Projektbeskrivning

<SchackMatte>

2020-03-12

Projektmedlemmar:
Matthias Gerdin <matge373@student.liu.se>
Lisa Green <lisgr350@student.liu.se>

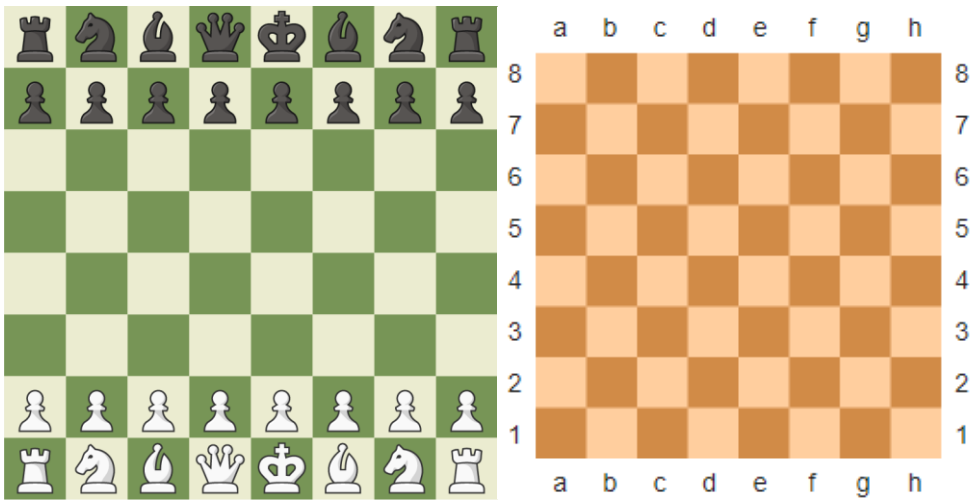
Handledare:
Hugo Cedervall <hugce564@ida.liu.se>

Innehåll

Projektplan

1. Introduktion till projektet

Vi kommer att utveckla ett schackspel. Detta är en typ av brädspel som utgår från en spelplan med 8x8 rutor (se bild nedan). I schack har man 6 olika slags pjäser till sitt förfogande, alla med olika ”förmågor”. Dessa består av 8 bönder, som är den enklaste pjäsen, två av följande; torn, löpare, springare (hästen), samt en kung och en drottning. Detta är ett turbaserat spel där man får flytta en pjäs innan det blir motståndarens tur. Poängen är att man på ett strategiskt sätt ska placera sina pjäser och/eller slå ut motståndarens pjäser tills man hamnar i ett läge där man ”hotar” motspelande persons kung. Detta kallas, i fallet man kan flytta kungen för att rädda den, ”schack”. Har man däremot placerat sina pjäser på ett tillräckligt listigt sätt, så att vad motståndaren än gör så kan den inte flytta kungen ”ur fara”, har man gjort ”schackmatt” och vunnit spelet.



(En komplett spelplan vid start. Spelet läses av som ett koordinatsystem,

där x = {a, b, c, d, e, f, g, h} och y = {1, 2, 3, 4, 5, 6, 7, 8}.)

2. Ytterligare bakgrundsinformation

Schack är ett relativt enkelt brädspel med få, men strikta, regler. Till att börja har varje pjäs en strikt startposition (se bilden ovan), som man alltid utgår ifrån. Utöver detta så har varje typ av pjäs ett specifikt mönster man får röra sig i. Vit spelpjäs börjar alltid spela först, därefter turas man om att göra ett steg varje runda. Man får inte ”passa”, utan måste alltid utföra ett steg. Om man skulle hamna i ett läge där man inte kan göra något ”lagligt” drag, men inte heller står i ”schack”, blir det ett så kallat dödläge och spelet blir oavgjort.

