**Designspec för Projekt i kurs TDDI02**

**Grupp Lingonsylt 21/10-2015**

**Av Robert Habrman, Ludvig Jonsson, Sixten König, Saleh Salim**

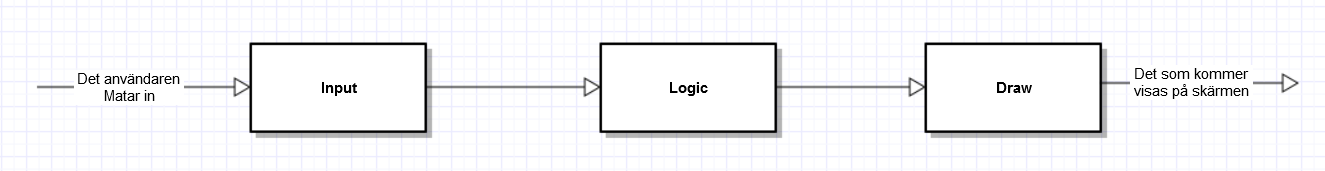
**Innehållsförteckning**

1. **Introduktion**
2. **Arkitektur**
   1. **Klassheirakin GameObject**
      1. **Beskrivning av ObjectPlayer**
      2. **Beskrivning av ObjectWeapon**
      3. **Beskrivning av ObjectBullet**
      4. **Beskrivning av ObjectWall**
   2. **Klassheirakin av Weapon**
   3. **Beskrivning av Gamelogic**
   4. **Klassheirakin State**
   5. **Beskrivning av Menu**
   6. **Beskrivning av Game**
   7. **Beskrivning av Pause**
   8. **Beskrivning av Data**
3. **Klassrelationer**
   1. **Interaktioner mellan olika GameObjects**
      1. **ObjectPlayer och ObjectWeapon**
      2. **ObjectPlayer och ObjectBullet**
      3. **ObjectPlayer och ObjectWall**
      4. **ObjectBullet och ObjectWall**
   2. **ObjectPlayer och Weapon**
   3. **Weapon och ObjectBullet**
   4. **Menu och Data**
   5. **Data och Game**
   6. **Game och Gamelogic**
   7. **Specifikation av Protokoll**
4. **Detaljerad Teknisk Specifikation**
5. **Design av användargränssnitt**
6. **Design av Databas/Strukturer**

**1. Introduktion**

Det projekt som grupp lingonsylt valt att göra i kursen TDDI02 är att i C++ med det utökade biblioteket SFML göra ett Spel. Spelet kommer vara ett multiplayer spel som är begränsat till två personer. Spelet går ut på att användarna spelar mot varandra och tävlar om vem som kan eliminera den andre spelaren först, genom att plocka upp ett vapen och skjuta mot sin motståndare. Användarna kan genom strategiska drag utnyttja banans upplägg för att utmanövrera motståndaren. Detta designdokument är baserat på den tidigare skrivna kravspecifikationen som grupp lingonsylt skrivit och kommer att beskriva en design som ska uppfylla kravspecifikationens krav.

**2. Arkitektur**



Spelet kommer att gå igenom följande loop som visas ovan under programmets användning. Spelet kommer ha en modul som kollar vad användaren matar in. Dessa indata skickas sedan till Logic som kommer att hantera den och manipulera de objekt som kommer att målas ut på skärmen. När objekten är manipulerade så skickas dessa vidare till Draw som med hjälp av Biblioteket SFML ritar ut våra objekt på skärmen.

För att underlätta programmeringen av detta spel så har det blivit uppdelat i följande klasser:

* Object
  + ObjectPlayer
  + ObjectWeapon
  + ObjectBullet
  + ObjectWall
* Weapon
* Gamelogic
* State
  + Menu
  + Game
  + Pause
* Data

**2.1 Klasshierarkin GameObject**   
   
Den abstrakta Gameobject-klassen kommer tillsammans med sina underklasser att vara en klassheirarki som används av programmet för att skapa alla synliga objekt på skärmen. Alla object kommer enligt basklassen att ha en: position, storlek och grafik. Gameobject består av underklasserna : objektplayer, objektweapon, objektbullet och objektwall som kommer att ha unika egenskaper.   
 

