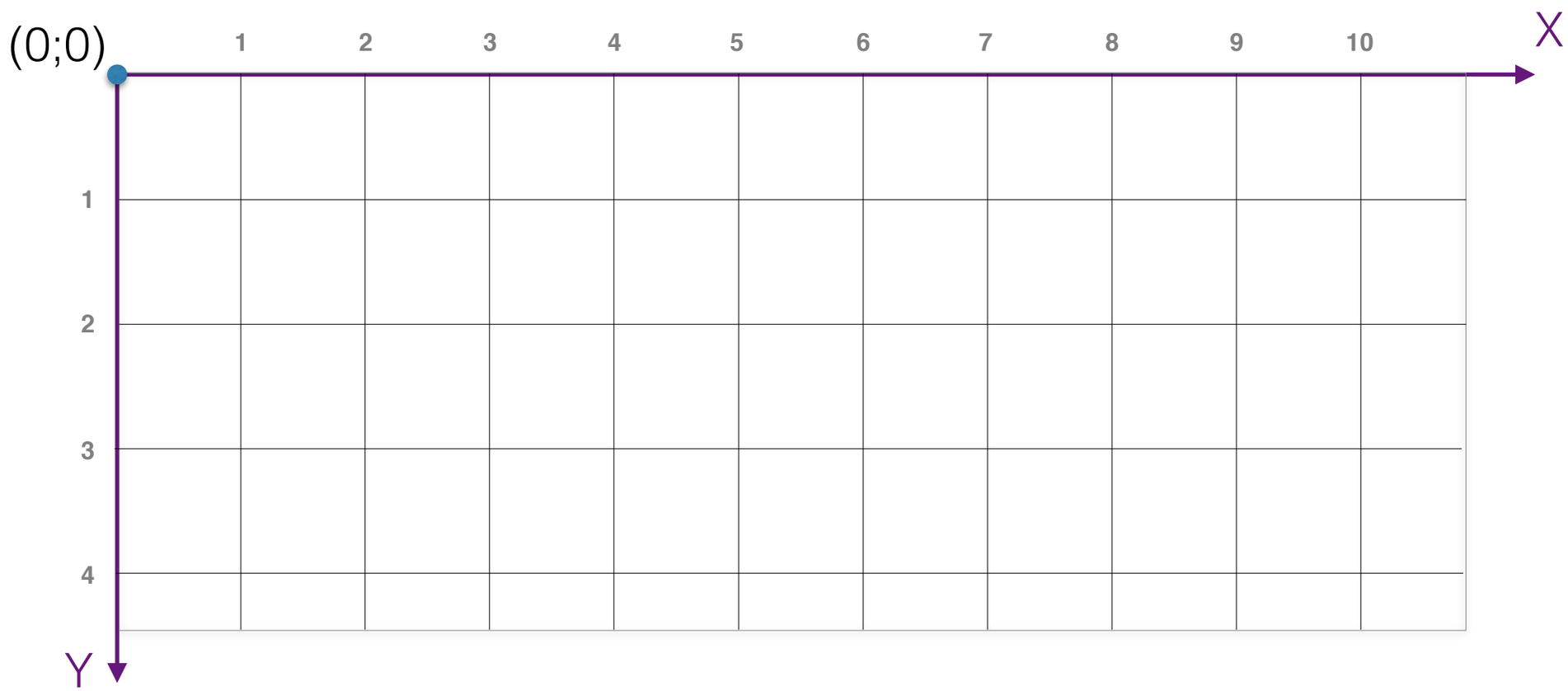

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
    paint.setTypeface(createTypeface("myfont.ttf"));
    canvas.drawText("Hola",x=1,y=2,paint);
}
private Typeface createTypeface(String s) {
    return Typeface.createFromAsset(getContext().getAssets(),s);
}
```

onDraw




```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
    paint.setTypeface(createTypeface("myfont.ttf"));
    canvas.drawText("Hola", x=1, y=4, paint);
}
```

onDraw