Bönderna får gå ett steg framåt (två steg framåt om det är första gången den rör på sig), och kan ta över motståndarens pjäser endast genom att ta ett steg vertikalt (mot motståndaren). Om en bonde lyckas ta sig till andra sidan planen, alltså om den lyckas ta sig 6 steg framåt till rad 8, respektive rad 1, kan den uppgraderas till antingen en drottning, springare, torn eller

löpare. Denna uppgradering påverkas inte av tidigare uppgraderingar. Man kan alltså teoretiskt sett ha till exempel 9 drottningar på brädet. Bönderna har också ett speciellt drag kallat ”En Passant” (se bild nedan).

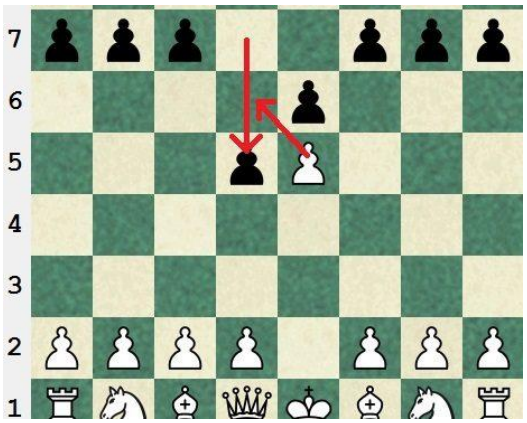
Tornen kan gå obegränsat antal steg vertikalt eller horisontellt, på lediga rutor. Tornet kan tillsammans med kungen, max en gång per spel, tillsammans med kungen utföra något som kallas ”Castling” (se bild nedan).

Kungen kan gå ett steg i alla riktningar, det vill säga: vertikalt, horisontellt och diagonalt.

Drottningen kan gå obegränsat med steg, på lediga rutor, i alla riktningar.

Löparna kan gå obegränsat med steg, på lediga rutor, diagonalt.

Springarna kan röra sig i ett så kallat ”L” mönster i vilken riktning som helst. Detta motsvarar alltså antingen ett steg vertikalt/horisontellt och därefter två steg horisontellt/vertikalt, vice versa.



(En Passant: när en bonde går två steg har motståndaren möjlighet att ”fånga” den med sin bonde, som att den bara skulle rört sig ett steg framåt, och då alltså hamna bakom.)

(”Castling”, det enda tillfället då två pjäser får flyttas samtidigt. Brädet måste vara uppdelat som det ser ut på bilden ovan, inget får vara mellan kungen och tornet. Härefter får man då flytta kungen två steg mot tornet, medan tornet flyttas till positionen mellan kungens nya och gamla position.)

3. Milstolpar

#	Beskrivning
1	Datatyper för spelplan och pjäser.
2	Grafik för att visa spelplanen.
3	Samtliga pjäser är implementerade och kan placeras ut på spelplanen. De kommer ha fasta startpositioner, men vi vill göra det enkelt att ändra om man skulle vilja spela schack med en twist. **I detta skede kan det vara relevant att t.ex visa text istället för att rita ut en bild på en pjäs.**
4	Spelresurser som håller koll på vart pjäser befinner sig, om ett drag är möjligt, om man förlorar en pjäs och dylikt. Just att kolla om ett drag är möjligt kommer inte bli relevant förrän vi implementerat regler för spelpjäserna, men vi kan fortfarande säga att ”position.Free = false => unavailable move” t.ex.
5	Regler för spelpjäser. Hur och var de får förflyttas. Precis som vi fått som tips så kommer vi försöka att implementera mer generella och simpla algoritmer för hur varje pjäs kan förflyttas.
6	Implementation av ”uppgradering” av bönder. Detta med någon typ av menyval så att bonden blir till en av de andra pjäserna.
7	Förflyttning av pjäser med actions. Vi vill kunna använda musen för att markera en pjäs och därefter placera den på en ruta genom att göra ett drag som reglerna tillåter.
8	Tvåspelarläge, alltså något som ser till att vita pjäser börjar spela och att det därefter är motståndarens tur att göra ett drag. Detta kommer implementeras så att man kan spela mot varandra på samma dator. (Ganska simpelt egentligen, varannan gång kan man bara flytta vit eller svart pjäs.)
9	Spelbarhet – tester och buggfixar för att se till att grunden fungerar som planerat.
10	Visa ”giltiga drag” på planen. Så att man enkelt ser vart man kan flytta olika pjäser. De valbara rutorna markeras på något sätt.
11	Välja namn på spelare1 och spelare2.
12	Timer för varje spelare, ska kunna ändras så att man t.ex kan ha 1, 3, 5 eller fler minuter på sig.
13	Startmeny som visas innan spelet börjar. Här ska man kunna välja mellan multiplayer, singleplayer. När man väljer något av de olika lägena ska man ställas frågan om man vill spela med en timer, och i så fall hur lång den ska vara.
14	Visa en ruta med ”döda” spelpjäser.
15	Nätverksstöd så att man kan spela mot varandra på olika datorer.

