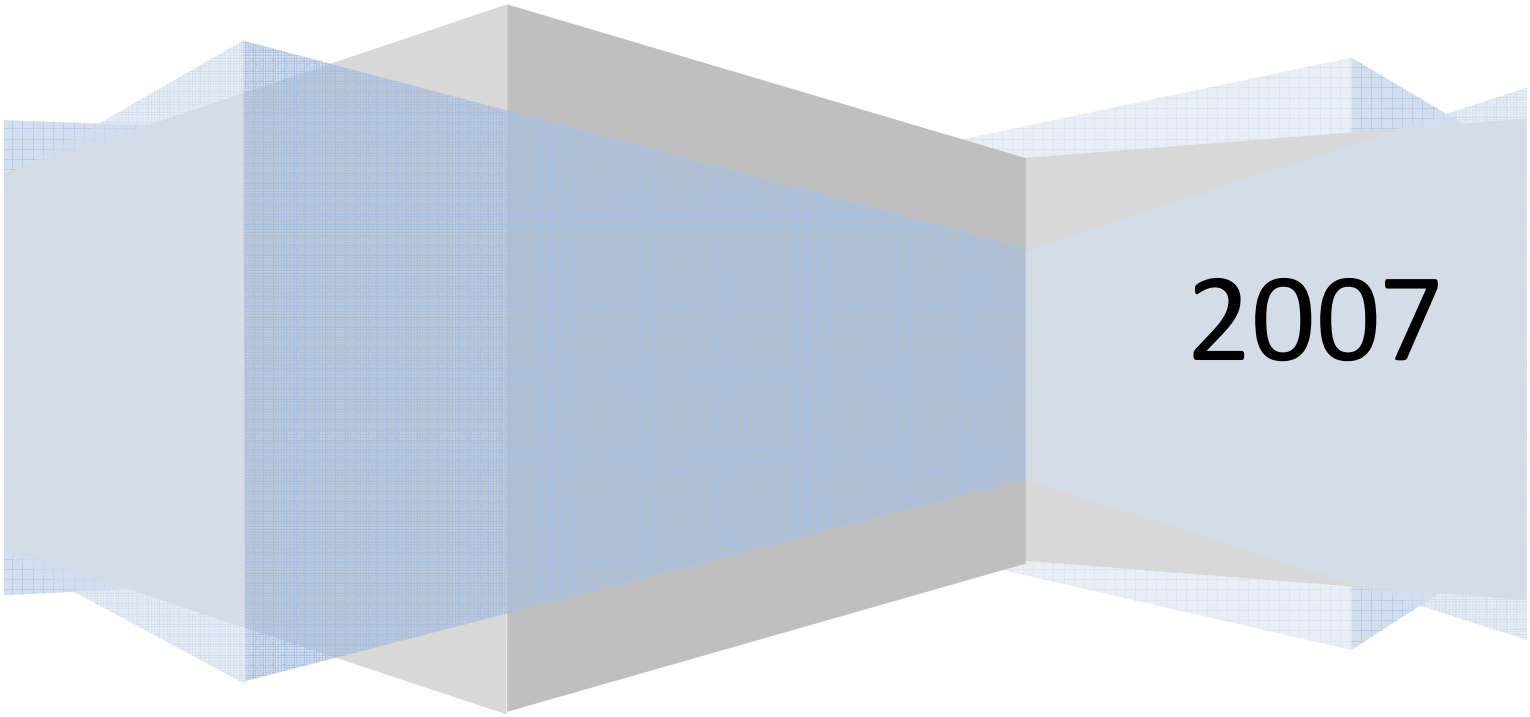
Linköpings Universitet

Designspecifikation

Projektarbete Arkadspel

Jimmy Dahl och Serdar Tovi

TDP005 – Projekt: Objektorienterade system



2007

Visualisering

När spelet startas möts spelaren av en meny med två alternativ, vilka väljs med ett tangentkommando. Spelaren kan välja att starta första nivån av spelet, visa High Score-listan eller avsluta spelet. De kommandon som används för att navigera dit spelaren önskar framgår av text skriven i menyn.

