The UNO game

- Ett projekt i kursen Nätverksprogrammering, EDA095

Josefine Axelsson, lan13jax@student.lu.se
Mikaela Giegold, mte13mgi@student.lu.se
Elisabeth Persson, lan13ep1@student.lu.s
Martin Sundeqvist, sys12msu@student.lu.se

Institutionen för Datavetenskap
Lunds Universitet
May 19, 2017

Bakgrund

Projektet syftar till att ge oss som grupp en djupare förståelse för nätverksprogrammering samt mer kunskap om hur kommunikationen mellan datorer sker vid användande av en nätverksapplikation och hur denna kan utvecklas. Vi har valt att utveckla ett interaktivt spel liknande det klassiska kortspelet UNO där två eller fler spelare tävlar mot varandra. Projektets implementering bygger på en server - klient modell där servern läser/hämtar information från spelarna och uppdaterar spelet i realtid. Denna modell gör det även möjligt för spelarna att kunna chatta med varandra.

Kravspecifikation

- Minst två spelare ska kunna spela kortspelet UNO mot varandra
- Möjlighet för en spelare att säga ”UNO” när den endast har ett kort kvar på hand
- Spelet sker över ett nätverk
- Kommunikation sker via en TCP server
- Möjlighet för spelare att kommunicera med varandra via chatt

Utökad kravspecifikation, om tid fanns för vidarutveckling av spelet.

- Möjlighet för spelare att välja användarnamn
- Spara statistik över vinster och förluster
- Möjlighet att spela med poängräkning
- Matchmaking - systemet skapar matcher mellan spelare som har samma ranking

Modell

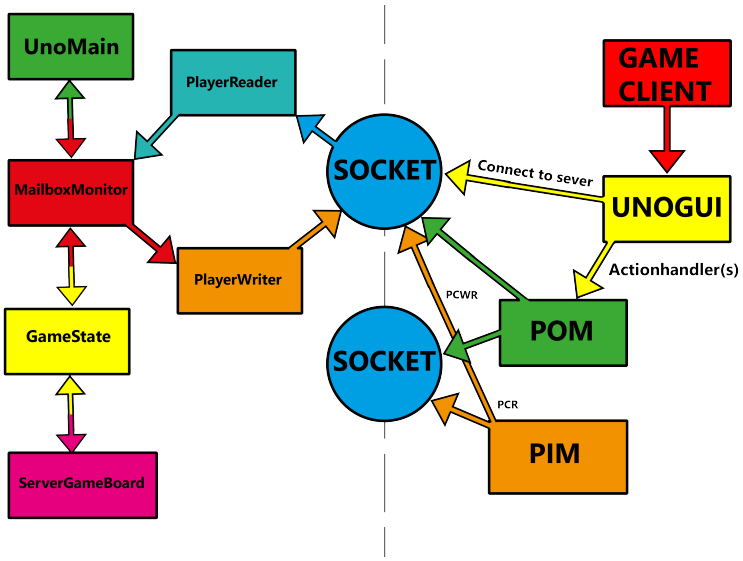
Spelet utgår huvudsakligen från tre klasser, UnoMain och GameClient samt UnoGUI. UnoMain körs på serversidan, i turordning startas servern, ägaren tillfrågas om antalet spelare för spelomgången och spelet startas efter att en accept-loop har skapat sockets till antalet spelare som efterfrågades. GameClient körs på spelarsidan, den startar upp UnoGUI som är en Application inom JavaFX-ramverket. I UnoGUI ombeds spelaren att berätta sitt namn, IP-adressen för en spelserver och porten på denna server. När tillräckligt många spelare har anslutit skickar UnoMain ett meddelande till alla spelare som berättar för dem att det är dags att visa GUI:t.

Spellogiken körs till viss del lokalt med enkla kontroller då spelaren exempelvis trycker på en viss knapp. Huvudsakligen körs spelet dock genom att

spelaren skickar ”förslag” på kommandon till servern, där kommandona tolkas i en klass som heter GameState och sedan passas vidare till ”spelmotorn” som kallas ServerGameBoard. Här körs metoder som bedömer bland annat om spelaren får lägga kort, om den har vunnit samt hur många kort den ska ta upp vid ett visst tillfälle. ServerGameBoard kommunicerar sedan resultatet av en viss spelarhandling genom GameState tillbaka till spelarna.

Nätverkskommunikationen använder tre stycken centrala monitorer, MailboxMonitor på serversidan samt PlayerInputMonitor och PlayerOutputMonitor på spelarsidan. MailboxMonitor skapar två trådar för varje spelare då dessa ansluter, PlayerReader och PlayerWriter, som lämnar in- och ut-meddelanden i en in- respektive ut-mailbox för varje spelare. MailboxMonitor initierar även den tidigare nämnda GameState-tråden som ansvarar för att skicka info från och till ServerGameBoard. På spelarsidan äger PlayerInputMonitor och PlayerOutputMonitor varsin mailbox med tillhörande skriv- och läs-trådar PlayerClientWriter och PlayerClientReader. I PlayerInputMonitor finns en referens till GUI:t, då ett nytt meddelande kommer in till in-maillådan så körs en ”update”-funktion i GUI:t som säger till den att läsa in från mailboxen och göra lämpliga ändringar i GUI:t.

