8. Részletes tervek

27 – NASchA

Konzulens:

Goldschmidt Balázs

Csapattagok

Czanik Bálint	H7EEPG	czanik.balint@gmail.com
Nagy Örs	P28RW9	nagyors456@gmail.com
Novák Bálint Huba	OHKFY9	novak.balint.huba@gmail.com
Szabó Bence Sándor	**NQB6UO**	**szabo.bence.sandor@gmail.com**
Tokovics Dávid Tamás	H9LGJI	tokovicsdavid00@gmail.com

2021. április 12.

8. Részletes tervek

8.0 Előzőek javítása/kiegészítése (interfész).

Setboard

Leírás: Itt adhatjuk meg a létrehozni kívánt aszteroidáink adatait. A parancs egy számot vár, ami az aszteroidák számát mondja meg, ezután ennyi egymás alatti sorból beolvassa az aszteroidákat. Ha nem talál a beolvasás során main aszteroidát, akkor a sorban utolsó aszteroidát fogja annak nyilvánítani. Üres magú aszteroidát úgy lehet létrehozni, ha a kapcsoló helyére egy -e írunk. Ekkor a mag nevét nem kell megadni. Az aszteroida magját a megfelelő kapcsoló és a mag nevével lehet megadni. A kapcsolók:
-c szén
-i vas
-s szilikon
-u urán
-w vízjég
Ekkor a mag alapértelmezett értékekkel jön létre
Blokk: SETUP
Opciók: `Setboard <szám>`
`<név>;<réteg szám>;<kapcsoló>;<mag neve>;<napközel van-e (1,0)>;<main aszteroida-e (1,0)>`
`pl.: Setboard 3`
`a1;4;-u;u1;1;0`
`a2;5;-i;i1;0;1`
`a3;3;-c;c1;1;0`

Makegate

Leírás: Ezzel a paranccsal létre lehet hozni egy kapupárt amik parban vannak.
Blokk: SETUP
Opciók: `Makegate <kapu 1 neve> <kapu 2 neve>`

Setnei

Leírás: Ezzel a paranccsal lehet beállítani az objektumok szomszédjait. A parancs egy számot vár, ami megmondja hány szomszédot fog beolvasni. Ezután ennyi egymás alatt lévő sorból beolvassa a szomszédokat.
Blokk: SETUP
Opciók: `Setnei <szám>`
`<aszteroida név1>;<aszteroida név2>`
`pl.: Setnei 4`
`a1;a3`
`a4;a1`
`a5;a2`
`a5;a3`

Settler

Leírás: Létrehoz egy adott nevű telepest, és lehelyezi az adott aszteroidára

Blokk: SETUP
Opciók: `Settler <név> <aszteroida neve>`
`pl.: Settler s1 a1`

Settler

Leírás: Létrehoz egy adott nevű telepest, és lehelyezi az adott aszteroidára
Blokk: SETUP
Opciók: `Settler <név> <aszteroida neve>`
`pl.: Settler s1 a1`

Setinventory

Leírás: Egy adott objektumot egy másik objektum inventoryjába rak. Ez lehet telepesnek nyersanyag vagy kapu, vagy aszteroida magjának utólagos beállítása
Blokk: SETUP
Opciók: `Setinventory <cél objektum> <forrás objektum>`
`pl.: Setinventory s1 u1`

Robot

Leírás: Létrehoz egy adott nevű robotot, és lehelyezi az adott aszteroidára
Blokk: SETUP
Opciók: `Robot <név> <aszteroida neve>`
`pl.: Robot r1 a1`

Buildbase

Leírás: Egy adott telepes építi a bázist a nála lévő nyersanyagokból
Blokk: PROGRESS
Opciók: `Buildbase <telepes neve>`
`pl.: Buildbase s2`

8.1 Osztályok és metódusok tervei.

8.1.1 Entity

- **Felelősség**

Egy absztrakt osztály, leszármazottai olyan lények, amik az aszteroida övben tevékenykedni tudnak. Ilyen tevékenység lehet a fúrás, bányászás, mozgás, stb.. Itt tárolódik a lény pozíciója, ami mindig egy Thing. Ezen kívül a közös metódusok vannak megvalósítva az Entity osztályban.

- **Attribútumok**
  - **location**: Egy Thing típusú protected(#) láthatóságú attribútum, amely a lény tartózkodási helyét tárolja. Csak 1 ilyen van, nem lehet több, viszont valahol mindenképp tartózkodnia kell.

- **Metódusok**
  - ***void done()***: Egy private(-) láthatóságú absztrakt metódus, amelyet minden lény úgy valósít meg, hogy a megfelelő vezérlőnek szóljon, hogy az adott lény készen van.

- **void drill()**: Egy public(+) láthatóságú metódus, amely a fúrni képes lényeknek (Settler és Robot) a fúrását valósítja meg. A **location** attribútum drill() metódusát hívja meg. Emellett meghívja a saját **done** metódusát.
- **void move(Thing destination)**: Egy public(+) láthatóságú metódus, amely a lényt mozgatja át a kapott **destination** Thing-re, ha a **destination** a **location** szomszédja. A mozgatás úgy történik, hogy a **location** removeEntity nevű metódusával eltávolítja a lényt a tartózkodási helyéről (argumentumként saját magát adja), majd a **destination** addEntity nevű metódusával hozzáadja a lényt a kapott Thing-re (argumentumként saját magát adja), ezután a **location** attribútumot beállítja a **destination**-re. Emellett meghívja a saját **done** metódusát.
- **void wait()**: Egy public(+) láthatóságú metódus, amely a saját **done** metódust hívja meg, ezen kívül nem csinál semmit.
- **void addMaterial(Material m)**: Egy public(+) láthatóságú metódus, amely itt nem csinál semmit, azok a lények, akik nyersanyagot képesek tárolni (Settler és Ufo), azok fogják felülírni (megvalósítani).
- **void mine()**: Egy public(+) láthatóságú metódus, amely a bányászni képes lényeknek (Settler és Ufo) a bányászását valósítja meg. A **location** excavate nevű metódusát hívja meg, majd attól függően, hogy ez a metódus adott-e vissza Materialt, cselekszik. Ha a visszakapott érték null, akkor nem történik semmi, viszont a nem null, akkor meghívja a saját **addMaterial** metódusát és argumentumként adja. Emellett meghívja a saját **done** metódusát.
- **void die()**: Egy public(+) láthatóságú metódus, amely a **location** removeEntity metódusát hívja meg, argumentumként saját magát adva. Ezt minden leszármazott pontosítja azzal, hogy a lényhez illő kontrollerből távolítsa el a lényt.
- **void explode()**: Egy public(+) láthatóságú metódus, amely a lény felrobbanását valósítja meg. Itt a saját **die** metódusát hívja meg. Ezt egyedül a Robot fogja felül írni.

8.1.2 Settler

- **Felelősség**

Egy telepest valósít meg és tárolja annak tulajdonságát. Az Entity közvetlen leszármazottja. A telepes egy olyan lény, akit a játékos irányít, emellett a ő az aki szinte mindenre képes, tehát tud bányászni, fúrni és nyersanyagokat tárolni (csak 10-et).

- **Ősosztályok**

Entity.

- **Attribútumok**
  - **gates**: Egy TeleportGate típusú private(-) láthatóságú attribútum, ezt egy tömb valósítja meg, hiszen 0-3 db TeleportGate objektumot tárolhat egyszerre.
  - **myInventory**: Egy Inventory típusú private(-) láthatóságú attribútum, a telepes inventoryját reprezentálja, ebből egy van, de egynek lennie kell. Ebben fognak tárolódni a kibányászott nyersanyagok.
  - **myController**: Egy SettlerController típusú private(-) láthatóságú attribútum, a telepes irányító kontroller, ahhoz kell, hogy a telepes halálánál, innen tudja eltávolítani magát.
  - **robotController**: Egy RobotController típusú private(-) láthatóságú attribútum, a robotokat irányító kontroller, ahhoz kell, hogy robot építés után hozzá tudja adni a vezérlőhöz a robotot.

- **solarSystem**: Egy SolarSystem típusú private(-) láthatóságú attribútum, a napkitörést vezérlő kontroller, ahhoz kell, hogy a lerakott teleport kaput hozzá tudja, adni a naprendszerhez.
- **active**: Egy boolean típusú private(-) láthatóságú attribútum, azt jelzi, hogy a telepes aktív-e, tehát értéke true, ha még nem végzett semmilyen akciót és false, ha már végzett.

