

**CSCI 5306**    
Applied Programming and Comprehension

**Assignment 3**

Build and Run

**Crew 7**

Akshit Jariwala (B00866255)

Ria Shah (B00882283)

Rishika Bajaj (B00902713)

Sanket Ushangbhai Mehta (B00881783)   
Sumit Singh           (B00882103)   
Tejaswini Rallapalli (B00888478)

**Introduction:**

**How Build and Run helps:**

**The procedure of how we approached the assignment:**

First, we studied SDK (Software Development Kit) and its details, how it works, and why it is important. Software Development Kit is a collection of software development tools in one installable package. We also learned that SDK is specific to platform hardware and operating systems. We started learning the difference between the SDKs (Software Development Kit) in various platforms, how they differ from each other, and how to use them in their respective environments. We have platform-dependent Software Development Kit because the two operating systems have different executable formats and different core libraries, hence the executables in the Windows version will not run-on Linux and vice versa. Every platform we picked had a different method of building the SDKs for its respective environment.

We decided to build Software Development Kit for multiple environments, that way we could also get exposure and hands-on with all popular platforms available. We split ourselves into teams so that we can divide the platform and build the Software Development Kit for those, rather than everyone working on the same platform. This way we could cover multiple platforms and build their Software Development Kit.

Rather than directly jumping into the process of building the Software Development Kit, we started to search for documents available online, that can guide us on how to build and run for that respective platform. We came across STK (Super Tux Kart) documentation that has all the instructions and steps to be followed to build Software Development Kit. There were build instruction and README.md files for different platforms that helped in getting some insight into the build process. We tried following those for the respective platform and started building the Software Development Kit, we came across a few issues while building them, we had to google search for those showstoppers and figure out how to resolve them. We were able to resolve most of our issues during the build process.

**Steps taken to build software development kit:**

**Android build**

**Steps were taken to build Android SDK:**

**Step 1 (Downloading source code and pre-requisites):**

To get started with building an executable file, we searched for STK code online and found a GitHub repository, and cloned the Master branch repository to our local machine. By cloning the Master branch, we ensured that we are fetching the most stable version and up-to-date code.

**Step 2 (Researching for documentation and instructions to build):**

Every good programmer does the documentation for their program, which helps newcomers to understand and start with the project. So, we started searching for any kind of documentation that might help us in the process of building android SDK. We started to look for a few keywords like build, install, android and found a few matches too but none of them were helpful for our work. For instance, we came across an ‘INSTALL.md’ file containing build information about three platforms: Linux, Windows, and Mac OS. It was not useful for android, so now started searching the source code and found a folder named ‘andriod’ and this folder had a ‘README.ANDROID’ file which contained build instructions on how to build an executable Android version of Super Tux Kart. Fairly, now we have a clear idea about where to start.

**Step 3 (Installing tools needed):**

To build an android application we need to have an android studio installed and we need java for android studio. So, we installed java and added the environment variables needed to run the java program and after successfully doing that, we installed android studio.

**Step 4 (Downloaded Dependencies):**

We had a list of required libraries to be downloaded to build Super Tux Kart. These libraries are not available in the STK source code because these are platform-dependent libraries, in this case, android version-specific. Downloaded dependencies-andriod-src.tar.xz from URL <https://github.com/supertuxkart/dependencies/releases>

**Step 5 (loading STK code into IDE):**

After ensuring we have all the required tools and dependencies available to load and get started in the process of building the android executable, we load the project source code into android studio. This gives us a directory structure of the code and its modules.

**Step 6 (extracting the libraries):**

Once the code is in android studio, the only thing left is to add the dependencies libraries to the source code. To do this, we extracted the downloaded library folder at the ‘stk-code-1.3 -> lib’ path.

