Turbo Git

The trouble is, you think you have time

So, save your time using Turbo Git!

Documentation

Date: 16.07.2021

Version: 1.01

Table of Contents

Overview

Turbo git is a tool that extremely simplifies the work for the user, and thus automatically saves and saves the data in a similar way to a kind of backup in the online platform.

You do not need to program and / or learn anything, but thanks to the beautiful user interface it allows you to access the program with just a few clicks, easily, quickly, and safely.

In the background this tool works with the git hub and git bash, but you do not have to do anything or set it up, just download and install this tool and it will do everything for you and your project.

Features

Must have:

- Create a Unity Editor for turbo git, which works perfectly and dynamic with git bash and git hub
- The user must be able to work with just a few clicks without having to program anything
- Status, Add, Commit, Push and Pull must be available
- Everything should be ready for *one-person (private)* user to work with
- A Similar behavior tree must be available for better overview
- UI should be verry nice *optically* and user friendly
- Tool Tips and Icons should be available
- Dynamically copy a .gitignore example of Unity in the actual project path

Optional:

- Rework the tool for team users (different branches, fetches etc.)
- Better and professional behavior tree with the important dates (comments, dates, colors etc.)
- Create git hub account directly in the tool (is it even possible?) → No, it's not!
- Installing git bash *automatically* using the tool (is it even possible?) → No, it's not!
- Auto git depending on user wish using time (every 30 minutes for example)

Demonstration

I am extremely positive that my tool will help many other users and that the work will be extremely simplified. I think there are enough users who would not mind saving and backing up their work forever with just a few clicks.

Especially those who are on their own or are just new starting out with unity and programming. My aim is that my tool fulfills the wishes of the user and speaks their expectations and, above all, can help / support them in their work.

Limits

Above all, the prerequisites are knowledge of C # and unity, one must still be able to be familiar with the unity editor and know how to program something as a unity editor. For the connection to the git hub and git bash, you must be familiar with the program and know what

different commands are, how it still works and what it all needs to work with. So that there are almost no problems that slow me down, thanks to my experience, I will start easily and simply and, if I have time left, continue with the program, and improve it. At the beginning I will certainly steal that my program works properly, every feature has what I wanted to incorporate and only then work on the beautiful user interface, since a beautiful design also takes a lot of time.

Precondition & Setup

- An Account on GitHub Website must be registered.
- Git Bash program must be installed.
- For better understanding how these tools works, quickly looking at the Documentation won't harm!

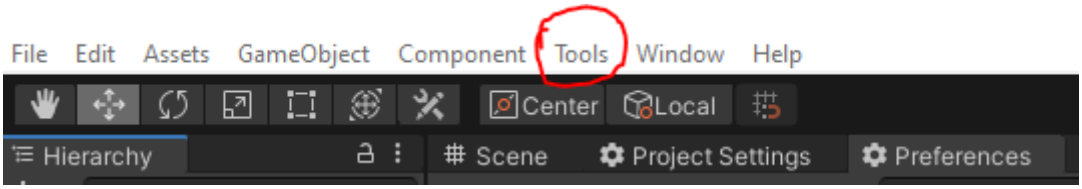
Planning: Task list/time estimations (Exel)

No.	Task	Est. Time	Eff. Time
1	Preparation	2h	3h
2	Define core features	2h	1/5h
3	Define optional feature	1h	1h
4	Create planning (what, how, when etc.)	1h	1h
5	Documentation	8h	7h
6	Get to know Git and GitHub	2h	2h
7	Get to know Unity Editor better and learn Editor Framework	2h	3h
8	Create Login window	2h	2h
9	Work on UI and user friendly (Tool tips and icons etc.)	4h	5h
10	Test the current status => Login window	1h	0/5h
11	Fix the current Bugs => Login	1h	4h
12	Reworking my scripts (used comments and better struct)	1h	2/5h
13	Create Account window	0/5h	0/5h
14	Create different Tabs for Account window	0/5h	1h
15	Add user information's	1h	1/5h
16	Add user dynamic change interface (Font, Background &Text color)	0/5h	0/5h
17	Create Status tab with Git information's	1h	1h
18	Create Git Tab	1h	1h
19	Create Status Tab	1h	1h
20	Reworking the current Tool state	1h	2h
21	Add tool tips and icons in Account window	1h	3h
22	Test the current state	0/5h	1h
23	Fix the new current Bugs	2h	5h
24	Rework all scripts (Comments added and better struct)	1h	2h
25	Final test	1h	0/5h
26	Fix new Bugs	1h	1h
27	Add dynamic working directory	0/5h	0/5h
28	Create a dynamic .gitignore file to copy into the current project	0/5h	1h
29	Last final test	1h	1h
30	Create a Manual PDF before using Turbo Git	0/5h	1/5h
31	Test everything with different PCs / Accounts of school friends	1h	1h
32	**Total Calculation**	**43/5h**	**58/h**

