KU Kaddie

Final Project Design

EECS 582: Computer Science Design II
Team 4
February 10th, 2020

Brian Clark, Saharsh Gupta, Alex Johnson,
Brandon Pramann, and Sandra Rech

Project Name

KU Kaddie

Project Synopsis

This program enables a user to record actions taken in the Windows OS environment. The program converts the recorded actions to code which then can be edited and run repeatedly by our program

Project Description

Our project will enable Windows users to eliminate time spent on repetitive tasks. Our project will do the mindless work to allow humans to add value in ways more suited to humans. Although there are already some macro recorders available to Windows users, our macro recorder and interpreter will give the user the ability to edit the code that the interpreter will run as well as add some features that will make the code seem like any other programming language. One of the features that we plan to include is the ability to add flow control statements such as while loops, for loops, gotos, and ifs. The end result of our project will be a program that gives the user all the tools necessary to add the desired automation and streamlining of processes. We hope to make a product that outshines all the similar products by its ability to be both easy-to-learn and powerful.

Project Milestones

1. Planning. (10/31/2019)
2. Specifications finished. (11/31/2019)
3. Research (12/31/2019)
4. Program records keys typed, mouse clicks, and time; program converts macro into code that can be executed by our interpreter (2/14/2020)
5. Extra features such as the interpreter working with control flow works with the interpreter (4/1/2020)
6. Final Testing will be executed; documentation on how to use the software and on how the code works is created (5/1/2020)

Project Budget

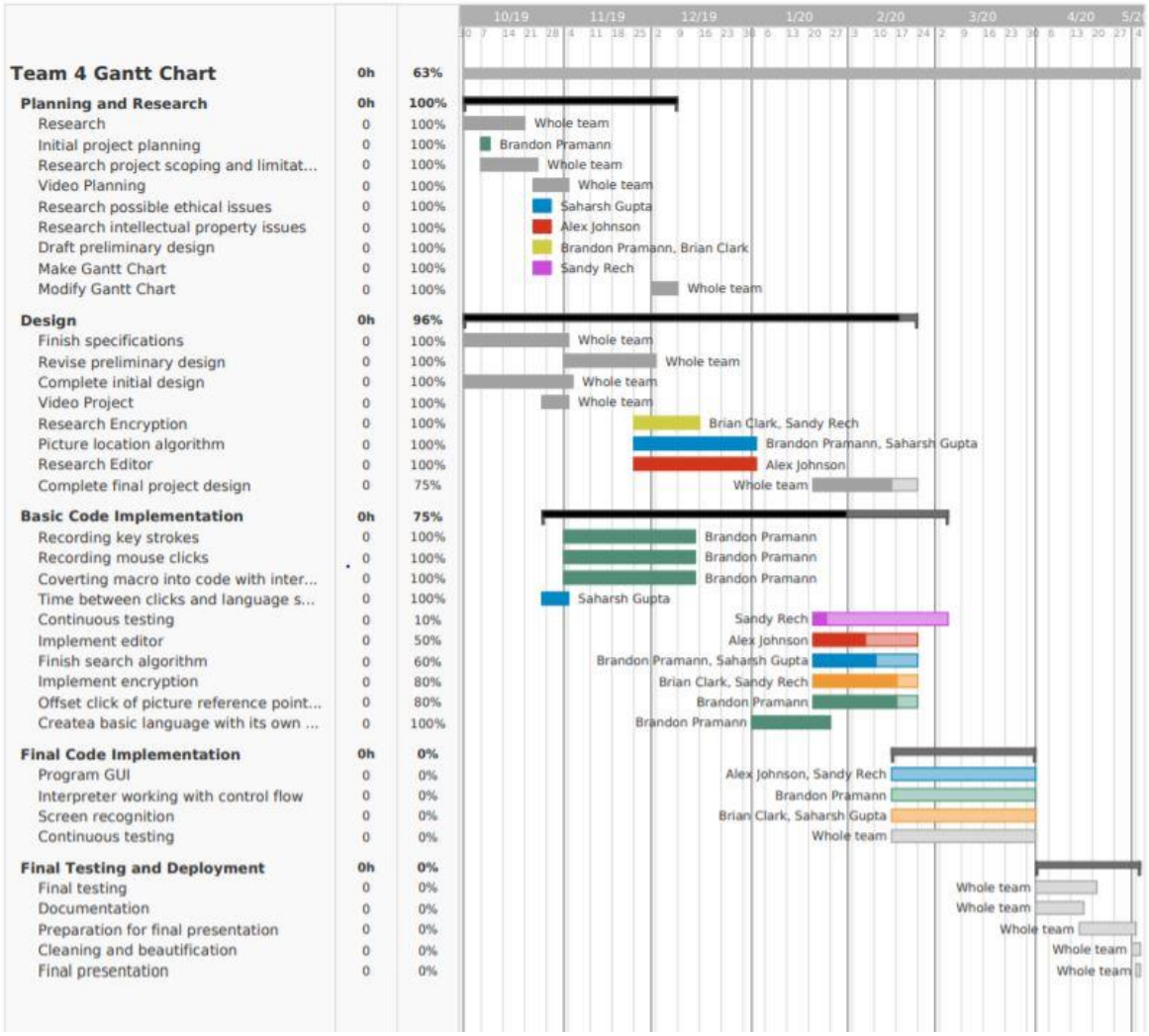
- 5 New Inspiron Small Desktop Requested by (12/1/2019)
- Intel® Pentium® Gold Processor G5420 (4M Cache, up to 3.8 GHz)
- Windows 10 Pro 64-bit English
- Intel® UHD Graphics 610
- 4GB, 4Gx1 DDR4, 2400MHz UDIMM
- 3.5" 1TB 7200 rpm Hard Drive