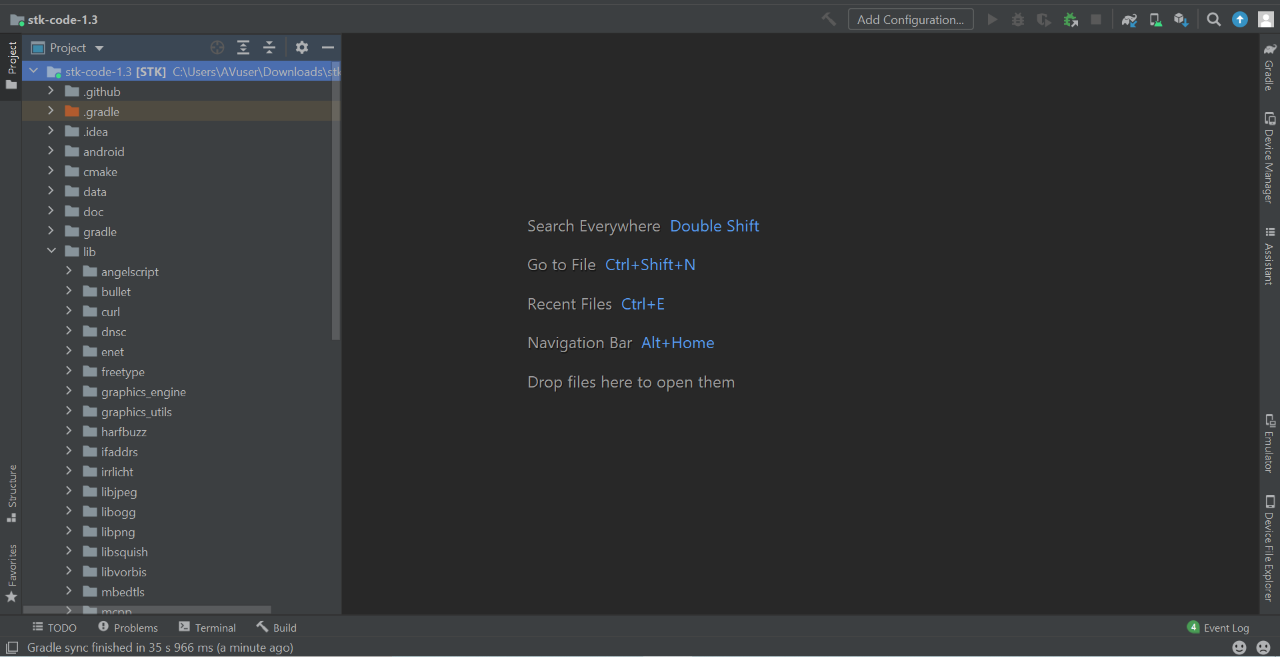


Figure 1 Source code structure after inserting dependencies

**Step 7 (Meeting Android-specific requirements):**

To build the android SDK of Super Tux Kart, we need Android DSK for the android 26 platform and Android NDK version r23. To understand more about these, google searched about how to download and add android SK and NDK to Super Tux Kart source code. Found an article on ‘FreeDevGames Forum’ about how to Android dependencies for Super Tux Kart (<https://forum.freegamedev.net/viewtopic.php?t=13890&p=92726>). The article helped me in understanding how to link Android SDK and Android NDK with STK source code.

**Step 8 (Reason behind picking Android studio IDE):**

To build the Super Tux Kart source code, we had two main options for IDEs: IntelliJ IDEA and Android Studio. Though IntelliJ can build android code, we decided to use Android Studio as it was recommended in ‘FreeDevGames Forum’ and it is also widely used to develop android applications.

**Step 9 (Adding android SDK to source code):**

As per the Android documentation to build Super Tux Kart, adding Android SDK platform 26 or higher was recommended. We added Android SDK Platform 32 dependency in the STK source code as it was the latest stable version.

