FSAE Tire Temperature Visualization

ECE3220 Final Project

Spring 2017

Max Houck

Marshall Lindsay

Abstract

The goal of this project was to aid in the interpretation of key tire temperature data during the design of Mizzou Racing's Formula SAE car. This data is used extensively during the suspension design and gives the engineers qualitative information to make the correct decisions. The current problem lies in the ability to interpret this data in its raw form. Thousands of data points, some completely invalid, are almost impossible to make sense of without any form data processing. Discussed here is our solution to this problem. By using signal and data processing, combined with an intuitive graphical interface, we've created a software package that not only allows engineers to interpret this raw data, but leaves room to be extended to real time telemetry.

Introduction

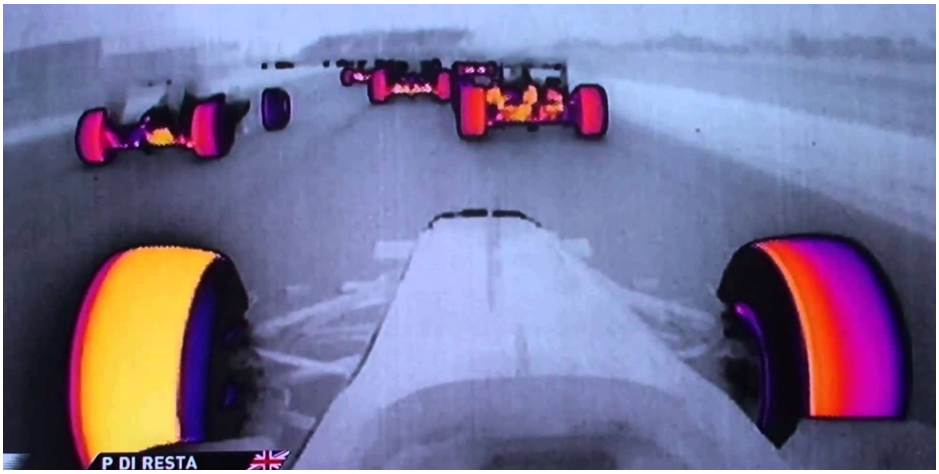
Formula SAE is a collegiate engineering team where students design and manufacture a quarter scale formula-style racecar. This racecar is built to compete in both static and dynamic events against other collegiate teams across the globe. During the design process, engineers rely on data acquisition to validate and optimize critical systems to maximize safety and performance. A portion of this data includes tire temperatures.

Tire temperature data is used extensively during the design and optimization of the racecars suspension system. Proper tire data allows engineers to ensure there is optimal energy transfer from the car to the ground. The Mizzou Formula SAE car is outfitted with twelve infrared tire temperature sensors (three per wheel). The data from these sensors is read by four AVR microcontrollers, via I2C serial communication, mounted at every corner of the car. This data is sent over the racecar's CANBUS to an onboard datalogger that outputs the data to an SD card. After a test drive, this data is used to validate suspension design. Currently the data is viewed in its raw form; thousands of numbers representing degrees Celsius at different points in time. In this form, the data is nearly impossible to interpret. The aim of this project was to process and display this data in an easy-to-interpret heat map allowing engineers to make proper optimization and design decisions.

Mizzou Racing's 2016 Competition Car

Background

Tire temperature data is an important metric for Formula One teams around the world. This data shows how efficiently the tread of the tire is being used to transfer energy from the car to the ground. Ideally, equal temperatures should be maintained across the entire face of the tire. Hot or cold spots represent poor suspension design or tire defects. There are currently high-end racecar data analysis software packages for sale, however these are extremely expensive and often not customizable. By developing our own software packages for visualizing tire temperature data, we can meet the current needs of the team while simultaneously creating a robust foundation to expand upon in the future (several more data channels could be added for visualization purposes).



Formula One Tire Temperature Cameras

Implementation

The program is broken down into three major parts: 1) the processing and storage of data from files, 2) reading data from serial bus and 3) the graphics. The relationship between the three parts is shown by the Class and Use Case diagrams below.

