Óscar Madrigal Cárdenas NIU : 1566337

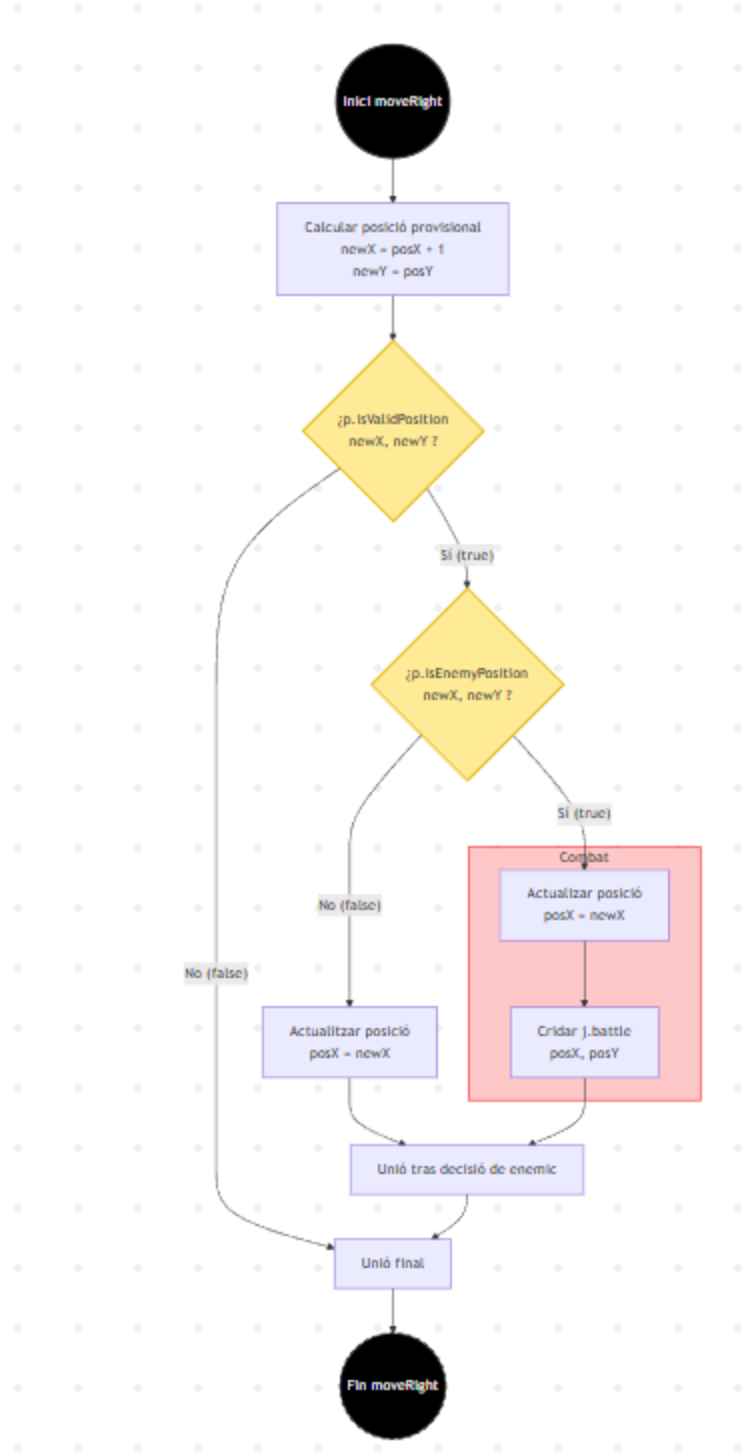
Testing elaborat: Masmorra_TQS

Classe Jugador

Classe que controla tota la lógica relacionada al jugador; moviment, estadístiques posicions etc. A aquesta classe he testejat els metodes moveRight, *moveLeft*, *moveUp* i *moveDown*, a més de *setHp*, *setEXP*, *openDoor*, *isPlayerAtDoor* i *lvUp*.