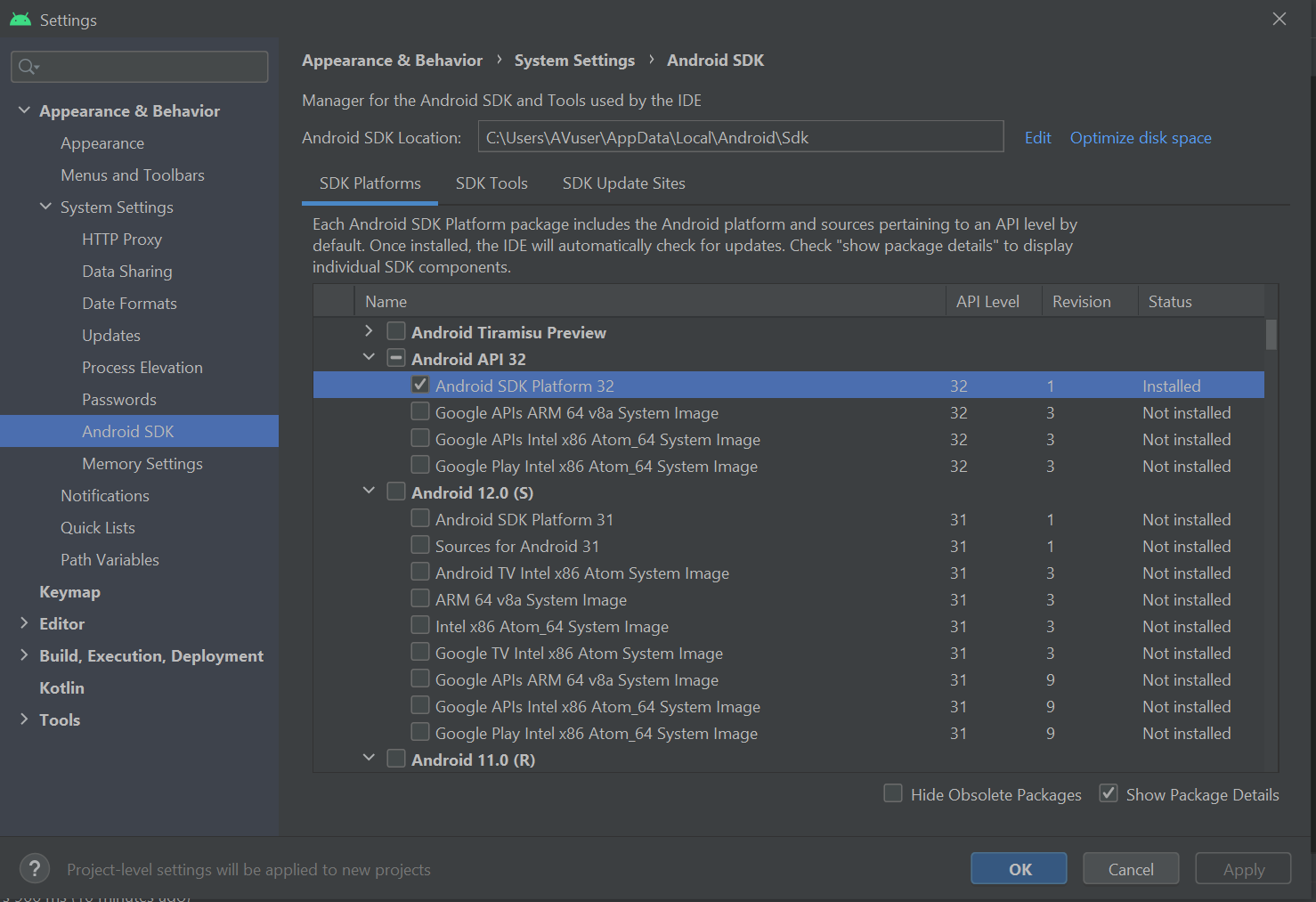


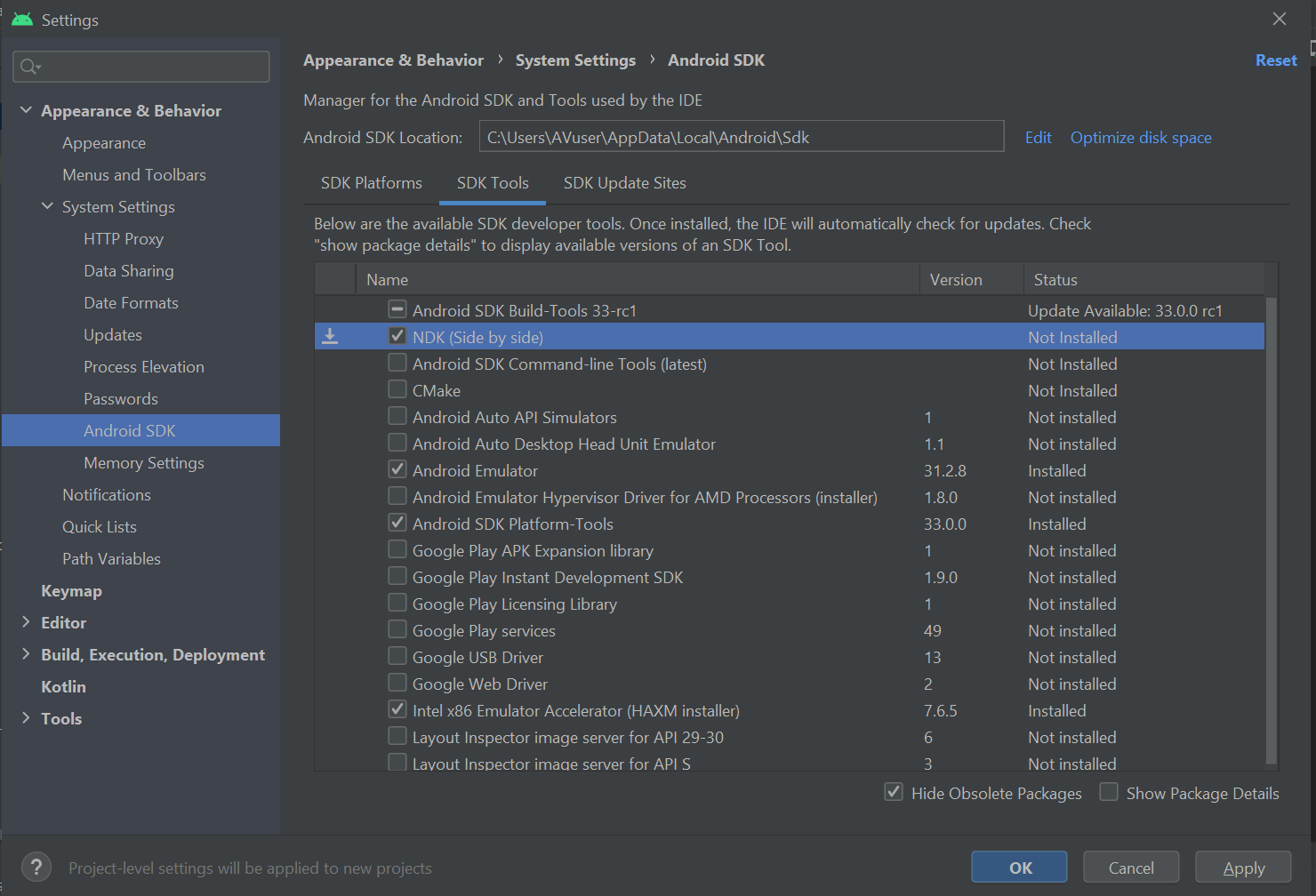
Figure 2 Adding Android SDK 32 to SuperTuxKart

**Step 10 (adding Android NDK to source code):**

Another requirement of the build was to link, Android NDK version r23.

