# 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)

|  |  |
| --- | --- |
| **Use-case neve** | Tectontörés |
| **Rövid leírás** | Az adott Tecton eltörik, vagyis a rajta lévő objektumok megsemmisülnek(kivéve a gombatestet) és egy új Tecton keletkezik, melynek egyetlen szomszédja az eredeti Tecton lesz. Az eredeti Tectonnak is beallitjuk az új Tectont mint szomszéd. |
| **Aktorok** | GameManager, User |
| **Forgatókönyv** | 1.Trigger: A BreakTimer eleri a 0-at vagy a User meghivja debug commandot  2. GameManager megkapja az uzenetet hogy ezt el kell torje  3. A GameManager eltavolitja a Tectonrol a fonalakat, bogarakat es sporakat(meghivva az ezekhez megfelelo metodusokat)  4. A GameManager meghivja az uj Tecton letrehozasat  5. A GameManager frissiti a szomszedsagi listajat mind az uj mind a jelenlegi Tectonnak |

|  |  |
| --- | --- |
| **Use-case neve** | Új Tecton létrehozása |
| **Rövid leírás** | A felhasználó paraméterként átadja az összes olyan Tectont amelyeket kivánja hogy szomszédjai legyenek az új Tectonnak. Itt legelső paraméterként azt adja át hogy milyen típusu Tectonná szeretné csinálni az új Tectont. |
| **Aktorok** | User, GameManager |
| **Forgatókönyv** | 1. Trigger: Vagy a User megivja a debug commandot vagy a Tectontores soran hivodik meg  2. A User vagy a GameManager atadja hogy milyen tipusura szeretne az uj Tectont es hogy kik lesznek a szomszedjai  3.A rendszer validalja hogy letezik olyan fajta Tecton es hogy leteznek ezek az uj szomszedok  4. Az uj Tecton legyartodik es elmenti a szomszedjait  5. A GameManager frissiti a szomszedsagi listajat mind az uj mind a jelenlegi Tectonnak |

|  |  |
| --- | --- |
| **Use-case neve** | Tecton BreakTimerjánek beallitasa egy adott ertekre |
| **Rövid leírás** | A User kivalaszthat egy adott Tectont es annak beallitja a breakTimerjet egy adott ertekre, igy a kovetkezo korben az a Tecton eltorik |
| **Aktorok** | User |
| **Forgatókönyv** | 1. A User kivalasztja a Tectont aminek be szeretne allitani a BreakTimerjet(itt a nevet kellene megadja)  2. A User megadja az erteket ami be fog allitodni a BreakTimernek (itt egy szam kellene) |

|  |  |
| --- | --- |
| **Use-case neve** | Jatek kor elkezdese |
| **Rövid leírás** | Felkesziti a jatek vilagot egy uj kor kezdetere |
| **Aktorok** | GameManager |
| **Forgatókönyv** | 1. Trigger: Megkapja a jelet valahonnan (!!! ezt el kell donteni) hogy uj kor indul  2. Frissiti a leszamolot a koronek  3. Meghivja az onRoundBegin() metodust minden objektumra aminek szuksege van erre  (!!! Itt meg kell beszelni hogy mikor tortenik a kor, milyen sorrendben kuldi ki az onTurnBegin()/onRoundBegin() jeleket es hogy majd ezeket kiirja e valahova vagy mit csinal veluk es hogy pl. a mycelium sustaint hogy ellenorzi ha egyaltalan o ellenorzi) |

|  |  |
| --- | --- |
| **Use-case neve** | Jatek vilag inicializalasa |
| **Rövid leírás** | Megteremti a kezdeti allapotu Tectonokat, hogy hogy nez ki a vilag a jatek inditasa elott |
| **Aktorok** | GameManager |
| **Forgatókönyv** | 1. Amikor az applikacio elindul meghivodik a metodus hogy generaljon Tectonokat  2. Betolti a Map-ot/Terkepet (!!!ezt meg kell beszelni hogyan szeretnenk) |

|  |  |
| --- | --- |
| **Use-case neve** | Jatek vegenek kezelese |
| **Rövid leírás** | Amikor veget ert az utolso kor, akkor eldonti a nyertest es bemutatja a vegeredmenyeket/score-okat a user-nek |
| **Aktorok** | GameManager |
| **Forgatókönyv** | 1. Trigger: amikor eleri a 0-at a Turn/Round counter es lefutnak az utolso onRoundBegin/onTurnBegin funkciok  2. A GameManager eldonti a pontszamok alapjan a nyertest  3. Ezeket kiirja a console-ra  4. Meghivja mindenre a destruktort es mindent visszaalit egy kezdeti allapotba ??? |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

## 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** | Új Tecton sikeres legyártása |
| **Rövid leírás** | A User felhasználja az Új Tecton létrehozása use-caset és legyárt egy új Tectont. Ennek ellerörzése, hogy tényleg megtörtént, hogy, az összes kiválasztott Tektonnak megnézzük hogy szomszédja-e az új Tecton és hogy az új Tectonnak tényleg szomszédja-e az összes kiválasztott Tecton, és hogy az új Tecton megfelelő típusu. |
| **Teszt célja** | Megnézni hogy a rendszer sikeresen tud bármely típusu Tecton-t gyártani |

|  |  |
| --- | --- |
| **Teszt-eset neve** | Tectontörés |
| **Rövid leírás** | A BreakTimert beallitani egy Tectonra 0-ra majd megviszgáni 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). |

|  |  |
| --- | --- |
| **Teszt-eset neve** | Vilag legeneralasa |
| **Rövid leírás** | Egy mock vilagot elore megepiteni majd lefutattni a vilag legeneralasat a GameManagernek es ha a kettoben az elemek megegyeznek akkor jol mukodik a vilag generalas |
| **Teszt célja** | Hogy jol mukodik a vilag generalas |

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