16	Animation vid vinst/förlust. ”Schackmatt!” Kanske någon ljudeffekt också. Här ska man också kunna välja om man vill spela igen , gå tillbaka till menyn eller avsluta.
17	Implementation av rörelser genom textkommando. Exempelvis a1 → a4, så ska det utföras om det går. Vet ej om detta ska vara här eller vid ett senare skede.
18	Animation för förflyttning av spelpjäser. Ska kunna togglas av och på.
19	Chatt så att man kan prata med varandra medan man spelar
20	Ljudeffekter vid förflyttning av pjäser.
21	”Gamemodes”? Schack med en twist. Exempelvis större spelplan, fler av en viss pjäs etc. Ska kunna väljas vid startmenyn.
22	Animation när pjäser ”dör”. Ljud?
23	Alternativa utseenden på pjäserna. Vill man ha star-wars tema? Då är stormtroopers bönder och Darth Vader kung. Kul ju!
24	

4. Övriga implementationsförberedelser

Likt Tetris vill vi ha en klass som har hand om boarden och vad som sker här. Tanken är att boarden i sig ska ha koll på när en pjäs förflyttar sig.

Någon form av viewer-klass där vi kan visa våra komponenter grafiskt. Framför allt spelplanen. Här kommer vi också behöva någon form av uppritning som känner igen om t.ex ”position x = pawn” → rita ut en bonde. Detta kanske kräver en separat, grafisk, klass.

En klass som för sig hanterar de olika pjäser som finns. Kanske en Enum där vi enkelt kan välja vilken typ av pjäs vi vill ha. Switch där vi kan kolla ”vilken typ av pjäs?” och därefter ge den de regler som den måste följa. Ett alternativ är att vi hanterar varje pjäs för sig i en varsin separat klass.

En separat klass för reglerna för hur en pjäs får röra sig? Antingen detta eller så har vi detta i board-klassen. Det känns dock som att koden blir mer strukturerad om man har någon form av regel-klass.

5. Utveckling och samarbete

- Vi båda har liknande ambitionsnivå och vill såklart färdigställa minst kärnan + nätverksspelandet innan deadline.
- Arbetstiderna kommer variera. Vi har samma schema och kör på med detta ganska flytande. Tanken är att vi ska gå på samtliga resurstillfällen såvida inte något annat dyker upp som ger någon av oss förhinder. Det kan ibland hända att vi jobbar kvällar eller helger, men förmodligen inte om vi känner oss i fas.

- Ingen specifik tidsplanering för när de olika delarna ska vara. Detta kanske vi skulle kunna få lite hjälp med?
- Vårdcentralsbesök, lägenhetsvisning eller annat oväntat.
- Vi jobbar nog inte alltid tillsammans utan kommer dela upp olika delar sinsemellan och jobba separat. Men vi sitter såklart så mycket med varandra som möjligt för att båda ska vara på samma nivå i förståelsen och för att vi ska kunna diskutera de initiala problemen/utmaningarna.