- **Metódusok**
  - **void done()**: Egy private(-) láthatóságú metódus, amely a **myController** done nevű metódusát hívja meg és az **active** attribútumot false-ra állítja.
  - **void addMaterial(Material m)**: Egy public(+) láthatóságú metódus, amely megnézi, hogy az **inventory** képes-e eltárolni a kapott **m** nyersanyagot, ezt az **inventory** hasSpace metódus hívásával teszi, ez ha true-val tér vissza, akkor elhelyezi az **m** nyersanyagot az inventoryjában, tehát meghívja az **inventory** addMaterial metódusát aminek argumentumának a kapott **m** nyersanyagot adja. Viszont ha nem tud tárolni többet, az **inventory** hasSpace metódusa false értékkel tért vissza, akkor vissza helyezi a nyersanyagot a **location**-ba tehát ezen meghívja a placeMaterial metódust. Ha valami miatt ez nem valósulhat meg, a **m** material megsemmisül.
  - **void die()**: Egy public(+) láthatóságú metódus, amely meghívja az ős által definiált die metódust, majd ezután eltávolítja magát a **myController**-ből, ezt a **myController** rmSettler metódusával teszi, aminek argumentumának saját magát adja.
  - **void rmMaterial(Material m)**: Egy private(-) láthatóságú metódus, amely eltávolítja az **inventory**-ból az **m** Material-t, mégpedig úgy hogy meghívja az **inventory** rmMaterial metódusát, aminek argumentumának az **m** -et adja.
  - **void addGate(TeleportGate g)**: Egy public(+) láthatóságú metódus, amely a **gates** attribútumhoz adja a kapott **g** TeleportGatet-et.
  - **void putGateDown(TeleportGate g)**: Egy public(+) láthatóságú metódus, amely megvalósítja az adott **g** teleportkapu lerakását. Először is a telepest szomszédául adja a **g** teleportkapunak a **location** Thinget, úgy hogy meghívja a **g** addNeighbour metódusát, amelynek argumentumaként a **location**-t adja, majd ezt fordítva is elvégzi. Majd a **solarSystem**-hez is hozzáadja a **g** kaput a **solarSystem** addThing metódusával, aminek argumentumként a **g**-t adja. Ezután aktiválja a **g** kaput, mégpedig a **g** kapu activate metódusával. Végül eltávolítja a **gates**-ből a **g** kaput, a saját rmGate metódus.
  - **void placeMaterial(Material m)**: Egy public(+) láthatóságú metódus, amely egy nyersanyag visszahelyezést reprezentál. Ezt úgy teszi, hogy elsőnek meghívja a **location** placeMaterial metódusát, ez ha false értékkel tér vissza, akkor nem történik semmi, de ha true értékkel tér vissza, akkor a adott **m** Materialt eltávolítja az **myInventory**-jából, mégpedig a **myInventory** rmMaterial metódusával, aminek argumentumának az **m** -et adja értékül.
  - **void buildRobot()**: Egy public(+) láthatóságú metódus, amely egy robot építést valósít meg. Ellenőrzi, hogy a robothoz szükséges nyersanyagok a **myInventory**-jában van e, ezt a **myInventory** containsRecipe metódusát, aminek argumentumának a Robot receptjét adja (ezt a GameManger **recipes** nevű staticus tagból kéri le), ami ha false-al tér vissza, akkor nem történik semmi. Viszont, ha true-val tér vissza, akkor elkezdődik a Robot építés, tehát a **myInventory**-jából eltávolítja az összes nyersanyagot, ami a Robot-hoz kell, ezt a **myInventory** rmAllFromRecipe metódusával teszi, aminek argumentumként a Robot receptjét adja. Ezután létre hozza a Robot-ot, majd hozzáadja a **robotController**-hez, a **robotController** addRobot metódusával, aminek a létrehozott Robotot adja argumentumként. Végül a **location**-

höz is hozzáadja a frissen készített Robotot, a **location** addEntity metódusával, aminek argumentumként a Robotot adja.

- **void buildGate()**: Egy public(+) láthatóságú metódus, amely teleport kapu páros megépítését reprezentálja. Elsőnek megnézi hogy a **gates**-be fér-e még 2 kapu, ha nem akkor nem csinál semmit, egyébként meg folytatja a készítést. Mégpedig azzal, hogy megnézi, hogy a kellő nyersanyagok megvannak-e a **myInventory**-ban, ezt a **myInventory** containsRecipe metódusát, aminek argumentumának a teleportkapu pár receptjét adja (ezt a GameManger **recipes** nevű staticus tagból kéri le), ami ha false-al tér vissza, akkor nem történik semmi. Viszont, ha true-val tér vissza, akkor elkezdődik a teleportkapu építés, tehát a **myInventory**-jából eltávolítja az összes nyersanyagot, ami a teleportkapu pár-hoz kell, ezt a **myInventory** rmAllFromRecipe metódusával teszi, aminek argumentumként a teleportkapu pár receptjét adja. Ezután létre hozza a TeleportGate-eket, majd hozzáadja a **gates** attribútumhoz őket. Végül összepárosítja őket, mégpedig úgy hogy mindkettőnek beállítja a pair attribútumát a másikra (setPair).
- **void buildBase()**: Egy public(+) láthatóságú metódus, amely a bázis építést reprezentálja. Ezt úgy teszi, hogy végig iterál a **myInventory**-ban lévő material-okon és mindegyiknél meghívja a **location** buildBase metódusát, argumentumként az éppen aktuális materialt, majd ha ez trueval tér vissza, akkor az adott materialt eltávolítja a **myInventory**-ból, annak rmMaterial metódusával, aminek az adott materialt adja argumentum ként. Ha a **location** buildBase false-al tér vissza, akkor nem csinál semmit és folytatja az iterálást.

8.1.3 Robot

- **Felelősség**

Az aszteroida övben mozgó robotokat reprezentálja, illetve azoknak a tulajdonságát tárolja.

- **Ősosztályok**

Entity

- **Attribútumok**
  - **myController**: Egy RobotController típusú private(-) láthatóságú attribútum, a robotot irányító kontroller, ahhoz kell, hogy a robot halálánál, innen tudja eltávolítani magát.

- **Metódusok**
  - **void done()**: Egy private(-) láthatóságú metódus, amely úgy ír felül, hogy ne csináljon smmit.
  - **void die()**: Egy public(+) láthatóságú metódus, amely meghívja az ős által definiált die metódust, majd ezután eltávolítja magát a **myController**-ből, ezt a **myController** rmRobot metódusával teszi, aminek argumentumának saját magát adja.
  - **void mine()**: Egy public(+) láthatóságú metódus, amelyet úgy ír felül, hogy ne csináljon semmit.
  - **void addMaterial(Material m)**: Egy public(+) láthatóságú metódus, amelyet úgy ír felül, hogy ne csináljon semmit.
  - **void explode()**: Egy public(+) láthatóságú metódus, ami a Robot felrobbanását reprezentálja. Ami elsőnek meghívja a **location** randomNeighbour metódusát, majd meghívja a saját move metódusát, aminek argumentumként a visszakapott Thing-et adja.

8.1.4 Ufo

- **Felelősség**

Az aszteroida övben mozgó ufokat reprezentálja, illetve azoknak a tulajdonságát tárolja.

- **Ősosztályok**

Entity

- **Attribútumok**
  - **myController**: Egy UfoController típusú private(-) láthatóságú attribútum, az Ufot irányító kontroller, ahhoz kell, hogy a ufo halálánál, innen tudja eltávolítani magát.

- **Metódusok**
  - **void die()**: Egy public(+) láthatóságú metódus, amely meghívja az ős által definiált die metódust, majd ezután eltávolítja magát a **myController**-ből, ezt a **myController** rmUfo metódusával teszi, aminek argumentumának saját magát adja.
  - **void done()**: Egy private(-) láthatóságú metódus, amely úgy ír felül, hogy ne csináljon smmit.
  - **void drill()**: Egy public(+) láthatóságú metódus, amelyet úgy ír felül, hogy ne csináljon semmit.
  - **void addMaterial(Material m)**: Egy public(+) láthatóságú metódus, amelyet úgy ír felül, hogy ne csináljon semmit.

8.1.5 Thing

- **Felelősség**

Egy absztrakt osztály, amely azokat az objektumokat reprezentálja, amelyekre az entitások (telepesek, robotok, ufo) mozogni tudnak, és tevékenységeket tudnak végrehajtani rajtuk (fúrás, bányászás, nyersanyagelhelyezés).