```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = createPaintFromResource(R.color.colorPrimary);
    canvas.drawPoint(...);
    canvas.drawArc(...);
    canvas.drawCircle(...);
    canvas.drawRect(...);
    ...
}
```

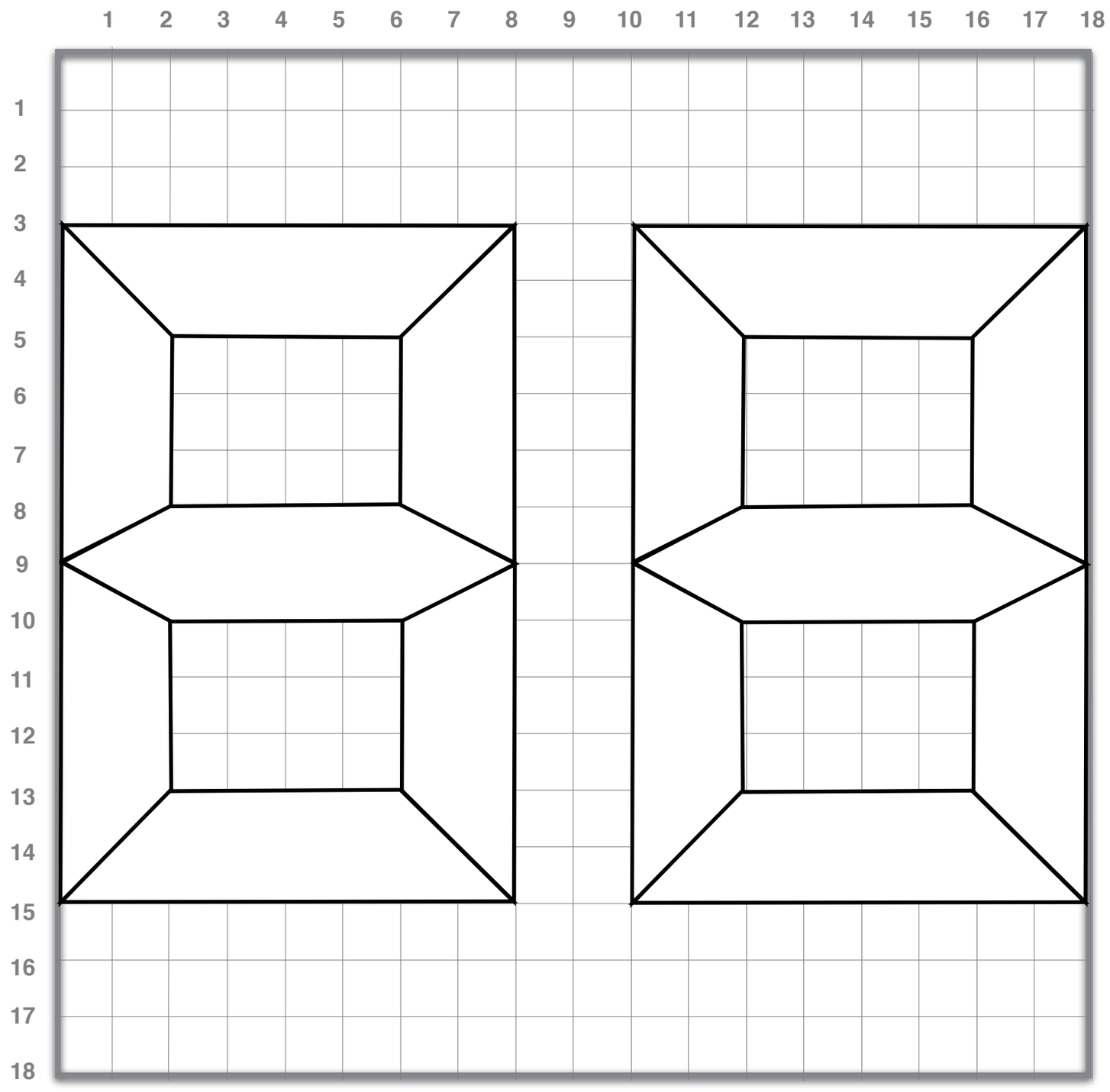
El reto

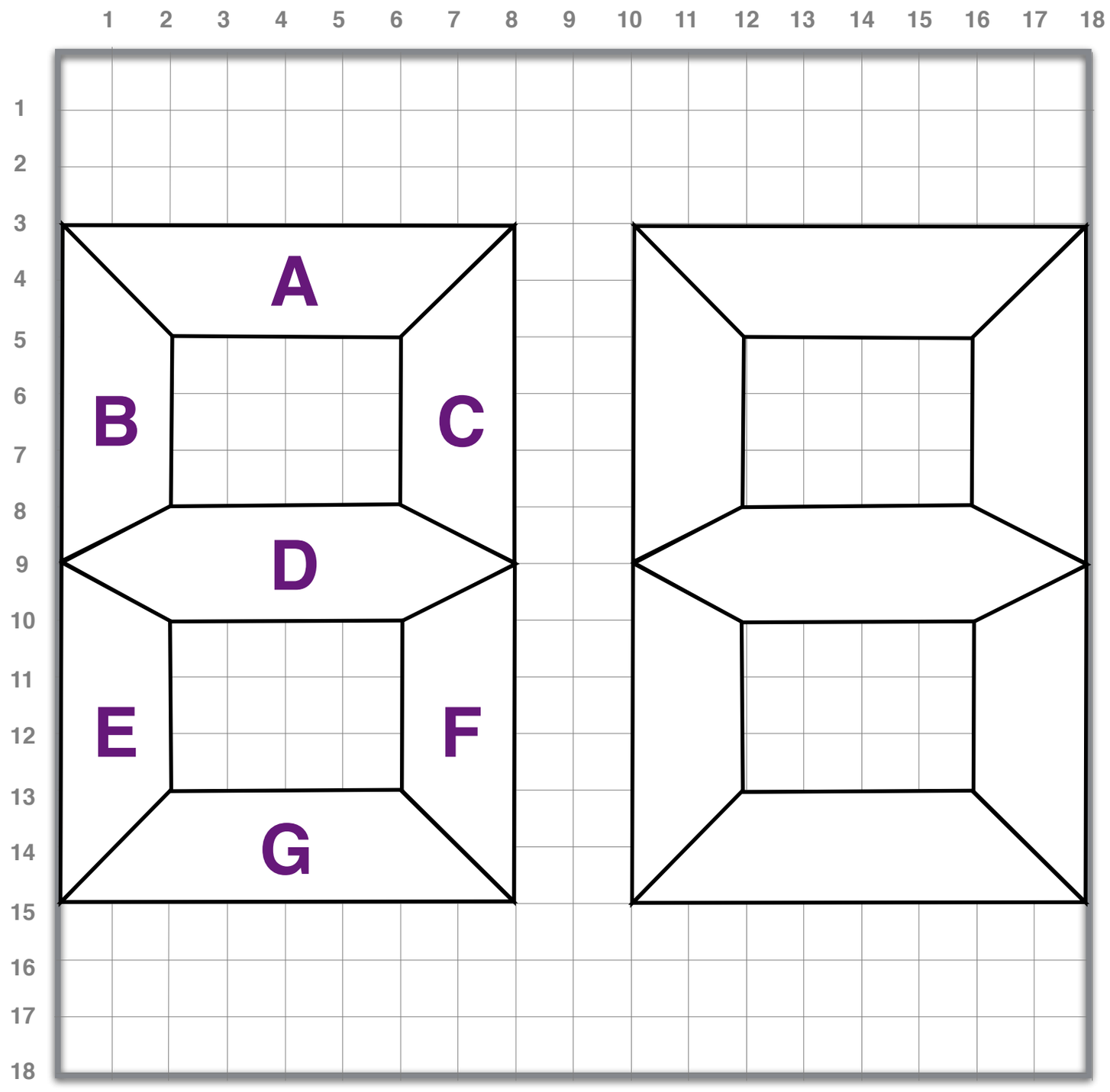

TS SPORTS
88:88
88
SAMSUNG

Requerimientos

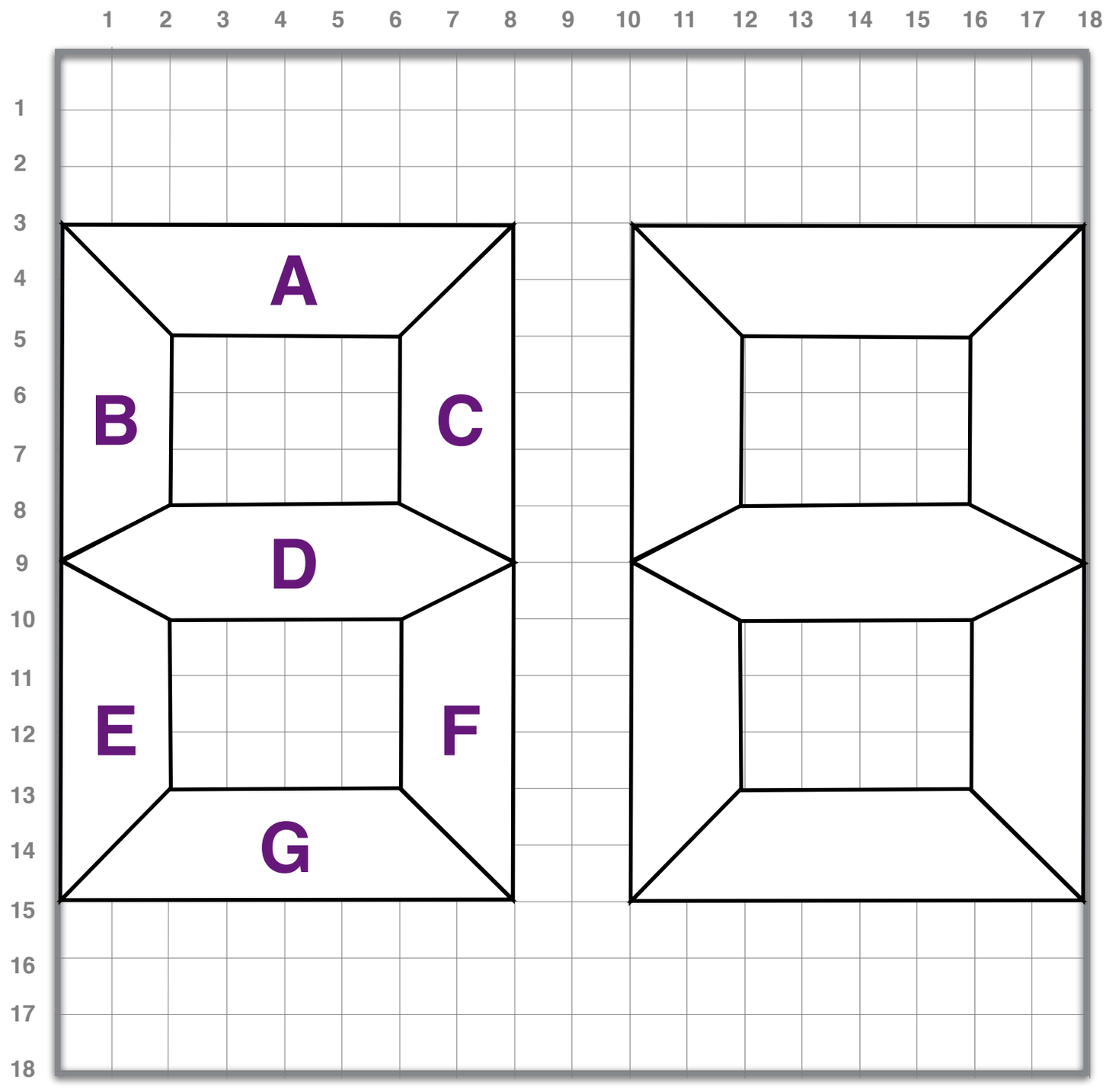
- Cada segundo baja el contador en 1
- Al llegar a 0, sonará una alarma
- El contador no puede bajar de 0 ni ser mayor a 24
- Se puede iniciar, detener y resetear el reloj
- Por defecto, debe tener un tamaño de 200x200dp

24

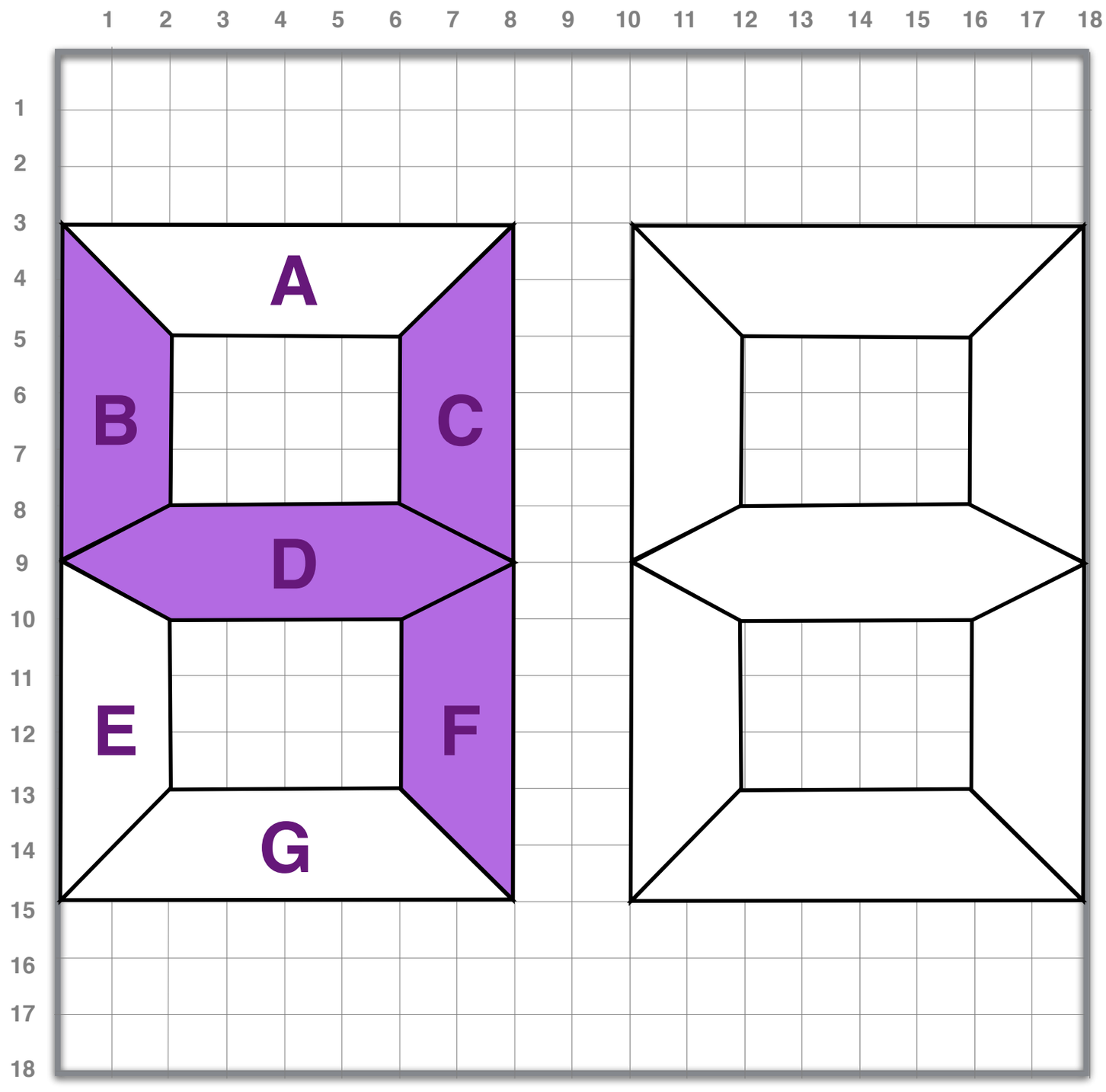

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
A
B
C
D
E
F
G

4: B,C,D,F


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
A
B
C
D
E
F
G

4: B,C,D,F


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
A
B
C
D
E
F
G

