Project 1

Due Jan 25 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip and rar
Available Jan 9 at 9am - Jan 29 at 6am 20 days

This assignment was locked Jan 29 at 6am.

See a demo of this project here: **Project 1 Overview (HD) (https://video.wpi.edu/Watch/Pd94XnBf)** | **Project 1 Overview (SD) (https://video.wpi.edu/Watch/Mj86Jsy9)**

Overview:

The aim of this project is to get you comfortable with working in 2D using WebGL. You will be setting viewports, drawing polylines, and exploring keyboard and mouse interaction routines. You will create a program called *Polibook* which views polyline files in interesting ways and also allows you to interactively draw your own polylines onto the screen.

This program will have two modes. *File* mode will allow a user to upload a *.dat file (more about this later) that draws a series of polylines to the screen. *Draw* mode will allow a user to construct polylines freely.

Part I: File Mode

For this part, the user will upload a file with a .dat extension. The file will consist of lines of numbers formatted as follows:

- The file may or may not begin with one or more rows of text followed by a line starting with at least one asterisk. All lines up to and including the asterisk line should be ignored.
- The first line after the asterisk line will consist of four numbers that define the "extent" of the figure. The numbers will be in this order: left, top, right, bottom.
- The next line will be the number of polylines in the figure.
- The rest of the file will be the list of polylines. Each polyline starts with a line that indicates the number of points in the polyline. Subsequent lines list the (x, y) pairs for each point.

Here are some example *.dat files: **cs4731_pjt1_dat_files.zip**

For a point of comparison, here are the images that each of the example *.dat files generates: **Project 1 Images**

The picture should appear as soon as the user uploads a file. When the user uploads a new file, it replaces the old one.

Implementation notes:

- Be careful with how you pass parameters to the `gl.viewport( )`, `ortho( )`, and similar commands. Make sure you understand how they work!
- The "extent" line of the file will be used to set the world window.
- If you look at the vertex coordinates of some of the files, they are specified in floating point numbers, so please be sure you use the correct format for your variables.
- How you choose to ignore all lines up to and including the asterisk line is up to you.

- The format for dino.dat is a little different in that it doesn't have the window dimensions (or comments) right at the top. Therefore, off the bat, a program which reads other files without problems will have new problems with dino.dat file. You can either use a default extents window or come up with another solution that also works. Hint: a world window of (0,640,0,480) should work.
- The default drawing color is black.
- The background color should be white.
- The user does not need to be able to edit any of the files.
- All images should maintain their aspect ratio.

Part II: Draw Mode

When the user enters draw mode, your program clears the screen and presents a blank drawing area. On the first click in the drawing area, a dot is drawn at wherever the user clicked. On subsequent clicks, a line is drawn from the last accepted mouse click position to the current mouse click position. If the "b" key is held down while clicking, the current click point is NOT joined to the previous click point and instead a fresh polyline is started and drawn in addition to whatever previous polyline had been drawn.

Implementation notes:

- Your program should be set up to accept up to 100 possible mouse clicks in one polyline.
- You can accept and store user-provided points in an array.
- The default drawing color is black.
- The background color should be white.

Additional Requirements:

- The user should press "f" to enter File mode and "d" to enter draw mode.
- If the user presses "c", the color of the drawing on the screen cycles between black, red, green, and blue. For instance, if the current drawing color is black, hitting "c" redraws everything in red. Hitting "c" again redraws it in green, etc.
- Your canvas must always be large enough to be visible to the viewer but not so large that it extends beyond the edges of the browser window on a typical laptop display.
- You have some discretion over the interface design. However, it should be obvious to the user whether he or she is in draw or file mode. In addition, since this is not an HCI class, please keep the interface simple and legible.
- You should not use any deprecated commands.
- As this is a senior-level CS class, I expect your code to look clean and professional. This means following good coding practices such as proper indentation, basic refactoring, descriptive variable and function names, no large chunks of code that are commented out, etc. Make it something that you would be proud to post on GitHub.
- In the spirit of the previous point, please comment your code appropriately. Commenting helps us better understand what you're doing and why. There are no strict conventions that you are required to follow, but I recommend, at minimum, preceding each JavaScript function with an explanatory comment.
- You will need to include a documentation file (see "Submitting Your Work" below).

Extra Credit:

For this project, you are allowed to include additional features above and beyond the basic requirements. The instructor has sole discretion over the point value of each feature, and the student may earn up to 10 points total. To earn any credit, *you must list each feature in the comments at the top of your main *.js file.*

To be eligible for extra credit, the additional features must be graphical in nature. This means that you show thought behind how something is presented to the user on screen or how the graphics are being processed behind the scenes.

