12/7/2006

IT2105 POKER PROJECT

Casino Royale | Håvard Sørbø & Jonas Follesø

Prosjektstruktur og filinndeling

Siden vi var to prosjektdeltagere valgte vi å benytte oss av Subversion til kildekodekontroll. For at vi enklest skulle kunne jobbe på forskjellige deler av spillet valgte vi og å dele opp koden over flere filer:

- **Lf.scm** – Siste versjon av løsningsforslaget. Denne filen definerer en rekke objekter og funksjoner, som kort, kortstokk, stokking, sortering, finne styrken til en hånd osv.

- **Poker.scm** – Dette er hovedfilen i prosjektet. Denne filen importerer de andre filene. Vi har og flyttet funksjonene ***gen-card-dealer*** og ***gen-card-player*** fra lf.scm og inn i poker.scm. Disse funksjonene definerer objektene for dealer og kortspiller. Dealeren er motoren i spillet.

- **Betting.scm** – Denne filen inneholder betting funksjonene våre, en ***simple-betting*** og en ***rollout-betting***.

- **Rollout.scm** – Denne filen inneholder roll-out simulatoren. Denne er implementert som en funksjon ***run-rollout-simulation***. Denne funksjonen har følgende parametere: antall runder du ønsker å simulere, dine private kort, felles kort, og antallet motstandere.

- **Intro.scm** – En enkel fil som skriver ut litt ASCII-art til skjermen og tittelen på programmet: Casino Royale.

- **Tests.scm** – Inneholder enhetstester for å verifisere at diverse funksjoner fungerer som de skal.

Pokersimulering

En runde pokersimulering skjer ved å lage en ny instans av et dealer objekt og deretter kalle ***reset***, ***texas*** og ***show-score*** funksjonene på objektet. Internt i dealer objektet har vi en ***texas-holdem*** funksjon som kjører selve simuleringen en runde og har ansvaret for hele flyten i en simulering.

Selve simuleringen av en pokerrunde begynner med at potten settes til 0, listen over aktive spillere settes til alle spillerne rundt bordet, og hjelpevariabel som husker hva resultatet av bettingen er settes til null. Deretter får hver spiller utdelt to kort, og første beting runde starter. Selve bettingen er implementert i en egen funksjon; ***betting***.

Betting funksjonen tar tre parametere, en liste med aktive spillere, maks antall runder og hvor stor potten er blitt. Maks antall runder sendes med for å hindre at spillerne høyner i det uendelige. Bud funksjonen er implementert ved hjelp av en itterator som har følgende stoppkriterier:

1. Vi har oppnådd maksimal antall runder betting og tvinges til å avslutte.
2. Alle spillerne med unntak av en kaster seg, og derfor vinner han potten. For å avgjøre dette kaller vi predikatet ***all-fold?***.
3. Alle spillerne unntatt de som har kaster seg syner. Dette avgjøres ved å kalle predikatet ***showdown?***.

Predikatene ***all-fold?*** og ***showdown?*** Benytter seg av funksjonen ***count-player-state*** til å telle opp antall spillere som høyner, syner eller kaster seg.

Dersom ingen av stoppkriteriene har slått til kjører følgende sekvens:

1. Alle spillerne som er i listen over aktive spillere blir bedt om å bette, og vi tar vare på resultatet som er en cons-celle bestående av hva spilleren byr (raise, call, fold), og eventuelt et beløp.
2. Deretter sjekker vi hva spilleren har byd. Basert på dette gjør vi følgende:
    a. Spilleren høyner: Vi legger til beløpet til variabelen som holder oversikt over nåværende bud, vi øker potten i denne budrunden, og potten for hele pokerrunden. Deretter tar vi i mot penger fra spilleren, og øker variabelen som holder oversikt over hvor mye spilleren har bidratt med i potten denne runden.
    b. Spilleren syner: Vi finner ut prisen for å syne ved å trekke antallet penger spilleren har bidratt med fra nåværende bud. Deretter tar vi penger fra spilleren og legger denne summen til potten for denne budrunden, og den totale potten. Deretter øker vi variabelen som holder oversikt over hvor mye spilleren har bidratt med i potten denne runden.
    c. En spiller kaster seg: Vi oppdaterer spillerens status slik at han ikke blir med videre i spillet.

