Slutrapport

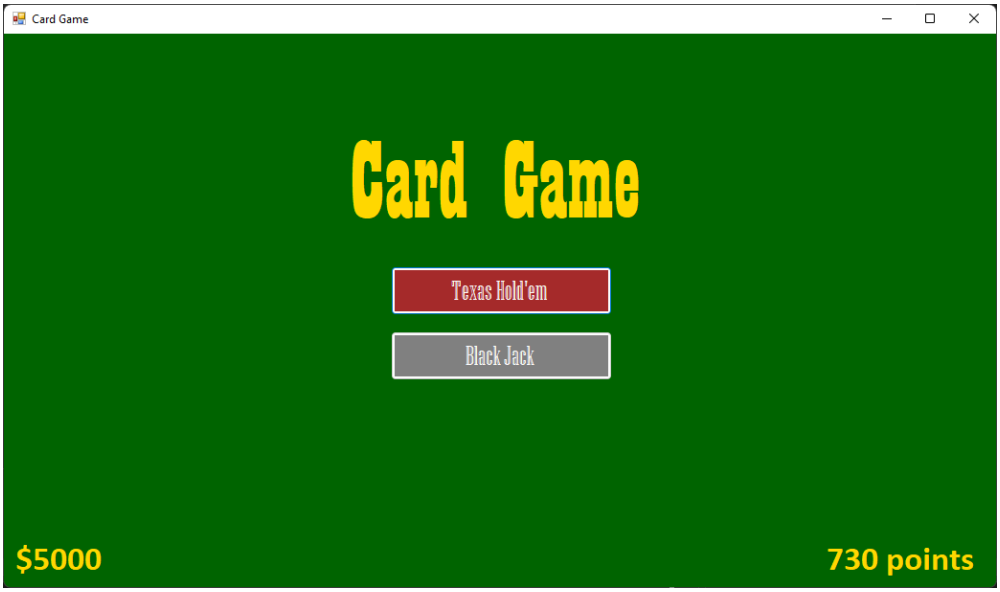
Sammanfattning

Under denna sommar har jag implementerat en Windows Forms C#-applikation som låter användaren spela Texas Hold'em och Black Jack. Genom att vinna i spelen kan användaren samla poäng som sparas när spelet stängs ned. Kunskaper jag fått ifrån projektet är främst en bättre känsla för objektorienterad programmering, men också en bättre känsla gällande logik i spelflöden.

