Prosjektrapport

Effektiv kode med C og C++
DAVE3605
Høgskolen i Oslo og Akershus

KnowWhenToFoldEm

Mads Myrbakken Karlstad
s193949

Dette er en rapport om utførelsen av prosjektoppgaven i faget Effektiv kode med C og C++ ved Høgskolen i Oslo og Akershus. Formålet med rapporten er å kartlegge metode og vurdering av eget arbeid, samt avdekke eventuelle mangler og bugs i programmet som ble laget. Man sto ganske fritt til å velge hva man ville gjøre i prosjektet da oppgavebskrivelsen var

"*Lag et fungerende og brukbart C++ program som du liker tanken på å vise frem."*

Valget av oppgave falt på å programmere et pokerspill med tre eller flere spillere, det vil si én spiller og én eller flere AI-spillere. Originalt skulle dette prosjektet utføres som et gruppeprosjekt bestående av to studenter, Mads M Karlstad og Erlend Westbye, men Erlend måtte hoppe av dette kurset da tiden ikke strakk til. Prosjektet leveres derfor ikke som et gruppeprosjekt. På grunn av dette har det ferdigstilte programmet viket litt bort fra det som ble beskrevet i Prosjektbeskrivelsen som ble levert 10.april.2015.

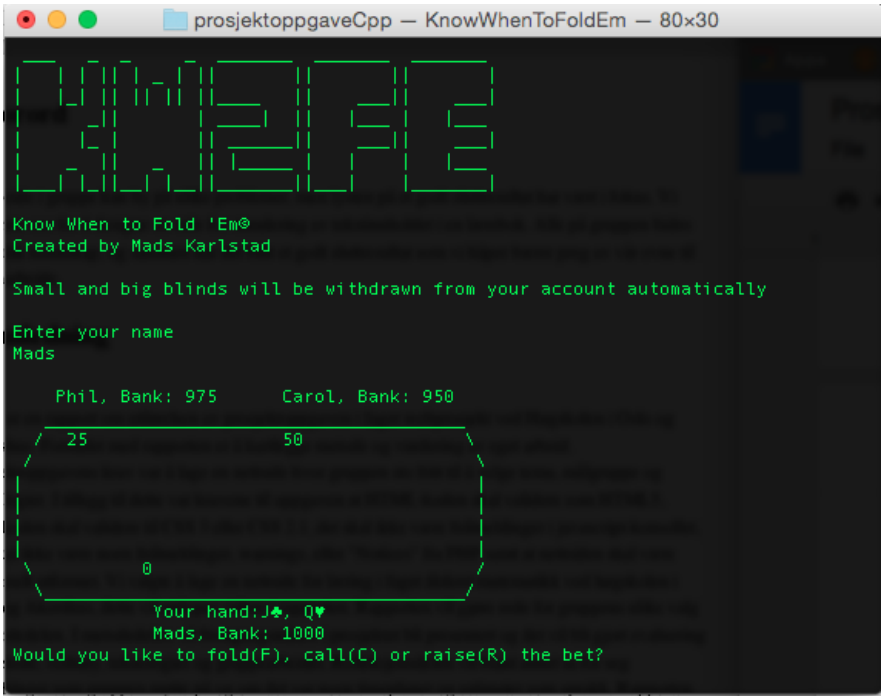
Rapporten vil bestå av en del som tar for seg steg for steg hvordan pokerspillet gjennomføres av en bruker, en del som omhandler metoder som er brukt i funksjoner og klasser, en refleksjonsdel hvor det reflekteres i utfordringer med prosjektet og til slutt vil rapporten avsluttes med en konklusjonsdel.

Vedlagt ligger et klassehierarki.

1. Gjennomførelse av spillet

Pokerspillet startes først ved at brukeren navigerer seg til mappen "prosjektoppgaveCpp" i terminalklienten sin. Programmet kompileres med kommandoen $ make og kjøres deretter med $ ./KnowWhenToFoldEm.

Det vil så komme opp en logo laget med Ascii-symboler og litt info etterfulgt av en beskjed om å skrive inn navnet sitt. Dette blir navnet på spilleren din. Spillet starter så opp og du vil se et enkelt bord, med informasjon om din spiller nederst på bordet.

