# Prototípus koncepciója

[A prototípus program célja annak demonstrálása, hogy a program elkészült, helyesen működik, valamennyi feladatát teljesíti. A prototípus változat egy elkészült program kivéve a kifejlett grafikus interfészt. Ez a program is parancssorból futtatható és karakteres ernyőkezelést alkalmaz. Az ütemezés, az aktív objektumok kezelése megoldott. A business objektumok - a megjelenítésre vonatkozó részeket kivéve - valamennyi metódusa a végleges algoritmusokat kell, hogy tartalmazza. A megjelenítés és működtetés egy alfanumerikus képernyőn vezérelhető és követhető, ugyanakkor a vezérlés fájlból is történhet és a megjelenítés fájlba is logolható, ezzel megteremtve a rendszer tesztelésének lehetőségét. Különös figyelmet kell fordítani a parancssori interfész logikájára, felépítésére, valamint arra, hogy az mennyiben tükrözi és teszi láthatóvá a program működését, a beavatkozások hatásait.]

## Változás hatása a modellre

### Módosult osztálydiagram

*[Az analízis modell osztálydiagramja a változások figyelembevételével.]*

### Új vagy megváltozó metódusok

*[Az analízis modell osztályleírásaiból azon metódusok újbóli felsorolása leírással együtt, amelyek a változtatás miatt módosultak vagy újonnan be lettek vezetve.]*

### Szekvencia-diagramok

*[Az analízis modell szekvenciadiagramjaiból a változás által érintett, előírt, módosított diagramok.]*

## Prototípus interface-definíciója

[Definiálni kell a teszteket leíró nyelvet. Külön figyelmet kell fordítani arra, hogy ha a rendszer véletlen elemeket is tartalmaz, akkor a véletlenszerűség ki-bekapcsolható legyen, és a program determinisztikusan is tesztelhető legyen.]

### Az interfész általános leírása

[A protó (karakteres) input és output felületeit úgy kell kialakítani, hogy az input fájlból is vehető legyen illetőleg az output fájlba menthető legyen, vagyis kommunikációra csak a szabványos be- és kimenet használható.]

A program interfésze csak a szabványos bemenetről fogad parancsokat, ezzel lehetővé téve mind a terminálból való vezérlést, mind a fájlból való parancsfogadást. Kimenttét a szabványos kimentere írja, így támogatva a már említett terminálból történő használatot, valamint átirányítható fájlba.

A fájlból való parancs fogdásnak és kimenet átirányításnak köszönhetően mód van az előre elkészített automatikus tesztek futtatására, és ezek eredményének elmentésére. Ezzel lehetséges az automatikus tesztelés, mely sokkal hatékonyabb és eredményre vezetőbb, mint a hagyományos „kézi” tesztelés. Ezek a tesztesetek parancsok sorozatából állnak.

### Bemeneti nyelv

[Definiálni kell a teszteket leíró nyelvet (szintakszis és szemantika). Külön figyelmet kell fordítani arra, hogy ha a rendszer véletlen elemeket is tartalmaz, akkor a véletlenszerűség ki-bekapcsolható legyen, és a program determinisztikusan is futtatható legyen. A szálkezelést is tesztelhető, irányítható módon kell megoldani. A programot egy adott konfigurációból is el kell tudni indítani, vagyis kell olyan parancs, amivel konkrét előre megadott állapotból indul a rendszer (pl. load).]

load-map <filename>

**Leírás:** Pályabetöltése fájlból.

**Opciók:**

* <filename> - Pályát tartalmazó fájl neve

control <worker-id> <direction>

**Leírás:** Adottmunkás léptetése egy irányba.

**Opciók:**

* <worker-id> - Munkás azonosítója
* <direction> - Irány [-u|-r|-d|-l] - fel (Up), jobbra (Right), le (Down), balra (Left)

drop-special <worker-id> <type>

**Leírás:** Adottmunkással méz vagy olaj elhelyezése az aktuális mezőjére.

**Opciók:**

* <worker-id> - Munkás azonosítója
* <type> - Anyag fajtája [-h|-o] - méz (Honey), olaj (Oil)

ls-workers

**Leírás:** Pályán lévő munkások listázása.

**Opciók:** -

ls-boxes

**Leírás:** Pályán lévő dobozok listázása.

**Opciók:** -

ls-fields

**Leírás:** Pályán lévő mezők listázása.

**Opciók:** -

log [<filename>]

