# 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.]

### 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).]

**Parancs1**

**Leírás:**

**Opciók:**

**Parancs2**

**Leírás:**

**Opciók:**

[Ha szükséges, meg kell adni a konfigurációs (pl. pályaképet megadó) fájlok nyelvtanát is.]

### 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.]

addneighbour, addOccupant, addMycelium, addMushroomBody

addOccupant → ez akkor hivodik meg ha az Insect mozog (resze az Insect mozgatasnak)

addMycelium → csak akkor hivodik meg ha arra a User megkerte hogy noljon oda (resze a fonal novesztesnek)

addMushroomBody → csak akkor hivodik meg ha arra a User mekerte hogy noljon oda (resze a gombatest novesztesnek)

addNeighbour → akkor hivodik meg ha Tectontores vagy ujTecton teremtese tortenik (tenylegesen resze a Tecton mukodesenek)

User, gombasz/Mycologist, rovarasz

Menet: Jatekosok hozzaadasa a jatekhoz → Jatek vilag legeneralasa → Jatek elinditasa(ellenorzi hogy van-e megfelelo mennyisegu jatekos es sorrend beosztasa)

→ End turn kuldese(repeating after every player)

round – minden jatekos egyszer

turn – egy jatekos kore

use-case : Kivalasztott objektum tulajdonsagainak kiirasa

: Jatek hosszanak beallitasa

RG/1

**Leírás: Tectontörés**

**Opciók: Nincsenek paraméterek**

|  |  |
| --- | --- |
| **Use-case neve** | Tectontörés |
| **Rövid leírás** | Az adott FertileTecton eltörik, vagyis a rajta lévő objektumok megsemmisülnek (kivéve a gombatestet) és egy új FertileTecton keletkezik, melynek egyetlen szomszédja az eredeti FertileTecton lesz. Az eredeti FertileTectonnak is beállítjuk az új FertileTectont mint szomszéd.  Hasonlóan történik a törés a többi Tecton típusra is. A törés mindig FertileTectont eredményez az új Tecton típusaként. |
| **Aktorok** | User |
| **Forgatókönyv** | 1. A BreakTimer eléri a 0-at  2. A Tectonrol minden eltörlődik (kivéve a gombatestet)  3.Új Tecton keletkezik  4.Az új Tecton és az eltört Tecton szomszédok lesznek |

RG/2

**Leírás: Új Tecton létrehozása**

**Opciók: Első paraméter eldönti milyen típusu legyen az új Tecton, A következő paraméter egy lista mely tartalmazza a Tectonokat amelyek szomszédjai lesznek az új Tectonnak**

|  |  |
| --- | --- |
| **Use-case neve** | Új Tecton létrehozása |
| **Rövid leírás** | A felhasználó paraméterként átadja az összes olyan Tectont amelyeket kívánja hogy szomszédjai legyenek az új Tectonnak. Itt legelső paraméterként azt adja át, hogy milyen típusú Tectonná szeretné csinálni az új Tectont. |
| **Aktorok** | User |
| **Forgatókönyv** | 1. Új Tecton létrehozása az átadott paramétereknek megfelelően  2. Az új Tectonnak beallítódnak a szomszédjai |

CREATE\_TECTON TectonType {Neihgbouring Tectons}

RG/3

**Leírás: Tecton BreakTimerjének beallítása egy adott értékre**

**Opciók: Egyetlen paramétere az a szám ami be lesz állitva mint új BreakTimer**

|  |  |
| --- | --- |
| **Use-case neve** | Tecton BreakTimerjének beallítása egy adott értékre |
| **Rövid leírás** | A User kiválaszthat egy adott Tectont és annak beállítja a breakTimerjét egy adott értékre. |
| **Aktorok** | User |
| **Forgatókönyv** | 1. A Tecton kiválasztása amelynek módosítanánk a BreakTimerjét  2. A kiválasztott Tecton BreakTimerjének beallítása a paraméterben átadott értékre |

SET\_BREAKTIMER number

RG/4

**Leírás: Játék világ legenerálása/inicializálása**

**Opciók: Nincsenek paraméterek**

|  |  |
| --- | --- |
| **Use-case neve** | Játék világ legenerálása/inicializálása |
| **Rövid leírás** | Megteremti a kezdeti állapotú Tectonokat, Gombatesteket, Gombafonalakat és Insecteket. |
| **Aktorok** | User |
| **Forgatókönyv** | 1. Amikor az applikáció elindul meghívódik a metódus hogy generáljon Tectonokat  2. Sorrendben leteremti a játékmező elemeit (Tectonok, Gombatestek, Gombafonalak, Insectek) |

