Jag hade från början svårt att komma på en idé till mitt spel. Idéen att använda hänga gubben som grund för mitt spel kom efter att jag fastnat i ett hänga gubben spel om länder och huvudstäder i två timmar när jag sökte efter inspiration. Jag tycker spelet i slutändan blivit rätt bra ändå. Under spelets gång så finns det fem olika kategorier av frågor. De bygger lite på varandra och spelaren måste klara en av frågorna från varje kategori för att bevisa att de behärskar varje kategori innan de kan gå vidare till nästa. På så viss så blir spelet progressivt svårare och spelaren får använda vad de lär sig tidigare i spelet senare i spelet.

Jag har använt mig av ARCS modellen när jag planerat detta spel.

ARCS

ARCS modellen är en teori skapad av John Keller som beskriver fyra olika kategorier som måste uppfyllas för att en elev ska bli motiverad till att studera. Dessa kategorier är: Uppmärksamhet, relevans, självförtroende och tillfredställelse.

Uppmärksamhet kan väckas på två sätt. Det första är genom vad Keller kallar perceptual arousal. Med det så menas att man använder överraskning och det oväntade för att fånga uppmärksamhet. Det kan vara något i stil med en bok som högljudet dunkas i golvet eller att man skriver ord baklänges på en whiteboard tavla. När det kommer till denna uppgift så skulle perceptual arousal vara något i stil med att man har roliga videos och färgglada bilder i spelet. Jag har inte använt mig av denna metod för att fånga uppmärksamhet. Detta för att jag helt enkelt inte visste om jag skulle ha tid att skapa avancerad multimedia.

Den andra metoden för att väcka uppmärksamhet, den jag använt mig av, är inquiry arousal. Med det så menas att man ställer utmanande frågor och ger eleverna komplexa problem att lösa för att väcka deras uppmärksamhet. Min tilltänkta målgrupp är personer som precis börjat programmera i unity. Frågorna som spelaren ställs är tagna från problem jag själv hade när jag började programmera. De är designade för att vara grundläggande och simulera vanliga problem som kan dyka upp när man är ny till spelgrogrammering. Tanken är att målgruppen ska känna igen vad de ser och bli intressade av problem som påminner om problem de själva haft när de försökt bygga spel. Detta för mig till Kellers andra punkt.

Relevans handlar om att eleven ska känna att det som lärs ut är något de vill kunna. I detta fall så är spelaren någon som precis börjat programmera i unity och de frågor som ställs är tänkt att påminna om problem eleven själv kan tänkas ha stött på. På så viss så känner eleven att det som lärs ut är relaterat till vad de vill lära sig.

Självförtroende handlar om att eleven ska känna att de kan klara av de uppgifter som ges. Detta kan åstadkommas genom att ge eleven varierade uppgifter samt genom att eleven själv får ha lite inflyttande över hur de lär sig. Under spelets gång så kommer eleven ha tillfällen att lösa många olika sorters problem. Det finns även olika uppgifter att lösa för varje kategori av problem så de kommer ha möjlighet att själva bestämma hur de ska lära sig.

Med tillfredställelse så menas att det eleven lär sig ska kännas meningsfullt och att de kommer ha nytta av det i framtiden. Detta åstadkoms genom att ge spelaren vad Keller kallar *intrinsic rewards.* Med det så menas att belöningen är immateriell eller ogripbar. Till exempel att spelaren får en känsla av framgång och lycka från att ha klarat uppgifterna. Det finns även så kallade *extrinsic rewards* vilket man lite lätt skulle kunna säga är något mer fysiska eller materiella så som pengar eller en trofé.

Utvecklingen av spelet

Det första större problemet jag hade när jag skulle göra detta spel var att skapa knapparna med tecken. Jag visste att jag skulle behöva många knappar. Jag visste dock inte exakt vilka knappar jag skulle behöva. Så jag ville ha något sätt att snabbt lägga till nya tecken om jag behövde det. Att skapa 50 - 60 knappar manuellt i mitt html dokument kändes lite tradigt. Jag tänkte att det kanske skulle gå att skapa dem med JavaScript lite effektivare. Detta visade sig vara mycket krångligare än jag trodde det skulle vara. Efter mycket funderande, googlande och många Youtube videos senare så hittade jag dock ett bra sätt att göra det. Vad jag gör är att jag har en variabel med alla tecken jag vill göra till knappar. Sedan så splitar jag den med ”” så att jag får en array med alla tecken. Sedan så mappar jag varje tecken med en funktion som skapar en knapp med tecknet. Därefter så använder jag join för att göra arrayen till en sträng igen. Då kan jag lägga in strängen i en paragraf så blir det en rad av knappar.

Något jag inte är supernöjd med i det JavaScript jag skrivit är hur jag gått tillväga för att se till så att rätt ord hänger ihop med rätt bild, ledtrådar och beskrivning. Som det är nu så ligger de i separata arrayer och det gäller att allt ligger på samma position. Till exempel så måste bilden, ledtrådarna och beskrivningen för det tredje ordet också ligga på position tre i sina arrayer. Detta kompliceras ytterligare av att många av dessa arrayer är arrayer av arrayer. Så för att den andra frågan på plats fyra ska ha rätt bild så måste den andra den bilden på plats fyra i bildarrayen ligga på rätt plats. Det skulle vara bra att skapa en separat frågeklass som innehåller en bild, ledtrådar och beskrivning. På det viset så skulle det bli lite mindre komplicerat om jag hade många frågor. Dock så har bara fem olika kategorier av frågor så är det inte så jätteförvirande än.

Något annat jag inte är jätteglad över är hur jag tog fram kodproblemsbilderna. Vad jag gjorde var att jag tog en skärmbild och la in den i Photoshop. Sedan så jagade jag pixlar och placerade bilden precis så att det ser ut som att endast koden ändrades. Detta var pilligt. Jag behövde även göra detta varje gång skulle ändra ett problem. Vet dock inte hur jag kunnat göra det bättre. En möjlighet skulle vara att skriva in koden direkt i spelet istället för att använda bilder. Det bra med att ha en bild är dock att det känns lite mer som att spelaren verkligen sitter och programmerar då de får se en riktig utvecklingsmiljö istället för bara lite text.

En ändring som skett under spelets utveckling är att det nu finns två någorlunda lika frågor till varje kategori av problem. Spelaren måste även svara rätt på en av dessa frågor innan de får gå vidare till nästa kategori av frågor. Dessa två ändringar gör så att spelaren måste bevisa att de behärskar varje område innan de kan gå vidare till nästa. Detta tillåter mig även att bygga senare frågor så att de använder element från tidigare frågor då spelaren visat att de kan hantera det. Då blir spelet progressivt svårare då problemen blir mer komplexa.