Veckoplanering

# Vecka 8

Veckans arbete kommer att delas upp i två olika steg. Första steget kommer vara att leta fram uppgifter som styrker våran problemställning att implementera multiplayer i spel är något som är svårt och omständligt i efterhand. Detta kommer ske genom att söka efter post mortem ifrån spelbranschen samt att söka på forum där folk har detta dilemma.

Andra delen utav veckans steg kommer vara att få upp en miljö att börja arbeta i samt att påbörja implementation utav ett basic FPS spel. Förfarandet kommer vara väldigt simpelt till en början med en kub som kan röra sig i en plan virtuell värld och kollidera med väggar.

# Vecka 8 - utvärdering

Vi började med att skicka ut ett mail innehållandes vår frågeställning och frågan om det finns ett problem ute i industrin och vad de tycker om problemet i stort. Vi fick ett svar ifrån studion arrowhead game studios som ligger bakom det välkända spelet magicka. Förutom de såg vi över post mortem's och forumtrådar inom området och kom fram till det är ett utbrett problem både ute i industrin och bland vanliga hobby-utvecklare.

På utvecklingsfronten påbörjade vi ett FPS 3D spel. Vi började väldigt simpelt med kub som kunde röra på sig. Sedan la vi till en mer avancerat kamera som betedde sig mer likt en FPS-kamera. Banan konstruerades i Blender genom att segmentera ett plan och förskjuta vissa segment för att skapa "väggar". Senare la vi till kollisionshantering genom att använda oss utav fysikmotorn Bullet som lätt kan integreras med LibGDX. Detta ledda till att fysiken som annars skulle bli en relativt stor sak att implementera snabbt och enkelt överstökades. Där av kunde mer tid läggas på att polera upp den enkla prototyp vi hade.

# Vecka 9

Denna vecka planerar vi på att ha en enklare nätverksstruktur uppe. Målet är att kunna koppla samman två datorer och få ut två stycken spelare i samma värld. Till en början kommer inget fokus lägga på att strukturera upp något direkt ramverk utan bara separera funktionaliteten.

# Vecka 9 - utvärdering

Till en början undersökte vi användandet utav Java IO vs. NIO. Vi valde att designa vårt nätverk utefter Java IO, detta p.g.a. att NIO är mer optimerat för flera tusen samtidiga uppkopplingar vilket vår applikation inte kommer att hantera. Java IO använder sig mer utav separata trådar för varje TCP uppkoppling samt en gemensam UDP uppkoppling.

När väl implementationen påbörjades valde vi först att försöka fokusera på endast en UDP uppkoppling för att bryta ned problemet lite. Detta var något som inte direkt följdes utav vi ändrade istället förfarande till att den ena utav av oss skrev klienten och den andra servern. Då med fokus på både TCP och UDP.

Vi fick till slut upp en grundläggande struktur för hur vill ville att nätverket skulle se ut, och påbörjade implementation utav pakethantering. Ett första utkast av denna funktionalitet resulterade i paketklasser som ärver från en basklass "Packet". Packet klassens funktionalitet är att lägga en bytebuffer runt datat som ska skickas in och lägga in paketets id först så den som tar mot paketet vet vilken typ av paket det är.

När väl nätverk och pakethantering var implementerat så valde vi att påbörja lite testimplementation mot koden genom att skicka ett uppkopplingspaket och ett paket som ger den frågande parten information om hur tillståndet på servern ser ut.

# Vecka 10

Då vi förra veckan fick små problem med nätverket i sista stund väljer vi att till en början denna vecka se till att fixa till dem. Övrigt ska vi ta fram en mall för gruppering utav klasser så man enkelt ska kunna avgöra om dem tillhör nätverk, spellogik eller mixade. Detta för att få något vi kan mäta mot när vi väl börjar ändra på vår design.

Vi ska även ta ett steg tillbaka och kolla på olika designer som kan tänkas fungera och reflektera över dessa.

