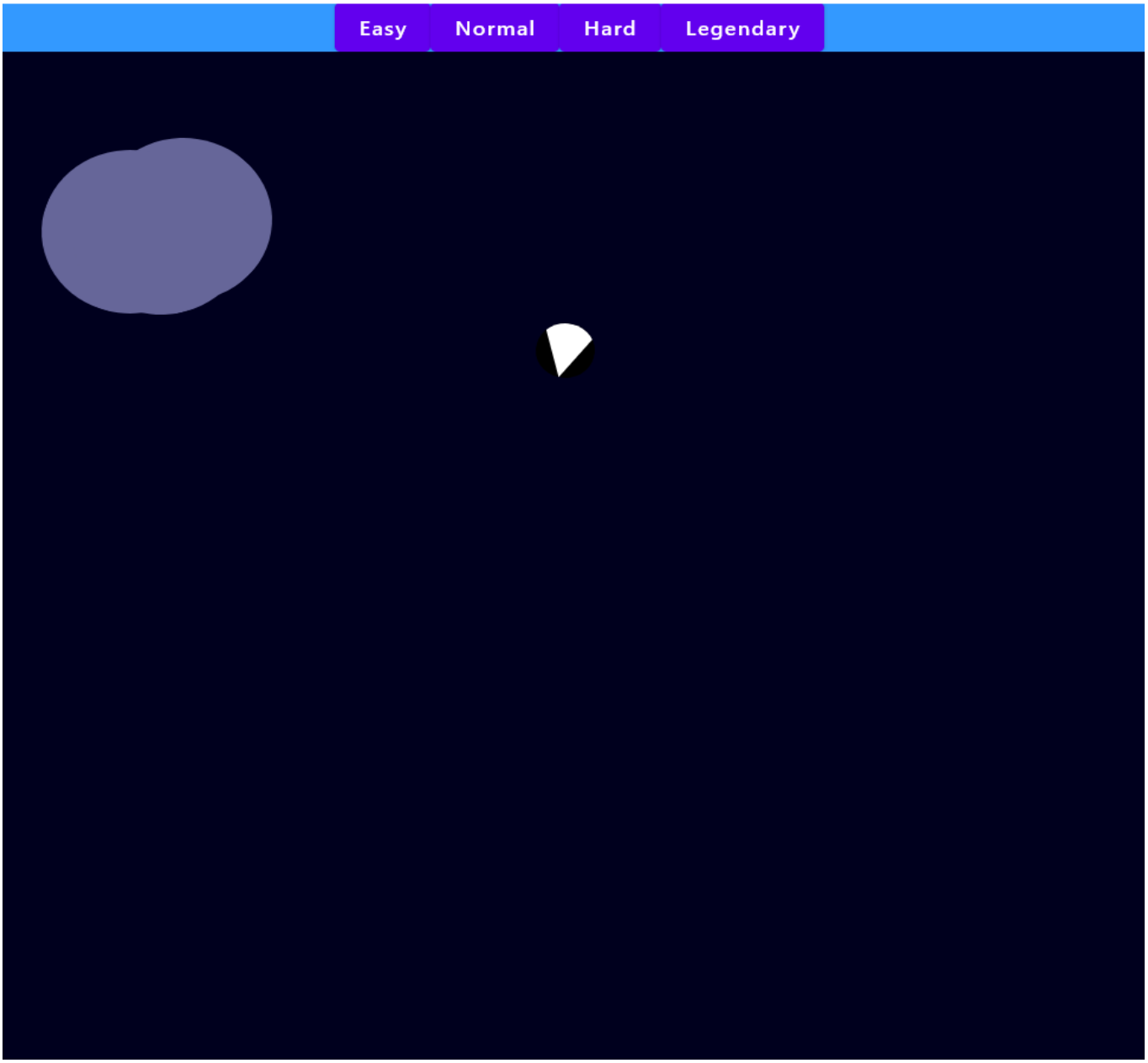
Documentación



Índice

Cambio 1: Dificultad en la velocidad

Respuesta de eventos. Respuesta visual. Reacción a input.