The Android NDK is a companion tool to the Android SDK that allows you to write native code for performance-critical parts of your apps. When programming in C or C++, it includes headers and libraries that let you create activities, handle user input, use hardware sensors, access application resources, and more. As SuperTuxKart is written in C++, we need Android NDK to support native code.

We followed this documentation to add the required NDK version to our Super Tux Kart project. (<https://developer.android.com/studio/projects/install-ndk>). We followed these steps: Tools-> SDK manager -> Appearance & Behaviour -> System settings -> Android SDK -> SDK Tools -> Select NDK (side by side)



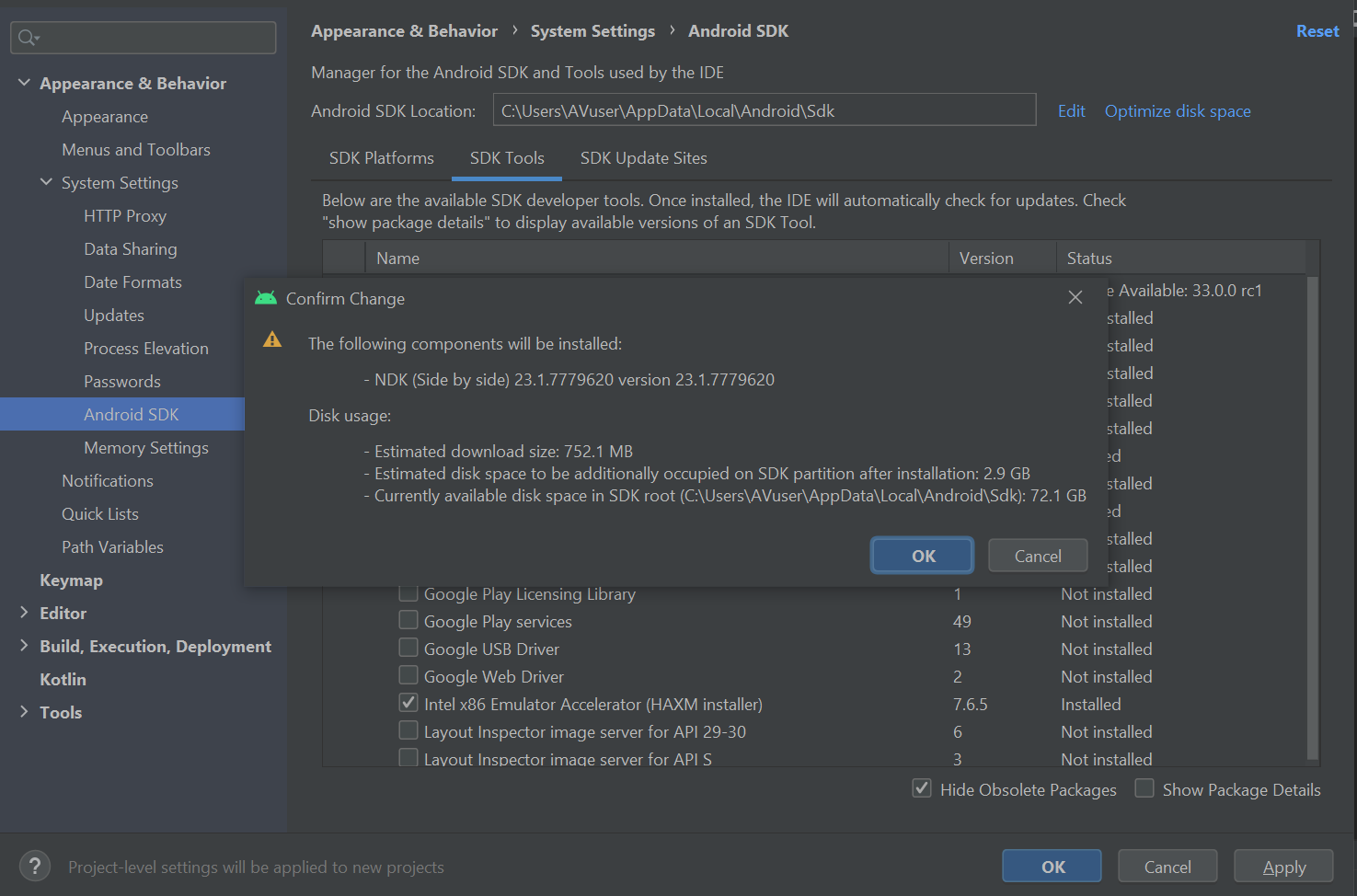


Figure 3 Adding NDK to SuperTuxKart

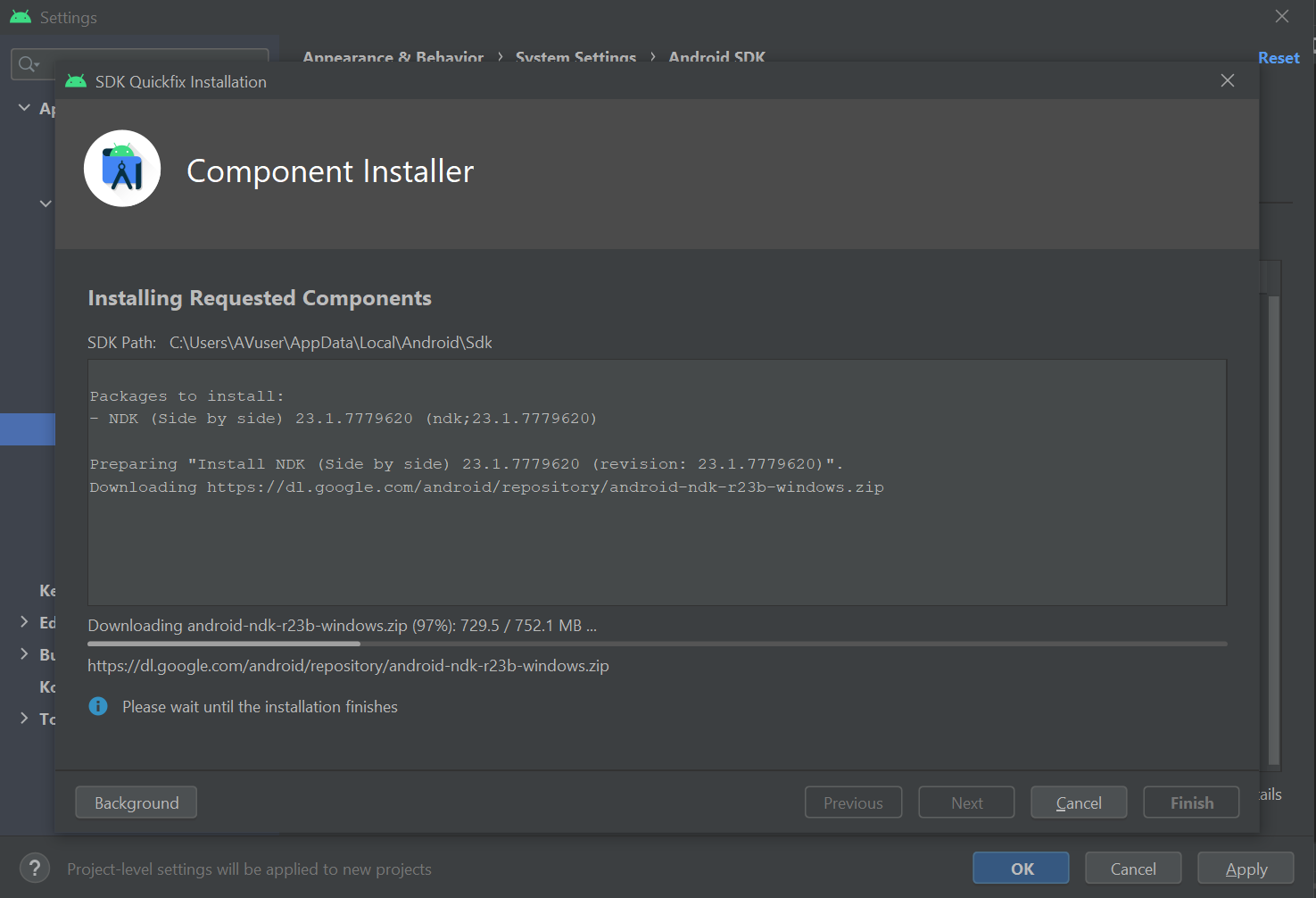


Figure 4 Installing NDK

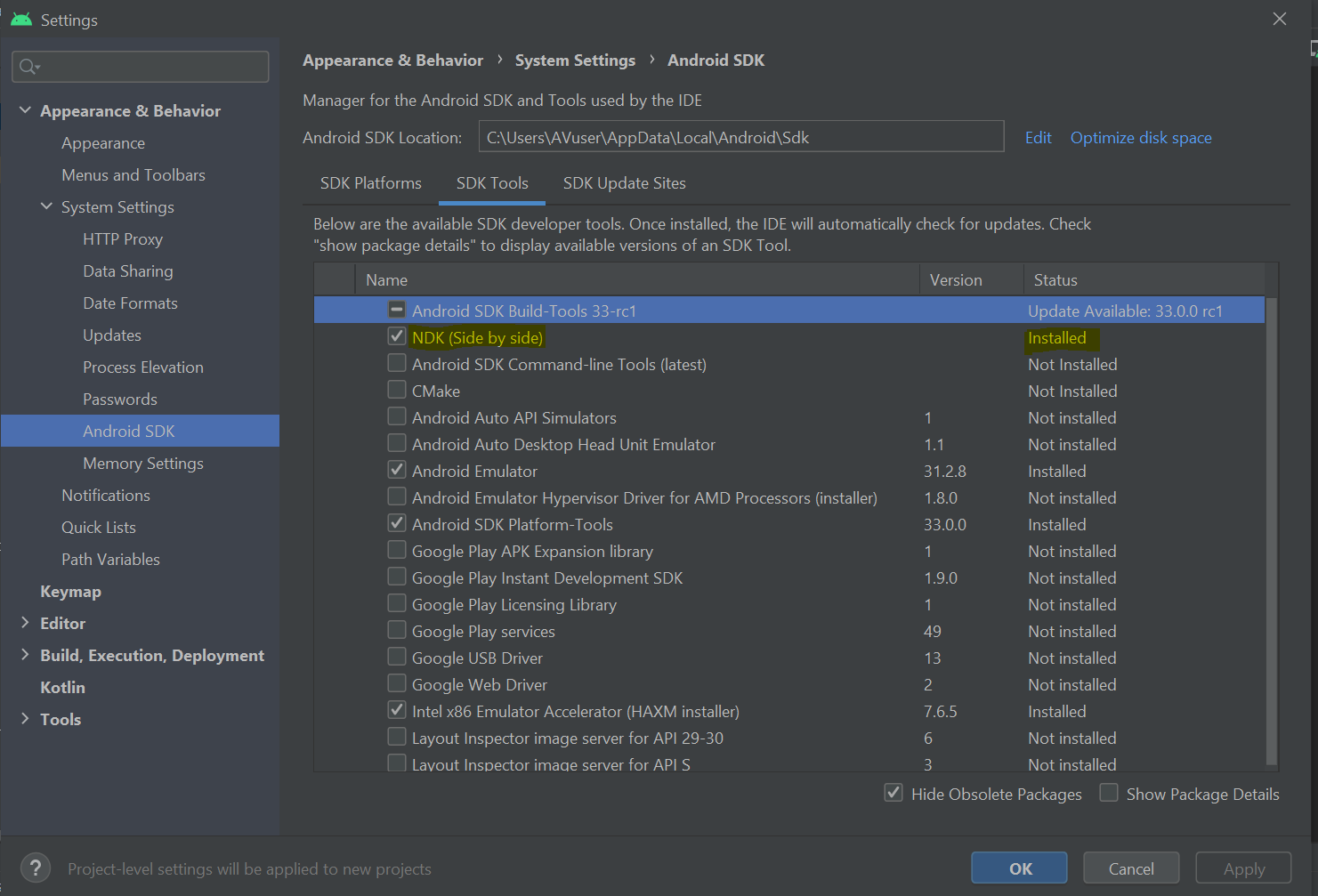


Figure 5 NDK Added

**Step 11 (Setting environment variables):**

In this step, we are about to set recommended environment variables to build the Super Tux Kart. Following is the list of variables Android SDK, Android NDK, minimum & target Android SDK version, and Android NDK version to be set as environment variables. To do this we ran the scripts below:

SET SDK\_PATH = C:\Users\AVuser\AppData\Local\Android\Sdk

SET NDK\_PATH = C:\Users\AVuser\AppData\Local\Android\Sdk\ndk

SET STK\_MIN\_ANDROID\_SDK = 16

SET STK\_TARGET\_ANDROID\_SDK = 32

SET STK\_NDK\_VERSION = 23.1.7779620

**Step 12 (run shell scripts):**

The next step was to run shell scripts, but my laptop did not have bash installed thus I had to download it. After successfully installing and downloading and installing bash, I ran all the scripts but one. The last script that was supposed to build the executable android was bash made. sh. As soon as we ran this script we came across errors, error said the JAVA\_HOME environment variable is not declared whereas it was and firstly without the JAVA\_HOME environment variable set we can run Java applications.

