1. Nuitrack integration guide for Unreal Engine 4.10

1.1. Setting up Unreal Engine for VicoVR

Change the template of Android app . Typically path to template folder is:
c:\Program Files (x86)\Epic Games\4.10\Engine\Build\Android\Java
Actions needed::

1) Copy **nuitrackhelper.jar** file from SDK to **<template_folder>\libs**
2) Add Nuitrack initialization into **SplashActivity** ({TemplateFolder}\src\com\epicgames\ue4\SplashActivity.java)
    a) Add import at the beginning of the file: **import com.tdv.nuitrack.sdk.Nuitrack;**
    b) Move **onCreate** content to the new method: **private void StartGame();**
    c) Create class field for logger: **private static Logger Log = new Logger("UE4");**
    d) Create Nuitrack loading method:

```
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();
                }
        });
}
```

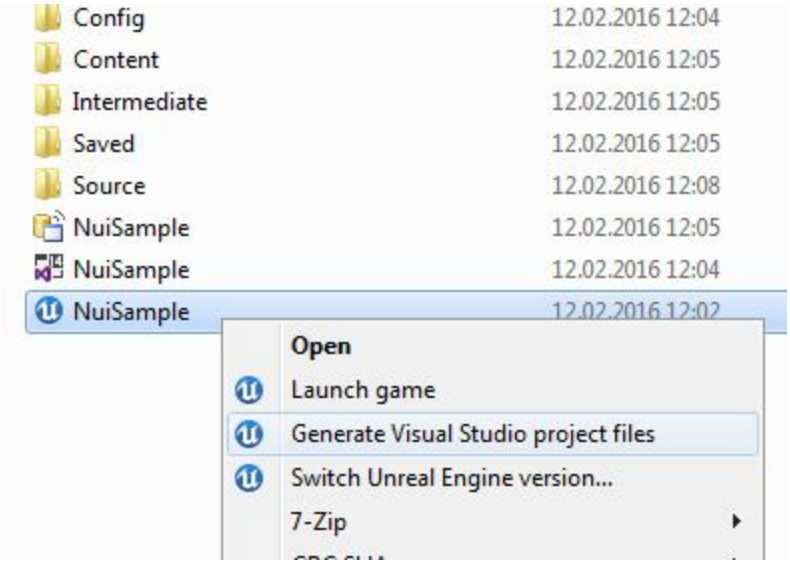
    e) Call **LoadNuitrack();** method in **onCreate** method.

Warning: You need to do UE4 setup for VicoVR only once until you have update for UE4 or VicoVR SDK - then you will need to repeat the adjustment procedure.

Optional: If you want to prevent automatic screen lock in your applications, add **AndroidThunkJava_KeepScreenOn(true);** line to the end of **onCreate** method in **GameActivity** ({TemplateFolder}\src\com\epicgames\ue4\GameActivity.java)