moveRight, moveLeft, moveUp, moveDown: fent us de *MockObjects* per les classes planta i Joc he testejat els diferents camins possibles(decision coverage, path coverage, statement coverage) alhora de moure's en el joc.

```
public void moveRight(Planta p, Joc j) {
 int newX = posX + 1; // movimiento provisional
 int newY = posY;

 if (p.isValidPosition(newX, newY)) {
   if (p.isEnemyPosition(newX, newY)) {
     posX = newX;
     j.battle(posX, posY);
   } else {
     posX = newX;
   } // solo movemos si es válido
 }
}
```



setHp, setEXP: testeig de valors límits i de frontera per comprovar que no poden tomar valors negatius, o sobrepassar els valors màxims de salut i experiencia.

openDoor: testeig simple de que es pot obrir la porta

isPlayerAtDoor: comprovació condition i decision coverage de que el jugador es troba a la mateixa casella de la porta.

```
public boolean isPlayerAtDoor(Planta p) {
 if (p.getDoorposX() == this.posX && p.getDoorposY() ==
this.posY) {
   return true;
 } else {
   return false;
 }
}
```

lvUp: comprovació de que els valors incrementen correctament al pujar de nivell i proves de caixa blanca per al redondeig correcte.

Classe Planta

getEnemy, isEnemyPosition i isDoorPosition: comprovació de casos limits i normals i condicions i decisions; quan l'enemic/porta es troba correctament i quan no.

- getEnemy: enemic trobat
- getEnemy: enemic no trobat
- isEnemyPosition: llista buida
- isEnemyPosition: cas positiu
- isEnemyPosition: cas negatiu
- isEnemyPosition: posicions negatives o molt grans
- isDoorPosition: cas positiu
- isDoorPosition: cas negatiu, posicions limits i frontera

isValidPosition: testeig de valors límit i fora de limit i de les diferents condicions.