Logged data was read in via text files whose contents consisted of three columns of numbers. Each column represents a temperature as it was recorded by the datalogger. A screenshot of this data is shown below where the leftmost column corresponds to the outer sensor and the rightmost the inner sensor. Four files are required to run the program, one file for each of the tires.

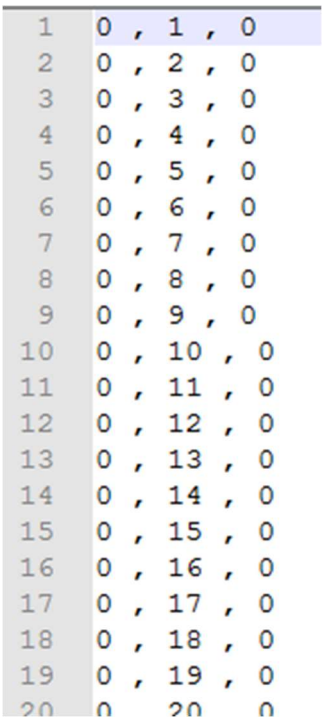
Each data point is added to the corresponding Sensor class temperature vector. There, the data is stored until it is used for graphical representation. The class hierarchy used creates a logical organization of data making access and modification simple and effective.

During the creation of the graphical interface, we opted to use filled rectangles to represent the four different tires on the car. These rectangles would be filled with a temperature gradient resembling an infrared heat map. Extra care was taken to insure the graphical interface would be extendable to a wide range of applications in the future. Customizability was also a major consideration, leading to a graphics interface that can be adjusted during run time.

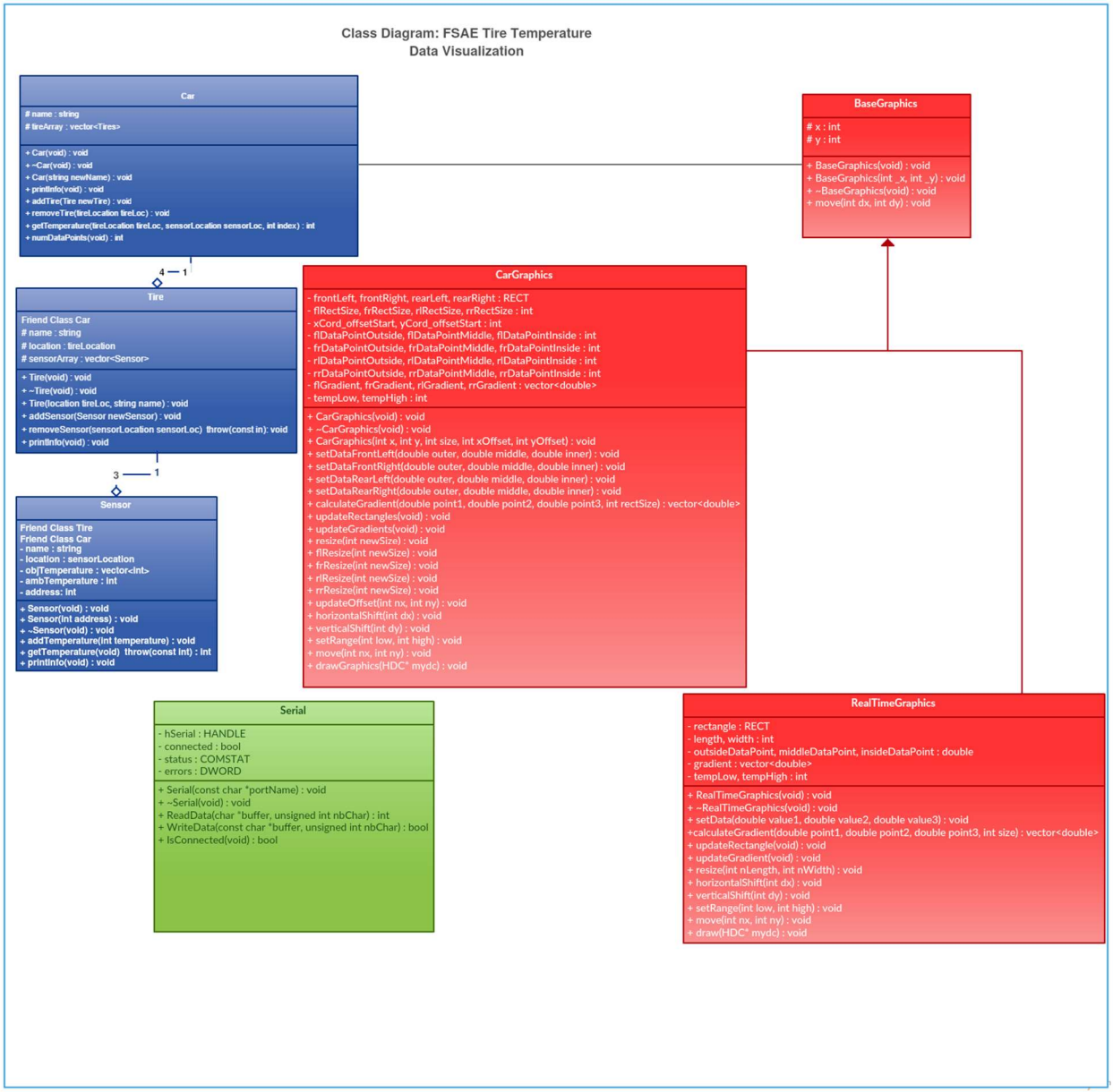
A text-based menu system was created to allow the user to interface with the various systems and controls. These menus allow the user to choose between real time or logged data representation and to change the location and size of the various graphical entities. Exception handling can be found throughout the program keeping the program safe from erroneous user input.