**2.1.1** **Beskrivning av ObjectPlayer**    
   
ObjectPlayer-klassen ska innehålla all information om sin spelare. Spelare ska kunna skapas och tas bort. Eftersom spelet ska spelas av minst två spelare åt gången så kommer det alltid skapas två ObjectPlayers i början av matchen. Alla variabler som har med spelaren att göra lagras i den här klassen. De variablerna som ObjectPlayer innehåller är:

* + - Vilken riktning spelaren har i grader
    - Spelarens position
    - Hastigheten spelaren har för tillfället
    - Lagring av vilka sex tangenterna som kontrollerar spelaren
    - Grafiken för spelaren i form av en texture samt en sprite
    - Namn på spelaren som sattes i menyn
    - Vilket vapen spelaren har

Spelaren har också ett antal konstanter lagrade i sig:

* + - Farten som spelaren decelererar och accelererar med.
    - Maxhastigheten för spelaren.

Spelaren kontrolleras alltså av sex olika tangenter. Fyra av dessa kontrollerar spelarens riktning, man kan trycka två knappar samtidigt för att gå diagonalt. De resterande knapparna används för att plocka upp/slänga ifrån sig vapen och för att använda vapnet.

**2.1.2** **Beskrivning av ObjectWeapon**    
   
ObjectWeapon-klassen representerar de olika vapnen när de befinner sig på kartan. Klassen innehåller information såsom vart på kartan vapnet är placerat samt information om vilken typ av vapen det är. En spelare ska kunna gå till vapnet och trycka på en knapp för att plocka upp det. Informationen om vilket vapen som spelaren plockat upp tilldelas då till spelaren i form av ett Weapon objekt. Vapnet försvinner ifrån kartan och spelarens vapen sätts till vapnet som plockades upp.

**2.1.3** **Beskrivning av ObjectBullet**    
   
ObjectBullet-klassen ska innehålla information om vem som sköt skottet och hur den ska agera. Bullets ska kunna skapas och tas bort. Bullet klassen kommer att kunna skapa ett bulletobject när en spelare vill skjuta, bulleten kommer att konstrueras enligt de egenskaper som ligger i spelarens vapen. Klassen kommer lagra data om bulleten som variabler för att hålla koll på kulans hastighet, riktning, position och eventuella andra egenskaper som det valda vapnet kommer ha samt vilken spelare bulleten tillhör.

**2.1.4** **Beskrivning av ObjectWall**  
   
ObjectWall-klassen innehåller all information om spelets alla väggar som skapats. Väggar ska kunna skapas när spelet startas enligt förbestämda kartor som väljs av spelarna innan spelet startar. Väggar ska inte andra objekt kunna ta sig igenom och inga objekt ska kunna skapas i en vägg. Väggobjekten ska tas bort när spelet är slut.   
 

**2.2 Klassheirarkin Weapon**  
   
Weapon-klassen är en basklass för en klassheirarki för alla vapen i spelet. Alla vapen kommer att vara ett arv ifrån basklassen weapon så t.ex. kommer vapen "ett" vara en egen klass, vapen "två" sin egen osv. Ett Weapon ska kunna skapas och sättas till en spelare när den plockar upp ett vapen och ska tas bort från spelaren när spelaren släpper vapnet. Weapon kommer att hålla reda på all information om vapnen som dess: Namn, bulletsize, bulletspeed, ammocount, firerate, spread, ägare, ljudeffekter och unika egenskaper. Weapons data kommer att behöva skickas ut till ObjectBullet för att hjälpa till att skapa just det vapnets bullet. Vid skapandet av en bullet kommer även en ljudfil att spelas upp beroende på vilken underklass av weapon som man befinner sig i.

**2.3 Beskrivning av Gamelogic**

Gamelogic är den klass som kommer innehålla all logik i spelet och är den klass spelet kommer att kommunicera med för att se att allt stämmer i game. Exempel på saker gamelogic kommer att kolla på är: kollision av GameObjects, om spelet är slut eller inte, räkna ut objekts hastighet och riktning, vart ObjectWeapons kommer placeras på kartan osv.

**2.4 Klassheirakin State**

State-klassen kommer att vara en klasshierarki som kommer att bestå av underklasser. Underklasserna kommer att beskriva hur programmet fungerar när det är i vissa lägen för att se vad som ska ritas, vad som ska beräknas samt vilka inputs man skall kolla.  States ska kunna skapas och kommer sedan pekas på med en statepekare för att välja det state som main-klassen kommer att hantera. States ska även ha möjligheten att ha en ljudfil som kommer att spelas upp i en loop när man är i det statet, ljudfilen kommer att spelas upp med hjälp av funktioner ifrån SFML biblioteket.

**2.4.1 Beskrivning av Menu**

En Menu efter initiering ska vara det State som ska visa upp en meny med ett antal delval och hålla koll på vad som sker när ett av delvalen väljs samt hålla koll på vilket val den har markerat och eventuell funktion som kommer med valet. Inputs som kommer att kollas kommer vara om användaren vill gå upp och ner i menyn eller att gå in i det val som är valt. Val som kan göras kommer bland annat vara: starta spelet, välj karta, välj spelarnamn, avsluta spelet.

