1. **Download Unity 2018.3.14f1 https://store.unity.com/download**

2. **Download and install Visual Studio Code**

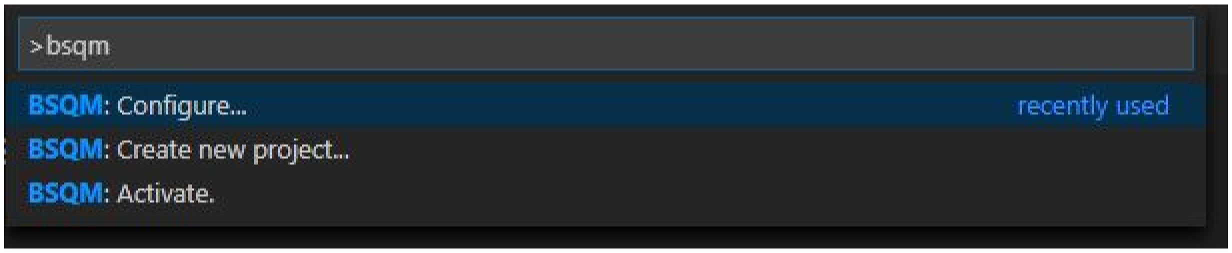
   https://code.visualstudio.com/download

3. **Open Visual Studio Code and setup**

4. **Download and install BSQM**

   https://marketplace.visualstudio.com/items?itemName=raftario.bsqm

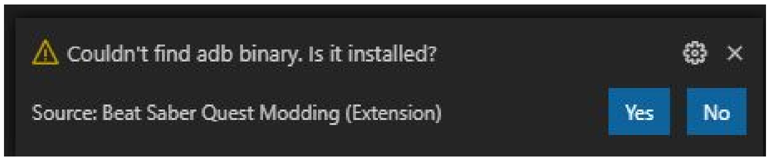
5. **Configure BSQM**



```
>bsqm
BSQM: Configure...                    recently used
BSQM: Create new project...
BSQM: Activate.
```

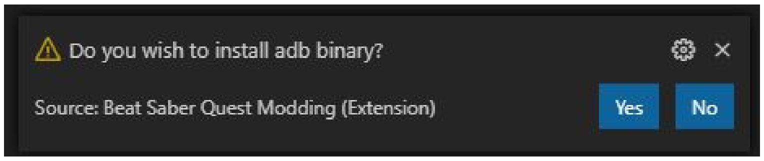
Ctrl+Shift+P
bsqm.configure
Enter

You might see this message…



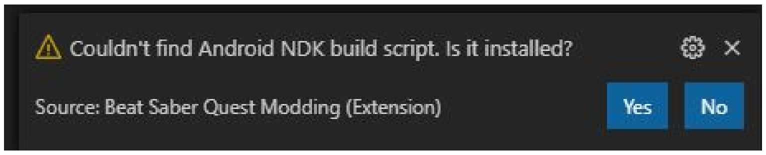
```
Couldn't find adb binary. Is it installed?
Source: Beat Saber Quest Modding (Extension)        Yes   No
```

If you know where it is, click yes to choose the directory. If not, click no to download.

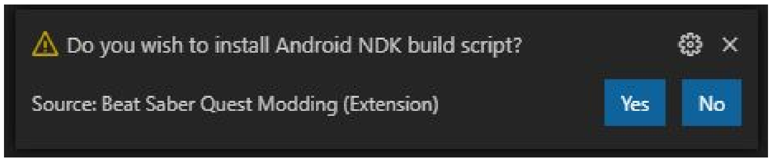


```
Do you wish to install adb binary?
Source: Beat Saber Quest Modding (Extension)        Yes   No
```

