Proposal

For my term project, I would like to make a simple but fun python editor using tkinter. It has the major features of a python editor. Python keywords, single-line comments, multiple-line comments and other codes would be shown differently. Users can also open file, save file, copy, paste, indent, dedent, make comments and so on. There would be a pattern drawn around each word the user type in. The pattern would be a growing fractal or some emitting dots and triangles, depends on what the word the user is typing. When the user press delete/backspace, the animation would proceed reversely. Another interesting thing is, if the code that the user is writing is having bad style, the color of the picture drawn would be dimmer and darker. In order to make the background of the editor good looking, the user should keep a good style.

To make this editor, I would use what I have learnt about event-based animation, string, and object-oriented programming most. In each files the user creates on the canvas, I would break down the whole file into code snippets or comment snippets, then break the snippets into lines, and then break the lines into words. Word class have sub class pythonKeyword class, line class has subclass of one-line comments, and snippet class has subclass multiple-line comment. They would be shown differently in canvas. I then will try to locate the current location of the cursor, and to highlight and draw things there.

I will break this problem down into 4 major parts: 1. Create a notepad to type on, 2. Identify python keywords, python codes and comments and check the file’s style, 3. Turn the notepad into a python editor by combing step1 and step2. 4. Identify the location of the cursor so that the user can highlight and draw something fun there. This step is the trickiest part in my project. I would present my algorithmic plan below.

First, I would use what we learnt about event based animation to make something like word or notebook. Basically, it’s draw the character I pressed, and there would be some buttons to save, open, or discard the text created. There would also be some buttons to edit the file. If the top-level buttons are pressed, a menu would unfold and more specific operations choices, like copy or indent, are shown to users. If the user presses the lower level buttons, command would be operated. It will have basic file I/O, and it can modify the text the user written to create indent.

Second, I would write another program to identify whether a piece of string is code or comment, and to identify python keyword. The first part is a little bit like a bonus problem in hw3, the topLevelFunctionName(code) problem. I would look over the python API and learn more about the styles of python code, and find out python keywords. This program should also be able to check the style of the code.

Third, I would import the program I wrote in the second step, and use it to evaluate the code that the user writes, and comments and python keywords would appear differently on the canvas. Python keywords shall have different font from other codes, and comments would appear in different color.

Fourth, I need to locate the cursor. I would do this by first create a cursor on the canvas, showing where is about to be edited. Then I would store the current locations of event.x and event.y into data, and draw something like a growing fractal at this position in redraw all. If the user wants to highlight something, I would store the endpoints in data, too. Then, using some geometry, draw the highlighted area as rectangles.

The changeable size of canvas also makes this project trickier. I can do this by changing the parameters in canvas.pack().

The only module that I will be using is tkinter. The class that I will use the most is canvas class.

Update 1:

Description:

When the user finish typing a word, a small planet is drawn, and it starts to move along a circle centered at the top-right corner of the window until it moves out of the canvas. The color of the planet is generated randomly, the size and shape of the planet is decided by the length of the word that is just typed. The color of the planets also will change according to the style of the code. If the user is writing with terrible style, the planet becomes darker.

When typing, the editor also makes a small window for auto-completing so that the user don’t have to type everything over and over again.

Algorithmic plan:

About how to draw the code and locate cursor and highlight:

I choose a monospace font for the texts on my editor so it’s easier to locate each character. I used the built-in cursor and highlight in tkinter canvas’s text object. But in this case, I cannot change the font and color of text. So I have two functions. One is called drawContent, which draws all the text altogether in the same size and font to locate the cursor. Then I use another function called drawCode to draw a second layer of code with different colors and fonts for python keywords with function to identify python keywords and function names. The drawCode calls drawLine and drawLine calls drawWord. I break the whole text down in to pieces to determine the color and font for each piece.

About zoom in and out:

Before the code is written, I calculated the width and height of each character of the font I choose and store them in a list in init(data). When the user presses zoom in and out, the size value changes and my editor would find the new corresponding width and height for it and redraw everything else.

About draw the planet:

I have a planet class that control all the planet drawn. I calculate the tangent line of the position of one planet and the top right corner and use trig to find the dx and dy to be used in timerfired. If the user presses backspace, the speed value of the nearest planet that the deleting takes place would become negative and the planet would move reversely. I would also have a styleCheck function to check the style. I would call it in timerFired and be ready to change the rgb value of the colors of planets when the style becomes bad. The planet would appear a little bit lower than the text so that it does not affect user from reading code.

About saving and opening other files:

My text editor can open more than one file at a time as well as saving and opening new file. In this part, I draw a small window for users to choose the directory. Directories are listed by os module and I draw them on the canvas so the user can click and change directory. The appearance is decided by a value called data.open. I also create a scroll bar on my own in the similar manner I learnt on the lecture (scroll demo).

To open more than one file at a time, I use a file class which stores the name, path, and content of every files. I store all those files in a list, and change data.currFile base on mouesPressed.

Auto- completing:

I would also store all the function names and variable names in a file object. I would also draw a small window and scroll bar like I did in showing the directories to save and open file. The user can choose these names by mouseClick.

Scroll:

I will draw a scroller on tkinter canvas by using mouse-motion event and what I learnt on scroll-demo.

Indent:

The editor automatically indents the next line by 4 spaces when a ‘:’ is typed. I have a number to keep track of the indention number in data to do this auto-indention thing.

Make indent, dedent, comment and uncomment:

I do indent, dedent, comment and uncomment by breaking text into a list of all the lines. Then these function take places on the chosen lines, at the end, I join all the lines back together.