KOMPONENS ALAPÚ KÉTSZEMÉLYES JÁTÉK – AI KOMPONENS

Komponens alapú szoftverfejlesztés, ELTE Proginf Msc 2014

Angeli Dávid

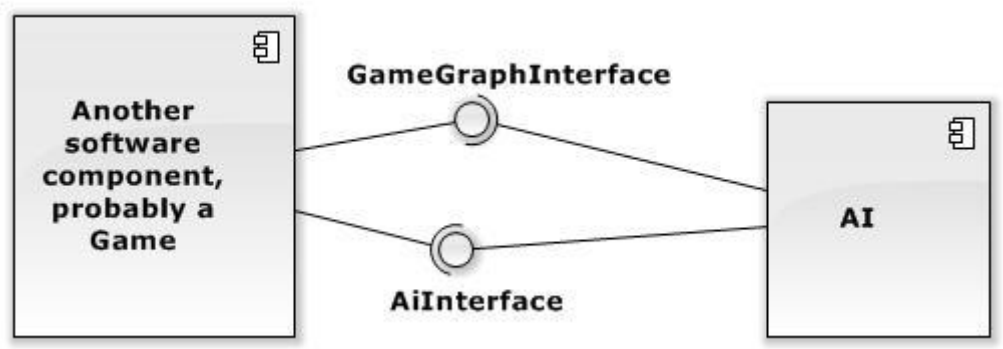
Tartalomjegyzék

Komponens specifikáció

A Komponens Alapú Kétszemélyes Játék projektben a mesterséges intelligencia komponensek feladata a játékok során a gépi ellenfél lépéseinek számítása – kiválasztása.

Strukturális modell

Az AI komponens a GameGraphInterface interfészen keresztül kap adatokat, és az AiInterface-t megvalósítva, annak getNextStep függvényével szolgáltatja a játék következő lépését.



GameGraphInterface:

Lehetővé teszi a játékfa bejárását, lekérdezhető az aktuális játékállás, egy adott állásból következő lépések halmaza, illetve egy adott álláshoz tartozó pontszám valamely játékos szemszögéből. Függvényei:

- getCurrentStep(): Object
- getPossibleSteps(step:Object): Iterable
- getScoreForPlayer(step:Object, forPlayerOne)

AiInterface:

A gépi ellenfél következő lépésének átadására szolgál. Függvényei:

- getNextStep(GameGraphInterface gameGraph: GameGraphInterface, youArePlayerOne: boolean): Object

Funkcionális modell

A komponens a külvilág felé egy eljárást nyújt, ez a getNextStep(). A getNextStep a paraméterként megkapott GameGraphInterface-en megvizsgálja a következő lehetséges lépéseket, és a komponens által megvalósított játékstratégiát követve kiválaszt egyet, amit a függvény visszatérési értékében át is ad. Két algoritmust, a MiniMax-ot és az Alfa-Bétát valósítottam meg, technikailag a kettő külön komponensbe került, de a dokumentációban tárgyalható egyszerre.

A MiniMax algoritmus[1]

- A játékfának az adott állás csúcsából leágazó részfáját felépítjük néhány szintig.
- A részfa leveleit kiértékeljük aszerint, hogy azok számunkra kedvező, vagy kedvezőtlen állások.
- Az értékeket felfuttatjuk a fában. (Saját szintjeink csúcsaihoz azok gyermekeinek maximumát, ellenfél csúcsaihoz azok gyermekeinek minimumát rendeljük.)
- Soron következő lépésünk ahhoz az álláshoz vezet, ahonnan a gyökérhez felkerült a legnagyobb érték.

Az Alfa-béta algoritmus[2]

- Visszalépéses algoritmus segítségével járjuk be a részfát (mélységi bejárás). Az aktuális úton fekvő csúcsokat ideiglenes értékekkel látjuk el:
  - a mi szintünkön α értékkel (ennél rosszabb értékű állásba innen már nem juthatunk),
  - az ellenfelén β értékkel (ennél jobb értékű állásba onnan már nem juthatunk) látjuk el.
- Lefelé haladva α =-∞, és β =+∞.
- Ezek visszalépéskor a felhozott értékre változnak, ha az nagyobb, mint az α, illetve kisebb, mint a β.
- Vágás: ha az úton van olyan α és β, hogy α >= β.

