Tervezési dokumentáció a SnapFish66 alkalmazáshoz

# Bevezető, célok

Célom egy olyan alkalmazás fejlesztése, amely képes különböző kettes snapszer[[1]](#footnote-1) állásokat elemezni, valamint optimális lépést javasolni az adott állásban. A programnak (egyelőre) nem célja a hármas, négyes snapszer állásainak elemzése.

# A felhasználói felület (GUI)

A felhasználói felületet úgy kell megoldani, hogy minden lehetséges szabályos snapszer állás beállítható, a szabályosság pedig ellenőrizhető legyen.

A 20 lap a következő helyeken lehet:

* Az A játékos kezében
* A B játékos kezében
* Az A játékos által gyűjtött lapok között
* A B játékos által gyűjtött lapok között
* A talonban, nem látható módon
* A talon legalsó lapjaként

A játékos a következőket lépheti:

* Letesz egy kártyát a kezéből
* Húszat, negyvenet mond (automatikusan, ha megfelelő lapot játszik ki)
* Kicseréli az adu alsót a talon alján található lapra (automatikus)
* Takar

A játék véget érhet az összes lap kijátszása előtt, ezt, és az ezért járó pontszámot a program automatikusan rögzíti.

Az állás szabályossága ellenőrizhető egy gomb segítségével.

Az elemzés indításakor egy háttérszál indul, ami leállításáig folyamatosan frissítve kiírja a különböző lépésekért járó pontok várhatóértékét. Az elemzés egy másik gomb megnyomására áll le.

# Az elemzés mechanizmusa

Az elemzés során a fő szálból egy BackGroundWorker szál indul, ami a leállításáig fut.

A szál mindig a kezdőállapotból indul, és véletlenszerűen\* választ a cselekvések között, ameddig véget nem ér a játék. A lehetséges lépések: (P1,P2,P3,P4,P5,C), ahol Px az adott kártya kijátszását, C a takarást jelenti.

Takarás esetén a következő játékos jön ismét.

A játék végén kapott pont „felszivárog” a levéltől a gyökérig. Több gyerek esetén az azonos cselekvéshez tartozó értékek átlagolódnak, és ezek maximuma (ellenfél esetén minimuma) lesz a csomópont értéke.

A játék végén újabb játék indul egészen addig, ameddig a megfelelő gombbal le nem állítják a szimulációt. A játék végén frissülnek a gyökérhez tartozó állás információi a GUI-ban.

\*Később hatékonyabb kereséssel helyettesíthető, ami a játékos számára jobb lépéseket preferálja.

# Implementáció

## Card

A kártyát képviseli.

ID: azonosító, pl.: M10, P2, Z11

Value: 2,3,4,10, vagy 11

## Place

A kártyák lehetséges helyei

ID: azonosító

Cards: az itt lévő kártyák listája

## State

Egy állás

Parent: az ezt megelőző állás, a legelsőnél null

Move: Milyen lépés volt ami ide vezetett (P1,P2,P3,P4,P5,C) ahol Px az adott kártya kijátszását, C a takarást jelenti

Next: ki jön? (A vagy B)

CoveredBy: ki takart? (A,B,-)

PointsA: A pontjai

PointsB: B pontjai

Places: A kártyák lehetséges helyei

GenerateValidChild() : legenerál egy lehetséges következő állást, új kártya húzását is beleértve

IsFinal(): kiszámítja, hogy véget ért-e a játék

Score: a kapott meccspontok várhatóértéke

ValidSteps(): a lehetséges lépéseket elég egyszer kiszámolni, ezzel gyorsíthatjuk a keresést

Closed: igaz, ha minden lehetséges gyerekét bejártuk (közvetett gyerekeit is)

# Publikáció, kódanalízis

GIT

Sonarqube

1. https://snapszer.hu/hu/hivatalos-versenyszabalyzat/ [↑](#footnote-ref-1)