Clase Main

```
Column(modifier = Modifier.background(Color( red: 51, green: 153, blue: 255)).fillMaxHeight()) { this: ColumnScope
    Row(modifier = Modifier.align(Alignment.CenterHorizontally)) { this: RowScope
        Button({
            game.startGame( dificultad: 1)
        }) { this: RowScope
            Text( text: "Easy")
        }
        Button({
            game.startGame( dificultad: 2)
        }) { this: RowScope
            Text( text: "Normal")
        }
        Button({
            game.startGame( dificultad: 3)
        }) { this: RowScope
            Text( text: "Hard")
        }
        Button({
            game.startGame( dificultad: 4)
        }) { this: RowScope
            Text( text: "Legendary")
        }
    }
```

Se ha añadido 4 botones, cada uno es una dificultad distinta que al ser pulsado llama al método del game dando de parámetro el valor que inicia una dificultad u otra.

Clase GameObject

```
abstract val size: Double // Diameter
fun update(realDelta: Float, game: Game) {
    val obj = this
    val velocity = movementVector * realDelta.toDouble()*0.5
    obj.position += velocity
    obj.position = obj.position.mod(Vector2(game.width.value.toDouble(), game.height.value.toDouble()))
}
fun update2(realDelta: Float, game: Game) {
    val obj = this
    val velocity = movementVector * realDelta.toDouble()
    obj.position += velocity
    obj.position = obj.position.mod(Vector2(game.width.value.toDouble(), game.height.value.toDouble()))
}
fun update3(realDelta: Float, game: Game) {
    val obj = this
    val velocity = movementVector * realDelta.toDouble()*12.0
    obj.position += velocity
    obj.position = obj.position.mod(Vector2(game.width.value.toDouble(), game.height.value.toDouble()))
}

fun update4(realDelta: Float, game: Game) {
    val obj = this
    val velocity = movementVector * realDelta.toDouble()*30.0
    obj.position += velocity
    obj.position = obj.position.mod(Vector2(game.width.value.toDouble(), game.height.value.toDouble()))
}
```

Se ha definido 4 métodos, cada uno corresponde a una velocidad distinta para los elementos del juego.

Clase Game

```
class Game() {
    var prevTime = 0L
    val ship = ShipData()
    var dificultad= 0
    var targetLocation by mutableStateOf<DpOffset>(DpOffset.Zero)
```

Se establece a la clase una variable llamada dificultad.

```
fun startGame(dificultad: Int) {
    this.dificultad = dificultad
    gameObjects.clear()
```

El método que recibe de game en la clase Main, hace que se guarde el valor en la variable dificultad.

```
fun update(time: Long) {
    val delta = time - prevTime
    val floatDelta = (delta / 1E8).toFloat()
    prevTime = time

    if (gameState == GameState.STOPPED) return

    val cursorVector = Vector2(targetLocation.x.value.toDouble(), targetLocation.y.value.toDouble())
    val shipToCursor = cursorVector - ship.position
    val angle = atan2(y = shipToCursor.y, x = shipToCursor.x)

    ship.visualAngle = shipToCursor.angle()
    ship.movementVector = ship.movementVector + (shipToCursor.normalized * floatDelta.toDouble())

    for (gameObject in gameObjects) {
        if(dificultad==1){
            gameObject.update(floatDelta,  game: this)
        }
        else if (dificultad==2){
            gameObject.update2(floatDelta,  game: this)
        }
        else if (dificultad==3){
            gameObject.update3(floatDelta,  game: this)
        }
        else{
            gameObject.update4(floatDelta,  game: this)
        }
```

Según el valor que le corresponda a dificultad, se aplicará un método u otro de los estabecidos en GameObject, el cual cada uno corresponde a una velocidad distinta. Valor 1, corresponde a Fácil. Valor 2, corresponde a Normal. Valor 3, corresponde a Difícil, y valor 4 corresponde a Legendario.

Cambio 2: Contador de muertes

Respuesta de eventos. Respuesta visual.