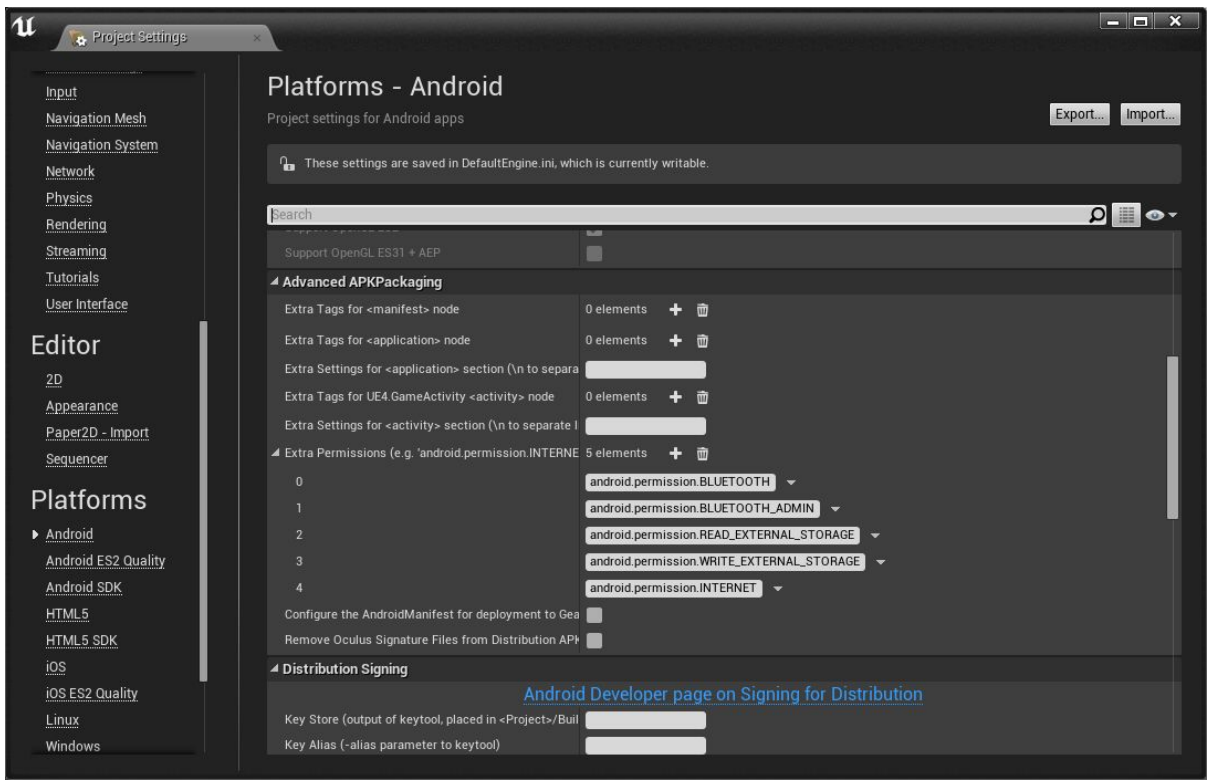
1.2. Setting up your project

1) Copy **Nuitrack** folder from SDK to project **Source** folder.
2) Generate Visual Studio project files (using Windows Explorer context menu option on **.uproject**-file)



3) Add **Extra Permissions** in **Project Settings -> Platforms -> Android** :

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



4) Add Nuitrack libraries and headers to project dependences file **Build.cs** (Games/{ProjectName}/Source/{ProjectName}/{ProjectName}.Build.cs):

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

(**Path** class requires **using System.IO;**)

1.3. Nuitrack usage

1) For Nuitrack initialization, skeleton tracker creation and event subscription you need to use the following code (you can put it to **BeginPlay()** method of **GameMode** class):

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

2) For Nuitrack terminating call **release()** method (you can put it to **BeginDestroy()** method of **GameMode** class):

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

3) For raising events with new data from modules call **update()** method (you can put it to **Tick(float dt)** method of **GameMode** class):

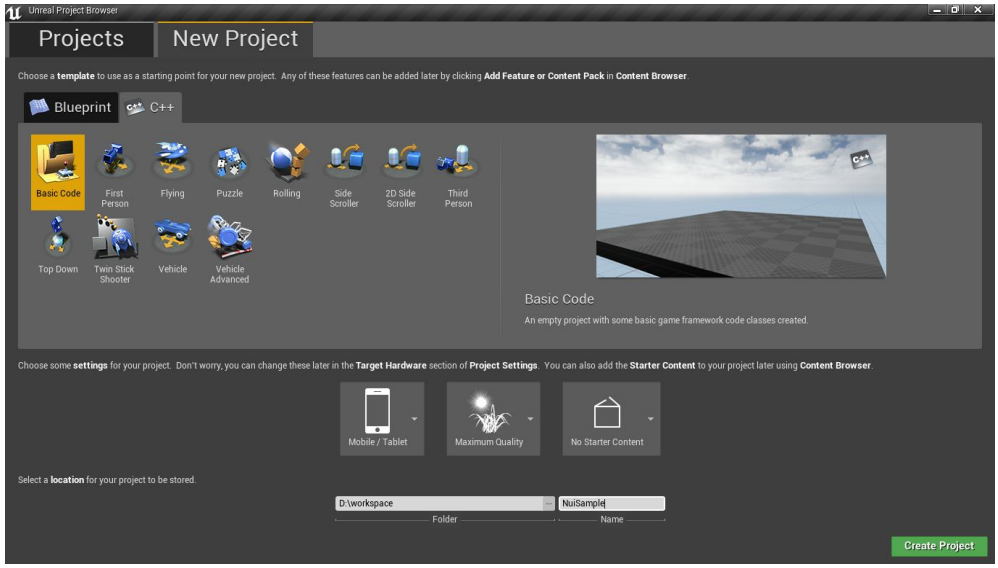
```
Nuitrack::update();
```

4) Skeleton tracking data will be available inside **OnSkeletonUpdate** method (was subscribed to OnUpdate event from SkeletonTracker):

