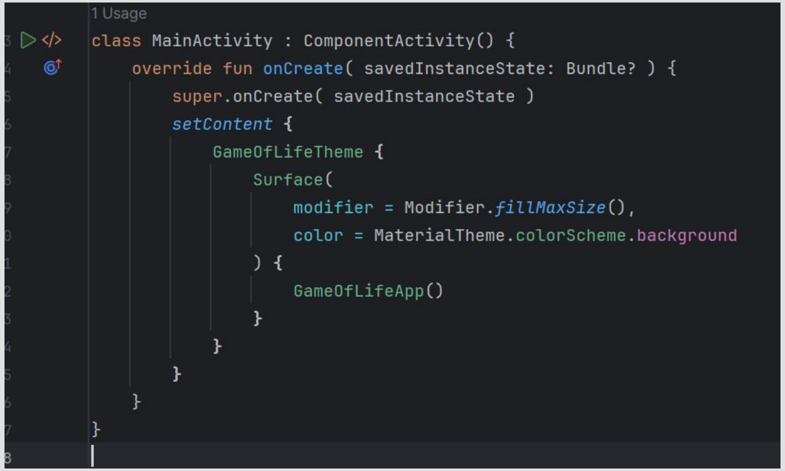
Conway’s Game of Life

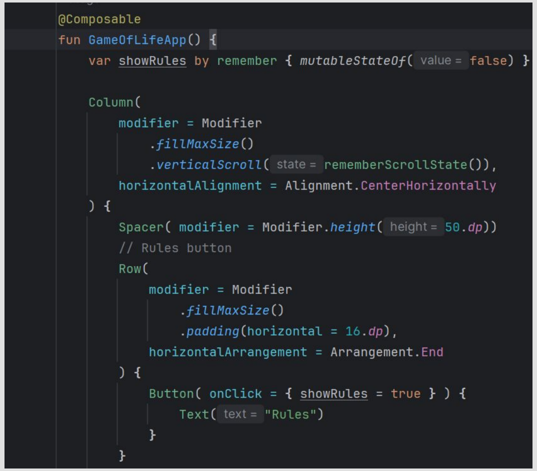
Trevor Dove & Liliana Martinez

December 3, 2025

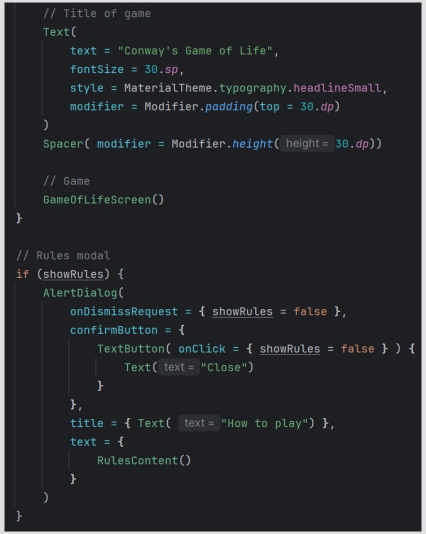
Content



```
1 Usage
class MainActivity : ComponentActivity() {
    override fun onCreate( savedInstanceState: Bundle? ) {
        super.onCreate( savedInstanceState )
        setContent {
            GameOfLifeTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    GameOfLifeApp()
                }
            }
        }
    }
}
```

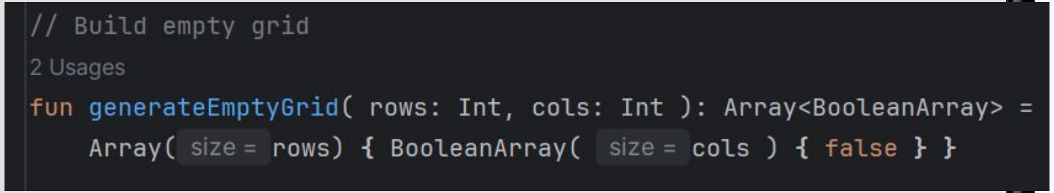


```
@Composable
fun GameOfLifeApp() {
    var showRules by remember { mutableStateOf( value = false) }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .verticalScroll( state = rememberScrollState()),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Spacer( modifier = Modifier.height( height = 50.dp))
        // Rules button
        Row(
            modifier = Modifier
                .fillMaxSize()
                .padding(horizontal = 16.dp),
            horizontalArrangement = Arrangement.End
        ) {
            Button( onClick = { showRules = true } ) {
                Text( text = "Rules")
            }
        }
```