Clase Main

```
    Row(modifier = Modifier.align(Alignment.CenterHorizontally)) {  this: RowScope
        Text(game.gameStatus, modifier = Modifier.align(Alignment.CenterVertically).padding(horizontal = 16.dp), color = Color.
        Text( text: "Contador de muertes: "+game.contador, modifier = Modifier.align(Alignment.CenterVertically).padding(horizont

    }
```

Se ha añadido un nuevo campo de texto, donde el contador se irá actualizando, según se ejecute la función EndGame()

Clase Game

```
class Game() {
    var prevTime = 0L
    val ship = ShipData()
    var dificultad= 0
    var targetLocation by mutableStateOf<DpOffset>(DpOffset.Zero)

    var gameObjects = mutableStateListOf<GameObject>()
    var gameState by mutableStateOf(GameState.RUNNING)
    var gameStatus by mutableStateOf( value: "Let's play!")
    var contador = 0
```

En la clase Game se ha implementado el contador.

```
    fun endGame() {
        gameObjects.remove(ship)
        gameState = GameState.STOPPED
        gameStatus = "Better luck next time!"
        contador++
    }
```

Cada vez que se llame la función EndGame(), se incrementará a uno el contador.

Cambio 3: Disparador de balas múltiples

Respuesta de eventos. Respuesta visual. Reacción a input.
Se ha permitido que la nave pueda disparar ráfagas de balas, puede variar entre una bala a 3, mediante un número aleatorio que aparecerá por cada disparo y según el número que pertenezca dicha variable hará un tipo de disparo u otro.

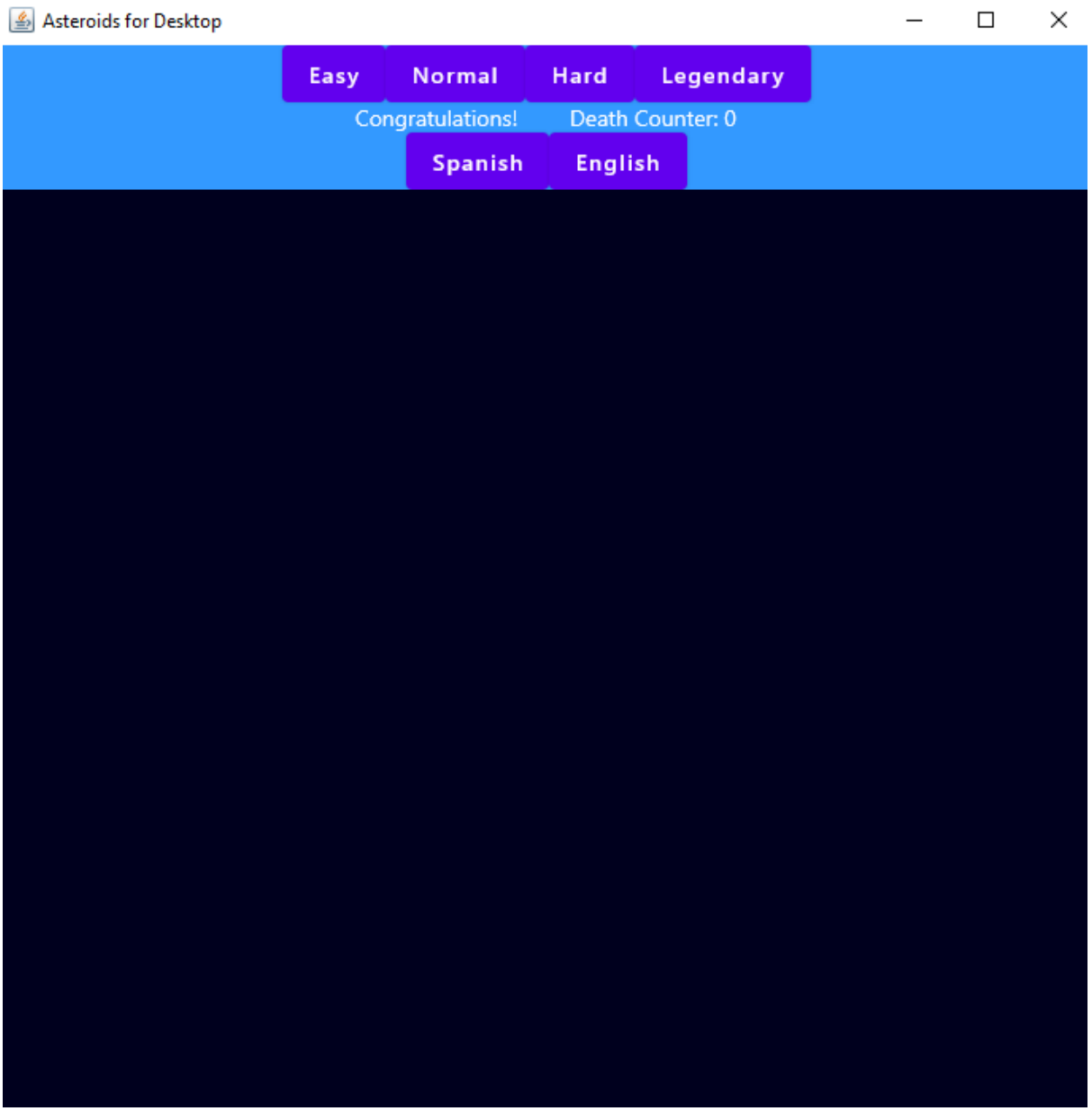
```
class ShipData : GameObject() {
    override var size: Double = 40.0
    var visualAngle: Double = 0.0

    fun fire(game: Game) {
        val ship = this
        val rnds = (1 ≤ .. ≤ 3).random()
        if(rnds==1){
            game.gameObjects.add(BulletData( speed: ship.speed * 4.0, ship.visualAngle,  position: ship.position*0.9))
        }
        if(rnds==2){
            game.gameObjects.add(BulletData( speed: ship.speed * 4.0, ship.visualAngle,  position: ship.position*0.9))
            game.gameObjects.add(BulletData( speed: ship.speed * 4.0, ship.visualAngle, ship.position))
        }
        if(rnds==3){
            game.gameObjects.add(BulletData( speed: ship.speed * 4.0, ship.visualAngle,  position: ship.position*0.9))
            game.gameObjects.add(BulletData( speed: ship.speed * 4.0, ship.visualAngle, ship.position))
            game.gameObjects.add(BulletData( speed: ship.speed * 4.0, ship.visualAngle,  position: ship.position*1.1))
        }
    }
}
```