# Vecka 10 - utvärdering

Då vi till en början var tvungen att förbättra vår nätverks-kod lade vi stor vikt vid att försöka se till att inga ytterligare problem skulle dyka upp vid ett senare tillfälle vilket ledde till att vi skrev om koden en del samt att pakethanteringen blev betydlig mer dynamisk. Detta genom att server/klient har vissa paket som denne registrerar och att användaren sedan kan registrera egna paket utan att behöva gå in i nätverkskoden över huvud taget.

Vi valde även att skriva en mall som man ska kunna följa för att gruppera sina klasser i de tre grupperna nätverk, spellogik samt mixade. Detta applicerade vi sedan på våra klasser vi hade och utförde mätdata av hur stor del av våra klasser som tillföll diverse grupp. Vi valde även att gå in mer noggrant på de mixade klasserna då vi tycker att dem är de som stor vikt bör läggas på. Vi påbörjade mätning av hur stor del procentuellt de mixade klasserna bestod utav nätverk kontra spellogik.

# Vecka 11

Planen för veckan är att påbörja implementation utav features och mäta hur dessa påverkar interaktionen mellan spellogik och nätverk. Till en början tänkte vi implementera en enklare rörelsehantering och skicka detta via nätverk. Vi påbörjar implementationen genom att endast göra den som spellogik och mäta tid samt hur många rader kod den innebär. Efter detta sker övergången till nätverk och implementationen där.

# Vecka 11 - utvärdering

Move i singleplayer var sedan innan redan implementerad men själva sättet den var utförd på var inget vidare då den smutsade ned andra klasser onödigt mycket. Däribland PlayerController eller PlayerInput som vi valde att döpa om den till. Efter rekonstruktionen av PlayerInput blev den en renodlad Logik-klass och inte en mixad. Detta lyfte ut nätverkshanteringen till NetworkScreen vilket resulterade i att saker kan skickas direkt därifrån över nätverket.

Multiplayerimplementationen av move sker genom att när spelaren flyttat på sig eller roterat skickas ett paket till servern. Detta är ett naivt sätt att hantera förflyttning på då servern inte är "boss", utan istället bara uppdaterar till vad klienten säger.

# Vecka 12

Veckans arbete kommer lägga fokus på att implementera funktionen att skjuta över nätverk. Funktionen är redan sedan innan implementerad visuellt i singleplayer så det som kommer behövas är att lägga till att man kan skjuta något/någon och få feedback via ljud.

Utöver det kommer fokus ligga på att försöka förändra den övergripande designen. En förbättring som ska utvärderas är att skapa upp en worldmodel som sedan serverworldmodel samt clientworldmodel ärver utav. Detta för att få en så ren singleplayer miljö som möjligt.

# Vecka 12 - utvärdering

På grund utav att vi sedan innan hade funktionalitet för att i singleplayer skjuta ett laserskott gjorde det denna veckas mätning lite tvetydlig och svårvaliderad. När vi försökte att implementera skjuta på multiplayer fick vi snabbt en insyn att singleplayer funktionaliteten var tvungen till att skrivas om för att kunna på något sett valideras. Detta ledde till att vi var tvungna att bryta ut saker som låg i player och som egentligen inte skulle ligga där i första början. Efter den omstruktureringen var det inga större problem med implementeringen i multiplayer.

# Vecka 13

Denna vecka kommer stor vikt läggas vid att refaktorisera koden till en mer strikt MVC struktur då det under implementationens lopp lyckats smyga in sig lite "fel". Efter en omstrukturering ska en feature att kasta granat implementeras, först i singleplayer sedan i multiplayer.

# Vecka 13 - utvärdering

Veckan började med att en refaktorisering skulle göras och detta för att förbättra vår design samt att upprätthålla reglerna MVC medföljer som vi nu brutit mot. Tidigare hade vi Modelinstancer i modellklasser. Detta är något som bryter extremt mycket mot reglerna då en modelinstance egentligen är en visuell representation av vad en modell står för. Detta har brutits ut till respektive vyklasser. Jobbet med att separera denna funktionalitet var inte särskilt krävande. Dock behöves validering utav designen för att hantera packet som skapar upp fysikkroppar då nätverk och gameplay ligger på olika trådar och fysikkroppar kan inte skapas när gameplay uppdatera fysiken. Detta medför att saker måste läggas i temp arrayer vilket inte känns bra.

SyncedUpdate....