**2.4.2** **Beskrivning av Game**

Ett Game efter initiering kommer vara det State som håller koll vad som sker under spelet, den kommer att kolla efter inputs som hanteras av spellogiken, utföra eventuella händelser i spellogiken som manipulerar spelets olika objekt och ritar ut alla definierade objekt på spelplanen. Game kommer även att hålla koll på vilken karta spelet ska spelas på i dess initering genom att hämta kartvalet ifrån data och sedan skapa den genom att sätta ut objekt på skärmen på förbestämda platser.

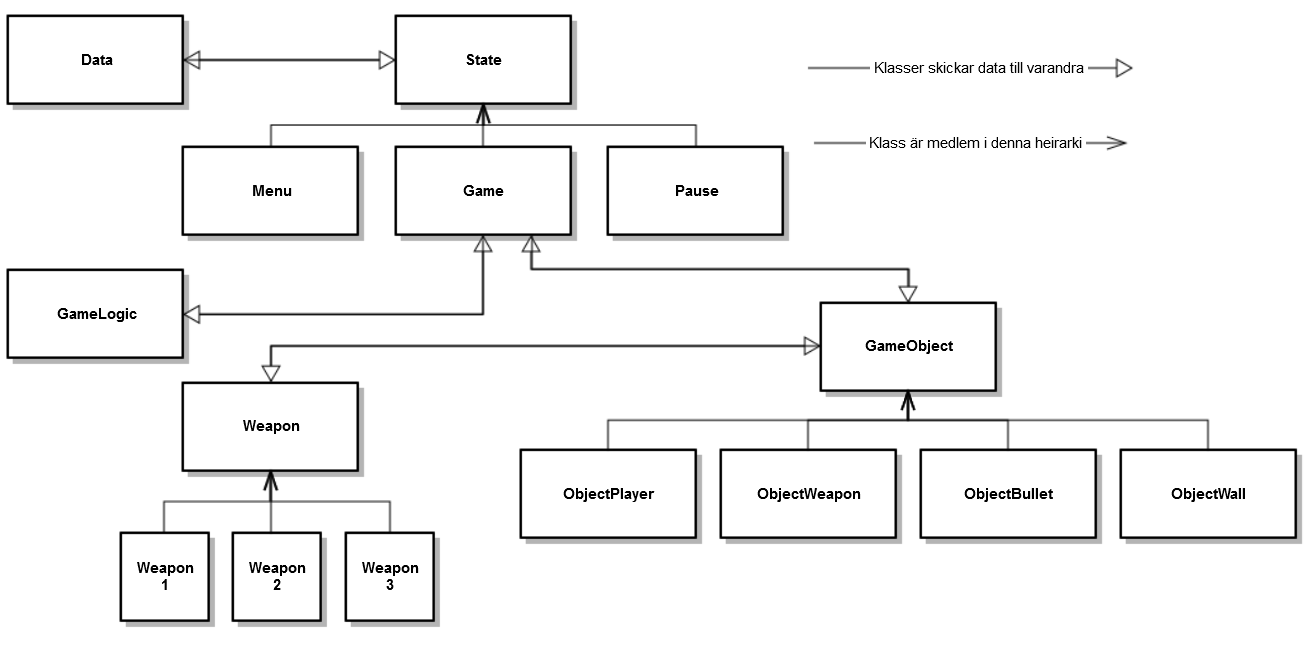
**2.4.3** **Beskrivning av Pause**

Pause kommer efter initiering att vara ett State som håller spelet Pausat och kommer bara att lyssna efter ett input för att unpausa och kommer att rita ut ett meddelande som visar att spelet är pausat. Pause kommer inte att hantera någon logik utom att vid input ändra tillbaka Statepekaren till Game. För att komma till Pause Statet så måste man trycka på ett specifikt input för pause som då ändrar statepekaren till Pause Statet och därmed ser till att Game inte kommer att göra något förrän spelaren lämnar pause.

**2.5 Beskrivning av Data**

Data kommer att vara en lagringsklass som kan kommunicera med alla underklasser till State som behöver skicka data mellan sig. Ett exempel är att vid start av spel så kommer Menu statet att skicka in information till Data som lagras, exempel på information är spelarnas namn samt val av karta. Denna information kan då Game ta ifrån vid initiering för att sätta spelarnas namn samt måla ut den karta som valts.

