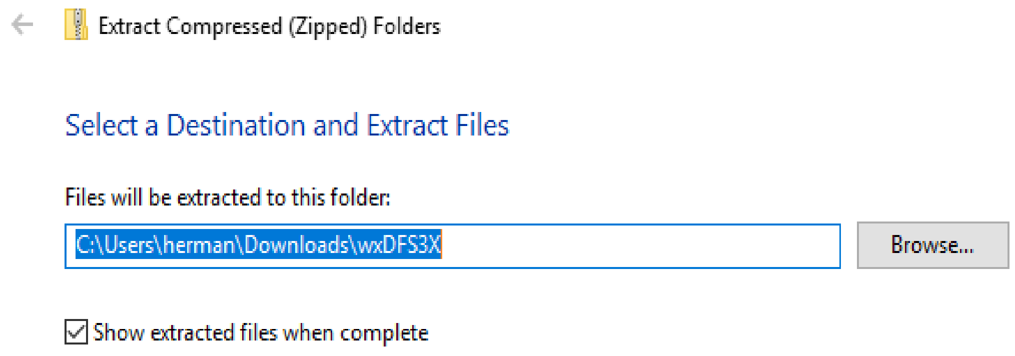
Install wxDFS3 wxWidgets framework and solution

First thing is to download the whole wxWidgets DFS solution.

Lessons Two and Three will go into this wxWidgets solution and duplicate the condition explained in the video thereby limiting the complete solution.

1. Download the zip file wxDFS3X.zip for the solution: http://web.eng.fiu.edu/watsonh/eel3370/Maze/Module2/wxDFS3X.zip

2. Unzip the file into the Downloads folder (Illustration 1):



Select a Destination and Extract Files

Files will be extracted to this folder:

C:\Users\herman\Downloads\wxDFS3X

☑ Show extracted files when complete

Illustration 1 Unzip to this exact folder

3. Start Code::Blocks and create a wxWidgets application using wxSmith and allowing it to create the default source and header files. The default files will be overwritten with the files from the unzipped folder. First create default wxWidgets application (with wxSmith)
   Start Code::Blocks
   File → New → Project

   See the screen shots shown below:

The New Project Wizard starts (each dot is a screen):