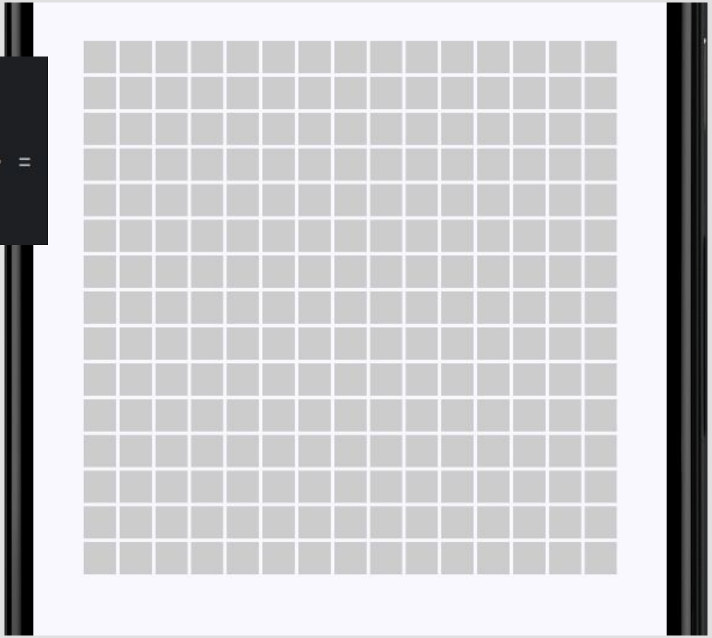


```
        // Title of game
        Text(
            text = "Conway's Game of Life",
            fontSize = 30.sp,
            style = MaterialTheme.typography.headlineSmall,
            modifier = Modifier.padding(top = 30.dp)
        )
        Spacer( modifier = Modifier.height( height = 30.dp))

        // Game
        GameOfLifeScreen()
    }

    // Rules modal
    if (showRules) {
        AlertDialog(
            onDismissRequest = { showRules = false },
            confirmButton = {
                TextButton( onClick = { showRules = false } ) {
                    Text( text = "Close")
                }
            },
            title = { Text( text = "How to play") },
            text = {
                RulesContent()
            }
        )
    }
```