6. Implementationsbeskrivning

6.1. Milstolpar

Milstolparna 1–9 och 11 är helt implementera. Alla utom milstolpe nummer 7 är implementerade ganska exakt enligt dess beskrivningar i avsnitt 3. Nummer 7 beskriver att förflyttning av pjäser ska ske genom att först markera pjäsen man vill flytta och där efter trycka på den ruta man vill flytta den till. Istället för detta flyttas nu pjäserna genom att man drar dem dit man vill flytta dem. Varken milstolpe nummer 10 eller någon efter 11 är implementerade.

6.2. Dokumentation för programstruktur, med UML-diagram

- **Övergripande programstruktur**

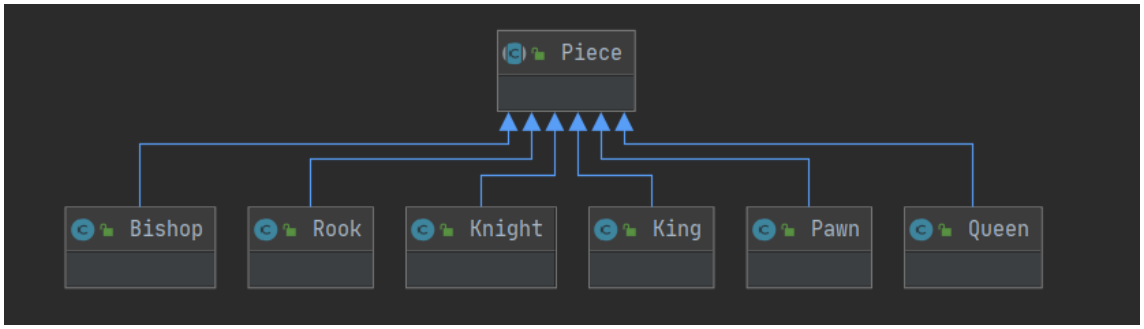
Vi har implementerat ett brädspel som beroende på vilket tillstånd det är i, tillåter att pjäser med en viss färg får flyttas. När användaren försöker göra ett drag kontrolleras det först om det är tillåtet. Om det är det så flyttas pjäsen dit användaren dragit den. Efter varje drag kontrolleras om spelet är slut, det vill säga om det var ett vinnande drag. Om inte ändras tillståndet och det är nu endast möjligt att flytta på pjäser av den andra färgen.

- **Implementering av regler och förflyttning**

Förflyttning av pjäser sker med hjälp av en MouseListener. Klassen PieceMove förlänger den redan befintliga MouseAdapter och override:ar tre metoder där ifrån. MousePressed(), mouseDraged() och mouseReleased(). I mousePressed() deklareras vilken pjäs som spelaren vill flytta. Med hjälp av information om bland annat vilken slags pjäs det är (Pawn, Queen, Bishop osv) och om det är första gången den flyttas (används bara i Pawn), kontrollerar sedan MouseReleased() om draget spelaren försöker göra är giltigt. Detta görs genom att jämföra destinationsrutan med en lista legalMoves som varje pjäs har. Vissa pjäser har liknande rörelsemönster och kan därför dela på den kod som lägger in positioner i legalMoves. Dessa metoder, som delas, ligger i superklassen Piece. Pjäserna pawn och knight där emot har så pass säregna rörelsemönster att metoderna som skapar deras legalMoves ligger i den egna pjäs-klassen. Dvs i Pawn respektive Knight.

- **Pjäser**

Pjäserna i schack har så pass mycket gemensamt att en superklass Piece och subklasser (Pawn, Bishop, Knight, Queen, King och Rook) är gynnsamt. Den abstrakta Piece innehåller alltså både fält och metoder som alla eller flera av dess subklasser drar nytta av. Dessa egenskaper är färg, typ av pjäs, pjäsens position i både x- och y-led och en URL väg till de bilder som används för att representera varje typ av pjäs. Några av metoderna i Piece nämndes i stycket om regler och förflyttning, men nämnvärt är också att Piece även har en abstrakt metod updateLegalMoves() vilken alla pjäser implementerar, men med olika variationer till den.