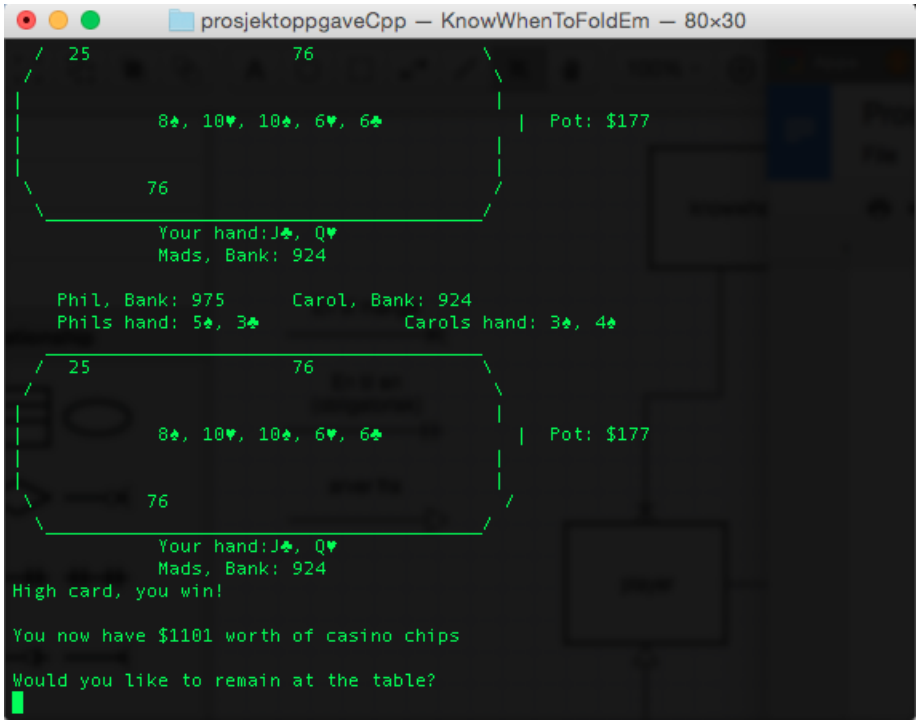


Skjermdump

Her ser man summen av sjetonger hver spiller har stående. Tallene på bordet representerer hvor mye en spiller har bettet. I dette tilfellet er Phil *small blind* og Carol er *big blind.* Mads er *dealer* og må derfor calle big blind for å få lov til å fortsette spillet.

Man får så valget mellom å kaste seg, se budet på bordet eller høyne. Etter dette får brukeren beskjed om hva motspillerene gjør, om de kaster seg, checker eller høyner etterfulgt av at det blir lagt ut tre felleskort på bordet, *The Flop* som det heter i poker. Når disse tre kortene er på bordet får man valget igjen om man vil kaste seg, se det høyeste budet eller høyne. Når alle spillerene har gjort sitt valg kommer et fjerde kort på bordet, *The Turn*. Og den samme prosessen utføres igjen til alle spillerene har tatt sitt

valg etter det femte, og siste, kortet er på bordet, *Fifth Street*. Nå legges alle kortene på bordet og en vinner kåres.



Skjermdump

På bildet over ser man at Mads vant runden på *High Card*. Phil kastet seg i første runde, derfor er hans bet kun på $25. Poten på $177 føres da over til vinneren sin konto. Brukeren får så spørsmål om han/hun vil bli sittende ved bordet, er tilfellet at brukeren vil dette startes en ny runde hvis ikke avsluttes spillet. Brukeren kan fortsette å spille frem til han/hun ikke har mer penger igjen eller begge de andre spillerene er tomme for penger.

2. Metoder og funksjoner

KnowWhenToFoldEm bruker relativt enkle metoder for å gjennomføre spillet. Spillet i seg selv kjører i en while-løkke og vil ikke avsluttes før kriteriene for løkken ikke stemmer. Programmet er bygget opp som et objektorientert C++-program, bestående av syv .cpp-filer, seks .hpp-filer og én Makefile.

Verdt å nevne er metoden som regner ut rank-nummeret til den beste hånden en spiller har, *calcHandRank()* i klassen *player.cpp.* Denne metoden tar en *hand* som parameter, nærmere bestemt kortene som ligger på bordet. Først bestemmer programmet hvilket kortene på hånden som er det høyeste og setter dette som High Card for spilleren. Etter

dette kjøres flere tester for å finne den beste sammensettingen av kortene spilleren har på hånden og felleskortene som ligger på bordet. Hendene blir rangert fra høy til lav, det vil si at Royal Straight Flush har høyest rangering, High Card har lavest, osv.

AI-klassen har også en metode som er verdt å forklare nærmere. *getAction()*-metoden er metoden som bestemmer oppførselen til de to kunstige spillerene i pokerspillet. Her skjer det meste tilfeldig, med forbehold så klart. Ut i fra rangeringen på hånden til den kunstige spilleren genereres et random bet. Betet har større interval desto bedre hånden er, det vil si at på en god hånd kan betet som genereres være høyere enn hvis hånden hadde vært rangert lavt. Det er også lagt inn at en AI kan kaste seg på en dårlig hånd.

3. Refleksjon

Prosjekt har bydd på flere utfordringer, noe som er å forvente, når man skal lage et omfattende produkt. Den største utfordringen ble nok omstillingen fra å ha anntatt at programmet skal skrives av to personer til at det skal gjøres individuellt. Tiden ble plutselig for knapp til å implementere alt som var ønskelig.