```java
public interface IClockView {
    /**
     * Inicia la cuenta regresiva
     */
    void start();

    /**
     * Regresa el reloj a 24 segundos
     */
    void reset();

    /**
     * Detiene el reloj
     */
    void stop();
}
```

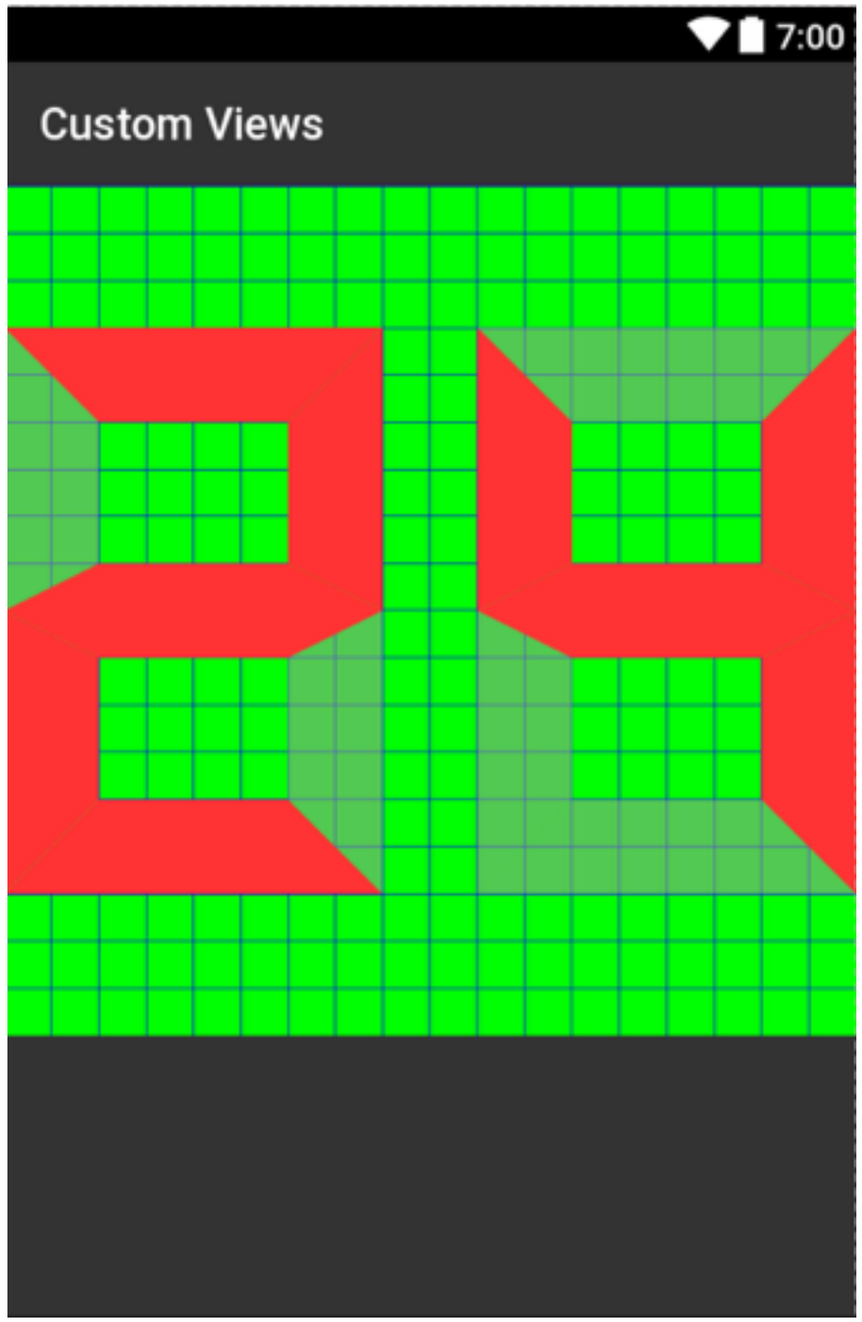
```java
public class ClockView extends View implements IClockView{

    /**
     * Se utiliza cuando se crean vistas manualmente, por código
     * @param context contexto en el cual se infla vista
     */
    public ClockView(Context context) {
        this(context,null);
    }

    /**
     * Se utiliza cuando se crea la vista desde XML.
     * @param context
     * @param attrs
     */
    public ClockView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
        init();

    }

    private void init(){
        backgroundPaint = createPaintFromResource(R.color.colorPrimary);
        squarePaint = createPaintFromResource(R.color.green);
        gridPaint = createPaintFromResource(R.color.blue);
        activeTextPaint = createPaintFromResource(R.color.colorAccent);
        inactiveTextPaint = createPaintFromResource(R.color.colorInactive);

        if(!isInEditMode())
            start();
    }
```

```
public ClockView(Context context, @Nullable AttributeSet attrs) {
    super(context, attrs);
    init();

}

private void init(){
    backgroundPaint = createPaintFromResource(R.color.colorPrimary);
    squarePaint = createPaintFromResource(R.color.green);
    gridPaint = createPaintFromResource(R.color.blue);
    activeTextPaint = createPaintFromResource(R.color.colorAccent);
    inactiveTextPaint = createPaintFromResource(R.color.colorInactive);

    if(!isInEditMode())
        start();
}

@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    final int canvasWidth = canvas.getWidth();
    final int canvasHeight = canvas.getHeight();


    setupGrid(canvasWidth,canvasHeight);

    List<ClockPath> paths = initNumberPaths();

    updatePathsStatesForNumber(paths, currentNumber);

    canvas.drawRect(0,0,canvasWidth,canvasHeight,backgroundPaint);
```

```java
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    final int canvasWidth = canvas.getWidth();
    final int canvasHeight = canvas.getHeight();


    setupGrid(canvasWidth,canvasHeight);

    List<ClockPath> paths = initNumberPaths();
    //prendemos / apagamos los paths para mostrar el numero
    updatePathsStatesForNumber(paths, currentNumber);
    //pintamos el fondo
    canvas.drawRect(0,0,canvasWidth,canvasHeight,backgroundPaint);

    if(showGridBackground) {
        canvas.drawRect(0, 0, canvasSize, canvasSize, squarePaint);
    }

    if(showGrid)
        paintGrid(canvas);

    for(ClockPath clockPath : paths){
        Paint pathPaint = clockPath.isActive ? activePaint : inactivePaint;
        canvas.drawPath(clockPath.path,pathPaint);
    }
}
```

```java
private class ClockPath{
    Path path;
    boolean isActive;

    ClockPath(Path path, boolean isActive) {
        this.path = path;
        this.isActive = isActive;
    }
}
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android=“http://schemas...”
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <com.example.customviews.ClockView
        android:layout_width=“match_parent"
        android:layout_height=“match_parent" />

</LinearLayout>
```



```
//region //Timer manager

private Runnable updateRunnable = new Runnable() {
    @Override public void run() {
        updateClock();
    }
};

@Override public void start() { updateClock(); }

@Override public void stop() { removeCallbacks(updateRunnable);  }

@Override public void reset() { currentNumber = 24; }
```

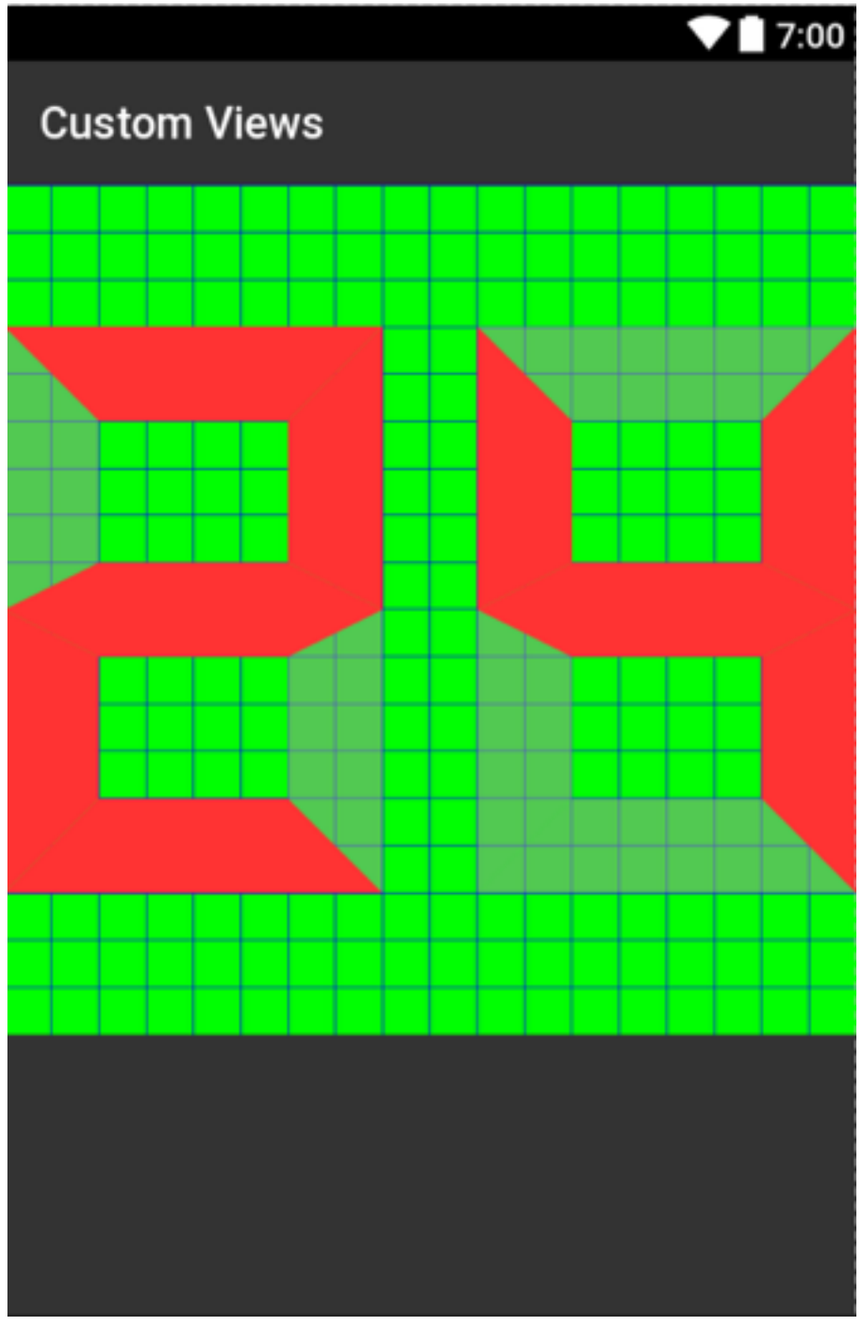
```
private void updateClock() {
    if(currentNumber > 0) {
        currentNumber--;
        invalidate();