- wxWidgets Project - Illustration 14
- wxWidgets 3.1x

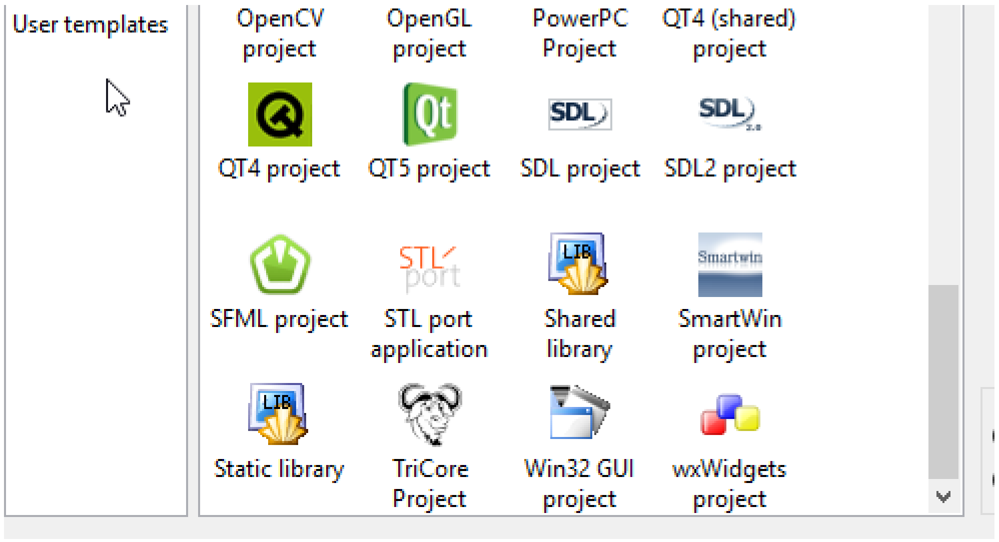
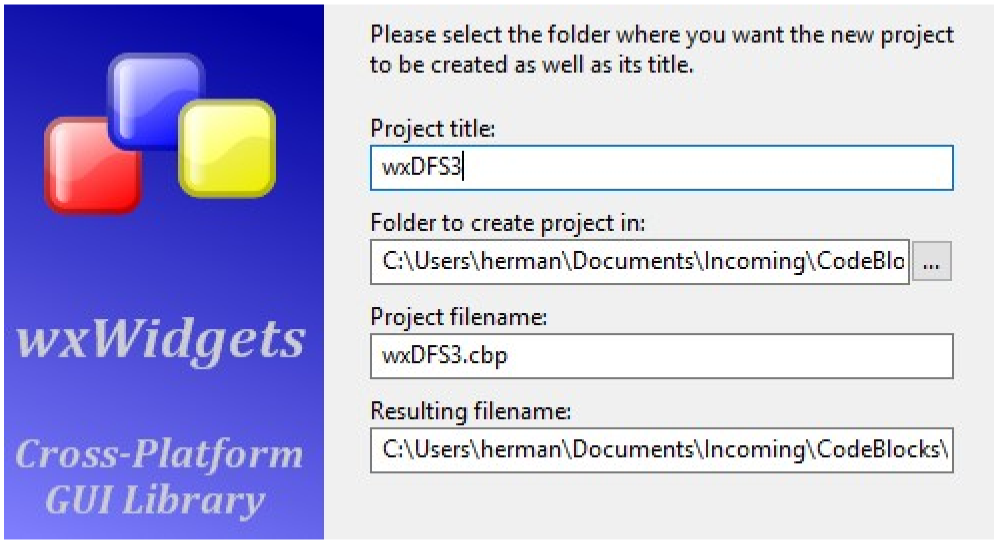


Illustration 2: New wxWidgets Project



Please select the folder where you want the new project to be created as well as its title.

Project title: wxDFS3

Folder to create project in: C:\Users\herman\Documents\Incoming\CodeBlo

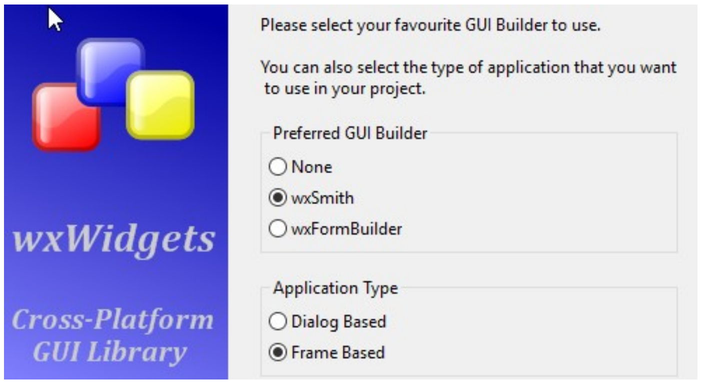
Project filename: wxDFS3.cbp

Resulting filename: C:\Users\herman\Documents\Incoming\CodeBlocks\

Illustration 3: Use EXACT same title

- Illustration 3 Project Title: wxDFS3 << important use this EXACT name in your project folder
- Author: hw

- Preferred GUI Builder: wxSmith Illustration 4
- Application Type: Frame Based



Please select your favourite GUI Builder to use.

You can also select the type of application that you want to use in your project.