I løpet av prosjektperioden gikk også mye tid til å finne en god metode for å sammenlikne hendene til de forskjellige spillerene, samt en del tid gikk på å få AI-ene til å oppføre seg på en nogenlunde fornuftig måte. Fra starten var det et mål å lage AI-er med forskjellig personlighet, men som nevnt ovenfor, så ble tiden for knapp da prosjektet mistet den andre studenten. For at AI-ene ikke skulle oppføre seg likt hver gang ble det derfor implementert en del tilfeldigheter for handlingene til disse.

Det ble også satt som mål å ha et grafisk brukergrensesnitt, tanken var å lage dette med Qt, men igjen strakk ikke tiden til for én person å kunne ferdigstille dette.

De aller fleste klasser og metoder i programmet er gjennbrukbare, noe som er viktig med tanke på videreutvikling av programmet og eventuellt gjenbruk av kode til andre prosjekter.

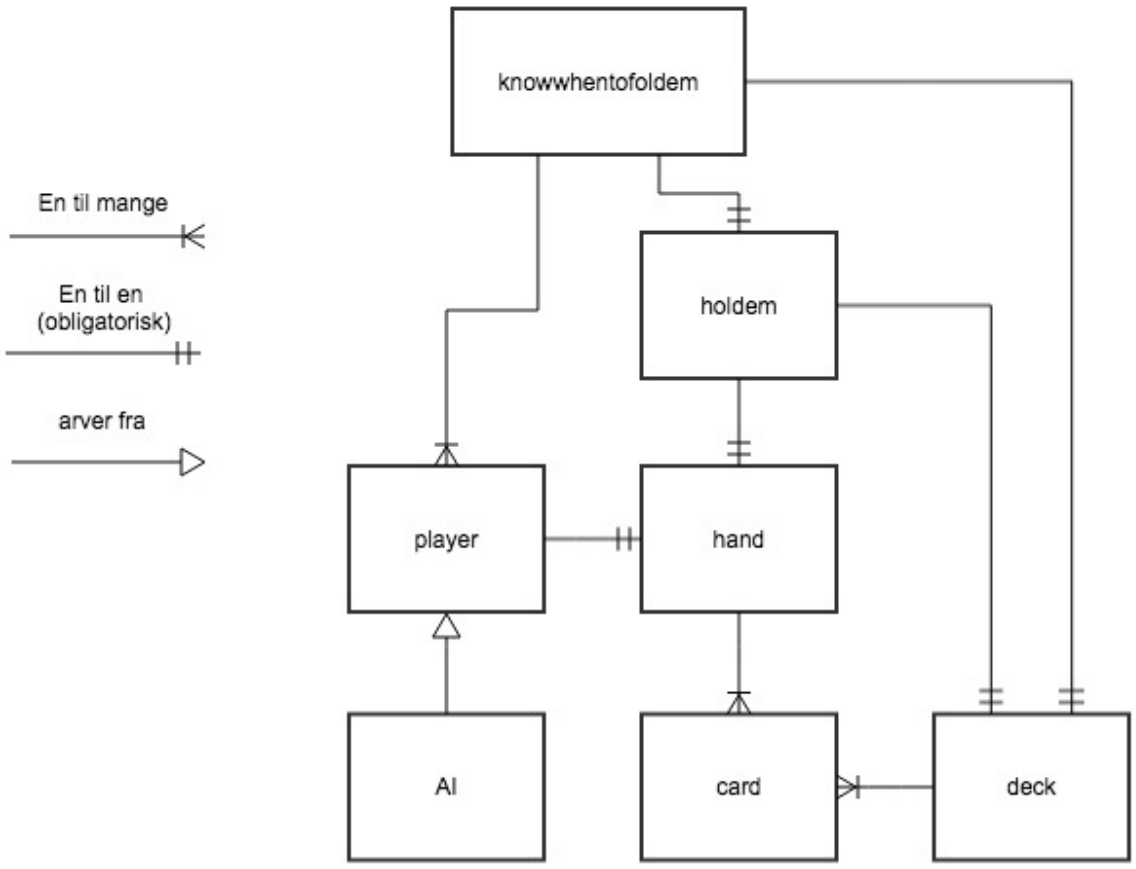
Programmet har fortsatt noen bugs, men det er, per dags dato, ikke funnet noen bugs som hindrer flyten i programmet. Dette kan så klart komme til lys ved senere tidspunkt, men det er noe som forventes ved en slags alpha-release. Én av bugene som er oppdaget er at noen ganger kan AI-ene foreta ulogiske (og ulovlige) bets, som for eksempel å plassere et bet på -2 millioner. Det er lagt inn en funksjon som skal hindre dette, men det er ikke alltid det blir fanget opp. Derfor er det også lagt inn en

tilleggsmetode som gjør at summen av bets på bordet til slutt alltid vil være riktig og ikke hinsides store eller negative tall.

4. Konklusjon

For å konkludere så er sluttproduktet godt, til tross for store forandringer i gruppestruktur. Prosjektet har gitt mye god kunnskap om språket C++ og objektorientert programmering generelt.

5. Vedlegg



Skjermdump: Klassehierarki