1. Интеграция Nuitrack в проект на Unreal Engine 4.10

1.1. Настройка Unreal Engine

Изменить шаблон для Android-приложения, который находится в папке:
c:\Program Files (x86)\Epic Games\4.10\Engine\Build\Android\Java
Для этого нужно:

1) Скопировать **nuitrackhelper.jar** в c:\Program Files (x86)\Epic Games\4.10\Engine\Build\Android\Java\libs\
2) Добавить в **SplashActivity** (c:\Program Files (x86)\Epic Games\4.10\Engine\Build\Android\Java\src\com\epicgames\ue4\SplashActivity.java) инициализацию **Nuitrack** (для использования **Nuitrack API** в нативном коде на C++)
    a) Добавить импорт: **import com.tdv.nuitrack.sdk.Nuitrack;**
    b) Содержимое метода **onCreate** перенести в отдельный метод **private void StartGame();**
    c) Добавить объект **private static Logger Log = new Logger("UE4");**
    d) Добавить метод загрузки Nuitrack:

```
private void LoadNuitrack()
{
	Nuitrack.init(this, new Nuitrack.NuitrackCallback() {
		public void onInitSuccess(Context context) {
			Log.debug( "Nuitrack init SUCCESS in onCreate()");
			StartGame();
		}
		public void onInitFailure(int errorId) {
			Log.debug( "Nuitrack init FAILED in onCreate()");
			finish();
		}
	});
}
```

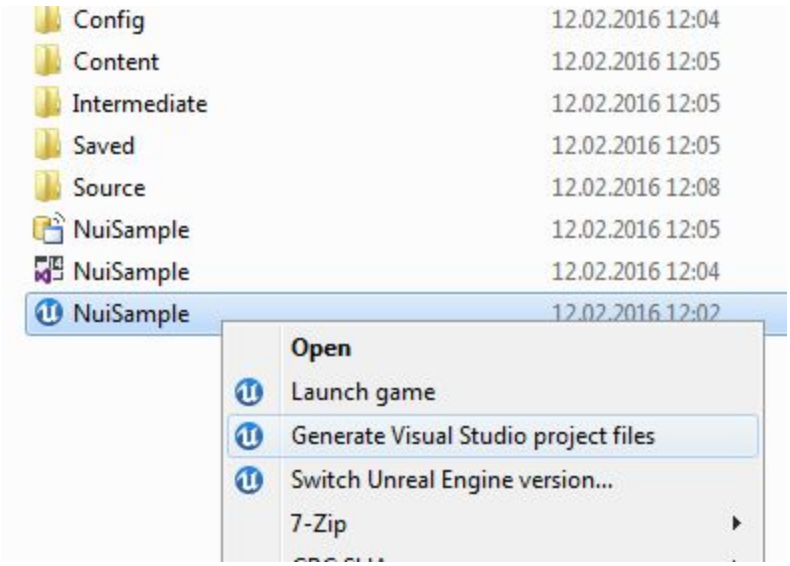
    e) В методе **onCreate** вызвать метод загрузки Nuitrack: **LoadNuitrack();**

Внимание: Все вышеуказанные действия выполняются единоразово. При обновлении Unreal Engine необходимо повторить процедуру настройки.

Опционально:
Если необходимо выключить автоматическое отключение экрана, добавить в **GameActivity** (c:\Program Files (x86)\Epic Games\4.10\Engine\Build\Android\Java\src\com\epicgames\ue4\GameActivity.java) в конец метода **onCreate** вызов **AndroidThunkJava_KeepScreenOn(true);**