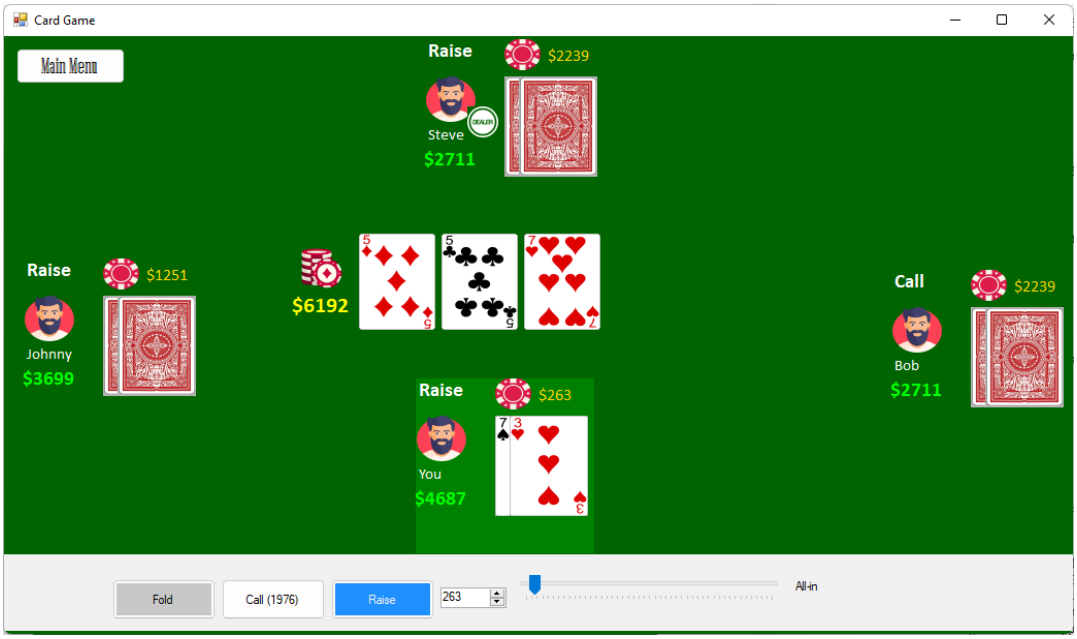
Produktbeskrivning

När programmet startas visas först huvudmenyn, där användaren kan välja antingen Texas Hold'em eller Black Jack. I Texas Hold'em spelar användaren mot tre AI-spelare som självständigt fattar sina beslut utifrån deras chans att vinna varje runda. I Black Jack spelar spelaren mot en Dealer som alltid fattar sina beslut utifrån de regler Black Jack har, dvs. att delarn alltid måste dra ett kort om dennes värde är 16 eller under.

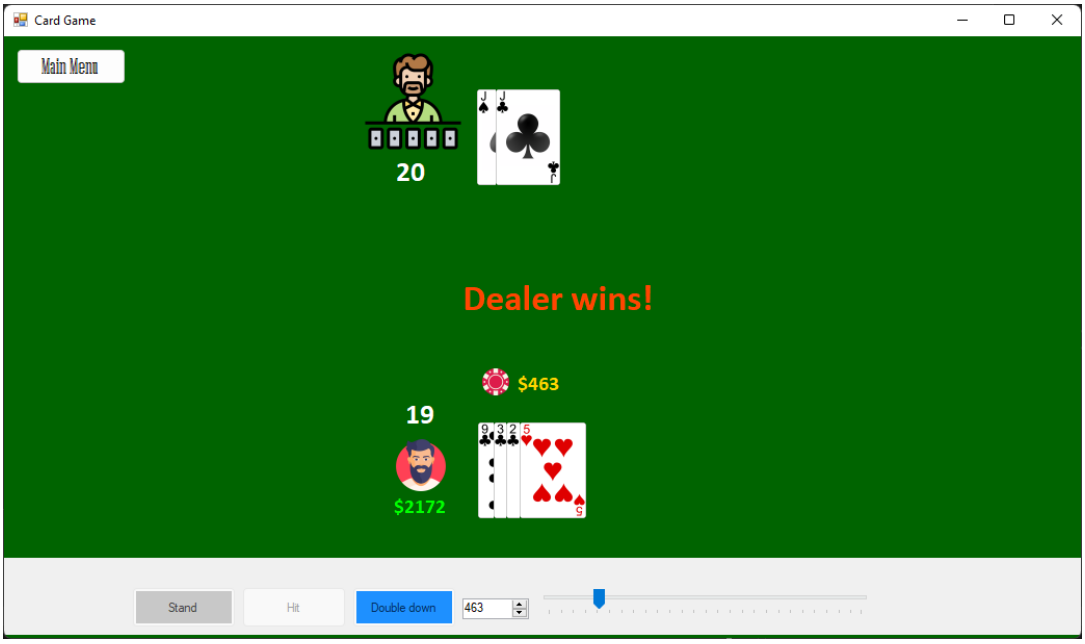
Genom att vinna i spelen kan spelaren samla poäng, vilket sparas till fil när programmet stängs ned. Spelarens pengar sparas också ned till filen.



Figur 1: Huvudmeny



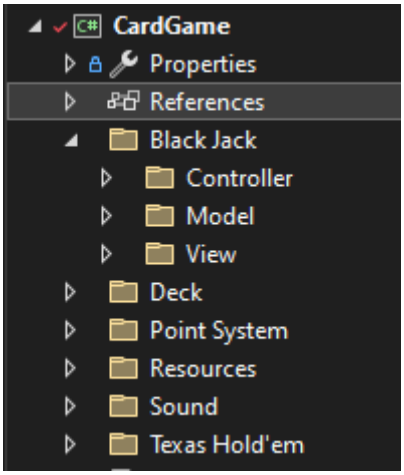
Figur 2: Texas Hold'em



Figur 3: Black Jack

Designstruktur

Under implementeringen har jag använt mig av designmönstret MVC i syfte att hålla isär spellogiken och gränssnittet. Jag har också delat upp programmet i ett flertal olika klasser där varje del har sin egen uppgift.