Preferred GUI Builder

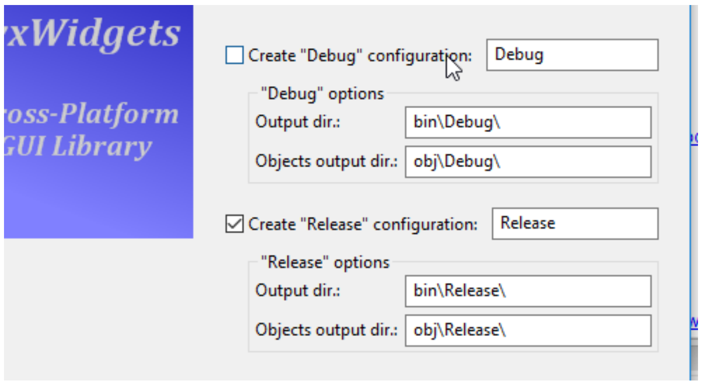
○ None
● wxSmith
○ wxFormBuilder

Application Type

○ Dialog Based
● Frame Based

Illustration 4: wxSmith GUI Builder

- Create "Release" configuration: Release - Illustration 5



☐ Create "Debug" configuration: Debug

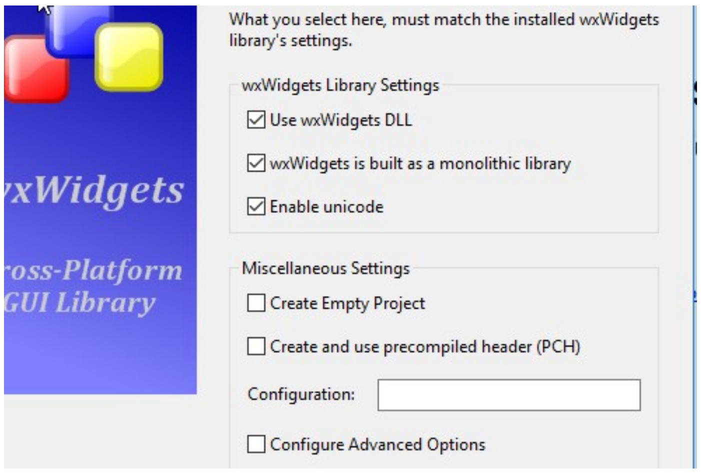
"Debug" options
Output dir.: bin\Debug\
Objects output dir.: obj\Debug\

☑ Create "Release" configuration: Release

"Release" options
Output dir.: bin\Release\
Objects output dir.: obj\Release\

Illustration 5: Create Release

- use Default wxWidgets configuration - Illustration 6



What you select here, must match the installed wxWidgets library's settings.

wxWidgets Library Settings
☑ Use wxWidgets DLL
☑ wxWidgets is built as a monolithic library
☑ Enable unicode

Miscellaneous Settings
☐ Create Empty Project
☐ Create and use precompiled header (PCH)
Configuration:
☐ Configure Advanced Options

Illustration 6: Default Library Settings

- Quit Code::Blocks. <<<<<<<<<<<<<
- Copy files from unzipped wxDFS3X/wxDFS3 folder to your wxDFS3 project folder

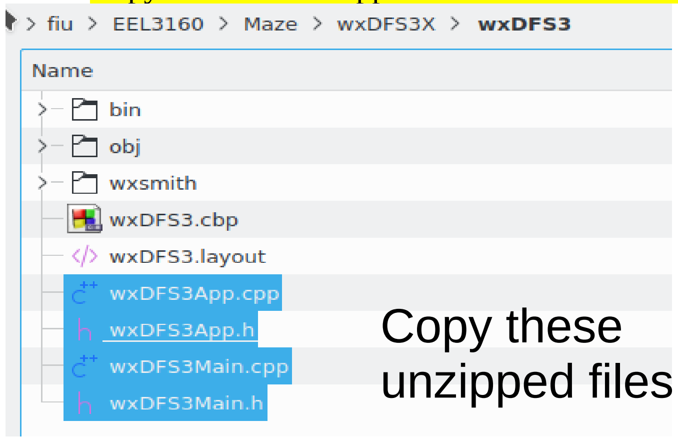


Illustration 7: Copy Files

- Paste into the new Code::Blocks wxDFS3 Project folder and overwrite the files with same name