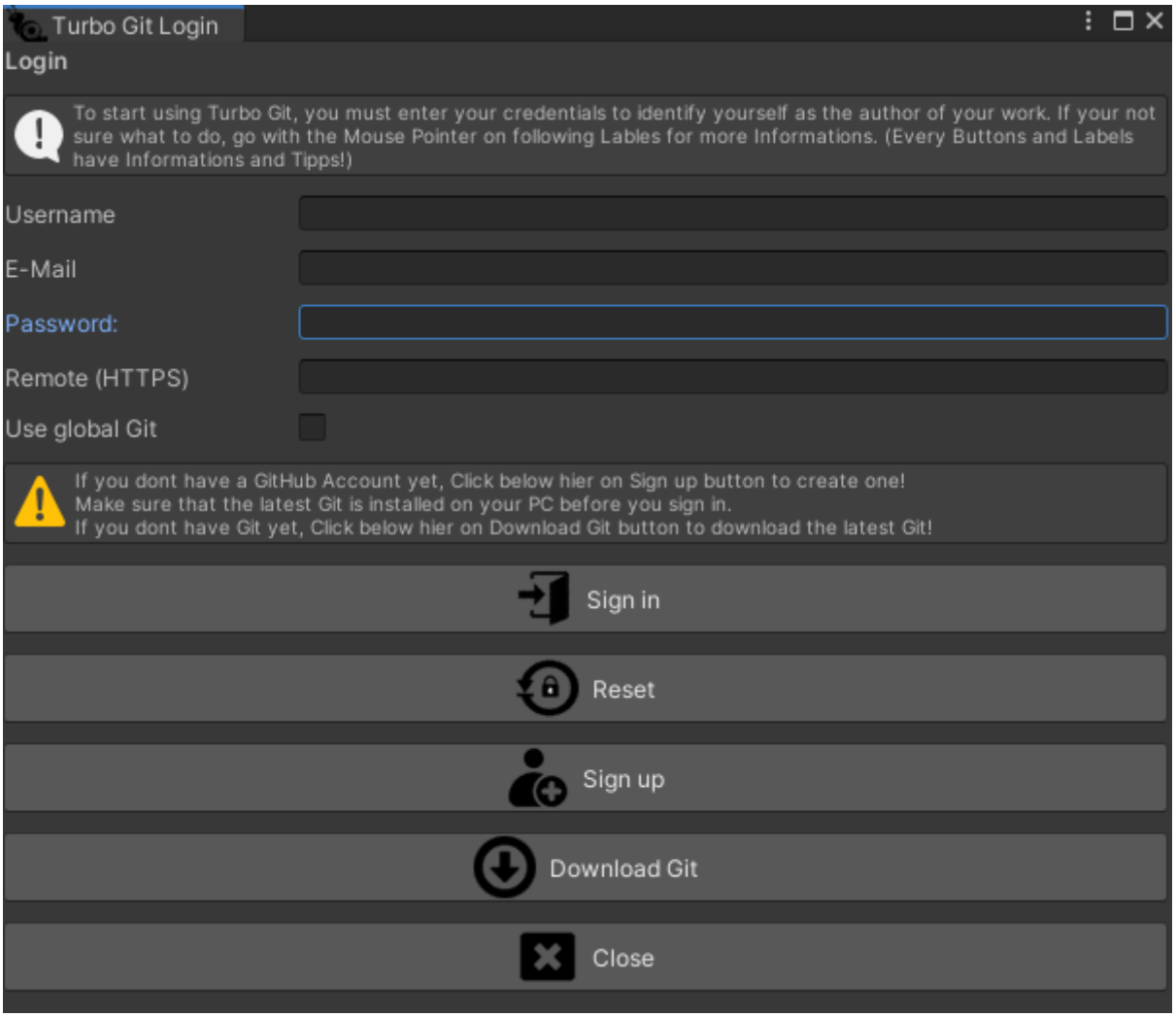
Technical Implementation

Hier are my description of the technical implementation (Diagrams, UI, Folder Structure etc.)

