Filer jag jobbade på:

BezierCurve.h

BezierPath.h

BezierPath.cpp

DrawCurve.h

DrawCurve.cpp

BaseEnemy.h

BaseEnemy.cpp

EnemyTypes.h

EnemySpawnManager.h

EnemySpawnManager.cpp

EnemySpawnParameter.h

EnemySpawnParameters.cpp

Background.h

Background.cpp

BezierCurveManager.h

BezierCurveManager.cpp

BezierCurve

Denna funktion skapar fyra BezierCurve punkter som ja kan använda i BezierPath. första punkten är startPositionen, andra är armen som styr var första punkten pekar och hur lång den är. Punkt tre och fyra är lika som ett och två, förutom att punkt fyra är slutpunkt och punkt tre är armen.

Jag förstår hur detta funkar funktionellt, då jag vet hur en bezier curve ska funka, men matematiken är något som jag inte är helt med på. Detta hämtades från en tutorial som jag gjorde, men tyvärr fastnade inte all information om det. Det får jag återgå till någon gång när jag väl har mer kunskap för att förstå matematik logik i programmering.

Dock så såg jag till att testa detta och få det funka till våra ändamål i projektet så att vi inte har något som är oanvändbart.

BezierPath

Denna class använder sig av BezierCurve för att göra en hel kurva med hjälp av en lista. Den lägger till punkterna i listan och sedan ger den en sample rate som styr hur många punkter det är i kurvan och även hur hög "poly count" det är.

Även detta är från samma tutorial som BezierCurve.

Originellt så använde jag detta för att sätta ett objekts position till samma position som en punkt i Path listan, men då behövdes flera hundra i samples för att få en smooth animation. Detta låste även objekten till satta positioner på skärmen. Men Martin ändrade på detta så att ett objekt använder punkterna som waypoints till ett enkelt pathfindingsystem och på detta sätt så kan objekten kunna gå ut och in i animationen ifall det behövs. Detta gör så att det behövs bra mycket mindre samples för att göra animationer på kurvor.

Det är sånt här som jag behöver fördjupa mig i, hur man kan använda det man skapar på bästa sätt. Men det kommer nog med tiden.

Här har jag samma problem som sist, jag förstår hur klassen funkar och vad den är till för, men jag kan nog inte skriva detta själv.

DrawCurve

Även denna är gjord från en tutorial och den är till för att visualisera hur kurvan ser ut på skärmen.

Detta visade sig vara ganska onödigt enligt mig, då jag ändå var tvungen att skriva koden för hand om hur man skapar kurvan, så jag var tvungen att köra på trial and error tills jag fick rätt kurvor ändå.

Det var även snäppet mer komplicerat än från tutorialen, då vi använder en egen renderare, så jag var tvungen att konvertera detta från SDL till vår egna renderare.

BaseEnemy

Detta är basklassen för alla enemies. Jag använde en grund som Martin skrev så att Update, render och collisions skulle fungera. Sedan skrev jag hur fienden sköt och rörde på sig. Jag använde mig av enumes för att styra hur man väljer rörelse och patterns för bullets. Detta gjorde så att vi kunde få ganska mycket variation på fiender. Och slår man ihop detta med Light och Dark funktionen vi har, så får man dubbelt av allt att välja på.

Health systemet är något Martin lade in, så det hade har inte mycket att säga om. Men av det jag läste från det, så är det inte så speciellt komplicerat. Det är ett system som styr en health meter med hjälp av en health/damage parameter.

Som sagt, så ändrades hur rörelser funkade av Martin vid en senare tillfälle, men detta förklaras i BezierPath.

Jag kommer skriva mer om inheritance när jag förklarar Enemy Types. Men jag är med på att detta är egentligen ganska knasigt sätt att använda inheritance. Men det funkar. :)

Det här är vad jag la mest tid på, att skapa olika rörelser och patterns för olika fiender och att få det att funka. det var en hel del trial and error och en hel del som fick göras om när vi väl bytte olika system i grunden, som t.ex. positionssystem och entitymanager.