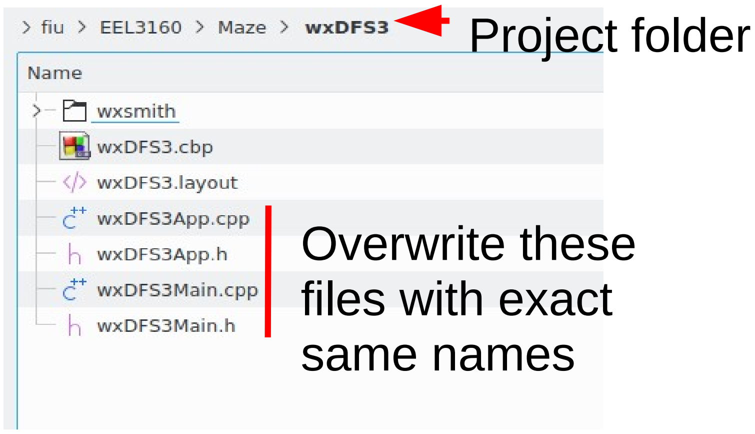


Illustration 8: Paste Files (Overwrite)

- Now back to the unzipped wxsmith folder. Copy and paste (overwrite) the wxDFS3frame.wxs file into the new Project folder.

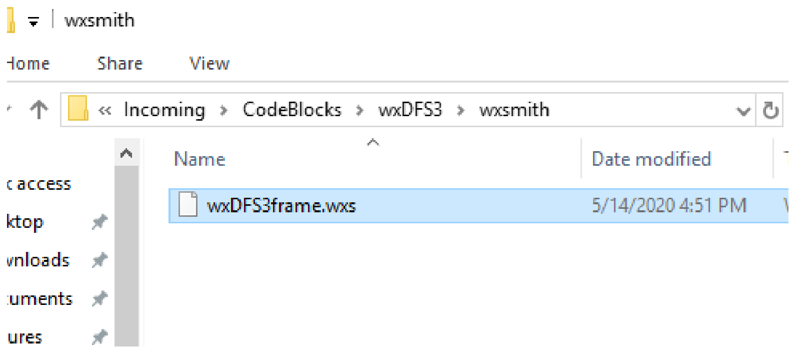


Illustration 9: Copy and Paste wxsmith file

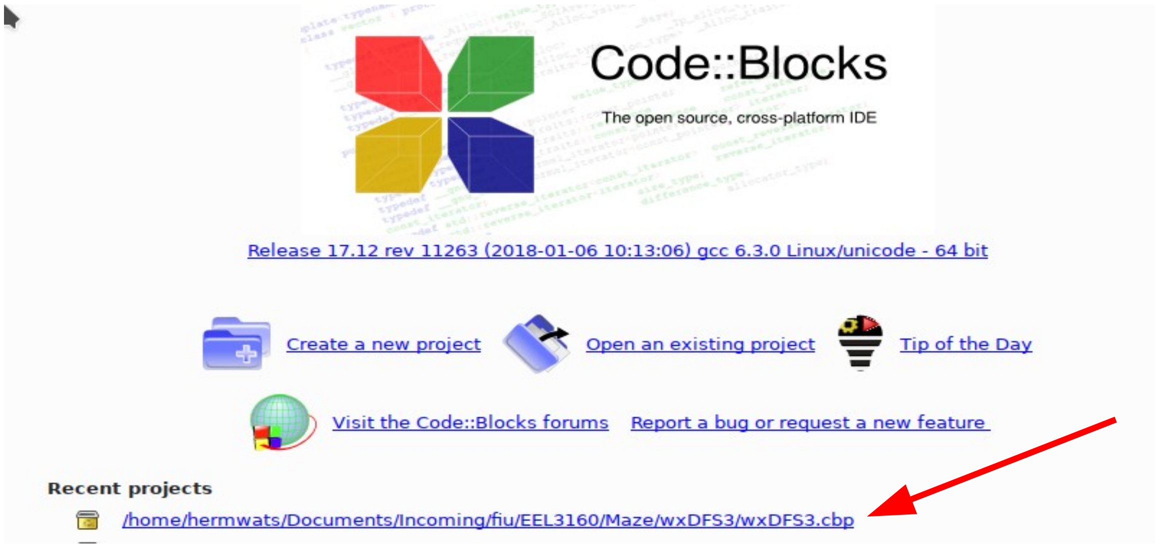

Code::Blocks
The open source, cross-platform IDE
Release 17.12 rev 11263 (2018-01-06 10:13:06) gcc 6.3.0 Linux/unicode - 64 bit
Create a new project
Open an existing project
Tip of the Day
Visit the Code::Blocks forums
Report a bug or request a new feature
Recent projects
/home/hermwats/Documents/Incoming/fiu/EEL3160/Maze/wxDFS3/wxDFS3.cbp