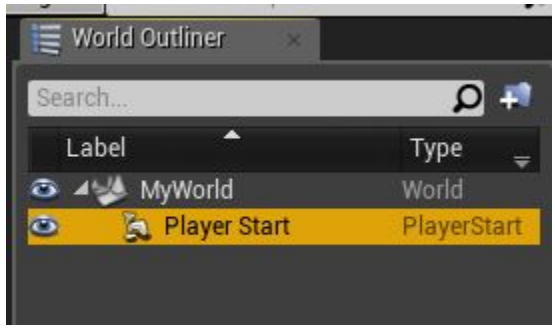
```
void ANuiSampleGameMode::OnSkeletonUpdate(SkeletonData::Ptr userSkeletons)
{
        auto skeletons = userSkeletons->getSkeletons();

        ...
}
```

2. Nuitrack sample project

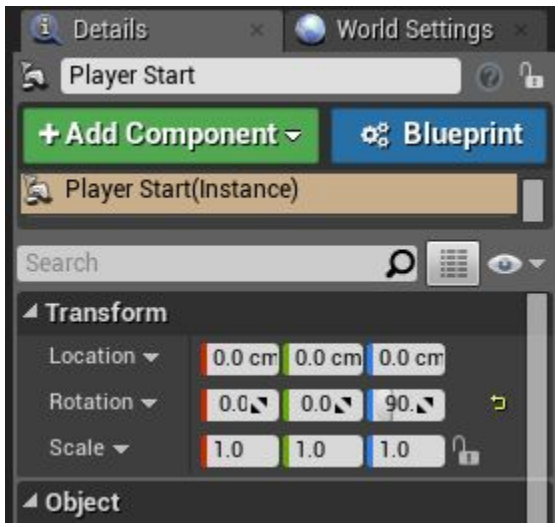
1) Create new project.



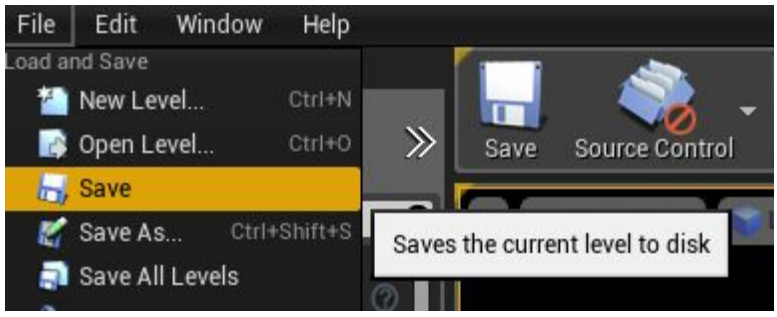
2) Delete everything except PlayerStart from the World.



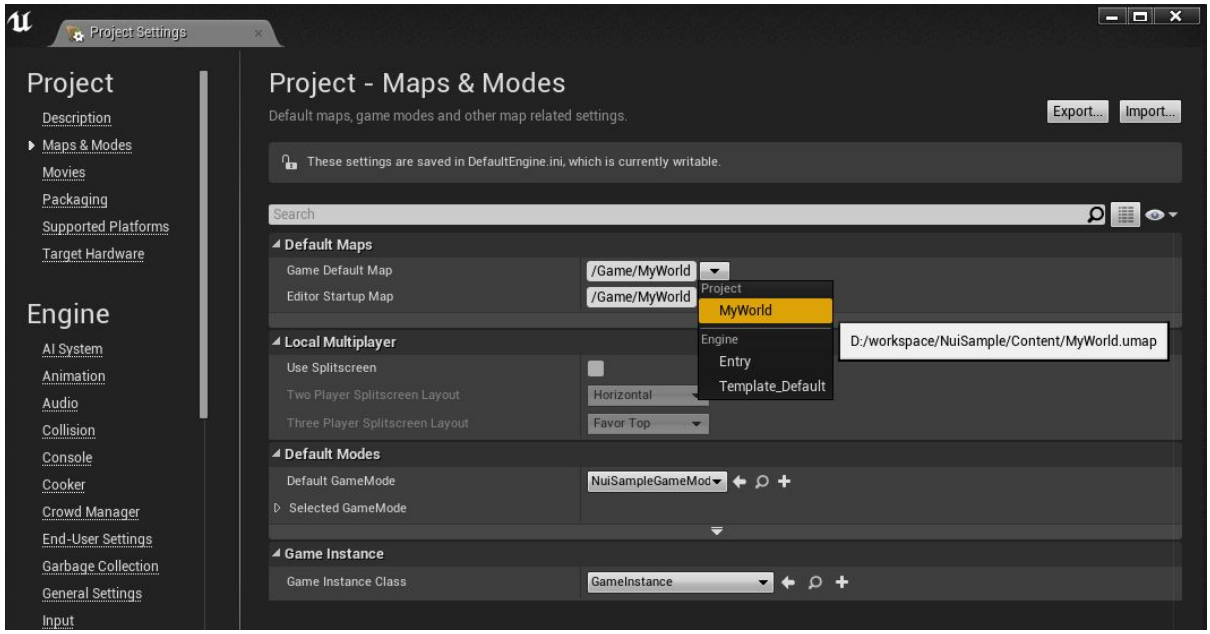
3) Set player locations and rotation as on screenshot.