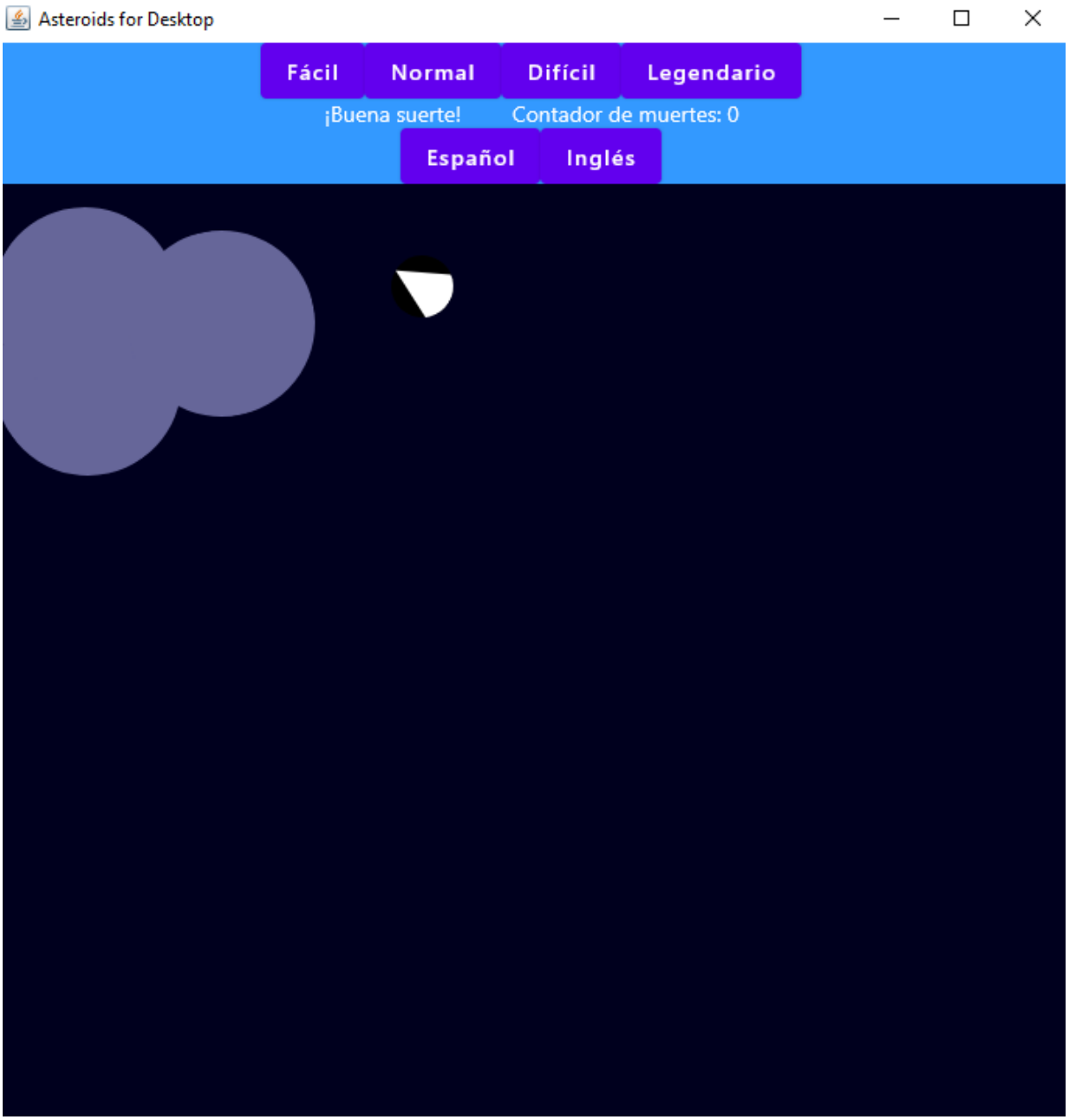
Aquí se ha permitido que el máximo de balas por pantalla sea 9 para permitir disparar más de una vez ráfagas de disparos.