Illustration 10: Reopen wxDFS3 project

- Reopen Code::Blocks wxDFS3 project - Illustration 10

Compile and run the Project:

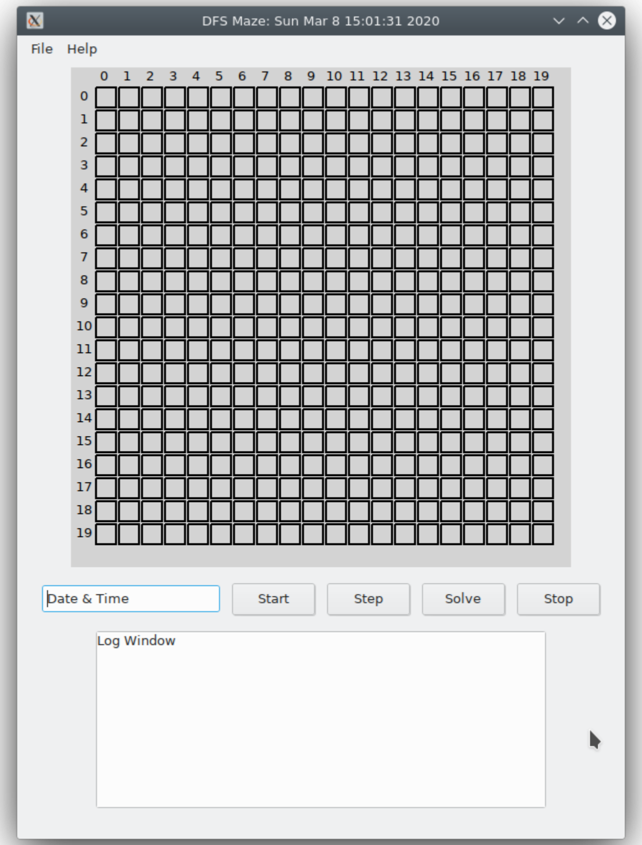


Illustration 11: wxDFS3 Start Up Screen

wxWidgets Code Architecture

wxSmith is used to create the Frame and application code. See the wxDFS3Main.cpp file.

Events are created

1. when a button is pushed,
2. when the timer ticks, and
3. when the Panel is redrawn.

When wxDFS3 is running, the timer (wxTimer) runs a State Machine integrating the application with it's Windows framework. (EEL 4730 :-)

wxEvents generated by application:

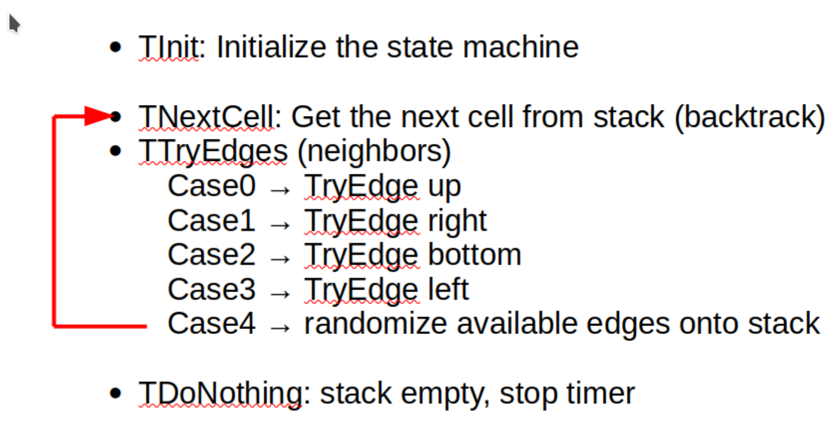
```
•   On Quit:  void wxDFS3Frame::OnQuit(wxCommandEvent& event)
•   On About:  void wxDFS3Frame::OnAbout(wxCommandEvent& event)
•   On Timer1: void wxDFS3Frame::OnTimer1Trigger(wxTimerEvent& event)
        Tick the State Machine

•   On Panel1 Paint:  void wxDFS3Frame::OnPanel1Paint(wxPaintEvent& event)
        Draw the Maze

•   OnButton1:  void wxDFS3Frame::OnButton1Click(wxCommandEvent& event
        Start (Timer) (optional- enter Orgin coordinates)
•   OnButton2:  void wxDFS3Frame::OnButton2Click(wxCommandEvent& event)
        Stop (Timer)
•   OnButtonStep:  void wxDFS3Frame::OnButtonStepClick(wxCommandEvent& event)
        Step (Single step Timer)
•   OnButton4:  void wxDFS3Frame::OnButton4Click(wxCommandEvent& event)
        Solve (Draw path from end point coordinates through Maze to origin)
```

