YRKESHÖGSKOLAN
NOVIA

Spelprototyp från grunden med Unreal Engine 4

Niklas Karvonen

Examensarbete för Tradenom(YH)-examen
Utbildningsprogrammet i Informationsbehandling
Raseborg 2020

EXAMENSARBETE

Författare: Niklas Karvonen

Utbildning och ort: Informationsbehandling, Raseborg

Handledare: Kim Roos

Titel: *Spelprototyp från grunden med Unreal Engine 4*

Datum 09.3.2020 Sidantal 38 Bilagor -

Abstrakt

I detta examensarbete går jag genom en grundläggande del för skapandet av en spelprototyp med spelmotorn Unreal Engine 4. Arbetet behandlar spelmotorns användargränssnitt och grundläggande terminologi som användaren kan stöta på. Spelmotorns utvecklare, Epic Games presenteras också samt affärsstrategin som företaget valt för att locka användare och företag att välja deras spelmotor.

Syftet med examensarbetet är att visa hur spelmotorn fungerar samt hur man utvecklar spel då jag för tillfället, enligt egen åsikt endast besitter grundläggande programmeringskunskaper, och jag vill lära mig mera om spelutveckling. Prototypen jag skapar ska också gå att utveckla i framtiden.

Arbetet består av tre olika delar -en teoretisk, en praktisk samt en avslutande del.
I den teoretiska delen går jag igenom spelmotorn och dess utvecklare samt uppbyggandet av själva spelprototypen. I den praktiska delen utformas och färdigställs denna och i den avslutande delen bedömmer jag mitt examensarbete.

Språk: Svenska Nyckelord: Spelutveckling, spelmotor, prototyp

---

BACHELOR'S THESIS

Author: Niklas Karvonen

Degree Programme: Business Information Technology, Raseborg

Supervisor(s): Kim Roos

Title: *A Video Game Prototype from Scratch with Unreal Engine 4*

Date 17.2.2020 Number of pages 38 Appendices -

Abstract

In this thesis I go through the process of creating a videogame prototype using the game engine Unreal Engine 4. The work describes the different parts of the engine and common terminology that the user will encounter. The game engine's developer Epic Games is also presented as well as the business strategy the company uses to attract users and companies into using their game engine.

The purpose of my thesis is to show how the game engine works as well as how core game development works. As of writing this thesis I, according to myself only have basic knowledge in programming and I want to learn more about game development. Future continuous development of the prototype should also be possible.

The work consists of three different parts -one theoretical, one practical as well as a concluding part. In the theoretical part I go through the game engine and I present its developer as well as the construction of my videogame prototype. In the practical part I model and complete my prototype and finally, in the concluding part I assess my own thesis.

Language: Swedish Key words: Videogame development, game engine, prototype

---

Innehållsförteckning

1 Inledning

Dagens spelmotorer möjliggör en snabb och relativt enkel process för skapandet av datorspel. Att endast använda statisk kod för spelskapande är dessutom väldigt svårt för personer som inte har kunskaper inom programmering sen tidigare och man kan få problem nästan överallt. En spelmotor använder sig av olika ramverk som består av olika komponenter t.ex. grafik, ljud och diverse logiska komponenter som används för framtagande av innehåll till spel. Detta ger en möjlighet åt personer som inte har programmeringskunskaper från förr att skapa sina egna spel då spelmotorn sköter de mest grundläggande funktionerna. (TheHappieCat 2015)

1.1 Arbetets bakgrund

Ända sen min barndom har jag tyckt om alla olika typer av spel. Det är ett intresse som inte förändrats särskilt mycket sedan barnsben. Något som däremot har förändrats är min egen kritiska granskning av spel. Jag anser att kvaliteten på många av dagens spel är sämre idag än vad den var förut. Mikrotransaktioner tvingas mer eller mindre över på dagens spelare.

Idén till examensarbete kom under en förberedande kurs inför examensarbetsskrivningen. Vi blev uppmanade att själva komma på ett valfritt ämne att skriva om, något som vi senare kunde vidareutveckla till själva examensarbetet om vi själva önskade. Jag hade då spelat igenom ett nyutgivet spel som gjorts av en liten och relativt ny speltillverkare och jag blev mycket imponerad av spelet och dess design. Detta ledde till att jag tog reda på mer om hur spelet skapades och det dröjde inte länge förrän jag stötte på spelmotorn Unreal Engine 4 och förstod då vilket slags slutarbete jag ville göra.

1.2 Syfte

Min ide är att ta fram en grundläggande spelprototyp, en prototyp som jag själv skapar. Användaren ges möjlighet att fritt röra sig och utforska omgivningen i prototypen.

Jag kommer att avgränsa arbetet rätt mycket, eftersom en alltför detaljrik analys lätt kan göra att mitt slutarbete blir för omfattande. Jag vill också få svar på min egen fundering ifall en person utan tidigare större programmeringskunskaper kan skapa en spelprototyp med Unreal Engine 4.

1.3 Begränsningar

Det finns en massa detaljer i en spelmotor och att gå in på precis alla kan lätt leda till brist på korrekta resultat samt bristfällig dokumentation. Från min sida kommer jag endast att presentera de mest grundläggande funktionerna samt övriga detaljer som krävs för att presentera av mitt arbete. Fokus kommer alltså att placeras på en funktionell spelprototyp istället för att börja detaljera alla detaljer och övrigt konkret i en spelmotor. Eftersom min kunskap om spelskapande är ganska begränsad så har jag också valt att använda färdiga mallar för texturer och dylikt.

1.4 Metodik

Jag har använt mig av en kvalitativ forskningsmetod som går ut på att leta upp nödvändig information om mitt ämne och som jag som skribent tolkar. Målsättningen är att jag ska få en djupare kunskap i den praktiska delen av arbetet. Mina nya kunskaper använder jag sen till att skapa en spelprototyp med Unreal Engine 4.

Jag har använt mig av öppen information som finns tillgänglig på nätet, främst då på spelmotorns egen dokumentationssida för att själv lära mig om spelmotorn.
Jag har också tittat på andra spel skapade med Unreal Engine 4 för att få inspiration. Jag har också använt bilder och skärmdumpar för att kunna presentera mitt arbete mera överskådligt.

2 Val av spelmotor

Trots att mitt arbete handlar om Unreal Engine 4 var det inte helt självklart att jag skulle använda Unreal Engine. Mitt andra alternativ var spelmotorn Unity, som är en av Unreal's största rivaler inom spelområdet. Valet föll slutligen på Unreal Engine 4, eftersom det var spelmotorn jag först blev intresserad av. Jag ville också personligen testa spelmotorn för att se vad jag som studerande kunde åstadkomma.

2.1 Unreal Engine 4

Unreal Engine 4 är uppbyggd med hjälp av programmeringsspråket C++ och skapades av Tim Sweeney, mannen som också grundade företaget Epic Games. Företaget är idag mest kända för Battle Royale-spelet Fortnite som vid skrivandet av detta arbete är ett av världens största spel om man på intäkterna. (Digital Trends 2020)

Spelmotorns namn härstammar från företagets allra första spel som hette Unreal.
Spelet lanserades 1998. Det var samma år som spelmotorn började marknadsföras åt andra spelutvecklare som en tredimensionell spelmotor. Det speciella med Unreal Engine på den tiden var att tredimensionell grafik inte alls var standardiserad hos många av utvecklarna och spelet Unreal gjordes i 3D.

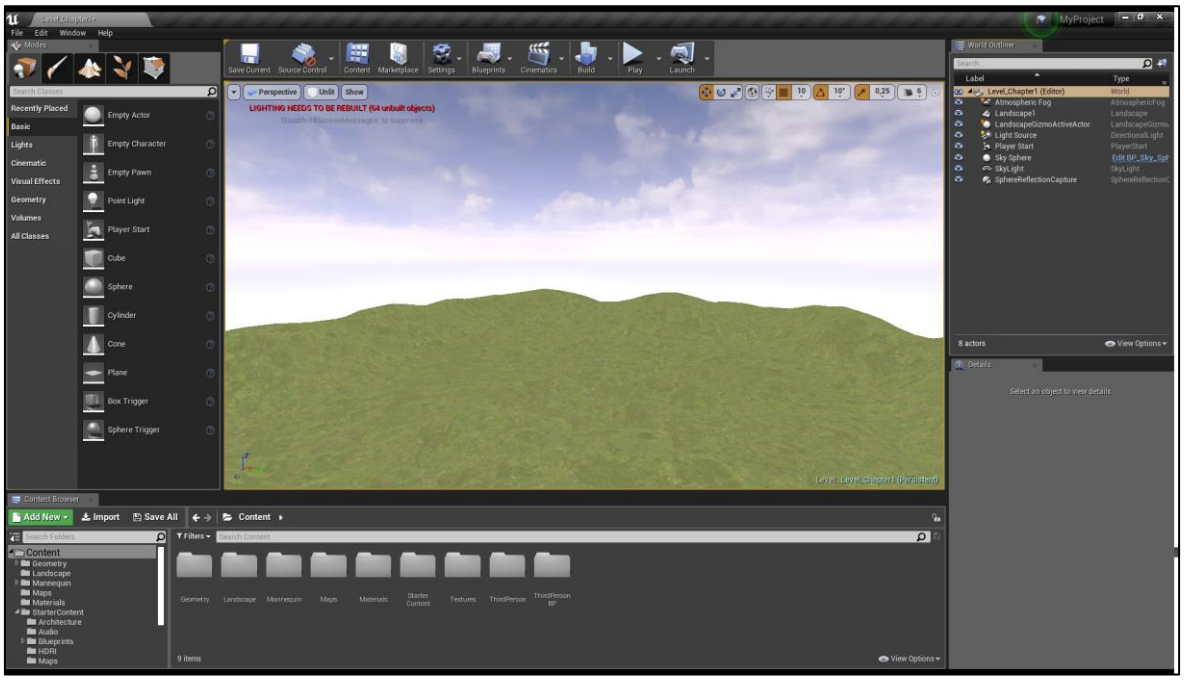
Sweeny hade implementerat en s.k. objektorienterad programmeringsstil med språket Pascal i ett tidigare spel, ZZT, som utkom 1991. Detta spel ses idag som startskottet till Unreal Engine 4. När Sony och Nintendo lanserade spelkonsolerna Playstation och Nintendo 64 på nittiotalet blev det tydligt att tredimensionella spel var framtiden inom spelvärlden. (IGN 2012)

Spelmotorn har utvecklats i omgångar. Dagens generation är den fjärde och den presenterades 2012. Spelmotorn är helt gratis att användas och uppdateras för alla som har eller skapar ett Epic-konto.

2.1.1. Användargränssnitt

Unreal Engine 4 har ett väldigt överskådligt användargränssnitt där användaren får tillgång till olika verktyg för spelskapandet.

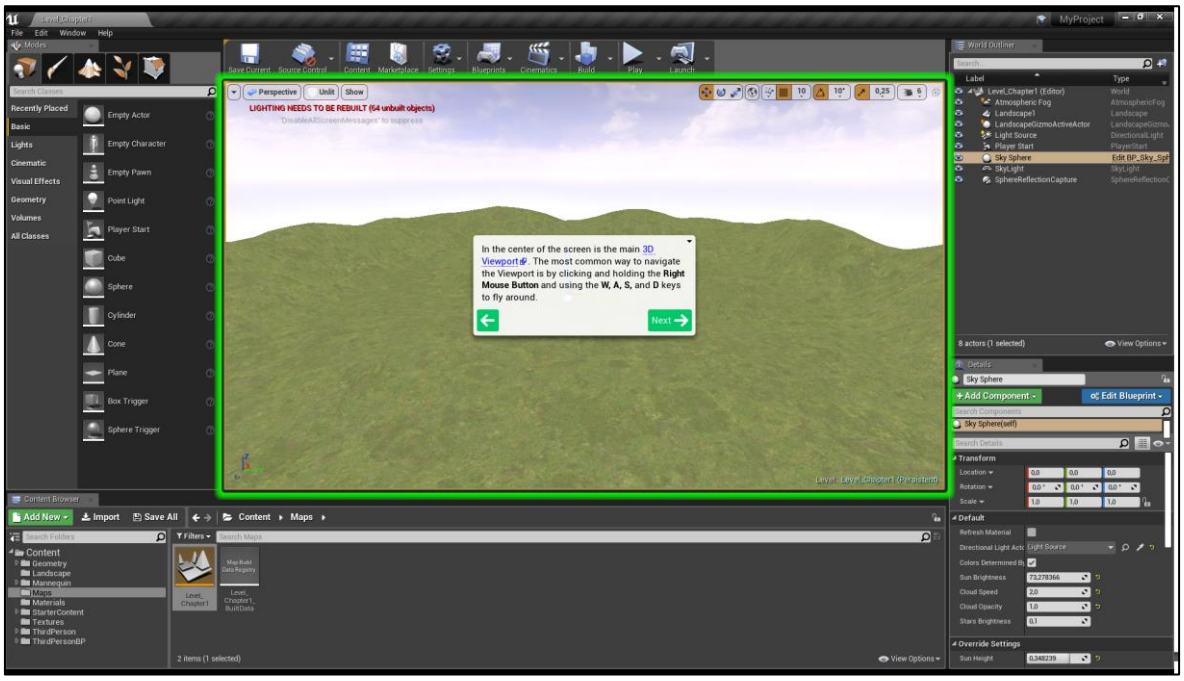
I figur 1 ser man hur hela gränssnittet presenteras vid skapandet av ett projekt. Med undantag för marken i bilden, som annars är otexturerade. Användaren har själv möjlighet att modifiera gränssnittet efter eget tycke så att alla fönster, menyer osv. kan placeras enligt eget önskemål.