        if(currentNumber != 0) {
            //sigue actualizando
            postDelayed(updateRunnable, 1000L);
        }else{
            //suena la bocina
            MediaPlayer mp = MediaPlayer.create(getContext(), R.raw.buzzer);
            mp.start();
        }
    }
}
//endregion
```

```
//region //Timer manager

private Runnable updateRunnable = new Runnable() {
    @Override public void run() {
        updateClock();
    }
};

@Override public void start() { updateClock(); }

@Override public void stop() { removeCallbacks(updateRunnable);  }

@Override public void reset() { currentNumber = 24; }
```

```
private void updateClock() {
    if(currentNumber > 0) {
        currentNumber--;
        invalidate();

        if(currentNumber != 0) {
            //sigue actualizando
            postDelayed(updateRunnable, 1000L);
        }else{
            //suena la vocina
            MediaPlayer mp = MediaPlayer.create(getContext(), R.raw.buzzer);
            mp.start();
        }
    }
}
//endregion
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android=“http://schemas...”
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <com.example.customviews.ClockView
        android:layout_width=“match_parent"
        android:layout_height=“match_parent”
        android:padding=“16dp" />

</LinearLayout>
```

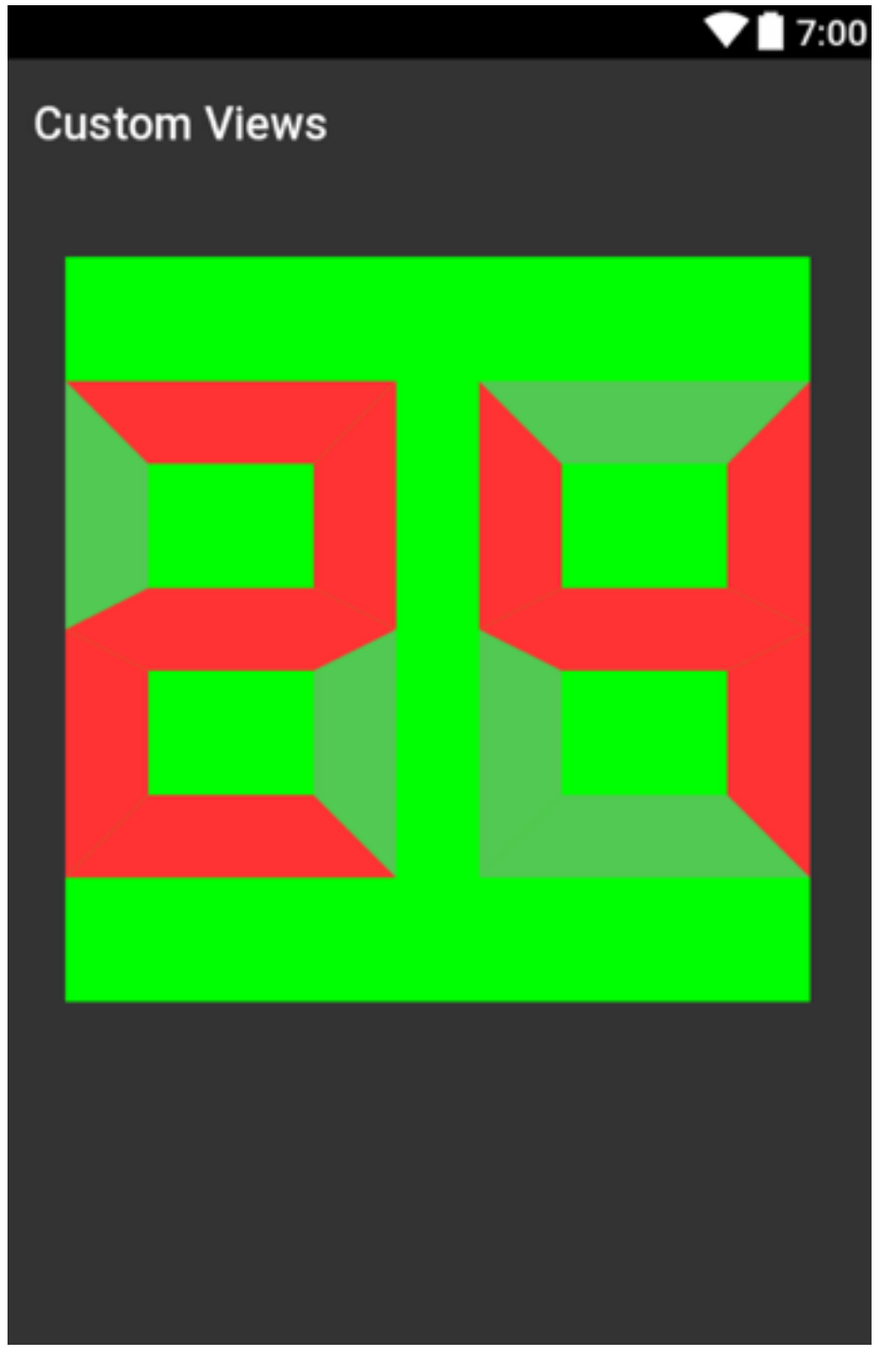
Se ignora el padding!



```java
/**
 * Setea los valores necesarios para dibujar los numeros, en base al espacio disponible para dibujar
 * @param availableWidth ancho disponible para dibujar
 * @param availableHeight alto disponible para dibujar
 */
private void setupGrid(float availableWidth, float availableHeight){
    canvasSize = Math.min(availableWidth, availableHeight);
    if(canvasSize < 0)
        canvasSize = 0;
    nColumns = 18;
    cellSize = canvasSize / nColumns * 1.0f;
}
```

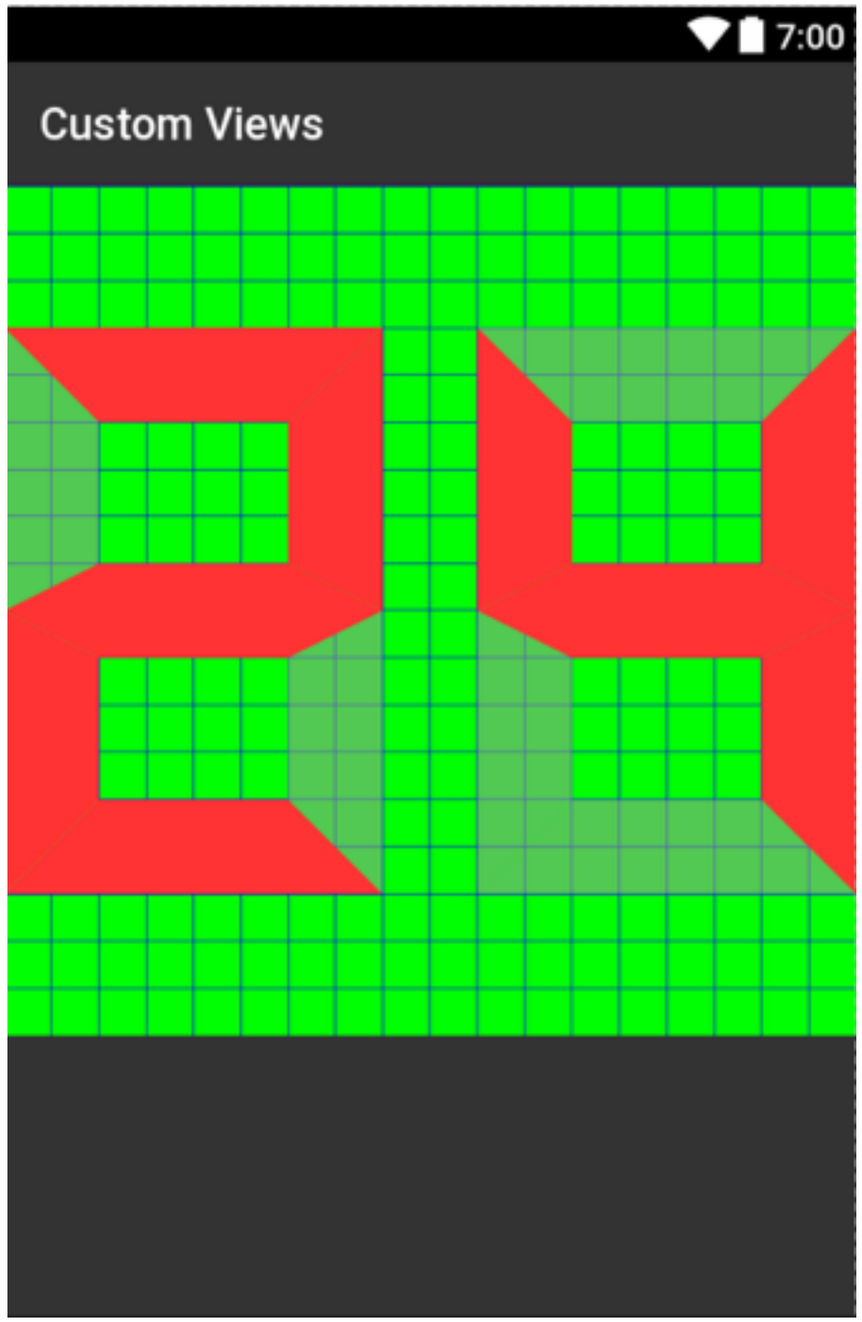
```java
/**
 * Setea los valores necesarios para dibujar los numeros, en base al espacio
disponible para dibujar
 * @param availableWidth ancho disponible para dibujar
 * @param availableHeight alto disponible para dibujar
 */
private void setupGrid(float availableWidth, float availableHeight){
    canvasSize = Math.min(
        availableWidth - getPaddingLeft() - getPaddingRight(),
        availableHeight - getPaddingTop() - getPaddingTop());
    if(canvasSize < 0)
        canvasSize = 0;
    nColumns = 18;
    cellSize = canvasSize / nColumns * 1.0f;
    horizontalOffset = getPaddingLeft();
    verticalOffset = getPaddingTop();
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android=“http://schemas...”
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <com.example.customviews.PaddingClockView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="30dp"
        />