Viselkedési modell

A komponens működése során rekurzív hívásokkal megvalósítja az algoritmusok által leírt gráf-bejárásokat, és a legfelső szinten kiválasztja a legjobb megoldást. Ha nem talál ilyet –például mert a mélységi korlát vagy a gyenge heurisztika miatt nem jut el olyan állapotokig, amiknek értékelhető pontszáma van-, akkor az egyiket. A gráf-keresések egyik fontos paramétere a mélység, amit előre el kell döntenünk. Ez a paraméter ebben az implementációban forráskódbeli konstans, amit a teszt során határozunk meg.

[1] forrás: people.inf.elte.hu/gt
[2] forrás: people.inf.elte.hu/gt

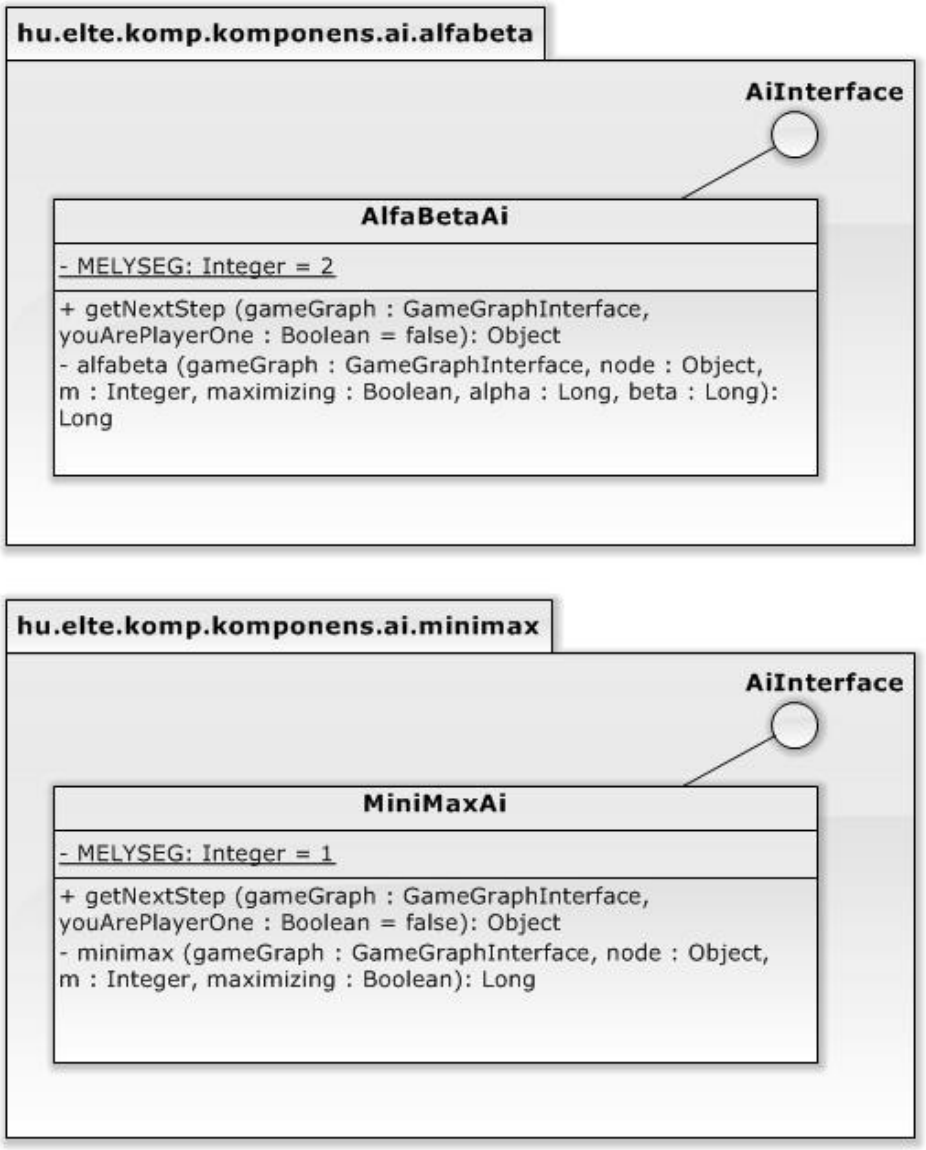
Komponens realizálás

A komponens realizálása során a meglévő funkcionális és strukturális specifikáció szerint lehetett eljárni.