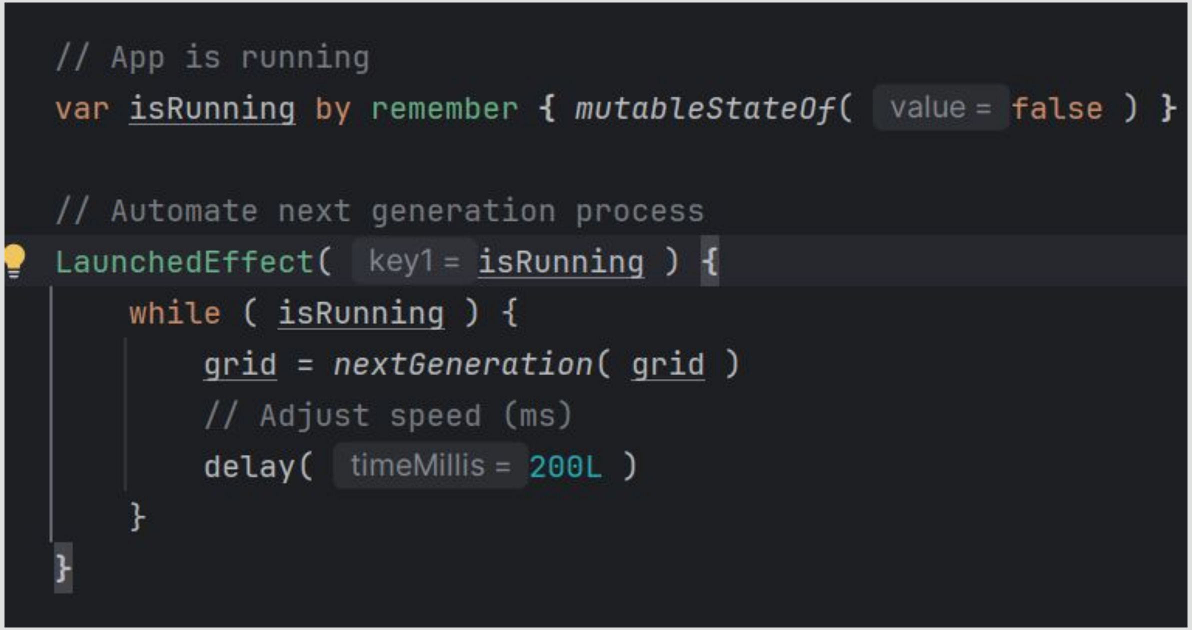
The App: Overall Structure



```
// Build empty grid
2 Usages
fun generateEmptyGrid( rows: Int, cols: Int ): Array<BooleanArray> =
    Array( size = rows) { BooleanArray( size = cols ) { false } }
```

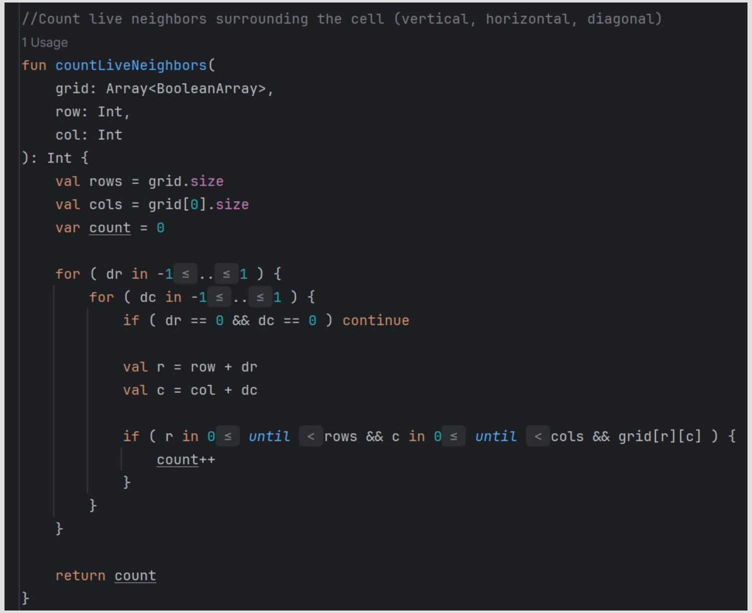


The App: Grid

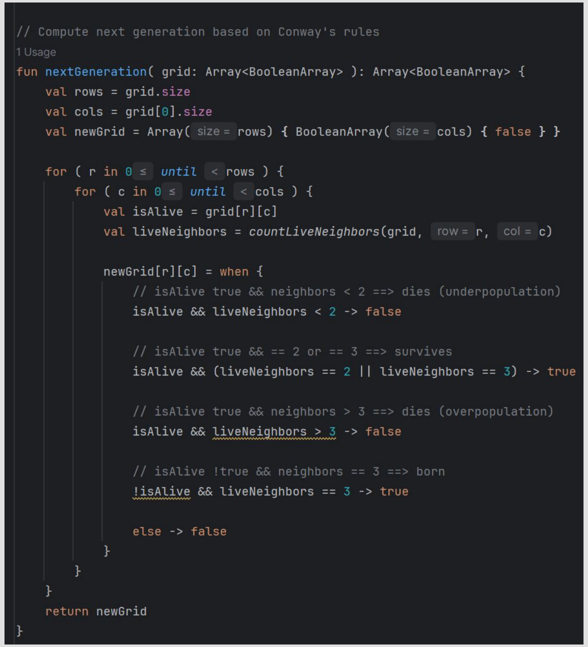


```
// App is running
var isRunning by remember { mutableStateOf( value = false ) }

// Automate next generation process
LaunchedEffect( key1 = isRunning ) {
    while ( isRunning ) {
        grid = nextGeneration( grid )
        // Adjust speed (ms)
        delay( timeMillis = 200L )
    }
}
```