</LinearLayout>
```

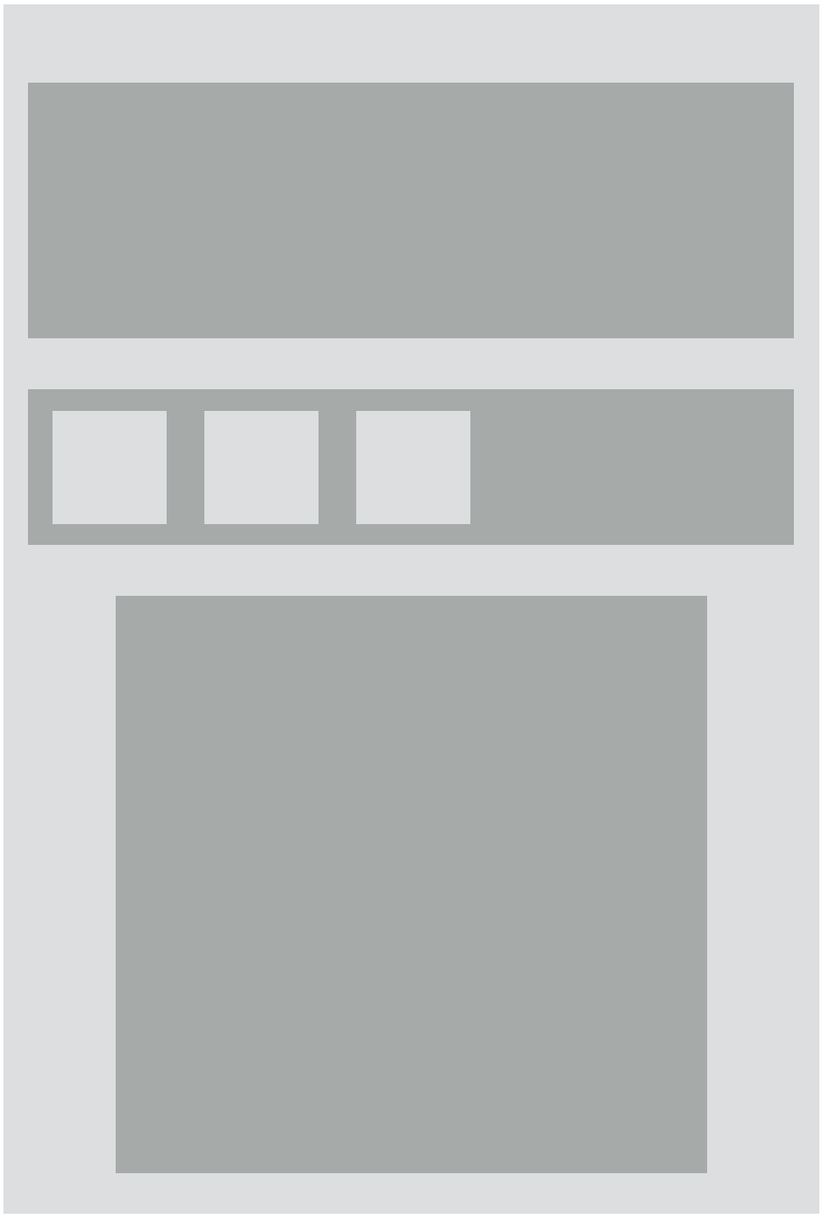


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android=“http://schemas...”
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <com.example.customviews.ClockView
        android:layout_width=“wrap_content”
        android:layout_height=“wrap_content” />

</LinearLayout>
```

Ocupa todo el espacio!

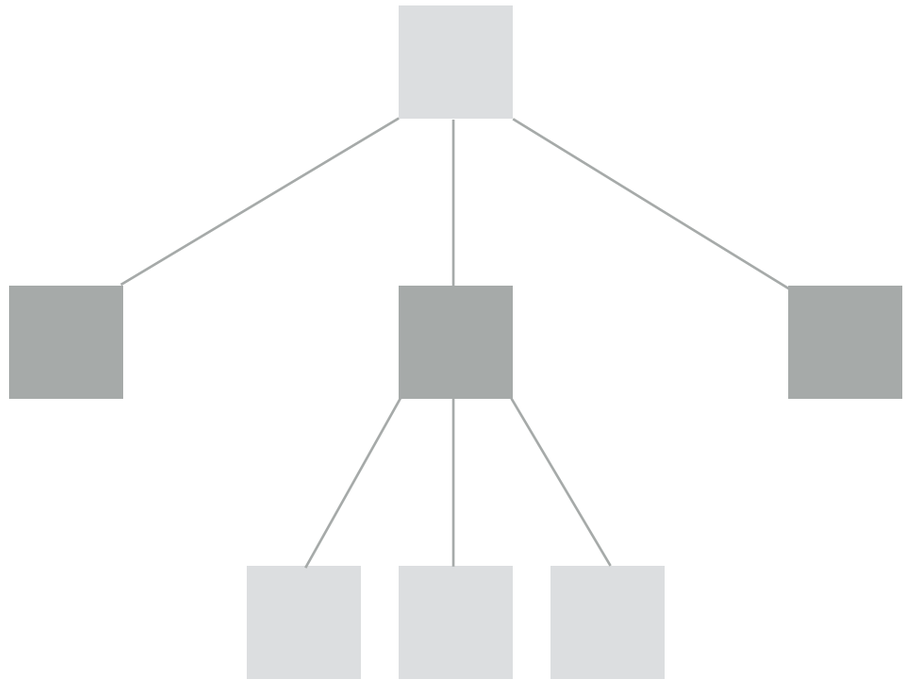


How measurement works?



How measurement works?

How measurement works?



How measurement works?

Créditos:
Huyen Tue Dao
@queencodemonkey

How measurement works?

1 Child defines LayoutParams
in XML or Java


```
<com.example.customviews.ClockView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
```

```
val params = ViewGroup.LayoutParams(
        width = LayoutParams.WRAP_CONTENT,
        height = LayoutParams.WRAP_CONTENT)
child.setLayoutParams(params)
```

How measurement works?

1 Child defines LayoutParams
in XML or Java


```
<com.example.customviews.ClockView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
```

```
val params = ViewGroup.MarginLayoutParams(
            width = LayoutParams.WRAP_CONTENT,
            height = LayoutParams.WRAP_CONTENT)
params.setMargins(…)
child.setLayoutParams(params)
```

How measurement works?

1 Child defines LayoutParams
in XML or Java

PARENT

CHILD

```
<com.example.customviews.ClockView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
```

```
val params = LinearLayout.LayoutParams(
            width = LayoutParams.WRAP_CONTENT,
            height = LayoutParams.WRAP_CONTENT)
params.gravity = Gravity.CENTER
child.setLayoutParams(params)
```

How measurement works?

1 Child defines LayoutParams
in XML or Java

PARENT

CHILD

```
<com.example.customviews.ClockView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
```

```
val params = ConstraintLayout.LayoutParams(
            width = LayoutParams.WRAP_CONTENT,
            height = LayoutParams.WRAP_CONTENT)
params.dimensionRatio = "1:1";
child.setLayoutParams(params)
```

How measurement works?

