2. beadandó feladat dokumentáció

Készítette: Mikus Márk

Neptun-kód: CM6TSV

Email: [kyussfia@gmail.com](mailto:kyussfia@gmail.com)

# Feladat:

Készítsünk programot, amellyel a következő játékot játszhatjuk. Adott egy n*x*n elemből álló játékpálya, amely labirintusként épül fel, azaz fal, illetve padló mezők találhatóak benne, illetve egy kijárat a jobb felső sarokban. A játék célja az, hogy egy játékost minél előbb kivezessünk a labirintusból a bal alsó sarokból indulva. A labirintusban nincs világítás, csak egy fáklyát visz a játékos, amely a 2 szomszédos mezőt világítja meg (azaz egy 5x5-ös négyzetet), de a falakon nem tud átvilágítani. A játékos figurája kezdetben a bal alsó sarokban helyezkedik el, és vízszintesen, illetve függőlegesen mozoghat (egyesével) a pályán. A pályák méretét, illetve felépítését (falak, padlók) tároljuk fájlban. A programot legalább 3 különböző méretű pályával lehessen használni. A program biztosítson lehetőséget új játék kezdésére a pálya kiválasztásával, valamint játék szüneteltetésére (ekkor nem telik az idő, és nem léphet a játékos), továbbá ismerje fel, ha vége a játéknak. A program játék közben folyamatosan jelezze ki a játékidőt.

# Elemzés:

* A játék során egy olyan felületet jelenítünk, amely három fő részre oszlik: a menüre, egy HUD-ra illetve magára a játéktérre.
* A menüben új játékot kezdhetünk egy választott szinten, illetve kiléphetünk.
* A HUD a játéksorán megtett lépéseket, a játék során eltelt időt, jeleníti meg, továbbá a játék szüneteltetéséhez, szükséges nyomógombot is tartalmazza.
* A játéktéren, egy a szintnek megfelelő méretű gomb-rács foglal helyet, amelyek az egyes szobákat reprezentálják, a falak, a szegélyek lesznek a gombokon. A világítás a gombokon szürke, míg a sötétség esetén fekete háttérszínként jelenik meg.
* A játékhoz tartozó 3 pályát, egy .qrc forrásfájlban egyesíti a project. A forrásfájlok .json formátumúak.
* Ha a játékos célba ér, egy felugró ablakban, közli ezt a játék.
* A játék ablaka, fix szélességű legyen

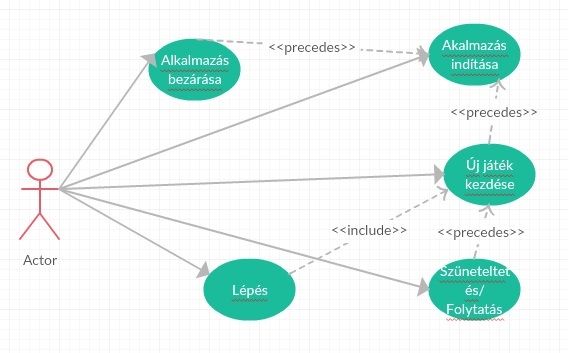
# Tervezés:

A program két szegmensre osztva valósul meg, a modell/nézet architektúrának megfelelően, melyben a modell eseménykezelőkkel kommunikál a nézettel.

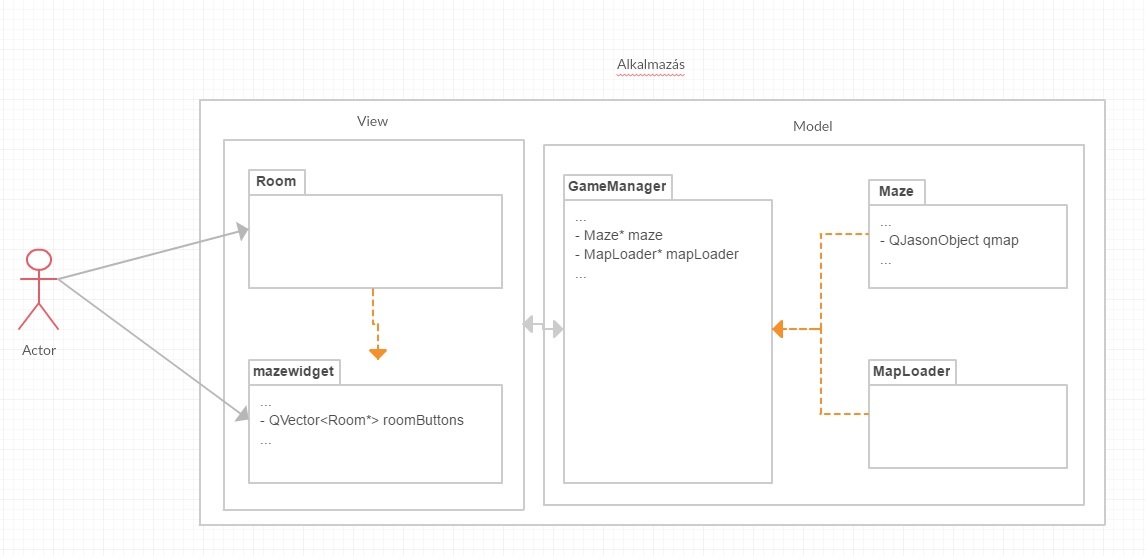
A játék alaplogikáját így egy GameManager osztály fogja végezni, amely egy QObject leszármazott. A játék nézete, az alapwidget, azaz a mazewidget osztály legyen.