The App: Running the Simulation

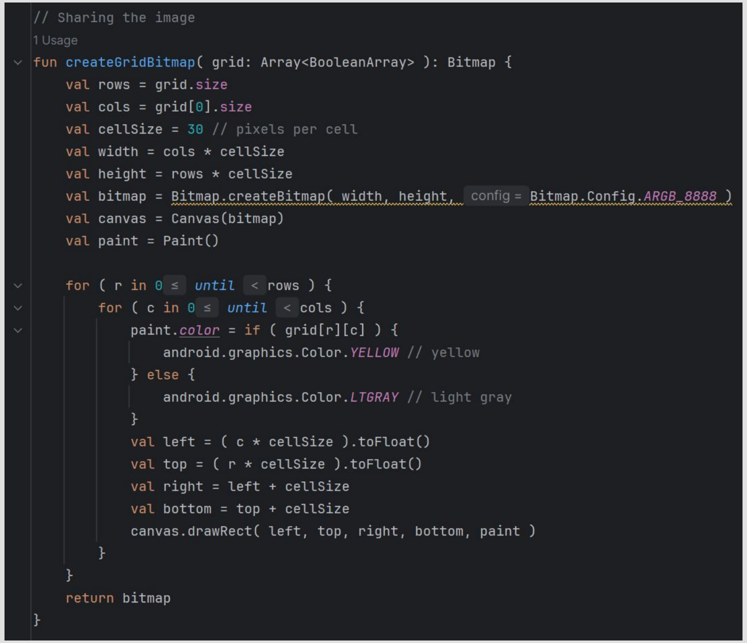


```
//Count live neighbors surrounding the cell (vertical, horizontal, diagonal)
1 Usage
fun countLiveNeighbors(
    grid: Array<BooleanArray>,
    row: Int,
    col: Int
): Int {
    val rows = grid.size
    val cols = grid[0].size
    var count = 0

    for ( dr in -1 ≤ .. ≤ 1 ) {
        for ( dc in -1 ≤ .. ≤ 1 ) {
            if ( dr == 0 && dc == 0 ) continue

            val r = row + dr
            val c = col + dc

            if ( r in 0 ≤  until  < rows && c in 0 ≤  until  < cols && grid[r][c] ) {
                count++
            }
        }
    }

    return count
}
```

