TDDD78 projekt: Tower Defence

1 Introduktion

Tower Defence är en kategori av spel med rötter till 1980-talet som går ut på att försvara en punkt (ofta symboliserat som en bas eller by) från horder av monster som går genom en bana. Detta görs genom att bygga torn som skjuter ner dem. Likt många andra spelkategorier finns det olika tappningar men man kan göra en grov uppdelning i två kategorier:

- Monstren går i en smal bana och spelarna bygger torn utanför banan.
- Monstren går i en bredare bana (kan vara hela spelplanen) och spelarna bygger torn inuti banan. I de här versionerna brukar tornen blockera motståndarna och/eller skjuta på dem.

2 Bakgrund

Se Wikipedias artikel: https://en.wikipedia.org/wiki/Tower_defense.

3 Milstolpar

Nästan oavsett vilket typ av Tower Defence spel man vill bygga har det ett par gemensamma grunder. I figur 1 ser man grunden som är kartan i spelet. Därefter följer 6 milstolpar innan spelet är spelbart i dess simplaste form. När det väl är spelbart finns det väldigt många olika förbättringar som är mer eller mindre oberoende av varandra. Några av de förbättringarna syns i samma figur men ännu fler finns i figurena 2 och 3.

Kort förklarat går ut på att först få en bana färdig där monstren går i raka linjer mellan vissa fasta punkter i en bana. Till att börja med kan spelaren placera ut olika torn på kanterna av banan i ett förutbestämt rutnät. Efter det finns det många olika förbättrings att välja från. De innefattar allt från att skapa fler typer av monster och torn, highscore, stöd för fler spelare till ljud.

Det är tillåtet att använda sig av en modifierad version av de milstolparna som är presenterade. Gör i så fall ett förslag för hur ni vill att det ska se ut och fråga därefter er assistent innan ni börjar.

3.1 Milstolpe beskrivningar

För att ge en tydligare början beskrivs de första milstolparna mer ingående här:

- **Karta**: Den här milstolpen inbegriper att en karta som torn och monster kan placeras på skapas. Vidare inbegriper det även att man implementerar en vy för kartan.

