Specificatie opdracht 2b: Speler 1

20 maart 2009

1 Inleiding

In dit document vinden jullie een specificatie van een door jullie te implementeren computerprogramma. Het programma implementeert een speler uit een spel. Verdere onderdelen uit het spel zijn een *bord*, een *controller*, een *viewer* en natuurlijk een *tegenspeler*. Voor jullie applicatie is alleen de *controller* van belang. Alle communicatie en informatie betreffende het spel zal namelijk geschieden tussen jullie programma en de controller.
Communicatie tussen jullie programma en de *controller* zal via een TCP-verbinding plaatsvinden. In dit document vinden jullie allereerst een aantal definities, gevolgd door een formeel gedefinieerde functionaliteit die wij van jullie product verwachten. Het is de bedoeling dat jullie een uitvoerbaar bestand (.exe) opleveren, samen met de (goed becommentarieerde) broncode in Delphi Pascal.

2 Definities

We gebruiken in deze specificatie enkele verzamelingen die als volgt gedefinieerd zijn:
$TDirection = \{$`'UP'`, `'DOWN'`, `'LEFT'`, `'RIGHT'`$\}$
$TAnswer =\{$ `'K'`, `'HINT` α', `'NOWAI'` `'LOL WUT?'` $\}$ waar $\alpha \in TDirection$
$TWinMsg =\{$ `'KITTEH WINS, GTFO'` `'SUPERCAT WINS, GTFO'` $\}$

3 Functionaliteit

3.1 Verbinding opzetten

Voordat er communicatie kan zijn, dient er natuurlijk een verbinding gemaakt te worden. Hiervoor moet het programma gebruik maken van TCP-sockets. De *controller* fungeert als server en dus zal jullie programma als client kunnen verbinden met de *controller*. Om deze verbinding op stand te zetten is een ip-adres en een poortnummer nodig. Deze zijn uit te lezen uit het door ons aangeleverde INI-bestand genaamd "KITTEH.ini". Hieruit zal in de *section* "Spel" onder de *identifier* "ip" het ip-adres van de controller als *string* uit te lezen zijn. Verder is onder dezelfde *section* maar onder de *identifier* "port" het poortnummer als *int* uit te lezen. Een formele specificatie van de structuur van ini-files is te vinden in sectie 4.

Met de zojuist ingelezen informatie moet nu een verbinding opgezet worden. Hierna kan de speler berichten sturen naar de controller. Een bericht is een regel ASCII-tekst, afgesloten door een standaard Windows-regeleinde (`\r\n`).
Zodra er verbinding wordt gemaakt met de controller zal deze een bericht sturen met de waarde `'HAI'`, hierop dient door de speler te worden gereageerd met `'I IZ KITTEH'`. Als alles goed gaat reageert de controller nu met `'K'`, als er iets mis is gegaan zal de controller reageren met `'NOWAI'` of `'LOL WUT?'`, in dit geval dient de verbinding verbroken te worden.

Als alle spelers, het bord en de view verbinding hebben gemaakt stuurt het bord het bericht `'HF GO'` waarna de spelers zetten kunnen versturen.

3.2 Zetten doen

De speler die jullie moeten implementeren verstuurt verzoeken tot zetten naar de *controller*. Iedere zet wordt gevolgd door een antwoord van die controller en pas als dat antwoord is binnengekomen mag het volgende verzoek tot zet verstuurd worden. Welke zet er verstuurd moet worden is afhankelijk van de staat van de speler op dat moment. Deze staat wordt gerepresenteerd als een variabele $h \in TDirection \cup \{null\}$ die aangeeft wat de laatste hint was die de speler heeft ontvangen, initieel is de waarde van h gelijk aan $null$. Bij het versturen van verzoeken van zetten zijn er twee mogelijkheden:

1. $h = null$:
   In dit geval dient een `CAN HAS MOVE` α bericht verstuurd te worden met $\alpha \in TDirection$ waar geldt dat:
   $\alpha =$ `UP` als $0 \leq U < 0,25$
   $\alpha =$ `DOWN` als $0,25 \leq U < 0,50$
   $\alpha =$ `LEFT` als $0,50 \leq U < 0,75$
   $\alpha =$ `RIGHT` als $0,75 \leq U \leq 1$
   Hier is U een willekeurig gekozen variabele uit de standaard uniforme verdeling, voor elk bericht wordt deze opnieuw bepaald.

2. $h \neq null$:
   In dit geval dient een `CAN HAS MOVE` α bericht verstuurd te worden met $\alpha \in TDirection$ waar α gelijk is aan h.

3.3 Antwoorden verwerken

Per verzonden `CAN HAS MOVE` bericht krijgt de speler één antwoord $ans \in TAnswer$ van de *controller*, aan de hand van dit antwoord dient de waarde van h aangepast te worden. Pas als dit is gedaan mag de volgende zet worden verstuurd.
Er kunnen vier mogelijke antwoorden worden gegeven op een `CAN HAS MOVE` bericht:

$$ans = \begin{cases} \texttt{K} & \text{in dit geval verandert } h \text{ niet.} \\ \texttt{NOWAI} & \text{in dit geval krijgt } h \text{ de waarde null.} \\ \texttt{HINT } \alpha & \text{waar } \alpha \in TDirection\text{: In dit geval krijgt } h \text{ de waarde van } \alpha. \\ \texttt{LOL WUT?} & \text{er is iets misgegaan, in dit geval krijgt } h \text{ de waarde } null. \end{cases}$$

3.4 Verbinding afsluiten

Wanneer het spel stopt, moet het programma de verbinding met de *controller* afsluiten. De controller zal ofwel het bericht `'KITTEH WINS, GTFO'` of `'SUPERCAT WINS, GTFO'` doorsturen om bekend te maken dat een van de twee spelers heeft gewonnen, en het spel afgesloten kan worden. Deze berichten kunnen op élk moment na het starten van het spel door de controller worden verstuurd.

4 INI-File structuur

In dit onderdeel wordt de structuur van de door ons aan geleverde ini-bestanden in Extended BNF gespecificeerd.

```
Ini-File      ::= <section>
<section>     ::= "[Main]" <newline> <sectionbody>
<sectionbody> ::= <identifier> <newline> <identifier>
<indentifier> ::= <name> "=" <value>
<name>        ::= "IP" | "Port"
<value>       ::= <ip> | <port>
<ip>          ::= 3*<digit> "." 3*<digit> "." 3*<digit> "." 3*<digit>
<port>        ::= 1*<digit> | 2*<digit> | 3*<digit> | 4*<digit> | 5*<digit>
<digit>       ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
<newline>     ::= <CR> <LF>
<CR>          ::= ? US-ASCII character 13 ?
<LF>          ::= ? US-ASCII character 10 ?
```