Opinnäytetyö (AMK)

Tietotekniikka

Sulautetut ohjelmistot

2018

Nianzu Yang

UNREAL ENGINE 4 -KEHITYSYMPÄRISTÖN LISÄOSAN KEHITTÄMINEN

– työkalu assetin hallintaan

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Sulautetut ohjelmistot

2018 | 35

Nianzu Yang

UNREAL ENGINE 4 -KEHITYSYMPÄRISTÖN LISÄOSAN KEHITTÄMINEN

– työkalu assetin hallintaan

Työn tavoitteena oli tutkia ja toteuttaa lisäosa Unreal Engine 4 -kehitysympäristöön. Projektissa toteutettiin yksinkertainen prototyyppi assetin hallintatyökalu lisäosa Unreal Engine 4 -kehitysympäristössä. Lisäosa toteutettiin C++ ohjelmointikielellä.

Tulokseksi saatiin toimiva ohjelma, joka lataa Viversion Oy:n palvelimesta rajanpinnan välityksellä halutut Zip-tiedostot ja purkaa ne sitten määritettyyn kansioon.

ASIASANAT:

Unreal Engine 4, Pelimoottori, C++

BACHELOR´S | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Embedded Software

2018 | 35

Nianzu Yang

DEVELOPING AN UNREAL ENGINE 4 PLUGIN

A Tool for asset management

The goal of the thesis was to study and implement a plugin to the Unreal Engine 4 development environment. A simple prototype asset management plugin was implemented in the project for the client company Viversion Oy. The plugin was implemented in C++ programing language.

As a result, a viable program was created. The plugin downloads ZIP files from the server and extracts them to the specified folder.

KEYWORDS:

Unreal Engine 4, Game engine, C++

SISÄLTÖ

LIITTEET

KUVAT

KOODIT

KÄYTETYT LYHENTEET

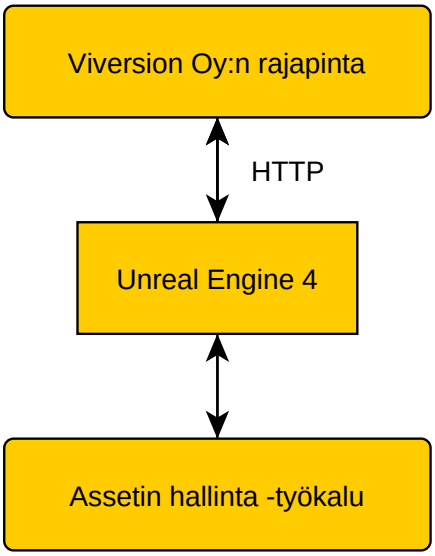
Blueprints	Unreal Engine 4 -kehitysympäristön oma visuaalinen ohjelmointikieli.
HTTP	Hypertext Transfer Protocol. Tiedonsiirtoprotokolla.
JSON	JavaScript Object Notation. Tiedonvälityksen tiedostomuoto.
Plugin	Lisäosa, ohjelman ominaisuuksia lisäävä tai muokkaava toinen ohjelma.
ZIP	Tiedonpakkausmenetelmä.

1 JOHDANTO

Opinnäytetyön aiheena on lisäosan kehittäminen Unreal Engine 4 -kehitysympäristöön. Opinnäytetyön tarkoituksena on toteuttaa assetin (pelin käyttömateriaalin) hallinta -työkalu ohjelmistoyhtiö Viversion Oy:lle. Käyttömateriaalit ovat esimerkiksi kolmiulotteinen mallinnus, äänet ja tekstuurit eli kaksiulotteinen kuva, jota käytetään kuvioimiseen.

Tässä opinnäytetyössä käsitellään, miten Unreal Engine 4 -kehitysympäristö saadaan laajennettua lisäosan avulla, ja assetin hallinta -työkalun toteutusta sekä lopputulosta. Salassapitovelvollisuuden vuoksi opinnäytetyössä ei käsitellä assetin hallinta -työkalun ja Viversion Oy:n rajapinnan toiminnollisuuksia.

Opinnäytetyön tavoitteena oli saada työkalu hakemaan Viversion Oy:n rajapinnasta tarvittavat ZIP pakettitiedostot HTTP-yhteyden avulla ja purkamaan ladatut ZIP pakettitiedostot samaan kansioon. Kuvassa 1 näkyy lisäosan ylimmän tason lohkokaavio.



Kuva 1. Lohkokaavio.

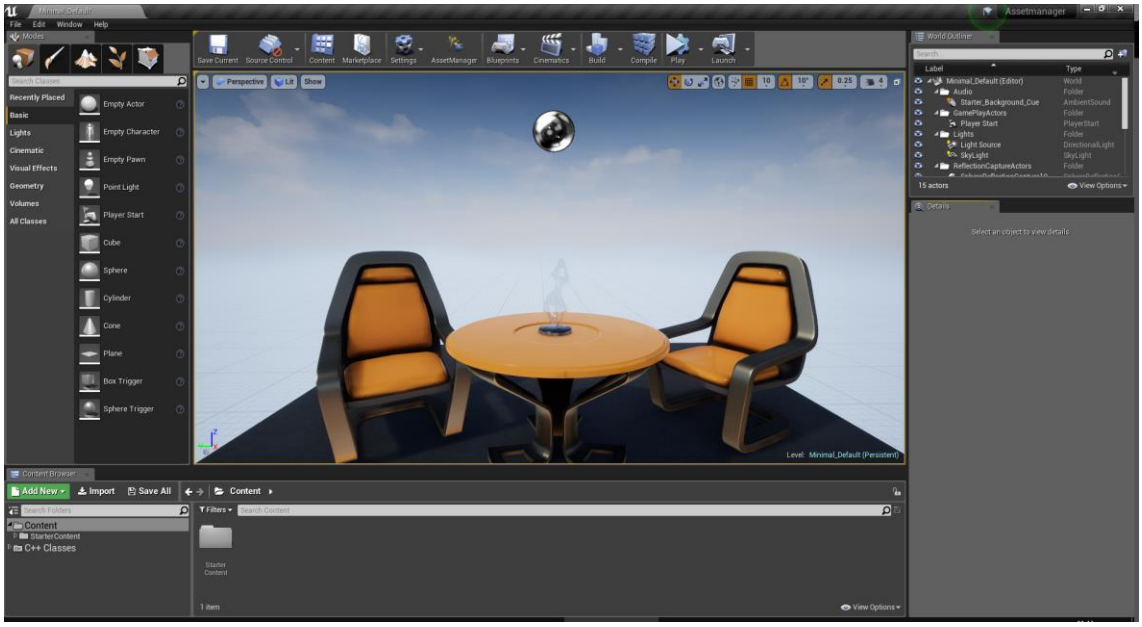
2 UNREAL ENGINE 4 -KEHITYSYMPÄRISTÖ

Unreal Engine 4 -kehitysympäristö on yksi suurimmista pelimoottoreista. Pelimoottorit ovat kehitystyökaluja, joiden tarkoituksena on vähentää työtä pelien kehityksessä. Ne sisältävät valmiita kirjastoja, kuten fysiikkamallinnus, ääni ja grafiikkarenderöinti kirjastot. Useimmat pelimoottorit, kuten Unreal Engine 4 mukana toimitetaan myös käyttöliittymiä pelikehittäjien avuksi.