Figur 1. Unreal Engine 4 användargränssnitt.

2.1.2. Viewport

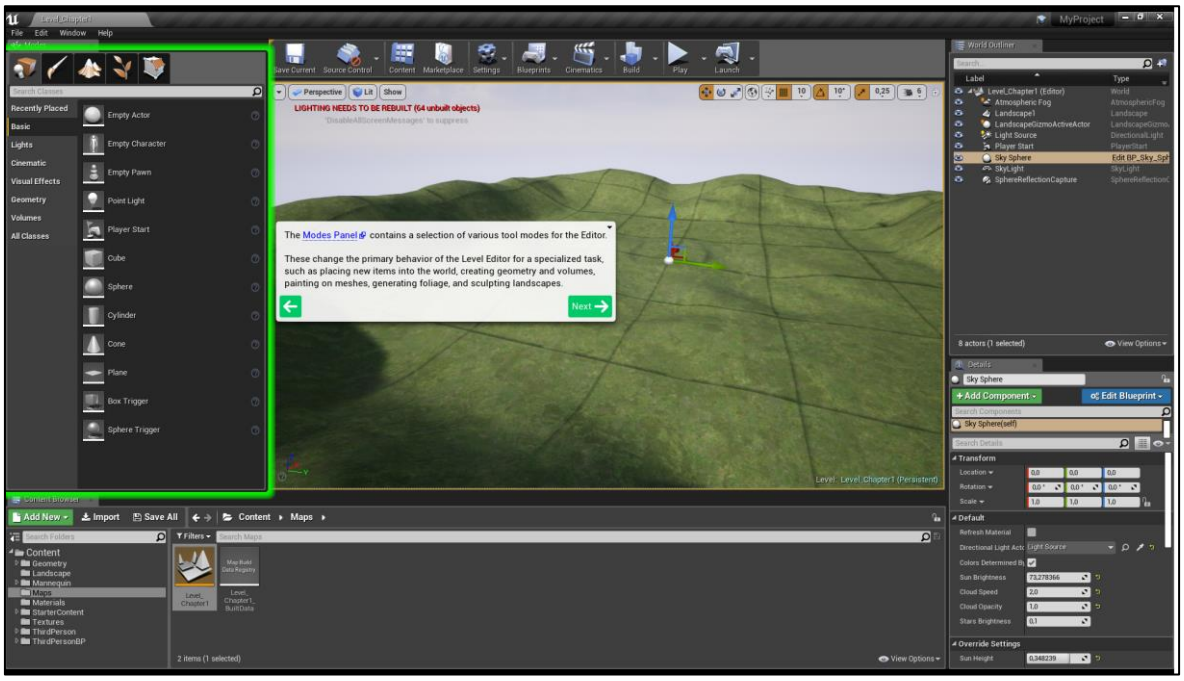
I mitten av Unreal Engine 4 finns Viewport. Denna ger en överblick över vad som skapats och vad som händer i pågående projekt, se figur 2. Man kan t.ex. se var spelaren börjar, var sol och himmel finns samt hur stor kartan är. Uppe i det högra hörnet finns det inställningar för styrkänsligheter och navigering runt i Viewport. I det vänstra hörnet hittar man presentationsalternativ, t.ex. hurudant ljus man vill ha i Viewport. För att styra kameran i Viewport så klickar man och håller in höger musklick och använder därefter W, A, S och D tangenterna för att flyga runt i pågående projekt eller karta. (Unreal Engine 2020)



Figur 2. Unreal Engine 4 Viewport.

2.1.3. Modes Panel

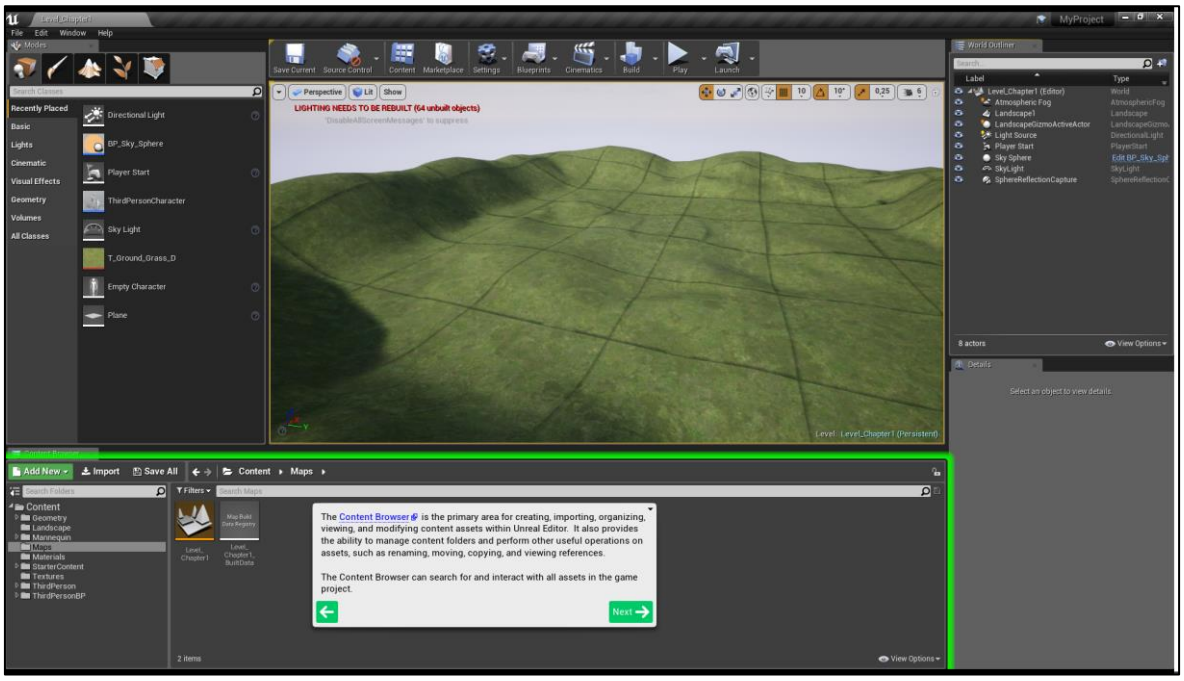
I Modes Panel hittar man olika slags färdiga verktyg och moduler för Unreal Engine 4, se figur 3. I panelen finns det också flera undermenyer där varje meny har hand om olika saker, t.ex. tilläggning av extra ljus, skapande av träd, buskar och stenar, redigering av marknivå och färgläggning av objekt osv. Dessa hittar man i respektive undermeny genom att klicka på ikonerna som motsvarar dessa. I vanlig ording hittar man dem från vänster: Place, Paint, Landscape, Foliage och Geometry Editing mode. (Epic Games u.å.)



Figur 3. Unreal Engine 4 Modes Panel.

2.1.4. Content Browser

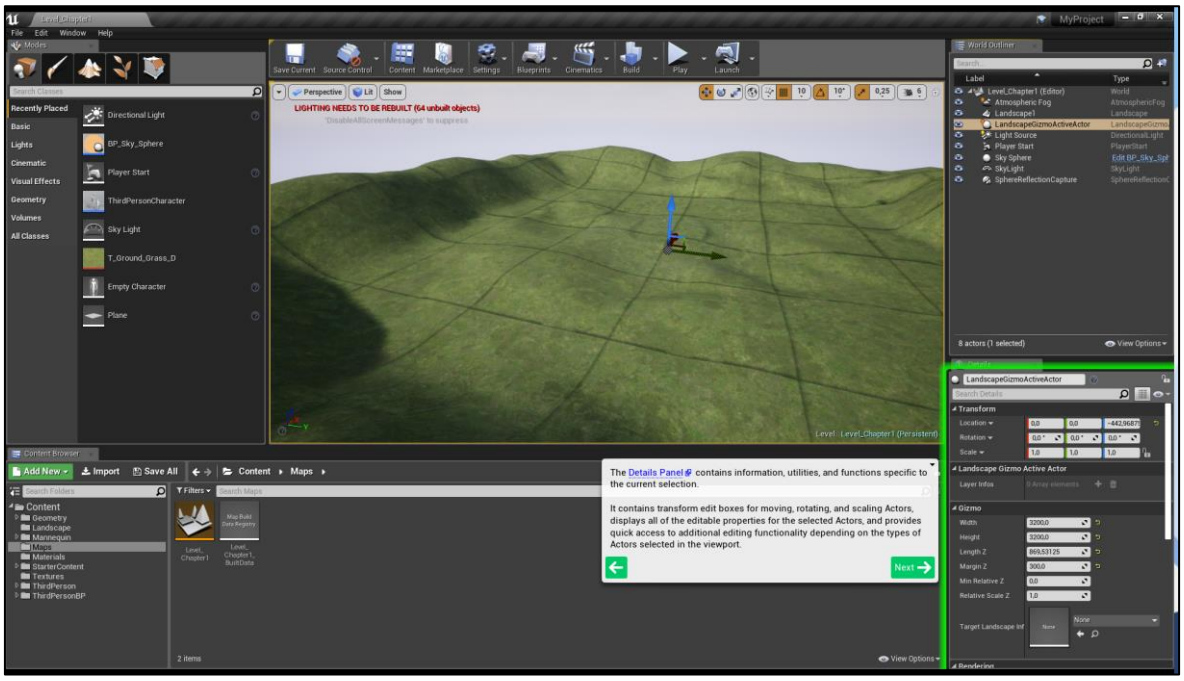
Content Browser är menyn där man skapar, hämtar, organiserar och visar innehåll som kan användas till projektet, se figur 4. Här finns också olika undermenyer med allt från texturer som kan användas för objekt i spelet till färdiga animationer. Det är också i Content Browser som användaren själv kan skapa, redigera och placera eget innehåll eller skapa mappar för dessa. (Unreal Engine 2020) Man kan alltså väljar att använda helt egna texturer och man kan skapa en egen mapp för att lättare kunna hålla reda på dessa.



Figur 4. Unreal Engine 4 Content Browser.

2.1.5. Details Panel

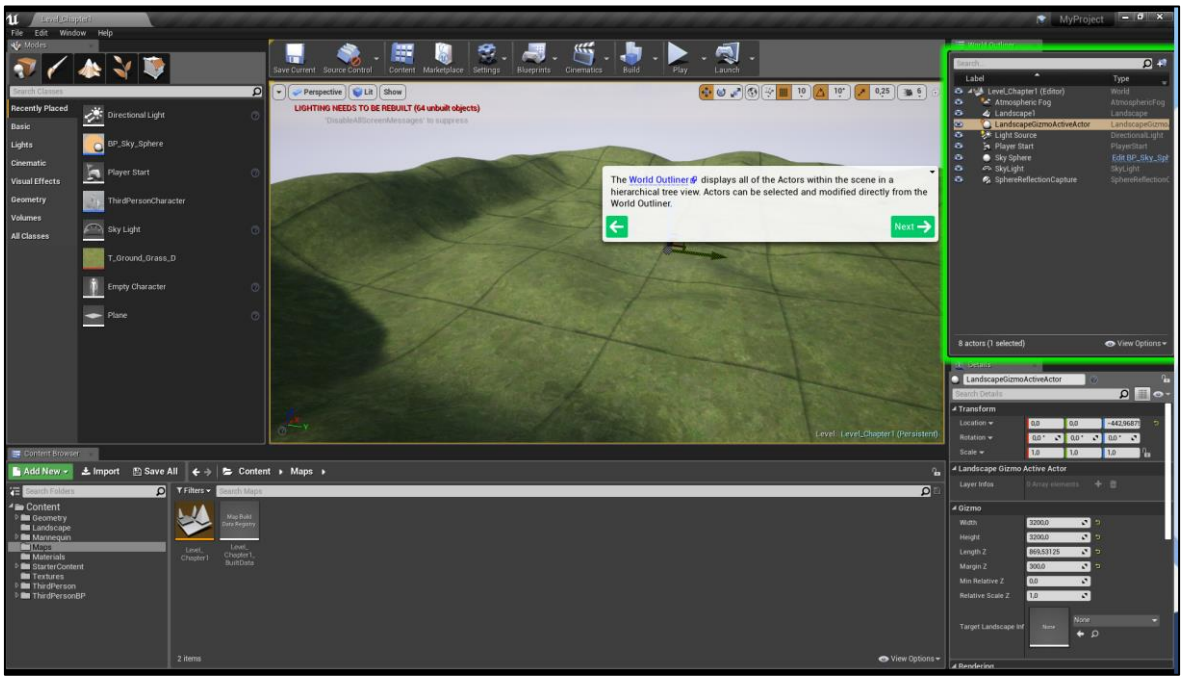
Under Details Panel hittar man information och får tillgång till snabb funktionalitet för markerat område eller objekt. Från denna panel kan man ändra storlek, flytta runt och rotera olika objekt som lagts in i pågående projekt, se figur 5. Man kan exempelvis skala om, dvs. ändra storlek på marken, ändra solens värme och ljusstyrka, sken och position med mera. (Unreal Engine 2020) Funktionaliteten skiljer sig förstås från objekt till objekt beroende på kategori och funktion.