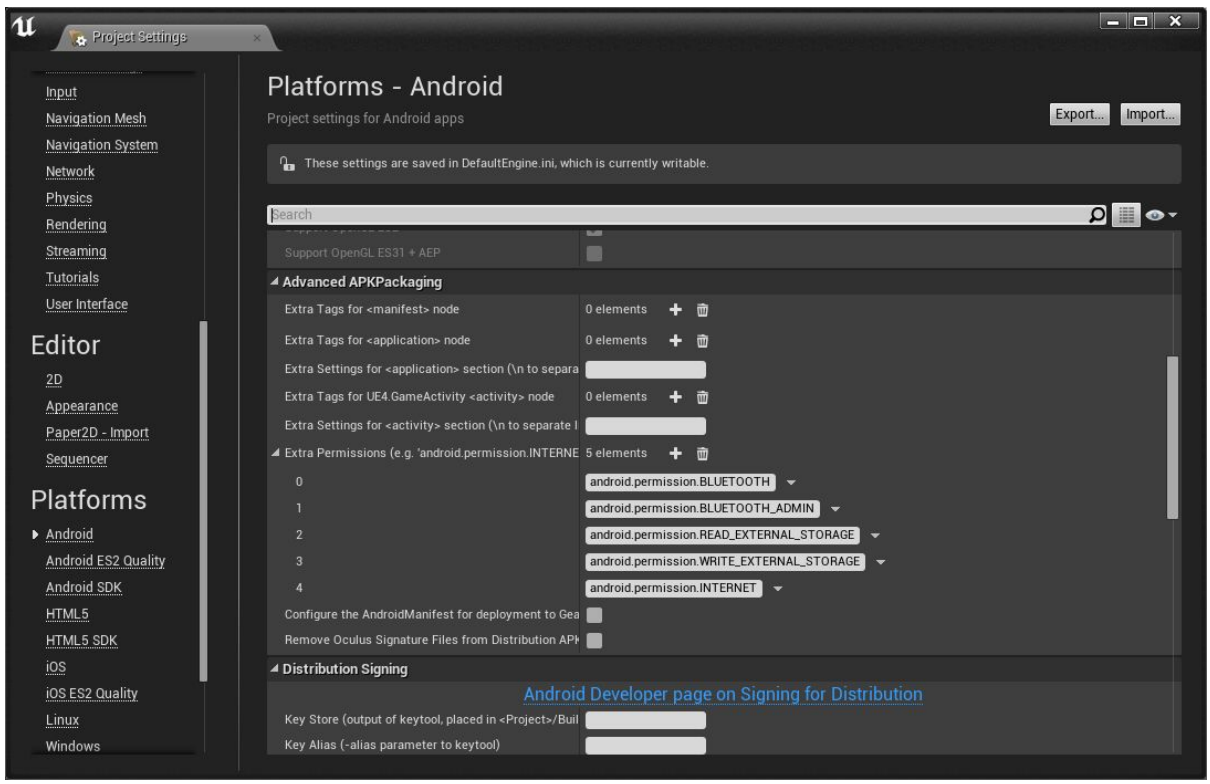
1.2. Настройка проекта

1) Скопировать папку **Nuitrack** в папку проекта в директорию **Source**
2) Сгенерировать файлы проекта для Visual Studio (в контекстном меню **.uproject**-файла)



3) В **Project Settings**, в раздел **Android** добавить **Extra Permissions**:

```
android.permission.BLUETOOTH
android.permission.BLUETOOTH_ADMIN
android.permission.READ_EXTERNAL_STORAGE
android.permission.WRITE_EXTERNAL_STORAGE
android.permission.INTERNET
```



4) Добавить **Nuitrack** в зависимости в файл **Build.cs** (Games/{ProjectName}/Source/{ProjectName}/{ProjectName}.Build.cs) для проекта:

```
PrivateDependencyModuleNames.AddRange(new string[] { "Nuitrack" });
PrivateIncludePaths.AddRange(new string[] {
            Path.GetDirectoryName(RulesCompiler.GetModuleFilename(
                                this.GetType().Name)) + "Nuitrack/Nuitrack/include" });
```

(Класс **Path** требует **using System.IO;**)

1.3. Использование Nuitrack

1) Для инициализации **Nuitrack** и создания трекера скелета необходимо добавить код (например, в метод **BeginPlay()** класса **GameMode** проекта):

```
Nuitrack::init();
SkeletonTracker::Ptr skeletonTracker = SkeletonTracker::create();
skeletonTracker->connectOnUpdate(std::bind(&ANuiSampleGameMode::OnSkeletonUpdate,
                                    this, std::placeholders::_1));
Nuitrack::run();
```

2) При завершении работы нужно уничтожить Nuitrack (например в методе **BeginDestroy()** класса **GameMode** проекта):

```
Nuitrack::release();
```