- **Attribútumok**
  - **neighbour:** Egy Thing[0..*] típusú protected(#) láthatóságú attribútum, amely a vele szomszédos objektumokat tárolja, ebből minimum 1 van, de lehet bármennyi. Ezekre léphet át a telepes, ufo vagy a robot.
  - **entities:** Egy Entity tömb típusú protected(#) láthatóságú attribútum, amely a rajta lévő entitásokat(robot, telepes, ufo) tárolja, ezekből lehet bármennyi, akár 0 is.
  - **mySystem:** Egy SolarSystem típusú private(-) láthatóságú attribútum, amely azt az egy naprendszert tárolja, amiben az objektum megtalálható.

- **Metódusok**
  - **void addEntity (Entity entity):** Egy public(+) láthatóságú metódus, amely egy entitást hozzáad az objektumon tartózkodókhoz, vagyis a megadott **entity** objektumot hozzáadja az **entities** attribútumban található objektumokhoz.
  - **void addNeighbour (Thing nei):** Egy public(+) láthatóságú metódus, amely hozzáad egy objektumot ezen objektum szomszédaihoz, vagyis a megadott **nei** objektumot hozzáadja a **neighbours** attribútumban található objektumokhoz.
  - **void applySunEruption():** Egy public(+) láthatóságú metódus, amely megöl minden rajta álló telepest, ufot és robotot, kivéve ha a Thing egy teljesen megfúrt üreges

aszteroida, ezesetben nem csinál semmit. Amennyiben tehát nem egy teljesen megfúrt üreges aszteroida, akkor az összes **entities** attribútumban található Entity objektumra meghívja a **die** függvényt.

- **void buildBase(Material m):** Egy public(+) láthatóságú metódus, amelyben egy telepes átadja a nála lévő nyersanyagot a kezdő aszteroidának vagyis a megadott **m** Materialt-t hozzáadja a **builtIn** attribútumban található objektumkhoz.
- **void drill():** Egy public(+) láthatóságú metódus, amelyben, egy entitás fúr az aszteroida köpenyén, vagyis a **layer** attribútum értékét csökkenti 1-gyel. Ha ekkor 0 lesz ez az attribútum, akkor ha napközelben van az aszteroida, akkor meghívja a **core** attribútumán a **nearSun** függvényt.
- **Material excavate():** Egy public(+) láthatóságú metódus, amelyben egy entitás kibányássza az aszteroidában lévő nyersanyagot. Ekkor a **core** attribútumot nullra állítja, és visszatér a függvény **core** eredeti értékével.
- **void placeMaterial(Material m):** Egy public(+) láthatóságú metódus, amelyben lerak egy materialt az aszteroida belsejébe a telepes, vagyis a **core** értékét beállítja a megadott **m**-re.
- **Thing randomNeighbour():** Egy public(+) láthatóságú metódus, amely a thing egy random szomszédját adja vissza, vagyis a **neighbours** attribútumban található objektumok közül az egyiket véletlenszerűen kisorsolja, majd a kisorsolt objektumot visszaadja visszatérési értékként.
- **void removeEntity (Entity entity):** Egy public(+) láthatóságú metódus, amely egy entitást eltávolít az objektumról, vagyis az **entities** attribútumban található objektumok közül törli az **entity**-t.
- **void removeNeighbour (Thing nei):** Egy public(+) láthatóságú metódus, amely eltávolít egy objektumot a szomszédok közül, vagyis a **neighbours** attribútumban található objektumok közül törli a **nei**-t.

8.1.6 TeleportGate

- **Felelősség**

A telepesek tudják építeni a teleport kapukat. Mindegyiknek van egy párja a pálya másik pontján, amire át tudnak mozogni 1 lépésben, akkor is, ha nem szomszédosok.

- **Ősosztályok**

Thing → TeleportGate

- **Attribútumok**
  - **pair:** Egy TeleportGate típusú private(-) láthatóságú attribútum, amely A teleportkapu párja, ahova át tud mozogni az entitás. Ebből mindig pontosan 1 van.
  - **normal**: Egy boolean típusú private(-) láthatóságú attribútum, amely azt tárolja, hogy normálisan működik-e a teleportkapu, vagy megbolondította a napvihar.

- **Metódusok**
  - **void activate():** Egy public(+) láthatóságú metódus, amelyben a teleportkapu aktiválódik.
  - **void applySunEruption():** Egy public(+) láthatóságú metódus, amely megöl minden rajta álló telepest, ufot és robotot, vagyis az összes **entities** attribútumban található Entity objektumra meghívja a **die** függvényt.

- **void makeTurn():** Egy public(+) láthatóságú metódus, amelyben ha meg van bolondulva a kapu, akkor elmozog egy szomszédjának egyik szomszédjára (ha magát sorsolja ki akkor nem csinál semmit, vagyis megvan rá az esély, hogy ugyanott marad), és az addigi szomszédaival megszakítja a szomszédságot. Tehát megvizsgálja, hogy a **normal** értéke false-e, ha igen akkor egy szomszédjára meghívja a **randomNeighbour** függvényt, és ha nem magát kapta vissza, törli az összes **neigbours**-ban található objektumot a **removeNeighbour** függvénnyel, valamint ezeknek az objektumoknak a neighbours attribútumából törli szintén a **removeNeighbour** függvénnyel saját magát, majd a kisorsolt új szomszédot az **addNeighbour** függvénnyel szomszéddá teszi, valamint erre a szomszédra is meghívja az **addNeighbour** függvényt hogy saját magát a szomszédjává tegye.

8.1.7 Asteroid

- **Felelősség**

Egy pályán lévő aszteroida, amelyet tudnak fúrni, valamint kibányászni belőle és elhelyezni benne nyersanyagot, emellett a napközelséget is tárolja, és fel is tud robbanni.

- **Ősosztályok**

Thing → Asteroid

- **Attribútumok**
  - **core:** Egy Material típusú protected(#) láthatóságú attribútum, amely az aszteroida belsejében található nyersanyagot tárolja. Maximum 1 lehet belőle, de az is lehet, hogy 0.
  - **layer:** Egy int típusú private(-) láthatóságú attribútum, amely a kőzetréteg aktuális vastagságát tárolja.
  - **nearBySun:** Egy boolean típusú private(-) láthatóságú attribútum, amely tárolja, hogy napközelben van-e az aszteroida.
- **Metódusok**
  - **void applySunEruption():** Egy public(+) láthatóságú metódus, amely megöl minden rajta álló telepest, ufot és robotot, kivéve ha az aszteroida teljesen megfúrt és üreges, ezesetben nem csinál semmit. Tehát először a függvény megvizsgálja, hogy a **layer** értéke nulla-e, ha igen, akkor megvizsgálja, hogy a **core** értéke null-e, és ha az is teljesül, akkor az összes **entities** attribútumban található Entity objektumra meghívja a **die** függvényt.
  - **void buildBase(Material m):** Egy public(+) láthatóságú metódus, amelyben egy telepes átadja a nála lévő nyersanyagot a kezdő aszteroidának vagyis a megadott **m** Materialt-t hozzáadja a **builtIn** attribútumban található objektumkhoz.
  - **void drill():** Egy public(+) láthatóságú metódus, amelyben, egy entitás fúr az aszteroida köpenyén, vagyis a **layer** attribútum értékét csökkenti 1-gyel. Ha ekkor 0 lesz ez az attribútum, akkor ha napközelben van az aszteroida, akkor meghívja a **core** attribútumán a **nearSun** függvényt.
  - **void explode():** Egy public(+) láthatóságú metódus, amelyben, felrobban az aszteroida. Ekkor az aszteroida **entities** attribútumában található összes objektumra meghívja az **explode** függvényt.
  - **Material excavate():** Egy public(+) láthatóságú metódus, amelyben egy entitás kibányássza az aszteroidában lévő nyersanyagot. Ekkor a **core** attribútumot nullra állítja, és visszatér a függvény **core** eredeti értékével.
  - **void placeMaterial(Material m):** Egy public(+) láthatóságú metódus, amelyben lerak egy materialt az aszteroida belsejébe a telepes, vagyis a **core** értékét beállítja a megadott **m**-re.

8.1.8 MainAsteroid

- **Felelősség**

Ez a kezdő aszteroida, ahova a bázist kell megépíteni. Ha a telepesek megépítik rajta a bázist, akkor megnyerik a játékot.

- **Ősosztályok**

Thing → Asteroid → MainAsteroid

- **Attribútumok**
  - **builtIn:** Egy Material típusú private(-) láthatóságú attribútum, amely a kezdő aszteroidán eddig felépített bázisrész, vagyis azok az anyagok, amik már megvannak hozzá. Ez bármennyi lehet, akár 0 is.

