Dokumentation slutprojekt

1. Veckoredovisning

.**Vecka**	Status
43	Skapade fysik motorn. En kub som kan röra sig och som kan känna av en kollision med andra kuber. Vid kollision så skrivs det i konsolen. En projektil som kan flyga över skärmen.
45	Skapade en klass för spelare och fiende och lade till animationer till de båda. Skapade en klass för skott och lagt till en funktion för att kunna skjuta på fienden.
46	Skapat en config-fil där man kan ändra stats för fienden, spelaren och spelet.
47	Skapat en klass för menyer och börjat bygga en startmeny, uppgraderings meny och meny för när man dör. Lagt till flera fiender, en som kan skjuta, en som är snabb och en som är stark.
48	Configurerat fiender, spelaren och uppgraderingarna samt lagt till ljudeffekter och samlat in feedback.

2. Utfört arbete

2.1. Fysikmotorn

Jag började med att skapa fysikmotorn. Fysikmotorn är ganska simpel och tillhandahåller kollision och projektiler. Projektilerna tar musens position och en startposition. Sedan beräknar den förhållandet mellan x och y positionen och multiplicerar det med hastigheten för att den ska flyga i rätt riktning. Kollisionerna kollar varje hörn på objekt ett mot varje hörn på objekt 2 för att kolla om de överlappar och kolliderar. Alla mina hitboxes i spelet är fyrkantiga för att skapa färre kalkylationer och därför optimera spelet. Ett undantag är granatens explosion som kollar om kollisionerna i en cirkel runt centrum för en bättre upplevelse.

2.2. Spelare och Fienden

Nästa steg var att skapa spelaren och fienderna. Kortfattat innehåller spelarnas klass följande: Liv, Hastighet, Skada, Position och Animation. Livet eller HP avgör hur mycket skada spelaren kan ta innan den dör. När spelaren tar skada beräknas skadan spelaren tar på fiendens skada och en resistans som spelaren har. Resistansen går att uppgradera senare. Hastigheten är hur många pixlar spelaren rör sig per tick. Skadan spelaren gör är hur mycket skada varje skott gör. Positionen består av x, y bredd och höjd. När spelaren går så ändras först en diffx och diffy. Detta är för att lätt kunna hålla ordning på om spelaren är i rörelse vilket underlättar animeringen. Nästa är

animeringen, animerings hastigheten och hur många bilder per animation. Fienden bygger på samma princip som spelaren, men den har färre animationer eftersom fienden alltid är i rörelse. Det som skiljer spelaren och fienden är att spelaren har en funktion för att skapa projektiler som kan skada fienden. Fienden har tre subklasser med olika stats för att det ska finnas olika fiender. De olika fienderna skapas en bit in i spelet för att göra det gradvis svårare.

2.3. Config

Nästa steg vara att skapa en config-fil. Detta för att lätt kunna ändra alla stats på spelaren, spelet och alla fienderna utan att behöva hoppa mellan massa filer. Det som inte finns med i config-filen är animationerna eftersom att det är smidigare att ha dem separat i varje klass istället.

2.4. Menyer

Därefter skapade jag en template för menyn. Templaten består av en bakgrund och knappar som man kan bygga ihop själv. Varje separat meny är en subklass till meny-klassen. Jag skapade även en ny klass för knappar som inte hör samman med en meny såsom startknappen.

2.5. Slutgiltig konfiguration och ljudeffekter

Det sista steget var att ställa in alla stats så spelet blir någorlunda balanserat. Det mest optimala är att göra en fältundersökning för att få feedback, vilket jag gjort, men inte tillräckligt storskalig. Tack vare feedbacken lade jag till ljudeffekter, granater och power ups.

3. Utvärdering av din projektplanering

Min planering fungerade ganska bra. Det första steget var att skapa fysik motorn, vilket är kärnan till spelet. Därefter skapade jag en spelare och en fiende. Detta var för att kunna testa fysikmotorn och kollisionerna. Eftersom att standard fienden är väldigt simpel. Den ska bara följa spelaren så skrev jag klart koden till den innan spelaren. Därefter lade jag till funktionen för att kunna skjuta eftersom att det också är en viktig del av testandet till fysikmotorn. När det var klart lade jag till animationerna för fienden och spelaren. Detta eftersom det satte prägel på hur resten av spelet byggs upp.

