

Nombre de la práctica	Figuras			No.	1
Asignatura:	Desarrollo de aplicaciones móviles	Carrera:	INGENIERÍA EN SISTEMAS COMPUTACIONALES	Duración de la práctica (Hrs)	2 horas

GRUPO: 3701

NOMBRE: Shania Kinnereth Diaz Moya

```
package com.tesji.figuras

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Canvas
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.geometry.Offset
import androidx.compose.ui.geometry.Size
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.graphics.Path

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Paisaje()
        }
    }
}

@Composable
fun Paisaje() {
    Canvas(modifier = Modifier.fillMaxSize()) {
        val ancho = size.width
        val alto = size.height

        // Cielo
        drawRect(
            color = Color(0xFF87CEEB), // Azul cielo
            topLeft = Offset(0f, 0f),
            size = Size(ancho, alto)
        )

        // Sol
        drawCircle(
            color = Color.Yellow,
            radius = ancho / 10,
            center = Offset(ancho * 0.15f, alto * 0.15f)
        )

        // Montañas (triángulos)
    }
}
```

```
val montaña1 = Path().apply {
    moveTo(ancho * 0.1f, alto * 0.7f)
   .lineTo(ancho * 0.4f, alto * 0.3f)
   .lineTo(ancho * 0.7f, alto * 0.7f)
    close()
}
drawPath(montaña1, color = Color(0xFF556B2F))

val montaña2 = Path().apply {
    moveTo(ancho * 0.4f, alto * 0.7f)
   .lineTo(ancho * 0.7f, alto * 0.25f)
   .lineTo(ancho * 0.95f, alto * 0.7f)
    close()
}
drawPath(montaña2, color = Color(0xFF6B8E23))

// Césped (parte inferior)
drawRect(
    color = Color(0xFF228B22),
    topLeft = Offset(0f, alto * 0.7f),
    size = Size(ancho, alto * 0.3f)
)

// Río
drawRect(
    color = Color(0xFF1E90FF),
    topLeft = Offset(ancho * 0.35f, alto * 0.7f),
    size = Size(ancho * 0.3f, alto * 0.3f)
)

// Árboles
val cantidadArboles = 4
val espacio = ancho / (cantidadArboles + 1)
for (i in 1..cantidadArboles) {
    val x = espacio * i
    val troncoAlto = alto * 0.15f
    val troncoAncho = ancho * 0.03f

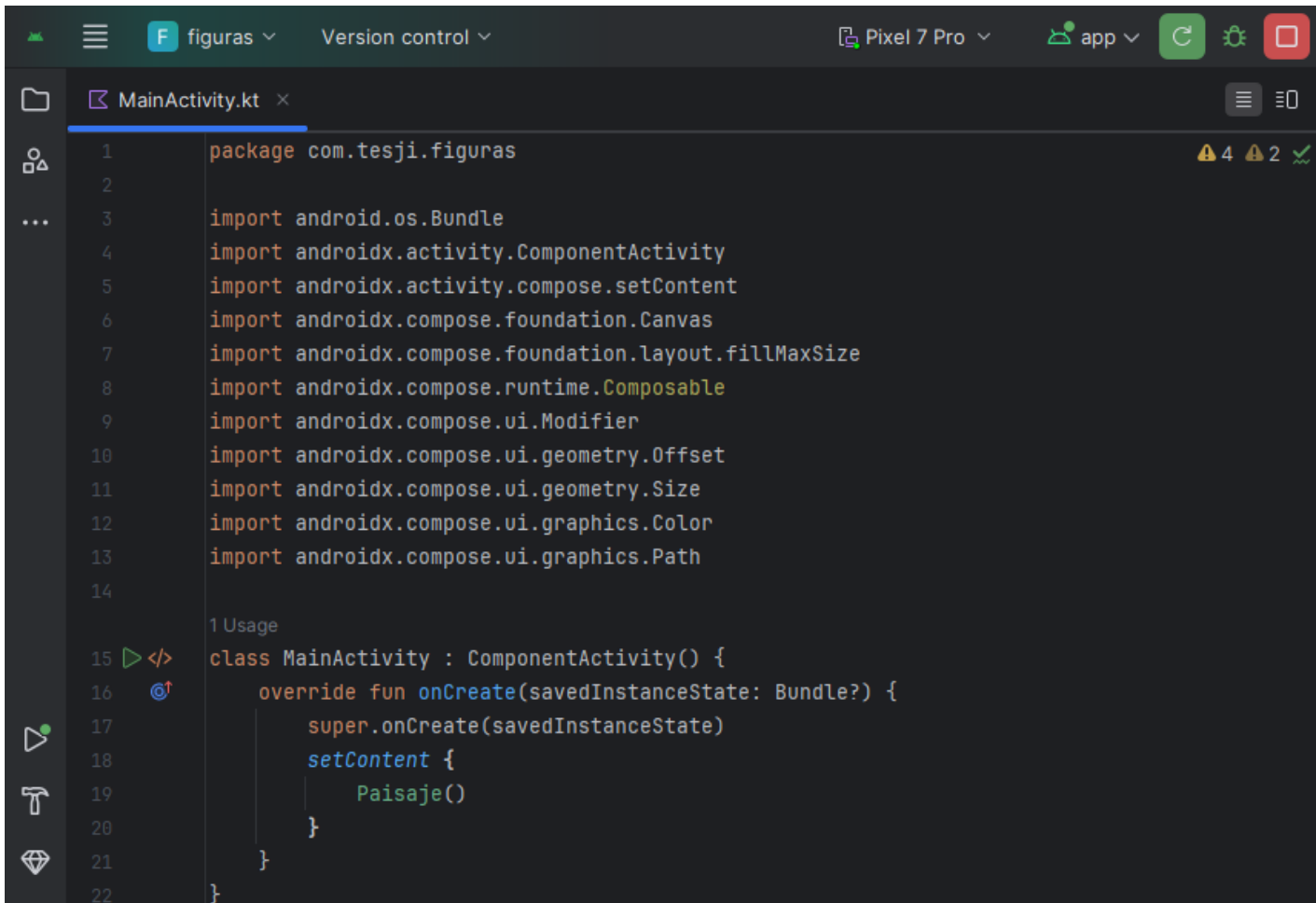
    // Tronco
    drawRect(
        color = Color(0xFF8B4513),
        topLeft = Offset(x - troncoAncho / 2, alto * 0.7f - troncoAlto),
        size = Size(troncoAncho, troncoAlto)
    )

    // Copa
    val radio = ancho * 0.07f
    drawCircle(
        color = Color(0xFF2E7D32),
        radius = radio,
        center = Offset(x, alto * 0.7f - troncoAlto)
    )
    drawCircle(
        color = Color(0xFF388E3C),
        radius = radio,
```

```
        center = Offset(x - radio * 0.6f, alto * 0.7f - troncoAlto + radio *  
0.3f)  
    )  
    drawCircle(  
        color = Color(0xFF43A047),  
        radius = radio,  
        center = Offset(x + radio * 0.6f, alto * 0.7f - troncoAlto + radio *  
0.3f)  
    )  
}  
}
```