Jag ändrade egentligen även hur rörelsen gick också, då originellt så gjorde ja bara olika matematiska funktioner som styrde olika roliga rörelser. men Curves är mer flexibelt att jobba på och det är mindre tänk bakom att använda det.

Det var inte mycket som var konstigt för mig här, det var bara att förstå hur jag ska använda Martins olika system för att få det att funka och det är detsamma med BezierCurves.

EnemyTypes

Här lade jag in några olika kombinationer av fiender, som jag skapade som children till BaseEnemy och kallade dom efter vilken bulletPattern och movementPattern de ska använda. Sedan la jag till variationerna för färgen på fienden också. Alla dessa användes inte i slutprodukten och många fler kunde skapas. Jag använde inte alla, då det blev ganska kluddigt på skärmen då.

Som sagt, så är detta system knasigt, då jag egentligen bara använder det för att skapa kopior av BaseEnemy, istället för att göra en riktig inheritance där jag byter ut variabler och funktioner i själva childen. Men jag behövde ett snabbt sätt att lösa något som hade tagit mycket längre tid att göra och lära sig, så det fick duga. Sen blir namngivningen också väldigt konstig, för att egentligen så har inte namnet inte något att göra med hur den beter sig. När den fienden väl skapas, så kan man sätta vilka parametrar som helst utan att namnet gör något överhuvudtaget. Jag får helt enkelt lita på att användaren sätter ett klokt namn.

Inheritance kan jag nog lösa i framtiden, men namngivningsproblemet är en svårare nöt att knäcka. Jag tror att jag måste antagligen ändra hela det systemet till att funka med den Engine som gjorts för att hitta en bra lösning.

Detta var en idé som jag fick från en av mina klasskamrater och det funkade väldigt bra, då vi behövde kunna skapa nya klasser av fiender lätt, då vi inte kan använda dubbletter av klasser i vårt entitymanager system. Iallafall inte dubbletter med olika parametrar.

EnemySpawnManager

Ett system som är till för att skapa fiender slumpmässigt. Här använder jag en Spawner som Martin skapade åt mig och sedan skapade jag nya funktioner för att kunna skapa nya patterns och för att få det att funka på mitt sätt. Hit flyttade jag även all sprite creation för fiender, då det kändes mest logiskt att det skulle vara. Innan så fanns det i vår "main"(Jag säger main, men det är vår GameState klass egentligen, men det är lättare att förklara på detta vis).

Det enda användaren behöver tänka på egetligen är Spawn funktionen, där man väljer hur många fiender det är i kö, då jag ville att det skulle vara simpelt att använda.

Mot slutet av projekttiden vi hade så la jag även till bossen som Martin hade skapats. just nu spawnar den ur tomma intet, där den borde röra sig in i skärmen, men det fanns inte tid för mig att fixa, så jag fixade bara att alla fiender slutade spawna när bossen skapades och att den skapades vid en slumpmässig tid mellan 30-45 sec.

När jag tittade på bullethell spel så brukar fiender komma bakom varandra som en kö ibland. Och det var det jag försökte lösa med SpawnXTimes funktionen. Den är dock lite buggad, då den första gången nästan alltid skapar en fiende för lite i kön, men det är inget man ser visuellt. jag trodde jag löste det ett tag, sen kom det tillbaks.

Allt detta gjordes originellt i våran "main", men det blev väldigt kluddigt och förvirrande att ha med det där, så jag flyttade allt till en egen klass.

Det här var inte så konstigt, jag löste det ganska fort efter lite trial and error.