The real-time interface was constructed as a basis for a more advanced telemetry system that may be implemented in the future. This system would need to consist of a fully wireless interface to interact with the car as it drives. Because this exceeds the scope of this course we opted to design the foundation to be added upon later. This foundation consists of reading data from the sensors in real time using an Arduino and then sending that data to our workstation via serial communication. This data is read, processed, and sent to the graphical interface.

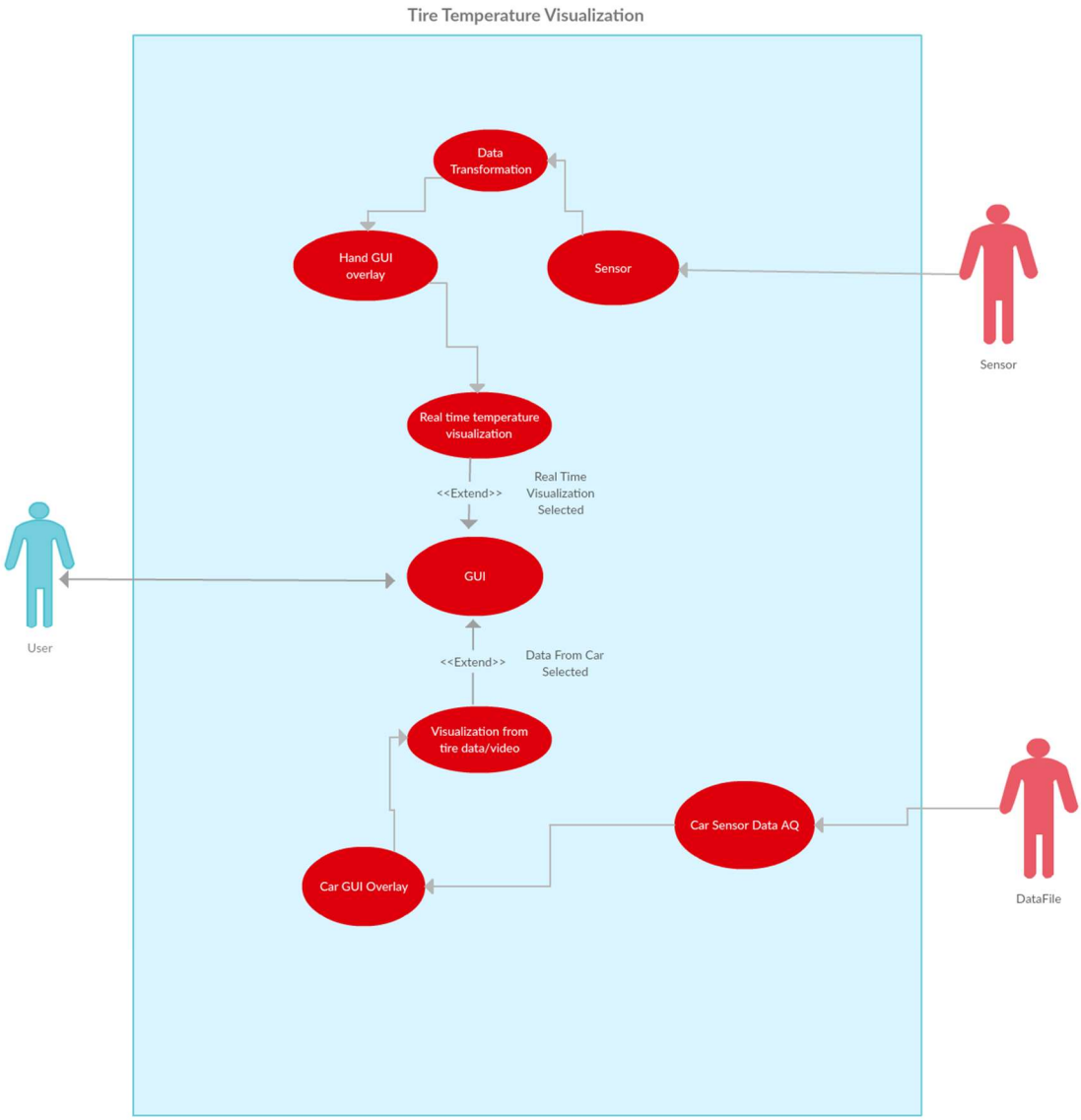
When accessing the data in real time via the serial bus, data is not stored (except for a small buffer used for signal validation and processing) but sent directly to the graphical interface. This choice to bypass all storage leads to an almost perfect real time graphical representation. The choice to read data in real time comes with some problems that have not been fully addressed. These problems will be discussed in detail during the Experiments and Results section of this report.



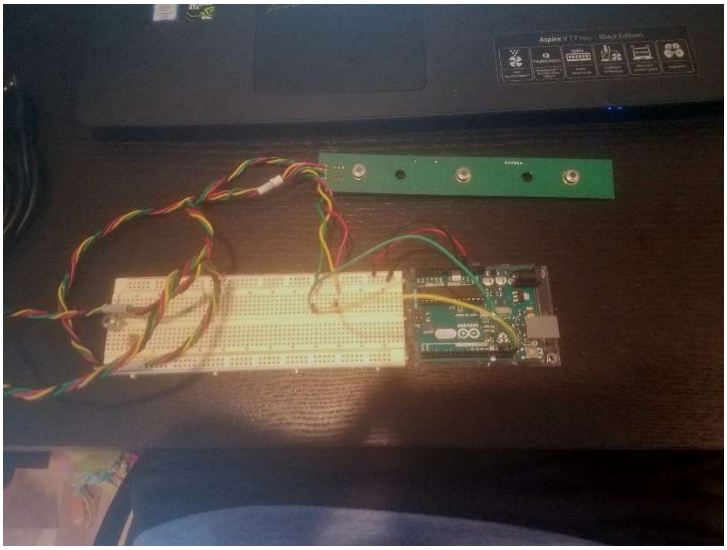
Screenshot showing data from file



UML Class Diagram



UML Use Case Diagram



Picture of Arduino and sensor setup

```
Graphics Setup Options:
1)Print
2)Move
3)Vertical Shift
4)Horizontal Shift
5)Resize Every Tire
6)Resize Front Left Tire
7)Resize Front Right Tire
8)Resize Rear Left Tire
9)Resize Rear Right Tire
10)Increase Offset From MidPoint
11)Set temperature range
12)Finish
```

Screenshot showing one of the option menus found in the program

Experiments and Results

This project was consistently put through various tests during the design process. Several of these tests involved the graphical representation of data. The starting point for the graphics was a simple filled rectangle shown below. Initially this was done by setting the colors of individual pixels within the confined area. This proved to take too much time (around one second for a 100x100 pixel box) and would not allow for high speed data visualization. After some research, another function was found in windows.h, FillRect. This function took a pointer to the console handle, a pointer to a RECT object, and a brush set to the desired RGB value. When given a 100x100 pixel rectangle to fill, this function finished instantaneously.

