Universe Engine Rapport

Alexander Mennborg

2017-05-08

Inledning

I denna rapport diskuteras utvecklingsprocessen till projektet Universe Engine. Denna diskussion omfattar hela utveckling från starten till nuläget och ger även förklaringar till olika större förändringar. Några nya viktiga idéer och mål till projektet kommer även att presenteras i rapporten. Till sist kommer jag att beskriva om vad jag gör just nu och hur framtiden ser ut för utvecklingen av projektet.

1. Bakgrunden till projektet

Projektet inleddes våren 2016 som ett enkelt verktyg som skulle lösa ett specifikt problem. Vår programmerings lärare Niklas Gyulai hade en idé om att skapa ett tre-dimensionellt spel tillsammans med andra-års gymnasie-elever. Vi fick ansvaret att programmera ett verktyg som skulle användas för att skapa tre-dimensionella miljöer. Vi skulle även inkludera några lärare från skolan som skulle programmeras med en enklare artificiell intelligens. Målet med hela projektet var att kunna modellera vår skola, Berzeliusskolan. Det var ett relativt ambitiöst projekt. Eftersom detta var första gången vi hade arbetat med tre-dimensioner eller mer specifikt Java3D biblioteket så tog det en stor del av tiden att öva och bli van vid det. Från början hade vi inte ens planerat att använda oss av något användargränssnitt eftersom det endast komplicerar utvecklingsprocessen. Jag har sedan tidigare utvecklat applikationer med användargränssnitt och bestämde mig tidigt att jag skulle implementera det själv. Naturligtvis tog det längre tid att utveckla programmet men det var mer motiverande att utveckla en användarvänlig applikation som kan användas av alla. Mot slutet av kursen blev vi introducerad till Datasaabs Vänner av vår lärare. Jag fick ett stipendium för mitt verktyg som jag kallade J3DBuild (Java3D Build). Även om jag inte var färdig med programmet då så var det ändå mer motiverande att fortsätta utveckla detta vidare. Vid mitten av sommaren bestämde jag mig att avsluta utvecklingen av J3DBuild projektet, läs mer om anledningen till detta i andra avsnittet.

Jag har sedan tidigare experimenterat med att implementera en mindre spelmotor och bestämde mig då för att fokusera på att skapa min egna spelmotor som mitt nya projekt. Spelmotorer har blivit större och nödvändiga för att skapa avancerade spel. Motivationen till att skriva en spelmotor är inte att göra utvecklingsprocessen snabbare utan endast att skapa återanvändbar kod till sina framtida spel. Med det nya projektet hade jag inte riktigt samma mål som med J3D Build. Jag gav detta projektet namnet Moonlight Engine men detta var inte exakt samma projekt som det nuvarande projektet Universe Engine. Båda projekten är en så kallad ”motor” (Engine), begreppet motor diskuteras i tredje avsnittet. Moonlight Engine var främst fokuserad på att producera spel och kallas därmed för en spelmotor. Begreppet spelmotor är ganska svårt att definiera eftersom det kan ha olika betydelser till olika sammanhang men generellt sätt så är det ett verktyg för att enklare kunna skapa spel. I början av hösten bestämde jag mig för att fokusera på att konstruera en generell motor som inte är optimerat för en specifik typ av applikation eller enhet. Detta projektet gav jag då namnet Universe Engine för att det ska bli en universell motor som kommer bli oändligt stort precis som universum självt. Tekniskt sett så är det inte möjligt men teoretiskt sätt om vi har tillräckligt med resurser så ska man kunna bygga vidare oändligt mycket eftersom jag har designat programmet att vara modulärt. Detta betyder att man programmerar små moduler som man kan utöka funktionaliteten.