När alla funktioner för spelet var tillagda skapade jag en config-fil där man kan ändra alla inställningar för klasserna. Därefter skapade jag en klass för en meny och subklasser till en huvudmeny, uppgraderingsmeny och meny till när man dör. Lade även till funktioner för att kunna starta och stanna spelet efter varje runda. Det sista var att konfigurera alla fiender, spelare och uppgraderingar.

I början följde jag planering, sen flyttade jag fram skapandet av fiender och animationer eftersom att det är en viktig del av spelmotorn och kärnan av spelet. Det är viktigt att dessa funktioner fungerar, annars skulle inte resten av spelet fungera.

4. Testning

Jag har testat spelet själv och på andra. Jag har valt att inte validera spelet i en HTML validator eftersom att spelet består av en rad HTML. Jag har själv testat koden när jag har bestämt alla stats för fienden och spelaren samt uppgraderingsmenyn. Jag har testat spelet på andra personer och då har jag fått följande feedback:

Person (1)
Lägga till granater och någon form av energidryck som gör spelaren snabbare eller starkare

Lägga till ljudeffekter och bakgrundsmusik för att liva upp spelet lite grann.

Person (2)
Göra så att fienden inte kan skapas i spelaren

Highscore eller någon form av global poängräkning.

Granater och energidryck tyckte jag var en bra idé eftersom att det är något som kan hjälpa en ur kniviga situationer. Därför valde jag att lägga till båda funktionerna. Samma med ljudeffekter. Lade även till lite bakgrundsmusik för att liva upp stämningen. Highscore är inget jag kan göra i den här kursen, men det är något jag kan lägga till i nästa kurs vilket är planen för tillfället. Att fienden inte ska kunna skapas i spelaren är något jag överväger att lägga till. Mycket av spelet är slumpat, vart fienden skapas och hur många av varje sorts fiende som skapas. Genom att låta fiender skapas i spelaren så blir spelet mer om tur än om skicklighet. Jag ser både för- och nackdelar med ett turbaserat spel. För att få ett svar vad andra tycker blir det till att göra en undersökning och utgå från det. Jag planerar att göra det i samband med high-score systemet eftersom att jag då måste ha ett väl fungerande system där alla ska ha samma förutsättningar.

5. Upphovsrätt och GDPR

Bilderna jag använder mig av kommer från itch.io som är en webbsida för bland annat tillbehör som ljud och grafik till spel. Jag har valt att använda gratis bilder som får användas i kommersiellt bruk. De generella kraven från skaparna är att man lämnar cred till dem, vilket jag har gjort. Från spelet finns en länk till github med credits. Musiken är tagen från samma källa och ljudeffekterna är från Minecraft. Minecraft tillåter en att använda ljud så länge man lämnar credits.

6. Koden

Nästan hela webbsidan bygger på grafik elementet canvas. Canvas initialiseras i HTML och sen skrivs resten av koden i Javascript. CSS:n som används är bara till för att canvas elementet ska täcka skärmen. Spelet bygger på en loop som kör alla funktioner. Hastigheten på loopen bestämmer hastigheten i spelet eller FPS. Antalet FPS bestäms genom .now() plus 1000/60 millisekunder. När det gått 1000/60 millisekunder från senaste framen så går spelet till nästa iteration i loopen. Detta borde i teorin vid optimala förhållanden resultera att spelet körs i 60 fps.

6.1. object_list

object_list är en lista med alla instanser i spelet. Spelaren, fienderna, menyerna, granaterna och energidrycken. Alla instanser har en update() function som körs från object_list en gång varje iteration av spel-loopen.

6.2. update()

update() uppdaterar instansen. För spelaren uppdaterar update() positionen, kollisionen, HP och Healthbaren. Detsamma gäller för fienderna. För menyerna uppdaterar update() om storleken och om menyn är synlig eller ej. Principen sträcker sig till alla instanser i object_list

6.3. collides()

collides() kollar kollisionen mellan två objekt o1 och o2. collides() kollar om o2.x + o2.width är större än o1.x och o2.x är mindre än o1.x + 01.width. collides() används för att kolla alla alla kollisioner i spelet.