Same again with the NDK build script



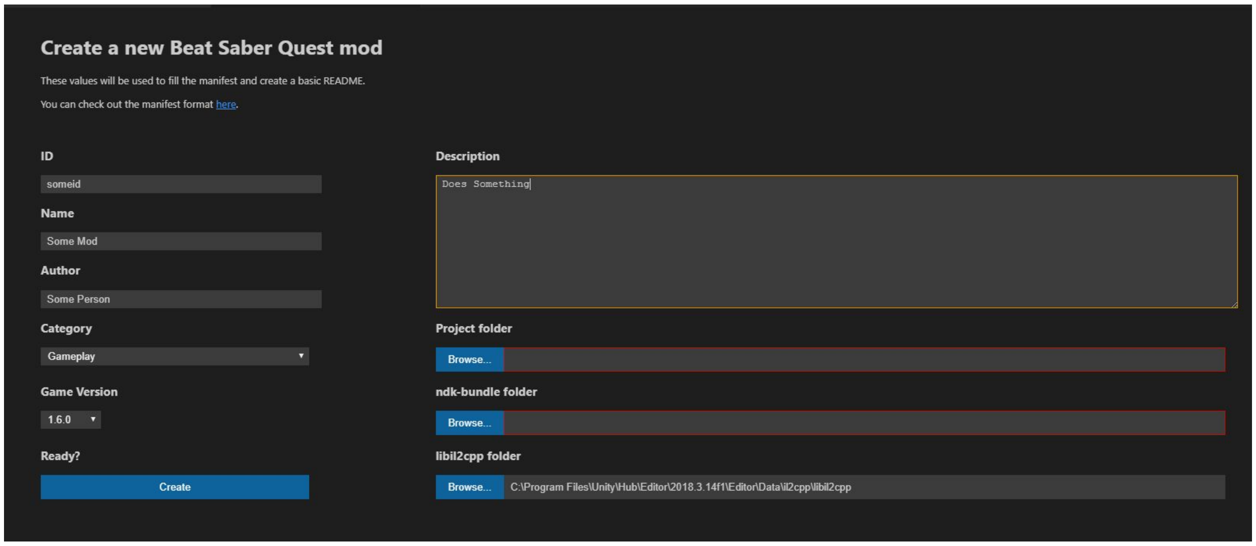
```
Couldn't find Android NDK build script. Is it installed?
Source: Beat Saber Quest Modding (Extension)        Yes   No
```


Do you wish to install Android NDK build script?

Source: Beat Saber Quest Modding (Extension)

6. **Create a BSQM project**

   Ctrl+Shift+P
   bsqm.create
   Enter


Create a new Beat Saber Quest mod

These values will be used to fill the manifest and create a basic README.

You can check out the manifest format here.

ID: someid

Name: Some Mod

Author: Some Person

Category: Gameplay

Game Version: 1.6.0

Description: Does Something

Project folder:

ndk-bundle folder:

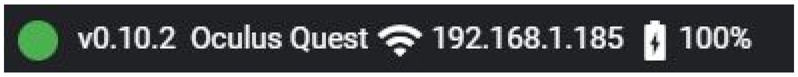
libil2cpp folder: C:\Program Files\Unity\Hub\Editor\2018.3.14f1\Editor\Data\il2cpp\libil2cpp

   Fill out the
   - ID: Something to identify the mod
   - Name: The mod's name
   - Author: your name or username
   - Category: Gameplay for this tutorial
   - Game version: IMPORTANT use that latest version. We will change it to your game version later
   - Project folder: Directory to a folder where the files will be generated
   - NDK folder: MAKE SURE IT IS FILLED OUT. If it isn't, browse and select the directory.

   Click Create.
7. **Get the APK**

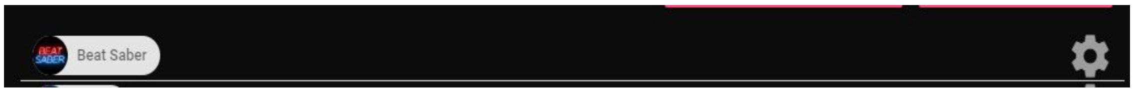
   Download Sidequest https://sidequestvr.com/setup-howto

   Connect your Quest

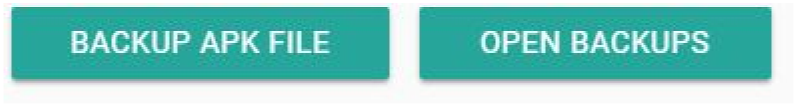

v0.10.2 Oculus Quest 192.168.1.185 100%

   Click the grid

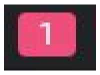
Click the gear next to Beat Saber



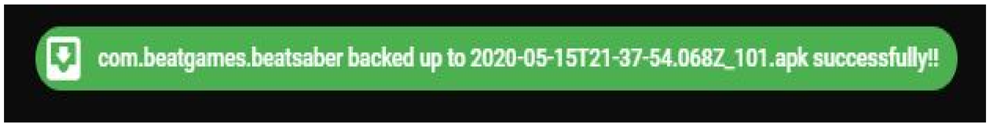
Click Backup Apk File



You can click the number to see the progress

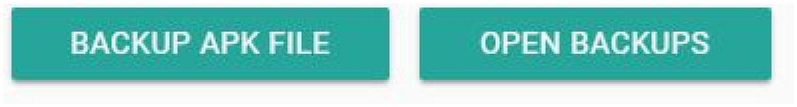


When you see this...



com.beatgames.beatsaber backed up to 2020-05-15T21-37-54.068Z_101.apk successfully!!