A játékhoz szükséges fájlműveleteket, egy GameManager osztályba injektált MapLoader osztály valósítsa, meg. (Ez is lehet QObject) Olyan módon, hogy a GameManager, miután meghatározza a választott szintnek megfelelő fájlnevet, (pl.: map3.json) meghívja a MapLoadert és elkéri tőle a megfelelő pálya információit.

## Használati esetek:



|  |  |  |  |
| --- | --- | --- | --- |
|  | Felhasználói eset | Leírás |  |
| 1 | Alkalmazás indítása | Given: | Az alkalmazás telepítve van, és futtatható. |
| When: | Alkalmazás indulása |
| Then: | Létrejön a model, standard nézet, üres játéktérrel. |
| 2 | Alkalmazás Bezárása | Given: | Az alkalmazás fut. |
| When: | Alkalmazás bezárása. |
| Then: | Bezárul az alkalmazás. |
| 3 | Új játék kezdése | Given: | Az alkalmazás elindult. |
| When: | Új játék gomb megnyomásakor |
| Then: | A kiválasztott szint legenerálódik. |
| 4 | Szüneteltetés/  Folytatás | Given: | Új játék közben vagyunk. |
| When: | A szünet gomb megnyomásakor. |
| Then: | Ha állt a játék, akkor elindul, ha ment, akkor megáll. |
| 5 | Lépés | Given: | Új játék közben vagyunk. |
| When: | Egy kiválasztott szobára (gomb) kattintáskor. |
| Then: | A szabályoknak megfelelő akció: lépés, vagy semmi sem történt, vagy győzelem. |

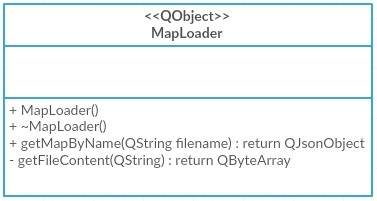
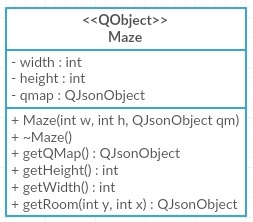
A fenti ábra szemlélteti az egyes objektumok kapcsolatát az alkalmazáson belül, és azt is, hogyan rétegződik az alkalmazás.

## Osztályok:

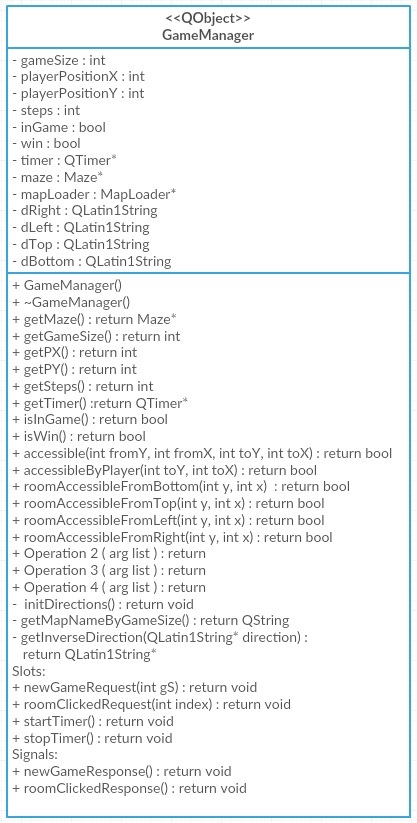
A ***főprogram*** létrehozza a mazewidget-et, amely a nézetet fogja adni. A ***mazewidget*** miután létrejön, létrehoz egy GameManager objektumot, összeköti a megfelelő eseményeket.