OnTimer1Trigger function implements the State Machine and switches on TickState variable value to pick the State actions: switch(MyPath.TickState)

The four States are as follows:



- TInit: Initialize the state machine
- TNextCell: Get the next cell from stack (backtrack)
- TTryEdges (neighbors)
  - Case0 → TryEdge up
  - Case1 → TryEdge right
  - Case2 → TryEdge bottom
  - Case3 → TryEdge left
  - Case4 → randomize available edges onto stack
- TDoNothing: stack empty, stop timer

Illustration 12: Cases loop implements DFS Algorithm

TNextCell to Case 4 loop is the implementation of the DFS Algorithm in the simplest form.

Watch Coding Challenge #2 video by Daniel Shiffman

Select Part 2 of this link: https://thecodingtrain.com/challenges/10-dfs-maze-generator

Video: time	wxDFSMain.cpp Lines
Part 1 review 00:00	Source split into 4 files: wxDFS3App.h, wxDFS3App.cpp, wxDFS3Main.h, wxDFS3Main.cpp
Cell Object review 00:40	wxDFS3Main.h references Cell Class
Forward tracking 01:00	
Mark cell as visited 01:20	method - Cell::SetCellVisited(bool bvalue)
Current cell 01:45	Line 91 Maze Class Cell object variable
Set Current to origin 02:00	wxDFS3Main.cpp code references Line 292 void Maze::InitExplore()
Mark current as visited 02:20	void Maze::NewCurrentFromStack () Line 308
02:30 Mark visited as different color	MazeSetCellColor(current.x, current.y, green); visited is green Line 322
03:35 While unvisited cells and neighbors Does current cell have unvisited neighbors?	void Maze::TryEdge (int dx, int dy, int from) mark tried as pink
05:08 Check each of 4 neighbors to see if unvisited	Cases 0-3 of State Machine Lines 193-212
05:42 make temp array Neighbors	MyPath.Edges Line 195
05:48 Explore the neighbors	Cases 0-3 of State Machine Lines 193-212
07:09 Index	Main.h MazeIndex (int X, int Y) Line 110
08:20 Have neighbor been visited?	TryEdge Line 414 Unvisited?
08:28 Unvisited – add to neighbors array	TryEdge: Line 422 Add to Edges vector
09:30 Boundaries	TryEdge: Line 406 Test boundaries
11:40 If available neighbors found, pick a random one	Case 4 of State Machine: Line 213 randomize neighbors
13:08 Mark next visited and set to current	case TNextCell of State Machine Line 180
13:34 March along to identify neighbors Run to no available neighbors AND no remove walls	Load the Application and push ‘Step’ button Observe the ‘Current’ (Red) and the neighbors being explored (Pink). The log text shows each Edge being explored and if it is available or not

For this lesson, modify the wxDFS3 solution to place only ONE available neighbor onto the stack. This prevents 'backtrack' of Stack operating for this Lesson.

Also, turn off the part that erases Walls. This way the solution will behave same as Video 2 example.

Note: the explored edges are pink. The solution can be single stepped by clicking 'Step'.

The log information gives details about the algorithm execution. Try single-stepping and see the information printed by the application.

