Online Mini-Golf

**Medlemmar:**

Daniel Åslund

~~John Westerberg~~

Max Gimnell

**Ansvarsområden:**

Bestämdes aldrig på förhand.

**Syfte:**

Ett browser baserat mini golf spel, där målet är att “putta” ett objekt till en specifik punkt I banan som motsvarar hålet.

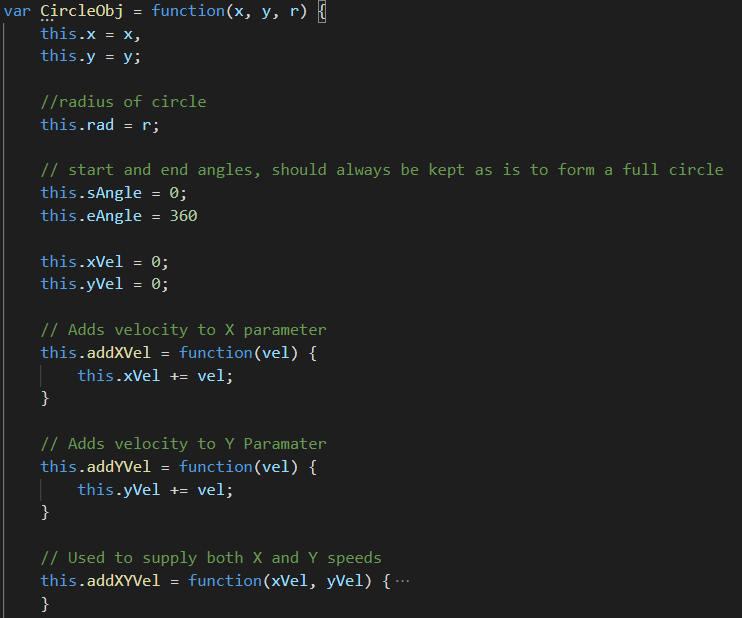
**GitHub**: <https://github.com/fabbrodan/GolfBrowserGame>

**Klasser:**

CircleObj:

Används för både bollen samt hålet.

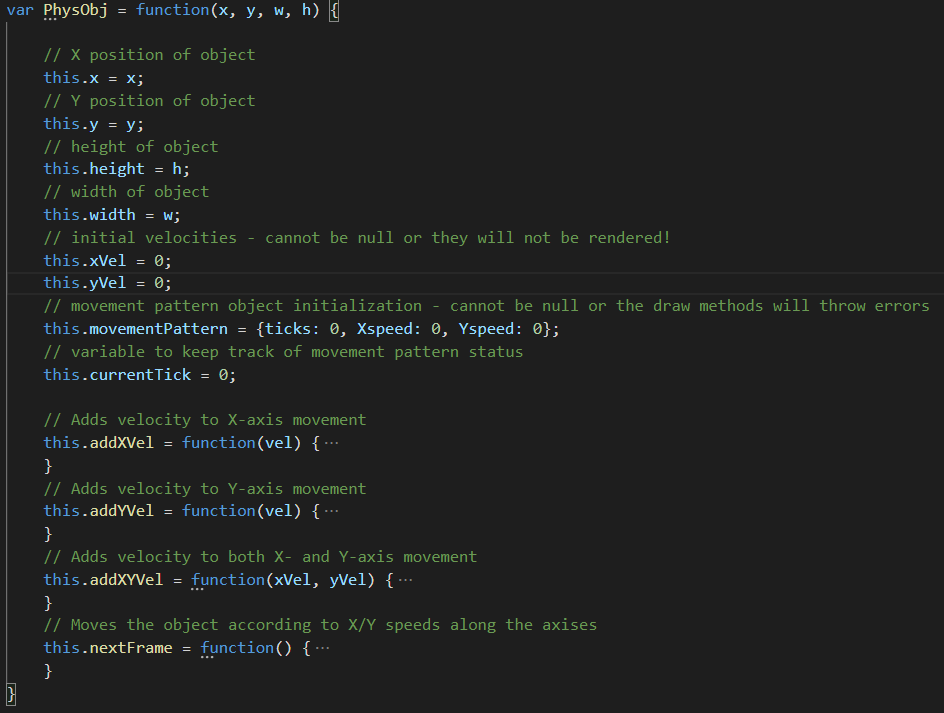
* Består av X och Y värden för att motsvara positionen.
* X-hastighet och Y-hastighet för att transformera X och Y positionerna genom inbyggd metod.
* Radie för att definiera storleken.
* Start- och Slutvinkel. Används endast I samband med Canvas och dess arc funktion för att rita upp det som en cirkel.
* Funktioner för att applicera både X och Y hastigheter.
* NextFrame funktion som itereras för varje bildruta som renderas, I vilken vi kollar efter kollision mot övriga objekt på banan, samt för att avgöra om vi har träffat hålet.