Az új játék, gombot, ha megnyomjuk, akkor a nézetnek üzen, s az meghatározza az aktuális nehézségi szintet, majd üzen egy newGameRequest signal formájában a ***GameManagernek*** (GM), amely inicializálja a játék kezdéséhez szükséges információkat: a játékos jelenlegi koordinátája, időzítő indítása, illetve a megfelelő adatfájl felolvasása, illetve adatok kinyerése, a labirintus (***Maze***) létrehozásához a kiválasztott pálya alapján, a ***MapLoader*** segítségével.

A MapLoader egy olyan QObject, amelynek egyetlen publikusan hívható metódusa a getMapByName, ezen keresztül hívja a GM.



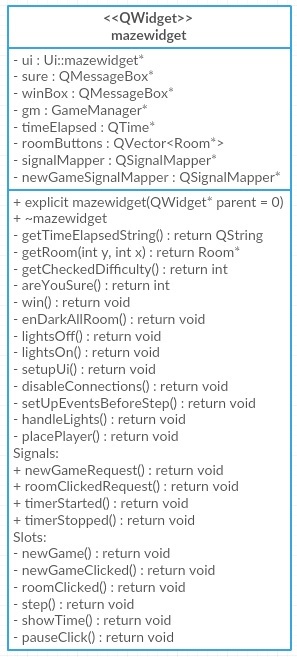
A **Maze** egy olyan QObject, amelynek feladata kvázi egybe fogni a szobák információit. Egy-egy szoba reprezentációja, egy QJsonObjectum, s ezen objektumok listájának listája, azaz egy mátrix-objektum az az információ, amelyet egy pályáról tárol a json adatfájl, s amit a Maze objektum tárol, szintén QJsonObject formájában.

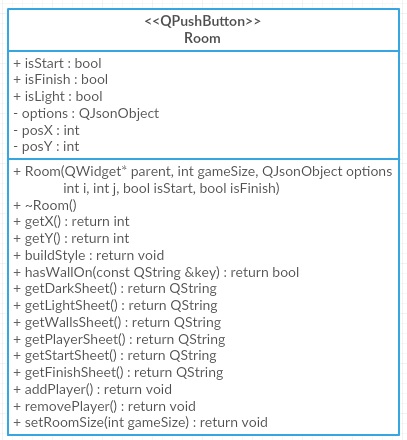


A GM az egyes lépési kérelmeknél, eldönti, hogy léphetünk-e a célra vagy sem, illetve vizsgálja a győzelmi állapotot, ha igen akkor visszaüzen a nézetnek.

A nézet, a játék állásának megfelelően változik. A bevilágított szobák kezelése is a nézet feladata, amelyhez a GM-ből nyer információkat.

A nézet, a triviális adattagokon, illetve önmagán kívül (***mazewidget***), egy ***Room*** nevezetű segéd-osztály is felhasznál. Ez egy QPushButton-ból származtatott osztály, amelynek a saját falait kell számon tartania, így adattagokkal bővül.