3) При обновлении кадра необходимо вызывать (например в методе **Tick(float dt)** класса **GameMode** проекта):

```
Nuitrack::update();
```

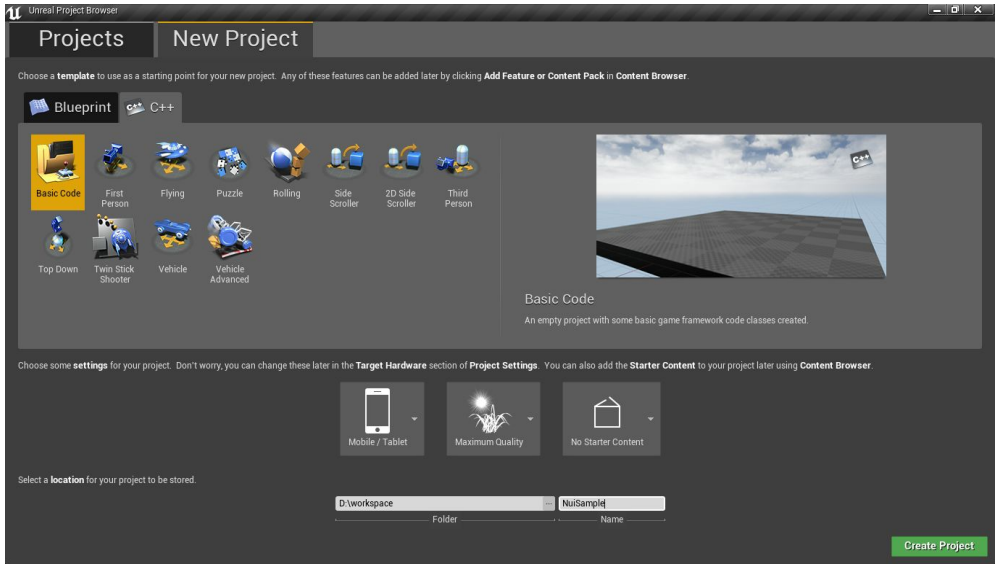
4) Данные скелета будут приходить в методе **OnSkeletonUpdate** (который мы указали при инициализации в **skeletonTracker->connectOnUpdate**):

```
void ANuiSampleGameMode::OnSkeletonUpdate(SkeletonData::Ptr userSkeletons)
{
        auto skeletons = userSkeletons->getSkeletons();

        ...
}
```

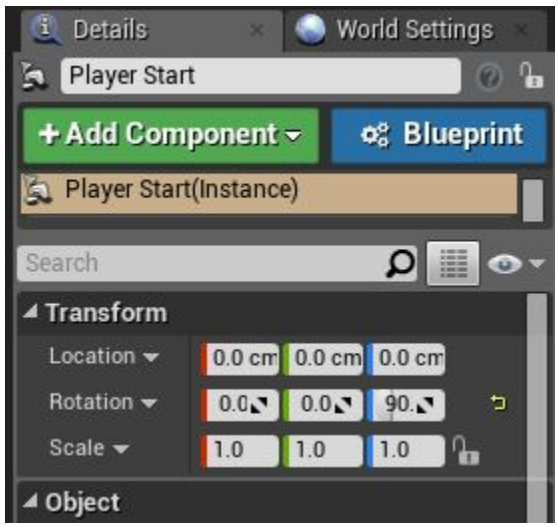
2. Пример проекта с использованием Nuitrack