… then click the grid and gear again, and this time click Open Backups

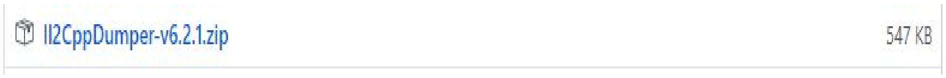


Unzip the file with WinRar or something else
Copy the following into a new folder:

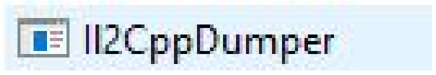
- */assets/bin/Data/Managed/global-metadata.dat*
- */lib/arm64-v8a/libil2cpp.so*

8. Extract the DLLs

Download il2cppdumper https://github.com/Perfare/Il2CppDumper/releases
Make sure it is the non-netcore version



Extract and run the il2cppdumper application

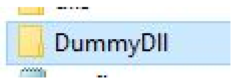


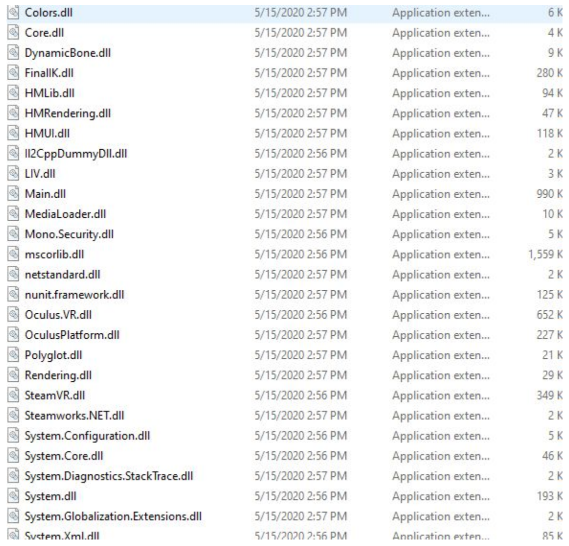
Open the two files from the apk in the FOLLOWING ORDER

1. libil2cpp.so
2. global-metadata.dat

Wait patiently until the program says “Press any key to exit”, then do so.

Your dlls will be in the DummyDLL folder.



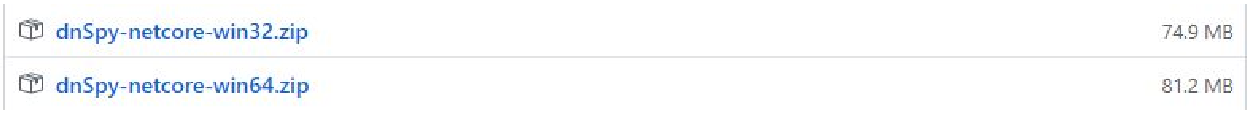


Colors.dll	5/15/2020 2:57 PM	Application exten...	6 K
Core.dll	5/15/2020 2:57 PM	Application exten...	4 K
DynamicBone.dll	5/15/2020 2:57 PM	Application exten...	9 K
FinalIK.dll	5/15/2020 2:57 PM	Application exten...	280 K
HMLib.dll	5/15/2020 2:57 PM	Application exten...	94 K
HMRendering.dll	5/15/2020 2:57 PM	Application exten...	47 K
HMUI.dll	5/15/2020 2:57 PM	Application exten...	118 K
Il2CppDummyDll.dll	5/15/2020 2:56 PM	Application exten...	2 K
LIV.dll	5/15/2020 2:57 PM	Application exten...	3 K
Main.dll	5/15/2020 2:57 PM	Application exten...	990 K
MediaLoader.dll	5/15/2020 2:57 PM	Application exten...	10 K
Mono.Security.dll	5/15/2020 2:56 PM	Application exten...	5 K
mscorlib.dll	5/15/2020 2:56 PM	Application exten...	1,559 K
netstandard.dll	5/15/2020 2:57 PM	Application exten...	2 K
nunit.framework.dll	5/15/2020 2:57 PM	Application exten...	125 K
Oculus.VR.dll	5/15/2020 2:56 PM	Application exten...	652 K
OculusPlatform.dll	5/15/2020 2:57 PM	Application exten...	227 K
Polyglot.dll	5/15/2020 2:57 PM	Application exten...	21 K
Rendering.dll	5/15/2020 2:57 PM	Application exten...	29 K
SteamVR.dll	5/15/2020 2:56 PM	Application exten...	349 K
Steamworks.NET.dll	5/15/2020 2:57 PM	Application exten...	2 K
System.Configuration.dll	5/15/2020 2:56 PM	Application exten...	5 K
System.Core.dll	5/15/2020 2:56 PM	Application exten...	46 K
System.Diagnostics.StackTrace.dll	5/15/2020 2:57 PM	Application exten...	2 K
System.dll	5/15/2020 2:56 PM	Application exten...	193 K
System.Globalization.Extensions.dll	5/15/2020 2:57 PM	Application exten...	2 K
System.Xml.dll	5/15/2020 2:56 PM	Application exten...	85 K