Download Turbo Git and import it to your Unity Project

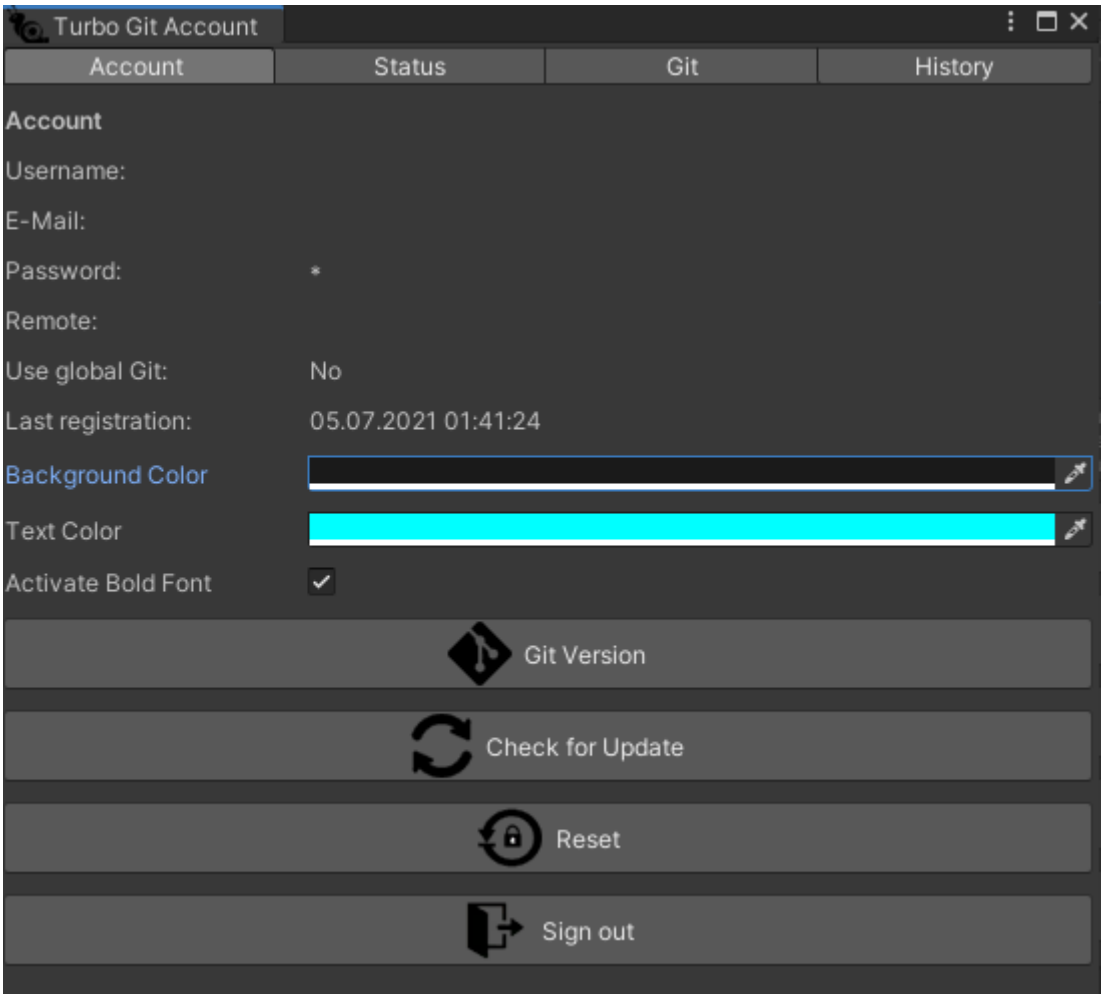


Under the tabs you will see “Tools” go there and open Turbo Git. (Alternatively, you can press CTRL + SHIFT + G)