1) Создать проект

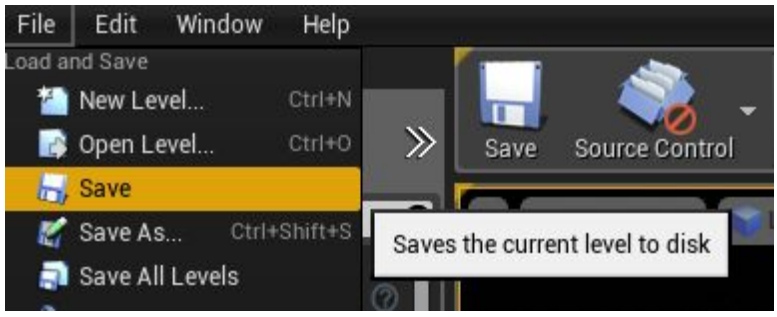


2) Убрать из мира всё, кроме PlayerStart

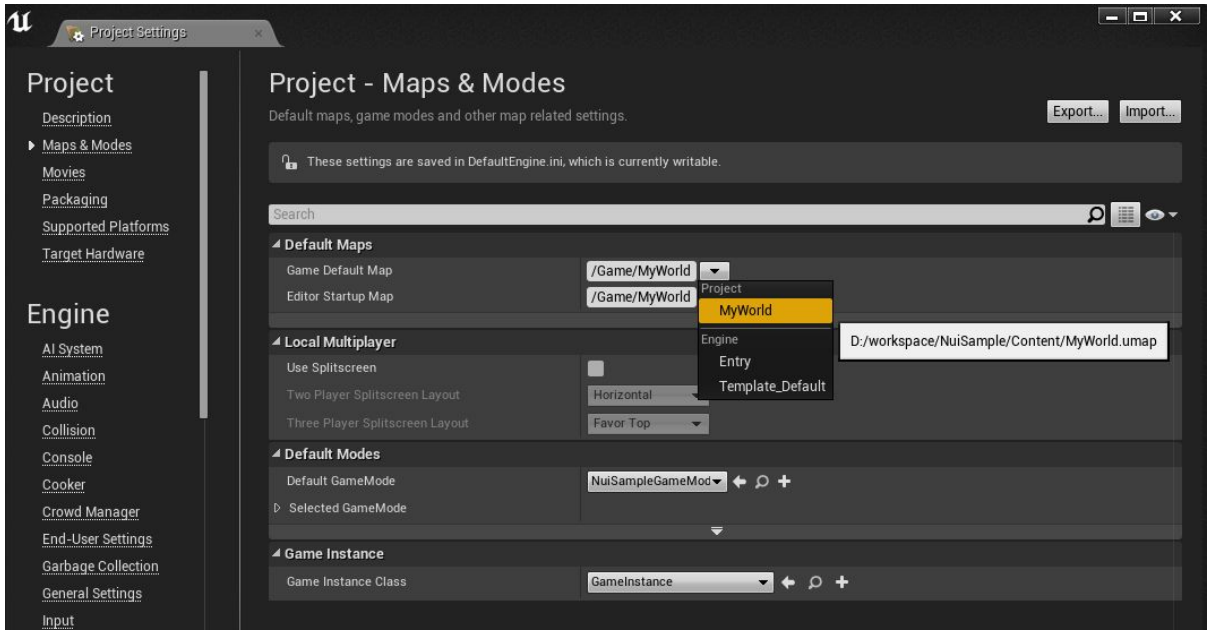
3) Изменить позицию и поворот игрока



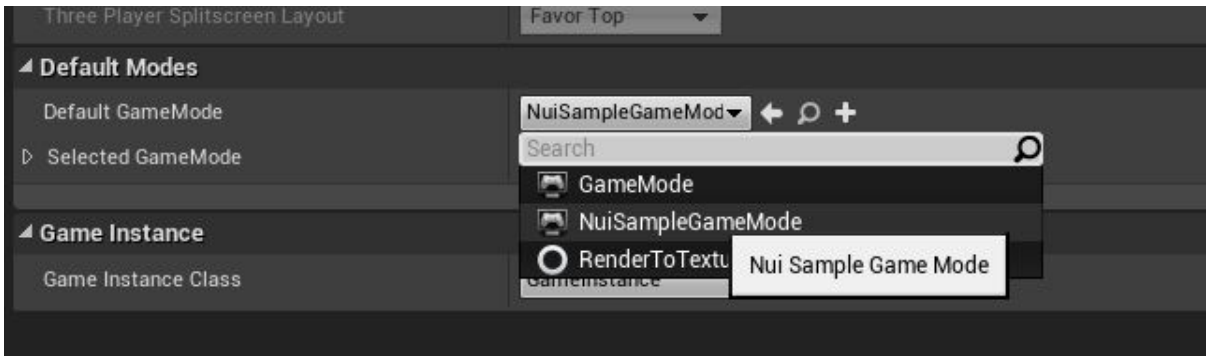
4) Сохранить мир



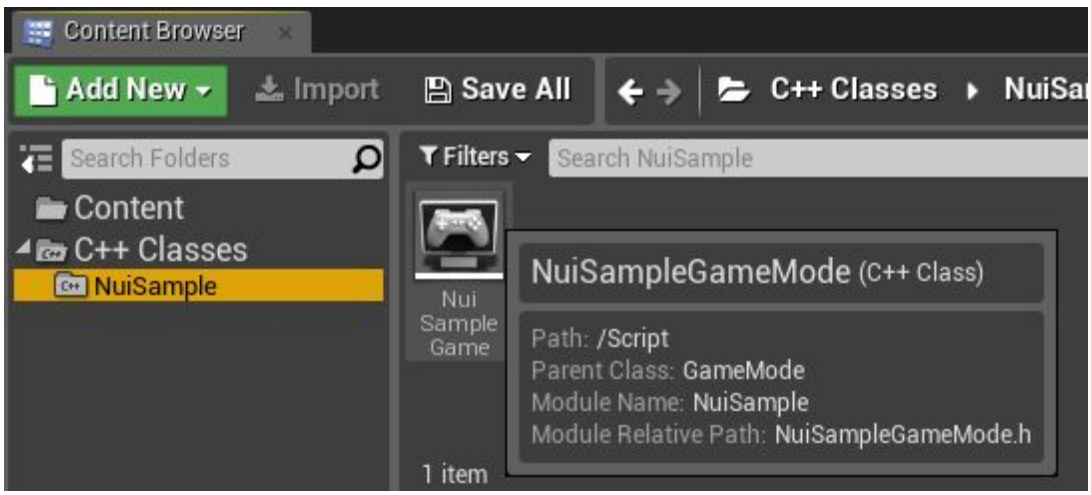
5) Настроить **Project Settings**, раздел **Maps & Modes**:
   a) Установить сохранённый мир в **Game Default Map** и в **Editor Startup Map**



   b) установить **Default GameMode**



6) В **Project Settings**, в раздел **Android** нажать **Configure Now** и ввести Android Package Name
7) Дополнить шаблон Android-приложения для Unreal Engine (смотри пункт 1.1)
8) Настроить проект в Unreal Editor (смотри пункт 1.2)
9) Для редактирования исходного кода нужно открыть **Visual Studio**:



- Файл **NuiSample.Build.cs**:

```
using UnrealBuildTool;
using System.IO;

public class NuiSample : ModuleRules
{
	public NuiSample(TargetInfo Target)
	{
		PublicDependencyModuleNames.AddRange(new string[] { "Core", "CoreUObject",
				"Engine", "InputCore" });

		PrivateDependencyModuleNames.AddRange(new string[] { "Nuitrack" });

		PrivateIncludePaths.AddRange(new string[] { Path.GetDirectoryName(