## Eseménykezelők:

* Mazewidget
  + Slot:
    - newGameClicked() : Az új játék kezdése gombra kattintás eseménye váltja ki. Meghatározza a új játék szintjét majd esemény kreál (**newGameRequest()**), amivel paraméterben átadja a GameManagernek a vezérlést: **newGameRequest(int gS)**
    - newGame () : A GameManager válasza (**newGameResponse**) fut be ide. Legenerálja a megfelelő számú Room-ot, s beállítja az eseményeit.
    - roomClicked() : A játék kezdésekor generált Room-ok kattintás eseményei futnak be ide. Meghatározza a küldő indexét, majd egy eseményt kreál (**roomClickedRequest()**) amiben átadja a paraméterrel együtt a vezérlést, a GameManager **roomClickedRequest(int index)** slotjának.
    - step() : A GameManager válasza (**roomClickedRequest()**) fut be ide. Frissíti a játékteret: a játékos új helyre kerül, a világítás is újraszámolódik, események frissülnek.
    - showTime() : A GameManagerben lévő időzítő timeout() eseménye fut be ide. Ezen esemény frissíti az órát.
    - pauseClick() : A szüneteltetés gomb kattintás eseménye fut be ide. Cseréli a gomb feliratát illetve kiváltja a megfelelő eseményt a GameManagerben lévő időzítőnek. (**timerStarted(), timerStopped()**)
  + Signal:
    - newGameRequest()
    - roomClickedRequest()
    - timerStarted()
    - timerStopped()
* GameManager
  + Slot:
    - newGameRequest(int gS) : Új játékot kezdünk, nullázódnak a számolók, és a változók. Beolvasódik a megfelelő pálya, majd üzenünk a nézetnek ha kész. (**newGameResponse()**)
    - roomClickedRequest(int index) : Megvizsgáljuk, hogy az aktuális lépés, valid-e. Ha valid akkor megtörténik a lépés, és figyeljük, hogy győzelmet ért-e. Ha a lépés megtörtént, akkor üzen a nézetnek. (**roomClickedResponse()**)
    - startTimer() : Ha a timer passzív, akkor újraindítjuk.
    - stopTimer() : Ha a timer aktív, akkor leállítjuk.
  + Signal:
    - newGameResponse()
    - roomClickedResponse()

## Világítás kezelése:



Abban a kérdésben, hogy mik a játékos aktuális pozíciójából bevilágított szobák, több alkérdésre válaszolunk. Először is fel kell mérnünk az 5x5-ös táblázat elemeit.

Az 1-essel jelölt rész a játékos pozíciója, ezt nyilván belátja, így ez a szoba világos lesz.

A 2-essel jelölt szobákra, az accessibleByPlayer függvény ad választ, hogy azok világosak-e.

A 3-assal jelölt sarkokra, már összetettebb szabály él: ha a játékos felöli 2 oldal valamelyikén nincs fal, de azzal a feltétellel, hogy az oda nyíló szobára viszont igaz az accessibleByPlayer.

A 4-essel jelölt szobák szabálya, hogy világítottak, ha a velük szomszédos -esel jelölt szoba, bevilágított.

Az 5-ös számmal jelzet szobák szabálya, hogy vagy a 3-mal jelzett szomszédjából, vagy a 4-essel jelzett szomszédjából, el lehessen oda jutni (nincs fal : accessible() függvény a gameManagerben) de az adott szomszéd szoba be legyen világítva.

A 6-ossal jelzett sarkok akkor világosak, ha elérhetőek egy szomszédos 5-össel jelölt, már bevilágított szobából.

### Pályák:

A pályákat JSON formátumban tároljuk, amelynek fix struktúrája kell, hogy legyen.

A struktúra az alap json objektummal kezdődik: [], viszont kulcsolva tárolunk benne értékeket, így az objektumformátumra alakítjuk:

{„rooms”: []}

A rooms egy olyan lista, amely az egyes sorokat reprezentálják a pályán. A tartalma egy lista, amely egy sor szobáit tartalmazza. Ebben a listában, van megfelelő számú szobaobjektum.

Példa egy szobaobjektumra:

{„type”: „room”, - A mező típusa: szoba (későbbi fejlesztésre alkalmas adat)

