Age of Civilizations 2 Modding Tutorial
3/17/2023
By Pumped

Introduction

UPDATE - A better guide that significantly shortens time of both decompilation and recompilation can be found at Bloody Europe's VK! This is highly recommended but you can still read this document to get some in-depth knowledge.
https://vk.com/@bloodyeurope2-gpk-1-zapusk-ishodnogo-koda

Hello,

The following is a coding tutorial for Age of Civilizations 2. To begin, you must have prior knowledge with the programming language Java, and have a valid Java IDE (preferably IntelliJ). Also, it is recommended that you have background knowledge about different Java file types (.jar, .class, .java) explained here.

Decompilation

NOTE: You can skip this step by downloading the decompiled .java code here. I simply wanted to outline the process to those interested or those who want to edit an already modded .jar file.

The first step of the process is decompilation. The game's code is packaged in a Java-compiled .jar, containing .class files that hold the game's code. However, in this state, they are uneditable. To make them editable, you must use one of the many .jar file decompilers online, and decompile the games .jar file (usually called AoC2.jar and located inside of the Age of Civilizations game folder) into a series of editable .java files. My personal recommendation is to use BCV due to its simple nature.

Decompilation using these applications are usually simple and outlined in respective websites. For BCV, you must open the AoC2.jar file, click 'File' > 'Decompile & Save All Opened Classes' > Choose a directory folder, and all the game's code should show up in the form of java files. This brings up to the next step,

Setting up the Java Environment

Editing the decompiled .java files is somewhat similar to editing a normal Java program. The game files specifically, which I assume most of you reading want to edit, are located in the decompiled folder Age > Of > Civilizations2 > Jakowski > Lukasz. However, a few notable things must be done to the environment beforehand. In your IDE, set your Java SDK (ie, Java version) to Java 6, 7, or 8 JDK, installed at Oracle's Java Archive. This is because the game's files

are coded on Java 6, and compiling them alongside files of another version lead to instability and issues.

Furthermore, you must set up a library that refers back to the original, compiled AoC2.jar. This is done to ensure that you can edit singular files without having to bring all their dependencies into the environment (many files that are required as dependencies by other files, such as the game's important CFG.java, also, are decompiled oddly and will cause errors, so this helps prevent that issue). Also, once you do this, many IDEs are able to recognize and show you the contents of functions from different files, which will likely aid you later on.

Finally, set the project artifact to compile into a .jar file. You will see why later.

Actually Editing

Now, choose whichever files you want to edit. There is no clear documentation for what each file does, so it might take a bit of looking through to find the code detailing a specific functioning, but most are recognizable by names (e.g. Commands.java => code for the in-game console commands). I recommend utilizing the Windows file search, which, as I have recently learned, can actually search through java file contents and recognize words, along with Visual Studio Code to quickly open and scan through java files.

Once you have identified all the files that you wish to edit, you should copy them into your IDE. It is highly recommended that you only copy files that you must edit; if you are simply looking at them for reference, use Visual Studio Code or another .java file viewer outside of your IDE, as many files have random errors in them that prevent you from compiling.

I apologize if the above was confusing, so I will try to give an example to put this into perspective. Let's say that I want to edit the in-game console commands, and add, for example, a war weariness command in the console that allows one to decrease a civilization's war weariness. Using the above method, I would look through the game files and eventually come across Commands.java. I would also look for the function that sets a civilization's war weariness, which I would discover in Civilizations.java. Now, I would copy Commands.java into my IDE, but not Civilizations.java, because I only want to edit the Commands file and avoid compilation errors. However, I would keep Civilizations.java open in the background in some other program (like VSCode) to reference.

Hopefully that made some sense. It is also important to replicate the package of the code that was found. For example, Commands.class is found in (and uses the package) Age > Of > Civilizations2 > Jakowski > Lukasz. So, in your IDE, you must create the folders Age > Of > Civilizations2 > Jakowski > Lukasz, and drop Commands.class in that.

This is the step that takes the most amount of time. See the section at the end for more help, if needed.

Recompilation

Once you have made the edits to the code, you must recompile it to implement into the game. If you have followed the 'Setting Environment' step above, it should work seamlessly and compile into a .jar file.

The compilation into a .jar file was done to replicate the game and allow implementation. The steps now are relatively simple. Duplicate the AoC2.jar file inside of your Age of Civilizations directory, open the file with a .jar viewer (WinRAR is likely the most common/easiest, simply right click on the .jar file > Open with WinRAR), and also open the .jar file that you just compiled. Now, extract the contents of the modified .jar file into the AoC2.jar file, allowing WinRAR or whatever editor to replace existing files.

From here, you should be able to run the game using the modified AoC2.jar file. If it crashes, you may debug by opening the .jar file using a windows terminal (simply CD to the directory, and run 'java -jar AoC2.jar', launching the game, and when it malfunctions the corresponding java error should be displayed in the terminal.

Conclusion

Hopefully this will help attract and bring new things to the AoC2 community. The modding community, especially the English community, has been declining in the past few years, even though the game has so much modding potential. Let us hope that this can be solved soon.

I understand if this tutorial has left you extremely confused. Feel free to message me at pm#3682 if you have trouble understanding this or have practical issues with Java.

- Pumped