			RulesCompiler.GetModuleFilename(this.GetType().Name)) +
			 "Nuitrack/Nuitrack/include" });
	}
}
```

- Файл **NuiSample.h**:

```
#pragma once

#include "Engine.h"

#include <iostream>
#include <vector>

using namespace std;

#include "nuitrack/Nuitrack.h"
#include "nuitrack/modules/SkeletonTracker.h"
#include "nuitrack/types/Skeleton.h"
#include "nuitrack/types/SkeletonData.h"

using namespace tdv::nuitrack;
```

- Файл **NuiSample.cpp**:

```
#include "NuiSample.h"

IMPLEMENT_PRIMARY_GAME_MODULE( FDefaultGameModuleImpl, NuiSample, "NuiSample" );
```

- Файл **NuiSampleGameMode.h**:

```
#pragma once

#include "GameFramework/GameMode.h"
#include "NuiSampleGameMode.generated.h"

UCLASS()
class NUISAMPLE_API ANuiSampleGameMode : public AGameMode
{
	GENERATED_BODY()

	UWorld* World;

	ANuiSampleGameMode();

	void Tick(float dt) override;
	void BeginPlay() override;
	void BeginDestroy() override;

	SkeletonTracker::Ptr skeletonTracker;

	void OnSkeletonUpdate(SkeletonData::Ptr userSkeletons);
	void DrawSkeleton(int skeleton_index, vector<Joint> joints);
	void DrawBone(Joint j1, Joint j2);

	static FVector RealToPosition(Vector3 real);
};
```