Strukturális modell realizálása

A belső struktúra kialakítása során a gráfbejárást-keresést elvégző rekurzív függvényt kellett előállítani, célszerű okokból az AiInterface által megkövetelt getNextStep() eljárástól külön. Ahogy a viselkedési modell tárgyalásakor már említésre került, az osztály statikus adattagja lett a gráfbejárásra vonatkozó mélységi korlát.

A GameGraphInterface elégségesnek bizonyult a gráfkeresés megvalósításához, így saját gráf-implementáció létrehozására nem volt szükség. A (két) komponens osztálydiagramja:



getNextStep:

Az AiInterface-ben definiált eljárás. Az AI komponensekben a gráf első szintjének vizsgálata –ami az algoritmus végrehajtása során az utolsó lépés - ebben a függvényben történik. Visszaadja a maximális értékű lépést.

minimax / alfabeta:

Private függvények. A gráfbejárást-keresést, illetve az egyes csúcsok tényleges értékeinek számítását végzi a megfelelő algoritmus szerint, rekurzív hívásokkal. Visszatérési értéke a paraméterként kapott állapothoz tartozó pontérték, ami alapján a getNextStep már ki tudja majd választani a következő lépést.

Funkcionális és viselkedési modell realizálása

A komponens tevékenysége az előzőek alapján jó definiálható: amikor a szoftver másik komponensei meghívják a kívülről egyedül látható getNextStep függvényt, akkor a komponens a kapott gráfon elvégzi a megfelelő algoritmust a minimax vagy alfabeta függvények hívásával. Az eljárás során a konstans mélységi korlátot figyelembe veszi.

A getNextStep függvény eredményével visszaadja következő lépést.

Választott implementációs technológia

A megvalósításhoz – a projekt egészétől korántsem függetlenül- Java környezetet választottam, a Maven EJB projekt-templatet felhasználva. Ez jelentősen megkönnyítette az egyes komponensek összefogását és szinkronizálását.

A fejlesztés - tesztelés során használt technológiák:

JDK: 1.7

Glassfish Server 4.0

MySQL 5.6

Teszt

Unit teszt tesztesetei

A tesztelés során a JUnit keretrendszer osztálytesztjét használtam. A teszt megvalósításához a teszt-osztályban létre kellett hozni egy GameGraphInterface-t megvalósító gráf alosztályt. Ezután a teszt inicializálás során a program véletlenszerű gráfot generál, 3 mélységben, minden csúcshoz maximum 0-6 gyerekkel, és véletlenszerű (0-100) csúcsokhoz tartozó pontszámmal. A pontszámok persze csak a levelek esetében játszanak szerepet, egyébként azonosítják a csúcsokat. Az alábbiakban egy konkrét tesztesetet láthatunk a MiniMax algoritmussal:

Running hu.elte.komp.komponens.ai.minimax.MiniMaxAiTest

getNextStep
root: 20 40 74
root: 20 40 61
root: 20 43 85
root: 20 43 4
root: 20 96 78
root: 20 96 78
root: 20 77 19
root: 20 77 19
root: 20 90 79
root: 20 92 68
root: 20 92 20
root: 20 92 25
Next step found: 90
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.086 sec

Results :
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

A program jól láthatóan megtalálta a helyes lépést.

Felhasználói teszt

A funkcionális-felhasználói-integrációs teszt a projekt legkorábban elkészült játék-komponensével, a kamisado játékkal zajlott, a tényleges keretrendszerben. A teszt során az AI komponens minden esetben lépést produkált, viszont az algoritmusba beépített mélységi korlát (MiniMaxnál 2, AlfaBétánál 3) jelentős hátrányt jelentett számára. Ezzel együtt kiderült, hogy a rendelkezésre álló heurisztikával (nyertes helyzet=100 pont) felismeri és meglépi a nyertes lépéseket – ha azok kellő közelségbe kerülnek.