RUN text\_file

RG/5

**Leírás: Játék végének kezelése**

**Opciók: Nincsenek paraméterek**

|  |  |
| --- | --- |
| **Use-case neve** | Játék végének kezelése |
| **Rövid leírás** | Amikor véget ért az utolsó Round, akkor eldönti a nyertest és bemutatja a végeredményeket/score-okat a user-nek |
| **Aktorok** | User |
| **Forgatókönyv** | 1. Véget ér az utolsó Round  2. Kiszámolódnak a pontszámok  3. Kiíródik a végeredmény/ki mennyi pontot ért el |

END\_GAME

RG/6

**Leírás: Az EndgameTimer beállítása**

**Opciók: Egyetlen paramétere egy szám mely az új maradék Round-okat jelöli**

|  |  |
| --- | --- |
| **Use-case neve** | Az EndgameTimer beállítása |
| **Rövid leírás** | A User beállítja a hátralévő Round-ok számát |
| **Aktorok** | User |
| **Forgatókönyv** | 1. Az EndgameTimer beállítása az új értékre |

SET\_ENDGAMETIMER number

RG/7

**Leírás: End turn küldése**

**Opciók: Nincsenek paraméterek**

|  |  |
| --- | --- |
| **Use-case neve** | End turn küldése |
| **Rövid leírás** | Akkor történik mikor az egyik játékos úgy gondolja, hogy mostmár befejezi a lépéseit és átadja a Turn-jét. Így elkezdődhet a következő játékos Turn-je. |
| **Aktorok** | User |
| **Forgatókönyv** | 1. A User kiküldi a parancsot, hogy vége van a Turn-jének  2. Végrehajtódnak a Turn-ök közötti események  3. Elkezdődik a sorban következő User/Játékos Turn-je |

END\_TURN

RG/8

**Leírás: Játékosok hozzáadása a játékhoz**

**Opciók: Első paraméter a játékos neve, a második paraméter hogy Gombász vagy Rovarász lesz a játékos**

|  |  |
| --- | --- |
| **Use-case neve** | Játékosok hozzáadása a játékhoz |
| **Rövid leírás** | Miután ki lett választva, hogy hányan szeretnének játszani. A rendszer kiválasztja, hogy ki mi legyen (Gombász vagy Rovarász) |
| **Aktorok** | User |
| **Forgatókönyv** | 1. Valaki megnyitja a játékot  2. Kiválasztódik hány játékos szeretne játszani  3. A rendszer kiossza a játékosokat szerep szerint (Gombász vagy Rovarász) |

ADD\_PLAYER player\_name player\_type

RG/9

**Leírás: Játék indítása**

**Opciók: Nincsenek paraméterek**

|  |  |
| --- | --- |
| **Use-case neve** | Játék indítása |
| **Rövid leírás** | Miután ki lettek választva a játékosok és a játék világ legenerálódott, a rendszer sorrendbe rakja a játékosokat majd az első játékosnak a sorból átadja az uralmat. |
| **Aktorok** | User |
| **Forgatókönyv** | 1. Meghívódik a játék elindítása  2. A rendszer sorrendbe rakja a játékosokat  3. A rendszer átadja az uralmat a sorban első játékosnak |

START\_GAME

## 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.]

RG/1

|  |  |
| --- | --- |
| **Teszt-eset neve** | Új Tecton sikeres legyártása |
| **Rövid leírás** | A rendszer sikeresen legyárt egy általa kiválasztott típusú Tectont. A felhasználó egy időben megmondja azt is, hogy ennek az új Tectonnak kik lesznek a szomszédjai. |
| **Teszt célja** | Megnézni, hogy a rendszer sikeresen tud bármilyen típusú Tecton-t gyártani |

RG/2

|  |  |
| --- | --- |
| **Teszt-eset neve** | Tectontörés |
| **Rövid leírás** | A BreakTimert beállítjuk 0-ra egy adott Tectonon, majd megvizsgáljuk, hogy az új kör után lett-e neki új szomszédja és hogy a gombatesten kívül minden más megsemmisült róla. |
| **Teszt célja** | Megnézi, hogy a rendszer sikeresen tud-e Tectontörést elvégezni és hogy ilyenkor tényleg minden letörlődik az adott Tectonról (kivéve a gombatest). |

RG/3

|  |  |
| --- | --- |
| **Teszt-eset neve** | Világ legenerálása |
| **Rövid leírás** | A rendszer felépít egy új játék világot. Benne Tectonokkal, rajtuk Gombatestekkel és Gombafonalakkal és Insectekkel. |
| **Teszt célja** | Hogy minden egyes játék világ eleme sikeresen legyártható és legyártódik |

RG/4

|  |  |
| --- | --- |
| **Teszt-eset neve** | Játék végének kezelése |
| **Rövid leírás** | Miután lejárt az utolsó Round is, meghatározni mind a Gombászok közül egy nyertest, mind a Rovarászok közül egy nyertest majd kiírni a pontszámukat. |
| **Teszt célja** | Megnézni, hogy jól számolódik-e ki a pontszám és hogy helyesen íródnak ki a végeredmények |

|  |  |
| --- | --- |
| **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. |
| … | … | … | … |