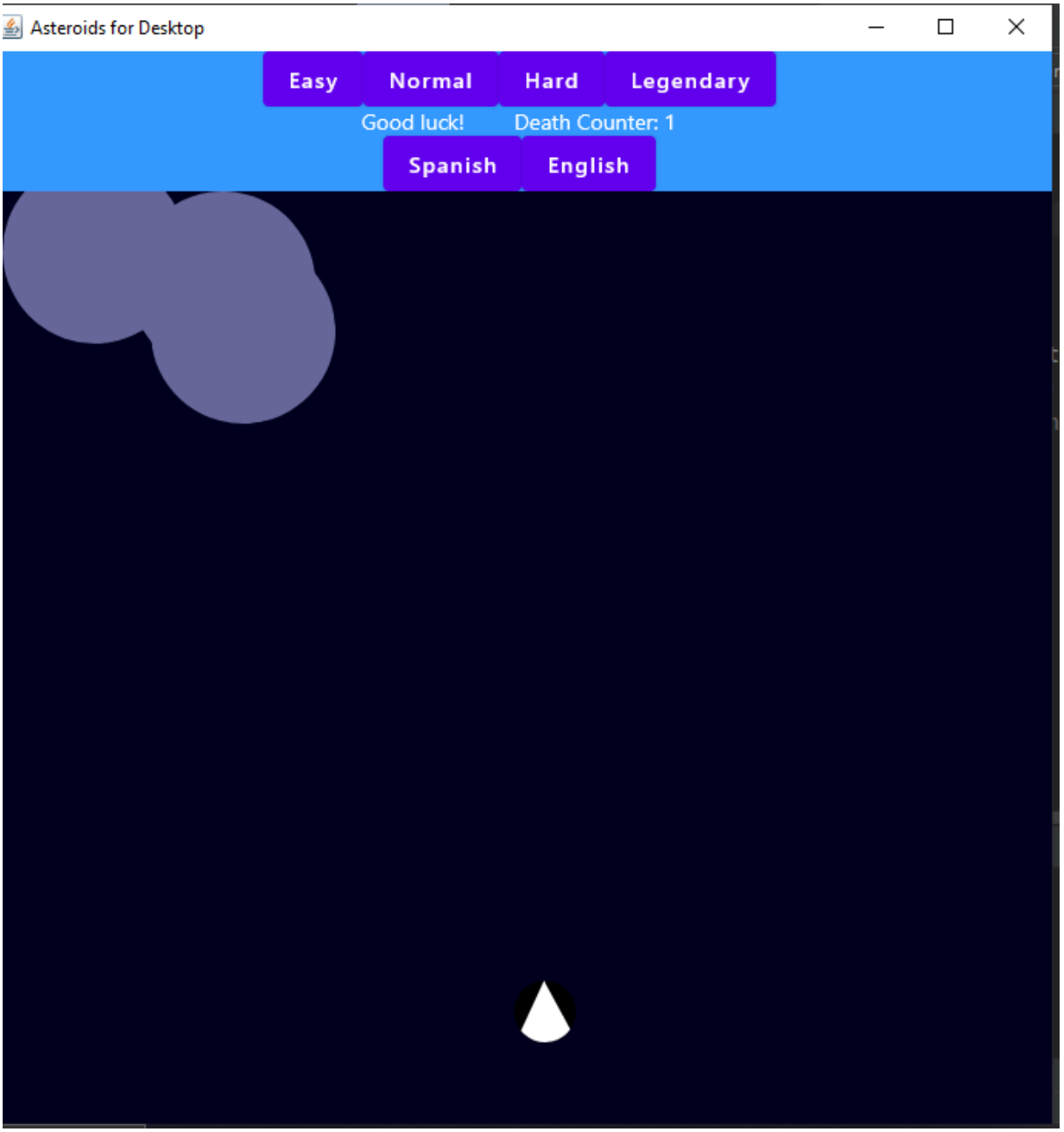
```
        // Limit number of bullets at the same time
        if (bullets.count() > 9) {
            gameObjects.remove(bullets.first())
        }
        val asteroids = gameObjects.filterIsInstance<AsteroidData>()
```