**3. Klassrelaktioner**

Bilden nedan beskriver hur våra klasser kommer att skicka information till varandra. Alla interaktioner som visas i bilden beskrivs nedan i textform.

**3.1 Interaktioner mellan olika GameObject klasser.**

Interaktionerna mellan det olika Gameobject klasserna kommer i grunden att bestå av att kolla om två objekt korsar varandra eller håller på att korsar med varandra. Klassen gamelogic kommer att ha funktioner som jämför gameobjects med varandra för att se om de korsar och om de gör det ska åtgärder tas beroende på vilka objekt det är.

**3.11 ObjectPlayer och ObjectWeapon**

När en ObjectPlayer och en ObjectWeapon korsar varandra ska efter input av användaren vapnet plockas upp av spelaren vilket innebär att spelarens Weapon sätts till det vapen som plockades upp och ObjectWeaponet ska tas bort ifrån kartan.

**3.12 ObjectPlayer och ObjectBullet**

När en ObjectPlayer och en ObjectBullet korsar varandra ska spelet kolla om ObjectBulleten tillhör spelaren. Om den inte gör det ska det ObjectPlayer som ObjectBulleten korsade ”ta skada” och ObjectBulleten ska sedan tas bort ifrån skärmen.

**3.13 ObjectPlayer och ObjectWall**

När en ObjectPlayer och en ObjectWall korsar varandra ska spelaren stoppas ifrån att röra sig som den håller på att kollidera ifrån och kommer inte kunna röra sig i den riktningen förens spelaren byter riktning och har förflyttat sig ifrån väggen.

**3.14 ObjectBullet och ObjectWall**

När en ObjectBullet och en ObjectWall korsar varandra så ska ObjectBulleten tas bort ifrån skärmen.

**3.2 ObjectPlayer och Weapon**

När en spelar har plockat upp ett ObjectWeapon så kommer informationen lagrad i ObjectWeapon att skickas till klassheirakin Weapon för att kolla igenom vilket vapen som ska skapas för spelaren. Efter det har skapats så kommer det vapnet att sättas till spelarens nuvarande vapen.

**3.3 Weapon och ObjectBullet**

När en spelare vill skjuta sitt vapen så kommer Weapon att skicka information till Bullets konstruktor. Den information den skickar (Size,Speed,Spread) är unikt för varje Weapon och den informationen kommer vara konstant tills spelaren byter vapen (Då får man in ny information) eller att spelaren får slut på ammuntion i vapnet (ingen bullet skapas).

**3.4 Menu och Data**

I menyn så matar man in spelarnas namn samt att man väljer den karta man ska spela på. Dessa val skickas sedan in till Data för att sparas där så att andra klasser kan hämta den informationen.

**3.5 Game och Data**

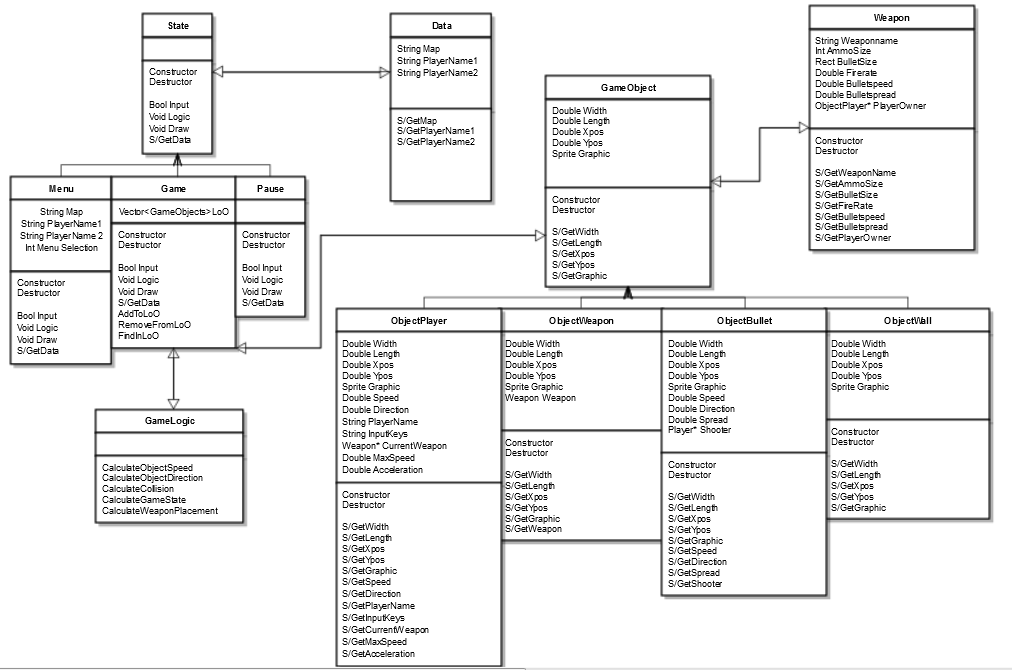
När ett Game initeras och förbereder spelet som ska starta så kommer Game att hämta information ifrån data för att kunna veta vilken karta som ska skapas samt vad de olika spelarna kommer att ha för namn.