Betting funksjonen returnerer en liste bestående av spillerne med oppdatert rundestatus (raise, call eller fold), samt hvor mange penger som ble lagt til i potten. Det neste som skjer er at vi oppdaterer variabelen som holder total sum i potten, samt at vi oppdaterer listen over aktive spillere ved å kalle en hjelpefunksjon; ***find-winners***. Denne funksjonen fjerner spillere som har kastet seg i løpet av bettingrunden.

Når vi er ferdig med første budrunde sjekker vi om alle utenom en har kastet seg ved hjelp av predikatet ***all-fold?***. Dersom dette er tilfelle vinner spilleren som ikke kastet seg og runden er over. Hvis ikke går spillet videre og tre floppen deles ut og vi starter en ny budrunde. Dersom ikke alle utenom en kaster seg gjør vi det samme to ganger, først for turn og så for river kortene.

Siste budrunde skjer etter femte felles kort, riveren. Dersom alle utenom en kaster seg vinner han som vanlig. Dersom dette ikke er tilfelle går vi videre til en showdown. Dette avgjør vi ved å bruke predikatet ***showdown?***. Dette er første gang vi avgjør hvem som vinner basert på kortene de sitter med. Måten vi

avgjør hvem som vinner er å kalle funksjonen ***showdown-winners***. Denne funksjonen sorterer spillerne basert på styrken i hånda. Funksjonen returnerer en liste med vinnerne. Grunnen til at vi ikke kun tar den første spilleren etter en sortering basert på hendene, er for å håndtere uavgjort. Deretter blir poten fordelt på vinneren.

Når runden er over skriver vi ut resultatlisten på skjermen som er alle spillerne sortert etter hvor mange penger de har. For å kunne gjøre simuleringen hvor mange ganger vi vil har vi lagd funksjonen ***run-simulation*** som tar antall spillere og antall runder som parametere.

Enkel betting

Vi har implementert to bettingfunksjoner. Hvilken av de to som skal brukes kan defineres som en global variabel i poker.scm fila. Den enkleste funksjonen, som er besvarelse på første fase av prosjektet, kaller vi ***simple-betting***. Denne funksjonen regner først ut power-indeks for kortene som er felles for spillerne rundt bordet. Deretter regner vi ut power-indeks til hånden til spilleren, som er de to private kortene kombinert med felleskortene. Dersom disse to indeksene er like betyr det at spilleren kun har kort i bordet, og dette er jo kort som alle de andre spillerne også har. I så tilfelle behandler vi dette som en dårlig hånd.

Basert på hvor god hånden er tar spilleren en avgjørelse om vi skal høyne, syne eller kaste seg. Reglene som avgjør hva spilleren gjør er listet opp i rekkefølgen de kjøres.

- Dersom alle har kastet seg så syner spilleren og vinner potten.
- Dersom du åpner med to kort og en power-indeks over 1, høyne med 25$
- Dersom du åpner med to kort, og et av kortene er høyere enn 9, syne.
- Dersom du åpner med to kort er det 25 % sjanse for at du høyner med 25$
- Dersom du åpner med to kort er det 33 % sjanse for at syner.
- Dersom du åpner med to kort og du ikke høyner eller syner så kaster du deg.
- Dersom du har en full hånd med kun 1 i power-indeks kaster du deg.
- Dersom du har en full hånd, der du kun har power-indeks i bordet er det 50 % sjanse for at du kaster deg.
- Dersom du har en full hånd og over 3 i power-indeks høyner du med 25$
- Dersom du har en full hånd og 3 i power-indeks syner du.
- Dersom du har en full hånd, og 1 eller 2 i power-indeks er det 75 % sannsynlighet for at du syner.
- Dersom du har en full hånd, og 1 eller 2 i power-indeks er det 66 % sannsynlighet for at du høyner
- Hvis ikke kaster du deg.