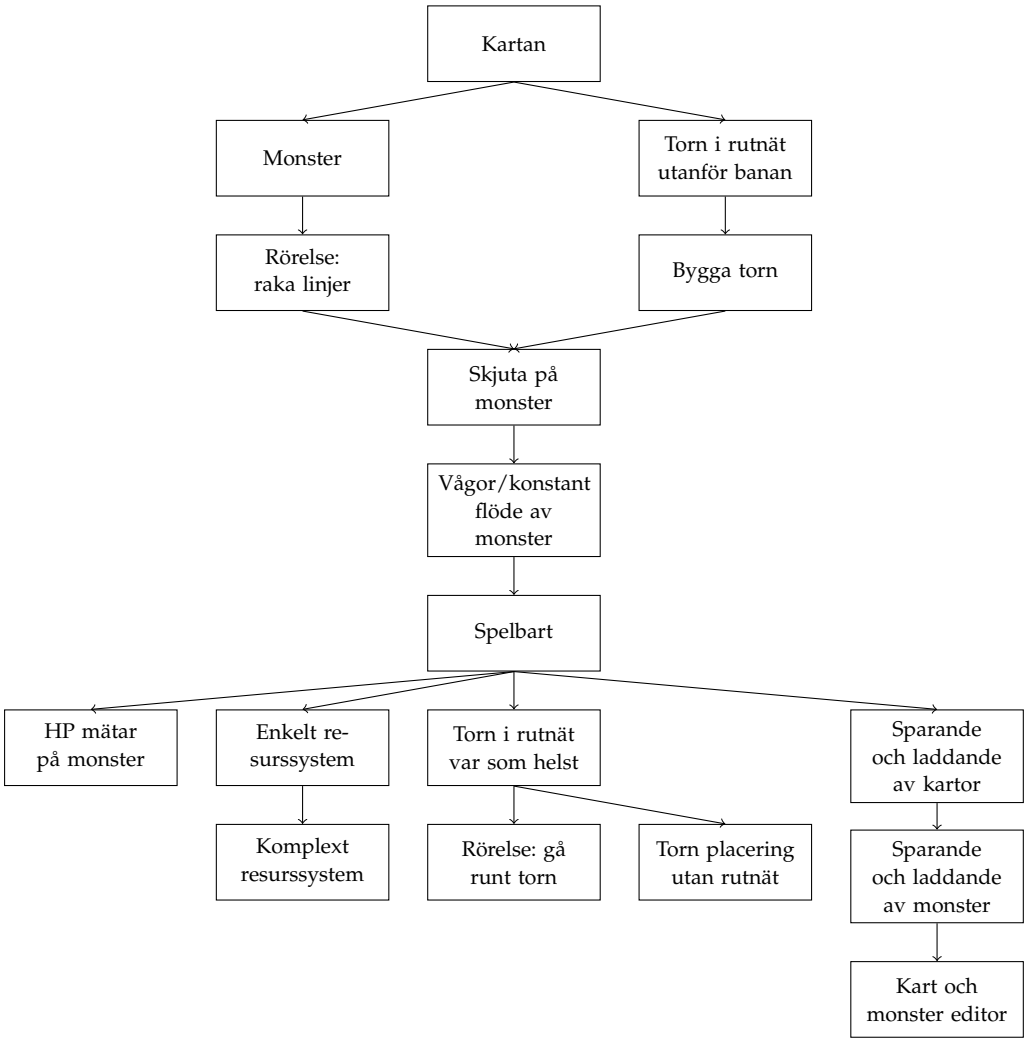

Fig. 1: De första milstolparna.

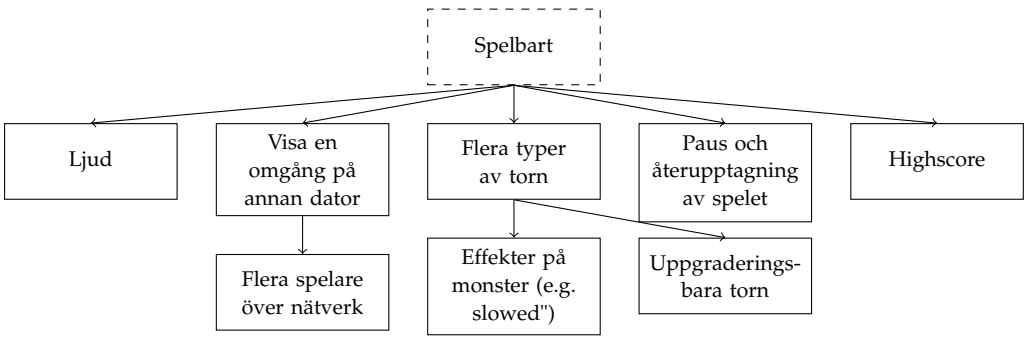

Fig. 2: Extra milstolpar som kräver att grunden är färdig (del 1).

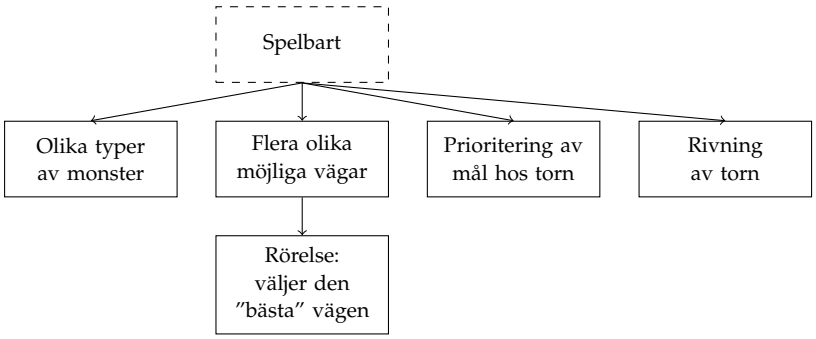

Fig. 3: Extra milstolpar som kräver att grunden är färdig (del 2).