**Leírás:** Metódus hívások naplózása, ha meg van adva opcionális fájlnév akkor fájlba, ha nincs, akkor a képernyőre.

**Opciók:**

* <filename> - Opcionális. Ha nincs meg adva akkor a képernyőre naplóz, ha meg van adva akkor az adott fájlba.

show-log

**Leírás:** Kiírja a naplózások helyét. (pl. scrn (képernyő), logfile.txt)

**Opciók:** -

log-off [<filename> | -all]

**Leírás:** Leállítja az adott egység naplózását.

**Opciók:** [<filename> | -all] – Opcionális. Kihagyva a képernyőre való naplózást állítja le. Fájlnevet megadva az adott fájlba való naplózást. -all kapcsolóval pedig minden naplózást.

drop-map

**Leírás:** Az aktuális pályá eldobása (például rossz betöltése esetén).

**Opciók:** -

show-map

**Leírás:** Az aktuális pálya kiírása (fájlnév).

**Opciók:** -

run-test <filename>

**Leírás:** Előre megírt teszteset futtatása.

**Opciók:** <filename> - A tesztesetet tartalmazó fájl.

exit

**Leírás:** Kilépés a programból.

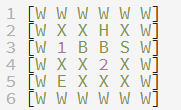
**Opciók:** -

**Konfigurációs fájlok nyelvtana**

**Pályát megadó fájl**

Annak érdekében, hogy minél átláthatóbban, egyszerűbben megadhassa a felhasználó a saját pályáját, egy saját, letisztultságra törekvő nyelvtant dolgoztunk ki. Formátuma egyszerű txt.

* **Először egy példa, utána magyarázat:**



Ez egy 6x6-os pálya, fallal körülvéve, két munkással, két dobozzal, lyukkal, kapcsolóval és egy célmezővel.

**Nyelvtan:**

A pálya méretét a sorok és oszlopok száma határozza meg, az oszlopok szóközzel elválasztottak. Minden sor *[* szimbólummal kezdődik és *]* szimbólummal végződik (sortörés, a jobb áttekinthetőség érdekében ajánlott. A sor nyitó és sor záró szimbólumok között különböző elemek szerepelhetnek (lsd. lentebb). A mezők alapból sima mezők, melyeken állhatnak entitások, ha nem sima mezőt szeretnénk módosítót kell elhelyezni.

***Megkötések*:** A soroknak ugyanolyan hosszúnak kell lenniük, különben érvénytelen a pálya.

**Elemek:**

* *W – Fal elem (Wall)*
* *H – Lyuk (mező módosító, Hole)*
* *S – Kapcsoló (mező módosító, Switch)*
* *E – Célmező (mező módosító, Endzone)*

Az előzőekben említett módosítók nélkül egy adott mező sima mező lesz, ekkor helyezkedhet rajta inicializáláskor entitás, vagy lehet üres.

* *X – Az adott sima mezőn nincs semmi.*
* *B – Az adott sima mezőn egy doboz található (Box)*
* *0-9 – Az adott mezőn egy munkás található, aminek az azonosítója (worker-id) az adott egyjegyű szám.*

Egyéb felmerülő kérdés, hogy melyik lyukhoz, melyik kapcsoló tartozik. Ezt a kérdést az inicializálás egy megadott módon dönti el, ami a következő:

Az egyes elemek inicializálása balról-jobbra, fentről-le történik, ha kapcsolóhoz, illetve lyukhoz ér, akkor annak a párja az inicializálást folytatva az első megfelelő elem. (A példában a 2. sor 4. helyén álló lyuk és a 3. sor 5. helyén álló kapcsoló tehát össze van kötve). Amennyiben nem jut az adott elemnek „pár”, akkor az annak megfelelően viselkedik.

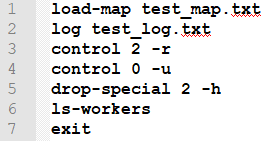
Ez a saját nyelvtan jelentősen leegyszerűsíti a pályák megadását, ami más általános formátumban (pl. XML, JSON) leírva sokkal kevésbé lenne áttekinthető. Jó példa a struktúra által nyújtott segítésére, hogy a megadás szemantikájában benne vannak a mező szomszéd kapcsolatok, így azokat explicit nem szükséges megadni. (Ez a példában szereplő 6x6-os pályán 120 szomszéd kapcsolat „megspórolása”, hiszen ezeket az inicializálás automatikusan felderíti és beállítja).