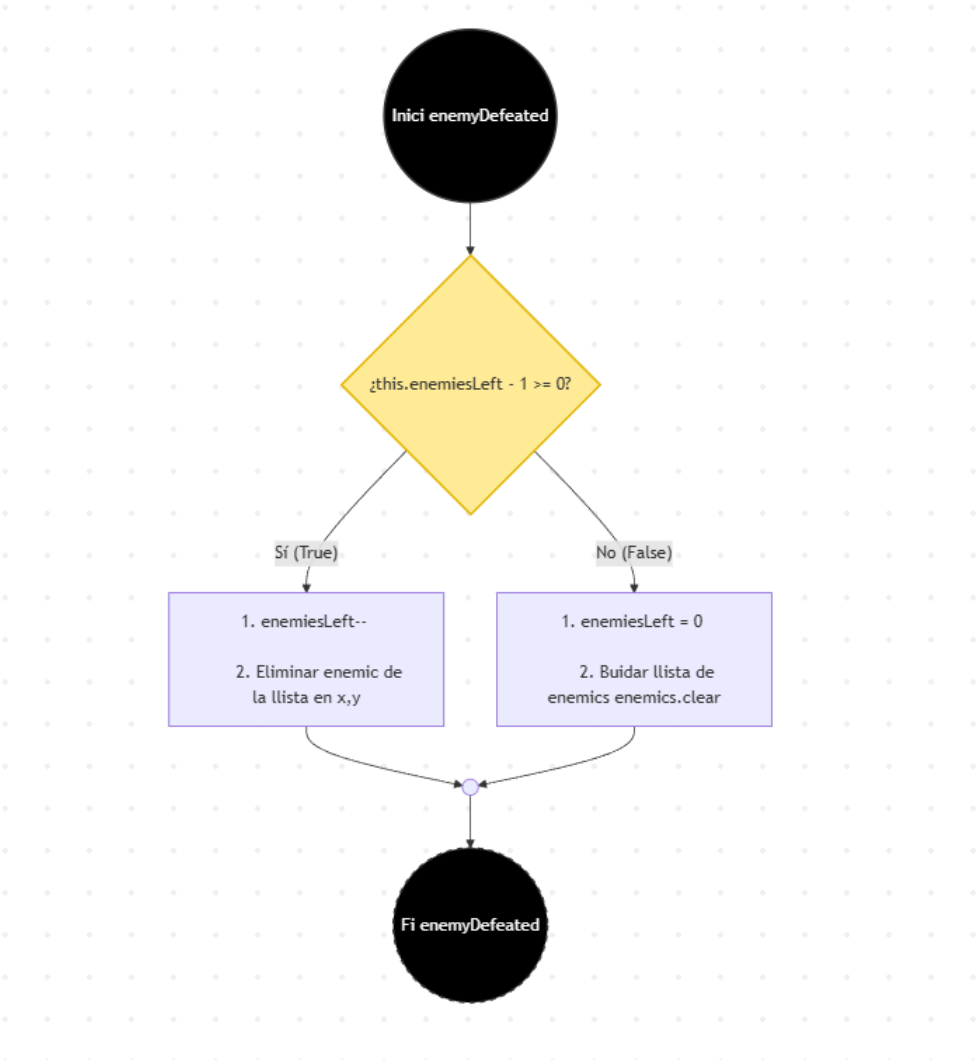
- Nombre de files del layout.
- Nombre de columnes del layout.
- Cas positiu
- Cas habitual: mur no deixa passar
- Cas inusual: valors fora de rang
- Limits i fronteres

```
public boolean isValidPosition(int x, int y) {
 int limX = 79;
 int limY = 19;
 return x >= 0 && x <= limX
         && y >= 0
         && y <= limY
         && this.floorLayout[y].charAt(x) != '#';
}
```

enemyDefeated: testeig de casos i path coverage.

- Cas normal: eliminar enemic.
- Cas normal: eliminar enemic a posicio intermitja de l'array.
- Eliminar últim enemic.
- Eliminacions consecutives.

```
public void enemyDefeated(int x, int y) {
 if (this.enemiesLeft - 1 >= 0) {
   this.enemiesLeft--;
   for (int i = 0; i < this.enemics.size(); i++) {
     if (this.enemics.get(i).getPos_x() == x &&
this.enemics.get(i).getPos_y() == y) {
       this.enemics.remove(i);
     }
   }
 } else {
   this.enemiesLeft = 0;
   this.enemics.clear();
 }
}
```



Classe Poció

usarItem: testeig de casos habituals i límits. Comprovació d'augment de salut i de múltiples crides.

Classe Joc

Classe encarregada de inicialitzar el joc i de gestionar tota la lógica entre diferents elements d'aquest.

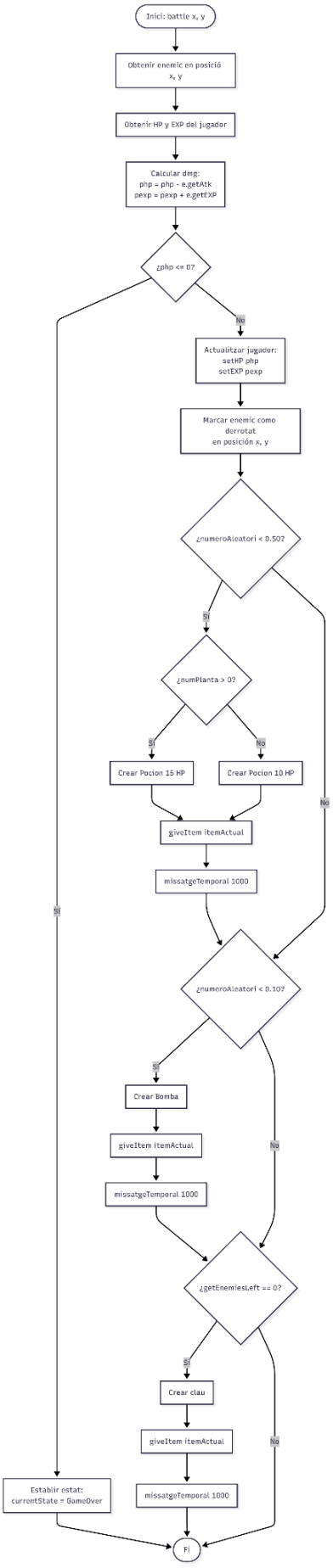
startGame, missatgeTemporal: comprovació de valors d'inicialització adequats i de múltiples trucades.