Unreal Engine 4 -pelimoottori tunnetaan näyttävimmistä grafiikoista. Pelimoottori on osittain hankala käyttää, sillä pelimoottorin laajuuden johdosta monet ominaisuudet ovat piiloutuneet useiden vaiheiden taakse, mikä hidastaa työskentelyä.

Pelimoottorissa pelin toiminnallisuudet voidaan toteuttaa pelimoottorin omalla ”Blueprints Visual Scripting”-visuaalisella palikkaohjelmointikielellä tai C++ -ohjelmointikielellä. C++ -ohjelmointikieli mahdollistaa tehdä monimutkaisia funktioita, joita ei palikkaohjelmointikielellä pysty.

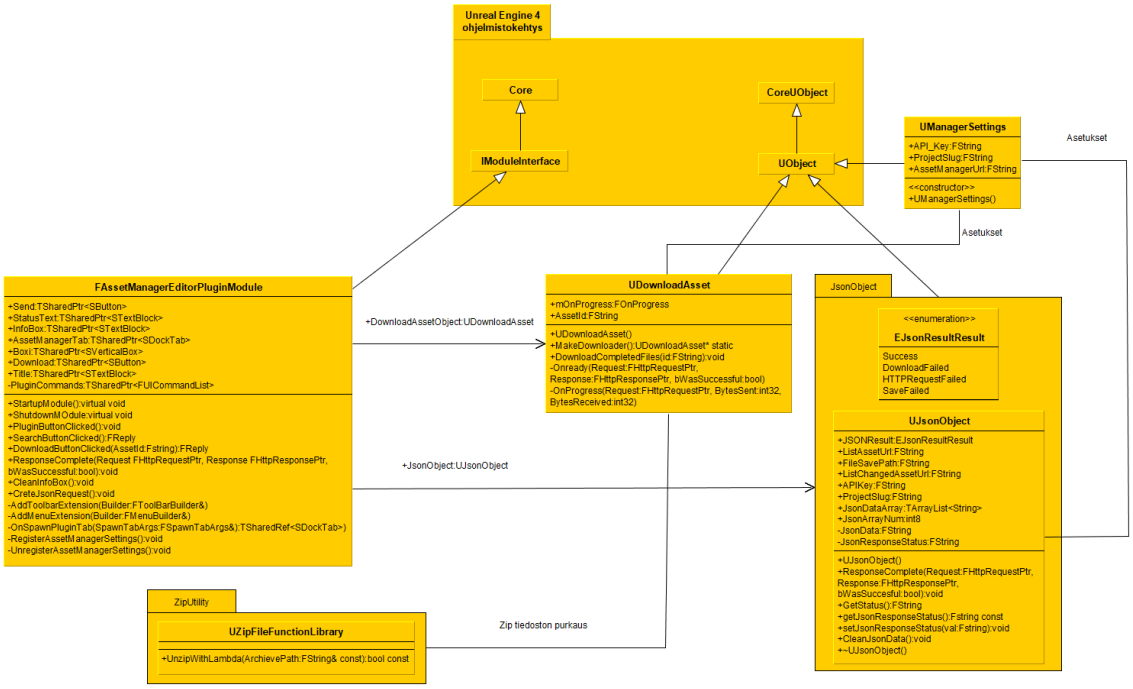
Kuvassa 2 näkyy pelimoottorin käyttöliittymä eli editorinäkymä. Se koostuu monista erilaisista moduuleista. Editorinäkymä avautuu aina ensimmäisenä, kun Unreal Engine 4 -pelimoottori avataan. Editorinäkymä keskellä on Viewport-ikkuna, johon lisätään peliobjektit, jotka tulevat näkyviin pelimaailmaan. Alalaidassa on projektinäkymä, jossa näkyy kaikki projektiin lisätyt peliobjektit. Oikealla yläpuolella on pelimaailman jäsennin, jossa näkyy pelimaailmaan lisätyt peliobjektit. Vasemmalla laidalla on pelimoottorin oma työkalu, josta saa luotua yksinkertaisia peliobjekteja.



Kuva 2. Unreal Engine 4 -kehitysympäristön käyttöliittymä (Unreal Engine 4.17).

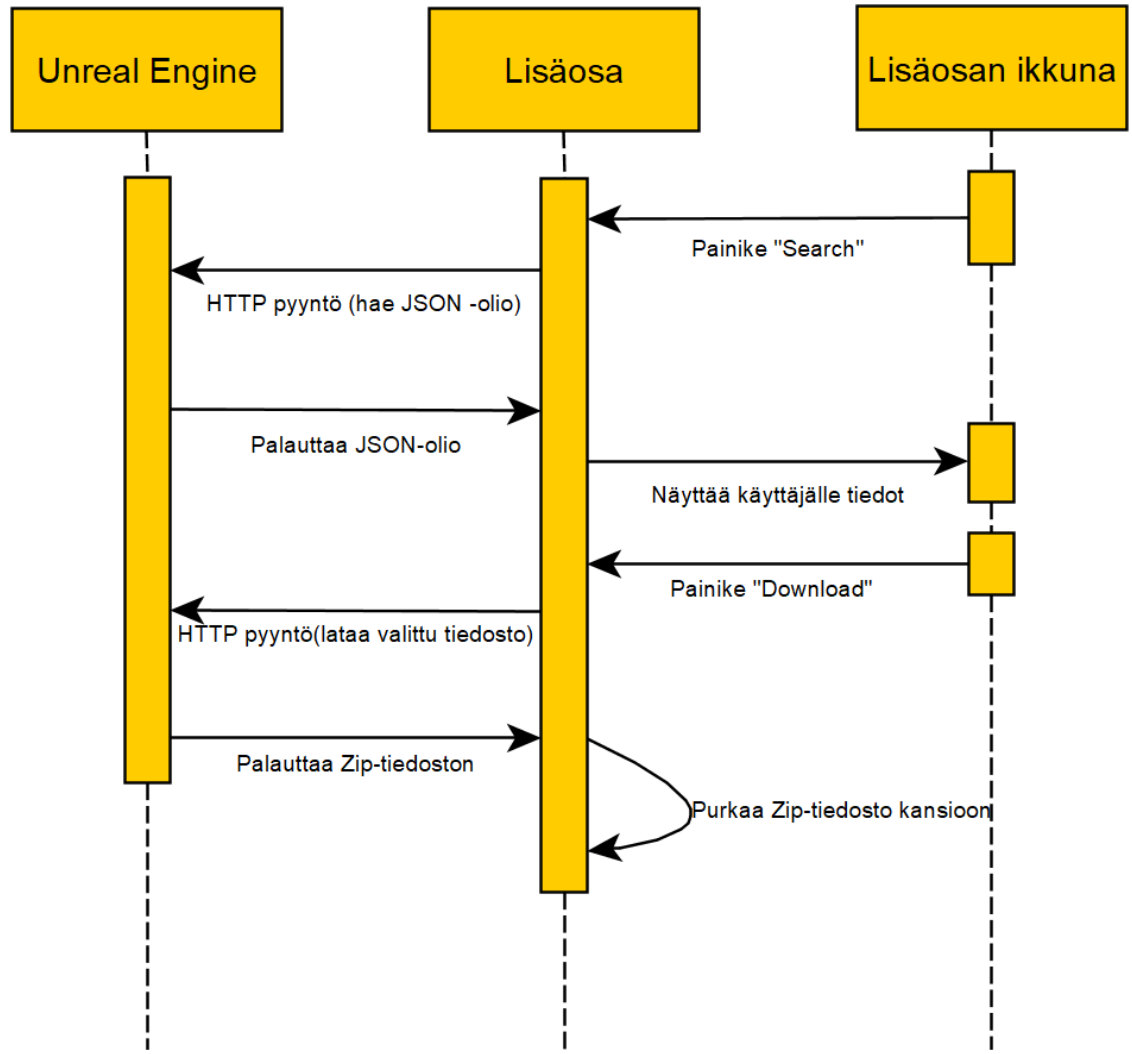
3 OHJELMOINTI UNREAL ENGINE 4 - KEHITYSYMPÄRISTÖSSÄ

Kuvassa 3 näkyy, mitkä lisäosan luokat periytyvät Unreal Engine 4-ohjemistokehityksestä.