High Score-listan kommer även den att ha en bakgrundsbild och ett utskrivet kommando för att gå till menyn. Själva listan, med namnen på de spelare som fått högst poäng, kommer att bestå av text. Namnen kommer att vara uppdelade i två kolumner (när dom blir tillräckligt många) och framför varje namn kommer spelarens poäng att skrivas ut. Listan är självklart sorterad efter högsta poäng.

Själva spelet kommer att innehålla flera visuella objekt varav de flesta består av bilder. Spelarens kanon kommer att vara en bild på en stridsvagn och de fientliga flygplanen kommer att använda en bild på ett flygplan. De skydd som spelaren kan gömma sin kanon under kommer att använda en bild på en avlång vägg. Även kanonens skott och flygplanens bomber kommer att använda passande bilder.

Om ett flygplan blir träffat av ett skott från kanonen kommer bilden på flygplanet (och dess objekt) att tas bort. När ett flygplan tas bort skapas det antingen ett nytt flygplan eller, om samtliga flygplan på nivån har skjutits ner, så påbörjas nästa nivå. Blir spelaren kanon träffad av en bomb så avslutas spelet.

Utöver de olika fiende- och spelarobjekten kommer det även finnas två textade visuella objekt i fönstrets övre kant. Där kommer poäng och nivå att skrivas ut med siffror och bokstäver. På samma ställe står även spelets namn utskrivet.

Interaktion

Spelaren ska enbart använda tangentbordet för att interagera med spelet. Eftersom spelet inte har så många olika moment kommer bara piltangenterna, mellanslag och retur att behöva användas. För att avsluta spelet används antingen escape-tangenten eller fönstrets X.

Det första spelaren möts av är menyn där spelaren kan välja att starta spelet eller visa High Score-listan. Alternativen representeras av 2 tangentkommandon, mellanslag och retur. Spelaren använder tangentkommandot för att navigera vidare. När High Score listan visas kan spelaren bara välja att gå tillbaka till menyn. Spelaren går tillbaka till meny genom att klicka på retur-tangenten.

När spelaren har valt att starta spelet från menyn kommer spelets första nivå att starta. Spelaren styr då sin kanon med piltangenterna höger och vänster. För att flytta kanonen lite åt ena hållet klickar spelaren på aktuell piltangent och för att flytta kanonen en längre bit håller spelaren aktuell piltangent nertryckt. När spelaren vill avfyra ett skott från sin kanon klickar spelaren på mellanslagtangenten.

När samtliga nivåer avklarats eller spelarens kanon har träffats och spelet avslutats kommer spelaren att få ange sitt namn. Namnet skrivs till High Score-listan som sparas på fil. Namnet skrivs in med hjälp av bokstavs-, mellanslag- och siffertangenterna och godkänns med retur. De 25 bästa resultaten i den sparade filen kommer att visas i High Score-listan i spelet.

Klassdiagram

(se bilaga för klassdiagrammet)
Main-funktionen i programmet börjar med att skapa en spelmotor av klassen *GameEngine*. Denna klass är vad som håller spelet igång, tack vare att spelets loop är en av GameEngines funktioner. GameEngine har ett *GameState*-objekt som håller ordning på i vilken fas spelet är, flyttar mellan spelets olika faser och, i ett fåtal fall, anpassar spel-loopen därefter.

GameEngine initialiserar SDL och sätter upp spelets fönster. GameEngine tar dessutom hand om all inmatning och vidarebefodrar uppgifterna till aktiv GameState. Dessutom har GameEngine en funktion för att ladda in bilder i spelet och anpassa dem efter spelets fönster. Denna funktion anropas från alla andra instanser som använder bilder.

