Formato de escenarios y casos de uso

Configuración de los Escenarios

Nombre	Clase	Escenario
setupStage1	MovementControllerTest	MapLoader map = MapLoader.of("llanuras.map"); Player p = new Player(2,2); MovementController mc = new MovementController(map,p); // (1,2) libre, (2,1) pared
setupStage2	GatewaysTest	MapLoader plains = MapLoader.of("llanuras.map"); MapLoader mountains = MapLoader.of("montanas.map"); SceneState state = new SceneState(vida=3, municionRifle=5, municionRevolver=10, invCap=5); Gateway g = new Gateway(plains,mountains, at(9,0));
setupStage3	AmmoManagerTest	AmmoManager ammo = new AmmoManager(rifle=1, revolver=0); FireControl fire = new FireControl("rifle", ammo);
setupStage4	ReloadingTest	AmmoManager ammo = new AmmoManager(rifle=0, revolver=6); Reloading re = new Reloading(auto=false, tRecargaMs=2000); FireControl fire = new FireControl("revolver", ammo, re);
setupStage5	InventoryTest	Inventory inv = new Inventory(capacity=3); Item comida = Item.food(1); Item cura = Item.medkit(1); Item mun = Item.ammo("rifle",3); inv.add(comida); inv.add(cura); // 1 slot libre
setupStage6	SpawnerTest	MapLoader map = MapLoader.of("montanas.map"); Spawner sp = new Spawner(densidadMax=5, radioSeguro=3, seed=42); Player p = new Player(10,10);
setupStage7	EnemyAiTest	EnemyAI ai = new EnemyAI(rangoVision=8); EnemyAttack atk = new EnemyAttack(danio=1, cadenciaMs=800); Player p = new Player(5,5, vida=3); Enemy e = new Enemy(0,0);
setupStage8	AchievementsBSTTest	AchievementsBST bst = new AchievementsBST(); Achievement a1 = new Achievement("Primer disparo", 10); Achievement a2 = new Achievement("Sobrevive Montañas", 30);
setupStage9	AimReticleTest	AimReticle ret = new AimReticle(); FireControl fire = new FireControl("rifle"); Mouse.setPos(100, 200);

* El nombre de los escenarios puede ser setupStage1, setupStage2, etc.
* La clase es la clase de testing correspondiente al modelo donde acontece el escenario. Por ejemplo si usted está probando User, clase será UserTest.
* El escenario es la descripción de las condiciones iniciales del escenario.

Diseño de Casos de Prueba

Objetivo de la Prueba: Validar movimiento en 4 direcciones y colisiones (no atraviesa paredes, se mueve en celdas transitables)				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado esperado**
MovementController	moveUp()	setupStage1	posIni=(2,2), celda(2,1)=pared	posición final=(2,2) (bloqueado)
MovementController	moveRight()	setupStage1	posIni=(2,2), celda(3,2)=libre	posición final=(3,2)
MovementController	moveDown	setupStage1	posIni=(3,2), celda(3,3)=libre	posición final=(3,3)

Objetivo de la Prueba: Transición entre escenarios preservando estado y bloqueando saltos indebidos				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado esperado**
Gateways	transferIfCollides()	setupStage2	playerPos=portal(9,0), state(vida=3, muniR=5, muniRev=10)	cambia a “montanas.map” y mantiene vida=3, muniR=5, muniRev=10
Gateways	transferIfCollides()	setupStage2	intento salto “llanuras”→“rio” (no adyacente)	transición denegada

Objetivo de la Prueba: Munición como recurso escaso y disparo bloqueado con 0				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado esperado**
AmmoManager	shoot()	setupStage3	arma="rifle", ammo.rifle=1	dispara OK y queda ammo.rifle=0
AmmoManager	shoot()	setupStage3	arma="rifle", ammo.rifle=0	no dispara; HUD notifica “Sin munición”

Objetivo de la Prueba: Recarga manual/auto bloquea disparo hasta completar				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado esperado**
Reloading	reload()	setupStage4	arma="revolver", ammo=0, auto=false	durante recarga, shoot() retorna false; al completar, ammo>0
Reloading	autoReloadIfNeeded()	setupStage4	arma="revolver", auto=true, ammo=0	inicia recarga automática y bloquea disparo hasta terminar

Objetivo de la Prueba: Inventario limitado, prioridad a esenciales y gestión cuando está lleno				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado esperado**
Inventory	add()	setupStage5	inv con 2/3 llenos; add munición	add exitoso (3/3)
Inventory	add()	setupStage5	inv 3/3; add comida	requiere gestión (descartar) y luego agrega
Inventory	use()	setupStage5	usar medkit	vida +1 y se reduce el item

Objetivo de la Prueba: Recolección de recursos respeta capacidad y puntos de spawn				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado esperado**
LootPickup	pickup()	setupStage5	inv 2/3; recurso=munición en celda	pickup OK, inv 3/3
LootPickup	pickup()	setupStage5	inv 3/3; recurso=medkit	rechazo o gestión (no sobrepasa capacidad)

Objetivo de la Prueba: Spawner limita densidad y respeta zona segura				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado esperado**
Spawner	tickSpawn()	setupStage6	densidadMax=5, ya hay 5 enemigos	no spawnea nuevos
Spawner	tickSpawn()	setupStage6	player=(10,10), radioSeguro=3	no spawnea dentro del radio seguro

Objetivo de la Prueba: Enemigo persigue y ataca; derrota con vida=0				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado esperado**
EnemyAI	update()	setupStage7	dist(player,enemy)=6, rango=8	estado=PERSEGUIR y reduce distancia
EnemyAttack	tryAttack()	setupStage7	en rango, daño=1, vida=3	vida del jugador pasa a 2

Objetivo de la Prueba: Árbol de logros (ABB) inserta/busca y vista independiente				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado esperado**
AchievementsBST	insert()	setupStage8	insertar a1(score 10), a2(score 30)	in-order = [a1, a2]

AchievementsBST	search()	setupStage8	buscar score=30	devuelve a2
AchievementsBST	openWindow()	setupStage8	bst con N logros	abre ventana con representación del Árbol de búsqueda

Objetivo de la Prueba: Retícula de puntería y orientación del arma hacia el mouse				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado esperado**
AimReticle	update(mousePos)	setupStage9	mouse=(100,200)	retícula en (100,200)
AimReticle	aimAt(reticula)	setupStage9	retícula=(100,200)	arma orientada al ángulo

* Una prueba se compone de un conjunto de casos de prueba.
* Cada fila representa un ***caso de prueba*** diferente
* En el objetivo de la prueba debe escribir una descripción sobre qué es lo que específicamente está probando del modelo del programa.
* La clase es la clase del modelo que está siendo puesto a prueba.
* El método es específicamente el método de la clase que está siendo puesto a prueba.
* El escenario se refiere al nombre del escenario que usted definió. Todos los casos de prueba corresponden a escenarios.
* Los valores de entrada son valores que entran al método puesto a prueba.
* El resultado esperado es lo que se espera que suceda luego de ejecutar el método.