- **Metódusok**
  - **void buildBase(Material m):** Egy public(+) láthatóságú metódus, amelyben egy telepes átadja a nála lévő nyersanyagot a kezdő aszteroidának vagyis a megadott **m** Materialt-t hozzáadja a **builtIn** attribútumban található objektumkhoz.

8.1.9 Material

- **Felelősség**

A pályán lévő különböző anyagokat reprezentálja a belőle leszármazó osztályok által.

- **Attribútumok**
  - **myAsteroid**: Egy Asteroid típusú attribútum, ami azt az aszteroidát tárolja, amelyiken van ő.

- **Metódusok**
  - **void nearSun()**: Egy void típusú függvény, ami teljesen üres. A leszármazottja fogják felül definiálni.

8.1.10 Coal

- **Felelősség**

A szén típusú anyagot reprezentálja és segít megkülönböztetni a többi különböző anyagtól.

- **Ősosztályok**

Material.

8.1.11 Iron

- **Felelősség**

A vas típusú anyagot reprezentálja és segít megkülönböztetni a többi különböző anyagtól.

- **Ősosztályok**

Material.

8.1.12 Silicon

- **Felelősség**

A szilícium típusú anyagot reprezentálja és segít megkülönböztetni a többi különböző anyagtól.

- **Ősosztályok**

Material

8.1.13 Uran

- **Felelősség**

Az urán típusú anyagot reprezentálja és segít megkülönböztetni a többi különböző anyagtól. Ezen felül számolja, hogy az adott objektum hányszor volt napközelben és hogyha már harmadjára kerül napközelbe, akkor felrobban.

- **Ősosztályok**

Material.

- **Attribútumok**
  - **countExposition**: Egy integer típusú privát (-) változó ami számolja, hogy az adott objektumot hányszor fedték fel (nem az inventoryban volt) napközelben.

- **Metódusok**
  - **void nearSun()**: Egy publikus (+) láthatóságú metódus, ami eldönti, hogy az anyag felrobban, vagy csak a countExposition attribútumát növeli.

8.1.14 WaterIce

- **Felelősség**

A vízjég típusú anyagot reprezentálja és segít megkülönböztetni a többi különböző anyagtól. Ezen felül ha az anyag napközelbe kerül, akkor elpárologtatja azt.

- **Ősosztályok**

Material.

- **Metódusok**
  - **void nearSun()**: Egy publikus (+) láthatóságú metódus, ami elpárologtatja (eltávolítja az inventoryból) az adott objektumot.

8.1.15 Controller

- **Felelősség**

A program során működő, a rendszer által működtetett/kontrollált osztályokat reprezentálja. Ez egy absztrakt osztály, leszármazottai a különböző controller osztályok. Kapcsolatban áll a GameManager osztállyal is, hiszen ez tudja irányítani.

- **Attribútumok**
  - **manager**: Egy GameManager típusú protected(#) láthatóságú attribútum, amely a játék irányításáért felelős.

- **Metódusok**
  - **void makeTurn():** Egy public(+) láthatóságú metódus, amely elvégzi a leszármaztatott osztályban a kért/kapott műveletet.

8.1.16 SettlerController

- **Felelősség**

A telepesek irányításáért felelős, feldolgozza a felhasználótól kapott információkat. A Controller osztály közvetlen leszármazottja. Emellett felelős azért, hogy tájékoztassa a GameManagert arról, hogy véget ért egy köre a játékosnak.

- **Ősosztályok**

Controller

- **Attribútumok**
  - **settlers**: Egy Settler típusú private(-) láthatóságú attribútum, ezt egy tömb valósítja meg. A játékban található telepesekre vonatkozik ez az attribútum, 0 és akármennyi között lévő Settler objektumot tárolhat egyszerre (attól függően, hogy hány telepes szerepel a játékban).
  - **doneSettlers**: Egy int típusú private(-) láthatóságú attribútum, amely azt jelzi, hogy a játékban szereplő telepesek közül hány telepes végzett már a maga körével. Ha egy telepes végzett, akkor 1-gyel nő az értéke.

- **Metódusok**
  - **void makeTurn()**: Egy public(+) láthatóságú metódus, amelyben lesz egy végtelen ciklus és ez a ciklus endTurn parancsig megy, vagyis amíg véget nem ér a kör. Mindig bekér egy parancsot a felhasználótól és mindig átadja ezt a handleCommand metódusnak. Ellenőrzi, hogy az adott parancs endTurn-e és akkor lesz az endTurn igaz, ha végzett a telepes és ilyenkor meghívja a done metódust.
  - **void handleCommand(String command):** Egy public(+) láthatóságú metódus, amely fogadja a parancsokat, feldolgozza őket és végrehajtja őket.
  - **void done()**: Egy public(+) láthatóságú metódus, amely jelzi hogyha minden telepes befejezte a kört.
  - **void addSettler(Settler s)**: Egy public(+) láthatóságú metódus, amely hozzáad egy **s** telepest a játékhoz.
  - **void rmSettler(Settler s)**: Egy public(+) láthatóságú metódus, amely kitöröl egy **s** telepest a játékból.

8.1.17 RobotController

- **Felelősség**

Az autonóm robotokat irányítja. Minden kör elején egy műveletet kiválaszt minden robotnak a játékban.

- **Ősosztályok**

Controller

- **Attribútumok**
  - **robots**: Egy Robot típusú private(-) láthatóságú attribútum, ezt egy tömb valósítja meg. A játékban található robotokra vonatkozik ez az attribútum, 0 és akármennyi között lévő Robot objektumot tárolhat egyszerre (attól függően, hogy hány robot szerepel a játékban).

- **Metódusok**
  - **void makeTurn()**: Egy public(+) láthatóságú metódus, amely elvégezteti a robotokkal a műveleteket, amikhez meghívja a calculateStep-et.
  - **void calculateStep(Robot r):** Egy public(+) láthatóságú metódus, amely megmondja egy algoritmussal, hogy melyik műveletet végezze a paraméterként megkapott **r** robottal és meg is csináltatja vele.
  - **void addRobot(Robot r)**: Egy public(+) láthatóságú metódus, amely hozzáad egy **r** robotot a játékhoz.
  - **void rmRobot(Robot r)**: Egy public(+) láthatóságú metódus, amely kitöröl egy **r** robotot a játékból.

8.1.18 UfoController

- **Felelősség**

Az ufókat irányítja. Minden kör elején egy műveletet kiválaszt minden ufónak a játékban.

- **Ősosztályok**

Controller

- **Attribútumok**
  - **ufos**: Egy Ufo típusú private(-) láthatóságú attribútum, ezt egy tömb valósítja meg. A játékban található robotokra vonatkozik ez az attribútum, 0 és akármennyi között levő Ufo objektumot tárolhat egyszerre (attól függően, hogy hány ufo szerepel a játékban).

- **Metódusok**
  - **void makeTurn()**: Egy public(+) láthatóságú metódus, amely elvégezteti az ufókkal a műveleteket, amikhez meghívja a calculateStep-et.
  - **void calculateStep(Ufo u):** Egy public(+) láthatóságú metódus, amely megmondja egy algoritmussal, hogy melyik műveletet végezze a paraméterként megkapott **u** ufóval és meg is csináltatja vele.
  - **void addUfo(Ufo u)**: Egy public(+) láthatóságú metódus, amely hozzáad egy **u** ufót a játékhoz.
  - **void rmUfo(Ufo u)**: Egy public(+) láthatóságú metódus, amely kitöröl egy **r** robotot a játékból.

8.1.19 GameManager

- **Felelősség**

A játék menetéért felelős osztály. A játékban lévő Controllereket tárolja és a játék köreit kezeli.

- **Attribútumok**
  - **controllers**: Controller típusú privát (-) attribútum, ami a játékban található Controller típusú objektumokat tárolja egy tömbben.
  - **recipes:** A játékban megépíthető dolgok receptjét tároló statikus HashMap típusú publikus(+) láthatóságú szótár. A receptek nevei, Stringként tárolva, lesznek a kulcsok és a Materialokból álló recept lesz az értéke.
  - **doneControllers:** Integer típusú privát (-) attribútum, ami az adott körben feladatukat befejezett Controllerek számát tárolja.

- **turnNum:** Integer típusú privát (-) attribútum, ami az adott kör sorszámát tárolja.

- **Metódusok**
  - **void jobsDone()**: Ha a doneControllers egyenelő a contrellers attribútumban található Controllerekkel, akkor meghívja a newTurn() függvényt és új kör kezdődik.
  - **void newTurn()**: A doneControllers értékét visszaállítja nullára, növeli a turnNum attribútum értéket, ezzel jelezve, hogy új kör van.
  - **void newGame():** Megkérdezi, hogy hány játékos legyen és új játékot indít.
  - **void win():** Megállítja a játékot és kiírja a képernyőre, hogy a játékosok nyertek. Meghívja a newGame() függvényt.
  - **void lose():** Megállítja a játékot és kiírja a képernyőre, hogy a játékosok veszítettek. Meghívja a newGame() függvényt.