6.4. keysPressed

keysPressed är en lista där alla knapptryck lagras. Det finns två eventlistners, en för keyDown och en för keyUp. Vid keyDown så sparas tangenten i listan och vid keyUp tas tangenten bort. I spel-loopen kollas det vilka tangeter som finns i listan och därifrån körs funktioner.

6.5. stats.js / stats

stats.js är config filen för alla stats för spelaren, fienderna och spelet. Ursprungsplanen för stats.js var att det skulle vara en json fil, men eftersom att det är komplicerat att importera json till Javascript så skapade jag ett dictionarie i stats.js som heter stats. Under spelets gång så ändras värdena i stats.js eftersom att nya fiender och spelaren får sin information och HP, skada och hastighet från stats.

6.6. Menu

Menu är en template för alla menyer. Menu har en bakgrund som kan ändras i storlek och position. Menu har även knappar som kan ändras i storlek och position. Till knapparna kan man lägga till en onClick() funktion som körs varje gång knappen trycks. Detta fungerar genom att en eventlistener för mouseclicks går igenom listans knappar och kollar om dom är synliga och sedan om inBounds() är true. inBounds kollar om musens koordinater är innanför objekts x, y, width och heigth.

6.7. UpgradeMenu

UpgradeMenu är en subklass till Menu och använder sig av Menu templaten. UpgradeMenu har fyra knappar och varje knapp har ett pris lagrat till sig. onClick() kollar om priset är mindre än spelarens pengar, isåfall ökar den valda variabel med vald multiplier från stats samtidigt som priset ökar med 10 och pengarna reduceras med summan av föregående pris.

6.8. restartGame()

restartGame() återställer stats till en kopia av stats som skapades vid starten av spelet. restartGame() återställer även object_list, keysPressed, spelaren och UpgradeMenu. UpgradeMenu måste återställas eftersom att priset för varje uppgradering lagras i knappen istället för stats filen.

6.9. Animationer

Animationerna bygger på att varje animerat objekt byter bild vid ett intervall. För spelaren, fienderna och mynten så består varje bild av fyra bilder. Efter varje 5 ticks eller 60/5 sekunder byts bilden till nästa och objektet upplevs vara animerat. Bilden byts genom att koordinaterna av den visade bilden flyttas. Explosionen till granaten innehåller fler frames så därför är varje frame en separat bild. Istället för att ändra koordinaterna i bilden som visas ändras istället filvägen till nästa bild.

6.10. spawnEnemy()

spawnEnemy() skapar en ny fiende vid vald koordinat. spawnEnemy() slumpar ett värde mellan 0 till hundra, beroende på vilket värde som slumpas fram så kommer olika fiender skapas. I stats.js kan man välja hur stor procentuell chans det är för en viss typ av fiende att skapas och det kalkyleras genom följande. Om random är mindre än F1.spawnChanse skapas en F1. Om random är mer än F1.spawnChanse men mindre än F2.spawnChanse skapas F2. Annars skapas F3.

7. Eventuella guider eller tutorials

Den guide jag använde mig av var W3schools exempel på Canvas kod. Detta eftersom att canvas är ett nytt sätt att skapa grafik på för mig, men jag har jobbat med liknande spelgrafik tidigare och principen är densamma. Jag har även fått hjälp från en klasskamrat Simon Fikse med gameloopen och principen bakom animationerna eftersom han också bygger ett spel.

8. Betyg

Jag anser mig uppnå ett B eftersom att mitt spel är välutvecklat och följer en någorlunda industristandard. Dock faller min dokumentation och planering vilket gör att jag endast landar på ett B istället för ett A. Förvisso så har min kod flera utvecklingsområden, men jag tycker att spelet ger ett gott helhetsintryck. Koden är väl strukturerad, men det finns förbättringspotential med namngivandet. Stats.js är mer som en config-fil eller en datafil. All namngivning är inte heller enhetlig där en del funktioner skrivs med _ istället för stor bokstav. Man kan även tycka att det borde finnas fler funktioner, men då är frågan vart man ska dra en gräns? Olika vapen? Multiplayer? En öppen värld där man kan bygga saker? I slutändan, med den givna tidsramen är jag nöjd med projektet.