Work Plan for the Spring Semester

- Brandon will be the project lead and will make sure that the project is reaching the necessary goals by helping teammates do their best

- Saharsh will solidify the picture matching algorithm as well as work on making the language look and perform gracefully
- Brian will make sure that the recorder is always recording to the updated language
- Sandy will perform testing and add language commands
- Alex will create the editor and the GUI and add language commands

Gantt Chart



Link to chart:

https://docs.google.com/spreadsheets/d/1-wAeClu1Hx5rKbMRQiyamIQJo1dhd1ojhhYJLZFai6I/edit?usp=sharing

Preliminary Project Design

For our EECS 581 and 582 project, we have chosen to implement a software program that we are calling the “KU Kaddie”. The name is a play on the word “caddie” which commonly

refers to someone that assists a golfer. While our program has nothing to do with golfing, the software is designed to perform as an assistant to the user. The program will work by recording the user's movements about the operating system. The mouse movements will be tracked and the keystrokes will be recorded. For example, if the user visits the same websites each morning upon arriving at work, checks their email, and logins into a server with a username and password, the program would record all of these actions. After the user has recorded the actions, they would like the program to be able to automate for them; the user will simply have to start the program and the automated process would begin. The tasks would be completed as the user has requested, in a much faster and more efficient manner. This automated process would allow the user to be more efficient in their daily activities.

As a team, we have some specific goals in mind for project implementation to assure that our program is capable of performing user-defined tasks. We are diligently working to ensure that our program will have the ability to record user keystrokes and save them to a file. Our program will record the coordinates of the mouse clicks and take a small picture or screenshot of the area immediately around the mouse click location. We can then use these coordinates to validate that our interpreter has located and will click on the correct spot. We want to ensure that our program records which mouse button was clicked and at which location the click occurred. Starting at the original mouse click location, our program will search around the screen and systematically search the screen in a broadening range to make sure that the interpreter is clicking the same spot that the user clicked. We cannot guarantee that our interpreter will always click the correct position on the screen if the user has moved the associated program from its original location, but also because it is possible that there are duplicate buttons or other things on the screen that appear alike, and it is possible that our mouse click will click the wrong button or spot on the screen. We also are not guaranteeing that the size of the picture around the mouse click stays the same as we may allow the user to change the size of the picture, or we may make the size very large or small depending upon how much detail is needed, or what turns out to be more efficient.