- Файл **NuiSampleGameMode.cpp**:

```
#include "NuiSample.h"
#include "NuiSampleGameMode.h"

ANuiSampleGameMode::ANuiSampleGameMode()
{
	this->PrimaryActorTick.bCanEverTick = true;
}

void ANuiSampleGameMode::Tick(float dt)
{
	Nuitrack::update();
}

void ANuiSampleGameMode::BeginPlay()
{
	Super::BeginPlay();

	UE_LOG(LogTemp, Warning, TEXT("ANuiSampleGameMode::BeginPlay()"));

	World = GetWorld();

	if (World)
	{
		UE_LOG(LogTemp, Warning, TEXT("Nuitrack::init() CALLING..."));
		Nuitrack::init();

		UE_LOG(LogTemp, Warning, TEXT("SkeletonTracker::create() CALLING..."));
		skeletonTracker = SkeletonTracker::create();

		UE_LOG(LogTemp, Warning, TEXT(
				"skeletonTracker->connectOnUpdate() CALLING..."));
		skeletonTracker->connectOnUpdate(
			std::bind(&ANuiSampleGameMode::OnSkeletonUpdate,
			this, std::placeholders::_1));

		UE_LOG(LogTemp, Warning, TEXT("Nuitrack::run() CALLING..."));
		Nuitrack::run();
	}
	else
	{
		UE_LOG(LogTemp, Error, TEXT("NULL WORLD"));
	}
}

void ANuiSampleGameMode::BeginDestroy()
{
	Super::BeginDestroy();
	Nuitrack::release();
}

void ANuiSampleGameMode::OnSkeletonUpdate(SkeletonData::Ptr userSkeletons)
{
	auto skeletons = userSkeletons->getSkeletons();

	FlushPersistentDebugLines(World);

	if (!skeletons.empty())
	{
		for (auto skeleton : skeletons)
		{
			DrawSkeleton(skeleton.id, skeleton.joints);
		}
	}
}

void ANuiSampleGameMode::DrawSkeleton(int skeleton_index, vector<Joint> joints)
{
	if (joints.empty())
		return;
```

```
        DrawBone(joints[JOINT_HEAD], joints[JOINT_NECK]);
        DrawBone(joints[JOINT_NECK], joints[JOINT_TORSO]);
        DrawBone(joints[JOINT_RIGHT_SHOULDER], joints[JOINT_LEFT_SHOULDER]);
        DrawBone(joints[JOINT_WAIST], joints[JOINT_LEFT_HIP]);
        DrawBone(joints[JOINT_WAIST], joints[JOINT_RIGHT_HIP]);
        DrawBone(joints[JOINT_TORSO], joints[JOINT_WAIST]);
        DrawBone(joints[JOINT_LEFT_SHOULDER], joints[JOINT_LEFT_ELBOW]);
        DrawBone(joints[JOINT_LEFT_ELBOW], joints[JOINT_LEFT_WRIST]);
        DrawBone(joints[JOINT_RIGHT_SHOULDER], joints[JOINT_RIGHT_ELBOW]);
        DrawBone(joints[JOINT_RIGHT_ELBOW], joints[JOINT_RIGHT_WRIST]);
        DrawBone(joints[JOINT_RIGHT_HIP], joints[JOINT_RIGHT_KNEE]);
        DrawBone(joints[JOINT_LEFT_HIP], joints[JOINT_LEFT_KNEE]);
        DrawBone(joints[JOINT_RIGHT_KNEE], joints[JOINT_RIGHT_ANKLE]);
        DrawBone(joints[JOINT_LEFT_KNEE], joints[JOINT_LEFT_ANKLE]);
}

void ANuiSampleGameMode::DrawBone(Joint j1, Joint j2)
{
        DrawDebugLine(World, RealToPosition(j1.real), RealToPosition(j2.real),
                FColor::MakeRedToGreenColorFromScalar((j1.confidence + j2.confidence)*0.5),
                true, -1, 0, 4);
}

FVector ANuiSampleGameMode::RealToPosition(Vector3 real)
{
        return FVector(-real.x, real.z, real.y)*0.1f;
}
```