Figur 5. Unreal Engine 4 Details Panel.

2.1.6. World Outliner

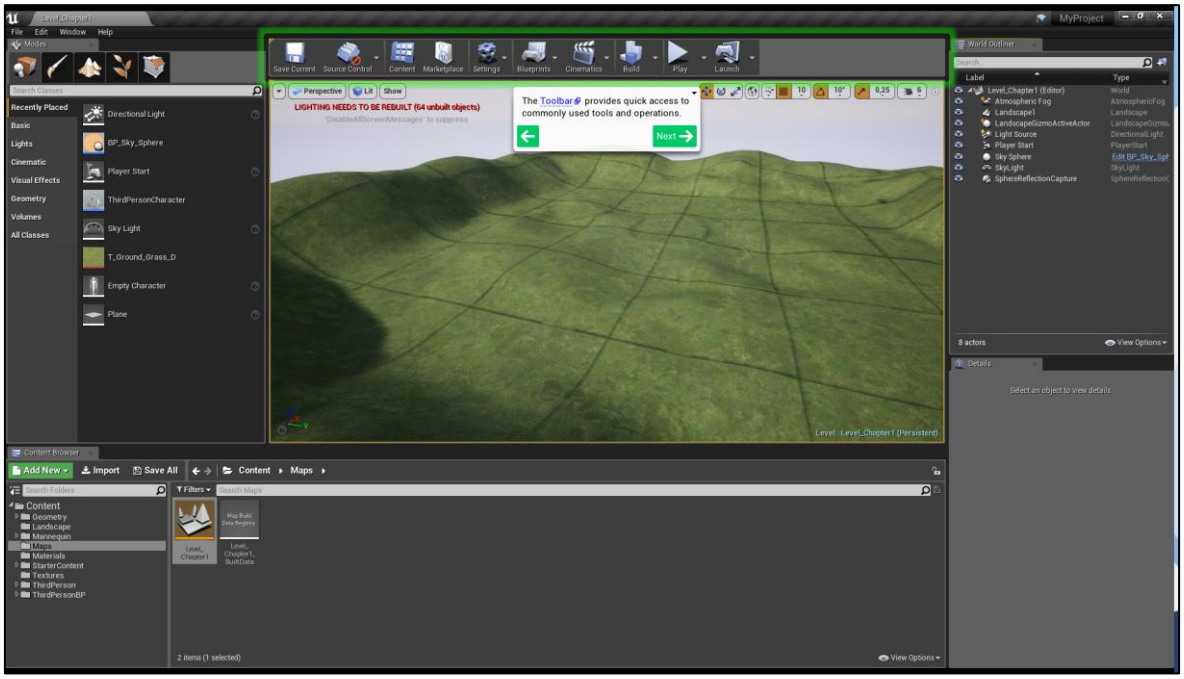
I denna meny presenteras alla innestående objekt i pågående projekt. Härifrån kommer användaren snabbt och enkelt åt valfritt objekt för modifiering. Man kan direkt ändra inställning för synlighet, placering och dylikt samt dölja objektet i Viewport. Man får också möjlighet att anpassa hierarkin enligt egna önskemål via kolumninställningspilen uppe i det högra hörnet, se figur 6.



Figur 6. Unreal Engine 4 World Outliner.

2.1.7. Toolbar

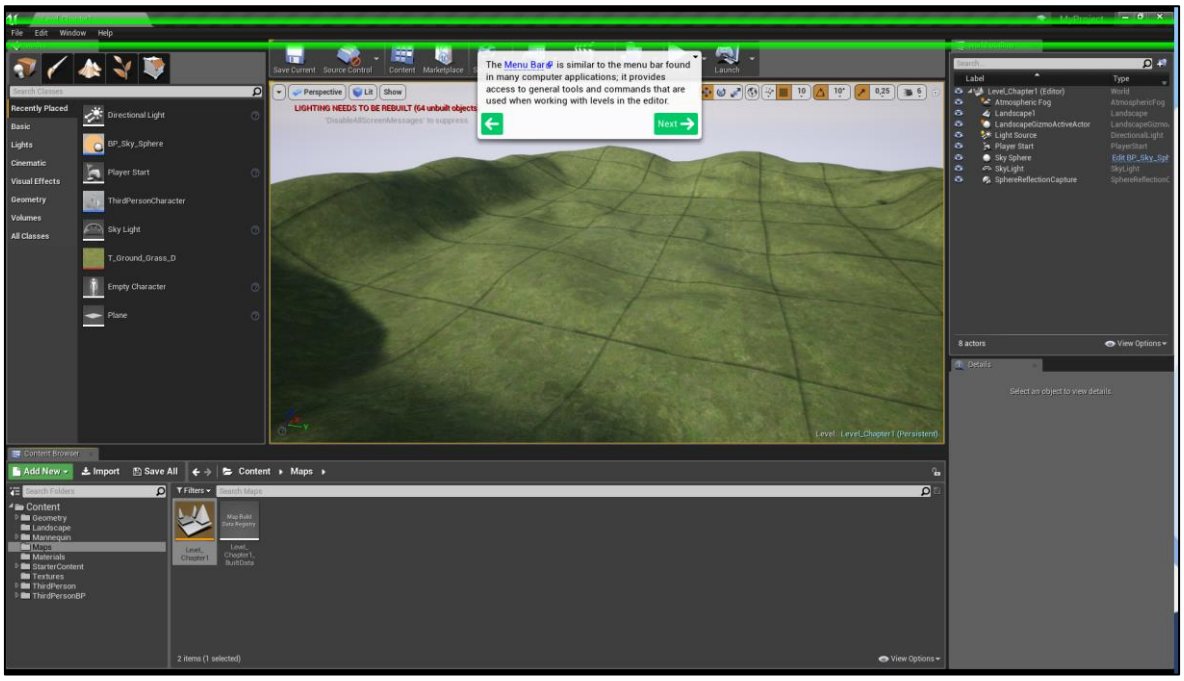
Via denna meny får användaren snabbt tillgång till vanliga projektverktyg och alternativ. Man kan t.ex. spara projektet, skapa eller visa alternativ för spelfilmer, komma åt ljudfunktionalitet samt generera, öppna eller spela spelet på olika sätt, se figur 7.



Figur 7. Unreal Engine 4 Toolbar.

2.1.8. Menu Bar

I Menu Bar hittar användaren standard applikationfunktionallitet för projektet och själva spelmotorn. Menyerna med olika undermenyer som presenteras är File, Edit, Window och Help, se figur 8. Dessa typer av menyer är väldigt vanliga bland windowsapplikationer och program och de flesta datoranvändarna är bekanta med dylika menyer.



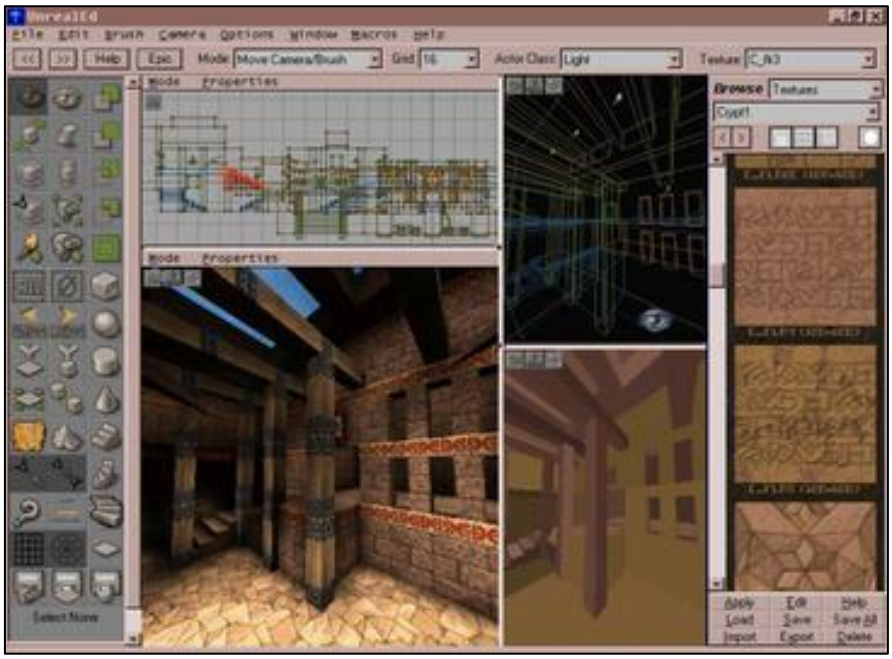
Figur 8. Unreal Engine 4 Menu Bar.

3 Unreal Engine historik

Som tidigare nämnt är Unreal Engine 4 uppbyggd med programmeringsspråket C++. Största delen av källkoden till spelmotorn finns också på webbplatsen GitHub, vilket möjliggör för de mer avancerade användarna att göra helt egna modifikationer och förändringar på spelmotorn. Spelmotorn har utkommit i olika versioner genom åren. Epic Games grundaren Tim Sweeny började programmera spelmotorn 1995 med ett framtida spel i åtanke. Spelet som byggdes med motorn fick namnet Unreal. Detta gav namnet till spelmotorn.

3.1 Version 1

Den första versionen debuterade offentligt som Unreal år 1998 efter tre års utveckling. Spelmotorn hade funktioner som Collision Detection, färgat ljus och texturfiltrering. Det fanns också en nivåredigerare vid namn UnrealEd, (Maximum PC 2009) som hade stöd för geometribyggande i realtid, se figur 9.



Figur 9. Första versionen av Unreal Engine från år 1998.

3.2 Version 2

Ett år efter att spelet Unreal ges ut började Tim Sweeny förbereda en andra version av spelmotorn. (Nytimes 1999) År 2002 presenterades spelmotorn i samband med spelet *America's Army*. Spelet utvecklades dock inte av Epic Games utan av USA:s armé för att locka folk till marinkåren. (Nytimes 2002)

Utöver generella förbättringar av tidigare funktioner hade den nya versionen ett helt nytt genereringssystem som klarade av att generera upp till 100 gånger större detaljrikedom i nivåerna. Det fanns också olika verktyg för filmer i spel och plug-ins som andra program som t.ex. 3D Studio Max kunde använda.

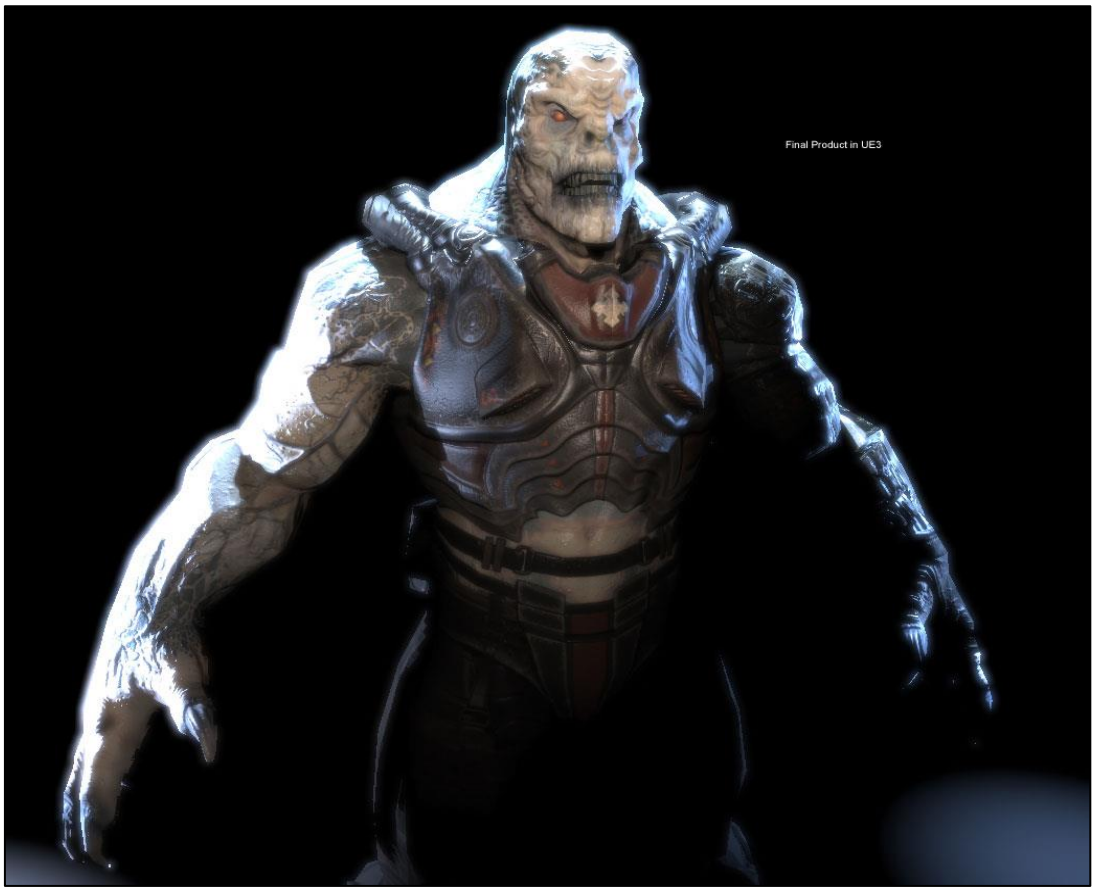
3.3 Version 3

Den tredje versionen av Unreal Engine presenterades för första gången för allmänheten år 2004, efter ca 18 månaders utveckling, i form av olika skärmbilder som demonstrerade spelmotorns möjligheter, se figurerna 10 och 11. Själva lanseringen skulle dröja fram till år 2006 då det första spelet *Gears of War*, gavs ut till konsolen Xbox 360.