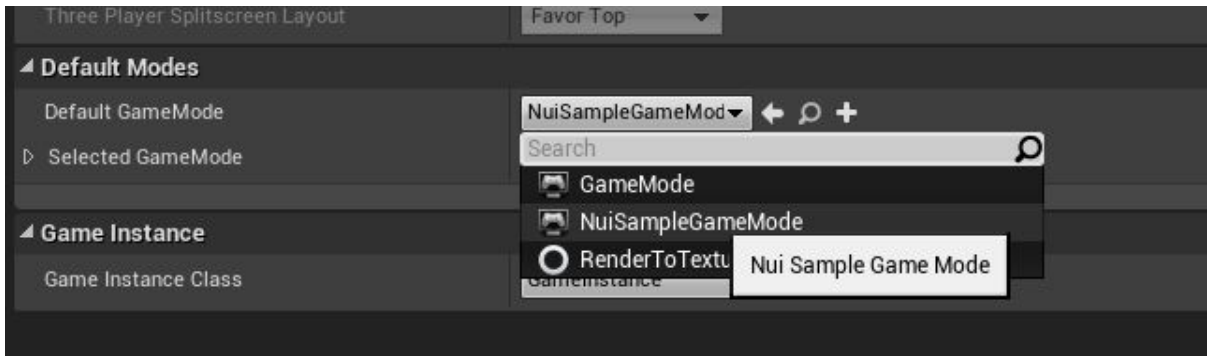
4) Save current World.



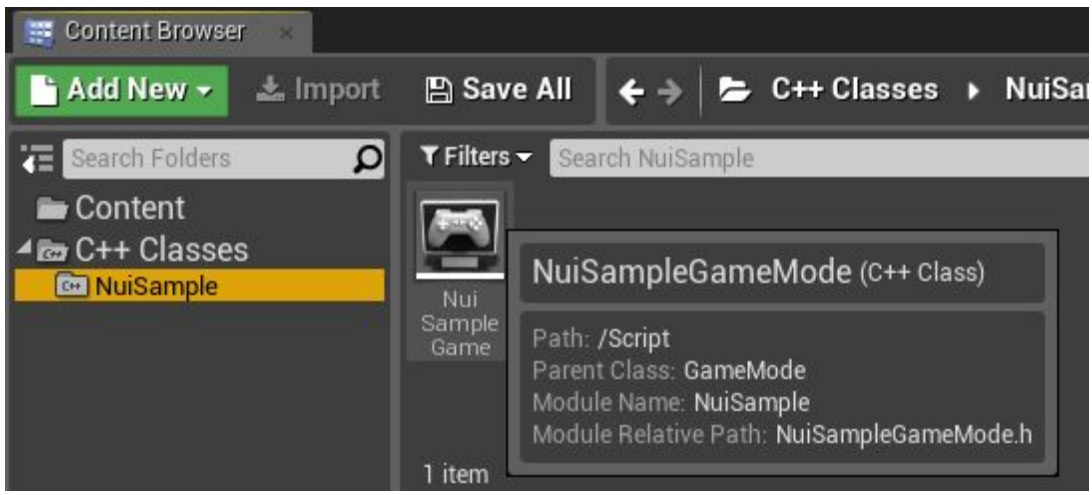
5) In **Project Settings -> Maps & Modes**:
   a) Set your World as **Game Default Map** and **Editor Startup Map**.



   b) Set **Default GameMode**.



6) In **Project Settings -> Android** press **Configure Now** and fill Android Package Name field.
7) Setup Android Unreal Engine for VicoVR (as it was described in point 1.1)
8) Set up your project (as it was described in point 1.2)
9) Open **Visual Studio** for writing C++ code:



- **NuiSample.Build.cs** File:

```
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" });
	}
}
```

- **NuiSample.h** File:

```
#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;
```

- **NuiSample.cpp** File:

```
#include "NuiSample.h"

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

- **NuiSampleGameMode.h** File:

```
#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);
};
```

- **NuiSampleGameMode.cpp** File:

```
#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;
}
```