Kuva 3. Lisäosan luokkakaavio.

Sekvenssikaavion avulla voidaan kuvata olioiden vuorovaikutusta ohjelmassa. Kuvassa 4 näkyy Lisäosan sekvenssikaavio. Ohjelma alkaa kohdasta ”Painike Search” ja loppuu kohtaan ”Purkaa Zip-tiedosto kansioon”.



Kuva 4. Lisäosan sekvenssikaavio.

3.1 C++ -ohjelmointi

C++ -lähdekoodin kääntäminen Unreal Engine 4 -kehitysympäristössä tarvitaan Windows pohjaisessa käyttöjärjestelmässä Microsoft Visual Studio -kehitystyökalua tai macOS pohjaisessa käyttöjärjestelmässä Xcode -kehitystyökalua. (Unreal Engine 2018b.)

Unreal Engine 4 -kehitysympäristön rajapinta on suunniteltu siten, että ohjelmoija voi kirjoittaa C++ -ohjelmointikielellä tavallisia funktioita tai UFunction-jäsenfunktioita. Koodissa 1 näkyy esimerkki, miten UFunction-jäsenfunktion määritellään C++ otsikkotiedostossa. (Unreal Engine 2018a.)

Esimerkiksi pelinsuunnittelija voi hyödyntää pelimekaniikan suunnittelussa UFunction funktiota Blueprints Visual Scripting -palikkaohjelmointikieltä käyttäen. (Unreal Engine 2018b.)

```
UFUNCTION(BlueprintCallable, Category="Esim")
void Esimerkki();
```

Koodi 1. UFunction-määrittely otsikkotiedostossa.

Koodissa 2 näkyy esimerkki yksinkertaisesta C++:lla tehdystä koodista, jossa luodaan lamppu pelimaailmaan ja asetetaan lampulle kirkkauden sekä väriarvot.

```
// Fill out your copyright notice in the Description page of Project Settings.

#include "PointLightCpp.h"


// Sets default values
APointLightCpp::APointLightCpp()
{
    // Set this actor to call Tick() every frame.  You can turn this off to improve performance if you don't need it.
    PrimaryActorTick.bCanEverTick = true;

    Brightness = 100.f;

    Poinlight1->SetIntensity(Brightness);
    Poinlight1->SetLightColor(FLinearColor(255,5,5));

}

// Called when the game starts or when spawned
void APointLightCpp::BeginPlay()
{
    Super::BeginPlay();

}

// Called every frame
void APointLightCpp::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);

}
```

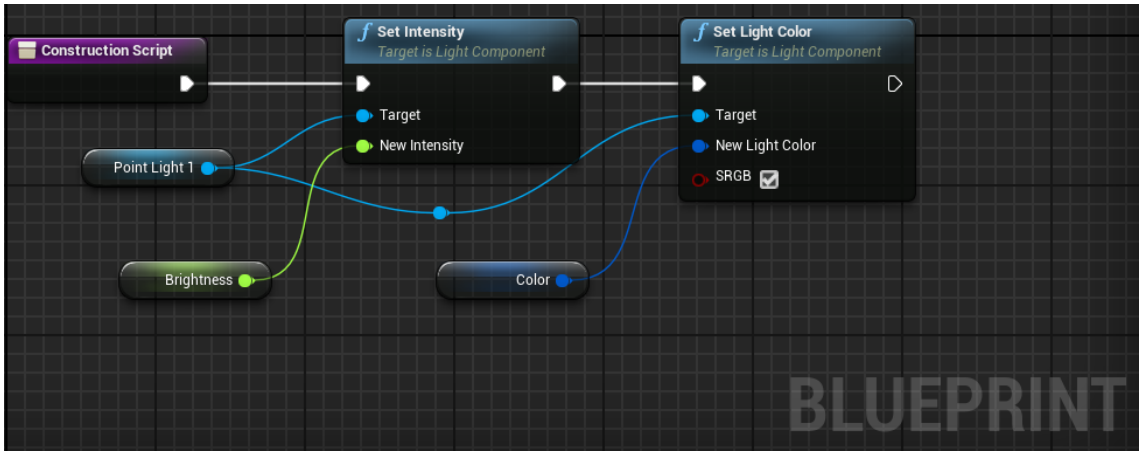
Koodi 2. C++ -lähdekoodi.

3.2 Blueprints Visual Scripting -ohjelmointi

Blueprints Visual Scripting -ohjelmointi on toinen tapa ohjelmoida pelimekaniikkoja Unreal Engine 4 -kehitysympäristössä. Blueprints Visual Scripting -ohjelmointikieli on pelimoottorin oma visuaalinen palikkaohjelmointikieli. Blueprints -ohjelmointi tapahtuu pelimoottorin sisään rakennetussa työkalussa.

Blueprints -ohjelmoinnissa käytetään valmiina olevia yksinkertaisia funktiota, joiden avulla voidaan luoda erilaisia komentoja.

Kuvassa 5 näkyy Blueprints Visual Scripting -ohjelmointikielellä toteutettu yksinkertainen komento, joka asettaa lampulle kirkkauden sekä väriarvot.