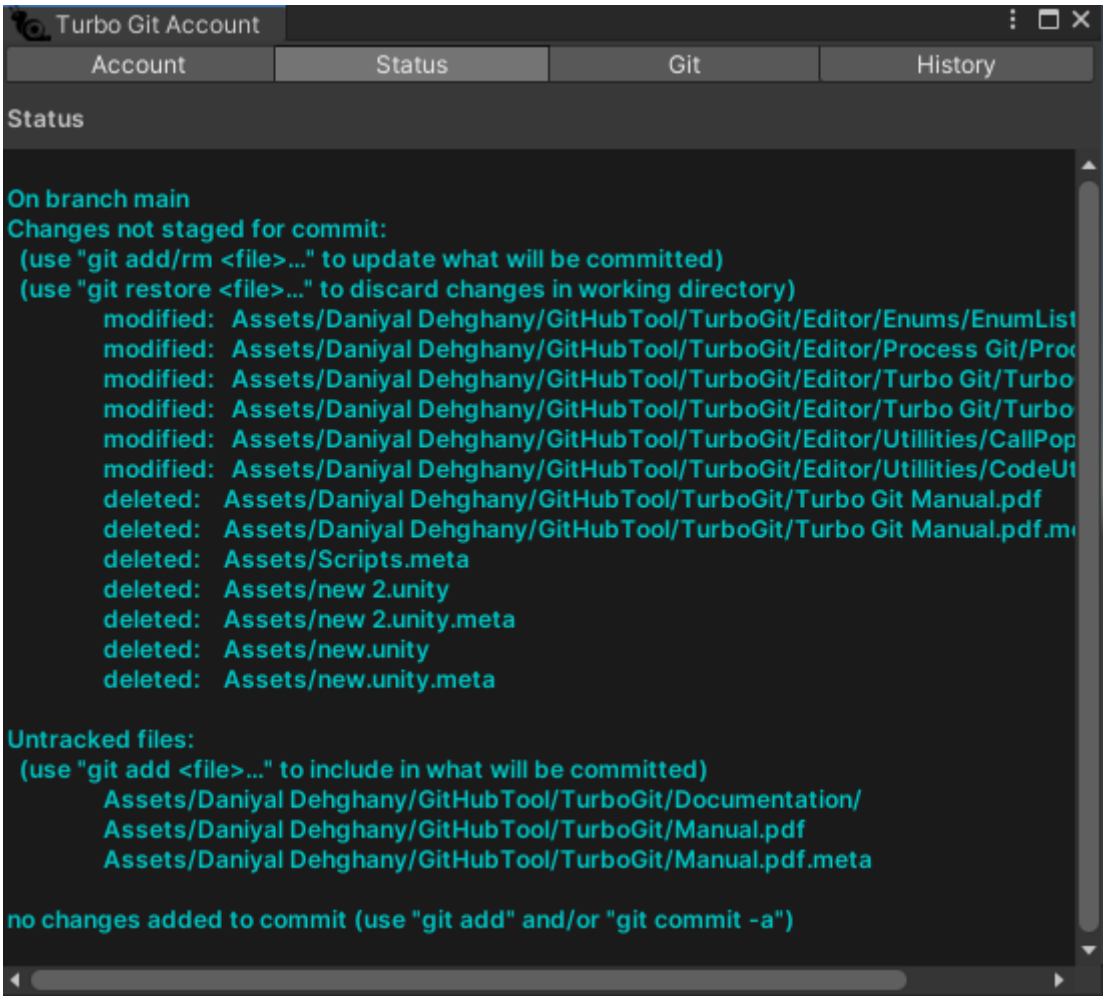
After that, this “Login” window will show up. You can resize and attach this window as you wish. Now you must put all your GitHub Information’s, which this tool is asking for, such as: Username, E-Mail, Password and Remote. (If you’re not sure what to do, you can always go with your mouse pointer to any fields, labels and buttons) Remember that everything has a Tool tip. Heir you can sign in if all Information’s are available. (Remember its case sensitive and everything matches exactly) If you don’t have an Account, you can press on “Sign up” button. It will forward you to official GitHub Website to create a new account. If you have not installed Git Bash yet, press on “Download Git Button” it will forward you to official Git Website to download the latest Git.

