Project 2

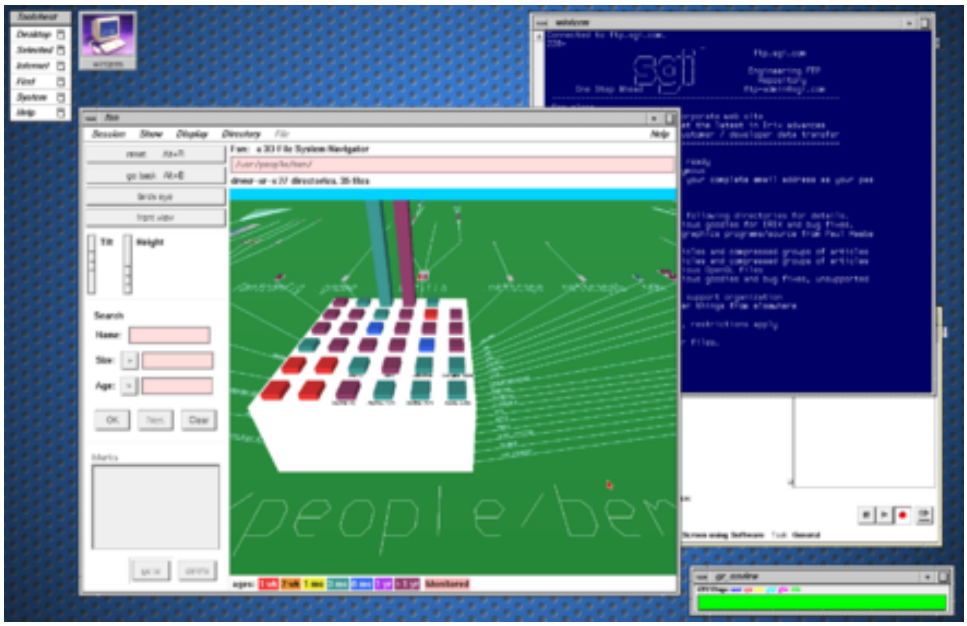
File Manager using Electron

The Premise

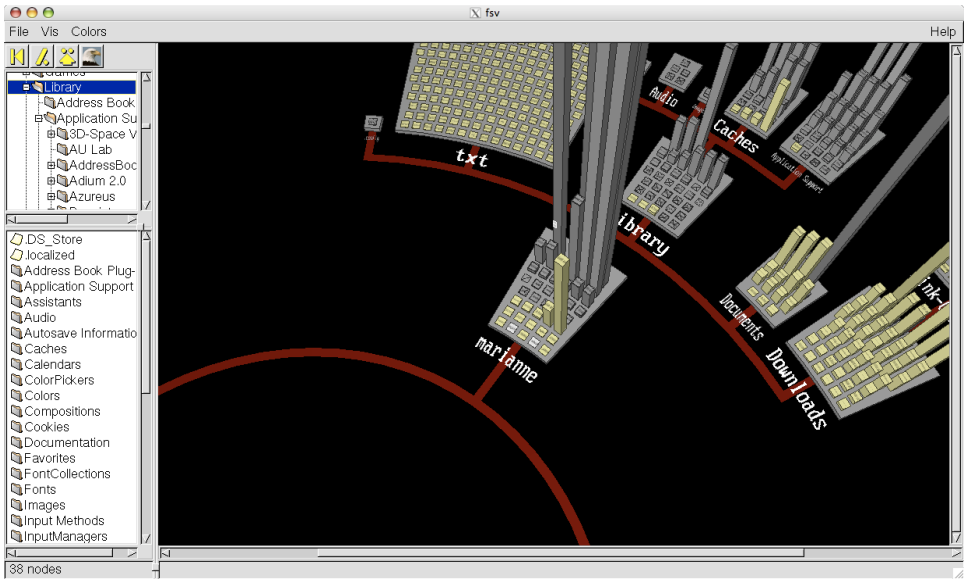
For the second project, you will be writing a file manager (sometimes called a file browser) using Electron. You should be very familiar with how a file manager works. As a modern computer user, you have likely used file managers such as macOS Finder, Windows File Explorer, Nautilus in Ubuntu, and many others.

In most modern file managers, the application acts as a graphical user interface shell. These applications extend the idea of the “desktop metaphor,” in which a program represents directories as folders holding files (like on a desktop or in a filing cabinet) and each of the files and directories as “physical” objects that the user can rearrange in some space. Most file managers are also navigational, meaning that the application user can move from location to location, between directories.

