2021.01.15.

UNO DOKUMENTÁCIÓ

Leírása a „megmérettetésnek” általánosságban

1.1 A játékosok célja az, hogy minél jobb Uno-zó programot írjanak. A logikát kell „csak” megírniuk, mert a többi részét a programnak készen kapják, mint valami sablont. A programokat ki lehet próbálni a szerver robot játékosai (botok), vagy akár más játékosok ellen is. Valós játékosok játszhatnak úgy is egymással, hogy van pluszban néhány bot köztük.

1.2 A programoknak nincs semmilyen megkötésük, hogy miket kéne vagy épp nem kéne tartalmazniuk. A játék a szervernek küldött és a szervertől kapott üzenetek alapján bonyolítódik le. A szervernek küldött adatokat a szerver ellenőrzi mindig, hogy az adott helyzetben helytállnak-e vagy sem. Ha nem megfelelő, akkor a játék leáll és egy hibaüzenetben értesül az illető, hogy mit rontott el. A többi játékos nem szükségesen értesül a hibáról, csak leáll nekik a játék, és kilép a programjuk. Hibát véteni tehát nem lehet, mert akkor leáll a játék. Hibának számít továbbá az is, ha a játékos nem küld üzenetet 1, azaz egy másodpercen belül (ami rengeteg idő).

1.3 A csatolt kliens program majdnem a lehető legbutább „brute force”-t alkalmazza. Azzal lehet megnézni hogy pontosan hogyan is működik a játék. Legalábbis, a szerver és a játékos közti üzenetváltás működését azon könnyen lehet végig követni. (Ehhez a programhoz hasonlóak a szerver robotjai.) Akár internet nélkül is tesztelhető a megírt program, a csatolt szerver programmal. Ez a program csak olyan szobákat képes létrehozni, ahol csak egy valódi játékos van, és a többi robot.

Játékszabályok

2.1 Az Uno-t rengeteg módon játszák, ezért itt leírom a játékszabályokat biztos, ami biztos. Fontos eltérés az eredeti játéktól, hogy itt nem kell azt mondani, hogy „Uno”, mert programok között ennek nincs értelme. Nincs csapás, azaz nem lehet lerakni egy színre és típusra is megegyező lapot, ha nem te vagy a soron következő játékos. Alapvetőleg sorban jönnek a játékosok, akik az utolsó lap színével vagy típusával egyező lapot rakhatnak le, illetve fekete „Joker” lapot. Ha valaki nem tud rakni, vagy szimplán csak húzni szeretne, akkor húz egy lapot.

2.2 A nem számos lapokról pár szót ejtenék: ha valaki letesz „Kimaradsz!” lapot, akkor a következő játékos kimarad. Ha „Fordulj!” lapot játsz ki valaki, akkor megfordul a játék iránya. Az előző két típusú lapot, ha akkor játsza ki a játékos, amikor már csak ketten vannak, akkor ő jön még egyszer. Ha plusz kettest rak a játékos, akkor a következő játékosnak kötelező pluszos lapot raknia, ha nem rak, akkor meg fel kell húznia annyi lapot amennyit az egymás után lerakott plusz-os lapok összegével egyenlő. Fontos, hogy plusz kettesre lehet rakni plusz négyest, de fordítva nem lehet. Akkor sem, ha olyan színt kér, amilyen plusz kettest pont tudna rakni a következő játékos.

2.3 A „Joker”, azaz fekete lapokat amikor soron vagyunk kijátszhatjuk mindig, kivéve a színkérőt, ha plusz-os lapot kell rakni éppen. A színkérős lappal kötelező a négy szín közül valamelyiket kérni. A plusz négyes lappal is lehet színt kérni, de utána csak plusz négyest lehet rakni, vagy húzni kell. Fontos, hogy lehet fekete lapra feketét rakni, azaz ha valaki éppen egy színkérős lappal kért kéket, arra lehet szintén egy színkérős lapot rakni.