Roll-out betting

Fase to i prosjektet består i å implementere en roll-out simulator hvor du simulerer et gitt antall runder poker og teller opp hvor mange ganger du vinner. Basert på prosenten runder du vinner kan spilleren ta en avgjørelse om å kaste seg, syne eller høyne. Selve simuleringen skjer ved å kalle funksjonen ***run-rollout-simulation*** med følgende parametere: antall simuleringer du ønsker å kjøre, dine private kort, felles kort, og antall motstandere. Det første som skjer er at vi oppretter en ny kortstokk der vi fjerner kortene vi kjenner til. Dette for at det ikke skal bli to like kort med i smuleringen. Deretter kaller vi funksjonen ***win-simulation?*** som returnerer sant/usant om du vinner eller ikke så mange ganger som definert i antall simuleringer. Deretter teller vi opp antall ganger vi vinner og returnerer prosenten som forteller hvor ofte vi vinner.

Basert på denne prosenten, og en ekstravariabel (1 til 10) som definerer ”gutsen” til spilleren, bestemmer spilleren seg for å høyne, syne eller kaste seg. Avgjørelsene tas basert på følgende regler, i angitt rekkefølge:

- Dersom vi er siste man som ikke har kastet seg, så syner vi og dermed vinner potten.
- Dersom vi vinner over 30 % av rundene, er det en viss sannsynlighet, avhengig av gutsvariabel, for at vi høyner med 25$.
- Dersom vi vinner over 10 % av rundene og det ikke er delt ut noen felles kort, er det en viss sannsynlighet, avhengig av gutsvariabel.
- Dersom vi vinner over 40 % av rundene høyner vi med 25$.
- Dersom vi vinner over 30 % av rundene syner vi.
- Ellers kaster vi oss (vi vinner mindre en 30 % og har ikke ”guts”).

Roll-out betting med tabell for åpning

Som en tredje betting funksjon utvidet vi roll-out bettingen med å bruke David Sklansky og Mason Malmuths tabell over åpningskort.[1]

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	1	1	2	2	3	5	5	5	5	5	5	5	5
K	2	1	2	3	4	6	7	7	7	7	7	7	7
Q	3	4	1	3	4	5	7						
J	4	5	5	1	3	4	6	8					
T	6	6	6	5	2	4	5						
9	8	8	8	7	7	3	4	5	8				
8				8	8	7	4	5	6	8			
7							8	5	5	7	8		
6								8	6	5	7		
5									8	6	6	8	
4										8	7	7	8
3												7	8
2													7

For eksempel så gir åpningskortene (8 kløver) og (9 kløver) verdien 4, mens (8 ruter) og (9 spar) gir verdien 7. Dersom kortene er i samme farge leser du minste verdi først i tabellen. Celler som ikke har verdi er de dårligste åpningshendene, mens par i ess, sammen med høye par i samme farge er de beste åpningskortene med verdi 1. Funksjonen fungerer på samme måte som roll-out bettingen, med unntak for åpningshender. Her bruker vi tabellen og følger følgende regler:

- 1 – 3: Høyne
- 4 – 6: Syne
- 7 – 9: Kaste deg

[1] http://en.wikipedia.org/wiki/Texas_hold_%27em_hands

Debugging av simuleringen

En av utfordringene vi hadde var å vite om ting fungerte som vi forventet. For å kunne skrive ut debug informasjon til skjermen lagde vi en ***dprint*** funksjon som tar uendelig mange parametere, der den første er debug nivå mens resten er tekst som skal skrives ut på skjermen. I tillegg har vi en global variabel som definerer hvilket debug nivå vi ønsker å kjøre simuleringen på. Funksjonen ***dprint*** tar hensyn til denne variabelen og skriver kun ut tekst på skjermen parameteren er større eller lik nivået vi kjører under. På denne måten kan vi enkelt kalle ***dprint*** rundt om i koden, og justere hvor mye av dette vi ønsker å se under en normal kjøring av simuleringen. Når vi ikke ønsker å debugge setter vi variabelen til 0, og ingen av debug beskjedene kommer på skjermen.