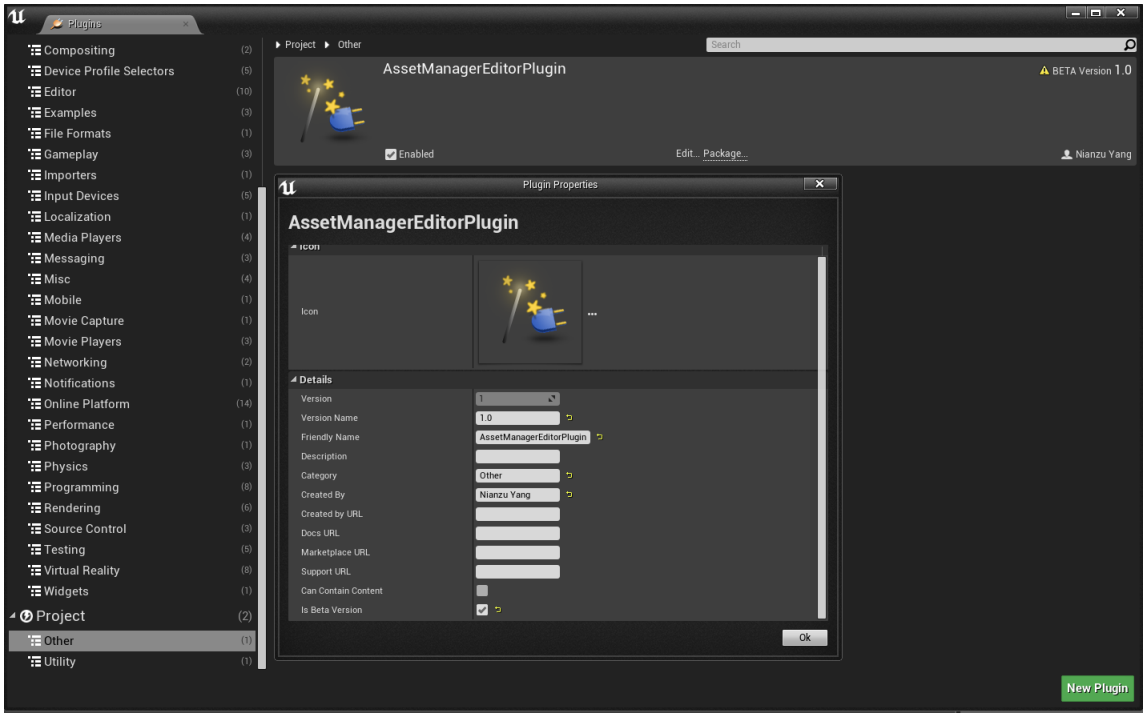


Kuva 5. Blueprints Visual Scripting -ohjelmointi.

4 LISÄOSAN OHJELMOINTI UNREAL ENGINE 4 - KEHITYSYMPÄRISTÖÖN

Lisäosa on nippu toiminnallisuuksia, joita voidaan käyttää joko pelinkehityksessä tai pelimaailmassa. Lisäosa voi olla myös itsenäinen ohjelma joka toimii pelimoottorissa, esimerkiksi tässä opinnäytetyössä tehty lisäosa on itsenäinen ohjelma. Unreal Engine 4 -kehitysympäristön alijärjestelmät ovat suunniteltu niin, että sitä pystytään laajentamaan lisäosien avulla muokkaamatta pelimoottorin lähdekoodia. Lisäosan kehityksessä voidaan kirjoittaa uusia toiminnollisuuksia tai käyttää hyödykseen olemassa olevia toiminnollisuuksia. (Unreal Engine 2018c.)

Kuvassa 6 näkyy käytössä olevat lisäosat Unreal Engine 4 -kehitysympäristössä.



Kuva 6. Lisäosan hallintaikkuna.

Lisäosan suunnittelussa pitää ottaa huomioon, mihin tarkoitukseen lisäosa tullaan käyttämään, sillä lisäosa voidaan tehdä kahteen eri tarkoitukseen. (Unreal Engine 2018d.)

1. Toiminnallisuus lisäosa: muut aliohjelmat ovat riippuvaisia. Eli itsenäinen ohjelma, joka suorittaa tietyn kutsutun toiminnon ja sitä voidaan kutsua eri puolilta ohjelmaa.
2. Itsenäinen lisäosa: Itsenäinen työkalu, muut aliohjelmat eivät voi kutsua tätä ohjelmaa, mutta tämä voi olla riippuvainen toisista aliohjelmista.

5 ASSETIN HALLINTA -TYÖKALUN TOTEUTTAMINEN UNREAL ENGINE 4 -KEHITYSYMPÄRISTÖÖN

5.1 Tavoitteet

Assetin hallinta -työkalun kehityksen tavoitteena oli saada assetit helposti ja nopeasti Viversion Oy:n rajapinnasta pelimoottoriin. Työkalun toiminnallisuuden tavoitteena oli saada työkalu siihen vaiheeseen, että pystyttiin lataamaan Zip -tiedostoja rajapinnasta ja paketteja pystyttiin purkamaan yhteen määritettyyn kansioon.

5.2 Käytetyt teknologiat

Projektissa käytettiin Unreal Engine 4 -kehitysympäristöä, jossa käytetään C++ ohjelmointikieltä. Työkalussa käytettiin HTTP-protokollan POST-metodia tiedostojen lataamiseen. Viversion rajapinnasta palautuneet JSON-oliotaulukko arvot tämän jälkeen voidaan ladata Zip-tiedostot. Ladatut Zip-tiedostot puretaan kolmannen osapuolen Zip Utility nimisellä lisäosaohjelmalla.

5.3 Lisäosan toteutus

Työkalulle tehtiin ensi oma asetusvalikko. johon määritetään rajapintakutsu asetuksia.

Kuva 7. Pakolliset arvot työkalulle.

Koodissa 3 näkyy asetusvalikon lähdekoodit, otsikkotiedosto sekä lähdekooditiedosto. Koodissa luodaan tyhjät kentät kolmelle asetukselle, jotka tulevat näkyviin projektin asetuksessa.

ManagerSettigs.h

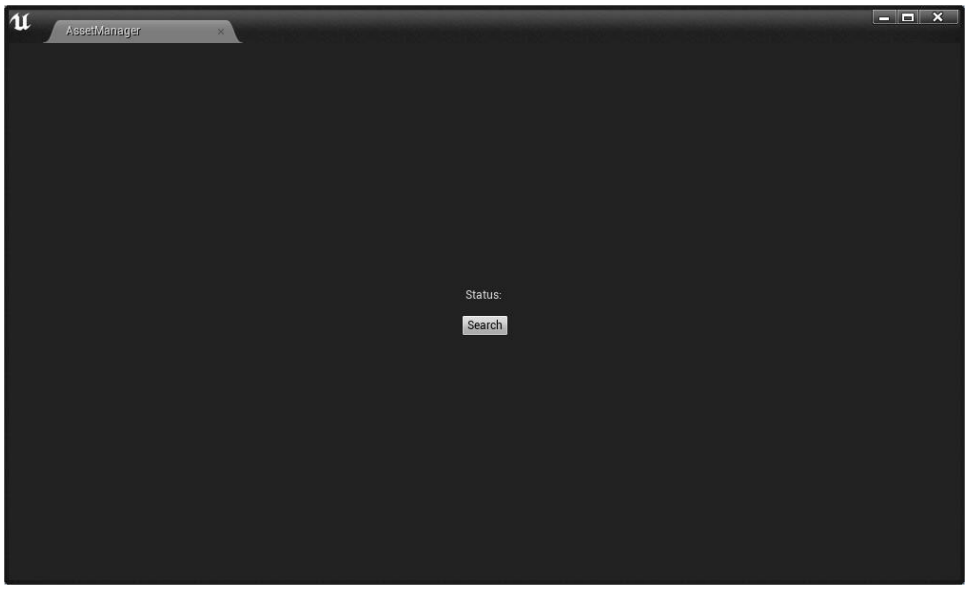
```
// Fill out your copyright notice in the Description page of Project Settings.

#pragma once

#include "ManagerSettings.generated.h"

/**
 *
 */
UCLASS(config = Engine, defaultconfig)
class UManagerSettings : public UObject
{
		GENERATED_UCLASS_BODY()

		UPROPERTY(config, EditAnywhere, Category = Settings)
		FString API_Key;

		UPROPERTY(config, EditAnywhere, Category = Settings)
		FString ProjectSlug;

		UPROPERTY(config, EditAnywhere, Category = Settings)
		FString AssetManagerUrl;
public:
		UManagerSettings();
};
```

ManagerSettigs.cpp

```
// Fill out your copyright notice in the Description page of Project Settings.

#include "ManagerSettings.h"

UManagerSettings::UManagerSettings(const FObjectInitializer& ObjectInitializer)
		:Super(ObjectInitializer)
		,API_Key("")
		,ProjectSlug("")
		,AssetManagerUrl("")
{

}
```

Koodi 3. Asetusvalikon lähdekoodit.