EnemySpawnParameters

Denna klass spakade jag för att jag behövde nånstans att spara information om själva enemies som ska spawnas som jag kunde hämta, istället för att göra listor av allt och sedan ta fram genom indexes, vilket är jobbigt. Jag ville egentligen ha själva spawners som jag använde i spawnManager här också, men av någon anledning så fick jag en superkonstig bugg som andra och jag inte hade svar på. När ajg gjorde det, så började programmet säga att min Constructor för GameState inte existerade, vilket den klart och tydligt gjorde. Så det fick vara separat för tillfället.

Inge konstigt här, allt säger sig självt enligt mig, det är bara info som jag tar fram senare för att använda.

Background

Bakgrunden är ganska simpel. den laddar som en entity så EntityManager kan Uppdatera och Rendera den.

Sen i Update Funktionen så flyttar den position tills den kommer till gränsen där den flyttas tillbaks så den loopar.

Här var det inga problem, fick testa lite, men gick ganska fort att lösa. Min baskunskap som artist löste det mesta logiska om hur en loopande textur ska funka.

BezierCurveManager

Den här managern var en lösning så att jag slapp skapa kurvor varje gång jag skapade en fiende, vilket var onödigt mycket datorkraft. Detta var ett tips av Martin som var en bra lösning.

Härifrån så hämtar man kurvorna, som man offsettar med objektets spawn position(objektet som ska använda kurvan).

Inge konstigt här heller, det var ganska lätt att lösa. Dock så tog det en stund, då det var en hel att flytta.

Generell Information

Overall så blev slutprodukten helt okej. Dock så hade vi lite problem med motivationen till att skapa projektet.

Jag ville ha ett system som i ett bulletHell spel som jag känner till (Ikaruga) och jag tyckte att vi fick in det ganska bra. Det var mest Martin som löste allt rörande hur det systemet skulle funka.

Jag ville egentligen skapa en ny engine tillsammans med Johannes, då jag tyckte att vi två behöver kunskapen. Men jag förstår att det hade tagit för lång tid att göra och det var okej att använda ett av dom som fanns(Vi valde ditt ;)).

Individuellt

Jag tyckte att det jag gjorde för projektet funkade ganska bra med resten av vår Engine och jag försökte verkligen implementera det som fanns i Enginen i allt jag gjorde, så att det inte skulle kännas som separata klasser.

Ett av mina svagheter är ju fortfarande att jag inte har tillräcklgit mycket kunskap, så det är väl nåt jag borde jobba mer på. men jag har börjat förstå grejen och ska nog koncentrera mig på att göra lite mer avancerade grejer framöver, så kommer baskunskaperna stärkas av det.

Jag använder fortfarande för mycket repeterande kod, för lite assertions och för lite kommentering, så jag behöver jobba på dom grejerna.

Jag själv tyckte att det var konstigt att det var ett gruppProjekt, då jag hade nog varit mer motiverad att göra det själv, så jag kan använda dom kunskaper jag samlat på mig de senaste månaderna. Och sedan jämföra det med det jag lyckades åstadkomma med Snake projektet.

Teamwork

Jag tyckte att samarbetet med Martin funkade bra, men Johannes hade jag mer problem med det. Han kommunicerade inte så ofta och var ganska envis med att jobba på det han ville få in i projektet, istället för att diskutera med oss andra.

Johannes har tydliga problem med att hänga med att lära sig programmering och antagligen inte pluggar tillräckligt mycket. Vi försökte få han att fråga mer om hjälp, men det gick inte så bra. Jag kanske borde ha pushat på mer, men för mig känns det konstigt att behöva göra.

Överlag gjorde han inte något nästan.

Sen var som sagt, motivationen ganska låg för detta projekt i gruppen, så vi kanske borde ha tagit det mer seriöst.

Jag vet inte om det var gott eller ont, men engine arbete var något som Martin fick göra på egen hand, men jag fick ju egentligen tillräckligt att göra på egen hand.

Idén och implementationen av idéer tyckte jag funkade ganska bra. Hade vi nåt att lägga till, så la vi till det. om något var för svårt för mig att göra, så tog Martin över och om han hade för mycket så tog jag över.