9. **Open the DLLs**

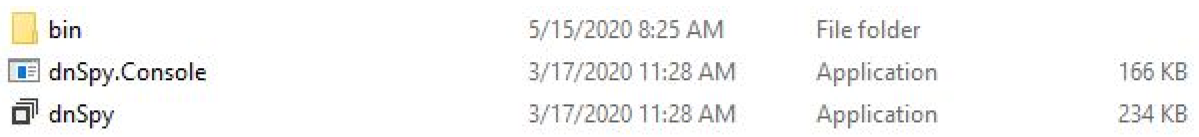
Download dnSpy https://github.com/0xd4d/dnSpy/releases

I recommend one of these, as it doesn’t require NET core to be installed.



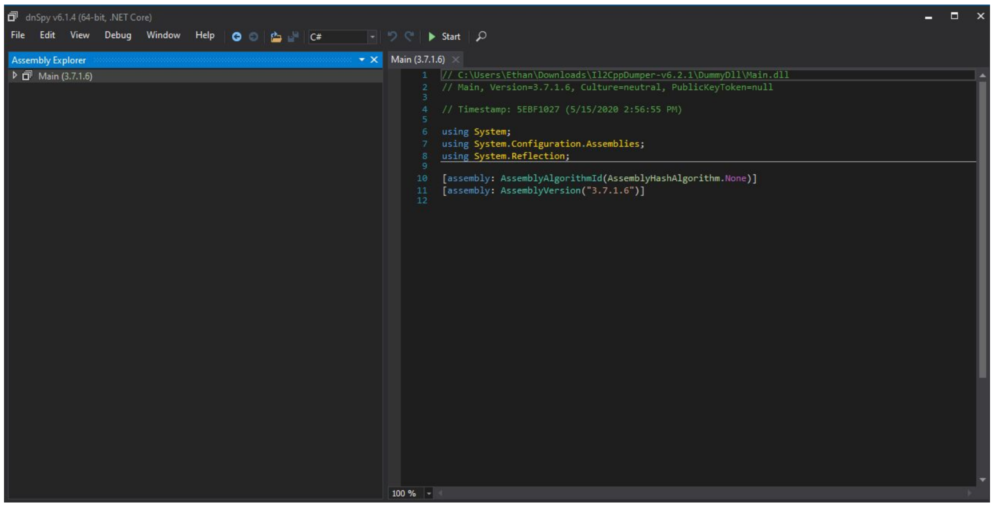
dnSpy-netcore-win32.zip	74.9 MB
dnSpy-netcore-win64.zip	81.2 MB

Open dnSpy (not dnSpy.console)



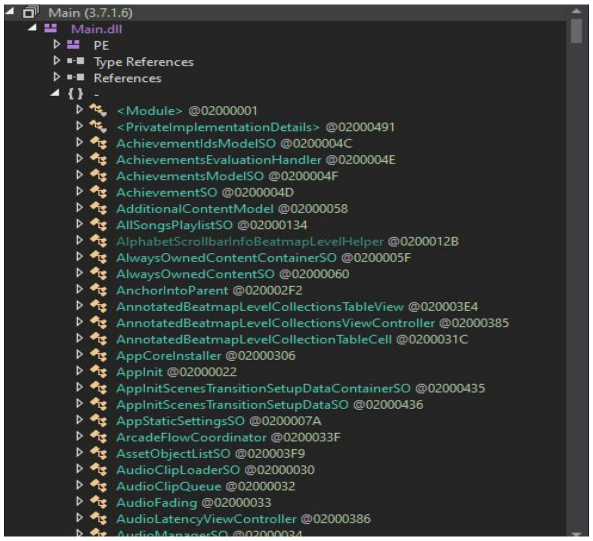
bin	5/15/2020 8:25 AM	File folder	
dnSpy.Console	3/17/2020 11:28 AM	Application	166 KB
dnSpy	3/17/2020 11:28 AM	Application	234 KB

Drag Main.dll (located in DummyDLL) onto the left of dnSpy


dnSpy v6.1.4 (64-bit, .NET Core)
File Edit View Debug Window Help
C#
Start
Assembly Explorer
Main (3.7.1.6)
Main (3.7.1.6)
100 %

```
// C:\Users\Ethan\Downloads\Il2CppDumper-v6.2.1\DummyDll\Main.dll
// Main, Version=3.7.1.6, Culture=neutral, PublicKeyToken=null

// Timestamp: 5EBF1027 (5/15/2020 2:56:55 PM)

using System;
using System.Configuration.Assemblies;
using System.Reflection;

[assembly: AssemblyAlgorithmId(AssemblyHashAlgorithm.None)]
[assembly: AssemblyVersion("3.7.1.6")]
```