2. Varför avslutade jag J3D Build?

Som jag tidigare nämnde i första avsnittet så utvecklades detta projektet i biblioteket Java3D. Java3D använder ett av de moderna grafikhanterings systemet OpenGL (Open Graphics Library) för att rendera tre-dimensionell grafik. Tidigare när man skulle rita grafik så använde varje hårdvara sina egna metoder och tekniker för att rita grafik på skärmen så när man vill att sitt program ska köras en annan dator så fungerar då inte grafiken eftersom den andra

datorn inte körs på samma grafikkort. För att fixa detta problemet så skapades det plattforms oberoende grafik biblioteket OpenGL som kan stödjas på flera grafikkort som har uppdaterade drivrutiner. Numera stödjer alla grafikkort någon version av OpenGL. Fördelen med att använda Java3D istället för ren OpenGL är att Java3D är ett objekt-orienterat hög nivå bibliotek och OpenGL är skrivet i låg nivå språket C. Den största nackdelen med Java3D är att det saknas funktionalitet eftersom den inte har uppdaterats sedan 2008 och är nu 9 år gammal nu. Jag hade en ganska dålig upplevelse av Java3D eftersom det var inte så bra skrivet och kändes mer av ett hinder varje gång jag programmera göra något nytt med det. När jag började bli bekväm med OpenGL så ville jag istället utveckla projektet utan Java3D och bestämde då att avsluta J3DBuild och startade Moonlight Engine projektet.

3. Nya Projektet Universe Engine

Projektet Moonlight Engine kom aldrig långt in i utvecklingen innan det gjordes om till Universe Engine. Den största skillnaden mellan dessa två är att Moonlight Engine är en konkret spelmotor men Universe Enigne är en generell motor. Men från början var detta projekt enbart ett verktyg. Man kan säga att en motor är ett verktyg också men det som skiljer är abstraktionen. Ett verktyg är anpassat för att en vanlig människa ska kunna använda det men en motor är endast anpassat för en programmerare och kallas vanligtvis för bibliotek inom datateknikens terminologi. Verktygen i detta fall är en vanlig applikation men motorn eller biblioteket är endast en bunt med kod som en programmerare kan använda sig av. Men det finns en annan del som oftast ingår i moderna spelmotorer (ex. Unity, UnrealEngine) är att det medföljer ett användarvänligt verktyg eller ”editor”. Det är tänkt att Universe Engine också ska ha ett liknande verktyg för att enklare utnyttja motor koden utan att behöva lära sig allt om koden. Ett tidigare mål som fortfarande ska uppfyllas var att det ska vara användarvänligt sådant att man kan lära sig programmering. Detta mål kommer även

förlängas sådant att man även ska kunna utnyttja grafiken delen av motorn för att enkelt kunna visualisera sin kod med figurer, diagram etc.

Som jag har nämnt tidigare så skulle J3DBuild användas för att utveckla tre-dimensionella miljöer. Java3D är byggt på en så kallad scengraf (engelska Scene Graph) eller inom grafteori terminologi kallas det även för riktad icke-cyklisk graf (engelska Directed Acyclic Graph). En scengraf liknar ett träd, där man har en så kallad rot nod och grenar ut trädet med flera noder och har löven längst ut på grenarna. Detta kallas för en datastruktur och är en objektorienterad metod för att lagra information. Det viktiga är inte definitionen men en tydlig förståelse av detta ger hela nyckeln till hur Java3D biblioteket och många andra spelmotorer fungerar. Man använder det för att lagra all information om sitt nuvarande tillstånd. Mitt nya projekt kommer implementera en liknande scengraf för att kontrollera alla element i en miljö. Skillnaden nu är att tidigare kunde man endast producera en fil som innehåller själva grafen men med det nya projektet ska man även kunna producera riktiga exekverbara applikationer. Det är tänkt att man kan skapa program för alla enheter som exempelvis Android, IOS, Windows, HTML osv.