1 Child defines LayoutParams
in XML or Java

PARENT

child.getLayoutParams()

CHILD

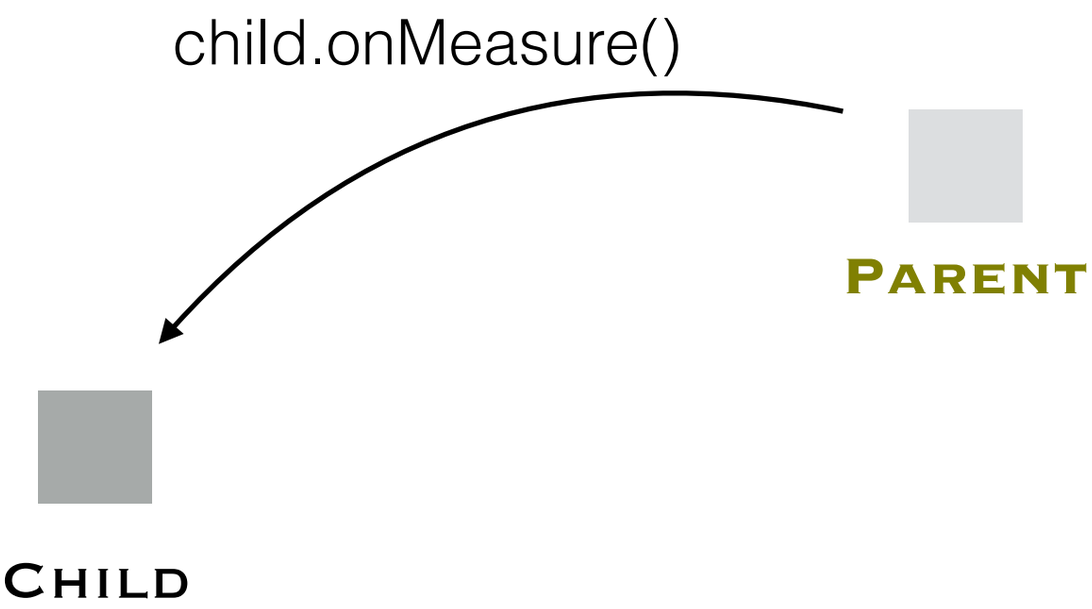
```
<com.example.customviews.ClockView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
```

```
val params = ConstraintLayout.LayoutParams(
        width = LayoutParams.WRAP_CONTENT,
        height = LayoutParams.WRAP_CONTENT)
params.dimensionRatio = "1:1";
child.setLayoutParams(params)
```

How measurement works?

1 Child defines LayoutParams in XML or Java

2 Parent calculates MeasureSpecs and passes to child.onMeasure()




```
@Override
protected void onMeasure(int widthMeasureSpec,
                         int heightMeasureSpec)
```

spec ⟶ { mode | size }

AT_MOST **X** ⟶ como máximo **X**

EXACTLY **X** ⟶ debe medir exactamente **X**

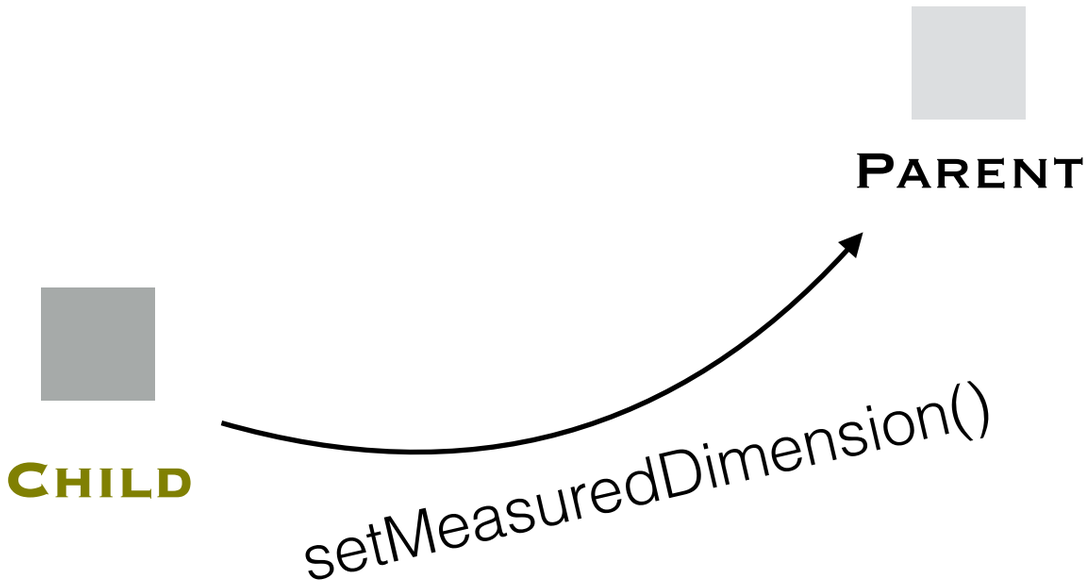
UNSPECIFIED ⟶ sin restricciones!

How measurement works?

1 Child defines LayoutParams in XML or Java

2 Parent calculates MeasureSpecs and passes to child.onMeasure()

3 Child calculates width / height; setMeasureDimension()




```
@Override
protected void onMeasure(int widthMeasureSpec,
                         int heightMeasureSpec){
    //la vista hija calcula cuan grande quiere
    //ser en base a los specs que recibe
    int measuredW = …
    int measuredH = ...
    setMeasuredDimension(measuredW, measuredH)
}
```

obligatorio!!

How measurement works?

1 Child defines LayoutParams in XML or Java

2 Parent calculates MeasureSpecs and passes to child.onMeasure()

3 Child calculates width / height; setMeasureDimension()

CHILD

PARENT

child.getMeasuredWidth()
child.getMeasuredHeight()

```
@Override
protected void onMeasure(int widthMeasureSpec,
                         int heightMeasureSpec){
    //la vista hija calcula cuan grande quiere
    //ser en base a los specs que recibe
    int measuredW = …
    int measuredH = ...
    setMeasuredDimension(measuredW, measuredH)
}
```

obligatorio!!

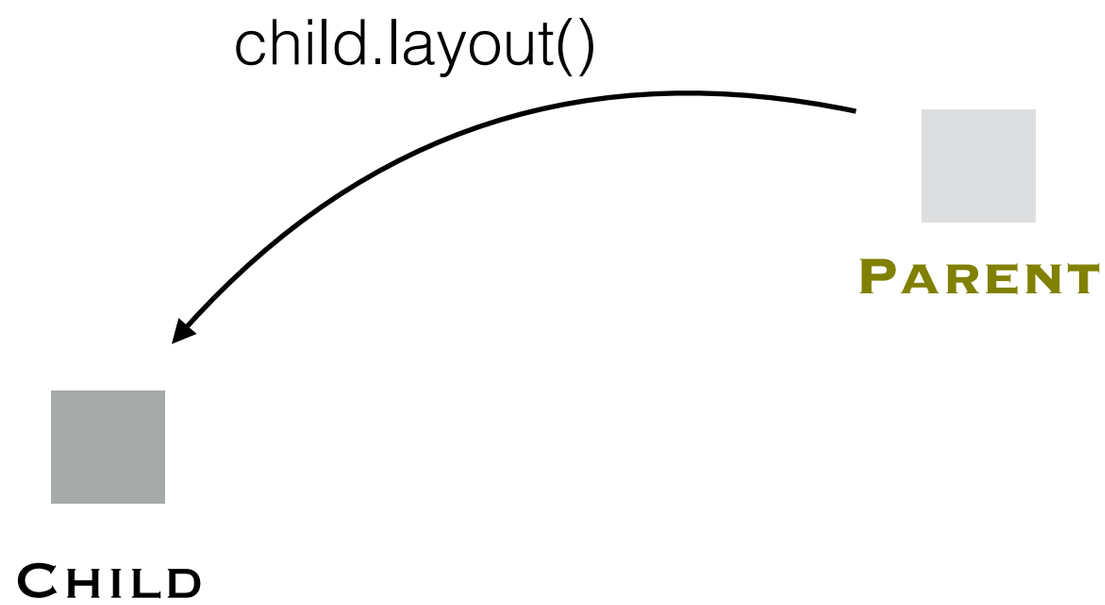
How measurement works?

1 Child defines LayoutParams
in XML or Java

2 Parent calculates MeasureSpecs
and passes to child.onMeasure()

3 Child calculates width / height;
setMeasureDimension()

4 Parent calls child.layout();
Final child size / position



child.getWidth()
child.getHeight()

```java
public class MeasuredClockView extends PaddingClockView {

    private int defaultSize;

    public MeasuredClockView(Context context) {
        this(context,null);
    }

    public MeasuredClockView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    private void init(){
        int density = getContext().getResources().getDisplayMetrics().density;
        defaultSize = (int) (200 * density);
    }

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        int resolvedWidth = resolveSize(defaultSize,widthMeasureSpec);
        int resolvedHeight = resolveSize(defaultSize,heightMeasureSpec);

        setMeasuredDimension(resolvedWidth,resolvedHeight);
    }
}
```

```java
public class MeasuredClockView extends PaddingClockView {

    private int defaultSize;

    public MeasuredClockView(Context context) {
        this(context,null);
    }

    public MeasuredClockView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    private void init(){
        int density = getContext().getResources().getDisplayMetrics().density;
        defaultSize = (int) (200 * density);
    }

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        int resolvedWidth = resolveSize(defaultSize,widthMeasureSpec);
        int resolvedHeight = resolveSize(defaultSize,heightMeasureSpec);

        setMeasuredDimension(resolvedWidth,resolvedHeight);
    }
}
```

```java
public class MeasuredClockView extends PaddingClockView {

    private int defaultSize;

    public MeasuredClockView(Context context) {
        this(context,null);
    }

    public MeasuredClockView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    private void init(){
        int density = getContext().getResources().getDisplayMetrics().density;
        defaultSize = (int) (200 * density);
    }

    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        int resolvedWidth = resolveSize(defaultSize,widthMeasureSpec);
        int resolvedHeight = resolveSize(defaultSize,heightMeasureSpec);

        setMeasuredDimension(resolvedWidth,resolvedHeight);
    }
}
```

```
    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        int resolvedWidth = resolveSize(defaultSize,widthMeasureSpec);
        int resolvedHeight = resolveSize(defaultSize,heightMeasureSpec);

        setMeasuredDimension(resolvedWidth,resolvedHeight);
    }
}
```