Cambio 4: Selector de idiomas

Utilización de clases abstractas e interfaces. Respuesta de eventos. Respuesta visual. Reacción a input.
Hemos realizado lo siguiente:


Asteroids for Desktop
Easy
Normal
Hard
Legendary
Congratulations!
Death Counter: 0
Spanish
English


Asteroids for Desktop
Fácil
Normal
Difícil
Legendario
¡Buena suerte!
Contador de muertes: 0
Español
Inglés



Comienza el juego en inglés, y según el idioma que elija, será cambiado por el elegido. Los cambios que se han realizado para llevar a cabo esto son los siguientes:

Clase Resultado

```
interface Resultado{
    val victoria: String
    val fracaso: String
    val suerte: String
    val jugar: String
    val d1: String
    val d2: String
    val d3: String
    val d4: String
    val i1: String
    val i2: String
    val cont: String
}
```

Creación de la interfaz resultado.

```
open class ResultadoEN(): Resultado{
    override val victoria: String= "Congratulations!"
    override val fracaso: String= "Better luck next time!"
    override val suerte: String = "Good luck!"
    override val jugar: String = "Let's play!"
    override val d1: String = "Easy"
    override val d2: String = "Normal"
    override val d3: String = "Hard"
    override val d4: String = "Legendary"
    override val i1: String = "Spanish"
    override val i2: String = "English"
    override val cont: String = "Death Counter"
}
open class ResultadoESP(): Resultado{

    override val victoria: String= "Victoria!"
    override val fracaso: String= "¡Más suerte para la próxima!"
    override val suerte: String = "¡Buena suerte!"
    override val jugar: String = "¡Jugar!"
    override val d1: String = "Fácil"
    override val d2: String = "Normal"
    override val d3: String = "Difícil"
    override val d4: String = "Legendario"
    override val i1: String = "Español"
    override val i2: String = "Inglés"
```

Creación de las clases abstractas donde se define el valor que tendrá cada string según cuál idioma.