If anything went wrong, you could always go back and change your data. If you press “Sign in” button everything will be saved, so that you won’t need to give your Information’s again. (Your password stays anonym and will be saved as random characters, so no one has access to)

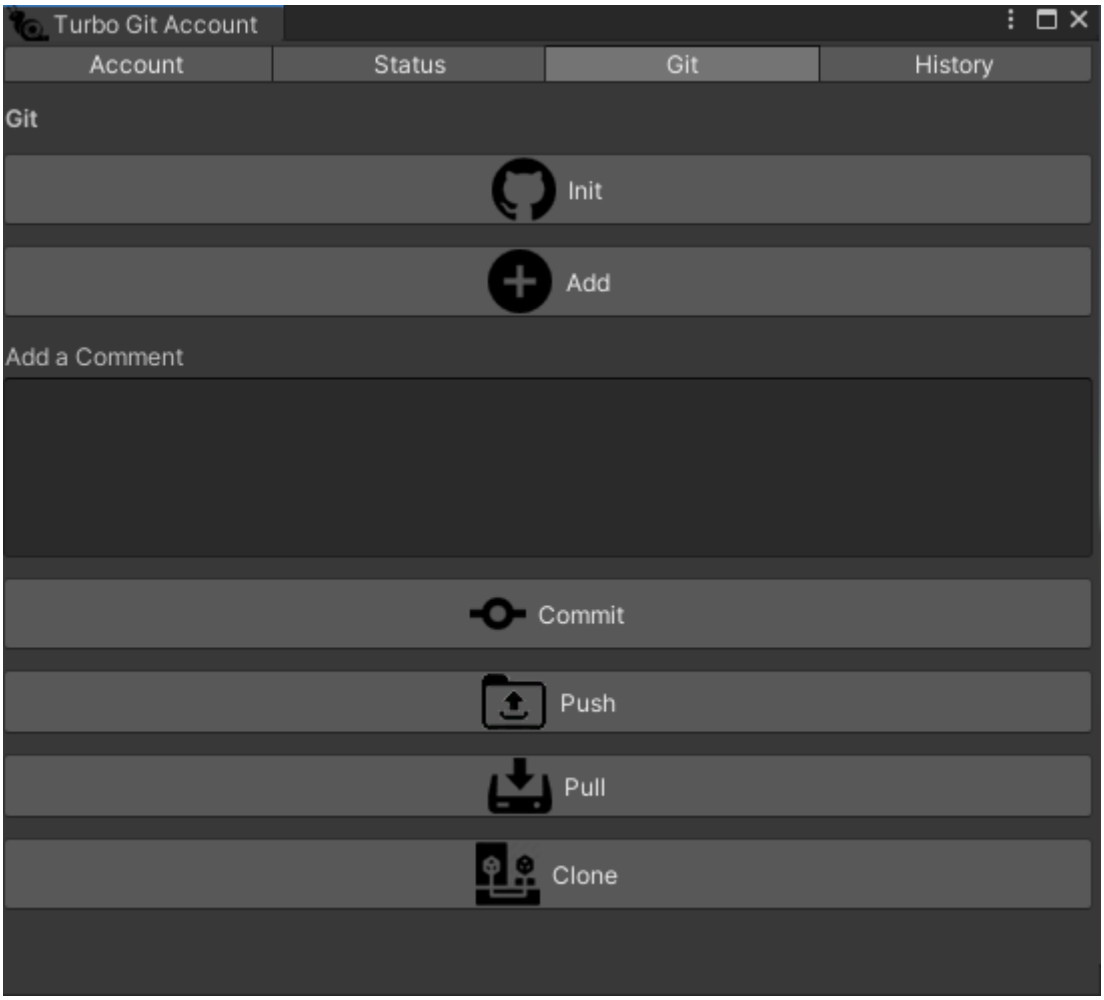


If your Sign in was successful, Login window will close, and this new Turbo Git Account will show up. As Login window, you can resize and attach it as you wish. This window has 4 different Tabs: Account, Status, Git and History. In the first Tab (Account) you will basically see all your information’s, which are used in the Login and also your last registration! Here you also will be able to customize the background and text color and even use bold font as you wish. Everything will be saved, if you change to a new customization for the next time. Pressing Git Version button will tell you your installed git version on your pc. Check for Update button checks if any update is available. If so, you have to update your git manually! The Reset button will change your customization back to default. And if you press on Sign up button, you’re going to sign up from Turbo Git and next time it will show you your last registration. Attention: all your customization will only save, if you press sign out, else they will be lost!