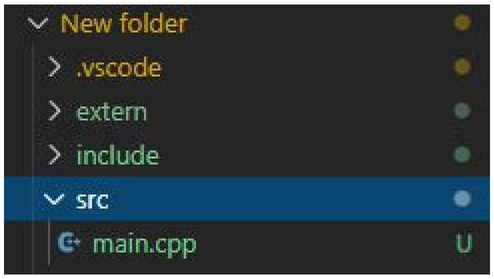
That's it! All the Hook methods are in the tree!



```
Main (3.7.1.6)
  Main.dll
    PE
    Type References
    References
    { } -
      <Module> @02000001
      <PrivateImplementationDetails> @02000491
      AchievementIdsModelSO @0200004C
      AchievementsEvaluationHandler @0200004E
      AchievementsModelSO @0200004F
      AchievementSO @0200004D
      AdditionalContentModel @02000058
      AllSongsPlaylistSO @02000134
      AlphabetScrollbarInfoBeatmapLevelHelper @0200012B
      AlwaysOwnedContentContainerSO @0200005F
      AlwaysOwnedContentSO @02000060
      AnchorIntoParent @020002F2
      AnnotatedBeatmapLevelCollectionsTableView @020003E4
      AnnotatedBeatmapLevelCollectionsViewController @02000385
      AnnotatedBeatmapLevelCollectionTableCell @0200031C
      AppCoreInstaller @02000306
      AppInit @02000022
      AppInitScenesTransitionSetupDataContainerSO @02000435
      AppInitScenesTransitionSetupDataSO @02000436
      AppStaticSettingsSO @0200007A
      ArcadeFlowCoordinator @0200033F
      AssetObjectListSO @020003F9
      AudioClipLoaderSO @02000030
      AudioClipQueue @02000032
      AudioFading @02000033
      AudioLatencyViewController @02000386
```

10. Let's Make a New UI

Now we will make an example mod this mod will spawn particle systems behind the player pointing forward, whenever they miss.
Open Visual Studio Code and navigate to *(your mod folder)/src/main.cpp*



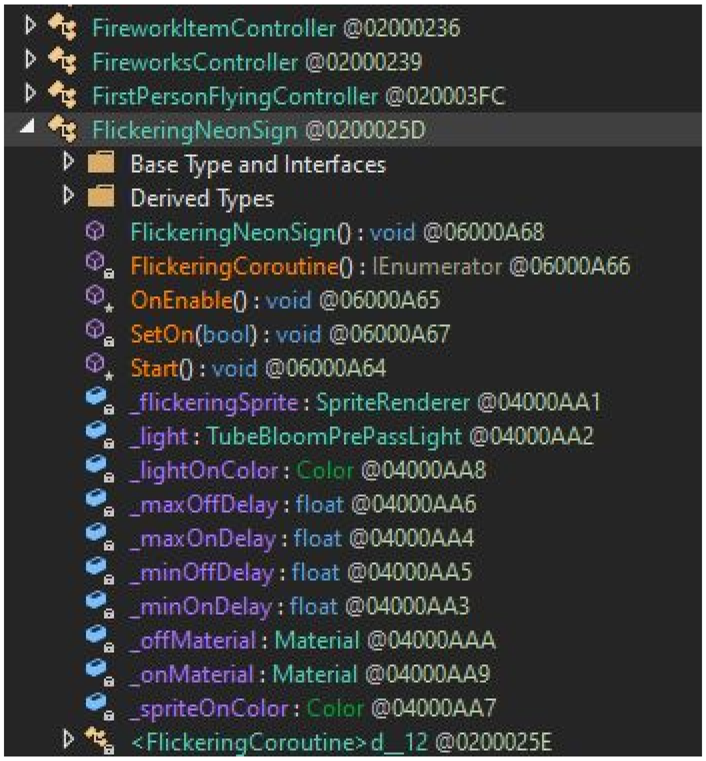
Replace the document with this:

```
#include "../include/mod_interface.hpp"

#include <unordered_set>

#include "../extern/beatsaber-hook/shared/utils/utils.h"
#include "../extern/beatsaber-hook/shared/utils/logging.hpp"
```

```
#include "../extern/beatsaber-hook/include/modloader.hpp"
#include "../extern/beatsaber-hook/shared/utils/il2cpp-utils.hpp"
#include "../extern/beatsaber-hook/shared/utils/il2cpp-functions.hpp"
#include "../extern/beatsaber-hook/shared/utils/typedefs.h"
#include "../extern/beatsaber-hook/shared/config/config-utils.hpp"

static ModInfo modInfo;

static Configuration& getConfig() {
    static Configuration config(modInfo);
    return config;
}

static const Logger& getLogger() {
    static const Logger logger(modInfo);
    return logger;
}


extern "C" void setup(ModInfo& info) {
    info.id = "tutorial";
    info.version = "0.1.0";
    modInfo = info;
    getLogger().info("Modloader name: %s", Modloader::getInfo().name.c_str());
    getConfig().Load();
    getLogger().info("Completed setup!");
}


extern "C" void load() {
    getLogger().debug("Hello from il2cpp_init!");
    getLogger().debug("Installing hooks...");
    il2cpp_functions::Init();

    getLogger().debug("Installed all hooks!");

}

enum class Space {
    World,
    Self
};

DEFINE_IL2CPP_ARG_TYPE(Space, "UnityEngine", "Space");
```