I Unreal Engine 3 omgjordes generering, fysik, ljud samt verktyg och fick enligt Tim Sweeny betydligt bättre utseende och prestanda. Även ljusutreckningen gjordes om från att tidigare räknat per triangel till per pixel. (Maximum PC 2004). Från början var det tänkt att den tredje versionen endast skulle stöda spelplatformerna Windows, Playstation 3 och Xbox 360, men senare tillkom också iOS och Android (Develop Online 2010).

Spelmotorn fick också många uppdateringar under kommande åren i form av dynamikförbättringar, stöd för iOS och Steam samt stöd för 11 generationens Flash Player. År 2013 presenterades också ett HTML5 demospel vid namn *Epic Citadel* som togs fram tillsammans med Mozilla. (Gamasutra 2013)

Figur 10. Skärmbild inför lansering av Unreal Engine 3.



Figur 11. Skärmbild inför lansering av Unreal Engine 3.

3.3.1. Unreal Development Kit

Alla fick ta del av- och använda Unreal Engine 3, men rättigheterna gavs endast för beviljade användare av spelmotorn. Övriga användare fick varken distribuera eller sälja produkter som de skapade. Detta ställde till med en del problem för bl.a. spelmodifierare som skapade egna modifikationer för olika spel.

För att motverka distributionsproblemen presenterade Epic Games år 2009 en gratisversion av spelmotorn vid namn UDK (Unreal Development Kit) som var gratis att använda för alla. (IGN 2009)

3.4 Version 4

År 2005 berättade Epic Games vice vd Mark Rein att man jobbat på en fjärde verion av Unreal Engine i ungefär 2 års tid. Vd:n Tim Sweeney berättade i en intervju att den fjärde versionen skulle användas till de uppföljande plattformerna av konsolerna Xbox 360 och

Playstation 3. Unreal Engine 4 presenterades år 2012 i form av en demonstreringsvideo vid Game Developers Conference. (Wired 2012)

År 2014 lanserades den fjärde versionen. Priset på månadsprenumerationen var 19 dollar. Ytterligare gick fem procent av användarnas försäljningsintäkter på över 3000 dollar per kvarttal till Epic Games. Detta kom senare att ändras år 2015 då månadsavgiften slopades och spelmotorn fick användas gratis av alla skolor och universitet.

De största nyheterna med spelmotorn funktionalitetsmässigt var ljussättning i realtid, koduppdatering i realtid samt ett nyutvecklat visuellt programmeringssystem vid namn Blueprint. Detta system möjliggjorde snabb spelutveckling utan kunskaper inom C++ och systemet var en vidareutveckling från den tredje versionens variant som hade namnet Kismet. (Rock Paper Shotgun 2012)

Det första spelet som presenterades med spelmotorns fjärde version var *Daylight* som presenterades år 2014. Spelet utvecklades av Zombie Studios som tidigt fick tillgång till spelmotorn.

4 Epic Games – företaget bakom Unreal Engine

Potomac Computer Systems grundades år 1991 av Tim Sweeney som på den tiden studerade vid Marylands universitet i USA. Namnet Potomac Computer Systems härstammar från staden Potomac där Sweeney studerade. Hans föräldrar bodde också i samma stad och Sweeneys dator, en PC från 1989 fanns hemma hos dem. Han använde sin dator både för extrajobb och på fritiden.

Sweeney hade i unga år intresserat sig för programmering, främst av BASIC-språket på dåvarande datorn Apple II. När PC köptes år 1989 experimenterade Sweeney med de olika textredigerarna som medföljde datorn. Han berättade att han inte fattade tycke för någon av dem så han började senare programmera en egen. (Gamasutra 2009)

Tanken med Potomac Computer Systems var från början att erbjuda datakonsulteringstjänster, t.ex. skapande av små databaser och dylikt. Men enligt Sweeney var arbetsbördan ganska stor och som han själv formulerade sig ”didn’t get anywhere with it”. (”Jag kom ingenstans med det”). Sweeney beslöt ändå att använda företagsnamnet när han gav ut sitt första spel, *ZZT*, år 1991. Allt annat kring företaget var i princip färdigställt.

Spelet *ZZT* kom att bli startskottet på dagens Epic Games. Spelet sålde betydligt bättre än vad Sweeney hade förväntat sig. Han började fundera på att konvertera Potomac Computer Systems till ett företag som utvecklade dataspel.

ZZT-framgången bidrog till att Sweeney utvecklade ett nytt spel vid namn *Jill of the Jungle* som släpptes följande år 1992.

Samtidigt tyckte Sweeney att företaget behövde ett bättre namn. Han beslöt att döpa om företaget till Epic MegaGames, se figur 12. I en intervju berättade Sweeney att namnet var ett trick för att få företaget att verka större än vad det egentligen var utåtsett. (Gamasutra 2009)

Figur 12. Logotypen för namnbytet till Epic MegaGames år 1992.

I samband med namnbytet började Sweeney leta efter en affärspartner. Mark Rein, som senare kom att bli vice VD på Epic Games, hade då nyligen slutat från sitt dåvarande jobb hos företaget id Software. Detta företag var en av Sweeneys största konkurrenter. Rein flyttade till Toronto i Kanada och började i huvudsak sköta om marknadsföring och försäljning samtidigt som han slöt en del publiceringsavtal. Enligt Sweeney bidrog Rein enormt mycket för företagets tillväxt.

År 1996 skapade Epic MegaGames spelet *Fire Fight* som följande år 1997 gavs ut av företaget Electronic Arts och vidareutvecklades av Polish studio Chaos Works. Under den här tiden hade Epic MegaGames 50 anställda, över hela världen.

Samtidigt började Tim Sweeney programmera spelmotorn Unreal. Den färdigställdes år 1998 tillsammans med spelet under samma namn. Företaget började också marknadsföra spelmotorn åt andra speltillverkare.

År 1999 flyttade huvudkontoret från Toronto till Cary i delstaten North Carolina i USA. Samtidigt tog man bort ”Mega”- delen från företagsnamnet, se figur 13.

Samma år släppte företaget, nu under namnet Epic Games – spelet Unreal Tournament, som var en uppföljare till *Unreal*.

Figur 13. Den förnyade logotypen för namnbytet till Epic Games år 1999.

I takt med att industrin utvecklades blev det tydligt för Epic Games att det var svårt att livnära sig på spel där man spelar ensam, s.k. singleplayer spel. Delvis p.g.a. den ökande piratkopieringen. Dessa spel hade fungerat som Epic Games affärsmodell under många år. Tim Sweeney och företaget beslöt år 2006 att ändra inriktning. Nu skulle de utveckla spel för spelkonsoler. Sweeney menar att beslutet var början på tredje generationens Epic Games. Företaget presenterade spelet *Gears of War* för Xbox 360 samma år och drog in ca 100 miljoner dollar i intäkter med en budget på runt 12 miljoner.

År 2007 släpptes spelet *Unreal Tournament 3* och samtidigt blev företaget största ägare i ett annat företag, People Can Fly. Följande år 2008 släpptes *Gears of War*-uppföljaren *Gears of War 2.* Det sålde över 3 miljoner kopior enbart under den första månaden.

Samma år, år 2007, började en lång rättstvist mellan Epic Games och företaget Silicon Knights. Silicon Knights hävdade att Unreal Engine 3 inte fungerade som den skulle och att dokumentationen över spelmotorn var bristfällig. Epic Games blev också anklagat för att tillbakahålla nödvändiga förbättringar för att på så sätt kunna få mera pengar till egna speltitlar.

Epic Games motsatte sig detta och hävdade att *Gears of War* färdigställdes med Unreal Engine 3 och att Silicon Knights visste om de begränsningar som fanns. Epic hävdade också att Silicon Knights hade använt sig av en otillåten metod då det använt Unreal Engines kod i företagets egen spelmotor. Tvisten kom först att avgöras till Epic Games fördel år 2012.

Samma år, år 2012, ingick företaget ett avtal med kinesiska företaget Tencent som blev majoritetsägare i Epic Games. Tim Sweeney menade att beslutet fattades då förändring inom industrin var i antågande. Syftet med avtalet var helt enkelt att företaget skulle bli mer

närvarande på markanden. Enligt Sweeney innebar Tencent-avtalet en början till den fjärde generationen av Epic Games.

Året innan Tencent-avtalet, år 2011, presenterade Epic Games spelet *Fortnite*, som inom några år kom att bli världens största intäktsgenererande spel. När detta examensarbete skrevs var *Fortnite* fortsättningsvis störst på marknaden. År 2015 förnyades företagets logo till dagens version, se figur 14.

Figur 14. Epic Games logotyp från år 2015.

4.1 Affärsstrategin för Unreal Engine

Som tidigare nämnt har Epic Games under åren haft en unik affärsstrategi för Unreal Engine. När spelet *Unreal* gavs ut år 1998 började man erbjuda licenser för att få använda spelmotorn Unreal Engine för egna produkter. En licenserad kopia av Unreal Engine för skapandet av 16 projekt kostade upp till 350 000 dollar. (Nytimes 1999) Epic tog dessutom upp till 7 procent av de totala intäkterna från projekt som skapades med spelmotorn.

Epic Games har inte presenterat några officiella priser för Unreal Engine 2 och 3, men på Unreal Engines officiella forum har detta diskuterats att man förr var tvungen att anmäla intresse för licenserna och att informationen således inte var tillgänglig för alla.

På samma forum har det också diskuterats att priserna på Unreal Engine 2 började från 700 000 och gick upp till 1 miljon dollar plus procentuella avgifter på upp till 7 procent, beroende på vilken omfattning man önskade av med spelmotorn. (Unreal Engine Forums u.å.)

När den fjärde generationen av Unreal Engine presenterades år 2014 introducerades en ny modell. De högsta avgifterna ersattes med en månadsavgift på 19 dollar för användning samt ytterligare 5 procent av intäkterna av spel gjorda med spelmotorn.

Senare under året 2014 lanserade Epic Games spelmotorn gratis för alla skolor och universitet i hela världen samt personliga licenser för studerande inom spelutveckling eller motsvarande områden.

Följande år, år 2015, under Game Developers Conference uppdaterades affärsmodellen för Unreal Engine 4. Månadsavgiften på 19 dollar för spelmotorn slopades och blev gratis för alla att använda och uppdatera. (Populartimelines u.å.)