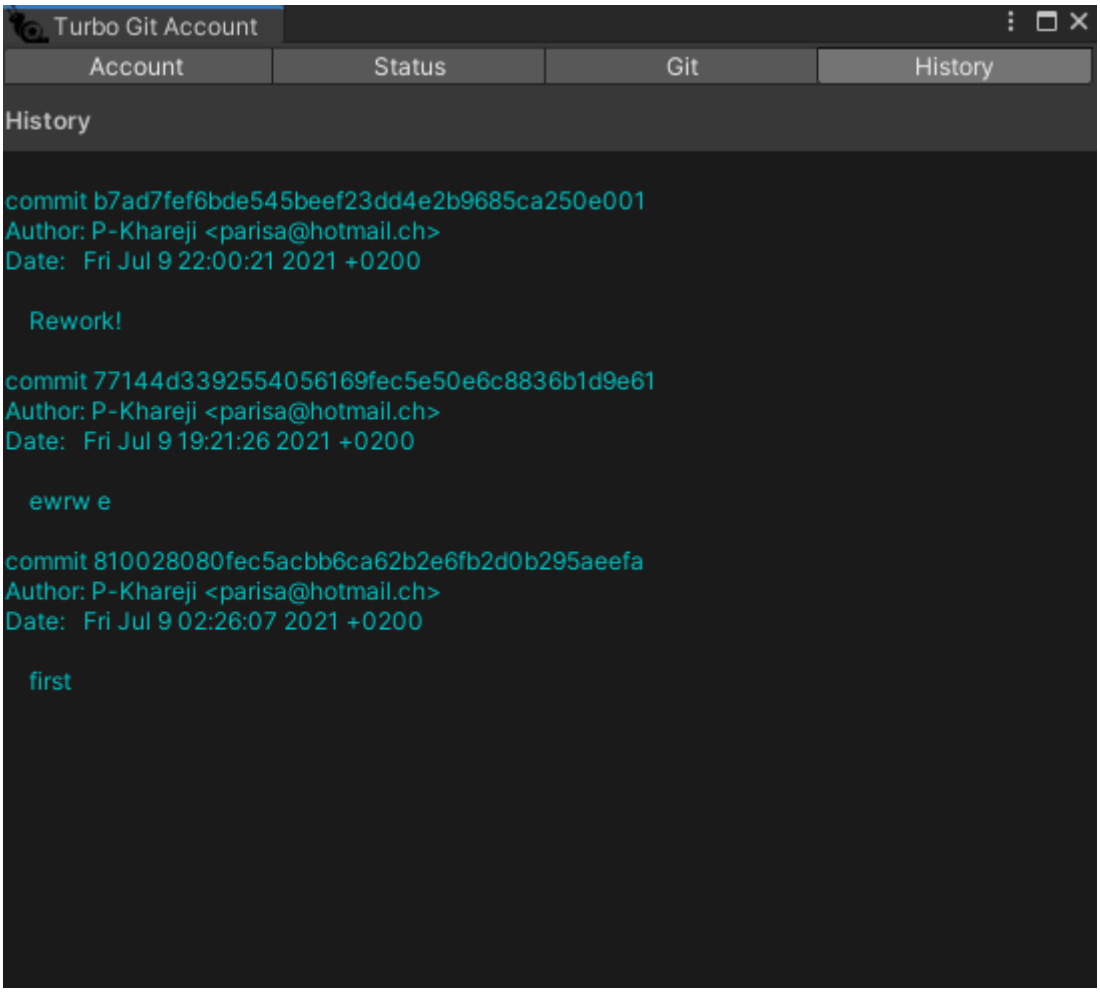
Remember that your customization is only meant for Status and History Tab like the example in the below!