Now, this is the BASE of any mod. The *#includes* at the top basically tell the file where other code to run is. *extern "C" void load()* is a function that runs the code in it when the mod is loaded. *enum class Space* defines the world area.

Now, let's add a floating UI that tells us some random word.
In dnSpy, there is a LevelSelectionFlowCoordinator class. We are going to hook that.



```
▷ FireworkItemController @02000236
▷ FireworksController @02000239
▷ FirstPersonFlyingController @020003FC
◢ FlickeringNeonSign @0200025D
    ▷ Base Type and Interfaces
    ▷ Derived Types
      FlickeringNeonSign() : void @06000A68
      FlickeringCoroutine() : IEnumerator @06000A66
      OnEnable() : void @06000A65
      SetOn(bool) : void @06000A67
      Start() : void @06000A64
      _flickeringSprite : SpriteRenderer @04000AA1
      _light : TubeBloomPrePassLight @04000AA2
      _lightOnColor : Color @04000AA8
      _maxOffDelay : float @04000AA6
      _maxOnDelay : float @04000AA4
      _minOffDelay : float @04000AA5
      _minOnDelay : float @04000AA3
      _offMaterial : Material @04000AAA
      _onMaterial : Material @04000AA9
      _spriteOnColor : Color @04000AA7
    ▷ <FlickeringCoroutine>d__12 @0200025E
```

First, we need to make a hook - to “hook” on to that function when it is called in the game. Under the *#includes*, type *MAKE_HOOK_OFFSETLESS(){}* to make a new hook. *MAKE_HOOK_OFFSETLESS* takes three parameters:

- Function name
- Function type
- Object

In this example, we will hook *NoteController* on *SendNoteWasMissedEvent* , and we are going to need a name for this function. Let's call it *NoteCut*

As you can see in **dnSpy**, *SendNoteWasMissedEvent* is a *void*, so the second parameter is *void*.

The object will be *Il2CppObject* self*. So the third parameter will be that, too. That basically means this function - it*self*. *Il2CppObject** is the return object of *il2cpp_utils*.

Altogether, the comma-separated parameters will be *NoteCut, void, Il2CppObject* self*, So put that in the parentheses:

```
MAKE_HOOK_OFFSETLESS(NoteCut, void, Il2CppObject* self) {
}
```

Because we made a hook, we have to install it. We will install it at the start of the game - in *extern "C" void load() { }*.

The function for installing a hook is *INSTALL_HOOK_OFFSETLESS()*. The first parameter is the same, *notecut*, because that is the name of our function.The other one is returned by *il2cpp_utils::FindMethodUnsafe*, the parameters of which are *NoteController*, *SendNoteWasMissedEvent*, and the number of extra parameters. There are none, if you look at *SendNoteWasMissedEvent* in **dnSpy**, there are no parameters in it. When there are parameters, you add there amount in the third parameter here, and themselves in the *MAKE_HOOK_OFFSETLESS* , with non c++ types as just *IL2CppObject**:

```
INSTALL_HOOK_OFFSETLESS(NoteCut, il2cpp_utils::FindMethodUnsafe("",
"NoteController", "SendNoteWasMissedEvent", 0));
```

Put it in *extern "C" void load() { }* to get

```
extern "C" void load() {
   getLogger().debug("Hello from il2cpp_init!");
   getLogger().debug("Installing hooks...");
   il2cpp_functions::Init();

   INSTALL_HOOK_OFFSETLESS(NoteCut, il2cpp_utils::FindMethodUnsafe("",
"NoteController", "SendNoteWasMissedEvent", 0));

   getLogger().debug("Installed all hooks!");

}
```