Para realizar el bosque

1.- Importamos las bibliotecas y dejamos la clase main



```
package com.tesji.figuras  
  
import android.os.Bundle  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import androidx.compose.foundation.Canvas  
import androidx.compose.foundation.layout.fillMaxSize  
import androidx.compose.runtime.Composable  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.geometry.Offset  
import androidx.compose.ui.geometry.Size  
import androidx.compose.ui.graphics.Color  
import androidx.compose.ui.graphics.Path  
  
1 Usage  
class MainActivity : ComponentActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContent {  
            Paisaje()  
        }  
    }  
}
```

2.- se realiza el cielo de la siguiente manera

```
drawRect(  
    color = Color( color = 0xFF87CEEB), // Azul cielo  
    topLeft = Offset( x = 0f, y = 0f),  
    size = Size( width = ancho, height = alto)  
)
```

3.- Realizamos el sol

```
drawCircle(  
    color = Color.Yellow,  
    radius = ancho / 10,  
    center = Offset( x = ancho * 0.15f, y = alto * 0.15f)  
)
```

4.- Comenzamos con las montañas

```
val montaña1 = Path().apply {  
    moveTo( x = ancho * 0.1f, y = alto * 0.7f)  
    lineTo( x = ancho * 0.4f, y = alto * 0.3f)  
    lineTo( x = ancho * 0.7f, y = alto * 0.7f)  
    close()  
}  
drawPath( path = montaña1, color = Color( color = 0xFF556B2F))  
  
val montaña2 = Path().apply {  
    moveTo( x = ancho * 0.4f, y = alto * 0.7f)  
    lineTo( x = ancho * 0.7f, y = alto * 0.25f)  
    lineTo( x = ancho * 0.95f, y = alto * 0.7f)  
    close()  
}  
drawPath( path = montaña2, color = Color( color = 0xFF6B8E23))
```

5.- Hacemos el pasto

```
drawRect(  
    color = Color( color = 0xFF228B22),  
    topLeft = Offset( x = 0f, y = alto * 0.7f),  
    size = Size( width = ancho, height = alto * 0.3f)  
)
```

6.- Hacemos un rio en medio

```
drawRect(  
    color = Color( color = 0xFF1E90FF),  
    topLeft = Offset( x = ancho * 0.35f, y = alto * 0.7f),  
    size = Size( width = ancho * 0.3f, height = alto * 0.3f)  
)
```

7.- Comenzamos a realizar el árbol por partes

```
val cantidadArboles = 4  
val espacio = ancho / (cantidadArboles + 1)  
for (i in 1 ≤ .. ≤ cantidadArboles) {  
    val x = espacio * i  
    val troncoAlto = alto * 0.15f  
    val troncoAncho = ancho * 0.03f
```