- **Monster**: Milstolpen inbegriper att det finns minst en typ av fungerande monster implementerade. De ska kunna placeras på kartan och synas där.
- **Rörelse: raka linjer**: Här förväntas man få monstren att röra sig i raka linjer, lämpligen mellan olika checkpoints på kartan.
- **Torn i rutnät utanför banan**: Den här milstolpen inbegriper att det finns minst en typ av torn som kan placeras på banan. Notera att den enbart ska kunna placeras på giltiga platser. Det är inte inbegripet i milstolpen att tornen ska kunna placeras i de områden som monstren rör sig i.
- **Bygga torn**: Milstolpen innebär att det går att bygga torn på banan under spelets gång. Milstolpen inbegriper inte att det ska kosta att bygga torn.
- **Skjuta på monster**: Målet med milstolpen är att få tornen som är tillräckligt nära minst ett monster att skjuta på det/dem.
- **Vågor/konstant flöde av monster**: Den här milstolpen inbegriper att det kommer fler vågor av monster alternativt ett konstant flöde av monster som försöker ta sig till slutet av banana.
- **Spelbart**: Den sista förklarade milstolpen är lite av en uppsamling. Beroende på hur mycket som har gjorts i de tidigare milstolparna kan det finns saker som saknas för att det ska gå att spela spelet i det mest grundläggande form. Den här milstolpen inbegriper att de sakerna implementeras.
- **HP mätare på monster**: Ha en mätare (till exempel en avlång rektangel) intill varje monster som visar hur mycket liv monstret har kvar.
- **Enkelt resurssystem**: Upp till nu har det varit gratis att köpa torn eftersom det inte finns något att köpa tornen för. Den här milstolpen syftar till att introducera ett enkelt resurssystem. Det vill säga en resurs (till exempel guld, ved eller sten) som spelaren får (till exempel till att börja med, periodiskt eller när man dödar monster). Resurserna ackumuleras och används sedan för att bygga byggnader.
- **Komplext resurssystem**: Den här milstolpen syftar till att utöka det existerande resurssystemet. Det kan till exempel vara med olika typer av resurser som är

olika sällsynta, krav som att spelaren bara får ha x torn och torn som producerar vissa resurser.

- **Torn i rutnät var som helst**: I tidigare tillät man enbart tornen byggas utanför området där monstren går. I den här milstolpen utökas det så att torn kan byggas även där monstren går. Tills vidare räcker det med att monstren går rakt igenom de tornen som byggs i vägen.

- **Rörelse: gå runt torn**: Den här milstolpen innebär att man måste implementera en typ av vägafinnande (en. path finding). Det vill säga monstren kan nu finna vägen förbi torn som har byggts i banan.

- **Torn placering utan rutnät**: I den här milstolpen ska man se till att torn kan placeras var som helst (där det finns plats). Det vill säga, tornens placering är inte längre begränsade av ett rutnät.

- **Sparande och laddande av kartor**: Den här milstolpen är en ganska stor milstolpe. Som namnet förtäljer går det ut på att man ska kunna ladda och spara kartor, inte under spelets gång men så spelaren kan välja från flera olika kartor samt skicka kartorna som separata filer till andra spelare. Det finns två huvudsakliga sätt att göra det på. Det första är att använda sig av Javas Serializable interface för att spara och ladda kartor direkt med klassen. Det andra alternativet är att spara banan strukturerad (till exempel databaser) eller semistrukturerad (till exempel xml och JSON) data. Det senare har fördelen att man redan i det läget kan skapa ny banor enkelt genom att skapa en ny fil som har samma format. Dock så kräver det förmodligen lite mer arbete.

- **Sparande och laddande av monster**: Exakt som sparande och laddande av kartor fast för monster.

- **Kart och monster editor**: Den här milstolpen är antingen ett separat program eller en del av originalprogrammet. Målet med milstolpen är att man ska kunna, via ett grafiskt interface eller kommando raden, kunna skapa och modifiera kartor och monster.

- **Ljud**: Upp till det här stadiet har spelet varit helt tyst. Det är något som är väldigt ovanligt. Hur ljudet ni introducerar låter är helt upp till er men två exempel är skjutljud för torn och bakgrundsmusik.

- **Visa omgången på annan dator**: Det här är ett första steg mot att kunna spela flera spelare på samma karta över ett nätverk. Målet med milstolpen är att man ska kunna skapa ett spel på en dator och att med samma program på en annan dator koppla upp sig mot spelet och se på när den andra spelar. Det här är lite skilt från att ”streama” eftersom man inte skickar en film.

- **Flera spelare över nätverk**: Målet här är att man inte enbart visar vad den andra spelare gör utan att man själv kan bygga torn som anfaller monstren och så

vidare. Det kräver att alla som är med i spelet skickar data till varandra för att beskriva vad spelarna gör.