PhysObj:

Detta är den typ av solida objekt som även har möjligheten för rörelse.

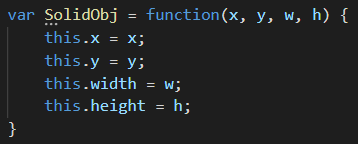
* X, Y, höjd och bredd för att definiera objekets position samt storlek.
* X och Y hastigheter, för att transformera positionen av objektet
* Array för rörelsemönstret, under hur många “ticks” rörelse sker, samt hur mycket X samt Y hastighet som skall appliceras
* Tick värde, för att hålla koll på rörelsens nuvarande läge
* Funktioner för att applicera X samt Y hastighet
* NextFrame funktion som kallas vid varje bildruta där vi beräknar ny X samt Y position baserat på nuvarande tick I rörelsemönstret samt den angivna X/Y hastigheten från rörelsemönstret.



SolidObj:

Solida objekt som är statiska, så som väggar och icke-rörliga hinder.

* X, Y, höjd och bredd attribut för att avgöra objektets position samt storlek.

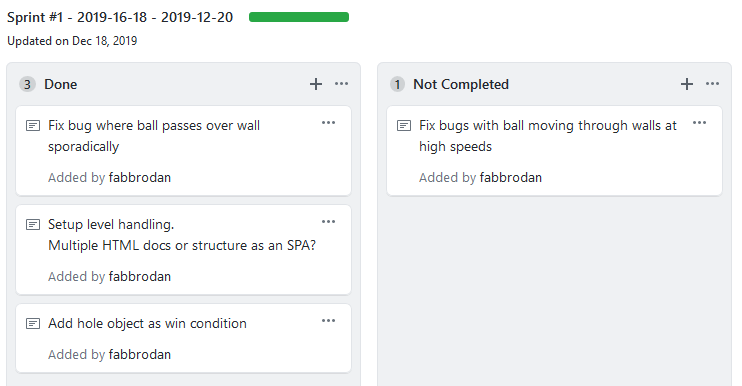


För highscore implementation användes Google’s Firebase lösning för att snabbt spinna upp en NoSQL databas och lagra resultat I molnet.

**Backlog** (Finns även tillgänglig som projekt på GitHub):

Sprint 1:

* Bygga de nödvändiga objekten
* Rita upp ett första utkast för en bana
* Koliisionshantering



Det började med skapandet av en enkel fysikmotor för att kunna hantera rörelse, friktion samt kollision.

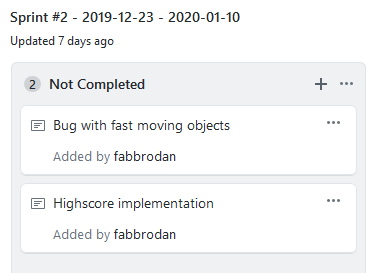
I fysik motorn byggdes de olika objekten in som skulle användas på nivåerna, och de fick funktioner för att simulera ett slag.

Sedan byggdes det en abstrahering av skapandet av banor. Så istället för att behöva rita varje bana enskilt, så byggdes det en separat klass som bara behövde ha en agnostisk Draw funktion samt Canvas Context initiering, denna kallades då via “huvud” scriptet där vi också definerade vilken nivå spelare är på, baserat på detta initerar vi banan, lägger alla nödvändiga objekt I en global array, som Draw funktionen sedan enkelt kan iterera över och rita upp objekten. Detta ledde till att det inte var några större problem att utöka spelet med fler nivåer.

I denna klass kollade vi även efter kollision med hålet för att avgöra om spelaren klarat nivån, och I sådana fall avancera spelet vidare.

Sprint 2:

* Användar interaktion
* Highscore lista – Implementera firebase



Här byggdes det in funktionalitet för musinteraktion.

Detta gjordes genom att på canvasen kolla efter mousedown event, lagra de dåvarande X- och Y-koordinaterna för muspekaren, och sedan registrera de X- och Y-koordinaterna vid mouseup eventet.