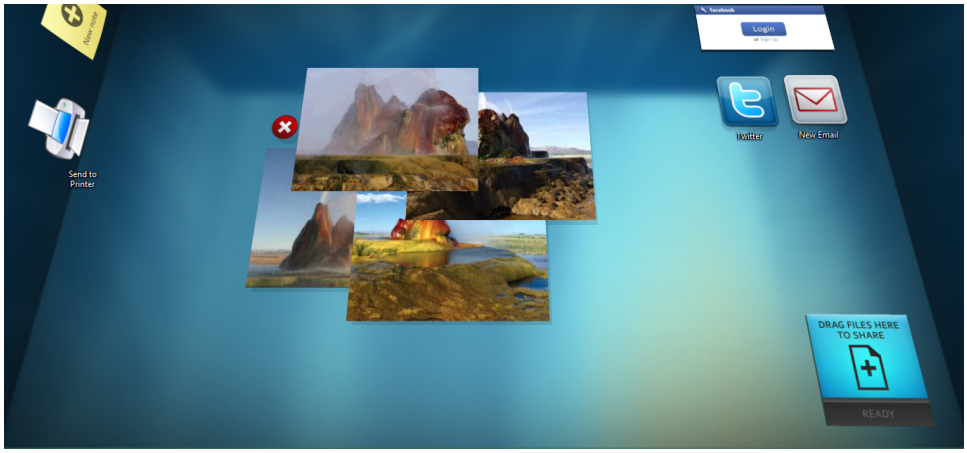
For your file manager, you will need to think outside of the simple file/folder metaphor. You will need to create a two-dimensional or three-dimensional file manager that will display files and directory structures on a system in a meaningful way. Your application will need to be able to navigate through directories on the system, add and remove new directories and files, get information about specific files, and give the user a sense of your chosen metaphor for the file/directory system. The images below show some examples of applications for inspiration:



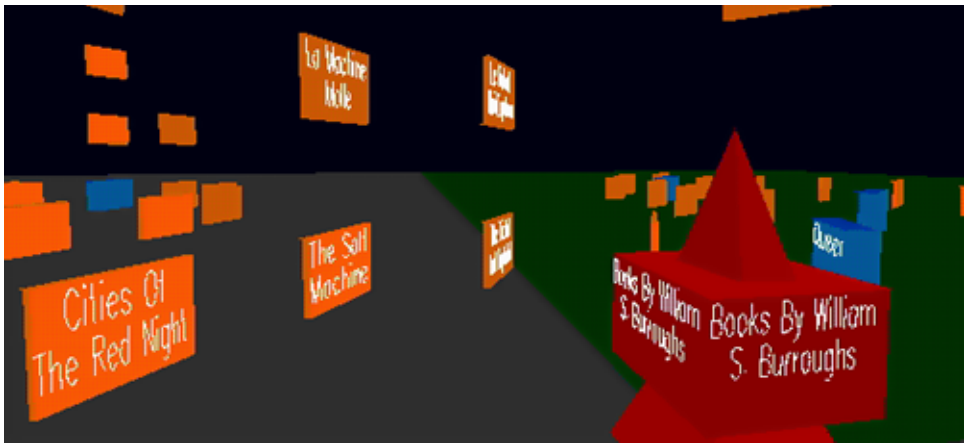
File System Navigator (by Silicon Graphics)



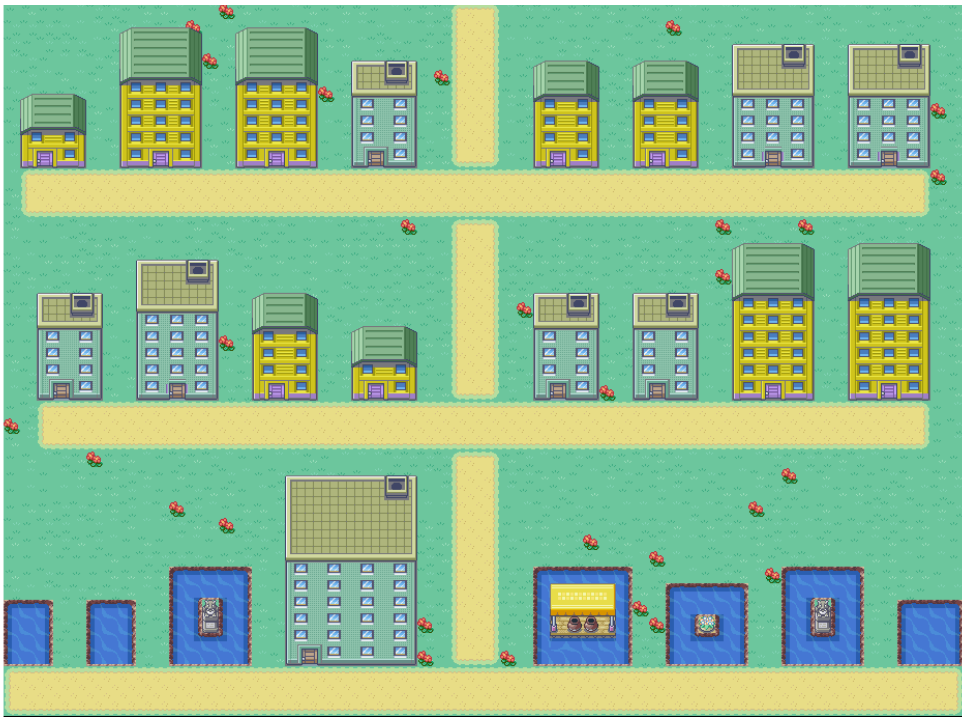
File System Visualizer (by Daniel Richard G.)



