Specificatie opdracht 2b: Speler 2
V1.0

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

Dit onderdeel bevat een aantal definities over berichten die verstuurd gaan worden tussen de *controller* en jullie programma.

Als een richting aangeduid wordt, gebruiken wij daarvoor de verzameling
$\mathbb{D} = \{$`'UP'`, `'DOWN'`, `'LEFT'`, `'RIGHT'`$\}$
Wanneer een aantal vakjes aangeduid wordt, gebruiken wij daarvoor de verzameling
$\mathbb{S} = \{$`'ONE'`, `'TWO'`, `'THREE'`$\}$
Communicatie geschiedt in de vorm van een verzoek tot een actie welke beantwoord wordt door de *controller*. Wanneer van zo'n bericht aangeduid wordt, gebruiken wij daarvoor de verzameling
$\mathbb{B} = \{$`'K'`, `'HINT` α`'`, `'NOWAI'`$\}$
waarbij $\alpha \in \mathbb{D}$.

3 Functionaliteit

3.1 Verbinding opzetten

Voordat communicatie kan plaatsvinden, 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 te zetten zijn een ip-adres en een poortnummer nodig. Deze zijn uit te lezen uit het door ons aangeleverde INI-bestand genaamd "SUPERCAT.INI". Hieruit zal in de *section* "Spel" onder de *identifier* "ip" het ip van de controller als String uit te lezen zijn. Verder is onder dezelfde *section* maar onder de *identifier* "port" het poortnummer als Integer uit te lezen. Een formele specificatie van de structuur van ons INI-bestand is te vinden in sectie 4.
Een bericht is een regel ASCII-tekst, afgesloten door een standaard Windows-regeleinde (`\r\n`).

Met de zojuist ingelezen informatie moet nu een verbinding opgezet worden.
Bij het openen van de verbinding zal de *controller* contact maken met het bericht "`HAI`". Om jezelf te legitimeren bij de *controller* dient allereerst een bericht "`I IZ SUPERCAT`" te worden verzonden. Indien dit wordt beantwoord met een "`K`" ben je nu correct geregistreerd als speler bij de *controller*. Wordt het echter beantwoord met een "`NOWAI`", dan moet de verbinding worden verbroken.
Indien de controller het bericht niet herkent, zal deze antwoorden met "`LOL WUT?`". Ook in dit geval moet de verbinding worden verbroken.

Zodra van de *controller* het bericht "`HF GO`" is ontvangen is het spel begonnen, en kunnen zetten worden gedaan.

3.2 Doe Zet

Voordat het doen van een zet mogelijk is, moet de controller het signaal "`HF GO`" geven, om aan te geven dat het spel geïnitialiseerd is, en gereed is voor de zetten. Tot die tijd mag er geen enkel verzoek tot een zet gedaan worden.
De speler die jullie moeten implementeren verstuurt verzoeken tot zetten naar de controller. Ieder verzoek tot zet wordt gevolgd door een antwoord van die controller, pas als dat antwoord is binnengekomen mag het volgende verzoek tot zet verstuurd worden. Welk verzoek er verstuurd moet worden is afhankelijk van de staat van de speler op dat moment. De staat wordt gerepresenteerd als een variabele $h \in \mathbb{D} \cup \{null\}$ (initiëel $null$) die aangeeft wat de laatste nuttige hint was die de speler heeft ontvangen. Ook is er een variabele $ds \in \{true, false\}$ voor wanneer $h \neq null$. Er zijn nu een aantal mogelijkheden:

1. $h = null$:
   Kies een nieuwe waarde voor t uit de standaard uniforme verdeling.

   (a) $0 \leq t < 0,50$:
   In dit geval dient een "`CAN HAS TURN` θ" bericht verstuurd te worden waar geldt dat $\theta \in \mathbb{D}$ en dan wel:
   $$\theta = \begin{cases} \texttt{UP} & \text{als } 0 \leq u < 0,25 \\ \texttt{DOWN} & \text{als } 0,25 \leq u < 0,50 \\ \texttt{LEFT} & \text{als } 0,50 \leq u < 0,75 \\ \texttt{RIGHT} & \text{als } 0,75 \leq u \leq 1 \end{cases}$$
   Hier is u een variabele met een willekeurige waarde uit de standaard uniforme verdeling. Voor elk bericht dient een nieuwe waarde gekozen te worden.

   (b) $0,50 \leq t \leq 1$:
   In dit geval dient een "`CAN HAS MOVE` ϕ" bericht verstuurd te worden waar geldt dat $\phi \in \mathbb{S}$, namelijk
   $$\phi = \begin{cases} \texttt{ONE} & \text{als } 0 \leq v < 0,34 \\ \texttt{TWO} & \text{als } 0,34 \leq v < 0,67 \\ \texttt{THREE} & \text{als } 0,67 \leq v \leq 1 \end{cases}$$
   Hier is v een variabele met een willekeurige waarde uit de standaard uniforme verdeling. Voor elk bericht dient een nieuwe waarde gekozen te worden.

2. $h \neq null$:

   (a) $ds = false$:
   In dit geval dient een "`CAN HAS TURN` θ" bericht verstuurd te worden waar geldt dat $\theta = h$. De variabele ds krijgt de waarde $true$.

   (b) $ds = true$:
   In dit geval dient een "`CAN HAS MOVE ONE`" bericht verstuurd te worden.

3.3 Verwerk Antwoord

Na ieder "`CAN HAS MOVE`" of "`CAN HAS TURN`" bericht krijgt de speler een antwoord van de controller, aan de hand van dit antwoord dient eventueel de waarde van h of de waarde van ds aangepast te worden. Pas als dit is gedaan mag de volgende zet worden verstuurd.
Er kunnen drie mogelijke antwoorden worden gegeven op een "`CAN HAS MOVE`":

"`K`":
Dit geeft een bevestiging dat de zet is goedgekeurd, en dat de speler nu een nieuw verzoek voor een zet mag versturen.

"`HINT` α":
Dit geeft een bevestiging dat de zet is goedgekeurd, en dat de speler op een hintvakje terecht is gekomen. De waarde van α ($\alpha \in \mathbb{D}$) dient te worden opgeslagen in h. Ook dient ds de waarde $false$ te krijgen. De speler mag een nieuw verzoek voor een zet insturen.

"`NOWAI`":
Dit geeft aan dat de zet niet is goedgekeurd, omdat deze niet mogelijk is. Indien $h \neq null$, dient deze de waarde $null$ te krijgen. De speler mag een nieuw verzoek voor een zet insturen.

Op een "`CAN HAS TURN`" is het enige antwoord de bevestiging "`K`", welke aangeeft dat de zet heeft plaatsgevonden. De speler mag een nieuw verzoek voor een zet insturen.

Het bericht "`LOL WUT?`" wordt teruggestuurd als er een onbekend bericht naar de controller is gestuurd. In dit geval dient h de waarde $null$ te krijgen, waarna een nieuw verzoek voor een zet mag worden ingestuurd.

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 ?
```