Och med de två seten, avgöra om X/Y var större eller mindre vid släpp jämfört med tryck, och sedan applicera följande formel för att ge bollen hastighet:

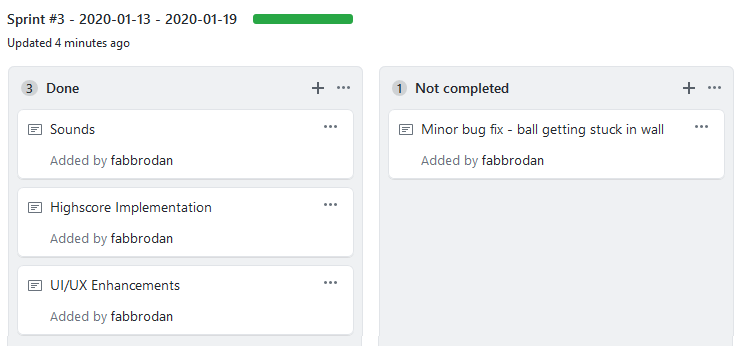
Om (Största X/Y Värdet – Minsta X/Y Värdet) / 3 > 15 sätt hastigheten på X/Y till 15, annars sätt det till (Största X/Y värdet – Minsta X/Y värdet) \* 0.25.

Sedan kollar vi om X/Y var störst vid klick kontra släpp, och sätter det till resultatet av ovan formel antingen till plus eller minus det värdet.

Tyvärr fanns det inte tid att implementera firebase som det var skrivit I backloggen, då parallelt med ovan implementation jobbades det med buggen med snabbt rörliga objekt, nödlösningen var att limitera hastigheten till 15. Samt julledighet.

Sprint 3:

* Design
* Assets
* Musik/Ljud



I sista sprinten implementerades FireBase funktionalitet för att lagra highscore I molnet, samt hämta ned datan och visa den rangordnad.

Vi implementerade också Bootstrap för lättare grid-view baserad design.

Så I stort sett allt designarbete skedde under denna sprint.

Detta ledde dock till att ny funktionalitet krävdes för uppritandet av nivåerna samt användarinteraktionen.

I och med att Bootstrap har fördefinerade “gutters” samt padding/margin på sina bas klasser “container” och “row”, så blev X/Y detektionen vid klickandet off, så vi fick använda oss av offset värden för att korrekt kalkylera vart muspekaren befann sig. Detta ledde dock till bonusutvecklingen att man nu från koden kan ange en offset till nivån så att det är lätt att förskjuta den på canvas elementet.

Hämtade ned gratis open source ljud för att ytterligare berika spelupplevelsen.

Implementerade även en omstartsfunktionalitet.

**Planerade utvecklingar:**

Bättre kollisionsdetektering:

Detta kunde ha uppnåtts genom att vid “slag” tillfället på bollen, kakylera alla de möjliga punkterna som bollen skulle skära baserat på den givna X/Y hastigheten, sedan iterera över dessa och se om de skulle skära med något av objekten.

Responsivitet:

I dagsläget är sidan inte alls responsiv, och de få element som är det är inte särskilt snyggt responsiva.

Om man ändrar storlek på sidan så behöver man ladda om den för att X/Y kalkyleringen skall ske korrekt.

Dessutom så är storleken på alla objekt hårdkodade istället för att relativa till Canvas/Viewport.

Ytterligare så finns det inte heller support för touchskärmar, då dessa nyttjar ett annat API som inte har implementerats.

Design/UX:

Det är inte jättesnyggt, helt enkelt om man skall vara ärlig.

Saknas en del UX element, exempelvis på indikation att bollen går att interagera med vid t.ex. mouseover.

Säkerehet:

Just nu kan man ta sig till sidan för att ladda upp highscore och även modifiera sin poäng genom att titta på SessionStorage där dessa värden ligger sparade.

API nyckeln för FireBase borde egentligen läggas I en separat JSON fil och läsas in via AJAX för att kunna låta programmet använda olika databaser utan att behöva ändra I kod.

**Sammanfattning/Utvärdering:**

Slutgiltig produkt levde inte riktigt upp till ambitionerna, men utvecklades med nya idéer under tiden.

Det blev endast raka väggar som implementerades istälet för vinklade, eller cirklar. Men istället implmenterades rörliga objekt som aldrig var en del av ursprungsplanen.

Samarbete var inte närvarande.

Men allt som allt en duglig alfa.