BumpTop (by Anand Agarawala/Bump Technologies Inc)



GopherVR (by University of Minnesota/Cameron Kaiser)



PokeFile (by Dr. Oropallo)

You will be using the Electron framework and Node to accomplish this project. Electron is a framework for creating native applications with web technologies like JavaScript, HTML, and CSS to combine Chromium and Node.js into a single runtime, allowing developers access to both the Node and Chromium APIs. You can package your application for Mac, Windows, and Linux. By using Electron, you will have access to any front-end view libraries, the HTML canvas, WebGL, as well as most of the Node standard APIs. Some popular apps built using Electron include Atom, Skype, Visual Studio Code, Discord, and Slack.

Required Features

View Files and Directories for Any given Directory

Given an arbitrary path, your application should be able to see the files and directories in that path. Similar to how the “ls” and “dir" commands work in the command line.

Navigate to Directories from the Current Working Directory

The user of your application should be able to navigate to different directories from the current directory they are viewing. Navigation should not rely on the user knowing the structure of the file system (the user does not need to know the exact paths of files/directories). The user should be able to move out of the directory they are viewing to the parent directory, or into the sub-directories inside that directory.

Display Files and Directories in an Aesthetically Interesting Way and Differentiates Between Them Visually

Similar to the examples in the above images, your application should show the user files and directories visually and uniquely. The execution is up to you, but you should aim to go above and beyond the typical “file and folder” view seen in standard file managers. Files and directories should also be visually distinct, and users should be able to discriminate between them quickly.

Use Either WebGL, Canvas, or a Library That Utilizes One of Those Technologies to Display Files and Directories

Your application should employ the use of either a Canvas tag, WebGL, or a library that uses those technologies to show users the files and directories in the current working directory. Good libraries to look into are:

- https://threejs.org
- https://phaser.io
- http://www.pixijs.com
- https://createjs.com/easeljs
- https://d3js.org (Not strictly for use with canvas, but can be adapted to work with it)
- http://paperjs.org

You do not have to use one of those libraries, and other libraries are fine to use as well as long as they use WebGL or Canvas.

Create Empty Files

The application should be able to create empty files in the current working directory.

Remove Files

The application should be able to remove files in the current working directory.

Create Empty Directories

The application should be able to create empty directories in the current working directory.

Remove Directories

The application should be able to remove directories in the current working directory.

Copy Files Anywhere

The applications should be able to copy files from the current directory to anywhere else. Copying does not include directories, only files.

Move Files Anywhere

The applications should be able to move files from the current directory to anywhere else. Moving does not include directories, only files.

Information About Files

The application should be able to present the user with information about the files in the current working directory. This information includes, but is not limited to file size, creation time, modification time, and absolute path on the file system.

Intuitive User Interface

The user interface of this application should not just be functional, but also intuitive to use. The user should not need a 4000 level course on how to navigate through your application.

Additional Feature

Aside from the features listed above, your application should have a unique feature. This feature can be something special about your user interface, a unique function, or something else. You should ask the instructor about your Additional Feature before attempting to implement if there are any concerns.

Project Documentation

For this project, you will also be submitting an instruction manual (PDF please) that documents all of the functions of the application. The instructions should have sections that include all of the above-required features, as well as any other relevant information. Please feel free to use images to help illustrate your text. Make sure your document is professional and well organized, with a cover page and table of contents.

Rubric (300 Points)

Project Organization (45 Points)

Criteria	0 to 6 Points	7 to 14 Points	15 Points
Is the code well documented?	The project has little or no documentation.	The project has some documentation.	The project is well documented.
Is there ReadMe file in the project directory explaining how to build and run the project?	The project did not contain a ReadMe or it was not helpful.	The project did contain a ReadMe or it was mostly helpful.	The project did contain a ReadMe or it was helpful.
Does the project separated style, view, code, and asset files where appropriate?	The style sheets, view files, javascript files, and assets are not well organized.	The style sheets, view files, javascript files, and assets are not somewhat organized.	The style sheets, view files, javascript files, and assets are organized.

Project Documentation (30 Points)

Criteria	0 to 4 Points	5 to 9 Points	10 Points
Is the document professional and well organized?	The document is unorganized and unprofessional.	The document is unorganized or poorly presented.	The document is organized and well presented.
Does the document explain the content well?	The document does not explain the content.	The document explains the content but is unclear in its meaning.	The document does explain all the content.
Did the document detail all of the application functions?	The functions are not present or lacking.	There are some functions but not many.	The document details the all functions well.

Program Features (225 Points)

Notice: If the project is not an Electron app, all of the points below will be forfeit.

Criteria	Points
Application can view files and directories for any given directory	30
Application can navigate to different directories from the current working directory without the user knowing the exact path	25
Application displays files and directories in an aesthetically interesting way and differentiates between them visually	20
Application uses either WebGL, Canvas, or a library that utilize one of those technologies to display files and directories	30

Criteria	Points
Application can create empty files	10
Application can remove files	10
Application can create empty directories	10
Application can remove directories	10
Application can copy files anywhere (directories not required)	10
Application can move files anywhere (directories not required)	10
Application gives information about files (size, creation time, etc)	15
Application user interface is intuitive to use	15
The “Additional Feature”	30