****

Your task is to develop an application that is capable of controlling your colony without any human interaction. It should be capable of scouting and conquering new planets, defending your own planets, and destroying your enemies. The level is 2D (the size of the level is known when the game starts); at the beginning there can be neutral, AI controlled, and player controlled planets. The players have the possibility to split a ship off the planets, and then send the ships into battle. The non-neutral planets are continuously growing, the power level of the new ships have to be subtracted from the power level of the planet. The power level is represented by an integer.

Each team has to prepare the followings:

* a client application, that is attached to the server as a DLL file (so there is no actual TCP/IP communication, the „client” is essentially a class, that has methods constantly called by the „server”. The client has no other way to communicate.);
* suitable methods that interpret the data received from the server;
* suitable methods to be able to control the game;
* an algorithm to perform every controls automatically.

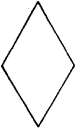
The following commands are interpreted by the server

**SPLIT ItemID NumberOfUnits // Split unit**

**MOVE ItemID TargetX TargetY // Movement. Negative = STOP**

**SHOOT ItemID OtherItemID NumberOfUnits  
 // Shoot. If the owner of the two units are the same, then transfer energy/merge units.  
 // Default shooting range: 2 map units**

Important information:

* **Try out the game where the idea came from: http://www.kongregate.com/games/badben/nano-war**
* **Every unit (ship, planet) has a unique identifier; it is used for referencing them in the commands.**
* **The players do not see the whole level (Fog of War): Around every ship, there is a visibility   
  field of 2 units; around every planet, this radius is 4. The visibility field is not circular, rather  
  a rhomboid (see the image on the right). As a result, the diagonal attack is harder to detect -   
  this might be useful!**
* **One game item has the following data fields: Type of the item; Unique identifier; Nickname of the   
  owner; X position; Y position; Number of units.**
* **For planets, extra data: IncTime (the amount of seconds required to increase the power level).**
* **For ships, extra data: Target X coordinate; Target Y coordinate**

Points awarded ont he levels

* Successful shots: **0.5 points/shot**
* capturing 1 planet: **100 points**
* losing 1 planet: **-150 points**
* at the end, for every planet owned: **200 points/planet**
* at the end, for every unit owned: **1 point/unit**

During the competition

* We want the teams to give us a managed .NET DLL.
* On the day of the competition, there will be a trial run; then the teams will have 2 hours to change their codes. After that, there will be a knockout phase to decide the winner team.

Feladatotok elkészíteni egy olyan alkalmazást, ami képes emberi beavatkozás nélkül irányítani a kolóniáitokat, felismerni és elfoglalni további bolygókat, megvédeni a saját bolygóitokat és elpusztítani az ellenségeiteket. A pálya kétdimenziós (a szerver előre elküldi a pálya méretét); kezdetben semleges, AI irányítású, illetve játékosok által irányított bolygók helyezkednek el rajta. A játékosoknak lehetőségük van a bolygókról hajókat leválasztani, és azokat csatába küldeni. A bolygók (a semlegesek kivételével) az idő folyamán folyamatosan növelik az energiaszintjüket, a létrehozott hajók energiaszintjét a bolygók energiájából kell levonni. Az energiaszintet egy egész szám jelképezi.

Készítsétek el a következőket:

* saját kliensalkalmazást, ami DLL formában csatolódik a szerverhez (tehát nincs tényleges TCP/IP kommunikáció, a „kliens” valójában egy osztály, aminek a metódusait a „szerver” folyamatosan hívogatja. A kliens más módon nem tud kommunikálni.);
* a megfelelő feldolgozó rutinokat, ami képes értelmezni a fogadott adatokat a szervertől;
* a parancsok kiadásához szükséges függvényeket, hogy lehetséges legyen irányítani a játékot;
* a játék automata irányításához szükséges algoritmust.

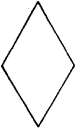
Szerver által értelmezett utasítások

**SPLIT ItemID NumberOfUnits // Egység szétdarabolása**

**MOVE ItemID TargetX TargetY // Mozgatás. Negatív = STOP**

**SHOOT ItemID OtherItemID NumberOfUnits  
 // Lövés. Ha a két egység tulajdonosa azonos játékos, akkor egyesíti a két egységet  
 // Alapértelmezett lőtávolság: 2 egység**

Fontos információk:

* **Próbáljuk ki a játékot, ahonnan az ötlet jött: http://www.kongregate.com/games/badben/nano-war**
* **Minden elemnek (hajó, bolygó) egyedi azonosítója van: az utasításokban ezzel hivatkozunk rájuk.**
* **A játékosok nem az egész pályát látják (Fog of War): Minden hajó körül alapértelmezetten 2,   
  minden bolygó körül 4 egységnyi a láthatóság. A láthatósági maszk nem kör, hanem rombusz  
  alakú (lásd kép). Ennek köszönhetően az átlós támadás nehezebben detektálható, mint a   
  vízszintes/függőleges – ez esetleg kihasználható.**
* **Egy játékelem a következő adatokkal rendelkezik: az elem típusa; Egyedi azonosító; a   
  tulajdonos felhasználó neve; X pozíció; Y pozíció; Egységek száma.**
* **Bolygók esetén, extra adat: IncTime (ennyi másodperc kell az energiaszint növeléséhez.**
* **Hajók esetén, extra adat: Célpont X koordináta; Célpont Y koordináta**

Verseny pontozása pályánként:

* Leadott lövések: **0.5 pont/lövés**
* 1 bolygó elfoglalása: **100 pont**
* 1 bolygó elvesztése: **-150 pont**
* pálya befejezése utáni bolygók száma: **200 pont/bolygó**
* pálya befejezése utáni egységek száma: **1 pont/egység**

Verseny folyamata:

* A csapatoktól egy felügyelt .NET DLL –t kérünk.
* A csapatoknak a verseny napján próbakör után lesz 2 órájuk módosítani a kódjukat, majd utána egyenes kieséses szakaszban derül ki, hogy ki a győztes.