Mindre förändringar gjordes också på den procentuella intäktsmodellen så att istället för 5 procent av totalintäkter för spel gjorda med spelmotorn debiteras endast 5 procent per kvarttal om spelet säljer mer än 3000 dollar. Dessutom upprättades en marknadsföringsplatform för Unreal Engine där användarna får sälja spel och övrigt innehåll ex. modifikationer för spelen. (Unreal Engine 2020)

4.2 Epic Games Store

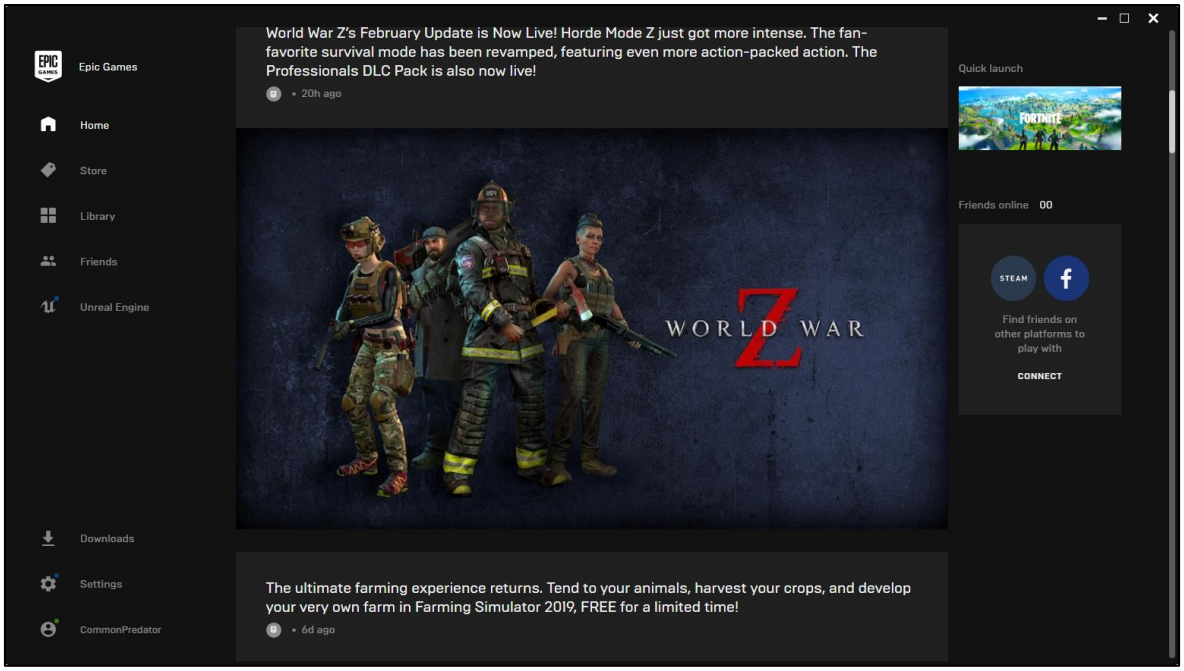
Epic Games Store är företagets plattform för marknadsföring av spel och tillägg. Plattformen presenterades i december 2018 både som webbtjänst och klientversion. Klienten går att installera på valfri enhet som stöds t.ex på. PC, konsol eller mobil, se figur 15. Via plattformen får användare uppdateringar, nyheter samt tillgång till spel och program som erbjuds.

Plattformen skapades efter succén med spelet Fortnite. Enligt Tim Sweeney var skapandet av en helt egen tjänst nödvändigt då han ansåg att den marknadsdominerande huvudkonkurenten Valve med tjänsten Steam tog alltför stora avgifter av utvecklare. Sweeney menade att Valve skulle klara sig med en intäktsmarginal på 8 procent och fortfarande gå med vinst istället för den dåvarande modellen på 30 procent.

Via egen plattform tar Epic Games tolv procent av totalintäkterna, undantag görs ifall produkten utvecklats med Unreal Engine och distribueras på Epic Games Store. I sådana fall uteblir 5 - 7 procent och Epic tar i sådana fall endast 5 - 7 procent som ingår i Unreal Engine-avtalet beroende på intäktsresultat. (Epic Games 2020)

För att marknadsföra sin nya plattform erbjöd Epic Games utvecklare att exklusivt lansera sina nya spel på Epic Games Store i utbyte mot ersättning. I fall förlusterna blir betydligt större än förväntat lovade Epic att betala mellanskilnaden åt utvecklaren. (PC Gamer 2018)

År 2019 gick Epic Games ut med en kampanj som erbjöd ett gratis spel två gånger per månad. Detta utökades senare i juni samma år till ett gratis spel i veckan. Spelen som erbjöds var fullständiga kopior med innehåll ämnade för spelare över sexton år, men ersattes av ett annat spel ifall individen var yngre. Epic Games har senare meddelat att de tänker fortsätta med programmet år 2020. (Epic Games 2020)



Figur 15. Epic Games Store i klientversion.

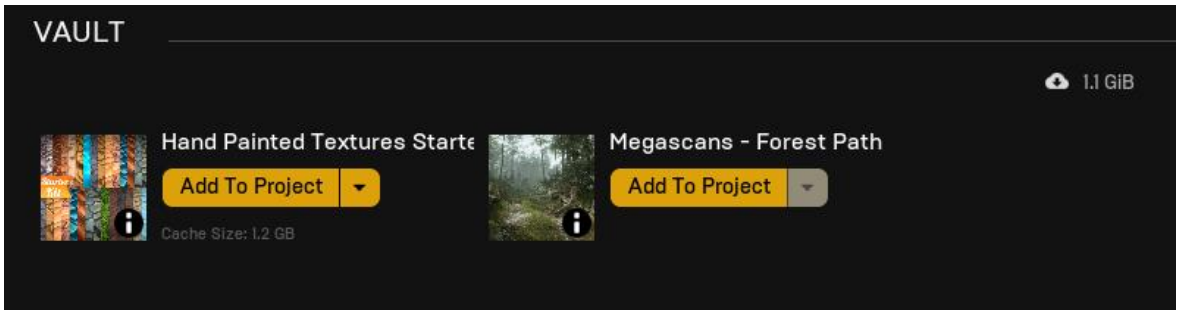
4.2.1. Unreal Engine Marketplace

I plattformen finns också en flik för Unreal Engine. Här hittar man förutom senaste nytt och kommande uppdateringar också guider och uppvisningar av innehåll som andra användare har skapat.

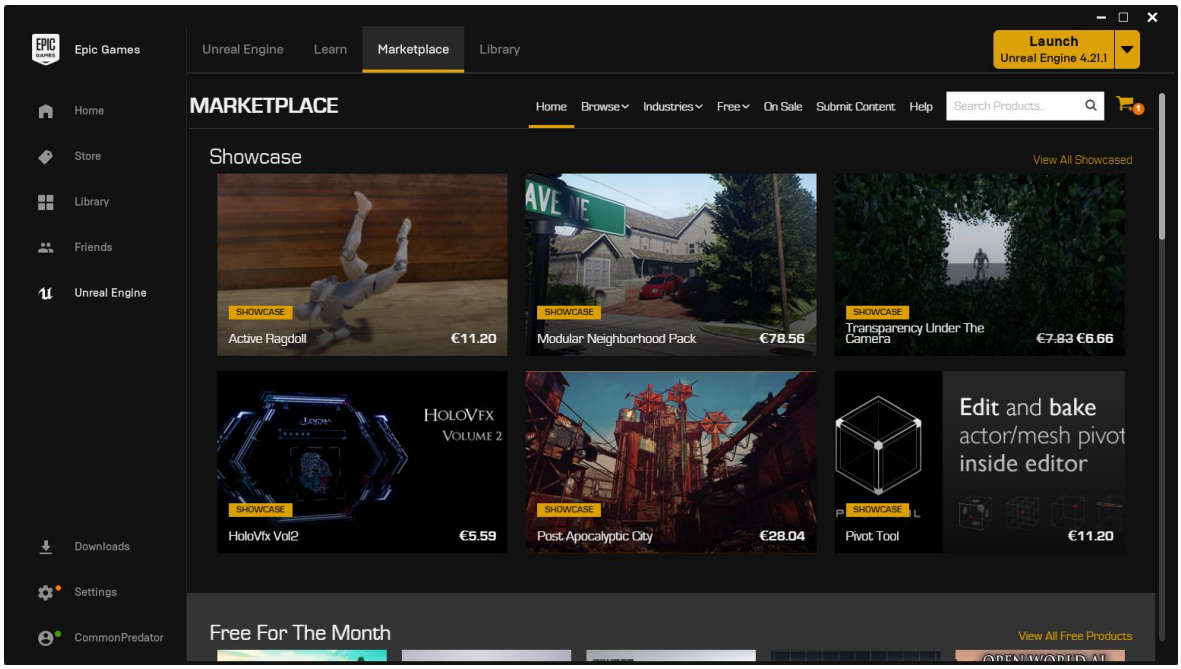
Via Unreal Engine Marketplace får användare tillgång till diverse tillägg som kan anändas i egna projekt i spelmotorn Unreal Engine 4, se figur 17. Tilläggen som erbjuds varierar i pris, form och storlek. Tilläggen kan t.ex. vara bakgrundsmusik, ljudeffekter, karaktärmodeller eller färdigbyggda kartor.

Användare kan köpa och förvara innehåll på klientens bibliotek i form av molnförvaring. Det innebär att utvalt tillägg endast förvaras på den egna användarens klient och sen sparas det som en s.k. cachefil på datorn. Det innebär att tilläggsinformationen lagras i form av snabba data som går snabbt att läsa in när det behövs. (Unreal Engine 2020)

För att kunna använda tillägg måste man ha Unreal Engine 4 installerad på sin enhet samt ha den version av spelmotorn som tilläggen stöder för att använda önskade tillägg. Kraven mellan tilläggen varierar. När alla krav uppfylls är det bara att lägga till innehåll i önskat projekt genom att klicka på Add To Project eller lägg till i projekt och välja önskat projekt efteråt, se figur 16.



Figur 16. Unreal Engine Marketplace tilläggsibliotek.

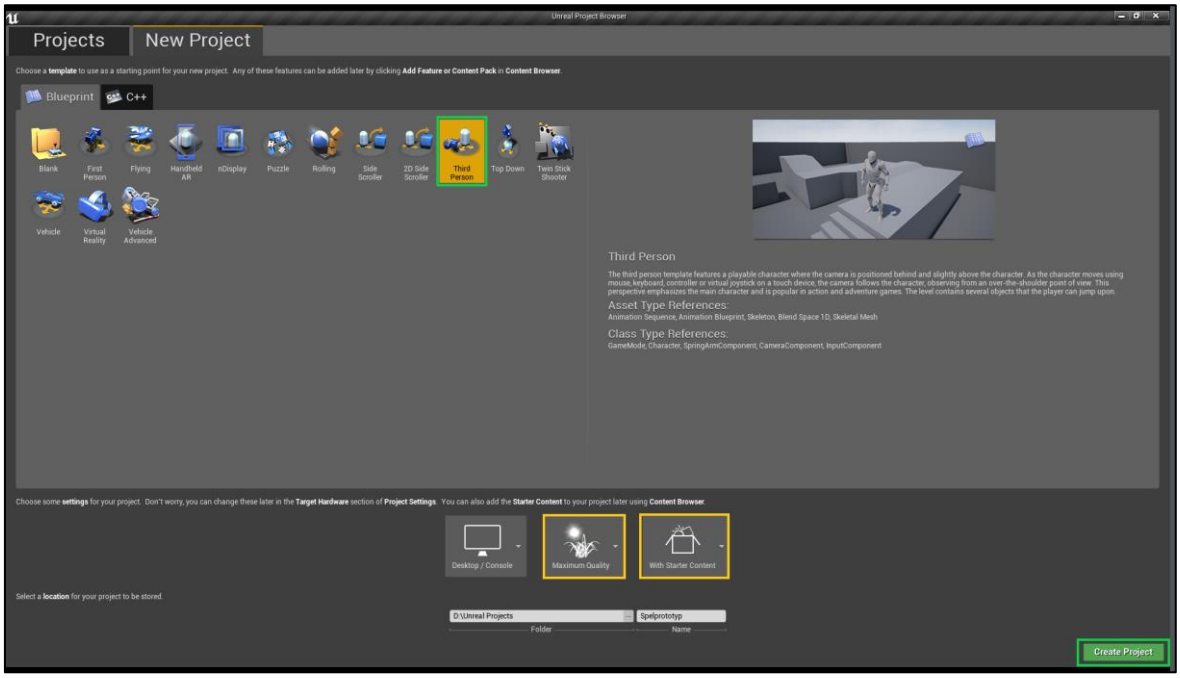


Figur 17. Unreal Engine Marketplace.

5 Spelprototypens konstruktion

I Unreal Engine 4 kan man, som jag tidigare nämnt, välja att skapa sin produkt med visuell programmering eller med vanlig programmering med språket C++. Jag valde att bygga min prototyp med visuell programmering d.v.s med Blueprint-systemet. När man väljer Blueprint får man olika alternativ att välja mellan. Alternativen ger användaren möjlighet att välja vilken typ av spel man vill skapa.

Är man van vid datorspel så vet man att det finns alla slags typer av spel och med olika utseenden. Det finns t.ex. speltyper i ”Super Mario” -stil där kameran aktivt följer spelaren från sidan och oftast utan möjlighet att styra kameran framåt eller bakåt. För att spelaren ska kunna se framåt i nivån måste denne ta sig framåt i den. Dessa typer av spel brukar kallas för Side view, Side-scroller eller sido-vy. (TVTropes u.å.)



Figur 18. Skapandet av nytt projekt i Unreal Engine 4.

5.1 Kategori och projektinställningar vid skapandet

Mitt val av spelprototyp blev Third Person, eller ”tredje person”. I spel i tredje person följer kameran oftast spelarens figur bakifrån och ibland lite snett ovanifrån beroende på situation eller speltyp. Oftast ges spelaren full kontroll över kameran även om detta kan variera från spel till spel. (Cellularnews 2019)

Man får också välja till vilken platform man önskar skapa spelet för. Detta alternativ lämnade jag orört på Desktop / Console, eller dator och konsol då jag anser att ta fram denna prototyp för dator.