Application modifications:

For this assignment:

1. In Case 4, comment line 225 and add code for line 226 to take only 1 edge to explore

```
//for (int i=0; i<numEdges; i++)
{  int i=0; // only one neighbor, not more
```

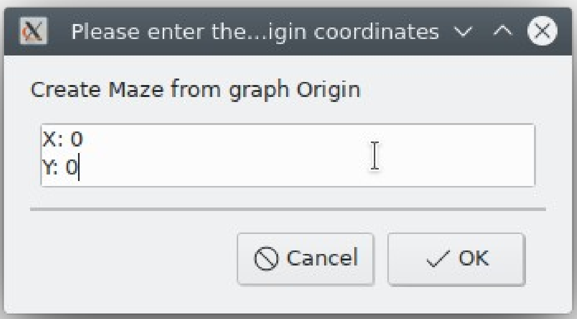
2. Turn off erasing Walls by commenting out line 347 and replace with

```
   //switch(current.from)
   switch(5)
```

3. Compile and run.

The 'Start' button allows specifying the graph origin location.

For now, just use the default origin of X:0, Y:0, so click OK

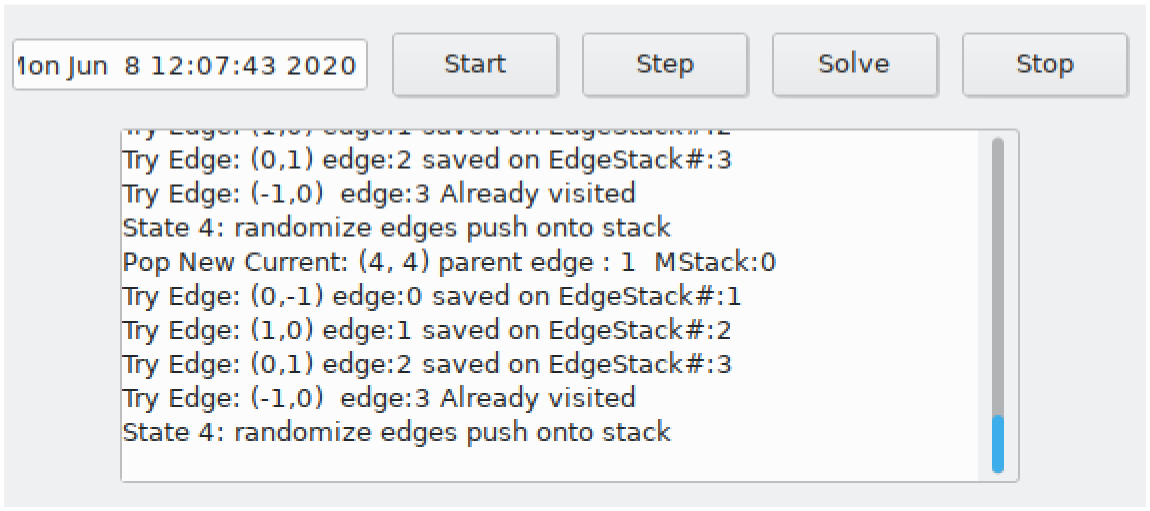


Create Maze from graph Origin

X: 0
Y: 0

Illustration 13: Start Button Dialog - Origin coordinates

The Pink squares show the cells being examined as neighbors. The algorithm can be single stepped by pressing the ‘Step’ button. That way the exploration process can be viewed and debugged in the log screen.
When ‘Step’ is pressed, the debugging log shows the sequence of exploration using relative coordinates.

