**Javier Cienfuegos & Aleksandra Cvetanovska**

**Motivation**:

One of the projects that was particularly interesting for us was the cave backtracking one. We had a lot of fun creating the logic behind it, and we decided to create a graphical user interface for it, so that we can make it interesting for other people too. Since we just created the GUI, we used the code we already had written for LAB02.

**Purpose**:

Our software is an extension of the Backtracking Cave assignment given to us prior to this final project. The user will be able to visualize the computer logic and how it is trying to find the treasures.

**Audience**:

The intended audience is anyone interested in how computer applications can make use of logical statements and data structures in backtracking. We subconsciously use stacks in our everyday life, and backtracking to find a particular object is a perfect example.

**Instructions**:

This is an extended and graphical version of the cave backtracking assignment done earlier.

First the user enters what file name to. There are three files that the user can choose from:

1.) cave\_sampleA.txt

2.) cave\_sampleB.txt

3.) cave\_sampleC.txt

Afterwards, the user gets to choose what colors they want to use in the application.

All you have to do is go through each list box and select which color do you want

the corresponding element of the map to have.

After pressing 'Submit', the application will run and the user can watch how the logic is

used within the turtle library.

**Design**:

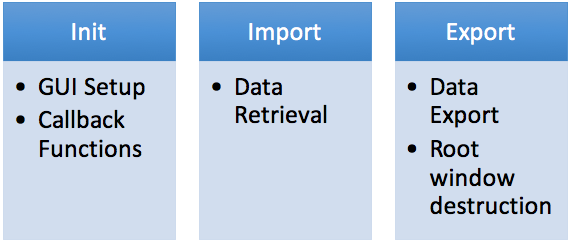
Our design was originally written on blank white paper, but we were able to outline it on a word document. The content is as follows:

Number of classes two encapsulate applications: 3

1. File Dialog
2. Editor Dialog
3. Main Dialog

Functionality description of each application:

1. File Dialog - This GUI application is used to retrieve what file the user wants to use as a representative of the matrix. The matrix build has already been built when we did the Cave Backtracking assignment, so just extract the filename, the file contents, and dump the data into the logic created before.
2. Editor Dialog - This GUI application is used to allow the user to choose what colors to build the walls, path, traversal path, and the treasure. We want to extract the user input and place the colors in a dictionary that maps out the text file content to the actual representation of the map i.e wall, treasure, etc.
3. Main Dialog - This GUI application is used to control all of the canvas drawing. At this point, we have the matrix, the colors to use, and we had the logic to traverse the matrix in a manner where we backtrack if we get stuck. After the game is finished, we output to the user that we have finished.

**Standard template of File and Editor GUI application:**

**Enhancement:**

The enhancements we made mostly deal with user interaction and visual extensions. We figured it would be visually appealing to see how the logic is represented. The users gets to choose what colors to utilize and gets to watch the entire backtracking process live.

**Functionality:**

Gets the user’s input and opens the specified map.

Gets the user’s input on which colors to use and draws the map accordingly.

Runs the program, i.e. does the backtracking.

Checks for treasures.

Displays success message if treasure is found.

**Files:**

* main.py
* GUIApp.py
* Instructions
* pos.py
* Stack.py
* cave\_sampleA.txt
* cave\_sampleB.txt
* cave\_sampleC.txt

**Utilized Data Structures:**

We are using stacks to keep track the backtracking.

We are using dictionaries to map the (i,j) position in the matrix to the actual coordinate position. We are also using dictionaries to keep track of the path taken to find the treasure.

**Big O Analysis:**

The backtracking logic behind the program is of time complexity O(n) because even though stacks have a insertion and deletion time complexity of O(1), the worst case scenario would mean that we have tracked n steps and that we have to access each single popped item from the stack. So although the ‘interior’ methods of the backtrack function are of O(1), the entire backtrack process or the ‘exterior’ method is of time complexity O(n).

The GUI part is of time O(n x m) where n represents the number of rows, and m represents the number of columns. Since we have to draw out the map we have to go through each point which creates this complexity.

**Resources:**

Platform: Cross-Platform

Languages: Python

Libraries: Turtle

Environment: PyCharm

**Challenges:**

We tried using the TKinter library to create the GUI but we realized that it doesn’t do exactly what we wanted. Instead of displaying each move we take separately, it only displayed the whole path which was not exactly what we wanted. Therefore, after some time wasted on trying to figure out how to this, we decided to use the turtle library instead.

**Testing:**

For testing purposes we created multiple files with different maps.

* Map A had treasure in just one place.
* Map B had more rows than map A, and treasure in 2 places.
* Map C had more columns but fewer rows than map A and two treasures.
* Map D had no treasures and fewer rows and columns.
* Map E also had no treasures, but had lots of rows and columns, so that we can check how well the spacing of dots is.

**Errors:**

As far as we know there are no errors.

**Measures and Assessment:**

The main measure of success was being able to draw the map, and draw the path taken while backtracking. Since we managed to implement this using the turtle library we consider our project a success. Also, we tweaked it and made the user able to choose the map file and the colors for the drawing part which was also a success for us.

**Video:**

<https://www.youtube.com/watch?v=YSeMDH1ekc0>

**Summary:**

From the start, the idea was to make a graphical user interface which would allow the user to see the path we take as we are doing the cave backtracking. At the beginning we tried using TKinter for everything which didn’t workout well for us. Because of that, we decided to split the process and make half of it with TKinter and the other half with turtles.

After we implemented the basic ideas, we decided to have fun with it. The next step was to let the user be able to choose what colors he/she wants for each part of the map.

Therefore, the final result is a GUI where the user can select the map he/she wants, can select the colors, and then the program runs the backtracking.   
 As programmers we spent one week implementing everything that has to do with this project.