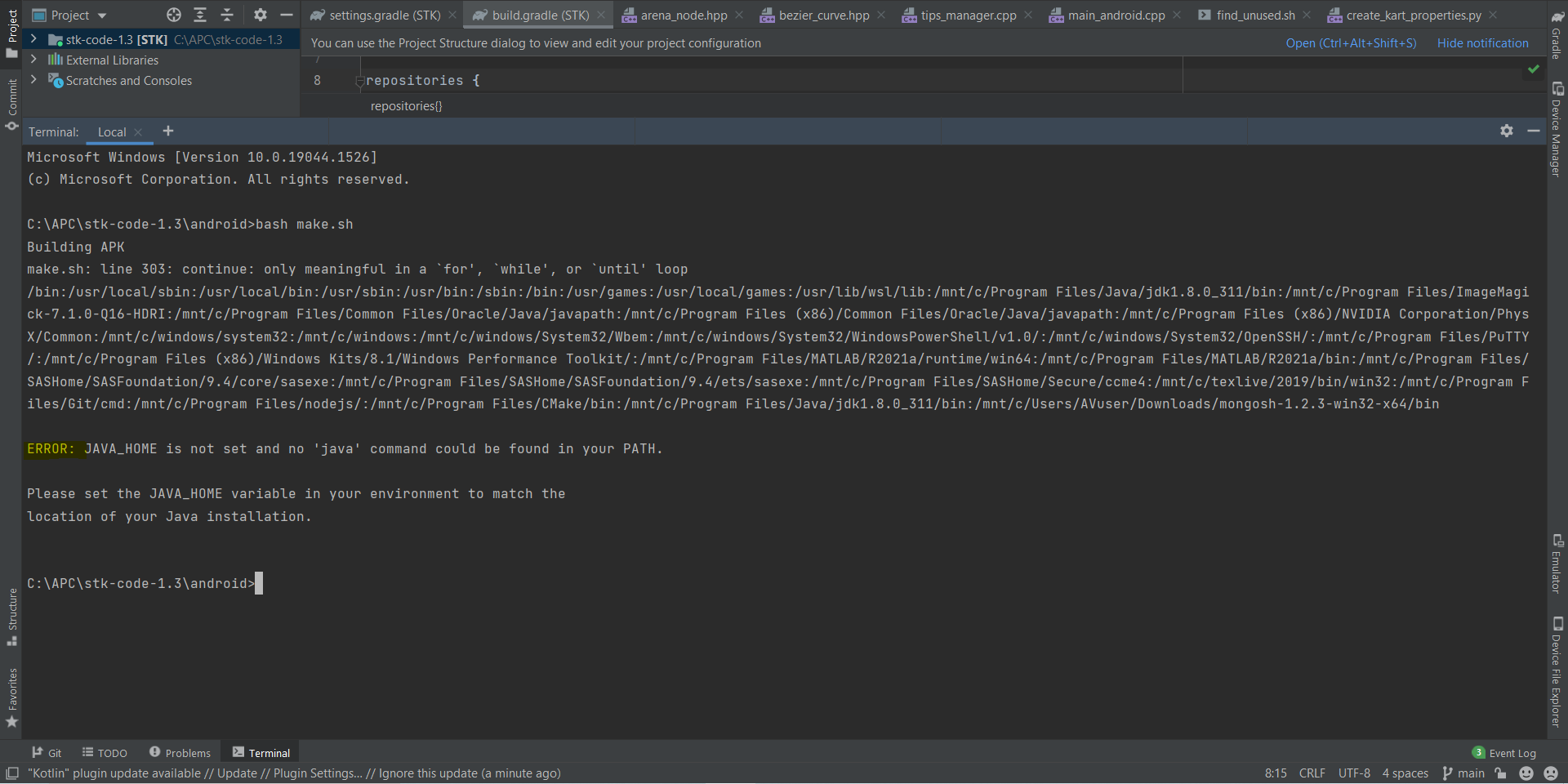
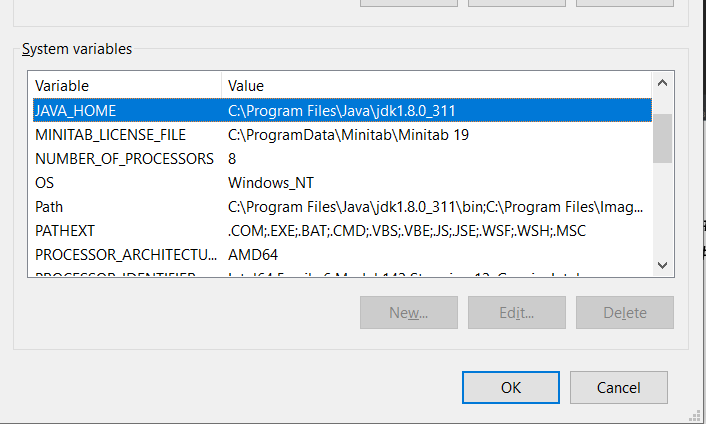


Figure 6 JAVA\_HOME not set error

Figure 7 JAVA\_HOME environment variable into our system

We made few changes to make.sh file in order to detect JAVA\_HOME environment variable from the system but the code was unable to detect the same. We commented out few lines of code that was assigning JAVA\_HOME and let the script access the system environment variable but still it was not able to detect java dependencies in the system.