Som jag nämnde i första avsnittet så har jag bestämt mig för att använda mig av en modulärt designmönster för att enklare utveckla kodbasen. Projektet är nu uppdelat i två delar där den första delen är själva kärnan som jag kallar för Universe Core och den andra delen är själva verktyget där man kan utnyttja kärnan och andra moduler för att utveckla sin egen applikation. Några fördelar med en modulärt designmönster är att man inte behöver låta alla användare använda sig av hela kodbasen utan kan sprida ut koden i moduler som användaren frivilligt kan ladda ner och använda. Det ger oss också möjligheten för andra utvecklare att lägga till sin egen funktionalitet som dessa själv saknar och dela med sig till andra att ladda

ner. Det blir också enklare att sköta uppdateringar eftersom man slipper uppdatera hela programmet om något behöver ändras i en av modulerna så kommer alla använda kräva en utveckling men med denna design så får endast personer som har valt att ladda ner den modulen updateringen. Ett bra exempel på denna strukturen är NodeJS som använder pakethanteringssystemet npm (Node Package Manager) och fungerar på liknande sätt.

Något begränsat med J3DBuild projektet är att det kunde endast användas för att konstruera statiska miljöer men i det nya projektet kommer även dynamiska miljöer bli till verklighet. Med en statisk miljö menas att det inte finns något beteende i världen utan allt i miljön är redan förbestämt av den som har designat världen. Men med en dynamisk miljö kan man konstruera gräs som blåser i vinden eller komplicerad artificiell intelligens för ett djur osv. Som jag tidigare nämnde så var artificiell intelligens redan planerat från början men skillnaden är att det beteendet måste programmeras separat med ett annat program i java men i det nya projektet så kommer detta vara integrerat i verktyget och motorn på ett sätt som kommer underlätta arbetsflödet.

4. Vad pågår just nu?

Just nu pluggar jag på Luleå Tekniska Universitetet till civilingenjör inom datateknik. Detta är mitt första år av fem planerade år. Just nu pluggar vi Linjär Algebra, Integralkalkyl och Digitalteknik. Under det första året har vi pluggat mestadels matematik som vi senare kommer använda i våra framtida kurser. Matematiken vi läser på universitetet, främst linjär algebra är viktig inom datateknik och även det projektet eftersom jag behöver kunna arbeta med två- och tre-dimensionell grafik vilket kräver kunskap inom linjära transformationer. Vi har tidigare läst diskret matematik vilket innehåller en del grafteori som jag tidigare nämnde kommer användas till projektet. Inom digitalteknik läser vi om hur datorn och främst processorn fungerar på kretsnivå då vi ska implementera en förenklad 32-bitars MIPS

processor på en FPGA (Field Programmable Gate Array) med VHDL-kod. Vi har även läst en kurs om programmering i Python och en annan om objektorienterad programmering i Java. Eftersom jag har varit upptagen med universitetsstudier denna termin och förra så har jag inte åstadkommit mycket för projektet.

5. Framtiden för projektet och stipendium

Framtiden för projektet är stort likt som projektet självt och därför kan det vara svårt att sätta en deadline just nu. Det kan ta många år innan jag kan släppa en färdig produkt men eftersom jag fortfarande kommer plugga i fyra år till så är målet att ha det mesta färdigt till examen. Under utvecklingstiden ska jag försöka dokumentera och ge uppdateringar så det går att följa utvecklingen av projektet. Om allt går enligt min ofullständiga plan så kommer det tillslut bli en produkt som går att sälja om det fortfarande finns behov av en sådan då. Det är inte säkert hur det kommer se ut i framtiden, det kan ändras fort. Det jag är säker på är att jag har valt rätt utbildning då det kommer behövas denna sortens kompetens även vid en hel digitalisering av samhället. Om detta projektet fortfarande kommer vara relevant i framtiden så är kommer stipendiet att förbrukas för att marknadsföra den slutliga produkten. Exempelvis kommer produkten att behöva en hemsida för att visa och distribuera programvaran. Jag kommer om någon månad att publicera koden för projektet online på Github där man kan följa utvecklingen av projektet. Jag ska även försöka uppdatera hemsidan för projektet där jag kommer ge mer detaljerade uppdateringar om projektet.

Avslutning

För att avsluta denna rapport vill jag tacka Datasaabs Vänner för att jag har kunnat tagit del av förra årets stipendieutdelning och jag hoppas att jag kommer kunna visa upp projektet nästa år. Jag tror att år 2017 kommer bli ett intressant år för projektet.