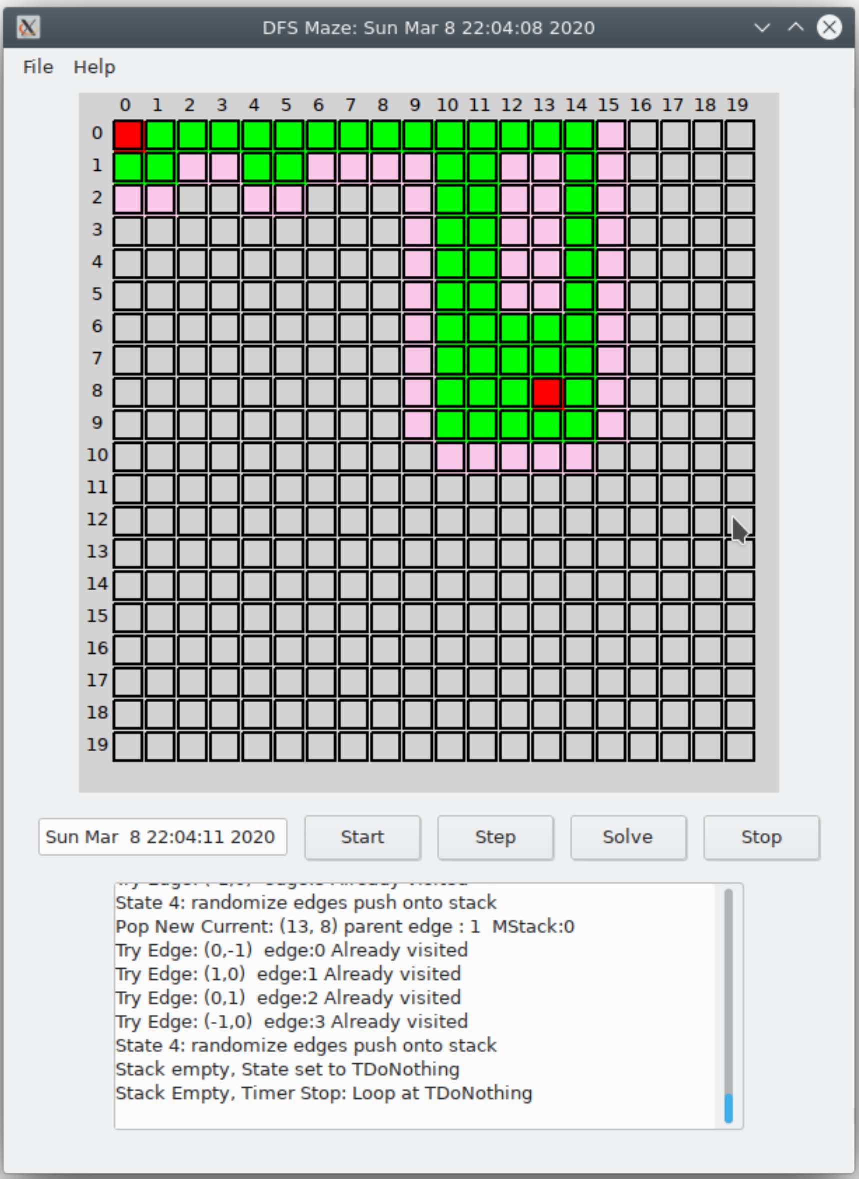


```
Try Edge: (0,1) edge:2 saved on EdgeStack#:3
Try Edge: (-1,0)  edge:3 Already visited
State 4: randomize edges push onto stack
Pop New Current: (4, 4) parent edge : 1  MStack:0
Try Edge: (0,-1) edge:0 saved on EdgeStack#:1
Try Edge: (1,0) edge:1 saved on EdgeStack#:2
Try Edge: (0,1) edge:2 saved on EdgeStack#:3
Try Edge: (-1,0)  edge:3 Already visited
State 4: randomize edges push onto stack
```

Illustration 14: Log showing expansion sequence

Pressing ‘Start’ again will ignore the 0,0 coordinates and continue the automatic run of the algorithm from where it is at.

4. Submit screen shot of this version configured just like Illustration 15. Note the Cell walls are not erased yet and the application stops when there are no unvisited neighbors. (See the log output) Yours will be different!



```
State 4: randomize edges push onto stack
Pop New Current: (13, 8) parent edge : 1  MStack:0
Try Edge: (0,-1)  edge:0 Already visited
Try Edge: (1,0)  edge:1 Already visited
Try Edge: (0,1)  edge:2 Already visited
Try Edge: (-1,0)  edge:3 Already visited
State 4: randomize edges push onto stack
Stack empty, State set to TDoNothing
Stack Empty, Timer Stop: Loop at TDoNothing
```

Illu
stration 15: Example assignment submission screen shot