2.4 Fontos, hogy lehet halmozni. Azaz ha rád kerül a sor, egy lap helyett többet is le lehet tenni. Csak ugyan olyan típusú lapból tehet le a játékos egyszerre többet, és egyszerre legalább hármat. Fekete, azaz „Joker” lapokat nem lehet halmozni. A halmozás itt viszont kap egy kiegészítést, mert megengedett, hogy saját körödben két színre és típusra is megegyező lapot rakjál le, mintha a saját lapodra csapnál, azaz letehetsz kér piros kettest, vagy két kék „Kimaradsz!” lapot. Ez nem érvényes szintén a fekete lapokra.

A lapokról egysmás

3.1 A szerver és a játékosok közötti üzenetváltásnak jól kell működnie, ezért mindenkinek ezekhez a jelölésekhez kell tartania magát. A lapok „kódja” két karakterből áll. Az első a színét jelenti, a második a típusát. Angolul a színek kezdőbetűi szerint, piros **R**(ed), zöld **G**(reen), kék **B**(lue), sárga **Y**(ellow) és fekete **J**(oker). A típusok meg az alábbiak: a színeknél léteznek **0-9**-ig a számok, a körfordítós a **S**(witch), a „Kimaradsz!” **L**(eft out) lapok. Feketével közös típus a plusz-os lap **P**(lus), és csak a feketéknél van a színkérős **C**(olor).

3.2 A lapok közül néhány: **R0**, **G1**, **B2**, **C3**, **RS**, **GL**, **BP**, **JP**, **JC**. Ilyen módon küldözgetik a programok egymásnak a lapokat.

3.3 A számos lapokból színenként (fekete az nem szín) kettő van, kivéve a nullásból, azokból csak egy. Továbbá kettő van színenként szintén a körfordítósból, a „Kimaradsz!”-ból és a plusz kettesből. Ezeken kívül 4-4 darab plusz négyes és színkérős fekete lap van. A szerver ezt észben tartja, és nem véletlenül kapnak lapot a játékosok, hanem egy összekevert pakliból. Ha elfogyna a pakli, akkor a dobópakli legfelső lapját kivéve, valamilyen sorrendben a húzópakli aljára teszi a lapokat.

Egy konkrét játék lebonyolítása

4.1 A kliens kezdetben megadja a szervernek, hogy milyen szobába szeretne csatlakozni. A szobákból több féle van. Ha rajtunk kívül csak robotokat tartalmazó szobába szeretnénk belépni, akkor kiválasztjuk a „robotok ellen" lehetőséget, ami az 1-es, majd utána a botok számát beírva a szerver bedobja a játékos programját egy szobába. Egy üzenetet küld ilyenkor, ami az alábbi: **CONNECT <típus> <robot darabszám, illetve szobaszám> <azonosító>**, ahol a típus az TEST (csak robotok esete) vagy ROOM.

4.2 1 ha robotok ellen, 2 ha valódi játékosok ellen is menne a program. A robot darabszáma 1 és 7 közötti szám, a szobaszám meg egy kétjegyű szám. A tízesek helyén a játékosok száma áll, az egyesek helyén meg az, hogy ezekből hány robot. Például ha két robottal akarsz tesztelni, akkor az egy ilyen üzenet formájában teheted meg: **CONNECT TEST 1 a1b2c3d4e5**, ha 5-en szeretnének játszani együtt, egy robot társaságában, akkor az üzenet: **CONNECT ROOM 61 a1b2c3d4e5**. Az azonosító mindenkinek egyedi, hogy tudjam a játékok „log"-jából (naplójából) kiolvasni, mi történt, esetleg mi volt a hiba.

4.3 Ezt a szerver egy **ACCEPTED** üzenettel nyugtázza. Innentől az üzenetek játékonként ismétlődnek. A kliens egy **READY** üzenettel jelezheti, hogy készen áll egy új játékra. A továbbiakban a szerver fog kezdeményezni.

4.4 Kliens minden játék elején történő **READY** üzenete után kap egy **DATA**… üzenetet, mikor minden játékos felkészült az új játékra. A **DATA** üzenet így néz ki: **DATA <játékosok száma> <játékos sorszáma> <kezdő lap> <saját lap>**(7 darab, ezért 7-szer), például: DATA 3 0 G6 Y1 Y1 RP BS GL JP JC.