battle: path coverage de les diferents situacions possibles al combatir, a més de testeig de diferents valors limits a les batalles i de l'obtenció d'items després d'aquestes.

- Jugador guanya
- Jugador perd
- Jugador perd exactament
- Enemic s'elimina correctament
- Drop de clau
- Drop de poció
- Drop de bomba
- Drop de poció i bomba

```
public void battle(int x, int y) {
 Enemic e = this.plantaActual.getEnemy(x, y);

 int php = this.player.getHP();
 int pexp = this.player.getEXP();

 php = php - e.getAtk();
 pexp = pexp + e.getEXP();

 if (php <= 0) {
   currentState = GameState.gameOver;
 } else {
   this.player.setHP(php);
   this.player.setEXP(pexp);
   this.plantaActual.enemyDefeated(x, y);

   //Drop de pocio
   if (numeroAleatori() < 0.50) {
     if(this.numPlanta > 0){
       this.itemActual = new Pocion(this.player,15);
     }else{
       this.itemActual = new Pocion(this.player,10);
     }
     giveItem(itemActual);
     missatgeTemporal(1000);
   }
```

```
    //Drop de bomba
    if(numeroAleatori() < 0.10 ){
      this.itemActual = new Bomba(this.player);
      giveItem(itemActual);
      missatgeTemporal(1000);
    }

    if (this.plantaActual.getEnemiesLeft() == 0) {
      this.itemActual = new Llave(this.player);
      giveItem(this.itemActual);
      missatgeTemporal(1000);
    }
  }
}
```


Inici: battle x, y
Obtenir enemic en posició x, y
Obtenir HP y EXP del jugador
Calcular dmg:
php = php - e.getAtk
pexp = pexp + e.getEXP
¿php <= 0?
No
Actualitzar jugador:
setHP php
setEXP pexp
Marcar enemic como derrotat en posición x, y
¿numeroAleatori < 0.50?
Sí
¿numPlanta > 0?
Sí
No
Crear Pocion 15 HP
Crear Pocion 10 HP
No
giveItem itemActual
missatgeTemporal 1000
¿numeroAleatori < 0.10?
Sí
Crear Bomba
giveItem itemActual
missatgeTemporal 1000
No
¿getEnemiesLeft == 0?
Sí
Crear clau
giveItem itemActual
missatgeTemporal 1000
No
Sí
Estabilir estat:
currentState = GameOver
Fi

boom: comprovacions de valors límit i estandars alhora d'eliminar múltiples enemics.

- Cap enemic
- 1 enemic restant
- 2 enemics restants
- 3 enemics restants
- Més de 3 enemics

passarDePlanta: testeig de condicions i modificacio correcte de variables. Finalització del joc.

Classe LoopJoc

Classe que gestiona el bucle principal de joc i que fa de controlador entre la vista (Interface) i el model (Joc). Per dur a terme el testeig necessari del seu únic mètode *startGame* he hagut de crear les funcions *crearPulsacions* i *crearCarac* per simular les pulsacions del teclat

startGame: testeig de les diferents accions possibles depenent de l'estat de joc i de les diferents transicions.