To finally spawn particle systems, make an empty GameObject. Put the following in the *MAKE_HOOK_OFFSETLESS* function.

```
 Il2CppObject* empty = CRASH_UNLESS(il2cpp_utils::New("UnityEngine",
"GameObject"));
```

CRASH_UNLESS crashes the game if it can't do what's inside. It also helps convert optionals to *Il2CppObject**. *il2cpp_utils::New* is the same as *new* in unity c#. To do c# *new GameObject*, you do the above. The parameters of il2cpp_utils::New are (most times) *UnityEngine* and the type. We have a GameObject type/

Next we need to add the particle system component:

```
  Il2CppObject* component = CRASH_UNLESS(il2cpp_utils::RunMethod(empty,
"AddComponent", il2cpp_utils::GetSystemType("UnityEngine", "ParticleSystem")));
```

To break that down. We get the type ParticleSystem from UnityEngine, and add that as a component to empty. The parameters of *RunMethod* are Class/reference, Function, and parameters of that function. The reference this time is *empty*, a variable we made for the gameobject. We *AddComponent* to it, and the parameter for *AddComponent* is the system type, which *GetSystemType* retrieves for us.

Setting a transform component is almost the same, but the component is there already, so we don't add we set:

```
    Il2CppObject* transform = CRASH_UNLESS(il2cpp_utils::RunMethod(empty,
"GetComponent", il2cpp_utils::GetSystemType("UnityEngine", "Transform")));
```

All we changed is the variable name, the Add to Get, and the component itself.

Now that we have the transform reference, we need to set the position to behind the player:

```
CRASH_UNLESS(il2cpp_utils::SetPropertyValue(transform, "position", Vector3{0,0,-5}));
```

CRASH_UNLESS is a good debugging tool. It crashes your game when its function doesn't work, so you can see what part broke. *SetPropertyValue* sets values in properties of components. We need a Class/reference (*transform*, our variable, not to be confused with *Transform* from unity), the variable that we're the value of (*position*), and the value we set it to (*Vector3{0,0,-5}*). C# is different than C++, so we use *{ }* not *( )* in *Vector3*.

Finally, we need to let the game run the original functions of the hook we changed. Call the function:

```
NoteCut(self);
```

There should be a new particle system spawned each miss now.

MAKE_HOOK_OFFSETLESS should look like:

```
MAKE_HOOK_OFFSETLESS(NoteCut, void, Il2CppObject* self) {

    getLogger().debug("Level Started!");

    Il2CppObject* empty = CRASH_UNLESS(il2cpp_utils::New("UnityEngine",
"GameObject"));

    Il2CppObject* component = CRASH_UNLESS(il2cpp_utils::RunMethod(empty,
"AddComponent", il2cpp_utils::GetSystemType("UnityEngine", "ParticleSystem")));
    Il2CppObject* transform = CRASH_UNLESS(il2cpp_utils::RunMethod(empty,
"GetComponent", il2cpp_utils::GetSystemType("UnityEngine", "Transform")));

    CRASH_UNLESS(il2cpp_utils::SetPropertyValue(transform, "position",
Vector3{0,0,-5}));

    NoteCut(self);
}
```

Your entire main.cpp file should be:

```
#include "../include/mod_interface.hpp"

#include <unordered_set>

#include "../extern/beatsaber-hook/shared/utils/utils.h"
#include "../extern/beatsaber-hook/shared/utils/logging.hpp"
#include "../extern/beatsaber-hook/include/modloader.hpp"
#include "../extern/beatsaber-hook/shared/utils/il2cpp-utils.hpp"
#include "../extern/beatsaber-hook/shared/utils/il2cpp-functions.hpp"
#include "../extern/beatsaber-hook/shared/utils/typedefs.h"
#include "../extern/beatsaber-hook/shared/config/config-utils.hpp"

static ModInfo modInfo;

static Configuration& getConfig() {
   static Configuration config(modInfo);
   return config;
}

static const Logger& getLogger() {
   static const Logger logger(modInfo);
   return logger;
}

MAKE_HOOK_OFFSETLESS(NoteCut, void, Il2CppObject* self) {

   getLogger().debug("Level Started!");

   Il2CppObject* empty = CRASH_UNLESS(il2cpp_utils::New("UnityEngine",
"GameObject"));

   Il2CppObject* component =
CRASH_UNLESS(il2cpp_utils::RunMethod(empty, "AddComponent",
il2cpp_utils::GetSystemType("UnityEngine", "ParticleSystem")));
   Il2CppObject* transform = CRASH_UNLESS(il2cpp_utils::RunMethod(empty,
"GetComponent", il2cpp_utils::GetSystemType("UnityEngine", "Transform")));

   CRASH_UNLESS(il2cpp_utils::SetPropertyValue(transform, "position",
Vector3{0,0,-5}));

   NoteCut(self);
}

extern "C" void setup(ModInfo& info) {
   info.id = "tutorial";
   info.version = "0.1.0";
```

```
    modInfo = info;
    getLogger().info("Modloader name: %s", Modloader::getInfo().name.c_str());
    getConfig().Load();
    getLogger().info("Completed setup!");
}


extern "C" void load() {
    getLogger().debug("Hello from il2cpp_init!");
    getLogger().debug("Installing hooks...");
    il2cpp_functions::Init();

    INSTALL_HOOK_OFFSETLESS(NoteCut, il2cpp_utils::FindMethodUnsafe("",
"NoteController", "SendNoteWasMissedEvent", 0));

    getLogger().debug("Installed all hooks!");

}

enum class Space {
    World,
    Self
};

DEFINE_IL2CPP_ARG_TYPE(Space, "UnityEngine", "Space");
```