GameState är basklassen för samtliga av spelets faser. GameState känner till GameEngine och kan därmed anropa dess funktioner. GameState har 5 olika underklasser, en för varje fas i spelet. 4 av dessa faser är olika menyer, den femte är spel-fasen. GameState innehåller virtuella funktioner som underklasserna omdefinierar. Den första GameState-underklassen som skapas är StateMenu. Den skapas i spelets main-funktion. Resterande GameState:s skapas av föregående GameState innan eller samtidigt som GameEngine blir upplyst om detta.

StateMenu är en underklass till GameState och den första GameState-klass som skapas i spelet. Denna klass innehåller bara två funktioner: en för att rita menyn (en bild) och en för att skicka vidare till nästa fas beroende på tangentkommando.

StatePlay är även den en underklass till GameState. StatePlay har ett antal *Sprite*-objekt och funktioner för att skapa, rita och flytta dessa. StatePlay har också två instanser av klassen Text. Några av Sprite-objekten skapas redan då StatePlay initieras, medan Sprite-objekten bullets och bombs skapas under spelets gång. Objekt av typen *BulletSprite* och *BombSprite* lagras i varsin vektor. Funktionen för att rita alla Sprites anropar aktuella Sprites Draw-funktioner och funktionen för att flytta alla Sprites anropar aktuella Sprites SetPos-funktioner med den nya positionen som argument. StatePlay har dessutom en funktion för att omvandla en integer till en string så att Text-objektens renderingsfunktioner kan rendera texten.

StateGameOver har en funktion för att rita ut bakgrund och för att ta emot tangentkommando för att gå vidare till nästa fas, *StateAddScore*. När StateGameOver initieras sätter den en variabel i GameEngine som gör att spelloopen tar emot text från tangentbordet. På detta vis kan spelaren skriva in sitt namn.

StateAddScore lägger till spelarens namn och poäng i High Score-listan. Namnet sätts av GameEngine och poängen vidarebefodras från StatePlay. StateAddScore har en funktion för att rita ut sin bakgrund men framför allt funktioner för att läsa från och skriva till filen med High Score-resultaten. Dessutom finns en funktion för att sortera resultaten och en för att omvandla poängen från objekt av typ integer till string.

Nästa fas i spelet är *StateShowScore*. Här skrivs de 28 bästa resultaten från High Score-filen ut. Även denna klass har en funktion för att skriva ut sin bakgrund. Dess viktigaste funktion är dock Read-funktionen som läser in de 28 översta resultaten från High Score-filen. Varje inläst rad sparas i en vektor. Alla element i denna vektor renderas sen till bilder som sparas i en ny vektor och ritas ut.

Utöver alla spelfas-klasser finns det en basklass för alla sprites och en klass för all text som ska renderas. Textklassen har en variabel för vilken font som används och variabler för position. Klassen har en funktion för att rendera texten, som tar en sträng som argument, och en funktion för att rita ut den renderade texten.

Basklassen för alla sprites innehåller variabler och funktioner för att sätta position för en sprite och för att spara en bild på en sprite. Dessutom innehåller basklassen ett flertal virtuella funktioner som alla underklasser behöver. De sprites som skiljer sig något från basklassen har egna klasser som ärver av Sprite-klassen, men i vissa fall (t.ex. background och shield) används basklassen direkt eftersom alla nödvändiga variabler och funktioner redan finns.

AirplaneSprite och *PlayerSprite* är klasserna för alla flygplan och för spelarens kanon och innehåller enbart funktioner för att förflytta objekten på skärmen. BombSprite och BulletSprite har också de funktioner för att förflytta objekten på skärmen och har dessutom funktioner för att sätta och hämta en variabel som talar om ifall objektet ska tas bort eller inte.

Aktivitetsdiagram

(se bilagor för samtliga aktivitetsdiagram)
Aktivitetsdiagrammen visar hur flödet för de olika spelfaserna fungerar (Game States Playing), hur man navigerar mellan menyerna (Menu Navigation) och flödet för spelloopen som körs när man spelar spelet.