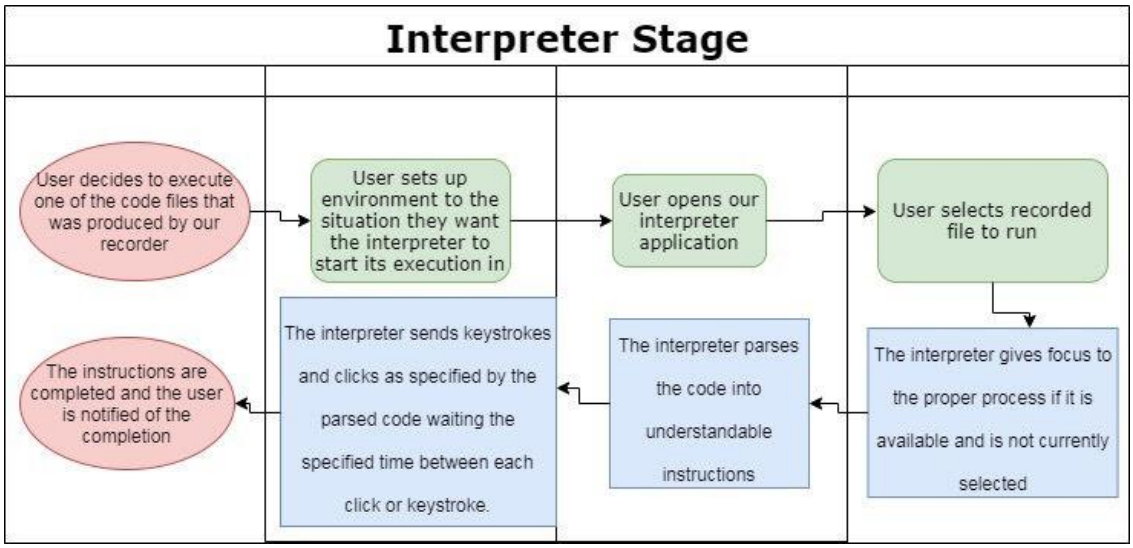


Figure 1: Interpreter Flowchart

We are confident and guarantee that the program will be able to record the amount of time between clicks or keystrokes as the user enters input. The user can also decide when they want to record time between inputs. If the user does not need a specific amount of time between actions, they will have the option of setting a minimum time between sent clicks or keystrokes. These features will enable the user to greatly increase the speed of a recorded macro. The program will be capable of recording the name of the focused process so that that environment variable can be held constant for the user at runtime. Cases may be encountered where the user runs the executable recorded program. For example, if they have many processes open at the same time, we cannot guarantee that our interpreter will give the correct program focus. However, we will do our best to ensure the correct process will have focus as long as our environment remains stable. Our program will also have the ability to work as a pseudo background process so that the user will be able to do whatever they want to while the program is running. This is as long as it does not directly interfere with the environment that our program is directly working with. Our program will switch the context between the user's context and the program's context every time that our program hits a wait or completes a wait.