**3.6 Game och Gamelogic**

Under spelets gång så kommer Game att kalla på funktioner i Gamelogic för att se att det som sker i spelet är tillåtet samt för att manipulera spelets objekt för att flytta runt på det som ska ritas ut.

**3.7 Specifikation av Protokoll**

Programmet kommer att kommunicera genom att skicka olika variabler mellan de olika klasserna enligt klassinteraktionerna i det programmet. State klasserna kommer att ta emot inputs från tangentbordet som programmet använder sig utav för att styra det som sker på skärmen.

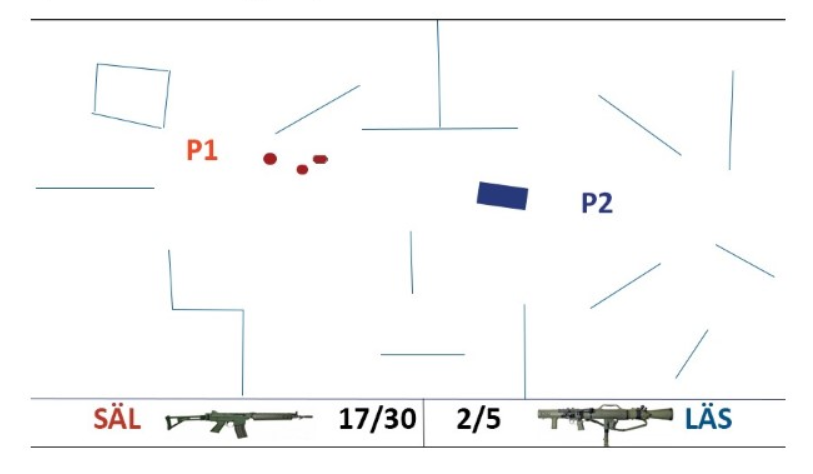
**4. Detaljerad Teknisk Specifikation**

Bilden ovan är en beskrivning av strukturen av programmets klasser som visar dess variabler och metoder som skall skapas. Pilarna i bilden visar hur klasserna kommer att kommunicera med varandra. S/G innebär Set and Get som finns till alla variabler som kan sättas om under programmets körtid. State, GameObject och Weapon beskriver klassheirakierna som nämndes under beskrivningen av klasserna och alla klasser under dom är delar av de klassheirakierna. Pilarna mellan klassheirakierna innebär att data kommer skickas mellan dom och kommer gå ner i de underklasser som skall ha datan.

**5. Design av användargränssnitt**

Huvudmenyn består utav tre alternativ vilka är: Starta Spelet, Inställningar och Stäng av spelet. Om val av Starta Spelet görs så kommer användarna till en ny meny där val av bana, antal spelare och val av spelarnamn skall kunna göras. Vid val av alternativ Inställningar så kommer användarna också till en ny meny där de ändra inställningar som finns i spelet t.ex. musik volym (av/på). När man är i menyn så kommer en ljudfil spelas upp i en loop.

Bilden ovan är ett exempel på hur menyerna kommer att se ut. Man kommer kunna se alla val som går att göra och till vänster så kan man se en pekare som pekare på det val som kommer utföras ifall man tycker på ”OK-knappen”. Det kommer även finnas knappar som tillåter användaren att gå upp och ner i menyn.

I Spelet kommer man då se den karta man valde samt alla spelarna i spelet, man kommer även på kartan se vapen som ligger utsprida som spelaren kan plocka upp. Under kartan finns en inforuta som visar spelarnas namn, vilket vapen dom har samt hur mycket ammunition som finns kvar i vapnet. Bilden nedan är en representation om hur spelet kan se ut (Dock ej representerad med ordentlig grafik.), Under spelet kommer även en ljudfil spelas upp i en loop under spelets gång och ljudeffekter kommer spelas upp när en spelare t.ex skjuter sitt vapen.

**6. Design av databas/filstrukturer**

Spelet kommer inte göra några externa lagringar som användarna behöver bry sig om. Spelets grafik och ljud kommer dock att lagras i samma mapp som de övriga filerna som spelet består utav om inte filerna kan sättas ihop till att bara vara ett körbart program.