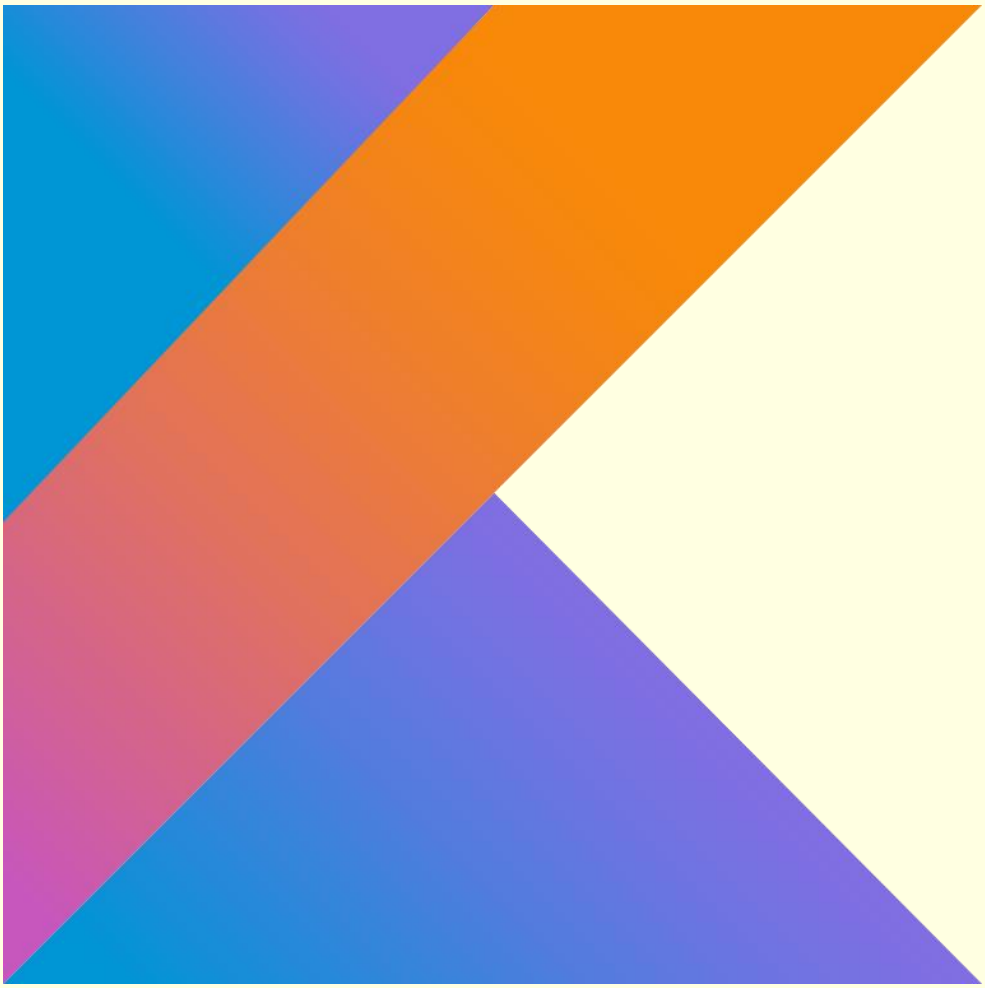
MARCELO SÁNCHEZ MACÍAS



LIBRE CONFIGURACIÓN

MARCELO SÁNCHEZ MACÍAS
IES RAFAEL ALBERTI

7.1 Mejora el juego

Relación 7.3 Lectura y escritura a fichero (RAx.y)

- Realiza mínimo 4 cambios

  -He añadido un atributo de vidas a la nave que disminuirán al impactar con el asteroide reposicionando a esta

```
class ShipData : GameObject() {
    override var size: Double = 40.0
    var visualAngle: Double = 0.0
    var vidas=3
    fun fire(game: Game) {
        val ship = this
        game.gameObjects.add(BulletData( speed: ship.speed * 4.0, ship.visualAngle, ship.position))
    }
}
```

```
fun startGame() {
    gameObjects.clear()
    ship.vidas=3
```

```
if(asteroids.any { asteroid -> ship.overlapsWith(asteroid)}){
    ship.vidas--
    gameLifes="  vidas: "+ship.vidas
    ship.position=Vector2( x: 700.0, y: 700.0)

    if(ship.vidas<=0) {
        endGame()
    }
}
```

  -He añadido un botón mas para el modo difícil que genera más asteroides y cambia el valor de las vidas, además de un tipo de asteroide diferente que se divide hasta 3 veces

```
@Composable
fun Asteroide(AsteroideData: AsteroideData) {
    val asteroidSize = AsteroideData.size.dp
    Box(
        Modifier
            .offset(AsteroideData.xOffset, AsteroideData.yOffset)
            .size(asteroidSize)
            .rotate(AsteroideData.angle.toFloat())
            .clip(CircleShape)
            .background(Color( red: 153,  green: 102,  blue: 102))
    )
}
```

```
gameObjects.add(AsteroideData().apply {  this: AsteroideData
    position = Vector2( x: 100.0,  y: 400.0); angle = Random.nextDouble() * 360.0; speed = 2.0})
```

```
class AsteroideData(speed: Double = 0.0, angle: Double = 0.0, position: Vector2 = Vector2.ZERO) :
    GameObject(speed, angle, position) {
    override var size: Double = 200.0
}
```

-Puntuación que se actualiza al derribar asteroides, para que esta modificación tenga consistencia con el juego he eliminado la función winGame para que sea un bucle infinito que no deja de generar asteroides y así puedas obtener la puntuación que quieras

```
asteroids.forEach { asteroid ->
    val least = bullets.firstOrNull { it.overlapsWith(asteroid) } ?: return@forEach
    if (asteroid.position.distanceTo(least.position) < asteroid.size) {
        gameObjects.remove(asteroid)
        gameObjects.remove(least)
        puntos++
        if (asteroid.size < 50.0) return@forEach
        // it's still pretty big, let's spawn some smaller ones
        repeat( times: 2) {  it: Int
            gameObjects.add(AsteroidData( speed: asteroid.speed * 2,
                 angle: Random.nextDouble() * 360.0,
                asteroid.position).apply {  this: AsteroidData
                size = asteroid.size / 2
            })
        }
    }
}
```

```
if (asteroides.isEmpty() && asteroids.isEmpty()) {
    gameObjects.add(AsteroideData().apply {  this: AsteroideData
        position = Vector2( x: 100.0,  y: 400.0); angle = Random.nextDouble() * 360.0; speed = 2.0})
    repeat(repeat){  it: Int
    gameObjects.add(AsteroidData().apply {  this: AsteroidData
        position = Vector2( x: 100.0,  y: 400.0); angle = Random.nextDouble() * 360.0; speed = 2.0})}

}
```

-Sistema de recompensas por puntuación, que aumentará tanto el numero de balas que puedes disparar, como el número de asteroides generados

```
if (puntos>=50){
    balas=4
    repeat=4
}
if (puntos>=75){
    balas=5
    repeat=5
}
val bullets = gameObjects.filterIsInstance<BulletData>()

// Limit number of bullets at the same time
if (bullets.count() > balas) {
    gameObjects.remove(bullets.first())
}
```

- Utilizar clases abstractas e interfaces
- Respuestas a inputs
- Respuestas mediantes cambios visuales
- Cambios relacionados con gestión de eventos