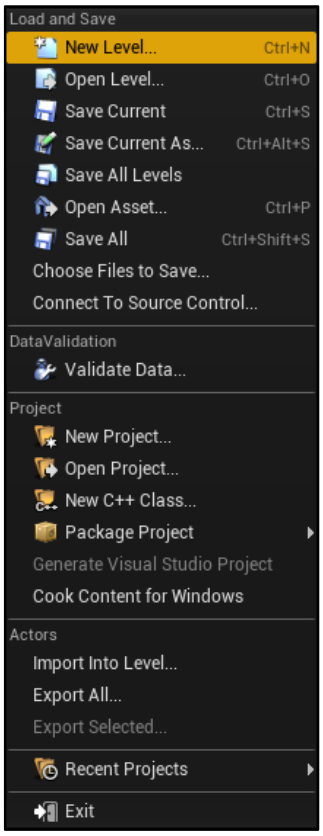
Två kvalitetsalternativ ges också i den mittersta rutan, Maximum Quality, eller maximal kvalitet och Scalable 2D or 3D eller skalbar 2D eller 3D. Detta kvalitetsalternativ går också bra att lämna orört. Jag valde alternativet maximal kvalitet då jag konstaterat att bara man har en någonlunda tidsenlig dator så skapar maximal kvalitet inga anmärkningsvärda problem.

I den tredje rutan får man välja ifall man vill eller inte vill inkludera färdiga mallar för olika tillägg, s.k. Starter Content, eller nybörjarinnehåll. Jag valde att ta med alternativet eftersom det skapar en del användbara saker åt en färdigt i spelmotorn. Det kan t.ex. vara en del texturer och former som man kan använda till själva skapandet.

Jag skapade mitt arbetsprojekt genom att klicka på Create Project i den nedre delen av det högra hörnet av skärmen, se figur 18.

5.1.1. Ny nivå och val av nivåtyp

När projektet skapats möts man av ett färdigskapat område. Jag väljer att skapa ett helt nytt område i form av en ny nivå genom att klicka uppe i skärmens vänstra hörn och välja ny nivå, se figur 19. Nu skapas en ny nivå med mindre inläst innehåll och jag får möjlighet att själv välja vad jag placerar in i nivån.

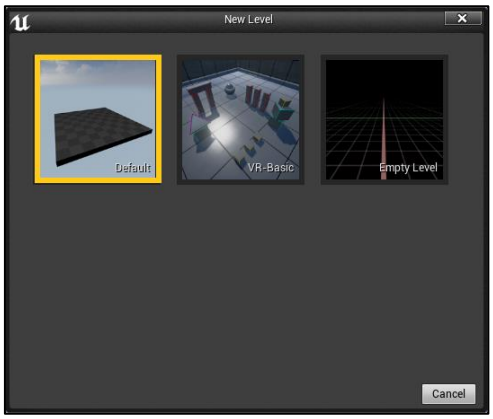


Figur 19. Skapandet av ny nivå i Unreal Engine 4.

Här får man tre olika alternativ beroende på vilken typ av nivå man vill skapa. Man kan välja mellan Default, VR-Basic eller Empty Level. Normal, Grundläggande virtuell eller Tom nivå, se figur 20.

Jag valde alternativet Normal eftersom man därmed får himmel, sol, spelfigur samt ett konfigureringsbart alternativ där spelfiguren börjar vid namn Player Start.

Dessa skulle förstås kunna placeras in efteråt men jag upplever att hela processen för skapandet löper smidigare då dessa redan är konfigurerade. Dessutom får man ett bättre intryck över hur slutprodukten kan komma att se ut. Jag ser alltså en färdigt konfigurerad spelfigur och en startpunkt som en fördel.



Figur 20. Alternativ för ny nivå i Unreal Engine 4.

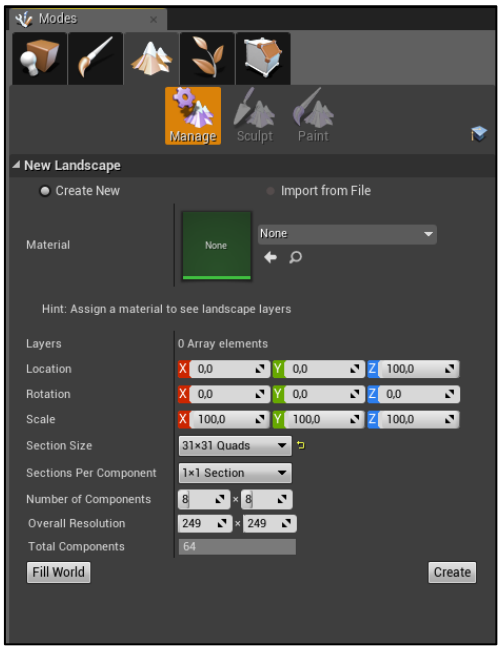
5.2 Generering av landskap

Väl inne på den nya nivån börjar jag med framtagningen av min prototyp. Först vill jag skapa marken till nivån. För att göra detta går jag in i Modes panelen, till vänster på skärmen, där man hittar olika alternativ.

Följande väljer jag ikonen för Landscape, eller landskap. Då får man fram olika alternativ för generering av mark. Man kan t.ex. ställa in hur stort område som ska genereras, hur långt, smalt eller brett det ska vara, upplösning och färdiga texturer m.m. Alternativt kan man också importera verkliga landskap via en annan metod som kallas för Heightmap, eller karthöjd, men detta alternativ kommer jag inte att använda mig av i mitt arbete.

För enkelhetens skull infogar jag inte någon färdig textur och ändrar heller inga alternativ förutom alternativet Section Size, alltså storlek på området från 63 x 63 till 31 x 31 Quads, eller fyrkanter. Detta förminskar det totala antalet genererade fyrkanter i nivån, vilket också innebär att området på nivån blir mindre, men mera detaljerat. (Unreal Engine 2020) Texturerna kommer jag att välja senare i skapandeprocessen.

Slutlingen klickar jag på Create, eller skapa, nere i skärmens högra hörn. Marken genereras då efter valda inställningar, se figur 21. Eftersom jag valde att inte lägga till en färdig textur för marken så genereras den i odefinierad textur med grått utseende.



Figur 21. Alternativ för generering av nytt landskap i Unreal Engine 4.

5.2.1. Formgivning av landskap

Nästa steg i processen är att börja forma nivån. För att kunna göra detta klickar jag på alternativet Sculpt, eller skulptera, inne i Landscape-menyn. Här får man fram alternativ för utformning av landskap. Några exempel på alternativ är val för markhöjning och sänkning, tillplattande och utjämning.

Först använder jag mig av det vanligaste alternativet, Sculpt och höjer upp kanterna på nivån. Detta får nivån att verka större än den egentligen är samt att spelaren inte ges direkt möjlighet att se förbi nivåns gränser. Jag justerar också storleken på pensel för att effektivera processen. Efter att jag höjt upp landskapet använder jag alternativet Smooth, eller förmjukning för att ta bort störande kantdykningar och liknande fenomen.

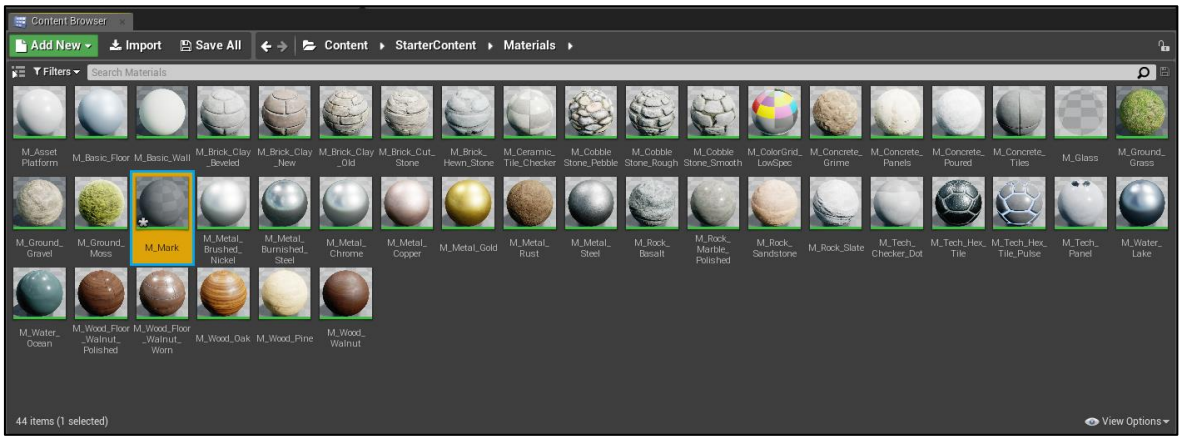
5.3 Skapandet av nytt material

Följande steg i mitt skapande är att texturera landskapet. För att texturera marken kommer jag att skapa ett Material, eller material. Ett Material i Unreal Engine 4 är en kombination av olika noder som beskriver materialets egenskaper. Man får själv bestämma vad materialet kommer att innehålla. (Unreal Engine 2020)

Fördelen med att skapa eller använda material för texturering är att valmöjligheterna blir flera. Exempel på egenskaper som går att definiera i ett material är reflektion, skala och skärpa. (Unreal Engine 2020)

I Content browser klickar jag in mig i mappen StarterContent och därefter in i undermappen Materials. I denna mapp presenteras alla färdiggjorda material som följer med i Starter Content, se figur 22. Jag valde tidigare att inkludera dessa i projektet när jag skapade det.

Jag namnger materialet jag kommer att arbeta med som M_Mark. Detta för att lättare snabbt kunna hitta materialet om så behövs. Den stora bokstaven M i namngivningen kommer från den generella namngivningen i Unreal för Material.



Figur 22. Undermappen Starter Content Materials i Unreal Engine 4.

5.3.1. Material Editor och materialkonfiguration

Följande steg blir att klicka in på M_Mark för att komma åt materialredigeraren som automatiskt öppnas i ett nytt fönster. Här kan jag börja konfigurera utseendet på materialet. För att göra detta kommer jag att kombinera olika texturer som jag senare kommer att kunna använda för att måla upp landskapet exakt som jag vill. På det här sättet får jag en större möjlighet att påverka omgivningens utseende med variation. (Virtus Learning Hub 2017)

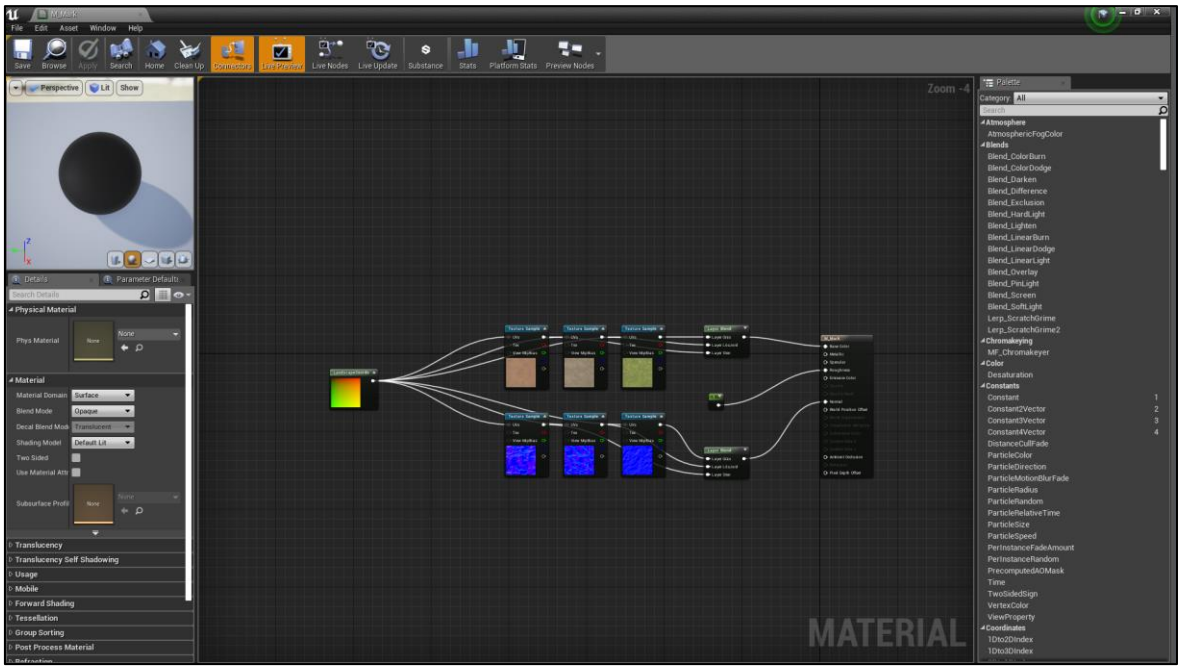
Inne i materialredigeraren letar jag via sökfönstret till höger upp noden LandscapeLayerBlend, eller landskaps lagerblandning. Jag placerar in noden i redigeraren och ansluter den till M_Mark-materialet. Det här gör att jag kan jobba med flera texturer på samma gång. Jag kommer att använda tre olika texturer. Jag markerar fönstret LandscapeLayerBlend och klickar två gånger på knappen Adds Element, eller lägg till element, i fönstret till vänster. Det blir alltså tre element totalt, ett för varje lager. I

ordningsföljd döper jag lagren individuellt till Gräs, Lös.Jord och Sten. Efter det definierar jag texturerna för lagren individuellt.