8.1.20 Inventory

- **Felelősség**

Elsődlegesen egyfajta tároló több Material tárolásához, másodlagosan a különböző anyagok összehasonlításához szükséges metódusokat tartalmazza. Ezzel az osztállyal lehet egy lénynél lévő tárolót, egy receptet vagy a MainAsteroidán a még szükséges anyagok tárolását megvalósítani.

- **Attribútumok**
  - **materials**: Egy Matrial[0..*] típusú private(-) láthatóságú attribútum, az **Inventory**-ban tárolt nyersanyagok tárolója. Egy public(+) láthatóságú metódus,

- **Metódusok**
  - **void addMaterial(Material m)**: Egy public(+) láthatóságú metódus, amely egy nyersanyagot ad a tárolóhoz. A kapott **m** Materialt adja hozzá a **materials** listához.
  - **void rmMaterial(Material m)**: Egy public(+) láthatóságú metódus, amely egy nyersanyagot eltávolít a tárolóból. Végig iterál a **materials** listán és megnézi hogy az adott material egyenlő-e a kapott **m**-el, ezt a saját **compareMaterial** metódusával teszi, ha egyenlő akkor kitörli az adott elemet a listából és abba hagyja az iterálást.
  - **void rmAllFromRecipe(Inventory recipe)**: Egy public(+) láthatóságú metódus, amely eltávolítja a kapott **recipe** Inventoryban lévő anyagokat a saját **materials**-ából. Ezt úgy csinálja, hogy végig iterál a **recipe**-ben lévő nyersanyagokon és mindnél meghívja a saját **rmMaterial** metódusát, aminek argumentumként az aktuálisan következő nyersanyagot adja argumentumként.
  - **bool compareMaterial(Material m1, Material m2)**: Egy private(-) láthatóságú metódus, amely két materialt hasonlít össze, ebben az esetben, ez a metódus csak, akkor fog meghívódni, ha a két Material nem azonos (a Materialnak két különböző leszármazottja), így ez simán false értékkel tér vissza.
  - **bool compareMaterial(Coal m1, Coal m2)**: Egy private(-) láthatóságú metódus, ami a **compareMaterial(Material m1, Material m2)** overloadja, amely két materialt hasonlít össze, ebben az esetben, ez a metódus csak, akkor fog meghívódni, ha a két Coal típust kap, amik megegyeznek, így mindig true értékkel tér vissza.
  - **bool compareMaterial(Iron m1, Iron m2)**: Egy private(-) láthatóságú metódus, ami a **compareMaterial(Material m1, Material m2)** overloadja, amely két materialt hasonlít össze, ebben az esetben, ez a metódus csak, akkor fog meghívódni, ha a két Iron típust kap, amik megegyeznek, így mindig true értékkel tér vissza.

- **bool compareMaterial(Silicon m1, Silicon m2)**: Egy private(-) láthatóságú metódus, ami a **compareMaterial(Material m1, Material m2)** overloadja, amely két materialt hasonlít össze, ebben az esetben, ez a metódus csak, akkor fog meghívódni, ha a két Silicon típust kap, amik megegyeznek, így mindig true értékkel tér vissza.
- **bool compareMaterial(Uran m1, Uran m2)**: Egy private(-) láthatóságú metódus, ami a **compareMaterial(Material m1, Material m2)** overloadja, amely két materialt hasonlít össze, ebben az esetben, ez a metódus csak, akkor fog meghívódni, ha a két Uran típust kap, amik megegyeznek, így mindig true értékkel tér vissza.
- **bool compareMaterial(WaterIce m1, WaterIce m2)**: Egy private(-) láthatóságú metódus, ami a **compareMaterial(Material m1, Material m2)** overloadja, amely két materialt hasonlít össze, ebben az esetben, ez a metódus csak, akkor fog meghívódni, ha a két WaterIce típust kap, amik megegyeznek, így mindig true értékkel tér vissza.
- **bool containsRecipe(Inventory recipe)**: Egy public(+) láthatóságú metódus, amely ellenőrzi és visszaadja, hogy a kapott recept tartalma megtalálható-e az inventoryban. Ezt úgy teszi, hogy először lemásolja a receptet. majd levonja a másolatból az összes nyersanyagot, ami a **materials**-ban van, ezt a lemásolt recept **rmAllRecipe** metódusával teszi. Majd megnézi, hogy a lemásolt recept üres-e, ha igen akkor true, egyébként false értékkel tér vissza.
- **bool containsMaterial(Material m)**: Egy public(+) láthatóságú metódus, amely ellenőrzi és visszaadja, hogy a kapott nyersanyag megtalálható-e a **materials**-ban. Végig iterál a **materials** listán és megnézi hogy az adott material egyenlő-e a kapott **m**-el, ezt a saját **compareMaterial** metódusával teszi, ha egyenlő akkor visszatér true értékkel. Ha egyiknél se tért vissza, a végén visszatér false értékkel.

8.1.21 SolarSystem

- **Felelősség**

Magát az aszteroida övet reprezentálja. Itt tárolódnak a Thingek (aszteroidák és teleport kapuk). Emellett ő a felelős a napkitörések létrehozásáért, illetve a már megkergült teleport kapuk mozgatásának triggereléséért.

- **Ősosztályok**

Controller

- **Attribútumok**
  - **things**: Egy Thing[0..*] típusú private(-) láthatóságú attribútum, az aszteroida lévő Thigeket tárolja egy listában.
  - **teleportGates**: Egy TeleportGate[0..*]típusú private(-) láthatóságú attribútum, az aszteroida lévő TeleportGate-eket tárolja, a megkergült TeleportGatek triggeléséért kell, viszont mindegyik benne van.
  - **untilEruption**: Egy int típusú private(-) láthatóságú attribútum, a következő napkitörésig lévő idő (kör), ez -1 ha nincs betervezve napkitörés.
  - **myManager**: Egy GameManager típusú private(-) láthatóságú attribútum, a saját managerének referenciája.

- **Metódusok**
  - **void rmThing(Thin t)**: Egy public(+) láthatóságú metódus, amely a **things**-ből eltávolítja a kapott **t** Thinget.

- **void rmThing(TeleportGate tg)**: Egy public(+) láthatóságú metódus, amely a **things**-ből eltávolítja a kapott **tg** TeleportGate-et, emellett a **teleportGates**-ből is eltávolítja a kapott **tg**-t.
- **void addThing(Thin t)**: Egy public(+) láthatóságú metódus, amely a **things**-hez adja a kapott **t** Thinget.
- **void addThing(TeleportGate tg)**: Egy public(+) láthatóságú metódus, amely a **things**-hez adja a kapott **tg** TeleportGate-et, emellett a **teleportGates**-hez is hozzáadja a kapott **tg**-t.
- **Thing[0..*] makeArea(int r)**: Egy private(-) láthatóságú metódus, amely egy Thing listát generál egy kapott r sugárból. Először random sorsol egy Thinget a **things**-ből, majd ez hozzá adja a visszaadandó listához. Ezek után elkezd futtatni egy BFS algoritmust a kisorsolt helyből kiindulva (a BFS a szomszédokat fogja bejárni). Majd a kapott **r** mélységig futatja a BFS-t, ez alatt a bejárt Thingeket hozzáadja a visszaadandó listához. Amikor vége a BFSnek vissza adja a listát amit csinált.
- **void makeTurn()**: Egy public(+) láthatóságú metódus, amely a SolarSystem körét szimulálja. Elsőnek végig iterál a **teleportGates**-en és minden benne lévő TelepotGate-nek meghívja amakeTurn metódusát. Ezután megnézi, hogy a **untilEruption** -1 e. Ha igen, akkor sorsolással eldönti, hogy tervezzen-e napkitörést vagy ne. Ha arra jut, hogy igen, akkor sorsol egy számot, hogy hány kör múlva legyen a napkitörés és beállítja a **untilEruption**-t erre a számra, viszont ha nem tervez újat, akkor nem csinál semmit. Ha már van betervezve, tehát ha nem -1 az **untilEruption** akkor ezt csökkenti eggyel. Ha a csökkentés után eléri a 0-át, akkor létrejön a napkitörés, ha nem nulla akkor itt se történik semmi. A napkitörés úgy zajlik, hogy meghívja a saját **makeEruption** metódusát. Mindenek után meghívja a **myManager** jobsDone metódusát.
- **void makeTurn()**: Egy public(+) láthatóságú metódus, amely egy napkitörést hajt végre. Elsőnek a saját **makeArea** metódusával létre hoz egy listát, aminek az összes elemének meghívja a applySunEruption metódusát.