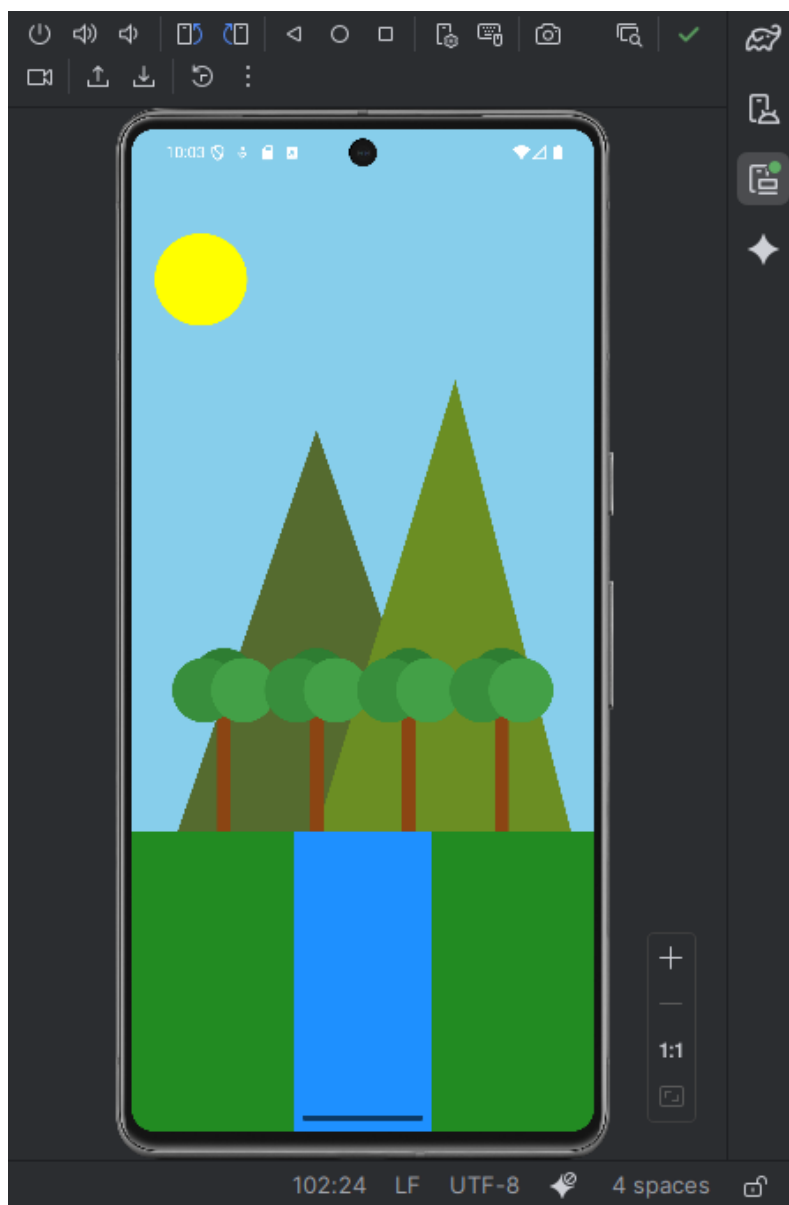
8.- Dividimos la realización del árbol primero con el tronco

```
drawRect(  
    color = Color( color = 0xFF8B4513),  
    topLeft = Offset( x = x - troncoAncho / 2, y = alto * 0.7f - troncoAlto),  
    size = Size( width = troncoAncho, height = troncoAlto)  
)
```

9.- Continuamos con la realización de la copa

```
val radio = ancho * 0.07f
drawCircle(
    color = Color( color = 0xFF2E7D32),
    radius = radio,
    center = Offset(x, y = alto * 0.7f - troncoAlto)
)
drawCircle(
    color = Color( color = 0xFF388E3C),
    radius = radio,
    center = Offset( x = x - radio * 0.6f, y = alto * 0.7f - troncoAlto + radio * 0.3f)
)
drawCircle(
    color = Color( color = 0xFF43A047),
    radius = radio,
    center = Offset( x = x + radio * 0.6f, y = alto * 0.7f - troncoAlto + radio * 0.3f)
)
```

10.- Resultado



Conclusión:

Usar Canvas en Compose es útil porque te da libertad total para dibujar lo que quieras: figuras, paisajes, animaciones o hasta juegos sencillos. No estás limitado solo a los componentes ya hechos, sino que puedes crear tus propios diseños a mano. Además, te ayuda a entender mejor cómo funcionan las coordenadas, los tamaños y los colores, y eso hace que puedas personalizar mucho más tus apps. En pocas palabras, con Canvas puedes dejar volar tu creatividad y darle un toque único a tus proyectos.