Here are some example features to consider:

- Allow the user to manually choose a color from a broad pallette (similar to a paint program). Hitting "c" reverts back to the red/green/blue/black cycle.
- Allow the user to upload a *.dat file in Draw mode and then modify the drawing accordingly.
- When the user presses the "t" key, display a tiling of the drawing in the canvas.

You are not limited to these ideas. If there is something else you would like to do but are not sure if it would qualify for extra credit, please contact me.

Submitting Your Work:

Make sure to double-check that everything works before submitting. Put all of your files (JavaScript, HTML, etc.) into a folder and zip it. You do NOT need to include the *.dat files. Upload it to Canvas. Do not email me your program or submit it via a third-party cloud storage account.

Create documentation for your program and submit it along with the project inside the zip file. Your documentation can be either a pure ASCII text or Microsoft Word file. The documentation does not have to be long. Briefly describe the structure of your program and what each file turned in contains. Name your zip file according to the convention *LastnameFirstname_Pjt1.zip*.

Additional Notes:

- You are free to consult any resources you need to help you complete this assignment, including your classmates, the TA, the instructor, the class text, and any other books and websites. However, the code you turn in must be your own. ***Any evidence of plagiarism will result in an automatic 0 for this assignment.***
- You are welcome to use any class coding examples (posted under "Modules") in your program.

FAQ:

Q: In the assignment, it says "Your program should be set up to accept up to 100 possible mouse clicks in one polyline." What is the expected behavior of the program on the 101th click?

A: The program should automatically begin a new polyline at that point (the same behavior as if the user had held down "b").

Q: There are two files that don't quite fit the spec. In the 'dino.dat' example, there is no extent line. In the 'scene.dat' example, every other line is blank. How do we handle these?

A: As discussed in the instructions, if the file does not have an extent, you should use a default one. The instructions recommend a good one to use.

Please accommodate any blank lines. This just involves iterating through the lines and ignoring all of the blank ones.

Q: If we are switching from Drawing to File Mode (or vice versa), should the existing drawing be cleared and a brand new File Mode canvas made ready to use? Or should the drawing be "stashed" and replaced with the File image last uploaded (and switching back will return to the user's in-progress drawing)?

A: When switching between canvases, a brand-new canvas should be ready for use. Don't worry about stashing the previous image.

Q: My drawing looks correct, except that it's flipped and/or upside down.

A: Double-check the order of your extents. Remember that the extent order for the `ortho()` function is different than the extent order defined in the *.dat files.

Q: I noticed in your demo video that when you start a new line in Draw mode, you can't actually see the first point. Is this okay?

Yes, just so long as when you draw your second point, the first and second points are connected as discussed in the instructions.

Project 1 Rubric

Criteria	Ratings			Pts
File Parsing (3 pts each) - Program has a place to upload *.dat files - Program parses *.dat files - Ignores lines up to and including asterisk - Handles discrepancies in some files (newlines, whitespace, etc.) - Parses out, converts, and stores numbers correctly	**15.0 pts** **Full Marks**		**0.0 pts** **No Marks**	15.0 pts
File Drawing (5 pts each) - Entire picture drawn to screen immediately - Aspect ratio preserved - Projection matrix set based on extent settings - Enough contrast in colors between foreground and background for the image to be visible	**20.0 pts** **Full Marks**		**0.0 pts** **No Marks**	20.0 pts
Draw Mode - First click draws a dot (8 pts) - Subsequent clicks draw a line by connecting a line segment to the previous point (10 pts) - Holding down "b" starts a new line (7 pts) No points awarded if there is not enough contrast between the foreground and background for the drawing to be visible	**25.0 pts** **Full Marks**		**0.0 pts** **No Marks**	25.0 pts
Mode Switching (5 pts each) - Can switch between file and draw modes using correct keys - Switching clears the canvas	**10.0 pts** **Full Marks**		**0.0 pts** **No Marks**	10.0 pts
Color Cycle Pressing "c" rotates the color between black, red, green, blue (-2 for each missing color)	**10.0 pts** **Full Marks**		**0.0 pts** **No Marks**	10.0 pts
Documentation and Organization (10 pts each) - Code is clean, professional, well-commented, and easy to read - Documentation file describing structure of program and what it contains	**20.0 pts** **Full Marks**	**10.0 pts** **Documentation file missing**	**0.0 pts** **No Marks**	20.0 pts
Extra Credit	**0.0 pts** **Full Marks**		**0.0 pts** **No Marks**	0.0 pts
				Total Points: 100.0