Welcome to the Dynamic Character Pose tool! This tool will look through the directories for each character and generate lines of Ren'Py code for each combination of poses and expressions it finds (while excluding incorrect combinations as we specify).

If you've downloaded this tool, you're probably excited to start using it to pose your characters without limits in your *Doki Doki Literature Club* mod, or perhaps even your own original Ren'Py project. This guide will go through the steps to get you started and working with the tool as quickly as possible, with more in depth guides and instructions found in other files. Let's get started!

Adding the Tool to Your Project

The tool and its documents came in a folder called "character_images". The easiest way to add the tool to your project is to take this folder and add it to the "mod_assets" folder you likely have.

You can skip this paragraph safely.
Strictly speaking, the only files which are important to have in your project are Create_Definitions.pl, settings.txt, and the folders for each character. These files can go anywhere as long as they're all in the same folder, there are no other folders in the directory unless they start with "i_" for "ignore", and they are within the "game" folder on some level. The tool will be looking for the "game" folder above it and will be looking through any folder not marked "i_", which is why this structure is needed.

Installing Perl (One Time Process)

Create_Definitions.pl is the code file where most of the magic happens. This is a Perl script which can be opened and read with a text editor (but don't make changes unless you know what you're doing—even then, make a backup!). To run the script, we'll need to install Perl on your computer and then run it from the command line.

Q: Command line? Am I going to accidentally kill my computer?
We won't be doing anything dangerous to your computer at all. When everything is ready, we'll be using two commands on the command line. First is "`cd`" which means "change directory" to tell the command line where the Perl script is. Then we'll be using "`perl`" and the file name to run the script itself. The script will be reading through some files in your game directory and will write a few new ones. When you run the script after the first time, it will delete anything in the files it wrote and write new data to them, but it will not alter or delete any other files. Because it needs access to some files, you may need to run the command line with administrator privileges so it can do it's thing. Hopefully this explanation tells you why you can do so without fear.

Q: Can the script be run without the command line?
I think so? I've never actually needed to do this, so I haven't looked into it. If anyone figures out a way to do so, please let me know so I can include it in this guide!

First we need to check if you have Perl installed already by opening up a command line. On Windows this is called Command Prompt, on Mac it's called Terminal, and if you're using

Linux then there's a good chance you or someone you know is already used to the command line as it is! On a Mac, you can find Terminal under Applications/Utilities/Terminal, or you can open Spotlight (Cmd + Space) and type "terminal" to open it. If you're on Windows, these instructions should help get you where you need to go.

Once you have a command line open, type "`perl -v`" (without quotes) and see what comes out. You'll either get an error message, meaning you need to install Perl, or you'll see version info on your installation. The tool was made with Perl v5.12, but it should work fine with any newer version (and possibly some older versions, though I recommend having at least v5.x). If you see a suitable version, you can skip to the next section!

If you need to install Perl, the official download page at https://www.perl.org/get.html is the best place to look. Whichever platform you're using, the download and install instructions are as straightforward as you would hope. On Windows, you'll be given the choice between Active Perl and Strawberry Perl. I think this is a matter of preference, but I was able to download Active Perl on an old Windows computer and get the tool to run.

Once you've finished your installation, close and reopen your command line and try "`perl -v`" again. If you see version info, you're all set. If not, then... that means this guide is incomplete and we need to do some collaborating to get everyone where they need to be.

Changing to the Correct Directory

Once you know Perl is installed on your computer, we'll need to tell our command line which folder we want to operate from. We do this with the "`cd`" command, either typing folder by folder until we get there or by typing the full path to the folder. Depending on how you prefer to do this, you can use "`cd ..`" to move one folder above the current one.

Windows

Navigating to the correct folder on Windows can be a bit tricky. When you type "`cd `" in your command line (notice the space), you'll probably want to type your file path in quotes so you don't need to worry about spaces. The command line usually takes spaces to mean that part of the command is done, and folders tend to have spaces in their names—using quotes helps it figure out what's going on here.

To see the full path to the character_images folder in your game project, you should be able to right click the address bar and choose "Edit Address" to see it and select it. You *can* copy and paste this into Command Prompt, but Ctrl + V won't work for this purpose—you'll need to go to the Edit menu and choose Paste. If "Edit Address" doesn't work, you can also open Folder Options, go to the View tab, and choose "Display the full path in title bar" under Advanced Settings. You'll need to type the path into Command Prompt by hand, but you can change the setting back afterward. Be sure to add your closing quote when it's entered and press enter, and your command prompt should be looking at the correct directory.

Mac or Linux

If you're on Mac or Linux, this part is much easier. Type "`cd `" (with the space), then drag your character_images folder into the command line and drop it there. It will automatically fill the

line with the correct path to your directory. You can just hit enter, and you're there. To double check you're in the right place, you can type "`ls`" which lists all files in this directory (a command Windows doesn't have).

Once your command line is in the correct location, you've done the hardest part. On my computer, I actually just leave my command line open here so I don't need to do this part again—consider this choice if no one else shares your computer and if you use sleep mode instead of shutting it down!

Optional Step (Mac or Linux)
An optional step you can take on Mac or Linux is to make the script itself executable. To do this, type "`chmod +x Create_Definitions.pl`". After doing this once, you will be able to run the tool by typing "`./Create_Definitions.pl`" instead of "`perl Create_Definitions.pl`", including adding flags afterward. This is entirely a matter of preference!

Running The Tool

Now that the command line is looking where we need it to, you can try your first run of the tool by typing "`perl Create_Definitions.pl`". If this just writes an error message, then it means something went wrong. If you're certain you're in the correct directory and the error is related to the script, double check that your Perl version is 5.x or higher by typing "`perl -v`". If that checks out, then we have a bigger issue to resolve that will need figuring out.

But if there are no errors, you should see some messages from the tool explaining what part of the process it's in the middle of. On my solid state drive with this default file structure, it takes about 15-30 seconds to finish depending on how many optional files are in use—it may take longer if you have a non-solid state drive.

The tool ran successfully if you see a message similar to this with different numbers:
"`File generation completed! Wrote 526 lines in 15 seconds.`"

When the tool finishes, you should see four new text files in the character_images folder: chardefinitions_monika.txt, chardefinitions_natsuki.txt, chardefinitions_sayori.txt, and chardefinitions_yuri.txt. If you open one and it looks like code, then give yourself a pat on the back—you're running the tool!

By default, the tool writes your code to these text files. This is so that it does not accidentally overwrite files in your project without your say so. To write the files directly into your project, run the tool again with the flag "`-d`" for direct. In other words, you'll type "`perl Create_Definitions.pl -d`". (You can press the up arrow key to bring back the last line you typed, then add to it and press enter again.) This time the text will be written to matching .rpy files in your game directory. Your project is ready to use them!

If you tried running the example scene before running the tool, you probably noticed the scene had dialogue, but nobody was showing up. Now all the poses they use are in those .rpy files and Ren'Py will be ready to go next time you launch your project. Running the scene now

will show everyone just how they're supposed to be!

The tool computes all possible tags every time, but only adds lines for tags which are currently being used in your project. When you add tags for custom poses to your own code (explained in another file), those tags will be added to the chardefinitions files when you run the tool again. You can type "`-s`" for a silent run which doesn't show progress messages but speeds things up slightly, and you can specify only to update select characters with the "`just`" command, followed by the names you wish to update ("`just monika`", "`just sayori natsuki`", etc).

With that, your basic intro is complete! Let's get posing! From here you can open the Pose Cheat Sheet and start constructing your own tags, or you can look at Hands On Practice and the example scene provided to see some working code lines and get a walkthrough of how to change them.