CSE 403 Project Proposal

Project Name: TyrantVC
Group members: Carson Wilk, Volodymyr Loyko, Annie Gesellchen, Ben Celsi

Problem

We are trying to create an interface that makes it easier for Maya users to store and access their script files, while also allowing them to revert to previous versions of their scripts.

Maya is a modeling and animation software created by Autodesk that is used widely amongst animation and gaming studios. Most of these studios have scripting teams that create specialized plugins and tools. In order to write these scripts, they must work in Maya's built-in script editor [1]. However, this script editor is lacking in a lot of basic functionality that a programmer would expect from a typical IDE or code editor. Part of the reason for this is that many of the people who write scripts for Maya come from an animation background, not a programming background, so any add-ons or plugins to the script editor must be simple, easy to use, and intuitive.

Motivation

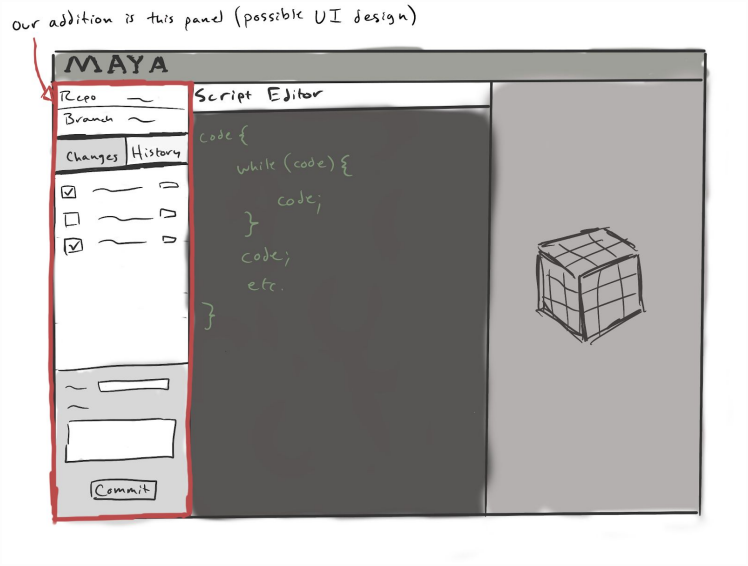
After talking to a script developer who has worked extensively with the Maya script editor, we found that one of their biggest problems is that it can be very inconvenient and inefficient to save out versions of the scripts they are working on and to keep track of changes. This is because any scripts must be run in Maya in order to test them, and if they fail and cause Maya to crash, the developer could lose any work they had in the script editor. Creating a straightforward way to access a version control system from within Maya would help alleviate a lot of these problems.

Currently, there are plugins that allow for auto-completion, better access to documentation, auto saving, and many more features, but no plugin for Maya implements version control for the script editor. One such plugin that we looked at is Charcoal Editor [2], which has many useful IDE-style features but provide version control and also costs $80. If developers want to have version control, they must leave the Maya environment to pull or push their scripts which results in a messy and inefficient workflow. The scripts that are written for Maya are not specific to the project that the developer is working in, so using version control on the entire Maya project would be ineffective and would not solve the problem. Major studios likely have created their own plugins to improve the Maya script editor, possibly including version control, but none of these are made public.

Approach

We want to create a plugin for Maya that adds an additional UI panel to the Maya Environment. This UI panel would allow you to see which repository you are working in, which files have been

modified, and allow for standard Git functionality of pulling, committing, and pushing files. See the diagram below for a rough idea of how this could look:



The plugin will be written in Python and tested using Maya and its script editor. The plugin will consist of three major elements: the UI, connection to git, and the connection between the UI code and the git code. The UI component will be written in using PyQT for Maya and will resemble GitHub for Desktop functionality. It will be mostly simplified to stating which repository and branch you are working in, which files have been changed, and buttons for pulling, committing and cloning/making a new repo. The component that will allow the connection to git will be written in Maya's own scripting language, MEL [3]. This component needs to be tested on various operating systems since access to bash shell in Maya using MEL varies between UNIX based systems and Windows. Finally the component that will connect the two previous pieces together will be written in PyMEL, Maya's python wrapper for MEL. The purpose of this component is to connect the MEL code and PyQT code into a working plugin for Maya.