I sökfönstret till höger letar jag upp noden TextureSample och placerar in tre olika TextureSample i redigeraren. På det här sättet ska jag definiera de tre texturerna jag namngav tidigare individuellt genom att använda färdiga modeller från StarterContent. Dessa motsvarar texturernas utseende. Slutligen ansluter jag TextureSample-noderna till respektive punkter i LandscapeLayerBlend-noden.

För att minska på eventuell ljusreflektion från M_Mark i nivån så letar jag upp noden Constant på samma sätt som med tidigare noder. Jag ställer in konstantvärdet på 1.0 och ansluter Constant-noden till Roughness på M_Mark, se figur 23.

För att underlätta eventuella storleksförändringar i texturerna infogar jag en LandscapeLayerCoords-nod och ansluter denna till alla TextureSample-noder. Slutligen sparar jag och väljer Apply i den övre delen av fönstret och lämnar materialredigeraren.

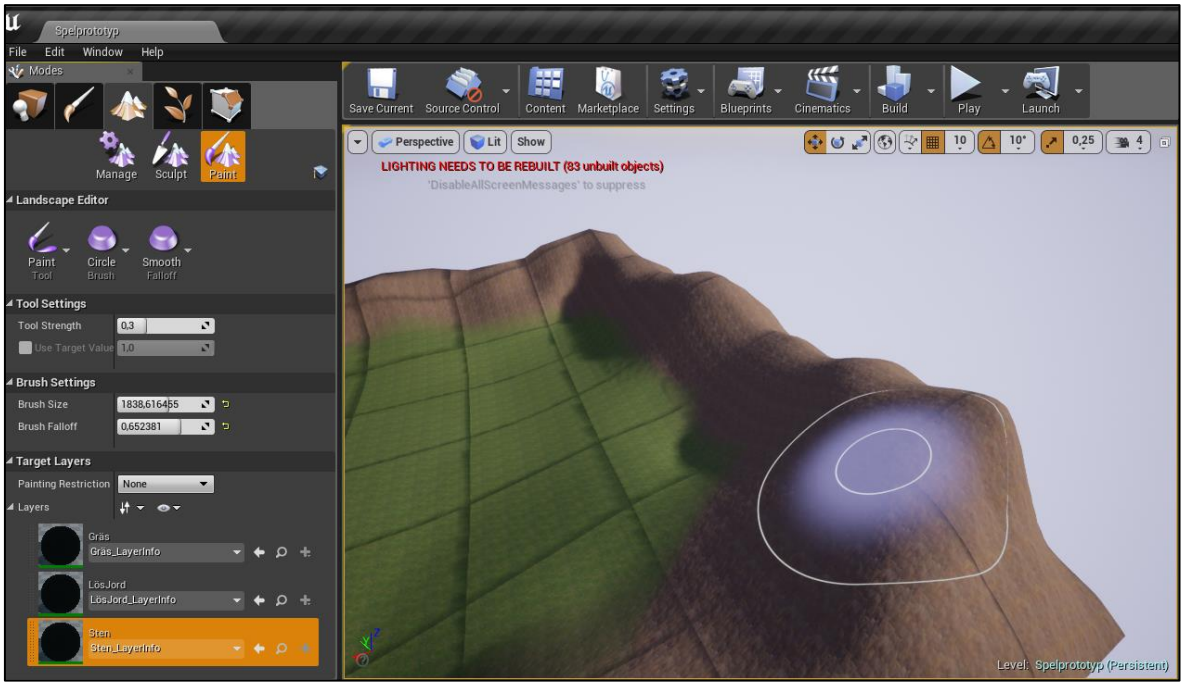


Figur 23. Material Editor i Unreal Engine 4.

5.4 Användning av skapat material för nivådesign

Då jag definierat materialet M_Mark kan jag använda det för att texturera marken. I fönstret World Outliner till höger markerar jag Landscape varefter jag i Details panel nedanför tilldelar materialet för landskapet M_Mark.

Därefter väljer jag Landscape i Modes panel och i fliken Target Layers skapar jag nya lagerfiler genom att klicka på Create Layer Info-knappen brevid lagren. Lagerfilerna behövs för användning av texturer från ett skapat material i nivån. Jag markerar därför lagertexturen Sten och målar kanterna av nivån så att de får ett bergsliknande utseende, se figur 24.



Figur 24. Landscape Paint i Unreal Engine 4.

5.5 Nedladdning och installation av tillägg

Utan gräs, träd och stenar ser nivån väldigt tom ut. För att snabbt kunna skulptera ut omgivningen med sådana objekt behöver jag använda mig av tillägg som innehåller modeller för dessa då en del inte medföljer i Starter Content. Jag har valt att använda mig av ett gratistillägg vid namn Meadow Pack där sådana tillägg ingår. Tillägget kommer från utvecklaren Megascans och hämtades från Unreal Engine Marketplace.

Nu återstår det bara att i tilläggsbiblioteket på Epic Games-klienten välja Add To Project och vänta på att tillägget laddas ner och visas i projektmappen. Efter det kan jag börja använda tillägget.

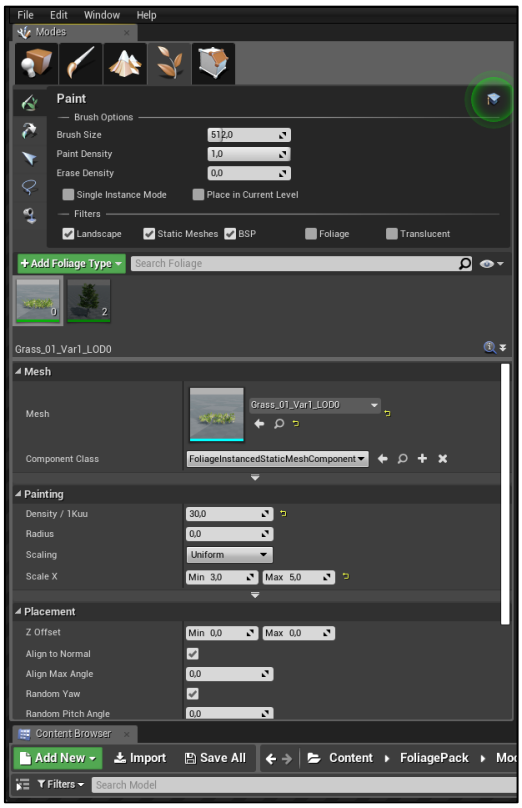
5.5.1. Tilläggsanvändning och fortsatt nivådesign

För att ställa in penseln för nivåutfyllning går jag nu in i Modes panel och klickar på ikonen Foliage som finns brevid Landscape till höger. Här hittar man alternativ för snabb objektutfyllning av nivån, se figur 25. I Content browser klickar jag in i tilläggsmappen och

därefter in i undermappen Model. Här presenteras alla objekttillägg som finns att tillgå i det nedladdade tilläggspaket.

Jag väljer ut ett par statiska objekt som jag vill fylla ut nivån med och drar jag in dem i Foliage-området i Modes panel. Nu kan jag använda dessa för att fylla ut nivån. Man kan välja ifall man vill att objekten ska användas gemensamt eller enskilt. Jag ändrar inte på många inställningar förutom Density. Detta görs på alla objektmodeller enskilt eftersom jag inte vill att träden ska komma för tätt och jag vill inte heller ha för mycket gräs på alla ställen där det finns träd.

Jag ställer in objekten så att de ska kunna användas gemensamt men också enskilt utan att det ser ut att bli fullt vid de olika platserna i nivån de placeras i. Jag strävar efter att uppnå en fin layout.



Figur 25. Foliage inställningar i Unreal Engine 4.

5.6 Simulering av ett vattenhål i nivån

På en plats i nivån vill jag skapa ett litet vattenhål. Som tidigare nämt så använder jag mig av Landscape Sculpt i Modes panel till vänster och skapar en grop i marken. Efter det går

jag in i Place som finns längst till vänster i panelen. Där väljer jag ut en tom Cube, eller kub och infogar den i nivån. Kuben ska motsvara vattenytan.

Jag skalar om kuben så att den täcker hela ytan av gropen jag skapade tidigare. Jag gör kuben tunn för att övergångseffekten från vatten till land ska bli smidig.

I Details panel letar jag upp kolumnen Materials och tilldelar kuben texturen M_Water_Ocean, vilket ger kuben ett vattenutseende. Efteråt bläddrar jag längre ner i Details panel till kolumnen Collision. I inställningen Collision presets, väljer jag NoCollision. Detta gör kuben genomskilning och då blir det möjligt för spelaren att gå under vattnet, se figur 26.

5.6.1. PhysicsVolume

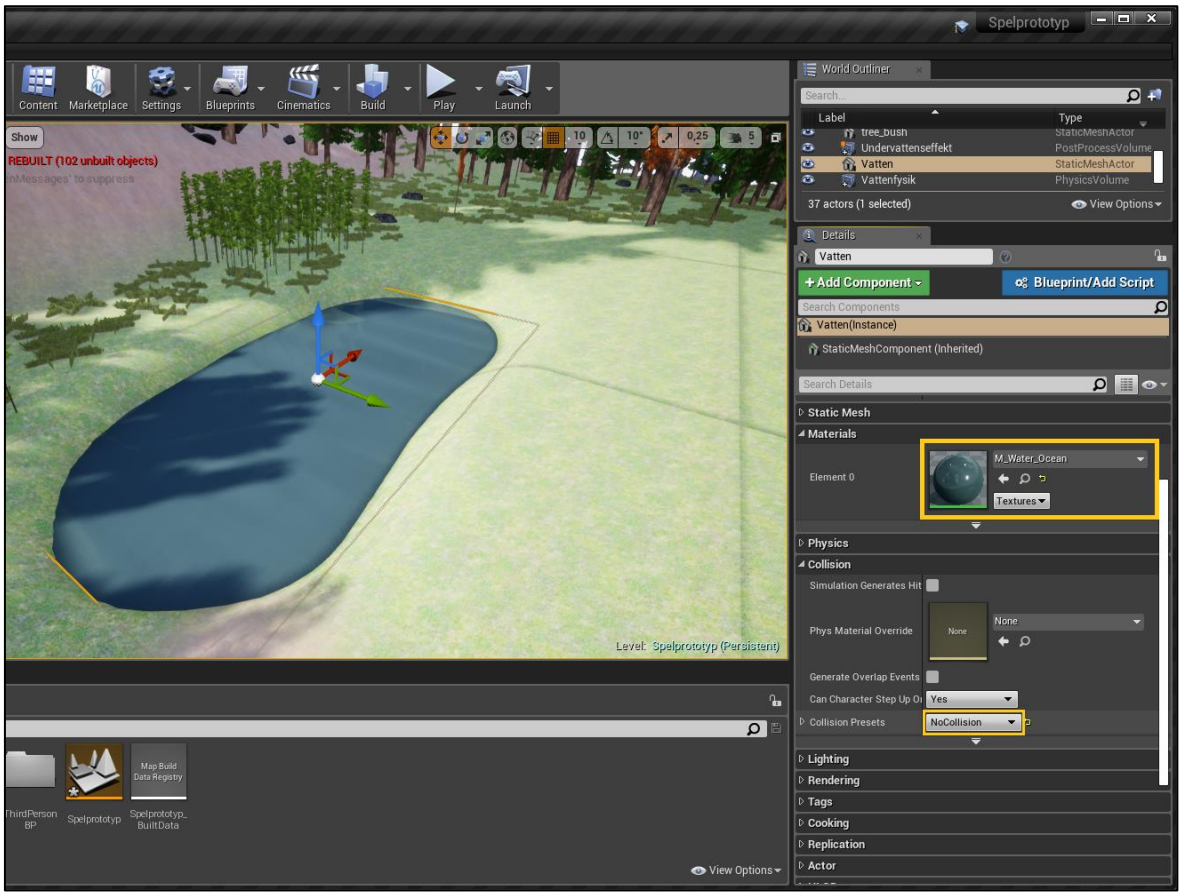
För att simulera effekten att röra sig långsammare under vatten använder jag mig av klassen PhysicsVolume som finns i Modes panel. Jag letar upp klassen med hjälp av sökrutan i panelen och drar in det i nivån. Jag skalar om objektet så att det så täcker hela gropen och slutar precis vid kuben.