The second Tab (Status) you will get all of your project information’s, based on what your actual state at the moment is. Both Status and History Tab get updated once per seconds, so if you change anything you will get to see it in a second.

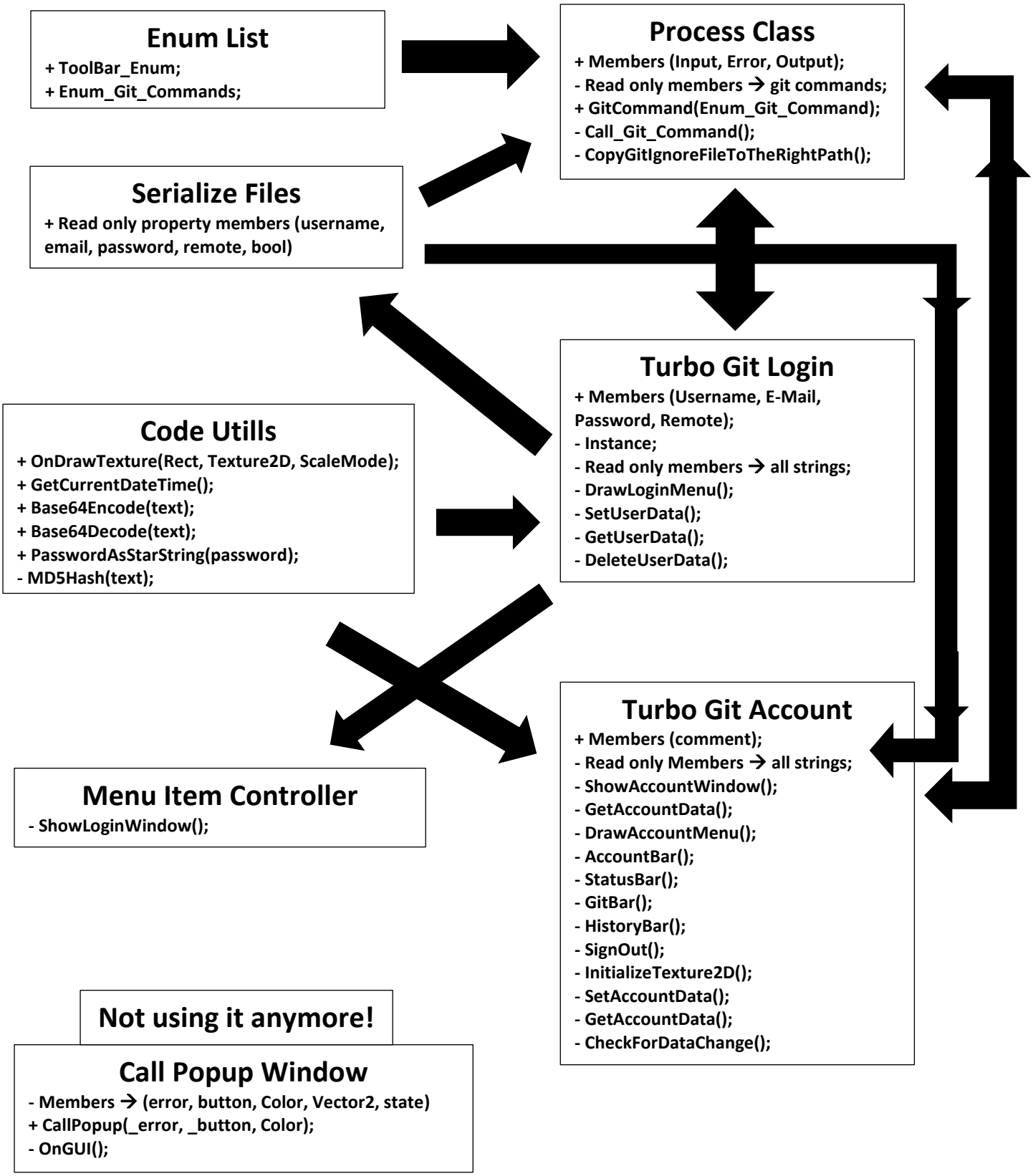


The third Tab (Git) is the most important Tab. Here you can form a new repository to push it on GitHub and even cloning your project as a backup like for your self with just a button click. As said, everything has a Tool Tip. You can basically start from top to below. With Init button you will create a new Repository. This will help you to create a connection with your GitHub Remote address for the connection. After that have you the Add button. This button as the name says will add all of your necessary project files to your Git. You can always see on the Status Tab, what your actual state is. If anything is added to Git you can now Commit your changes. Commit is only a comment for your self to understand later, what was the purpose of this codes, which you have added in Git. Git will automatically create Information's in the History Tab if your commit was successful. Also remember that commit always requires a comment! You won't be able to commit without a comment. After that you will be able to Push using Push button. Push means, that the added project with the comment is ready and will be uploaded to the GitHub website. You can Always reach it and get to see, what you have changed, when and why. You can even get your old Code and Script back if anything went wrong. But remember, only after pushing your project will be available on GitHub. With the Pull button you will download and update your project to the latest state if you're working with another device. However, you should use exactly your same GitHub info's. Clone button is remarkably like the backup. It will make a clone (backup) of your actual state project and put it to a folder.



The fourth and last Tab (History) as said will get all your added and committed project state and show it to you. There is the date, comment and any useful informs for you to see. Like the Status Tab it gets one update pro frame, which means you should see any changes within a second.

Technical Implementation



Difficulty and Ease

At the beginning, when I started with this tool, I thought that I wouldn't have it for that long and that I would progress very quickly. However, it was exactly the opposite! I had never made experiences with a tool and didn't know how to proceed, what I need and how to continue.

I found it exceedingly difficult to establish a connection with a program and the tool and to access it. So, I didn't know how to read the inputs, errors and outputs and show them. Besides that, I am not a trained computer scientist, like everyone else, so I usually find it difficult to decide what the right way of proceeding would be! I just do it the way I feel. Of course, it already works at the end, but I'm not sure whether that would be a (professional) approach.