Evaluation

Our goal is to allow for a more seamless development experience inside of Maya and our evaluation will reflect that. There are two parts that we will want to evaluate: the correctness and the functionality of our plugin. The correctness can be tested through comparison to a predefined specification file which would define all of our plugin's functionality. Some of this could be done with automatic testing and some would have to be performed by hand. To test sufficiency of the functionality we will perform usability with a small user base. We have access to the Animation Capstone students and staff who we will ask to perform a series of tasks using

our plugin. We will log all of their actions and see whether they are able to perform these tasks successfully or run into problem with our code or not sufficient enough functionality. We have already discussed this user study with them and they are willing to help us out and are excited to see what we come up with.

Risks and Payoffs

The main risk of the project is that we have no experience building Maya plugins, and only two of us have experience using Maya at all. For this reason, it is hard to estimate how steep the learning curve will be, and how long it will take to complete our various goals. This could potentially cause us to miss deadlines, and hinders our ability to plan for the future. One way we can help mitigate this is by consulting with a staff member of the UW animation lab who has experience working with the Maya UI and writing plugins for Maya. Given our inexperience with building Maya plugins, there is also a risk that we will come up against an obstacle (say, a feature that is necessary for our project that is unsupported in Maya's API), which could require a workaround that is overly complicated or sub-par (or both!)

Another risk, like with any version control software, is that we are handling our users files, and problems with our software could potentially cause these files to become lost, inaccessible, or otherwise damaged. This is less likely given that we are using Git to handle most of the actual version control, but it is still a real risk that we should keep in mind.

Application

The target audience for this plugin would be casual Maya scripters who would like to use Git to manage their scripting files. This piece of software is meant to be fairly small and easy to understand so it would be possible for someone to build on top of it and modify it for their own needs. We imagine people in schools using this as well, as it would be free and easy to install in a vanilla Maya environment where not many other plugins exist.

The current process for a user trying to script in Maya follows these steps. Let's imagine you are trying to write a script that places some cubes around in the scene using a new UI element. First you would write the scripts in Maya's script editor. You would save them to python files on your computer. You would attempt to run your code in Maya. If your code works the panels show up, if it does not, Maya crashes and any unsaved work is gone. Let's assume your code becomes fairly long and you decide to use an IDE. Now to run your code you have to save everything, open up maya, open up the files there and then run them. If you want to use any Git commands you have to run them outside of Maya, on the files that you've created.

With our plugin the user would get access to intuitive and simplified Git functionality right inside of Maya. As the user works they would be able to save to a repository which they would be able to access from Maya on other devices with this plugin. Saving should feel more reliable and loading up the last iteration of scripts should be quick and easy.

Schedule

Week 4 (1/29):

Finish project proposal. Interview Maya users to draft up the specification file for our plugin listing key elements of its functionality, as well as a list of “nice-to-have” features

Week 5 (2/5):

Test each of the technologies needed for our project, and create a detailed architecture of the program and how these technologies interact. Write a manual for our program.

Week 6 (2/12):

Front end: Have simple Maya plugin working, which displays a disconnected interface for the program’s intended functionality

Back end: Implement the functions required for communication with Github, have a basic interface for the front end to interact with

Week 7(Midterm, 2/19):

Connect front end and back end, to create a usable version of our program (functional, but unpolished)

Week 8 (2/26):

Conduct user tests on with at least 3 Maya users:

- Test their ability to understand the program without guidance

Given a request to fulfill some action, observe how they easily they can carry it out

Interview the user about their experience, what their difficulties were, and what features they would most likely use in their work.

From this, compile detailed list of their feedback and newly discovered issues with our software. Come up with ways to potentially fix or restructure aspects of the program to improve their experience.

Week 9 (3/5):

Complete about half of the bug fixes/redesigns from Week 8, once ready, get more feedback from Maya users.

Week 10 (Final, 3/12):

Have a finished product which implements all of the specified functionality (taking into account feedback from user tests), and has an interface which is easy to use. Implement nice to have features, if time allows.

Citations

[1] Maya Script Editor, https://help.autodesk.com/view/MAYAUL/2018/ENU/?guid=GUID-7C861047-C7E0-4780-ACB5-752CD22AB02E

[2] Charcoal Editor, http://zurbrigg.com/charcoal-editor-2

[3] Introduction to Maya Embedded Language (MEL) for Programmers, https://help.autodesk.com/view/MAYAUL/2018/ENU/?guid=GUID-0F7C50D1-FF45-4868-8EBC-FE044F54B82E

Hours: 6.5