Week 5 Milestone 4 List of Enhancements -

Joel Garcia

SNHU CS-499

Reset Random Integers

Clears the vectors containing random numbers for generating random object color and position, then generates new random numbers. This allows new object location data to be generated every time the drone object loops through the map.

```
479             // empties random integer vectors
480             vect.clear();
481             zAxis.clear();
482
483             // generate new random integers for random color generation
484             for (unsigned int i = 0; i < (numObjects); i++) {
485                 vect.push_back(random());
486             }
487
488             // generate new random integers for random position data
489             for (unsigned int i = 0; i < (numObjects); i++) {
490                 zAxis.push_back(randomZ());
491             }
```

Write CSV File Function

Writes position coordinates to a csv file so that it can be used for the heatmap.

```
640  // write  object locations to a csv file for heatmap use
641  void WriteFile() {
642      // file pointer
643      fstream fout;
644
645      // opens an existing csv file or creates a new file.
646      fout.open("mapsize.csv", ios::out | ios::app);
647
648      // Insert the data to file
649      fout << (int)dronePosX << ", "
650          << (int)dronePosZ << "\n";
651  }
```

Write File Call

Write file is called in the color reader function. Color reader checks pixel color for flagged objects, and upon finding flagged objects outputs location to console and write location to csv file. The csv file allows the dataset to be passed to python so that we can use python data visualization tools to generate a heatmap.

```
696     // Function to read pixel color and output coordinates of flagged objects
697     void ReadColor(int x, int y) {
698
699         unsigned char pixel[4];
700         glReadPixels(x, y, 1, 1, GL_RGB, GL_UNSIGNED_BYTE, pixel);
701
702         // Prints coordinates of objects with greater Red color value
703         if ((int)pixel[0] > (int)pixel[1] && (int)pixel[0] > (int)pixel[2]) {
704             cout << "Flagged object at position: " << (int)dronePosX << ", " << (int)dronePosZ << endl;
705             WriteFile(); // write flagged object locations to a csv file
706         }
707     }
```

Clear CSV File Function

This function erases the csv file so we can create a new dataset every time the program is run.

```
653     // used to erase csv file
654     void ClearFile() {
655         ofstream file("mapsize.csv");
656         file << "";
657     }
```

Clear File Call

File is cleared at the start of program. This allows us to inspect the dataset in the csv file even after the program is closed.

```
297     int main(int argc, char* argv[])
298     {
299         if (!UInitialize(argc, argv, &gWindow))
300             return EXIT_FAILURE;
301
302         ClearFile(); // erase csv data - allows for a new dataset on each program use
303
```

Heatmap Function

Heatmap is generated in python using PyRun_SimpleString. After importing modules, we initialize a list and the read the csv file and append each line to the list. The list is then converted to a dataframe and printed to the console to compare with previously output object location. Finally, the dataframe is formatted and the heatmap generated and displayed.

```
618  // use python to generate a heatmap of flagged object position occurrences
619  void heatmap()
620  {
621      Py_Initialize();
622      PyRun_SimpleString("import matplotlib.pyplot as plt\n"
623          "import seaborn as sns\n"
624          "import pandas as pd\n"
625          "import csv\n"
626          "data = []\n"
627          "with open('mapsize.csv') as file:\n"
628          "    reader = csv.reader(file, delimiter=',')\n"
629          "    for line in reader:\n"
630          "        data.append(line)\n"
631          "df = pd.DataFrame(data,columns=['X','Y'])\n"
632          "print(df)\n"
633          "df2 = pd.crosstab(df['X'], df['Y'])\n"
634          "sns.heatmap(df2, annot=True)\n"
635          "plt.show()\n"
636      );
637      Py_Finalize();
638  }
```

Heatmap Display

The heatmap is called and displayed after ending the drone window but before terminating the program as a whole. This allows us to generate a dataset of any size and display before closing the program.

```
505      heatmap(); // displays heatmap after program end but before termination
506
507      // Release object mesh data
508      meshes.DestroyMeshes();
509
510      // Release shader program
511      UDestroyShaderProgram(gCubeProgramId);
512      UDestroyShaderProgram(gLampProgramId);
513
514      //this_thread::sleep_for(chrono::seconds(100)); // for debugging
515
516      exit(EXIT_SUCCESS); // Terminates the program successfully
517  }
```