Dan Scarafoni

Write-up project 2

This project aimed to take a an unreadable (to humans) data structure and convert it to a file format that could be understood by humans. The program would also be able to take a properly formatted text file and convert it into a data structure. This project used much of the code given by the professor. Much alteration was made, however, to the struct_to_file and file_to_struct file.

The structs (face, vert, edge) in my program are very similar to the ones already made, with a few streamlining features. The printable vertex struct was eliminated (because it was redundant), and all other printable version structs were changed to accept a serial number for pointer values instead of the index of an array. This was done to increase flexibility and avoid errors. The arrays are temporary, used only as intermediate entities to hold data before it is located into it's final form. Using the serial number allows multiple configurations of the same object to be accepted as the same structure.

For example, if array indecies were used as pointers, one would not be able to reverse the order of the faces in the read-in file, the program would read and enter the wrong faces in the wrong indecies, and the program would crash. With my program, the order in which the structs are entered into the array is irrelevant, the program can compensate.

The struct2file converts a struct to a file using a greedy algorithm by going through several nested layers of structure simultaneously, thus removing much redundancy. Whenever the program encounters a face, it enters it. When it encounters an edge, it enters it, and when it encounters a vertex, it enters it. There is no need to double back and go through a list again, as this program can access and store all valuable structs in one pass of the loop of faces.

The file2struct file works in two parts. In the first part, it retrieves all the information from the file, and writes it into several arrays of printable structs. Using this information, it recreates the object in memory (using the file_link_structs function) in the same style as the tetra.c file, but without the

need to create vertecies or put them in a separate array(which was already done in the previous step). Starting with a printable face, it creates a separate non-printable face; it then creates the edges for this face using the next_edge serial pointer. Once all the faces are made, they are linked together in a loop, and the front face is returned. This also implements a greedy algorithm similar to the one used in file2structs, the program makes the structures is needs as it needs them, thus preventing the need to double back in order to fill in gaps in information.