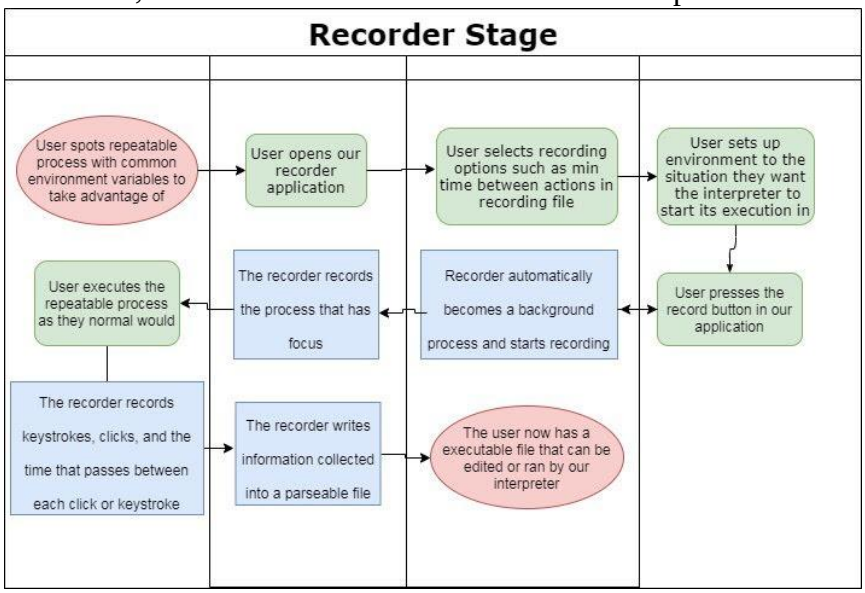


Figure 2: Recorder Flowchart

A certain problem our program may encounter is that our macro recording solution will be unable to solve the problem of duplicates in the executing environment, ie. identical exit buttons shared by separate programs, having multiple tabs open for the same website, or having multiple instances of the same program open at the same time. The program would have no way of discerning or distinguishing between these duplicates in the environment. Unfortunately, we are left with very few options as to how to ensure our program would operate correctly under these circumstances. Clearly, this is a difficult problem that we have discussed in-depth and have been unable to find an accomplishable solution because our program does not have enough information. The user would have to be made aware of this problem and attempt to avoid these circumstances in order to use our program accordingly.

Additionally, another problem that may arise within our program would be when it encounters a change that has occurred in the environment since the program was recorded. A solution we intend to try and implement will attempt to work around this problem by taking a picture around the mouse and keeping track of what process has focus. In that way, we can match the picture around the mouse click as closely as we possibly can, making sure that the right process has focus. A larger and potentially more invasive problem that may be encountered is the ethical problem of recording user passwords and whether we should allow the program to

be able to do so. If allowed, we would then have to decide how we would go about being able to store the information. One solution is that we could possibly implement would be adding a translator in order to enable the ability to encrypt the data entered by the user. Our program will not have any good way to discriminate between keystrokes entered to know that what a user is typing is a password. Our solution will be to encrypt the recorded user's file. The only way that a user can edit the file that is recorded is to open it in our editor which will be password protected. The editor part of our project will open the password to protect macro files as well as displaying a text editor that has the unencrypted code. The editor will then give the user the ability to edit the code and add control flow type statements. The editor will have a save button that will enable the user to save changes. When the user presses the save button, the editor will check the syntax of the code and make sure that code is executable by our interpreter.