4.5 Ezt a játékos egy **OK** üzenettel nyugtázza. Innentől kezdődik a játék.

4.6 A kliens vár, hogy a szerver közölje vele, hogy a kliens jön. Ekkor a kliens 5 féle üzenetet kaphatott. Az első esetben (A) nincs semmi kötelezettsége, és letehet akármit, ami illik az utolsó laphoz, vagy a kért színhez. B esetben plusz-os lap van érvényben, így most azt kell raknia. C esetben kimarad a játékos. Ez az eset a legegyszerűbb, mert itt csak egy OK üzenetet küld vissza a kliens, hogy nyugtázza azt, hogy kimaradt. A D és E esetekről később.

4.7 A szerver üzenete amikor a klienshez fordul az alábbi: **YT! <eset> <utolsó lap> <alkalmas szín> <alkalmas típus> <a játék iránya> HISTORY <esemény>x**(0…*). Esemény az 4 dolog lehet. **SHUFFLE**, azaz a dobópakliból vissza lettek keverve a kártyák a húzópakli aljára. Ez önmagában áll, nem követi semmi plusz infó, azaz a következőnek esemény olvasható be megint. Lehet **PUT <játékos sorszáma> <lerakott lapok darabszáma> x<lap> (<kért szín>)**. Így lehet nyomon követni, ki mit rakott le. Továbbá a kért szín csak egy fekete lap után jöhet, más esetben nyilván nem. Lehet még **DRAW <játékos sorszáma> <felhúzott lapok darabszáma>**. Végül **WIN <játékos sorszáma>**. A **DRAW** esetben nyilván lehet tartani így, hogy kinél hány lap van, és mennyi lap van a húzó és dobó pakliban. A **WIN** esetében a bent lévő játékosokat tudjuk számon tartani.

4.8 Az eset lehet „**NONE**" (A eset), „**PLUS**" (B eset), illetve „**OFF**" (C eset). Az utolsó lap valamilyen lapot ad át, azaz két karaktert. Az alkalmas szín és típus az a tehető lap színét és típusát adja meg, egy-egy karakterben. Ez arra kell, hogy ha valaki színt kért, akkor a kért szín az alkalmas szín helyébe lép, a kért típus helyébe meg nem az utolsó lap 'C' van 'P' típusa, hanem 'N'(one) lép, amiről lehet tudni, hogy nem kell a típust tartani. A játék iránya 0 vagy 1. Ha növekvő sorrendben haladunk a játékosokon, akkor 1, ellenkező esetben meg 0. Ezt követi a többi játékos cselekvéseiről egy összesítő, hogy mi történt addig, amíg nem jött a játékos.

4.9 Az A és B esetben **PUT <lap>x**(max. 8) üzenetet elküldve lehet jelezni, hogy milyen lapokat akar lerakni a játékos. Erre a szerver nem válaszol, ha szabályos a lap(ok) letétele. Továbbá ezekben az esetekben húzni a **DRAW** üzenetet elküldve lehet. A szerver ezután elküldi a húzott lapo(ka)t **CARD(S) <lap>x**(1-*) formában. Ezt a játékosnak egy **OK** üzenettel nyugtáznia kell.

4.10 A D eset az ezektől különbözik, mert azt közli, hogy te nyertél, vagyis kiestél. Szerver üzenete: **WIN <helyezés>**, ahol a helyezés 0-6 között van.

4.11 Az E eset azaz, hogy vége van a játéknak. Ekkor minden játékosnak elküldi a szerver hogy: **END <van folytatás>**, ez azt jelenti, hogy ez az utolsó játék a lejátszandó 1000-ből vagy sem. Ha igen, akkor 0, különben 1. **Fel kell készíteni a programot, hogy WIN üzenet helyet END üzenettel és vége szakadhat a játéknak.**

4.12 Ez után két eset van. Ha ez volt az utolsó játék, akkor a kliens elküldi a szervernek, hogy **BYE**, mire a szerver elküldi az eredményeket: **RESULT <a játékos pontszáma>**, és ezután leállhat a kliensprogram. Ha nem ez volt az utolsó játék akkor a kliensnek fel kell készítenie magát egy új játékra, és „visszaugrik” arra a pontra, ahol ki fogja küldeni a **READY** üzenetet.