Figur 4: Mappstruktur

För att inte blanda ihop källkoden till Black Jack och Texas Hold'em har dessa placerats i egna mappar, där varje kortspel har sin egen Controller, Model och View. Anledningen till detta är för att det då blir lättare att skapa nya spel samt att det blir mer ordning för att hitta vart respektive klass ligger. Då både spelen kräver en kortlek har koden till kortleken placerats i Deck-mappen. Detsamma gäller för ljuduppspelning och för poängsystemet.

Produktkvalitet

Jag tycker att programmet håller en ganska hög kvalité just nu, främst gällande Black Jack där jag inte lyckats hitta några buggar. Då Texas Hold'em är mer avancerat är det fler delar som behöver fungera tillsammans för att spelet ska fungera stabilt vilket gör att det kan finnas buggar i spelet just nu. Det finns exempelvis ingen logik som hanterar tillstånd när spelaren inte har några pengar kvar i Texas Hold'em. Jag skulle heller inte påstå att AI:n alltid fattar rätt beslut och en riktigt duktig pokerspelare skulle nog ganska lätt kunna klura ut hur AI:n fungerar och därmed utnyttja detta.

Kunskapsanvändning

Den största kunskapen jag använt i detta projekt kommer nog från kurser som hanterar objektorienterad design samt kursen C#.NET. Jag fick dessutom användning från en C++ kurs jag tog förra sommaren där vi simulerade en kortlek. Stack Overflow har även använts som källa för att hitta lösningar på olika problem.

Framgångar

Tekniska framgångar

Tack vare den objektorienterade designen har det varit väldigt enkelt att återanvända kod mellan de två olika kortspelen, exempelvis kortleken. Jag tycker också koden och mappstrukturen gör det lätt att lägga till nya funktioner i spelet, då koden exempelvis är skriven på ett sätt som gör det lätt att lägga till fler spelare i Texas Hold'em. Jag tycker också jag lyckades få en relativt välfungerande AI, trots om jag är osäker på om detta skulle kunna klassas som en ”äkta AI”.

Projektarbetet

Jag har under arbetets gång alltid haft bra koll på vad som behöver göras härnäst och det har aldrig varit en dag där jag kände mig osäker på nästa steg i processen. Jag tycker även att jag hållit tidsschemat bra då jag fått med funktionaliteten jag önskat.

Problem

Tekniska problem

Det största tekniska problemet under implementeringen var definitivt själva AI:n. Eftersom jag inte arbetat med någon avancerade AI-algoritmer tidigare var det svårt att hitta en lösning som både var effektiv och relativt lätt att implementera. Jag kom tillslut på idén av att låta varje AI simulera spelet ett antal gånger med de kort de kände till så att de kunde fatta besluten på deras chans att vinna en omgång. Ju högre chans att vinna en runda, desto mer är AI:n beredd att satsa.

Ett annat problem som uppkom var gällande kortens värden. Tillexempel är Ess värt 14 i Texas Hold'em medan det är värt 11 eller 1 i Black Jack. För att slippa göra två olika kortlekar har jag skapat klassen CardAdapter som har som uppgift att konvertera kortens värde från Texas Hold'em till Black Jack-värde.

Projektarbetsproblem

Under arbetets gång har jag varit rätt dålig på att hålla koll på tiden. När jag programmerar sitter jag gärna och växlar mellan två saker, typ 30 minuter kodning och sedan 30 minuter annat, vilket gör det svårt att veta exakt hur mycket tid som faktiskt går åt. Sedan jobbar jag också i ganska ojämna spurtar, då jag kan sitta runt 12 timmar en dag för att inte göra något dagen efter.

Arbetsplan

Jag skulle säga att jag nått huvudmålet som var att implementera båda kortspelen Texas Hold'em och Black Jack. Det var dock några små funktionaliteter som inte implementerades, t.ex. planerade jag att olika AI-spelare hade olika personligheter, dvs. att vissa var mer riskbenägna eller försiktigare än andra. Poängsystemet som implementerades var också relativt simpelt och det skulle säkert kunna göras mer tillfredställande.

Sluttidrapport

Total under projektet

Projektdel	Tot
Texas Hold'em	88h
Black Jack	26h
Övergripande (GUI, kortlek + poängssystem)	22h
Dokumentation	12h