(Diagram över pjäs-klassernas relation till varandra vilken beskrivs ovan)

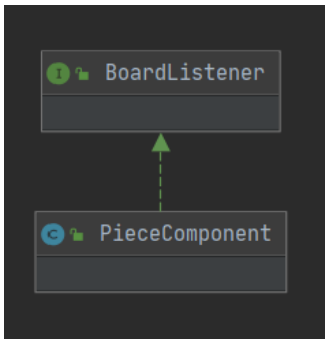
- **Spelbrädet**

Den klass som har primärt ansvar över spelbrädet är Board. Fälten i board innehåller i princip all information om hur brädet ser ut just nu. Det är också Board som placerar ut pjäser när spelet initieras och som håller koll på spelets state, dvs vems tur det är.

- **GUI**

Klasserna PieceComponent, PiecePainter, Frame och Panel sköter, men hjälp av BoardListener, alla grafiska delar av programmet. PieceComponent implementerar interfacet BoardListener vilken hanterar alla ändringar som sker på spelbrädet. Den förlänger även den redan existerande klassen JComponent. Från JComponent har PieceComponent override:at metoderna boardChanged(), getPreferedSize() och paintComponent(). BoardChanged() målar om hela brädet varje gång något på det ändras. GetPreferredSize() väljer dimensioner till spelbrädet och paintComponent() sköter uppritningen av både brädet (bakgrunden) och pjäserna på brädet. Just uppritningen av pjäserna görs genom att kalla på metoderna som finns i PiecePainter.

Frame skapar fönstret som spelbrädet målas upp i. Panel hanterar den ruta som visar vems tur det är att spela.



(PieceComponent implementerar BoardListener)

- **Övriga klasser**

För att möjliggöra listorna med legalMoves skapade vi vår egen klass Position. Innehåller enbart två fält (x och y) och getters till dessa två.

Enum-klassen PieceType används först vid utplacering av pjäserna till sina startpositioner och sedan främst till att identifiera vilken slags pjäs ett objekt har. Fältet type i pjäs-objekten innehåller alltså den enum som motsvarar pjäsens klass. Det vill säga en av de sex olika enumen PAWN, QUEEN, KING, KNIGHT, BISHOP, ROOK och EMPTY. Dessa enums återfinns även i en array squares i Board som motsvarar alla rutor på brädet, om de är tomma och om inte, vad som står där.

7. Användarmanual

- Starta spelet

Schackspelet startas genom att köra main-metoden i klassen BoardTest. När det har startats kommer det först upp två rutor där man får skriva in namnen på personerna som ska spela. Där efter kommer ett fönster med schackbrädet upp. Längst ner i detta fönster finns en panel som meddelar vilken spelare som börjar spela. Vit börjar alltid.

- Spelets gång

För att göra sitt drag drar man valfri pjäs till den ruta man vill flytta till. Om det föreslagna draget är tillåtet flyttas pjäsen när du släpper musknappen, annars stannar den kvar där den stod. När ett tillåtet drag har genomförts blir det den andra spelarens tur och panelen längst ner visar dennes namn istället. Om någon spelare under spelets gång lyckas lytta en av sina bönder (Pawn) till motsatta sidan av brädet, kommer ett fönster med alternativ att komma upp. I detta fönster kommer användaren att få välja vilken annan pjäs som denna bonde ska ”omvandlas” till. Man klickar på den ruta som motsvaras sitt val och bonden byts då ut mot den valda pjäsen. Den finns ett speciellt drag, castling, som beskrivs i slutet på avsnitt 2. Det draget genomförs genom att dra kungen till den rätta rutan så kommer respektive torn att flyttas automatiskt.

- Vinna spelet

Spelet fortsätter tills programmet känner av att någon står i schackmatt och meddelar då vem som är vinnaren.