Kuvassa 8 näkyy työkalun aloitusnäkymä. Search-painiketta painamalla ohjelmaa etsii kaikki palvelimissa olevat assetit.



Kuva 8. Työkalun ikkuna.

Kuvassa 9 näkyy kaikki palvelimessa olevat assettien nimet. Download-painiketta painamalla ohjelma lataa assetin Zip-tiedostona ja purkaa ladatun Zip-tiedoston sisällön kansioon. Status kohdalla numero ’200’ on jätetty testi mielessä näkyviin, joka kertoo pyynnön onnistuneen.

Kuva 9. Palvelimessa olevat assetit.

Ennen HTTP-pyyntöä, määritetään ensi otsikkotiedot, pyynnön metodi sekä viestin esitysmuoto. X-APIKEY otsikko on Viversion Oy:n rajapinnan oma salainen avain, jonka avulla päästään rajapintaan. HTTP-pyynnön metodi POST lähettää ”project_slug” eli projektin nimen palvelimelle, esim. luo uusi JSON-olio tästä projektista.

HTTP-pyyntö suoritetaan ”HttpRequest->ProcessRequest” komennolla. Onnistuneen pyynnön jälkeen suoritetaan ”ResponseComplete” funktion, joka näkyy koodissa 4.

```
void FAssetManagerEditorPluginModule::CreateJsonRequest()
{
            TSharedRef<IHttpRequest> HttpRequest = FHttpModule::Get().CreateRequest();
            HttpRequest->SetHeader(TEXT("X-APIKEY"), JsonObject->APIKey);
            HttpRequest->SetHeader(TEXT("Content-Type"), TEXT("application/x-www-form-
            urlencoded"));
            HttpRequest->SetURL(JsonObject->ListAssetsUrl);
            HttpRequest->SetVerb(TEXT("POST"));

            HttpRequest->SetContentAsString("project_slug=" + JsonObject->ProjectSlug);
            HttpRequest->OnProcessRequestComplete().BindRaw(this,
            &FAssetManagerEditorPluginModule::ResponseComplete);
            HttpRequest->ProcessRequest();
}
```

Koodi 4. Asetetaan arvot HTTP-pyyntöä varten.

Unreal Engine 4 -kehitysympäristön käyttöliittymä on rakennettu Slate nimisellä käyttöliittymä ohjelmistokehyksellä. (Unreal Engine 2018e.) Koodissa 5 näkyy Slate pienoisohjelman koodi For-silmukan sisällä. Koodi suoritetaan onnistuneen HTTP-pyynnön jälkeen. Silmukkaa luo tekstilaatikon sekä painikkeen JSON-olio määrän mukaan.

Epäonnistunut HTTP-pyyntö palauttaa HTTP-vastauskoodin, esimerkiksi virheellinen API Key palauttaa koodin ’401’ eli käyttäjä ei tunnistettu.

```
void FAssetManagerEditorPluginModule::ResponseComplete(FHttpRequestPtr Request, FHttpRespon-
sePtr Response, bool bWasSuccessful)
{
		JsonObject->ResponseComplete(Request, Response, bWasSuccessful);
		if (Response.IsValid())
		{
			StatusText->SetText(FText::FromString(FString::FromInt(Response-
			>GetResponseCode())))
		}
		for (int i = 0; i < JsonObject->JsonArrayNum; i +=2)
		{

			FString AssetId = JsonObject->JsonDataArray[i+1];

			Boxi->AddSlot()
			.HAlign(HAlign_Left)
			.VAlign(VAlign_Bottom)
			[
				SAssignNew(Title, STextBlock)
				.Text(FText::FromString(JsonObject->JsonDataArray[i]))
			];
			Boxi->AddSlot()
			.HAlign(HAlign_Left)
			.VAlign(VAlign_Bottom)
			.AutoHeight()
			[
				SAssignNew(Download, SButton)
				.Text(FText::FromString(TEXT("Download")))
				.OnClicked(FOnClicked::CreateRaw(this, &FAssetManage-
				rEditorPluginModule::DownloadButtonClicked, AssetId))
			];
		}

}
```

Koodi 5. Tekstilaatikot ja painikkeet työkalu ikkunassa.

Lisäosa työkalussa käytetään kolmannen osapuolen ZipUtility nimistä ohjelmaa, jonka avulla saadaan ladattu Zip-tiedoston sisältö purettua. Koodissa 6 näkyvä ”OnReady” jäsenfunktio suoritetaan onnistuneen HTTP-pyynnön jälkeen.

Ohjelmassa asetetaan tiedostopolku ladatulle tiedostolle. Tämän jälkeen haetaan ladattu tiedosto ja puretaan se kutsumalla ZipUtilityn staattista funktiota.

```
void UDownloadAsset::OnReady(FHttpRequestPtr Request, FHttpResponsePtr Response, bool
bWasSuccessful)
{
		RemoveFromRoot();
		Request->OnProcessRequestComplete().Unbind();
		FString FileSavePath = "D:\\TESTI\\AssetManagerFile" + AssetId + ".zip";

		if (Response.IsValid() && EHttpResponseCodes::IsOk(Response-
		>GetResponseCode()))
		{
				// SAVE FILE
				IPlatformFile& PlatformFile =
				FPlatformFileManager::Get().GetPlatformFile();

				// create save directory if not existent
				FString Path, Filename, Extension;
				FPaths::Split(FileSavePath, Path, Filename, Extension);
				if (!PlatformFile.DirectoryExists(*Path))
				{
						if (!PlatformFile.CreateDirectoryTree(*Path))
						{
								return;
						}
				}

				// open/create the file
				IFileHandle* FileHandle = PlatformFile.OpenWrite(*FileSavePath);
				if (FileHandle)
				{
						// write the file
						FileHandle->Write(Response->GetContent().GetData(),
						Response->GetContentLength());
						// Close the file
						delete FileHandle;

						UZipFileFunctionLibrary::UnzipWithLambda(FString("D:/T
						ESTI/AssetManagerFile" + AssetId + ".zip"),
						[]()
						{
								nullptr;
								//Called when done
						},
								[](float Percent)
						{
								nullptr;
								//called when progress updates with % done
						});
				}

}
```

Koodi 6. Tallentaa ja purkaa Zip-tiedoston.

6 YHTEENVETO

Opinnäytetyön tavoitteena oli toteuttaa lisäosaohjelman Unreal Engine 4 -kehitysympäristöön, joka mahdollistaa assetin lataamista palvelimesta.

Lisäosaohjelmasta jäi puuttumaan ominaisuusia, joita rajattiin pois ajan puutteen vuoksi. Tärkeimmät ominaisuudet saatiin kuitenkin valmiiksi, ja toimeksiantaja oli tyytyväinen tuloksiin.