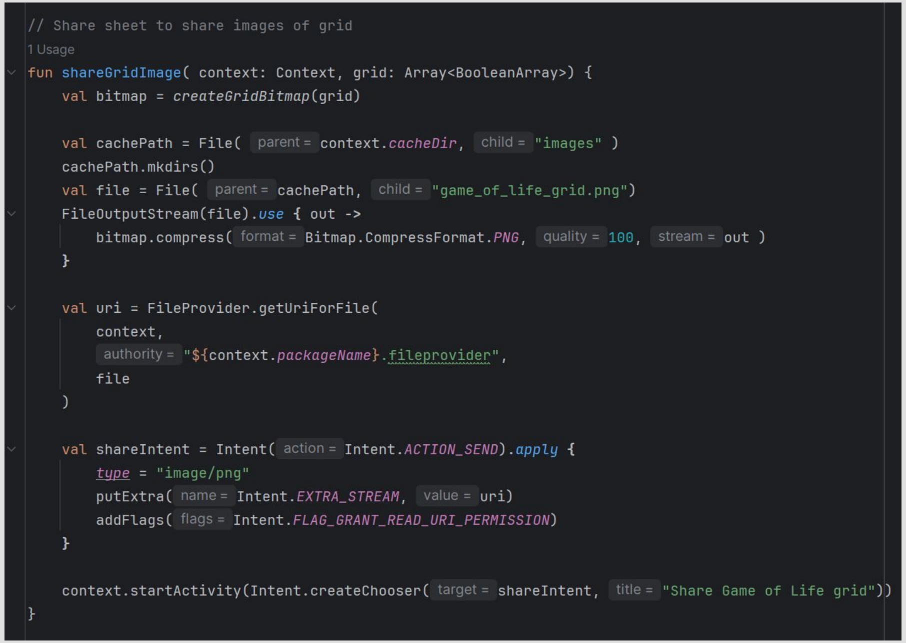


```
// Compute next generation based on Conway's rules
1 Usage
fun nextGeneration( grid: Array<BooleanArray> ): Array<BooleanArray> {
    val rows = grid.size
    val cols = grid[0].size
    val newGrid = Array( size = rows) { BooleanArray( size = cols) { false } }

    for ( r in 0 ≤  until  < rows ) {
        for ( c in 0 ≤  until  < cols ) {
            val isAlive = grid[r][c]
            val liveNeighbors = countLiveNeighbors(grid,  row = r,  col = c)

            newGrid[r][c] = when {
                // isAlive true && neighbors < 2 ==> dies (underpopulation)
                isAlive && liveNeighbors < 2 -> false

                // isAlive true && == 2 or == 3 ==> survives
                isAlive && (liveNeighbors == 2 || liveNeighbors == 3) -> true

                // isAlive true && neighbors > 3 ==> dies (overpopulation)
                isAlive && liveNeighbors > 3 -> false

                // isAlive !true && neighbors == 3 ==> born
                !isAlive && liveNeighbors == 3 -> true

                else -> false
            }
        }
    }
    return newGrid
}
```

The App: Actual Game



```
// Sharing the image
1 Usage
fun createGridBitmap( grid: Array<BooleanArray> ): Bitmap {
    val rows = grid.size
    val cols = grid[0].size
    val cellSize = 30 // pixels per cell
    val width = cols * cellSize
    val height = rows * cellSize
    val bitmap = Bitmap.createBitmap( width, height, config = Bitmap.Config.ARGB_8888 )
    val canvas = Canvas(bitmap)
    val paint = Paint()

    for ( r in 0 ≤ until < rows ) {
        for ( c in 0 ≤ until < cols ) {
            paint.color = if ( grid[r][c] ) {
                android.graphics.Color.YELLOW // yellow
            } else {
                android.graphics.Color.LTGRAY // light gray
            }
            val left = ( c * cellSize ).toFloat()
            val top = ( r * cellSize ).toFloat()
            val right = left + cellSize
            val bottom = top + cellSize
            canvas.drawRect( left, top, right, bottom, paint )
        }
    }
    return bitmap
}
```



```
// Share sheet to share images of grid
1 Usage
fun shareGridImage( context: Context, grid: Array<BooleanArray>) {
    val bitmap = createGridBitmap(grid)

    val cachePath = File( parent = context.cacheDir, child = "images" )
    cachePath.mkdirs()
    val file = File( parent = cachePath, child = "game_of_life_grid.png")
    FileOutputStream(file).use { out ->
        bitmap.compress( format = Bitmap.CompressFormat.PNG, quality = 100, stream = out )
    }

    val uri = FileProvider.getUriForFile(
        context,
        authority = "${context.packageName}.fileprovider",
        file
    )

    val shareIntent = Intent( action = Intent.ACTION_SEND).apply {
        type = "image/png"
        putExtra( name = Intent.EXTRA_STREAM, value = uri)
        addFlags( flags = Intent.FLAG_GRANT_READ_URI_PERMISSION)
    }

    context.startActivity(Intent.createChooser( target = shareIntent, title = "Share Game of Life grid"))
}
```

The App: Sharing Feature

DEMO

Q&A