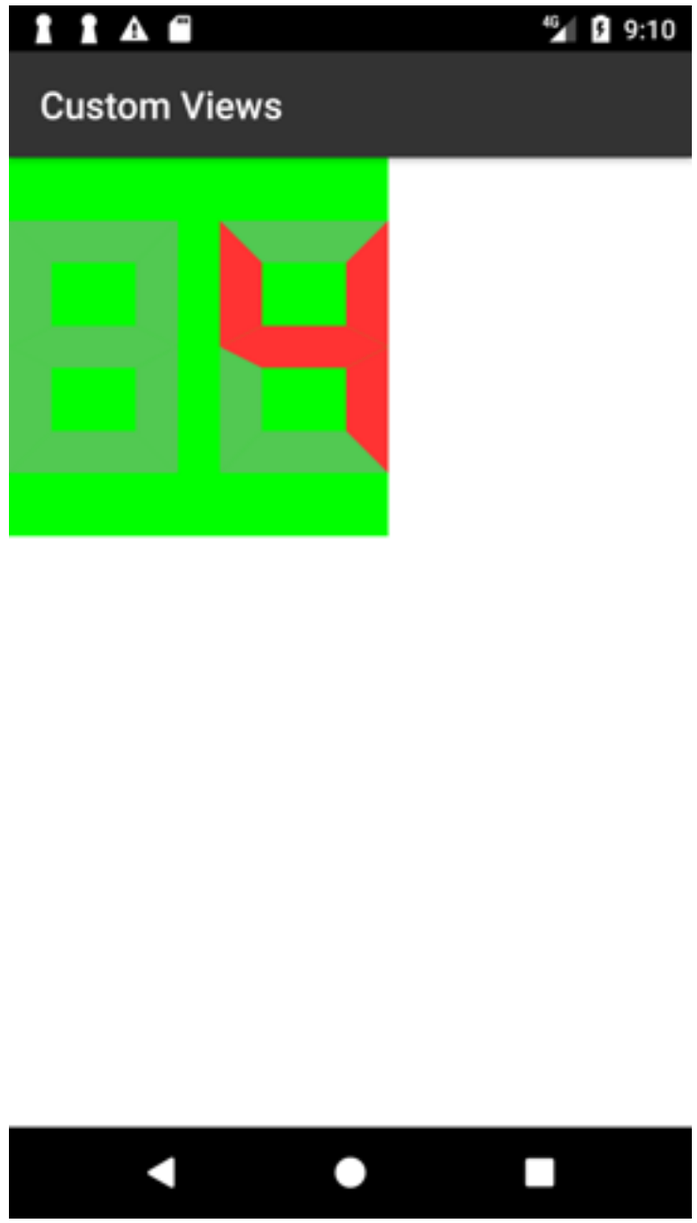
```
    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        int resolvedWidth = resolveSize(defaultSize,widthMeasureSpec);
        int resolvedHeight = resolveSize(defaultSize,heightMeasureSpec);

        setMeasuredDimension(resolvedWidth,resolvedHeight);
    }
}
```

```
    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        int resolvedWidth = resolveSize(defaultSize,widthMeasureSpec);
        int resolvedHeight = resolveSize(defaultSize,heightMeasureSpec);

        setMeasuredDimension(resolvedWidth,resolvedHeight);
    }
}
```

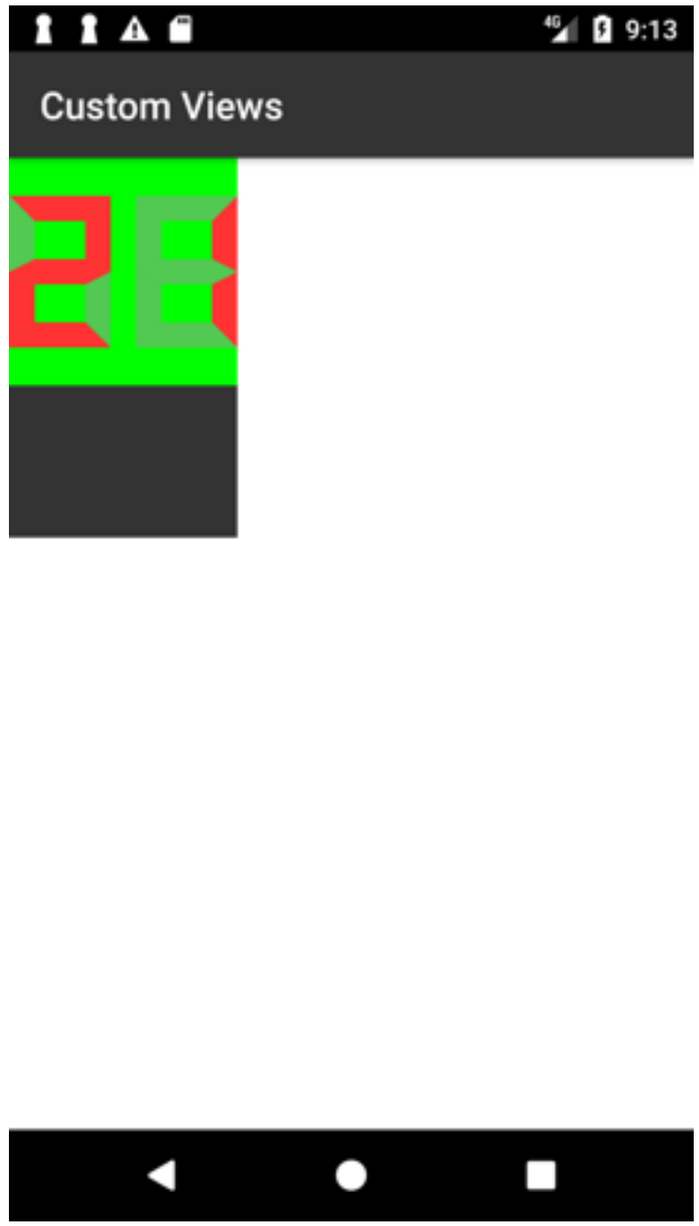
```
    public static int resolveSize(int size, int measureSpec) {
        return resolveSizeAndState(size, measureSpec, 0) & MEASURED_SIZE_MASK;
    }

    public static int resolveSizeAndState(int size, int measureSpec, int state) {
        int specMode = MeasureSpec.getMode(measureSpec);
        int specSize = MeasureSpec.getSize(measureSpec);
        final int result;
        switch (specMode) {
            case MeasureSpec.AT_MOST:
                if (specSize < size)
                    result = specSize | MEASURED_STATE_TOO_SMALL;
                else
                    result = size;
            case MeasureSpec.EXACTLY:
                result = specSize;
            case MeasureSpec.UNSPECIFIED:
            default:
                result = size;
        }
        return result | (state & MEASURED_STATE_MASK);
    }
```

```
    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        int resolvedWidth = resolveSize(defaultSize,widthMeasureSpec);
        int resolvedHeight = resolveSize(defaultSize,heightMeasureSpec);

        setMeasuredDimension(resolvedWidth,resolvedHeight);
    }
}
```

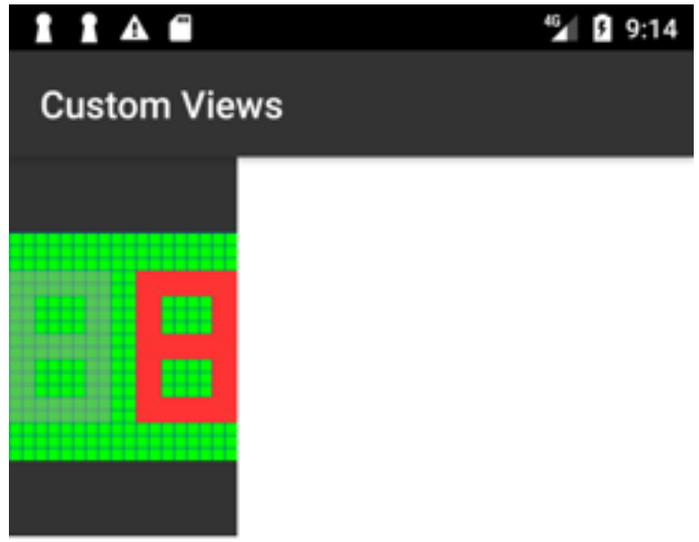
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android=“http://schemas...”
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <com.example.customviews.MeasuredClockView
        android:layout_width=“wrap_content"
        android:layout_height=“wrap_content"
        />


</LinearLayout>
```



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android=“http://schemas...”
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <com.example.customviews.MeasuredClockView
        android:layout_width=“120dp"
        android:layout_height=“200dp"
        />


</LinearLayout>
```



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android=“http://schemas...”
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <com.example.customviews.AdaptableClockView
        android:layout_width=“120dp"
        android:layout_height=“200dp"
        />


</LinearLayout>
```



Check out the code!

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    final int canvasWidth = canvas.getWidth();
    final int canvasHeight = canvas.getHeight();


    setupGrid(canvasWidth,canvasHeight);

    List<ClockPath> paths = initNumberPaths();

    updatePathsStatesForNumber(paths, currentNumber);

    canvas.drawRect(0,0,canvasWidth,canvasHeight,backgroundPaint);

    if(showGridBackground) {
        canvas.drawRect(0, 0, canvasSize, canvasSize, squarePaint);
    }

    if(showGrid)
        paintGrid(canvas);

    for(ClockPath clockPath : paths){
        Paint pathPaint = clockPath.isActive ? activePaint : inactivePaint;
        canvas.drawPath(clockPath.path,pathPaint);
    }
}
```

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    final int canvasWidth = canvas.getWidth();
    final int canvasHeight = canvas.getHeight();


    setupGrid(canvasWidth,canvasHeight);