Työn tuotoksena saatiin toimiva ohjelma, joka lataa palvelimesta rajanpinnan välityksellä Zip-tiedoston, ja ladatun Zip-tiedoston sisältö puretaan määritettyyn kansioon.

LÄHTEET

Unreal Engine 2018a. UFunctions 2018. Viitattu 9.4.2018. https://docs.unrealengine.com/en-us/Programming/UnrealArchitecture/Reference/Functions

Unreal Engine 2018b. Introduction to C++. Viitattu 6.1.2018. https://docs.unrealengine.com/latest/INT/Programming/Introduction/

Unreal Engine 2018c. Plugins. Viitattu 18.1.2018. https://docs.unrealengine.com/latest/INT/Programming/Plugins/

Unreal Engine 2018d. An Introduction to UE4 Plugins. Viitattu 26.3.2018. https://wiki.unrealengine.com/An_Introduction_to_UE4_Plugins

Unreal Engine 2018e. Slate UI Framework 2018. Viitattu 2.4.2018. https://docs.unrealengine.com/en-us/Programming/Slate

Lähdekoodit

AssetManagerEditorPlugin.h

```
// Copyright 1998-2017 Epic Games, Inc. All Rights Reserved.

#pragma once

#include "CoreMinimal.h"
#include "ISettingsModule.h"
#include "ISettingsSection.h"
#include "ISettingsContainer.h"
#include "ManagerSettings.h"
#include "JsonObject.h"
#include "DownloadAsset.h"
#include "ModuleManager.h"

#define LOCTEXT_NAMESPACE "FAssetManagerEditorPluginModule"
DECLARE_LOG_CATEGORY_EXTERN(LogAssetManagerPlugin, Log, All);
class FToolBarBuilder;
class FMenuBuilder;

class FAssetManagerEditorPluginModule : public IModuleInterface
{
public:

            /** IModuleInterface implementation */
            virtual void StartupModule() override;
            virtual void ShutdownModule() override;

            void PluginButtonClicked();

            FReply SearchButtonClicked();

            FReply DownloadButtonClicked(FString AssetId);

            void ResponseComplete(FHttpRequestPtr Request, FHttpResponsePtr Response, bool
            bWasSuccessful);

            void CleanInfoBox();

            void CreateJsonRequest();

            //SearchButton
            TSharedPtr<SButton> Send;
            //StatusTextBlock
            TSharedPtr<STextBlock> StatusText;
            //InfoBox
            TSharedPtr<STextBlock> InfoBox;

            TSharedPtr<SDockTab> AssetManagerTab;

            TSharedPtr<SVerticalBox> Boxi;

            //Downloadbutton
            TSharedPtr<SButton> Download;

            //AssetTitle
            TSharedPtr<STextBlock> Title;
            UPROPERTY()
            UJsonObject* JsonObject;

            UPROPERTY()
            UDownloadAsset* DownloadAssetObject;
```

```
private:
		void AddToolbarExtension(FToolBarBuilder& Builder);
		void AddMenuExtension(FMenuBuilder& Builder);

		TSharedRef<class SDockTab> OnSpawnPluginTab(const class FSpawnTabArgs&
		SpawnTabArgs);

		TSharedPtr<class FUICommandList> PluginCommands;



		void RegisterAssetManagerSettings()
		{
			if (ISettingsModule* SettingsModule =
			FModuleManager::GetModulePtr<ISettingsModule>("Settings"))
			{
				SettingsModule->RegisterSettings("Project",
				"AssetManagerSettings", "General",
				LOCTEXT("RuntimeGeneralSettingsName", "General"),
				LOCTEXT("RuntimeGeneralSettingsDescription",
				"Configure 'Asset Manager' settings"),
				GetMutableDefault<UManagerSettings>()
				);
			}
		}

		void UnregisterAssetManagerSettings()
		{
			if (ISettingsModule* SettingsModule =
			FModuleManager::GetModulePtr<ISettingsModule>("Settings"))
			{
				SettingsModule->UnregisterSettings("Project",
				"AssetManagerSettings", "General");
			}
		}
};
```

