Projektplan för Breakout

Introduktion till datalogi

Anders Hagward och Fredrik Hillnertz

Innehåll

1 Projektplan

1.1 Programbeskrivning

Programmet skall underhålla och stjäla tid från användaren. Detta genom att låta denne styra ett racket för att upprepade gånger studsa en boll mot ett antal förstörbara block. Lär även vara gynnsamt för användarens pricksäkerhet och reflexer.

1.2 Användarbeskrivning

Användaren förutsätts ha "lagom" datorvana, det vill säga vetskap om hur man startar en dator, kör ett program och interagerar med mus och tangentbord. En ungefärlig åldersskala är 6-60 år. Användaren bör även ha grundläggande kunskaper i det engelska språket.

1.3 Användarscenarier

Nedan följer två vanliga scenarier.

1.3.1 Scenario I

1. Användaren startar programmet och möts av en huvudmeny med menyvalen *New game*, *Settings*, *Help* och *Quit*.
2. Användaren väljer det första alternativet och en ny vy. Denna består av ett racket längst ned som går att styra med musen, en boll fastsittandes mitt på racketet, en hord av förstörbara block längst upp samt solida väggar och tak.
3. Användaren trycker på vänster musknapp och får bollen att fara iväg uppåt, studsandes mot blocken, väggarna och taket.
4. Blocken försvinner successivt i och med att bollen studsar på dem.
5. Alla block har slutligen försvunnit och användaren vinner spelet.
6. En ny 'karta' presenteras (det vill säga, blocken är uppställda i en annan formation än tidigare).
7. Användaren väljer att avsluta spelet genom att trycka `Esc` för att få upp spelmenyn och sedan välja *Quit*.

1.3.2 Scenario II

1. Användaren startar programmet och möts av samma huvudmeny som i föregående scenario.

2. Användaren väljer menyvalet *Help* och får reda på information om hur spelet fungerar.
3. Användaren backar tillbaka till huvudmenyn genom att trycka `Esc`.
4. Användaren väljer menyvalet *Settings* och väljer att stänga av ljudet.
5. Användaren backar återigen till huvudmenyn på samma vis som tidigare men väljer nu *New game*.
6. Samma spelvy som i föregående scenario visas och användaren spelar på samma sätt.
7. Användaren är inte så pricksäker och lyckas inte få bollen att studsa på racketet. För varje gång han missar förlorar han ett 'liv'.
8. Användaren förlorar tre liv och därmed även spelet.
9. En meny med valen *Play again*, *Settings*, *Help* och *Quit* visas. Användaren väljer det sistnämnda.

1.4 Testplan

Förutom de två scenarierna ovan kommer menyerna testas ordentligt, så alla menyval fungerar korrekt. Inställningsmöjligheterna kommer testas, så att ljudeffekterna verkligen försvinner när man gör den inställningen, och verkligen kommer tillbaka om man gör inställningen igen.

Själva spelmekaniken kommer givetvis testas grundligt så att inte bollen verkligen studsar mot väggarna, taken, blocken och racketet, så att blocken försvinner vid kontakt med bollen, så att ett 'liv' försvinner och att bollen återställs på racketet vid varje 'miss'. Vidare skall testas så att rätt meny med fungerande menyval dyker upp vid förlorat spel, så att en ny bana laddas när alla block är borta, så att en poänglista visas när man klarat alla fem banor och att den nya poängen sparas.

Självklart skall spelet testas av en tredjepart.

1.5 Programdesign

Programmet kommer bygga på spelbiblioteket *Slick*. Nedan följer en lista på de grundläggande klasserna och dess viktigaste metoder.

- `MainMenuState, GameplayState`: Då biblioteket Slick använder sig av så kallade *states*, eller *tillstånd*, kommer minst två 'state'-klasser implementeras. Deras viktigaste metoder är `init`, där alla bilder och ljud laddas, `render` där allt ritas upp på skärmen och `update` där alla beräkningar sker (för kollisioner, positionsuppdateringar med mera).

- `GameObject`: En abstrakt klass som representerar ett objekt på skärmen. Viktigaste metoder: `getImage, setImage` som hanterar objektets bild, `getXPos, getYPos` returnerar objektets position.
- `Ball, Block, Racket`: Klasser som ärver `GameObject`.
- `Movable`: Ett interface med metoden `move`, där implementerande klasser får beskriva hur objektets position skall uppdateras.
- `Highscore`: Sköter all skrivning och läsning till och från textfilen där poängen sparas. Viktigaste metoder är `addScore` som lägger till ett poäng, `getScores` som returnerar en lista med poängen.
- `BreakoutGame`: 'Huvudklassen' som innehåller `main`-metoden.

1.6 Tekniska frågor

Nedan följer en lista med tänkbara tekniska problem och eventuella lösningar.

Problem: Vid eventuellt bildbyte på objekt medan spelet körs (till exempel om ett block kräver flera bollstudsar för att försvinna) skulle en viss fördröjning kunna uppstå innan programmet har läst in den nya bilden. **Lösning:** Ladda in flera bilder för ett objekt vid initiering?

Problem: Hur 'kartorna' skall representeras. **Lösning:** Läsa in dem från filer?

Problem: Exakt vilka klasser som skall finnas och vad var och en skall sköta i spelet, samt undvika att skapa klasser det redan finns motsvarigheter till i spelbiblioteket. **Lösning:** Träna objektorientering, tänka efter ordentligt och läsa bibliotekets dokumentation.

Problem: Hur skall programmet bete sig om någon väsentlig fil saknas? **Lösning:** Ljudfil - meddela användaren, highscorefil - skapa en ny fil utan att meddela användaren, kartfil eller bildfil - meddela användaren och avsluta.

1.7 Arbetsplan

Arbetet kommer delas upp på det viset att var och en får några bestämda klasser att implementera. De skall implementeras efter en specifikation som bestäms i förväg så att båda har koll på vad de gör och vad de skall användas till. Om eventuella problem skulle uppstå och en fastnar med någon del av implementationen kan givetvis den andre hjälpa till.

Arbetsplanen är upplagd på följande vis:

7 - 14 april: Nödvändiga klasser skall skapas och grundläggande spelmekanik implementeras. Spelet skall vara 'minimalt' spelbart, det vill säga att racketet skall kunna styras och bollen skall studsa korrekt mot de övriga objekten.

15 - 22 april: I första hand skall spelmekaniken putsas på. I andra hand skall ljud, poängsystem, inställningsmeny och hjälpmeny implementeras (så många som hinns med).

23 - 30 april: Slutrapporten skall skrivas och eventuell putsning av spelet skall göras inför prototypvisning.

1 - 7 maj: Eventuell implementation av detaljer som inte hunnits med samt finputsning.