PhysicsVolume fungerar så att den kan förändra viktlagarna inom området av objektet (Unreal Engine 2020). Med andra ord kommer spelaren att försinkas när den rör sig inom objektets gränser. Här ändrar jag endast på en inställning. Jag kryssar för Water Volume, som finns under Character Movement. Denna inställning simulerar rörelseförändringen i vatten.

5.6.2. PostProcessVolume

En blåare bild behövs när spelaren befinner sig under vatten. Jag letar reda på klassen PostProcessVolume på samma sätt som med ovanstående klass och infogar den i nivån. Jag ändrar storlek på klassen så att den täcker hela vattendelen och slutar vid vattenytan. Det innebär att undervattenseffekten kommer att upphöra när spelaren kommer över vattenytan.

PostProcessVolume ändrar på olika värden innanför objektets gränser. Spelaren kan se eller uppleva dessa när denne befinner sig inom objektområdet. (Unreal Engine 2020) Här ändrar jag på bildtoningen genom att leta upp Color Grading längre ner i Details panel och går in fliken Misc där jag till sist kryssar för och ändrar på färgen i Scene Color Tint, eller scen färgtoning. Jag väljer en passasnde nyans som jag tycker passar bra för undervattensimulering.



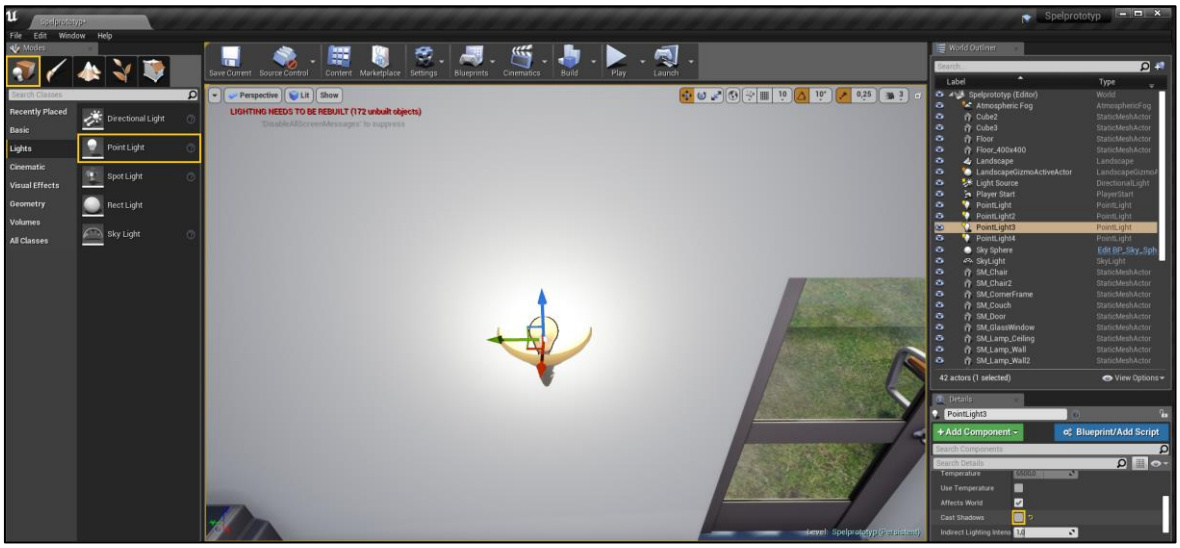
Figur 26. Vattensimulering och inställningar i Unreal Engine 4.

5.7 Ljussättning

För att ljussätta element som t.ex. lampor och liknande objekt måste man använda olika ljusobjekt. Dessa ger ifrån sig olika ljustyper och riktningar beroende på vilket ljusobjekt man väljer.

I Unreal Engine 4 finns det fem olika ljustyper att välja bland. Dessa är Directional, Point, Spot, Rect och Sky Light. När nivån skapades i början valde jag som tidigare nämt alternativet Default, eller standard. Med valet standard placeras ett ljusobjekt färdigt in i nivån i form av Directional Light, ett ljus som simulerarar solljuset. (Unreal Engine 2020)

Jag har byggt upp en statisk modell av ett hus med hjälp av statiska objekt. Inne i huset har jag också placerat ett par statiska modeller som föreställer lampor. Här kommer jag att placera Point Lights. Jag ändrar ljusstyrkan på dessa ljus manuellt till sådant ljus som jag anser vara passande. Jag går i Details panel in i kolumnen Light och justerar Intensity, eller intensitet. Därefter väljer jag att kryssa för eller lämna bort alternativet Cast Shadows, eller avge skugga, beroende på om det är lämpligt för området eller inte, se figur 27.



Figur 27. Point Light och inställningar i Unreal Engine 4.

6 Kritisk granskning

Arbetet var omfattande på flera sätt. Jag fick lägga mycket tid på att lära mig hur spelmotorn och alla dess funktioner fungerar. Det var också utmanande på flera sätt att dokumentera allting som gjordes då det kom in fler detaljer än jag först trodde.

Det var också svårt i början att få en tydlig bild över hur jag skulle strukturera arbetet och hur mycket jag skulle avgränsa. Innan arbetet inleddes hade jag en egen förutfattad aning om hur prototypen skulle komma att se ut då den var färdig. I vilket fall som helst så anser jag att prototypen blev betydligt bättre än jag förväntat mig och målet för att kunna fortsätta utvecklingen av prototypen i framtiden uppnåddes.

Ett problem som uppstod under prototypskapandets gång var att ett av tilläggspaketen inte ville fungera. Mer specifikt varje gång jag försökte läsa in ett träd så kraschade spelmotorn. Av den anledningen tvingades jag till slut komma på en alternativ lösning i form av en annan tillverkares tillägg. Valet föll på tillverkaren PureDogs Foliage Tree Pack, som köptes på Unreal Engine Marketplace. Jag ansåg att detta vara det bästa alternativet. Tilläggspaketet kostade runt 8 euro, en överkomlig summa.

I slutändan var det inte så farligt, även fast jag temporärt tvingades avbryta arbetsprocessen flera gånger för att starta om spelmotorn. Som tur var så blev inga filer skadade under krascharna och det ersättande tillägget fungerade ändå mycket bra.

7 Avslutning

Jag har lärt mig mycket om spelutveckling och spelmotorer under arbetets gång. Tidigare visste jag knappt någonting om spelmotorer och hur de fungerade. Trots att det tog lite tid för mig att inskola mig själv anser jag att spelprototypen blev riktigt bra i slutändan. Jag anser att det blev en riktigt bra början på en produkt som jag kan fortsätta med i framtiden. Det är otroligt hur mycket information som faktiskt finns i en spelmotor och det har fått mig att tänka på hur långt vi faktiskt har kommit inom området sen de första tredimensionella datorspelen introducerades någon gång på nittiotalet.

Jag är mycket nöjd med mitt arbete och med slutresultatet. Jag tycker att jag fick en bra avgränsning på arbetet med tanke på all information som finns i en spelmotor. Själva avgränsningen var just det jag funderade mycket på i början av arbetetsprocessen.

Jag oroade mig för att det antingen skulle bli för mycket eller för lite information i min dokumentation, men i slutändan gick det bra. Jag kan också säga att jag efter detta arbete inspirerade mycket till en möjlig framtida karriär inom spelutveckling. Jag blev mer intresserad än vad jag trodde att jag skulle bli och jag kommer att fortsätta använda spelmotorn samt fördjupa mina kunskaper.

Avslutningsvis vill jag tacka min studiehandledare Nina Hillo som hjälpte mig att få struktur på arbetet. Jag vill också tacka min handledare Kim Roos och min förra handledare, numera pensionär, Klaus Hansen för all handledning och lärorika lektioner jag fått ta del av under min studietid vid Novia i Raseborg.

Källförteckning

About Us (u.å.). [Online] https://quixel.com/megascans/library/about [Hämtat 08.02.2020].

Edwards B., 2009. *Feature* [Online] https://gamasutra.com/view/feature/132426/from_the_past_to_the_future_tim_.php? [Hämtat: 04.02.2020].

Epic Games, 2020 *Frequently Asked Questions* [Online] https://www.epicgames.com/site/en-US/epic-games-store-faq [Hämtat 08.02.2020].

Epic Games., 2020 *Epic Games Store Weekly Free Games in 2020!* [Online] https://www.epicgames.com/store/en-US/news/epic-games-store-weekly-free-games-in-2020 [Hämtat 08.02.2020].

Eurogamer, 2005. *Unreal Engine 3.0* [Online] https://www.eurogamer.net/articles/ss_unrealengine3_e32004 [Hämtat 05.02.2020].

Gamasutra, 2013. *See Epic's Unreal Engine 3 running in HTML5* [Online] https://www.gamasutra.com/view/news/191642/See_Epics_Unreal_Engine_3_running_in_HTML5.php [Hämtat 05.02.2020].

Game Dev Academy, 2017 *Unreal Engine 4 Tutorial for Beginners - Season 1* [Online] https://www.youtube.com/playlist?list=PLsPHRLf6UN4m1LQhNlRM22np4-4bIcDa4 [Hämtat 12.01.2020].

Herz, J.C., 1999. *GAME THEORY; For Game Maker, There's Gold in the Code* [Online] https://www.nytimes.com/1999/12/02/technology/game-theory-for-game-maker-there-s-gold-in-the-code.html [Hämtat 05.02.2020].

IGN, 2012 *Epic Games Announces Unreal Development Kit, Powered by Unreal Engine 3* [Online] https://www.ign.com/articles/2009/11/05/epic-games-announces-unreal-development-kit-powered-by-unreal-engine-3 [Hämtat 22.01.2020].

Kennedy, B., 2002. *Uncle Sam Wants You (To Play This Game)* [Online] https://www.nytimes.com/2002/07/11/technology/uncle-sam-wants-you-to-play-this-game.html [Hämtat 04.02.2020].

Long, K., 2019 *Top 20 Third Person Shooters For Mobile* [Online] https://cellularnews.com/mobile-games/top-20-third-person-shooters-for-mobile/ [Hämtat 05.02.2020].

Maximum PC, 2004. *The Ultimate 3D Gaming Tech Guide* [Online] https://archive.org/details/Maximum_PC_Autumn_2004/mode/2up [Hämtat 03.02.2020]

McLeroy, C., 2008. *Improving "America's Army"* [Online] https://www.army.mil/article/11935/improving_americas_army [Hämtat 04.02.2020].

PC Gamer, 2018 *Epic's Tim Sweeney reveals how the company lands exclusives for the Epic Store* [Online] https://www.pcgamer.com/epics-tim-sweeney-reveals-how-the-company-lands-exclusives-for-the-epic-store/ [Hämtat 05.02.2020].

PopularTimelines (u.å.) *Epic Games* [Online] https://populartimelines.com/timeline/Epic-Games [Hämtat: 04.02.2020].

Rock Paper Shotgun, 2012 *Fortnite's Jessen Talks Minecraft, PC Gaming, UE4* [Online] https://www.rockpapershotgun.com/2012/07/20/fortnites-jessen-talks-minecraft-pc-gaming-ue4/ [Hämtat 10.02.2020].

TheHappieCat, 2015 *How Game Engines Work!* [Online] https://www.youtube.com/watch?v=DKrdLKetBZE [Hämtat 17.02.2020].

Thomsen M., 2012 *History of the Unreal Engine* [Online] https://www.ign.com/articles/2010/02/23/history-of-the-unreal-engine [Hämtat 22.01.2020].

TVTropes (u.å.). *Side View* [Online] https://tvtropes.org/pmwiki/pmwiki.php/Main/SideView [Hämtat 05.02.2020].

Unreal Engine Documentation (u.å.). [Online] https://docs.unrealengine.com/en-US/index.html [Hämtat 25.01.2020].

Unreal Engine Forums, 2016 *Game engines price comparison* [Online] https://forums.unrealengine.com/community/general-discussion/101974-game-engines-price-comparison [Hämtat 04.02.2020].

Unreal Engine Forums, 2018 *Is UE4 the most expensive Game Engine of all times?* [Online] https://forums.unrealengine.com/unreal-engine/feedback-for-epic/1450843-is-ue4-the-most-expensive-game-engine-of-all-times [Hämtat 04.02.2020].

Valich, T., 2008. *Tim Sweeney, Part 3: Unreal Engine 4.0 aims at next-gen console war* [Online] https://www.tgdaily.com/36436-tim-sweeney-part-3-unreal-engine-40-aims-at-next-gen-console-war [Hämtat 06.02.2020].

Virtus Learning Hub, 2018 *Level Design Essentials - Unreal Engine 4 Course* [Online] https://www.youtube.com/playlist?list=PLL0cLF8gjBpo3EUz0KAwdZrDYr6FzfLGG [Hämtat 05.02.2020].

Wired, 2012. *Unreal Engine 4 Behind Closed Doors at GDC* [Online] https://www.wired.com/2012/02/unreal-engine-4-gdc/ [Hämtat 06.02.2020].

Figurförteckning