AssetManagerEditorPlugin.cpp

```
// Copyright 1998-2017 Epic Games, Inc. All Rights Reserved.

#include "AssetManagerEditorPlugin.h"
#include "AssetManagerEditorPluginStyle.h"
#include "AssetManagerEditorPluginCommands.h"
#include "LevelEditor.h"
#include "Widgets/Docking/SDockTab.h"
#include "Widgets/Layout/SBox.h"
#include "Widgets/Text/STextBlock.h"
#include "Widgets/Layout/SScrollBox.h"
#include "SButton.h"
#include "SEditableTextBox.h"
#include "Framework/MultiBox/MultiBoxBuilder.h"

static const FName AssetManagerEditorPluginTabName("AssetManager");
DEFINE_LOG_CATEGORY(LogAssetManagerPlugin);
void FAssetManagerEditorPluginModule::StartupModule()
{
		// This code will execute after your module is loaded into memory; the exact
		timing is specified in the .uplugin file per-module

		FAssetManagerEditorPluginStyle::Initialize();
		FAssetManagerEditorPluginStyle::ReloadTextures();

		FAssetManagerEditorPluginCommands::Register();

		//Register Manager settings
		RegisterAssetManagerSettings();


		//Create JsonObject class
		JsonObject = NewObject<UJsonObject>();
		JsonObject->AddToRoot();

		//Create DownloadAsset class
		DownloadAssetObject = NewObject<UDownloadAsset>();
		DownloadAssetObject->AddToRoot();



		PluginCommands = MakeShareable(new FUICommandList);

		PluginCommands->MapAction(
		FAssetManagerEditorPluginCommands::Get().OpenPluginWindow,
		FExecuteAction::CreateRaw(this,
		&FAssetManagerEditorPluginModule::PluginButtonClicked),
		FCanExecuteAction());

		FLevelEditorModule& LevelEditorModule =
		FModuleManager::LoadModuleChecked<FLevelEditorModule>("LevelEditor");
		{		TSharedPtr<FExtender> MenuExtender = MakeShareable(new
		FExtender());
		MenuExtender->AddMenuExtension("WindowLayout",
		EExtensionHook::After, PluginCommands,
		FMenuExtensionDelegate::CreateRaw(this,
		&FAssetManagerEditorPluginModule::AddMenuExtension));

		LevelEditorModule.GetMenuExtensibilityManager()->AddExtender(MenuExtender);
		}

		{
		TSharedPtr<FExtender> ToolbarExtender = MakeShareable(new FExtender);
```

```
//Button
                                                    + SVerticalBox::Slot()
                                                    .HAlign(HAlign_Center)
                                                    .VAlign(VAlign_Center)
                                                    .AutoHeight()
                                                    [
                                                    SAssignNew(Send, SButton)
                                                    .Text(SendButtonText)
                                                    .OnClicked(FOnClicked::CreateRaw(this,
                                                    &FAssetManagerEditorPluginModule::SearchBu
                                                    ttonClicked))
                                                    ]
                                        ]
                              ];
}

FReply FAssetManagerEditorPluginModule::SearchButtonClicked()
{
            CleanInfoBox();

            CreateJsonRequest();

            return FReply::Handled();
}

FReply FAssetManagerEditorPluginModule::DownloadButtonClicked(FString AssetId)
{
            DownloadAssetObject->DownloadCompletedFiles(AssetId);
            return FReply::Handled();
}

void FAssetManagerEditorPluginModule::CreateJsonRequest()
{
            TSharedRef<IHttpRequest> HttpRequest = FHttpModule::Get().CreateRequest();
            HttpRequest->SetHeader(TEXT("X-APIKEY"), JsonObject->APIKey);
            HttpRequest->SetHeader(TEXT("Content-Type"), TEXT("application/x-www-form-
            urlencoded"));
            HttpRequest->SetURL(JsonObject->ListAssetsUrl);
            HttpRequest->SetVerb(TEXT("POST"));

            HttpRequest->SetContentAsString("project_slug=" + JsonObject->ProjectSlug);
            HttpRequest->OnProcessRequestComplete().BindRaw(this,
            &FAssetManagerEditorPluginModule::ResponseComplete);
            HttpRequest->ProcessRequest();
}

void FAssetManagerEditorPluginModule::ResponseComplete(FHttpRequestPtr Request,
FHttpResponsePtr Response, bool bWasSuccessful)
{
            JsonObject->ResponseComplete(Request, Response, bWasSuccessful);
            if (Response.IsValid())
            {
                        StatusText->SetText(FText::FromString(FString::FromInt(Response->
                        GetResponseCode())));
            }
```

```
		for (int i = 0; i < JsonObject->JsonArrayNum; i +=2)
		{

			FString AssetId = JsonObject->JsonDataArray[i+1];

			Boxi->AddSlot()
				.HAlign(HAlign_Left)
				.VAlign(VAlign_Bottom)
				[
					SAssignNew(Title, STextBlock)
					.Text(FText::FromString(JsonObject->
					JsonDataArray[i]))
				];
			Boxi->AddSlot()
				.HAlign(HAlign_Left)
				.VAlign(VAlign_Bottom)
				.AutoHeight()
				[
					SAssignNew(Download, SButton)
					.Text(FText::FromString(TEXT("Download")))
					.OnClicked(FOnClicked::CreateRaw(this, &
					FAssetManagerEditorPluginModule::
					DownloadButtonClicked, AssetId))
				];
		}


		//UE_LOG(LogAssetManagerPlugin, Warning, TEXT("asda %s"), );
}

void FAssetManagerEditorPluginModule::CleanInfoBox()
{
		JsonObject->CleanJsonData();
		//InfoBox->SetText(FText::FromString(""));
}

void FAssetManagerEditorPluginModule::PluginButtonClicked()
{
		FGlobalTabmanager::Get()->InvokeTab(AssetManagerEditorPluginTabName);
}

void FAssetManagerEditorPluginModule::AddMenuExtension(FMenuBuilder& Builder)
{
		Builder.AddMenuEntry(FAssetManagerEditorPluginCommands::Get().OpenPluginWindow)
;
}

void FAssetManagerEditorPluginModule::AddToolbarExtension(FToolBarBuilder& Builder)
{
		Builder.AddToolBarButton(FAssetManagerEditorPluginCommands::Get().OpenPluginWin
dow);
}

#undef LOCTEXT_NAMESPACE

IMPLEMENT_MODULE(FAssetManagerEditorPluginModule, AssetManagerEditorPlugin)
```

```
		ToolbarExtender->AddToolBarExtension("Settings", EExtensionHook::After,
		PluginCommands, FToolBarExtensionDelegate::CreateRaw(this,
		&FAssetManagerEditorPluginModule::AddToolbarExtension));

		LevelEditorModule.GetToolBarExtensibilityManager()->
		AddExtender(ToolbarExtender);
		}

		FGlobalTabmanager::Get()-
		>RegisterNomadTabSpawner(AssetManagerEditorPluginTabName,
		FOnSpawnTab::CreateRaw(this,
		&FAssetManagerEditorPluginModule::OnSpawnPluginTab))
		.SetDisplayName(LOCTEXT("FAssetManagerEditorPluginTabTitle", "AssetManager"))
		.SetMenuType(ETabSpawnerMenuType::Hidden);
}

void FAssetManagerEditorPluginModule::ShutdownModule()
{
		// This function may be called during shutdown to clean up your module.  For
		modules that support dynamic reloading,
		// we call this function before unloading the module.
		FAssetManagerEditorPluginStyle::Shutdown();

		FAssetManagerEditorPluginCommands::Unregister();

		FGlobalTabmanager::Get()-
		>UnregisterNomadTabSpawner(AssetManagerEditorPluginTabName);

		//Unload AssetManager settings
		UnregisterAssetManagerSettings();
}

TSharedRef<SDockTab> FAssetManagerEditorPluginModule::OnSpawnPluginTab(const FSpawnTabArgs&
SpawnTabArgs)
{
		FText SendButtonText = FText::FromString(TEXT("Search"));

		return SAssignNew(AssetManagerTab, SDockTab)
				.TabRole(ETabRole::NomadTab)
				[
						// tab content
						SNew(SBox)
						.HAlign(HAlign_Center)
						.VAlign(VAlign_Center)
						[
								SAssignNew(Boxi, SVerticalBox)
								//Static status Text
								+ SVerticalBox::Slot()
								.HAlign(HAlign_Center)
								.VAlign(VAlign_Center)
								[
								SNew(STextBlock)
								.Text(FText::FromString(TEXT("Status: ")))
								]
								//Status Text
								+ SVerticalBox::Slot()
								.HAlign(HAlign_Center)
								.VAlign(VAlign_Center)
								[
								SAssignNew(StatusText, STextBlock)
								.ColorAndOpacity(FLinearColor::Red)
								]
```

DownloadAsset.h

```
// Fill out your copyright notice in the Description page of Project Settings.

#pragma once

#include "CoreMinimal.h"
#include "UObject/NoExportTypes.h"
#include "Http.h"
#include "ZipFileFunctionLibrary.h"
#include "DownloadAsset.generated.h"

/**
 *
 */
DECLARE_DYNAMIC_MULTICAST_DELEGATE_ThreeParams(FOnProgress, const int32, BytesSent, const
int32, BytesReceived, const int32, ContentLength);

UCLASS()
class ASSETMANAGEREDITORPLUGIN_API UDownloadAsset : public UObject
{
		GENERATED_BODY()

public:
		UDownloadAsset();

		FOnProgress mOnProgress;

		static UDownloadAsset* MakeDownloader();

		void DownloadCompletedFiles(FString id);

		FString AssetId;

private:
		void OnReady(FHttpRequestPtr Request, FHttpResponsePtr Response, bool
bWasSuccessful);
		void OnProgress(FHttpRequestPtr Request, int32 BytesSent, int32 BytesReceived);


};
```