8.2 A tesztek részletes tervei, leírásuk a teszt nyelvén.

8.2.1 Build base and win

- **Leírás**

A telepesek megkísérlik a bázis megépítését a náluk lévő nyersanyagokból ezáltal megnyerik a játékot.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszt célja a bázis építés ellenőrzése, és a játék befejezésének tesztelése.

- **Bemenet**

  <SETUP>

  Setboard 1

  main;3;-e;0;1

  Settler s1 main

Settler s2 main

Makematerial -c c1

Makematerial -c c2

Makematerial -c c3

Setinventory s1 c1

Setinventory s1 c2

Setinventory s1 c3

Makematerial -i i1

Makematerial -i i2

Makematerial -i i3

Setinventory s1 i1

Setinventory s1 i2

Setinventory s1 i3

Makematerial -s si1

Makematerial -s si2

Makematerial -s si3

Setinventory s1 si1

Setinventory s1 si2

Setinventory s1 si3

Makematerial -u u1 0

Makematerial -u u2 1

Makematerial -u u3 1

Setinventory s2 u1

Setinventory s2 u2

Setinventory s2 u3

Makematerial -w w1

Makematerial -w w2

Makematerial -w w3

Setinventory s2 w1

Setinventory s2 w2

Setinventory s2 w3

</SETUP>

<PROGRESS>

Buildbase s1

Buildbase s2

List

</PROGRESS>

- **Elvárt kimenet**

  +------------------+

  name: main

  neighbours: null

  entities: s1 s2

  layer number: 3

  core: null

  nearsun: false

  +------------------+

  name: s1

  materials: null

gates: null

location: main

stepped: true

+------------------+

name: s2

materials: null

gates: null

location: main

stepped: true

+------------------+

game win

8.2.2 Settler build a robot, all condition set

- **Leírás**

A telepes megkísérel egy robot megépítését úgy, hogy minden körülmény adott.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A telepes robotépítési funkcióját teszteli úgy, hogy ahhoz minden körülmény adott.

- **Bemenet**

<SETUP>

Setboard 1

a1;2;-e;0;0

Settler s1 a1

Makematerial -i i1

Makematerial -c c1

Makematerial -u u1

Setinventory s1 i1

Setinventory s1 c1

Setinventory s1 u1

</SETUP>

<PROGRESS>

Buildrobot r1 s1

List

</PROGRESS>

- **Elvárt kimenet**

+------------------+

name: a1

neighbours: null

entities: s1 r1

layer number: 2

core: null

nearsun: false

+------------------+

name: s1

materials: null

gates: null

location: a1

stepped: true

+------------------+

name: r1

location: a1

stepped: false

8.2.3 Settler build a robot, without materials

- **Leírás**

A telepes megkísérel egy robot megépítését úgy, hogy nincs nála semmilyen szükséges nyersanyag

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A telepes robotépítési funkcióját teszteli úgy, hogy nem lesz képes azt végrehajtani és megépíteni a robotot

- **Bemenet**

  <SETUP>

  Setboard 1

  a1;2;-e;0;0

  Settler s1 a1

  </SETUP>

  <PROGRESS>

  Buildrobot r1 s1

  List

  </PROGRESS>

- **Elvárt kimenet**

  Robot can't be created

  +------------------+

  name: a1

  neighbours: null

  entities: s1

  layer number: 2

  core: null

  nearsun: false

  +------------------+

  name: s1

  materials: null

  gates: null

  location: a1

  stepped: true

8.2.4 Settler build gate, all condition set

- **Leírás**

A telepes megkísérel egy kapupár építését a nála lévő nyersanyagokból.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A tesztelés során lehetőség lesz a kapukészítés mechanizmusát letesztelni, és hogy a kapuk párba állítása megtörtént e. Hiba lehet, ha nincs megfelelő nyersanyag, vagy a kapuk nincsenek párba.

- **Bemenet**

  <SETUP>

  Setboard 1

  a1;2;-e;0;0

  Settler s1 a1

  Makematerial -i i1

  Makematerial -i i2

  Makematerial -u u1

  Makematerial -w w1

  Setinventory s1 i1

  Setinventory s1 c2

  Setinventory s1 u1

  Setinventory s1 w1

  </SETUP>

  <PROGRESS>

  Buildgate g1 s1

  List

  </PROGRESS>

- **Elvárt kimenet**

  +------------------+

  name: a1

neighbours: null

entities: s1

layer number: 2

core: null

nearsun: false

+------------------+

name: s1

materials: null

gates: g1a g2b

location: a1

stepped: true

8.2.5 Settler build gate, without materials

- **Leírás**

A telepes egy kapupárt próbál építeni hogy nincs meg hozzá a megfelelő nyersanyagja

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszt azt ellenőrzi, hogy mi történik ha a telepes nem tudja megépíteni a kaput mert nincsenek nyersanyagjai.

- **Bemenet**

<SETUP>

Setboard 1

a1;2;-e;0;0

Settler s1 a1

</SETUP>

<PROGRESS>

Buildgate g1 s1

List

</PROGRESS>

- **Elvárt kimenet**

Gates cant't be created

+------------------+

name: a1

neighbours: null

entities: s1

layer number: 2

core: null

nearsun: false

+------------------+

name: s1

materials: null

gates: null

location: a1

stepped: true

8.2.6 Settler tries to drill, but no layer left

- **Leírás**

A telepes megpróbál megfúrni egy olyan aszteroidát aminek már nincsenek rétegei. Ez nem lesz lehetséges, ezért egy hibaüzenetet fog kiírni, hogy nincs több réteg.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Az aszteroidák rétegeit teszteli, hogy képes e a telepes akkor is fúrni, ha már nincs réteg amin ezt megteheti. Ezért hibát fog visszadobni a hiányzó rétegre.

- **Bemenet**

<SETUP>

Setboard 1

a1;0;-i;i1;0;0

Settler s1 a1

</SETUP>

<PROGRESS>

Drill s1

List

</PROGRESS>

- **Elvárt kimenet**

  There is no layer

  +------------------+

  name: a1

  neighbours: null

  entities: s1

  layer number: 0

  core: i1

  nearsun: false

  +------------------+

  name: s1

  materials: null

  gates: null

  location: a1

  stepped: false

8.2.7 Settler drill the last layer of IceWater near sun

- **Leírás**

A telepes egy vízjég aszteroida utolsó rétegét fúrja meg ami napközelben van. Ezáltal a benne lévő vízjég elpárolog.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A vízjég elpárolgását teszteli, ha napközelben megfurják az utolsó rétegét az aszteroidának. Ezáltal aktiválódik, hogy automatikusan történjen meg a napközeli esemény. Hiba lehet ha nem aktiválódik a napközeli esemény

- **Bemenet**

  <SETUP>

  Setboard 1

a1;1;-w;w1;1;0

Settler s1 a1

</SETUP>

<PROGRESS>

Drill s1

List

</PROGRESS>

- **Elvárt kimenet**

+------------------+

name: a1

neighbours: null

entities: s1

layer number: 0

core: null

nearsun: true

+------------------+

name: s1

materials: null

gates: null

location: a1

stepped: true

8.2.8 Settler place random material, not near sun

- **Leírás**

A telepes egy akármilyen nyersanyagot visszahelyez egy előre megfúrt üreges aszteroidába.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Ez a teszt azt ellenőrzi hogy vissza lehet e helyezni nyersanyagot egy aszteroida magjába, ha már azt előtte megfúrták. Lehetséges hiba, ha az aszteroida nem üres.

- **Bemenet**

<SETUP>

Setboard 1

a1;0;-e;0;0

Settler s1 a1

Makematerial -c c1

Setinventory s1 c1

</SETUP>

<PROGRESS>

Putdown c1 s1

List

</PROGRESS>

- **Elvárt kimenet**

+------------------+

name: a1

neighbours: null

entities: s1

layer number: 0

core: c1

nearsun: false

+------------------+

name: s1

materials: null

gates: null

location: a1

stepped: true

8.2.9 Settler place Uran back near sun, it’s the Uran’s third time

- **Leírás**

A telepes visszahelyez a napközelben lévő aszteroida belsejébe egy olyan uránt ami már 2x volt napközelbe és mivel ez lesz a harmadig napközele az fel fog robbanni.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszt azt ellenőrzi, hogy az urán felrobban e a harmadik alkalomra, ha napközelbe ér és hogy az ezt követő események bekövetkeznek e. A telepeseknek és az aszteroidának meg kell semmisülniük.