„left” : 1, - A bal fal: 1 ha van, 0 ha nincs

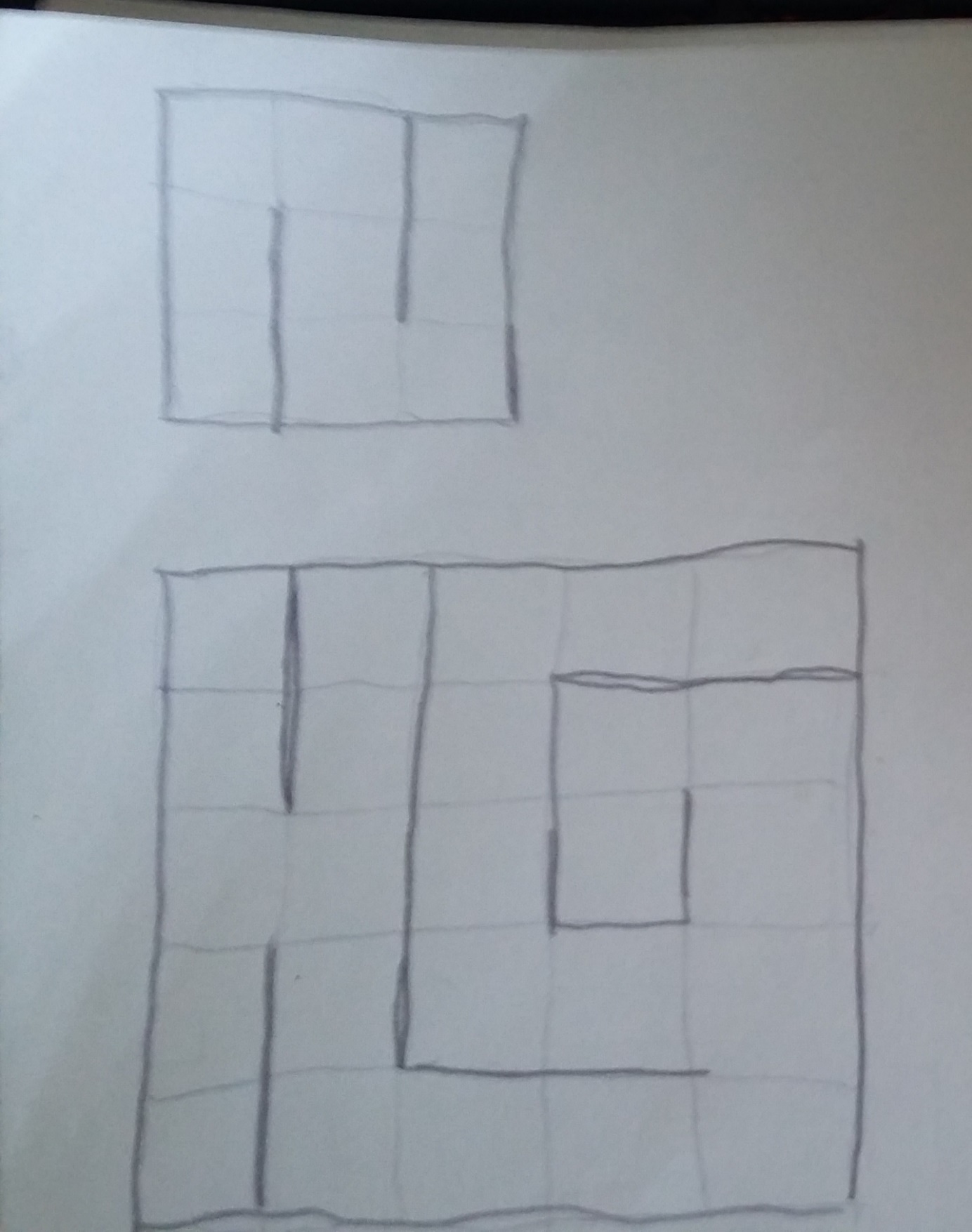
„top” : 0, - A felső fal: 1 ha van, 0 ha nincs