I projektet används LinkedList flitigt, exempelvis för att modellera mailboxes och kortlekar. BufferedReader och PrintWriter används i läs- och skrivtrådarna och för att modellera GUI:t används JavaFX-ramverket.



En grafisk representation av modellen. Observera att varje socket har varsitt av elementen de kommunicerar med.

Användarhandledning

Då spelet startas upp får den som ansvarar för spelservern möjlighet att ange hur många spelare han eller hon önskar spela med. Därefter får samtliga spelare som ansluter sig möjlighet att ange ett användarnamn, samt till vilket maskin och port de vill ansluta. Då antalet anslutna spelare är lika många som serverägaren angivit startas spelet och varje spelare får en spelplan. Spelplanen består av en korthand med 7 slumpade kort, en chatt/meddelanderuta, en nedvänd korthög med de resterande korten samt det senast uppvända kortet. Turen slumpas till någon av deltagarna och spelet kan starta!
Beroende på de kort spelaren har i sin hand kan något av följande läggas:

- Ett kort av samma färg.
- Ett eller flera kort med samma siffra. Om spelaren har fler kort med samma siffra men olika färg kan samtliga av dessa läggas. För att göra detta måste spelaren markera korten i den ordning de ska läggas. Turen går vidare.
- Ett eller flera +2-kort om det första valda kortet har samma färg som det senast lagda kortet. Den nästkommande spelaren får då ta upp två kort (för varje kort som lagts) och turen går vidare.
- Ett eller flera stopp-kort om det första valda kortet har samma färg som det senast lagda kortet. Nästkommande spelare blockas då och turen går vidare till spelaren efter den som blivit blockad.
- Ett ”vänd spelriktning”- kort om det har samma färg som det senast lagda kortet. Spelarordningen ändras då och turen går vidare till spelaren innan.
- Ett svart kort. Spelaren får då även välja ny färg och turen går vidare.
- Ett eller flera svarta +4-kort. Spelaren får då välja ny färg. Nästkommande spelare får ta upp fyra kort (för varje kort som lagts) och turen går vidare till spelaren efter den som fått plocka upp kort.

Om spelaren inte kan lägga något av ovanstående får denne ta upp ett kort och turen går vidare. Om spelaren har ett kort kvar måste denne klicka på UNO innan det sista kortet läggs. Om det sista kortet läggs utan att UNO-knappen markerats får spelaren ta upp tre nya kort. Om UNO-knappen däremot markerats och samma spelare lägger sitt sista kort är spelet avgjort och spelaren har vunnit.

Utvärdering

Då projektet avslutas kan spelet spelas av två eller fler användare över ett nätverk. Spelarna ges möjlighet att bestämma användarnamn, chatta med

varandra samt att säga ”UNO”, precis som i det ”riktiga” spelet. Detta innebär att spelet uppfyller samtliga krav i den grundkravsspecifikation som sattes vid projektets start samt en av de punkter som definierades i den uttökade kravspecifikationen.

De övriga punkterna i den utökade kravspecifikationen skulle eventuellt kunnat implementerats om tidsramen för projektet varit längre. Möjligheten att spela med poängräkning hade förmodligen varit den enklaste att implementera utav dessa, då varje spelares poäng räknas samman baserat på korten denne har på handen då någon av motspelarna vunnit.

Möjligheten att spara statistik över vinster och förluster samt organisera något slags matchmaking-system hade förmodligen krävt lite mer arbete. För tillfället startas servern om inför varje ny spelomgång men för att kunna spara och använda spelardata skulle servern behöva hållas igång. Alternativt skulle data kunna lagras i molnet, på en central server eller sparas lokalt.

Vi tycker att projektet har varit roligt, lärorikt och givande. Om vi skulle gjort om uppgiften hade vi nog försökt lägga mer tid i början på projektet för att bestämma vilka kommandon som behövdes och hur dessa skulle implementerats. Då hade vi nog lättare kunnat felsöka vår kod och lättare kunnat avgöra hur lång tid olika moment skulle ta. Vi hade även tyckt det vore intressant att höra hur det gått för de andra grupperna och se hur de gått tillväga. En idé hade varit att ha ett seminarium/gruppdiskussion där man kunde utbyta strategier, tillvägagångsätt och presentera projekten.

Programlistor

Källkoden är tillgänglig via GitHub. `https://github.com/micsandbricks/UNO_network_game`