- **Bemenet**

  <SETUP>

  Setboard 2

  a1;0;-e;1;0

  a2;2;-i;i1;0;0

  Setnei 1

  a1;a2

  Settler s1 a1

  Makematerial -u u1 2

  Setinventory s1 c1

  </SETUP>

  <PROGRESS>

  Putdown u1 s1

  List

  </PROGRESS>

- **Elvárt kimenet**

  +------------------+

  name: a2

  neighbours: null

  entities: null

  layer number: 2

  core: i1

  nearsun: false

8.2.10 Settler place WaterIce back near sun

- **Leírás**

A telepes visszahelyez egy napközelben lévő üreges aszteroida belsejébe egy vízjég anyagot. Ezáltal az elpárolog.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Ellenőrzi, hogy a vízjég képes e elpárologni napközelben.

- **Bemenet**

  <SETUP>

  Setboard 1

  a1;0;-e;1;0

  Settler s1 a1

  Makematerial -w w1

  Setinventory s1 w1

  </SETUP>

  <PROGRESS>

  Putdown w1 s1

  List

  </PROGRESS>

- **Elvárt kimenet**

  +------------------+

  name: a1

  neighbours: null

  entities: s1

  layer number: 0

  core: null

  nearsun: true

  +------------------+

  name: s1

materials: null

gates: null

location: a1

stepped: true

8.2.11 Settler mines core

- **Leírás**

A telepes megkísérli kibányászni, egy előre megfúrt aszteroida magját, ami nincs napközelben.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Ellenőrzi, hogy a telepes ki tudja e bányászni az aszteroida magját.

- **Bemenet**

<SETUP>

Setboard 1

a1;0;-i;i1;0;0

Settler s1 a1

</SETUP>

<PROGRESS>

Mine s1

List

</PROGRESS>

- **Elvárt kimenet**

+------------------+

name: a1

neighbours: null

entities: s1

layer number: 0

core: null

nearsun: false

+------------------+

name: s1

materials: i1

gates: null

location: a1

stepped: true

8.2.12 Settler tries to mine empty core

- **Leírás**

A telepes megpróbál bányászni egy olyan aszteroidán, aminek nincsen már magja.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Ellenőrzi hibát dob e a program, ha olyan aszteroidán próbálunk bányászni, aminek nincsen már magja.

- **Bemenet**

<SETUP>

Setboard 1

a1;0;-e;0;0

Settler s1 a1

</SETUP>

<PROGRESS>

Mine s1

List

</PROGRESS>

- **Elvárt kimenet**

Core is empty

+------------------+

name: a1

neighbours: null

entities: s1

layer number: 0

core: null

nearsun: false

+------------------+

name: s1

materials: null

gates: null

location: a1

stepped: true

8.2.13 Settler put down first gate

- **Leírás**

A telepes lehelyezi egy kapupár egyik tagját úgy, hogy a másik még nincs lerakva, hanem az a zsebében van.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A tesztel ellenőrizhető, hogy a kapu a megfelelő helyre ált e be és a megfelelő szomszédot megkapta e.

- **Bemenet**

<SETUP>

Setboard 1

a1;0;-e;0;0

Settler s1 a1

Makegate g1 g2

Setinventory s1 g1

Setinventory s1 g2

</SETUP>

<PROGRESS>

Putdown g1 s1

List

</PROGRESS>

- **Elvárt kimenet**

  +------------------+

  name: a1

  neighbours: g1

  entities: s1

  layer number: 0

  core: null

  nearsun: false

  +------------------+

  name: s1

  materials: null

  gates: g2

  location: a1

  stepped: true

  +------------------+

  name: g1

  neighbours: a1

  entities: null

  pair: g2

  setted: true

  active: false

  +------------------+

  name: g2

  neighbours: null

  entities: null

  pair: g1

setted: false

active: false

8.2.14 Settler put down second gate

- **Leírás**

A telepes lehelyezi egy kapupár második tagját úgy, hogy már a másik kapu is le van helyezve.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszt ellenőrzi, hogy a kapuk között megfelelően létre jött e a kapcsolat és automatikusan beaktiválódta e a kapuk, továbbá, hogy mindegyik kapu a megfelelő szomszédokkal rendelkezik.

- **Bemenet**

<SETUP>

Setboard 2

a1;0;-e;0;0

a2;0;-e;0;0

Settler s1 a2

Makegate g1 g2

Setinventory s1 g2

Setnei 2

a1;a2

a1;g1

</SETUP>

<PROGRESS>

Putdown g2 s1

List

</PROGRESS>

- **Elvárt kimenet**

+------------------+

name: a1

neighbours: a2 g1

entities: null

layer number: 0

core: null

nearsun: false

+------------------+

name: a2

neighbours: a1 g2

entities: s1

layer number: 0

core: null

nearsun: false

+------------------+

name: s1

materials: null

gates: g2

location: a2

stepped: true

+------------------+

name: g1

neighbours: a1

entities: null

pair: g2

setted: true

active: true

+------------------+

name: g2

neighbours: a2

entities: null

pair: g1

setted: true

active: true

8.2.15 Settler move to an active TeleportGate

- **Leírás**

A teleportálás funkcióját teszteli. Ha a telepes rálép az egyik kapura akkor a másik kapura kell, hogy kerüljön ezáltal, ahonnan tovább tud lépni annak szomszédjaira.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A teszt ellenőrzi, hogy a kapuk párban állnak e és megvalósul e a teleportálás mechanizmusa automatikusan ha a telepes rálép az egyik kapura.

- **Bemenet**

<SETUP>

Setboard 2

a1;0;-e;0;0

a2;0;-e;0;0

Settler s1 a1

Makegate g1 g2

Setnei 2

a1;a2

a1;g1

a2;g2

</SETUP>

<PROGRESS>

Move s1 g1

List

</PROGRESS>

- **Elvárt kimenet**

  +------------------+

  name: a1

  neighbours: a2 g1

  entities: null

  layer number: 0

  core: null

  nearsun: false

  +------------------+

  name: a2

  neighbours: a1 g2

  entities: null

  layer number: 0

  core: null

  nearsun: false

  +------------------+

  name: s1

  materials: null

  gates: null

  location: g2

  stepped: true

  +------------------+

  name: g1

  neighbours: a1

  entities: null

  pair: g2

  setted: true

active: true

+------------------+

name: g2

neighbours: a2

entities: s1

pair: g1

setted: true

active: true

8.2.16 Settler move to an inactive TeleportGate

- **Leírás**

Egy kapupár közül csak az egyik van lehelyezve és a teszt azt nézi, ha igy lép a telepes rá a kapura, akkor nem szabad, hogy bárhova is elteleportáljon.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A kapu inaktivitását teszteli, hogy ha nincs lerakva a kapu párja akkor nem működik a funkciója sem.

- **Bemenet**

<SETUP>

Setboard 2

a1;0;-e;0;0

a2;0;-e;0;0

Settler s1 a1

Makegate g1 g2

Setinventory s1 g2

Setnei 2

a1;a2

a1;g1

</SETUP>

<PROGRESS>

Move s1 g1

List

</PROGRESS>

- **Elvárt kimenet**

  +------------------+

  name: a1

  neighbours: a2 g1

  entities: null

  layer number: 0

  core: null

  nearsun: false

  +------------------+

  name: a2

  neighbours: a1

  entities: null

  layer number: 0

  core: null

  nearsun: false

  +------------------+

  name: s1

  materials: null

  gates: g2

  location: g1

  stepped: true

  +------------------+

  name: g1

  neighbours: a1

entities: s1

pair: g2

setted: true

active: false

+------------------+

name: g2

neighbours: null

entities: null

pair: g1

setted: true

active: false

8.2.17 Settler move to Asteroid

- **Leírás**

A telepes egyik aszteroidáról egy másikra való mozgását nézi.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A mozgás funkcióját teszteli, hogy a telepes olyan aszteroidák között tud mozogni, amik szomszédjai egymásnak. Hibás a teszt, ha olyan aszteroidák között akarunk mozogni, amik nincsenek párban.