It is also about the structure and construction of the entire code.
I also had to find out more about Unity Editor and visit the documentation, but since I think unity has very, very very good online documentation, I made quite good progress.
However, I could still finish my tool in time, and I am very satisfied with the result.

On the other hand, I am extremely overjoyed that I programmed something towards the tool. I would never have done anything like it and it was very exciting for me and I learned a lot about it and could improve myself and my strengths a lot. It was worth it, and I had a lot of fun with it.

Conclusion

For the most part it has already been said. The module really taught me a lot and I had a lot of fun with my work and my project and could learn a lot new to improve myself and my knowledge, and thus to gain more experience. With the status of my project, everyone can use it, of course it is simply not yet suitable and intended for professional work, but it is great for private people. You have everything you need. Even I, had a lot of fun with it and tried it out hundreds of times during debugging and tried to make it as user-friendly as I found it to be advantageous on a third-party tool. So I gave it to my colleagues and classmates from the SAE and showed them to try them out, they found it very good and everything worked. I'm glad I got such good feedback, that means that you did your job pretty well and well. And of course, who knows, maybe I'll work on my tool at some point and put it in the unity asset store so that others can use it too!

Testing

I will always test the program after every small change, etc. so can i check every step individually to check for various bugs and problems.

I have made the experience that changing / programming a lot at once and then debugging it would be extremely time-consuming, so I can have my work better under control and more safely.

I even tested my Tool multiple times always with any new futures. I tested it on different PCs with different Account and even asked my school friends to test my tool. Everything works perfectly and easily. My tool is finished, and I got it exactly as I expected and planned.

References

Unity – Editor Dokumentation → https://docs.unity3d.com/ScriptReference/Editor.html

GitHub → https://guides.github.com/activities/hello-world/

Git → https://docs.gitlab.com/ee/gitlab-basics/start-using-git.html

Unity – Editro Tutorials

→ https://learn.unity.com/tutorial/editor-scripting

→ https://learn.unity.com/tutorial/introduction-to-editor-scripting

→ https://www.raywenderlich.com/7751-unity-custom-inspectors-tutorial-getting-started

Unity – Editro Book from SAE → Extending Unity with Editor Scripting