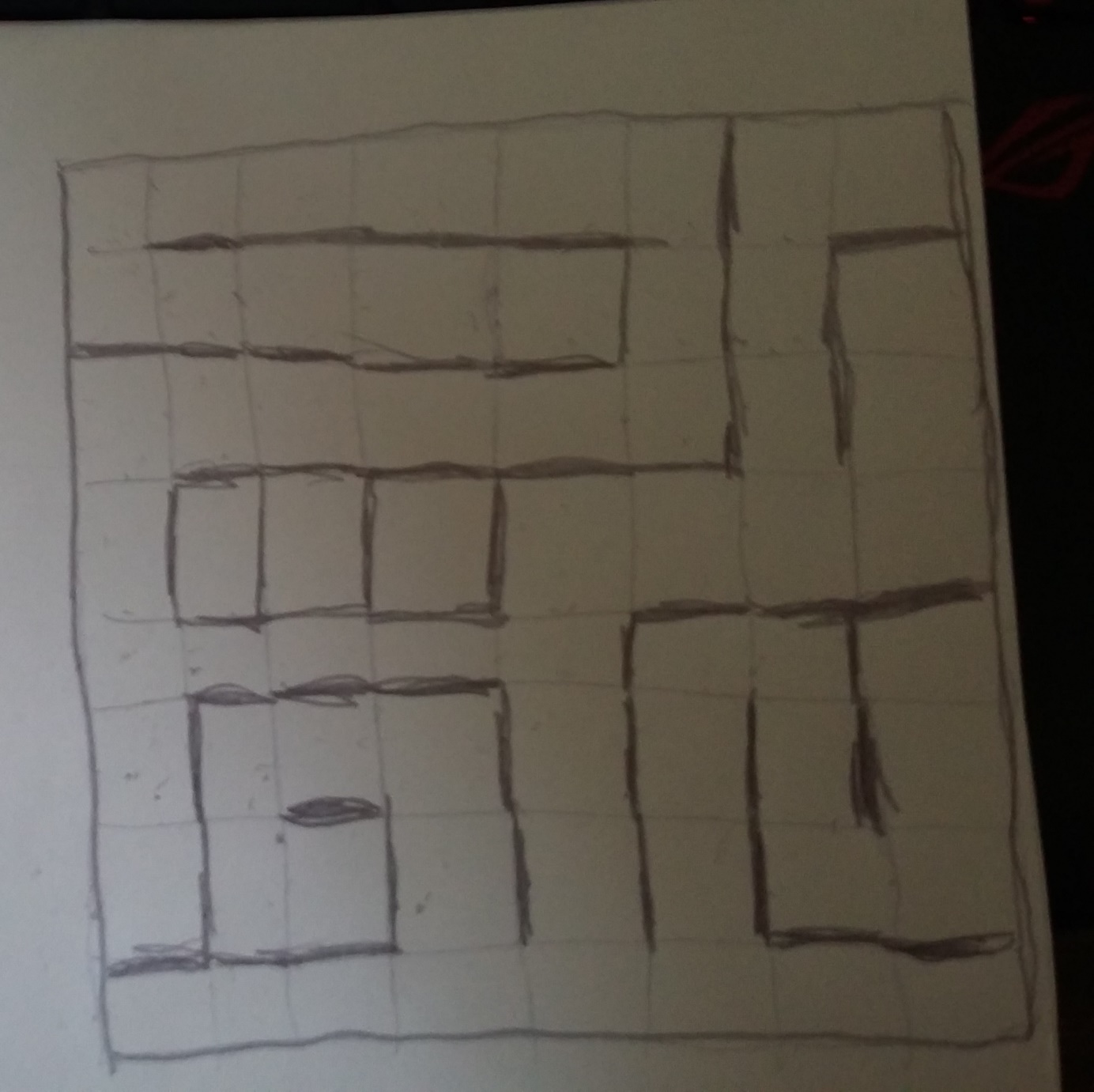
„right” : 1, - A jobb fal: 1 ha van, 0 ha nincs

„bottom”: 0} - Az alsú fal: 1 ha van, 0 ha nincs

A forrásfájlokat (map01.json, map02.json, map03.json) a data.qrc-ben tároljuk.

#### Szintek/pályák:

(3x3, 5x5)

(8x8)

## GameManager-Model Tesztesetek:

* initTestCase : új GameManager példányosítása
* gameManagerTestCase : Az új GM-en, megvizsgáljuk, hogy létrejött, hívható, és az alapbeállításokkal rendelkezik.
* newGame3Test : Új játékot kezdünk az első szinten, ami egy 3x3-as mátrixot jelent, a lépéseink száma 0, az időzítő elindul, a játékos a bal alsó sarokban van.
* newGame5Test : Új játékot kezdünk a második szinten, ami egy 5x5-ös mátrixot jelent, a lépéseink száma 0, az időzítő elindul, a játékos a bal alsó sarokban van.
* newGame8Test : Új játékot kezdünk a második szinten, ami egy 8x8-ös mátrixot jelent, a lépéseink száma 0, az időzítő elindul, a játékos a bal alsó sarokban van.
* stepTestCase : Új játékot kezdünk, majd megpróbálunk lépni, kétszer olyan helyre ahova nem tudunk, majd egyszer ahová tudunk. A lépéseink száma 1 kell legyen ,nem győzhettünk, és játékban kell lennünk.
* winTestCase : Új játékot kezdünk 3x3-as pályán, majd a legrövidebb úton a célba érünk, ekkor igaz lesz hogy győztünk, és hogy játékon kívülre kerültünk.
* newGameNewGameTestCase : Új játékot kezdünk a 3x3-as szinten, és megnézzük, hogy a játék mérete 3-e. Ezután újra új játékot kezdünk a 8x8-as, majd az 5x5-ös szinten is. Eredményként azt várjuk, hogy nem győztünk, játékban vagyunk és a játék mérete 5.
* testCase1 : A tesztkörnyezet tesztelése.
* failureTestCase : A tesztkörnyezet negatív tesztelése.