- **Flera typer av torn**: Torn har hitintills enbart varit av en typ. Nu är målet att man ska kunna bygga flera olika typer av torn. Till exempel kan det finnas torn som skjuter på alla monster runt omkring dem, kan skjuta på fler monster samtidigt och som skjuter på hela området runt dem.

- **Effekter på monster**: Ge torn en möjlighet att skapa effekter på monstren. Till exempel kan monstren bli bromsade så de inte går lika fort, fastlåsta så de inte går alls skadade över tid.

- **Uppgraderingsbara torn**: Introducera olika nivåer på torn. Det vill säga tornen ska kunna uppgraderas så att de, till exempel, gör mer skada, når längre eller applicerar fler effekter på monstren.

- **Paus och återupptagning av spelet**: Ge spelaren möjligheten att ta en paus i spelet för att sedan återuppta det.

- **Highscore**: Utvärdera hur väl spelaren har klarat av sitt spel. Det kan till exempel vara genom hur många vågor av monster hen har klarat eller hur mycket resurser hen har samlat ihop.

- **Olika typer av monster**: Fram till den här milstolpen har enbart en typ av monster funnits. Nu är det dags att introducera flera typer av monster. Till exempel kan det vara monster som flyger, kan gå igenom torn eller är odödliga under vissa perioder.

- **Flera olika möjliga vägar**: Låt kartorna inkludera flera olika möjliga vägar som monstren kan gå. Det kan betyda att hälften av monstren går i en riktning och den handra hälften i en annan riktning eller att alla väljer samma riktning.

- **Rörelse: Väljer den ”bästa” vägen**: Nu ska monstren välja den bästa möjliga vägen. Hur den bästa vägen bestäms är upp till er (kanske vill man basera det på hur många torn som finns runtomkring dem) men ett vanligt sätt är att låta den bästa vägen vara samma som den kortaste.

- **Prioritering av mål hos torn**: Tidigare har prioriteringen av vilket monster som tornen skjuter på varit godtycklig. Låt nu lika torn ha olika prioriteringar (det monster som är först, snabbast, minst hp och så vidare) alternativt låt spelaren välja prioritering för varje torn.

- **Rivning av torn**: Möjligheten att göra om saker om man gör fel är oftast väldigt uppskattat i spel. Av den anledning är det även bra om man har möjligheten att riva ett torn som man tidigare har byggt. Om man vill och har implementerat ett resurssystem är det möjligt att ha möjligheten att sälja istället för att riva byggda torn.

3.2 Designbeslut att tänka på

I ett Tower Defence projekt bör ni tänka en extra gång på hur monstren rör sig längs banan. Rör de sig mellan rutor eller rör de sig längs x och y axlarna oberoende av ett rutnät? Det här är ett designbeslut som kan vara svårt att ändra under projektets gång så fundera på det i förväg.

3.3 Exempel på projektgenomförande

Ett exempel av en log (i form av interna anteckningar) av ett projektgenomförande av Tower Defence projektet kan vara följande:

- Beslut om kartrepresentation: Använd x och y koordinater för att placera saker på kartan.
- Implementation av milstolpe Karta (enligt model view controller design mönstret):
  - Skapade en modell över kartan i form av en klass. Klassen innehåller en kollektion av monster (ett nytt skapat *interface*), en kollektion av torn (ett nytt skapat *interface*), en kollektion av områden (ett nytt skapat *interface*) som kan vara de olika typer av områden på kartan (byggbara, möjliga för monster att gå på, etc.), storleken på banan samt en linjär sekvens av checkpoints som monstren måste ta sig igenom (första är där monster kommer in och sista är där de går ut). Klassen har även en uppdaterings metod som ser till att alla monster och torn går ett tidssteg.
  - Skapade två implementationer för områden: Ett område som kan bebyggas men monster kan inte gå på dem samt ett område som kan bebyggas och som monster kan gå på.
  - Skapade en vy för en kartklass som skapar ett fönster för spelet och ritar ut kartan (och dess komponenter) i hela skärmen.
  - Skapade en statisk klass (en kartgenerator enligt factory method design mönstret) som genererar en exempelkarta med tre olika checkpoints i formen av ett L. Området som monstren kan gå i är trångt och kan inte bebyggas. Runtomkring är det möjligt att bygga torn men inte för monstren att gå.
  - Skapade huvudklassen i programmet. Låt den generera kartan och rita ut den.
  - Verifierade att kartan ritades ut.
- Implementation av milstolpe Monster:
  - Refaktorerad monster interfacet så att det blev en abstrakt klass. Den lagrar nu generell information så som x och y koordinater samt hur mycket liv den har (hp).