    List<ClockPath> paths = initNumberPaths();

    updatePathsStatesForNumber(paths, currentNumber);

    canvas.drawRect(0,0,canvasWidth,canvasHeight,backgroundPaint);

    if(showGridBackground) {
        canvas.drawRect(0, 0, canvasSize, canvasSize, squarePaint);
    }

    if(showGrid)
        paintGrid(canvas);

    for(ClockPath clockPath : paths){
        Paint pathPaint = clockPath.isActive ? activePaint : inactivePaint;
        canvas.drawPath(clockPath.path,pathPaint);
    }
}
```

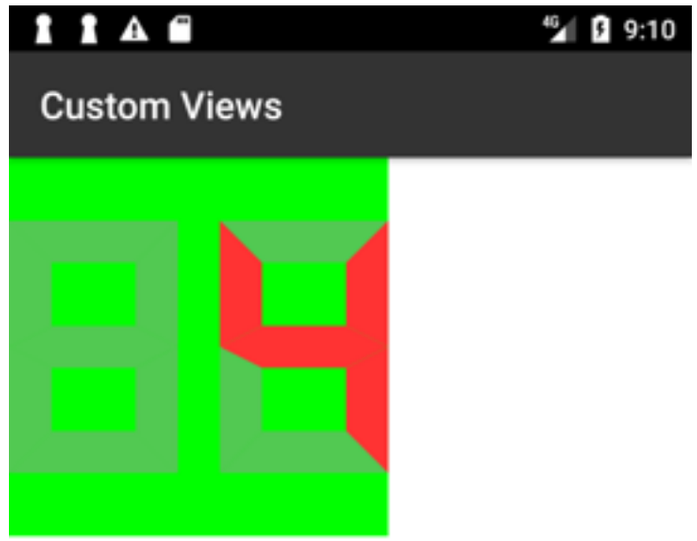
```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    final int canvasWidth = canvas.getWidth();
    final int canvasHeight = canvas.getHeight();


            (canvasWidth,canvasHeight);

    List<ClockPath> paths =

    updatePathsStatesForNumber(paths, currentNumber);

    canvas.drawRect(0,0,canvasWidth,canvasHeight,backgroundPaint);

    if(showGridBackground) {
        canvas.drawRect(0, 0, canvasSize, canvasSize, squarePaint);
    }

    if(showGrid)
        paintGrid(canvas);

    for(ClockPath clockPath : paths){
        Paint pathPaint = clockPath.isActive ? activePaint : inactivePaint;
        canvas.drawPath(clockPath.path,pathPaint);
    }
}
```

```
setupGrid

initNumberPaths();
```

```
                                                    setupGrid
                                                    initNumberPaths();
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    int resolvedWidth = resolveSize(defaultSize,widthMeasureSpec);
    int resolvedHeight = resolveSize(defaultSize,heightMeasureSpec);




    setMeasuredDimension(resolvedWidth,resolvedHeight);
}
```

```java
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    int resolvedWidth = resolveSize(defaultSize,widthMeasureSpec);
    int resolvedHeight = resolveSize(defaultSize,heightMeasureSpec);

    setupGrid(resolvedWidth,resolvedHeight);
    paths = initNumberPaths();

    setMeasuredDimension(resolvedWidth,resolvedHeight);
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android=“http://schemas...”
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <com.example.customviews.OptimizedClockView
        android:layout_width=“wrap_content"
        android:layout_height=“wrap_content"
        />


</LinearLayout>
```



Personalizando nuestro view

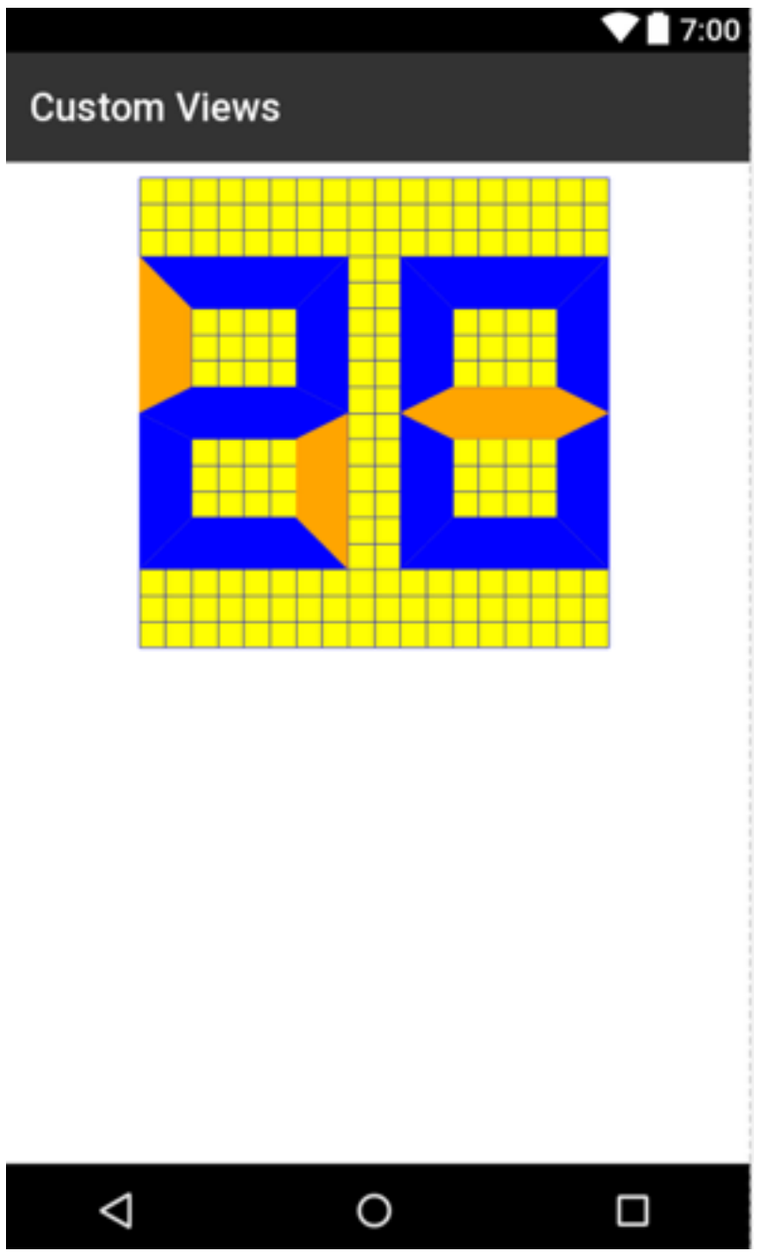
attrs.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <declare-styleable name="clock_view">
        <attr name="background_color" format="color"/>
        <attr name="active_text_color" format="color"/>
        <attr name="inactive_text_color" format="color"/>
        <attr name="show_grid" format="boolean"/>
        <attr name="show_square" format="boolean"/>
        <attr name="default_value" format="integer"/>
    </declare-styleable>

</resources>
```

```java
public class CustomizableClockView extends OptimizedClockView {
    public CustomizableClockView(Context context) {
        this(context, null);
    }

    public CustomizableClockView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
        init(attrs);
    }

    private void init(@Nullable AttributeSet attrs){

        TypedArray a = getContext().getTheme().obtainStyledAttributes(
                attrs,
                R.styleable.clock_view,
                0, 0);
       try {
        //Read custom background color
        int backColor = a.getColor(R.styleable.clock_view_background_color, -1)
        if(backColor != -1){
            mBackgroundColor = backColor;
        }
        //Read custom active text color
        int activeColor = a.getColor(R.styleable.clock_view_active_text_color, -1);
        if(activeColor != -1){
            mActiveTextColor = activeColor;
        }
        //Read custom inactive text color
        int inactiveColor = a.getColor(R.styleable.clock_view_inactive_text_color, -1);
        if(inactiveColor != -1){
            mInactiveTextColor = inactiveColor;
        }
        //Set grid
        mShowGrid = a.getBoolean(R.styleable.clock_view_show_grid,false);
        mShowGridBackground = a.getBoolean(R.styleable.clock_view_show_square,false);
```

```
private void init(@Nullable AttributeSet attrs){

    TypedArray a = getContext().getTheme().obtainStyledAttributes(
            attrs,
            R.styleable.clock_view,
            0, 0);
    try {
        //Read custom background color
        int backColor = a.getColor(R.styleable.clock_view_background_color, -1)
        if(backColor != -1){
            mBackgroundColor = backColor;
        }
        //Read custom active text color
        int activeColor = a.getColor(R.styleable.clock_view_active_text_color, -1);
        if(activeColor != -1){
            mActiveTextColor = activeColor;
        }
        //Read custom inactive text color
        int inactiveColor = a.getColor(R.styleable.clock_view_inactive_text_color, -1);
        if(inactiveColor != -1){
            mInactiveTextColor = inactiveColor;
        }
        //Set grid
        mShowGrid = a.getBoolean(R.styleable.clock_view_show_grid,false);
        mShowGridBackground = a.getBoolean(R.styleable.clock_view_show_square,false);
        //Set default number
        int defaultNumber = a.getInt(R.styleable.clock_view_default_value,24);
        if(defaultNumber >= 0 && defaultNumber < 25)
            mCurrentNumber = defaultNumber;

    } finally {
        a.recycle();
    }

    super.init();
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:background="#fff"
    android:orientation="vertical">


    <com.example.customviews.CustomizableClockView
        android:layout_width="240dp"
        android:layout_height="240dp"
        android:layout_gravity="center_horizontal"
        android:layout_margin="8dp"
        app:background_color=“#ff0" //amarillo
        app:active_text_color=“#00f" //azul
        app:inactive_text_color=“#ffa500" //naranja
        app:show_grid="true"
        app:default_value="20" />

</LinearLayout>
```



Recapitulando...

- Nos dan flexibilidad, pero no son sencillos de crear
- Para los custom views, importante:
    - onDraw
    - onMeasure
- Para los custom view groups, importante:
    - onMeasure
    - onLayout

Info adicional

- Huyen Tue Dao (@queencodemonkey)
  Measure, Layout, Draw, Repeat

- Caster.IO
  Custom Views & View Groups

- Android Docs
  Creating a View Class

- Alberto Ballano (@aballano)
  Android layouts to the next level: Custom Views, Compound ViewGroups and Custom ViewGroups

- Caren Chang (@calren24)
  Advanced Android Touches

Gracias!

Repositorio:
https://github.com/Bruno125/Custom-Views-Demo

Slides:
https://speakerdeck.com/bruno125/custom-views

@brunoaybarg
@bruno.aybar
Bruno125