- **Bemenet**

<SETUP>

Setboard 2

a1;0;-e;0;0

a2;0;-e;0;0

Settler s1 a1

Setnei 1

a1;a2

</SETUP>

<PROGRESS>

Move s1 a2

List

</PROGRESS>

- **Elvárt kimenet**

+------------------+

name: a1

neighbours: a2

entities: null

layer number: 0

core: null

nearsun: false

+------------------+

name: a2

neighbours: a1

entities: s1

layer number: 0

core: null

nearsun: false

+------------------+

name: s1

materials: null

gates: null

location: a2

stepped: true

8.2.18 Settler tries to pick up material, but the inventory is full

- **Leírás**

A telepes megpróbál nyersanyagot az inventoryjába rakni, de az már teli van, ezért hibát fog dobni.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A telepes inventoryját teszteli, hogy mi történik ha már teli inventoryba szeretnénk nyersanyagot rakni.

- **Bemenet**

  <SETUP>

  Setboard 1

  a1;0;-i;i1;0;0

  Settler s1 a1

  Makematerial -c c1

  Makematerial -c c2

  Makematerial -c c3

  Makematerial -c c4

  Setinventory s1 c1

  Setinventory s1 c2

  Setinventory s1 c3

  Setinventory s1 c4

  Makematerial -i i2

  Makematerial -i i3

  Makematerial -i i4

  Setinventory s1 i2

  Setinventory s1 i3

  Setinventory s1 i4

  Makematerial -s si1

  Makematerial -s si2

  Makematerial -s si3

  Setinventory s1 si1

Setinventory s1 si2

Setinventory s1 si3

</SETUP>

<PROGRESS>

Mine s1

List

</PROGRESS>

- **Elvárt kimenet**

Inventory is full

+------------------+

name: a1

neighbours: null

entities: s1

layer number: 0

core: i1

nearsun: false

+------------------+

name: s1

materials: c1 c2 c3 c4 i2 i3 i4 si1 si2 si3

gates: null

location: a1

stepped: true

8.2.19 Apply sun erruption

- **Leírás**

Egy napkitörést szimulál úgy, hogy a telepesek, robotok nincsenek menedékben ezáltal el fognak pusztulni, továbbá a kapuk megkergülnek e.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A napkitörés mechanizmusát teszteli, hogy ha az általa érintett aszteroidákon állnak entitások, amik nincsenek fedezékben akkor meg fognak e automatikusan halni. A kapuk megkergülésének a mechanizmusát is teszteli, hogy ha napvihart kapott a kapu akkor a következő lépéseben egy másik aszteroida mellé kell hogy kerüljön.

- **Bemenet**

  <SETUP>

  Setboard 4

  a1;2;-i;i1;0;0

  a2;3;-c;c1;0;0

  a3;2;-w;w1;0;0

  a4;3;-s;si1;0;0

  Makegate g1 g2

  Setnei 6

  a1;a2

  a2;a3

  a2;a4

  a3;a4

  a2;g1

  a3;g2

  Settler s1 a2

  Settler s2 a1

  Settler s3 a3

  Robot r1 a3

  Robot r2 a4

  Setrandom 2

  </SETUP>

  <PROGRESS>

  Makeeruption a2 2

  List

</PROGRESS>

- **Elvárt kimenet**

  +------------------+

  name: a1

  neighbours: a2 g1

  entities: null

  layer number: 2

  core: i1

  nearsun: false

  +------------------+

  name: a2

  neighbours: a1 a3 a4

  entities: null

  layer number: 3

  core: c1

  nearsun: false

  +------------------+

  name: a3

  neighbours: a2 a4

  entities: null

  layer number: 2

  core: w1

  nearsun: false

  +------------------+

  name: a4

  neighbours: a2 a3 g2

  entities: null

layer number: 3

core: si1

nearsun: false

+------------------+

name: g1

neighbours: a1

entities: null

pair: g2

setted: true

active: true

+------------------+

name: g2

neighbours: a4

entities: null

pair: g1

setted: true

active: true

8.2.20 Apply sun erruption with entities in cover

- **Leírás**

Szintén egy napkitörést szimulál, de most azzal a körülményekkel, hogy az entitások el vannak bújva az aszteroidák előre kibányászott üregében. Továbbá a kapuknak szintén meg kell kergülniük.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

A napkitörés mechanizmusát teszteli, de most úgy, hogy az entitások el vannak bújva ezáltal nem fognak meghalni.

- **Bemenet**

<SETUP>

Setboard 4

a1;2;-i;i1;0;0

a2;0;-e;0;0

a3;0;-e;0;0

a4;3;-s;si1;0;0

Makegate g1 g2

Setnei 6

a1;a2

a2;a3

a2;a4

a3;a4

a2;g1

a3;g2

Settler s1 a2

Settler s2 a1

Settler s3 a3

Robot r1 a3

Robot r2 a4

Setrandom 2

</SETUP>

<PROGRESS>

Makeeruption a2 2

List

</PROGRESS>

- **Elvárt kimenet**

  +------------------+

  name: a1

  neighbours: a2 g1

  entities: null

layer number: 2

core: i1

nearsun: false

+------------------+

name: a2

neighbours: a1 a3 a4

entities: s1

layer number: 0

core: null

nearsun: false

+------------------+

name: a3

neighbours: a2 a4

entities: s3 r1

layer number: 0

core: null

nearsun: false

+------------------+

name: a4

neighbours: a2 a3 g2

entities: null

layer number: 3

core: si1

nearsun: false

+------------------+

name: g1

neighbours: a1

entities: null

pair: g2

setted: true

active: true

+------------------+

name: g2

neighbours: a4

entities: null

pair: g1

setted: true

active: true

8.2.21 Asteroid explodes

- **Leírás**

Az aszteroidák felrobbanását teszteli egy urán hatására. Az aszteroidáknak meg kell semmisülniük és a rajtuk lévő telepeseknek is meg kell halniuk, a robotoknak pedig egy szomszédos aszteroidára kell kerülniük.

- **Ellenőrzött funkcionalitás, várható hibahelyek**

Az aszteroida felrobbanását és azt követő hatásokat teszteli a különböző entitásokra. A telepeseknek meg kell halniuk a robotoknak viszont nem, de egy másik aszteroidára kell, hogy kerüljenek. Az aszteroida csak akkor robbanhat fel ha a magjában lévő urán már harmadszor kapott napközelt, ezáltal az felrobbantja.

- **Bemenet**

<SETUP>

Setboard 2

a1;0;-e;1;0

a2;2;-i;i1;0;0

Setnei 1

a1;a2

Settler s1 a1

Robot r1 a1

Makematerial -u u1 2

</SETUP>

<PROGRESS>

Putdown u1 s1

List

</PROGRESS>

- **Elvárt kimenet**

  +------------------+

  name: a2

  neighbours: null

  entities: r1

  layer number: 2

  core: i1

  nearsun: false

  +------------------+

  name: r1

  location: a2

  stepped: false

8.3 A tesztelést támogató programok tervei

A tesztelést egy erre dedikált osztály fogja megvalósítani, ami a bemenetről kapott parancsoktól függően fogja a program megfelelő részeit meghívni és az elszigetelt tesztelési környezetet biztosítani minden egyes teszt számára. A bemenet kezelését is ő fogja megvalósítani.

8.4 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2021.04.06. 17:00	3 óra	Czanik Nagy Novák Szabó Tokovics	Értekezlet: Az előző beadandó értékelésének átbeszélése. Az eheti beadandó felvázolása, ötletek átbeszélése, feladatok kiosztása. Döntés: Czanik csinálja a tesztesetek kidolgozását. Döntés: Nagy, Novák, Szabó és Tokovics csinálja az osztályok leírását szétosztva.
2021.04.07. 14:00	5 óra	Czanik	Tevékenység: Tesztesetek kidolgozása, bemenetek és várt kimenetek megfogalmazása az általunk megfogalmazott nyelven
2021.04.08. 12:00	4 óra	Szabó	Tevékenység: Osztályok leírása
2021.04.09. 19:00	2,5 óra	Novák	Tevékenység: Osztályok leírása

2021.04.10. 10:00	2 óra	Nagy	Tevékenység: Osztályok leírása
2021.04.10. 12:00	2 óra	Tokovics	Tevékenység: Osztályok leírása
2021.04.10. 13:00	2 óra	Czanik	Tevékenység: 8.2 – 8.3 Tesztesetek dokumentumba való átírása és leírásuk, funkcionalitásuk megfogalmazása.
2021.04.10. 17:00	4 óra	Szabó	Tevékenység: A tesztesetek átnézése, javítása
2021.04.11. 17:00	1 óra	Czanik	Tevékenység: Módosítások, változtatások kiegészítése az előző heti dokumentumban.

2021.04.11. 22:00	3 óra	Czanik Novák Szabó	Tevékenység: A dokumentum összeállítása, átnézése és az utolsó simítások, módosítások elvégzése.