Clase Game

```
class Game(var idioma: Resultado){
    var prevTime = 0L
    val ship = ShipData()
    var dificultad= 0
    var targetLocation by mutableStateOf<DpOffset>(DpOffset.Zero)

    var gameObjects = mutableStateListOf<GameObject>()
    var gameState by mutableStateOf(GameState.RUNNING)
    var gameStatus by mutableStateOf(idioma.jugar)
    var contador = 0
    var contador2 = 0
    fun changeIdioma(idioma: Resultado){
        this.idioma= idioma
    }
```

Se ha establecido que en la creación de la instancia juego se elija ya el idioma que tendrá, pero también se le ha establecido un método para cambiar de idioma ya una vez iniciada la instancia.

```
    fun endGame() {
        gameObjects.remove(ship)
        gameState = GameState.STOPPED
        gameStatus = idioma.fracaso
        contador++
    }

    fun winGame() {
        contador2++
        gameState = GameState.STOPPED
        gameStatus = idioma.victoria
    }
```

```
    var gameState by mutableStateOf(GameState.RUNNING)
    var gameStatus by mutableStateOf(idioma.jugar)
```

```
    }
    gameState = GameState.RUNNING
    gameStatus =idioma.suerte
}
```

Se ha establecido que los valores de texto de fracaso, victoria, comenzar el juego y el deseo de buena suerte para el jugador, serán definidos según el idioma que se ha establecido, el string guardado que le correspondería.

Clase Main

```
fun main() = Window(size = IntSize( width: 800,  height: 900), title = "Asteroids for Desktop") {
    val esp = ResultadoESP()
    val en = ResultadoEN()
    var game = remember { Game(en) }
```

Se han establecido las variables de los idiomas español, e inglés, haciendo referencia a las clases abstractas que heredan de la interfaz, y por defecto se iniciará la instancia de juego en inglés.

```
    Column(modifier = Modifier.background(Color( red: 51,  green: 153,  blue: 255)).fillMaxHeight()) {  this: ColumnScope
        Row(modifier = Modifier.align(Alignment.CenterHorizontally)) {  this: RowScope
            Button({
                game.startGame( dificultad: 1)
            }) {  this: RowScope
                Text(game.idioma.d1)
            }
            Button({
                game.startGame( dificultad: 2)
            }) {  this: RowScope
                Text(game.idioma.d2)
            }
            Button({
                game.startGame( dificultad: 3)
            }) {  this: RowScope
                Text(game.idioma.d3)
            }
            Button({
                game.startGame( dificultad: 4)
            }) {  this: RowScope
                Text(game.idioma.d4)
            }

        }
```

Se han establecido los textos de la dificultad según el idioma que se encuentra instanciado el juego en dicho momento.

```
    Row(modifier = Modifier.align(Alignment.CenterHorizontally)) {  this: RowScope
        Text(game.gameStatus, modifier = Modifier.align(Alignment.CenterVertically).padding(horizontal = 16.d
        Text( text: game.idioma.cont+": "+game.contador, modifier = Modifier.align(Alignment.CenterVertically).p
        //Text("Score: "+game.contador, modifier = Modifier.align(Alignment.CenterVertically).padding(horizon
    }
```

Se ha establecido el idioma del contador de muertes.

Se ha creado una tercera columna donde aparecerán dos botones, el primero será el español, y el segundo será el inglés, los textos de los idiomas serán dependiendo del idioma del juego, una vez pulsado el botón llamará al método changeIdioma que establecerá el idioma en cuestión, y se comenzará un juego para reiniciar la interfaz y aparezca el nuevo idioma.