  - Implementerade en ”Grunt” Monster model (klass som ärver från den abstrakta klassen för monster) som enbart ska kunna springa längs marken.
  - Skapade en vy för monster, en abstrakt klass, samt en konkret klass för ”Grunt” monstren.
  - Utökade kartgeneratorn så att en ”Grunt” finns på den första checkpointen.
  - Förändrade huvudklassen så att den nu använder den nya kartan.
  - Verifierade att monstret skapades och ritades ut.

- Implementation av milstolpe Rörelse: raka linjer:
  - Utökade den abstrakta klassen för monster så att den nu har en metod för att flytta monstret samt med en hastighets variabel.
  - Utökade klassen med ett index för vilket mål den ska gå till samt metoder för att sätta och hämta variabeln.
  - La även till en metod i den abstrakta klassen som beräknar den nya positionen givet hur lång tid som har gått och den nästa målpositionen. Metoden antar att monstret går att gå rakt fram.
  - Utökade uppdateringsmetoden i modellen för kartan så att den anropar en uppdatering av positionen av monstren genom att beräkna hur lång tid som har gått sedan senaste anropet (krävde att en extra variabel användes i klassen). Den testar även om monstret har kommit fram till dess nuvarande mål och om så är fallet uppdaterar den indexet till nästa mål.
  - Testade så att monstren rör sig längs kartan.

- Implementation av milstolpe Torn i rutnät utanför banan:
  - Tog ett beslut om att alla kartor kan ha olika stora rutor och att det specificeras i klassen hur stor rutan är (alltid kvadratisk).
  - Uppdaterade kartklassen så att rutnätet ritas ut men enbart de rutor som är helt inne i områden som är byggbara.
  - Refaktorerade interfacet för torn så att det är en abstrakt klass. Den innehåller nu information om vilken x och y koordinat den börjar byggas på (koordinaten längst upp till vänster) samt hur många rutor den tar upp i x och y led.
  - Implementerade ett torn ”Arrow” som en klass, ärver den abstrakta klassen för torn.
  - Lade till ett torn i kartan som har använts för testning och såg till att den faktiskt ritades ut.

- Implementation av milstolpe Bygga torn:
  - Gjorde vyn för kartan till en MouseListener. Den kan nu identifiera om en punkt är i en ruta som är helt inom byggbara områden.

  - Gjorde så att det byggs ett ”Arrow” torn i de byggbara rutorna som det klickas på om det inte redan finns ett torn där.

- Implementation av milstolpe Skjuta på monster:
  - Uppdaterade uppdateringsmetoden i kartklassen så att den går igenom alla torn och skickar in listan av monster så tornet kan skjuta på dem om det kan.
  - Implementerade funktionen för att skjuta på monster i klassen för ”Arrow” torn.
  - Testade så att tornet skjuter på monster när de går förbi och att det skjuter lagom ofta.

- Implementation av milstolpe Vågor av monster:
  - Utökade kartklassen så att den nu innehåller en kollektion av events (ytterligare ett nytt *interface*) vars uppgift är att till exempel skapa monster efter en viss tid.
  - Skapade ett generator event (implementation av event interfacet) som under ett par tidsenheter skapar nya monster i startpunkten på kartan när en trigger (ytterligare ett nytt *interface*) sker.
  - Implementerade ett fabrik (factory method design mönstret) för att skapa nya event. Den första metoden tar en trigger och konstruktorn för monstret (använder här java reflections, dependency injection design mönster, men övervägde att helt enkelt ge en klass som genererar rätt typ av monster med metoden *createMonster*).
  - Utökade kartgeneratorn så att tre vågor av monster av ”Grunt”s (ett monster per sekund i 2, 5 och 10 sekunder). Först vågen har en trigger som aktiveras efter 10 sekunder. De följande aktiveras när alla monster på kartan är döda eller har nått sin checkpoint.
  - Utökade uppdateringsmetoden i kartan så att event testas korrekt och i rätt ordning.