DownloadAsset.cpp

```
void UDownloadAsset::OnReady(FHttpRequestPtr Request, FHttpResponsePtr Response, bool
bWasSuccessful)
{
		RemoveFromRoot();
		Request->OnProcessRequestComplete().Unbind();
		FString FileSavePath = "D:\\TESTI\\AssetManagerFile" + AssetId + ".zip";

		if (Response.IsValid() && EHttpResponseCodes::IsOk(Response->GetResponseCode()))
		{
			// SAVE FILE
			IPlatformFile& PlatformFile =
			FPlatformFileManager::Get().GetPlatformFile();

			// create save directory if not existent
			FString Path, Filename, Extension;
			FPaths::Split(FileSavePath, Path, Filename, Extension);
			if (!PlatformFile.DirectoryExists(*Path))
			{
				if (!PlatformFile.CreateDirectoryTree(*Path))
				{
					return;
				}
			}

			// open/create the file
			IFileHandle* FileHandle = PlatformFile.OpenWrite(*FileSavePath);
			if (FileHandle)
			{
				// write the file
				FileHandle->Write(Response->GetContent().GetData(),
				Response->GetContentLength());

				// Close the file
				delete FileHandle;

				UZipFileFunctionLibrary::UnzipWithLambda(FString("D:/TESTI
				/AssetManagerFile" + AssetId + ".zip"),
				[]()
				{
					nullptr;
					//Called when done
				},
					[](float Percent)
				{
					nullptr;
				});
			}
			else
			{
			}
		}
		else
		{
		}
}
```

```
// Fill out your copyright notice in the Description page of Project Settings.

#include "DownloadAsset.h"
#include "ManagerSettings.h"

UDownloadAsset::UDownloadAsset()
{

}

UDownloadAsset* UDownloadAsset::MakeDownloader()
{
		UDownloadAsset* Downloader = NewObject<UDownloadAsset>();
		return Downloader;
}

void UDownloadAsset::DownloadCompletedFiles(FString id)
{
		AssetId = id;

		FString Url = GetDefault<UManagerSettings>()->AssetManagerUrl;
		FString APIKey = GetDefault<UManagerSettings>()->API_Key;

		FString DownloadAssetUrl = Url + "/api/projects/download-asset/";

		TSharedRef<IHttpRequest> HttpRequest = FHttpModule::Get().CreateRequest();
		HttpRequest->SetHeader(TEXT("X-APIKEY"), APIKey);
		HttpRequest->SetHeader(TEXT("Content-Type"), TEXT("application/x-www-form-
		urlencoded"));
		HttpRequest->SetURL(DownloadAssetUrl);
		HttpRequest->SetVerb(TEXT("POST"));

		HttpRequest->SetContentAsString("id=" + id);
		HttpRequest->OnProcessRequestComplete().BindUObject(this,
		&UDownloadAsset::OnReady);
		HttpRequest->OnRequestProgress().BindUObject(this, &UDownloadAsset::OnProgress);

		// Execute the request
		HttpRequest->ProcessRequest();
}

void UDownloadAsset::OnProgress(FHttpRequestPtr Request, int32 BytesSent, int32 BytesReceived
{
		int32 FullSize = Request->GetContentLength();
		mOnProgress.Broadcast(BytesSent, BytesReceived, FullSize);
}
```

JsonObject.h

```
// Fill out your copyright notice in the Description page of Project Settings.
#pragma once

#include "CoreMinimal.h"
#include "Http.h"
#include "Json.h"
#include "JsonObject.generated.h"

class AssetManagerEditorPlugin;

enum class EJsonResultResult : uint8
{
	Success,
	DownloadFailed,
	HTTPRequestFailed,
	SaveFailed
};

UCLASS(	)
class ASSETMANAGEREDITORPLUGIN_API UJsonObject : public UObject
{
	GENERATED_UCLASS_BODY()

public:
	UJsonObject();

	EJsonResultResult JsonResult;

	UPROPERTY()
	FString ListAssetsUrl;
	UPROPERTY()
	FString FileSavePath;
	UPROPERTY()
	FString ListChangedAssetsUrl;
	UPROPERTY()
	FString AssetManagerUrl;
	UPROPERTY()
	FString APIKey;
	UPROPERTY()
	FString ProjectSlug;
	void ResponseComplete(FHttpRequestPtr Request, FHttpResponsePtr Response, bool
	bWasSuccessful);
	TArray<FString> JsonDataArray;
	FString GetStatus();
	int8 JsonArrayNum;
	FString getJsonResponseStatus() const { return JsonResponseStatus; }
	void setJsonResponseStatus(FString val) { JsonResponseStatus = val; }

	void CleanJsonData();

	~UJsonObject();
private:
	UPROPERTY()
	FString JsonData;

	UPROPERTY()
	FString JsonResponseStatus;
};
```

JsonObject.cpp

```
// Fill out your copyright notice in the Description page of Project Settings.

#include "JsonObject.h"
#include "ManagerSettings.h"

UJsonObject::UJsonObject(const class FObjectInitializer& ObjectInitializer)
		: Super(ObjectInitializer)
{
		AssetManagerUrl = GetDefault<UManagerSettings>()->AssetManagerUrl;
		ProjectSlug = GetDefault<UManagerSettings>()->ProjectSlug;

		ListAssetsUrl = AssetManagerUrl + "/api/projects/list-assets/";
		ListChangedAssetsUrl = AssetManagerUrl + "/api/projects/list-changed-assets/";
		APIKey = GetDefault<UManagerSettings>()->API_Key;

		FileSavePath = "";
}

void UJsonObject::ResponseComplete(FHttpRequestPtr Request, FHttpResponsePtr Response, bool
bWasSuccessful)
{
		if (!Response.IsValid())
		{
				JsonResult = EJsonResultResult::HTTPRequestFailed;
		}
		else
		{
				//Create a pointer to hold the json serialized data
				TSharedPtr<FJsonObject> JsonObject = MakeShareable(new FJsonObject);

				TArray< TSharedPtr < FJsonValue > > JsonArray;

				//Create a reader pointer to read the json data
				TSharedRef<TJsonReader<>> Reader =
				TJsonReaderFactory<>::Create(Response->GetContentAsString());
```

```
//Deserialize the json data given Reader and the actual object to deserialize
                        if (FJsonSerializer::Deserialize(Reader, JsonArray))
                        {
                                TSharedPtr<FJsonValue> ArrayValue;
                                TSharedPtr<FJsonObject> JsonArrayObject;

                                TArray<FString> JsonResultArray;

                                for (int i = 0; i < JsonArray.Num(); i++)
                                {
                                        ArrayValue = JsonArray[i];
                                        JsonArrayObject = ArrayValue->AsObject();

                                        int8 Id = JsonArrayObject->
                                        GetIntegerField("id");
                                        FString Name = JsonArrayObject->
                                        GetStringField("title");
                                        FString Description = JsonArrayObject->
                                        GetStringField("description");
                                        FString Status = JsonArrayObject->
                                        GetStringField("status");

                                        JsonDataArray.Add("Title: " + Name);
                                        JsonDataArray.Add(FString::FromInt(Id));
                                }
                                JsonArrayNum = JsonDataArray.Num();

                                setJsonResponseStatus(GetStatus());

                                JsonResult = EJsonResultResult::Success;
                        }
                }
}

FString UJsonObject::GetStatus()
{
                if (JsonResult == EJsonResultResult::HTTPRequestFailed)
                {
                        return "HTTP Request Failed";
                }
                else if (JsonResult == EJsonResultResult::Success)
                {
                        return "Success";
                }

                return "";
}


UJsonObject::~UJsonObject()
{

}

void UJsonObject::CleanJsonData()
{
                JsonData.Empty();
}
```