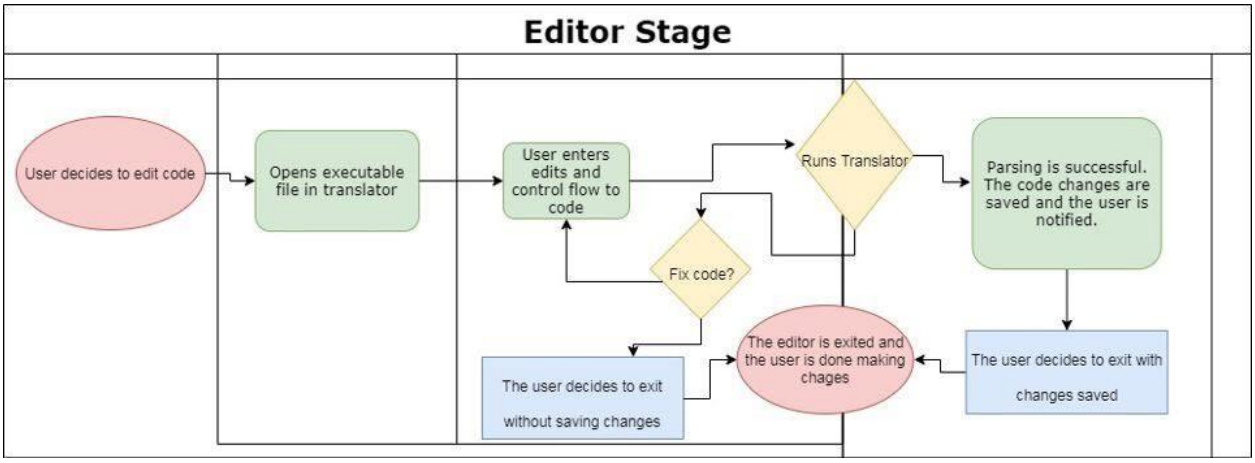


Figure 3: Editor Flowchart

Ethical Issues

As a macro recorder, our project KU Kaddie can complete many tasks and operations. Among those are a few which could violate some ethical issues. Principle 1.2 Avoid harm of the ACM Code of Ethics states that any computing software should not have any unjustified negative consequences. Our project, as a macro recorder, records everything a user does (keystrokes and clicks) for the task the user wants to automate in the given environment. In the process of recording the task the user wants to automate, there could be some confidential information that could be recorded by our program. An example of this is user passwords. To make sure nothing confidential or otherwise is not accessed by someone not authorized, as stated in the Principle 1.7 that is Honor Confidentiality and Principle 1.6 that is Respect Privacy, our program stores the recorded information in an encrypted form. To further avoid unauthorized access to the macro itself, the recorded macro could be password protected, and the recorded macro is stored on the user's machine. Our program also has no way of transferring any kind of information, ensuring the macro or any information stored in it is not disclosed to anyone. Our team will follow the ACM code of ethics to honor user confidentiality when designing and building our recorder.

Intellectual Property Issues

When designing or inventing, it is imperative that anyone else's copyright is not infringed upon. To do this, one must know the bounds of what copyright infringement is. From the U.S. Copyright Office, its definition of copyright infringement states that it is the reproduction, distribution, performance, or derivation of work without permission from the copyright owner. As we build our project, we must take certain actions to avoid infringement. This includes not copying any owned code found to be relevant and helpful to our project. ACM code of conduct principle 1.5 states that computing professionals should respect the work required to produce new ideas, inventions, and creative works. Principle 1.2 of the ACM code states that professionals should avoid harm. Avoid harm, in this case, is pertaining to any harm to the individual or to society as a whole. By using someone else's code, the original owner may suffer potential economic losses, which is detrimental to society. There are some particular caveats that could be useful to our team, as long as we decide not to monetize this project. In copyright law, there exists a doctrine known as Fair Use. The Fair Use Act was passed in 2007 and allowed use of copyrighted material by non-owners. From this act, anyone accused of infringement can defend themselves with a claim to Fair Use, which is judged through a few characteristics: the purpose and character of use, the type of copyrighted work, the amount of the copyrighted work used, and its possible effect on society or the individual holding the copyright. Under this fair use principle, our team could use parts of owned code that would benefit our project. If we decide not to monetize, and develop the project for scholarly purposes, we have an even better claim to Fair Use. By respecting the ACM codes and following other laws or acts, our team will ensure that no copyrights will be infringed throughout the duration of our project.

Change Log

1. Password protection: We are debating encrypting user passwords and are coming up with ways to do so. This was changed to ensure user confidentiality
2. Added GUI aspects: We added GUI aspects for a user-friendly program so that they can easily record their tasks.
3. Environment duplication issues: The user must understand the constraints of our program and cannot have duplicates within their environment.
4. Process Recording: Recording of the processes open and which one has focus will be impossible as far as we can see because there is no identifying factor of what process is which other than name. The name will often have many duplicates and is likely subject to change. Unfortunately, we cannot record with process ID as that will change too.
5. Search Area Restriction: We added the ability to programmatically restrict a search area to match a mouse click picture in order to increase the speed of the algorithm if speed is necessary to the user.
6. Added "waitfor" command: this command allows the user to make a more responsive macro. This enables the macro to be used with a hotkey. We are going to look into adding events.