Your code is ready! Make sure to save it

11. Change game version

In *bmbfmod.json*, there is a line:

```
"gameVersion": "1.6.0",
```

Change it to

```
"gameVersion": "1.10.0",
```

Or your latest Beat Saber version, and save.

12. A few more things

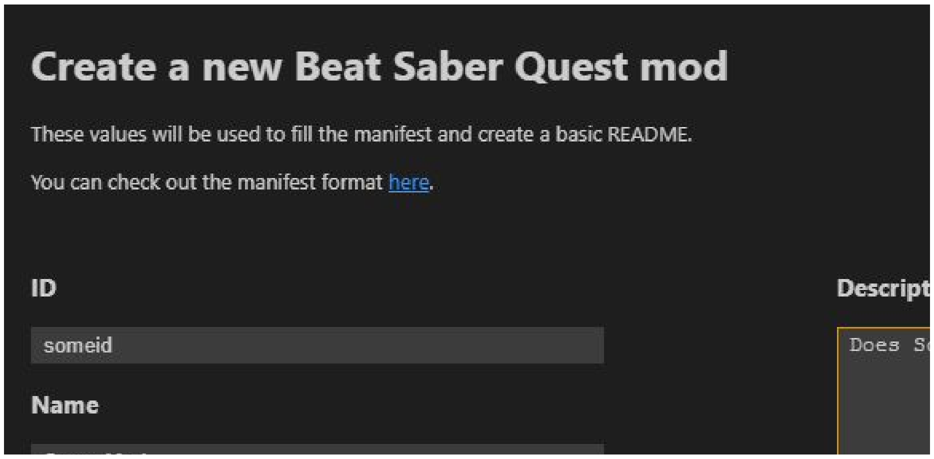
In *Android.mk* at Line ~43, change

```
-I'c:/Users/Ethan/Desktop/Unity/2018.4.15f1/Editor/Data/il2cpp/libil2cpp'
```

to

-I'c:/Users/Ethan/Desktop/Unity/2018.4.15f1/Editor/Data/il2cpp/libil2cpp' -frtti

Make sure in *main.cpp*, in extern "C" void setup, your mod id and version are correct. Remember mod id?



Create a new Beat Saber Quest mod

These values will be used to fill the manifest and create a basic README.

You can check out the manifest format here.

ID

someid

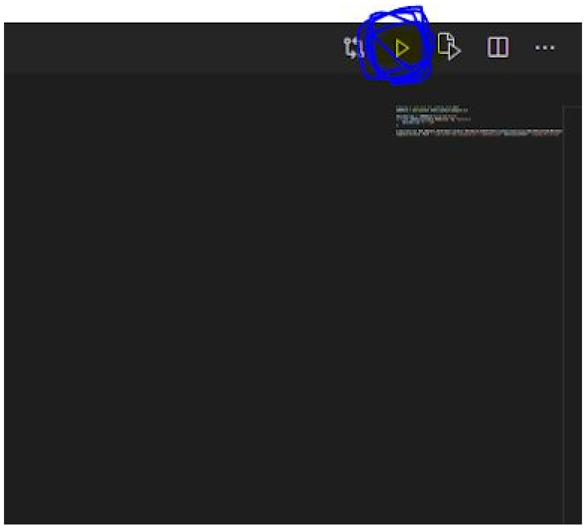
Name

Descript

Does S

13. Build

Open buildbmbf.ps1
Click play in the top right



Open File Explorer and open your mod folder. If there is a zip file, you are good to go! Upload that to BMBF.

14. Test

Play a song! The game will punish you if you mess up!