# **Netværksspil - Arnold**

# **Afgrænsning**

**Lav en afgrænsning/præcisering med baggrund i ovenstående krav til spillet. Det skal hovedsageligt være en præcisering af kravene/reglerne og analyse af, hvilke data spillet skal indeholde. Det er tilladt at ændre regler eller tilføje nye regler – du må dog ikke ændre spillets karakter uden at aftale det med underviseren.**

Schwarzenegger-protokollen er en protokol designet til at håndtere kommunikation mellem spillere i spillet Arnold. Protokollen er en client/server protokol, der bruger UDP som transportlag. Vores spil anvender central synkronisering. Vi har valgt at bruge port 7542. Tilmelding foregår ved at sende en Login besked. Afmelding foregår ved at sende en Logoff besked. Vores fulde protokol er illustreret herunder.

**1) Handshake**

1.1) Når en spiller logger på sender hans klient en LogonRequestPacket til en server. Pakken indeholder et ønsket brugernavn.

1.2) Hvis brugernavnet er ledigt og protokol-versionen matcher udføres punkt 1.2.a. Ellers udføres punkt 1.2.b.

1.2.a) Brugeren tildeles et ID-nummer af serveren, og serveren sender en loginGrantedPacket til klienten. Pakken indeholder det nye brugernavn samt det aktuelle map-layout. Der springes til punkt 2.

1.2.b) Serveren sender en loginDeniedPacket til klienten. Forbindelsen betragtes nu som afsluttet.

**2) Normal**

Her er to muligheder: Klienten kan sende en actionPacket til serveren, eller serveren kan sende en statePacket til klienten.

**2.a) actionPacket**

Klienten sender en actionPacket, der beskriver hvilke handlinger brugeren har foretaget siden sidste actionPacket.

Klienten bør holde en pause på minimum 50 ms mellem hver pakke for at holde netværksbelastningen nede. Hvis brugeren foretager sig flere handlinger i dette tidsrum sendes disse handlinger i den samme actionPacket med handlingerne listet kronologisk, med den ældste handling først.

Klienten bør sende en actionPacket minimum hvert 5. sekund. I tilfælde af at brugeren ikke har foretaget sig noget siden sidst sendes blot en actionPacket uden actions.

Hvis serveren ikke har modtaget en actionPacket i 20 sekunder skal klienten betragtes som logget ud.

**2.b) statePacket**

Serveren sender en statePacket til klienten, der beskriver alle spilleres nuværende position, score, brugernavn og retning.

Hvis en spiller ikke er nævnt i pakken betragtes han som logget ud.

Serveren bør holde en pause på miminum 50 ms mellem hver statePacket for at holde netværksbelastningen nede.

Serveren bør sende en statePacket minimum hvert 5. sekund.

Hvis klienten ikke har modtaget en statePacket i 20 sekunder skal serveren betragtes som død.

**3) logoffPacket**

En klient bør logge af når den lukker ned. Dette foregår ved at sende en logoffPacket. Når serveren modtager en logoffPacket er brugeren ikke længere med i spillet, og klienten modtager ikke flere statePacket's. Efterfølgende actionPackets ignoreres af serveren. Klienten modtager ikke noget svar på en logoffPacket.

**Protokolnavn: Schwarzenegger**

**Protokolversion: 1.0**

**Klient til server:**

**Login:** = ”protokolnavn + space + protokolversion + return + Login + space + Brugernavn”

**Logoff** = ”protokolnavn + space + protokolversion + return + Logoff”

**Action:** = ”protokolnavn + space + protokolversion + return + action”

Hvor en action enten kan være ”Shoot” eller ”move” plus den retning man vil bevæge sig i, som illustreret herunder.

**Skyd:** = ”protokolnavn + space + protokolversion + return + shoot”

**Bevægelse:** = ”protokolnavn + space + protokolversion + return + move + retning”

Vores UDP pakke kan indeholde adskillige actions af gangen, som fx:

Schwarzenegger 1.0  
moveup  
shoot  
moveright  
moveright

Det fungerer på den måde, at vi bruger en actionbuffer til at gemme vores actions i, indtil det er tid til at sende pakken. Hvis der ikke er fyldt noget i actionbufferen når det er tid til at sende en pakke, bliver der ventet et antal millisekunder med at prøve at sende igen. Dermed undgår vi at sende for mange tomme pakker til serveren.

**Server til klient:**

**LoginGranted:** = ”protokolnavn + space + protokolversion + return + Granted + space + id + return + map”

Mappet ligger på serveren og bliver sendt som et stringarray, som klientens computer kan omforme til grafiske layout på skærmen.

**LoginDenied:** = ”protokolnavn + space + protokolversion + return + Denied”

**State:** = ”protokolnavn + space + protokolversion + return + id + space + xPos + space + yPos + space + Score + space + Username + space + Direction + return + osv………”

En statepakke indeholder informationer om alle spilleres id, nuværende positioner, antal point, brugernavn og retning(som et tal mellem 0 og 3), som fx:

Schwarzenegger 1.0  
1 5 4 10 Søren 2  
2 2 2 -15 Anders 1  
3 10 3 666 Satan 0  
4 1 11 0 Morten 3

**Design**

Under design skal en række beslutninger tages, herunder

* **Hvorledes skal tilmelding til spillet foregå?**  
  En klient starter Eclipse, kører Main-metoden i klassen ”Game” og indtaster et brugernavn. Herefter sørger programmet for, at sende en Login-besked til serveren. Hvis klienten modtager en ”Login granted” tilbage, kan spilleren starte programmet op. Indtil videre er der ingen begrænsninger for, hvor mange brugere der kan tilmelde sig et spil.
* **Hvorledes foregår autoriseret afmelding?**  
  Automatiseret afmelding sker ved et tryk på Escape-knappen. Klienten sender en ”Logoff” besked til serveren, og herefter termineres programmet og klienten bliver fjernet fra serveren. Vi har desværre ikke nået at lave et vindue, der skal spørge på, om vi vil bekræfte lukning. Derudover er det gældende både for klienten og serveren, at hvis de ikke har svaret i 20 sekunder, bliver de automatisk stoppet.
* **Hvorledes sikres det, at alle spillere er enige om den samme stilling?**

Serveren sørger altid for at have den endelige information, som den sender ud til klienterne. Derved er det altid serverens informationer der gælder.

* **Hvordan kan spillet gøres fejltolerant?**

Vi kunne på længere sigt have implementeret timestamping algorithm, så man rent faktisk ville kunne afgøre, hvilken spiller der har bevæget sig først.

**Installationsvejledning**

Importér projektet til Eclipse. For at starte serveren køres Main-metoden i klassen ”Main” i pakken Server. Herefter startes en klient ved at køre Main-metoden i klassen ”Game” i pakken ”game”.