**Tesztesetet megadó fájl (parancs szkript)**

Lehetőség van atesztesetek előre való összeállítására, ezzel automatikusan futtatva őket, így sokkal hatékonyabb tesztelést elérve.

Ennek módja, hogy parancsokat előre leírunk megfelelő sorrendben, megfelelően paraméterezve egy egyszerű txt fájlba. A formátum választása itt is azért esett erre az egyszerű esetre, mert így lehet a lehető legegyszerűbben összeállítani egy ilyen fájlt, szinte magától értetődően, hiszen igazából egy parancssorozat.

* **Először egy példa, utána magyarázat (***run***-***test test.txt***) :**



Ez a teszteset betölt egy pályát, beállítja a naplózás helyét, irányítja a 2-es azonosítójú munkást jobbra, a 0-s azonosítójút felfele. Ezután a 2-es azonosítójú munkással lerak egy adag mézet a mezőjére. Végezetül kilistázza a munkásokat. Utána a tesztesetből való kilépés következik.

**Nyelvtan:**

A tesztesetek leírása tulajdonképpen parancsok sorozata, amiket a terminálban is ugyanígy adnánk ki. Fontos, hogy minden parancs külön sorban kell, hogy elhelyezkedjen.

***Tudnivalók:*** Minden tesztesetbe való belépéskor a terminálban beállított naplózási módok törlődnek, tehát azok nem befolyásolják a tesztesetben való naplózást (kilépéskor visszaállítódnak a teszteset előtti naplózási módok, de a tesztesetben beállítottak nem lesznek érvényben.)

Az egyes tesztesetekben ugyanazok a parancsok érhetőek el, ugyanolyan formában, mint a program átlagos futtatásánál. Tesztesetet a ***run-test <filename>*** paranccsal indíthatunk. Minden tesztesetet az ***exit*** parancsnak kell zárnia.

### Kimeneti nyelv

[Egyértelműen definiálni kell, hogy az egyes bemeneti parancsok végrehajtása után előálló állapot milyen formában jelenik meg a szabványos kimeneten. A program képes legyen olyan kimenetet előállítani, amellyel az objektumok állapota ellenőrizhető (pl. save). Ebben az alfejezetben is precízen definiálni kell, hogy a kimenet nyelve milyen elemekből és milyen szintakszissal áll elő.]

## Összes részletes use-case

[A use-case-eknek a részletezettsége feleljen meg a kezelői felületnek, azaz a felület elemeire kell hivatkozniuk a bemeneti nyelv parancsai alapján.

Alábbi táblázat minden use-case-hez külön-külön.]

|  |  |
| --- | --- |
| **Use-case neve** |  |
| **Rövid leírás** |  |
| **Aktorok** |  |
| **Forgatókönyv** |  |

## Tesztelési terv

[A tesztelési tervben definiálni kell, hogy a be- és kimeneti fájlok egybevetésével miként végezhető el a program tesztelése. Meg kell adni magas szintű teszt forgatókönyveket. Az egyes teszteket elég informálisan, szabad szövegként leírni, tesztesetenként egy-öt mondatban. Minden teszthez meg kell adni, hogy mi a célja, a proto mely funkcionalitását, osztályait stb. teszteli. Az alábbi táblázat minden teszt-esethez külön-külön elkészítendő.]

|  |  |
| --- | --- |
| **Teszt-eset neve** |  |
| **Rövid leírás** |  |
| **Teszt célja** |  |

## Tesztelést támogató segéd- és fordítóprogramok specifikálása

[Rövid bemutatással (elvárt funkcionalitás) specifikálni kell a tesztelést támogató segédprogramokat.]

## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2010.03.21. 18:00 | 2,5 óra | Horváth  Németh  Tóth  Oláh | Értekezlet.  Döntés: Horváth elkészíti az osztálydiagramot, Oláh a use-case leírásokat. |
| 2010.03.23. 23:00 | 5 óra | Németh | Tevékenység: Németh implementálja a tesztelő programokat. |
| … | … | … | … |