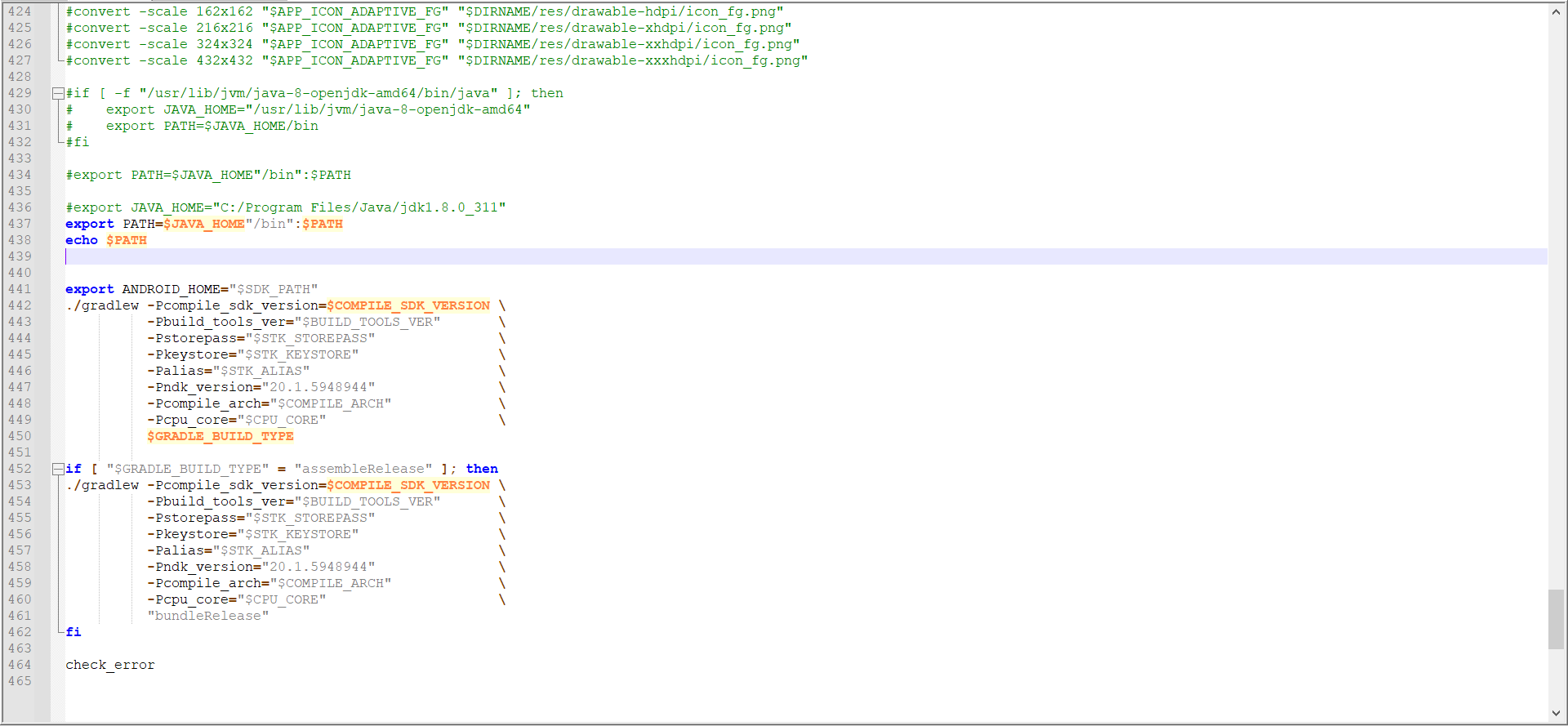


Figure 8 Changes made to the make.sh script

**Proof of Success:**

**Challenges faced and limitation while compiling Android:**

**Challenge 1:**

As per the document, to build executable we need to have android-SDK and android-NDK dependencies attached to the source code. And Super Tux Kart source folder was supposed to contain directory for Android-NDK and Android-SDK. But we did not have android-NDK and android-SDK.

**Solution**: In order to link them we Created new folder for both and added android SDK and android NDK in their respective folders.

**Challenge 2:**

Android folder did not have assets folder. And to get the data related to kart, user, track etc. we need assets data in assets folder.

**Solution:** Created a directory named assets in android folder and copied data from Super Tux Kart source code to this assets folder. This ensures that whenever the program starts, we have a place from which we can pull required data about the assets.

**Challenge 3:**

At the very end we had to run a few scripts to run the program successfully and to run scripts we needed to have cmake installed on our personal computers.

**Solution:** downloaded and installed cmake to run cmake commands in build script.

**Challenge 4**:

We ran into an Issue with last command ‘bash make.sh,’ it threw an error saying JAVA\_HOME variable is undefined (figure 6). Which in fact was properly defined in environment variables (figure 7).

**Tools used:**

**Mac OS:**

**Linux:**

**Windows:**

**Android:**

**Comparison between different build:**

**Learning outcome:**

**References:**

<https://github.com/supertuxkart/dependencies/releases>

<https://forum.freegamedev.net/viewtopic.php?t=13890&p=92726>

<https://developer.android.com/studio/projects/install-ndk>

<https://docs.microsoft.com/en-us/windows/wsl/about>