Játék vége, pontozás

5.1 Akkor van a játéknak vége, ha csak egy játékos marad a játékban. Egy csoport 200 vagy 1000 meccset játszik le. 200 ha nem éles a játék, 1000 ha éles. A játékosokoknak (a robotokat nem számítva) a célja az, hogy minél kevesebb pontot gyűjtsenek a játékok során. A pontokat az egyes játékokban elért helyezésük alapján kapják a játékosok, méghozzá a helyezés-1 pontot. Azaz az első 0, második 1, harmadik 2 pontot kap, és így tovább. Az utolsó kivétel, ő a helyezését kapja meg pontszámként. Így három játékos esetén 0,1 és 3 pontot kapnak a játékosok.

Javítások

A 4.1 és 4.2-ben a típus 1 és 2-es értéke most már helyesen TEST és ROOM. A 4.4-esben a játékosok száma és a játékos sorszáma felcserélve volt eddig, most már javítva szerepel. A 7.3-as pont már nem volt aktuális, ezért töröltem. Kisebb elírásokat javítottam.

Fontos apróságok

A program igényel socket kezelést, ami három dolgot von magával.

6.1.1 **Egy**, hogy ha nem a Code Blocks-ban írja meg a programot valaki, akkor a projektnél a linker settings-ben egy parancsot bele kell tűzni az Other Linker options-ban az alábbi szöveget:” -lWs2_32 ”. Code Blocks-ban a megadott sablon program projektjében ez már be van írva.

6.1.2 **Kettő**, a kód C++ -ban van megírva, így más nyelvekkel nem nagyon tudok mit kezdeni, de ha valamiért nem felelne meg a C vagy a C++, akkor szóljatok nekem. Máté átírta a socket-es részét Python-ba, így akár abban is meg lehet írni a programot sablonból.

6.1.3 **Három**, a socketeket minden OP rendszer máshogy kezeli, legalábbis nem biztos, hogy minden úgy megy ahogy annak kéne. Ezért amit írtam Windows-on az nem biztos (szinte kizárt), hogy futna MAC-en, de Linuxon sem biztos. Ezért ezt csak Windows-on lehet csak megírni, de ha valakinek ezzel lenne problémája, tudok adni Windows felhő gépet neki. Az OP rendszerre átírását a sablonnak nem garantálom, mert nem tudom tesztelni.

Továbbiak

7.1 Nem szabad hibáznia a programoknak, ez alatt az alábbiakat értem: Nem állhat le a program játék közben. Nem szabad rossz lapokat leraknia a kliensnek, se rossz sorrendbe, se nem megfelelőt, se nem olyat, amije egyébként nincs, vagy nincs elég a játékosnak. Nem szabad nem alkalmas üzenetet visszaküldeni, például egy YT! OFF…-ra egy PUT…-ot. Minden üzenetet el is kell küldeni, az utolsó OK vagy BYE üzenetig. Nem küldhet olyan üzenetet ami értelmetlen, pl.: PUT JP, mert itt nem tudni, hogy milyen színt kért. Van timeout, ami 1 másodperc, azaz a késleltetést leszámítva van minimum 800 ms-e a programoknak válaszolni a szervernek.

7.2 Ezek a hibák mind ki vannak küszöbölve a csatolt kliens programban. Onnan lehet ihletet meríteni, vagy a kódot akár sablonnak is használni (legalábbis az üzenet küldéses részét kimondottan javaslom sablonnak, a többit, pl. a kártyákat tároló adatszerkezetet, meg kb. minden mást ami nincs a main-ben megemlítve, illetve a Network.h és .cpp-t azt kifejezetten átírásra szántam nektek, csak megmutatom ezzel, hogy működik a szerver) (Nem a kódok kódja hanem olyan ami éppen működik típusú)