While the use of static colors is useful, it was not what we envisioned for the project. We wanted to see a dynamic heat map across the tire. Naturally the next step was to create a color gradient. This posed a problem given our current knowledge and tools for creating graphics. The function we were currently using took rectangle with fixed dimensions and filled it with a fixed color. We needed to develop a way to create a seemingly dynamic graphics package with completely

static tools. This was achieved by breaking down the rectangle we wanted to fill into columns of pixels. Then, by filling each column of pixels with a slightly different color, we could achieve a "dynamic" color gradient.

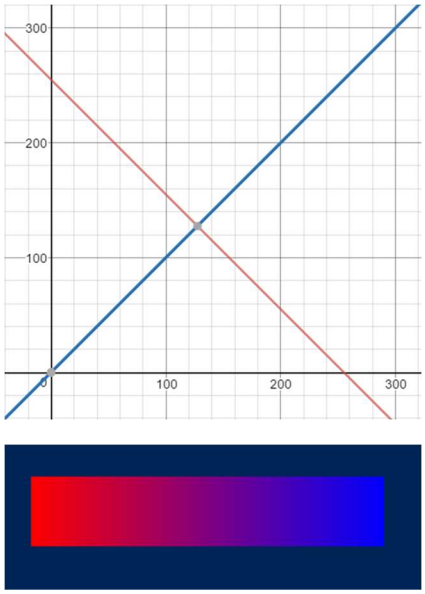
Three different gradient production algorithms were created and tested to achieve the intuitive and pleasing heat map representation of data. These included a 510-linear interpolation, a 1020-linear interpolation, and a 510-quadradic interpolation. After several experiments, the 510-linear interpolation was picked to create the gradients inside the tire boxes. Each of these graphs and their outputs are shown below (Note: the 510-quadraic interpolation was not simulated due to problems with the way the program handled the squaring of numbers).

Always taking extendibility into account, the graphics were implemented with zero static qualities. Everything from the size of the boxes to the temperature scale within the boxes are customizable to fit the user's needs.

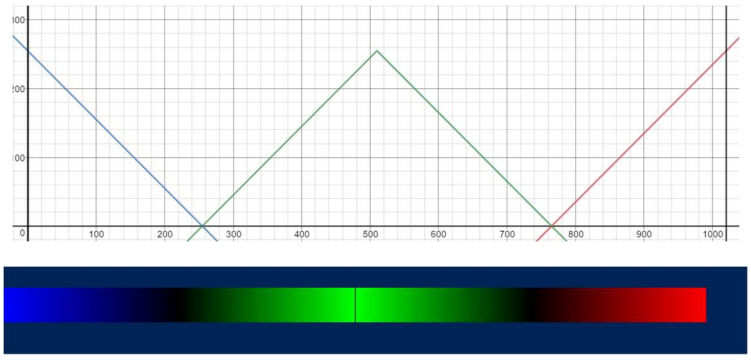
The program was tested with a wide array of possible user inputs to check for valid exception handling. Perfect exception handling is found throughout most the program. Exception handling was not implemented fully during the graphical adjustment menus purely due to time constraints. If erroneous input is entered during these menus, the program will fail.

Reading data via serial communication created several problems that we have yet to fully solve; the main problem being signal processing. Some of the data received over the serial bus is completely invalid and leads to a graphical representation that lacks validity. This 'garbage' data was dealt with through rudimentary filtering using a five-data-point residual buffer. This buffer contains the last five data points whose average is calculated. If a new data value doesn't correspond with the trend of the data, the data point is removed and replaced with the average. This filtering removes outliers in the data while maintaining data trends.

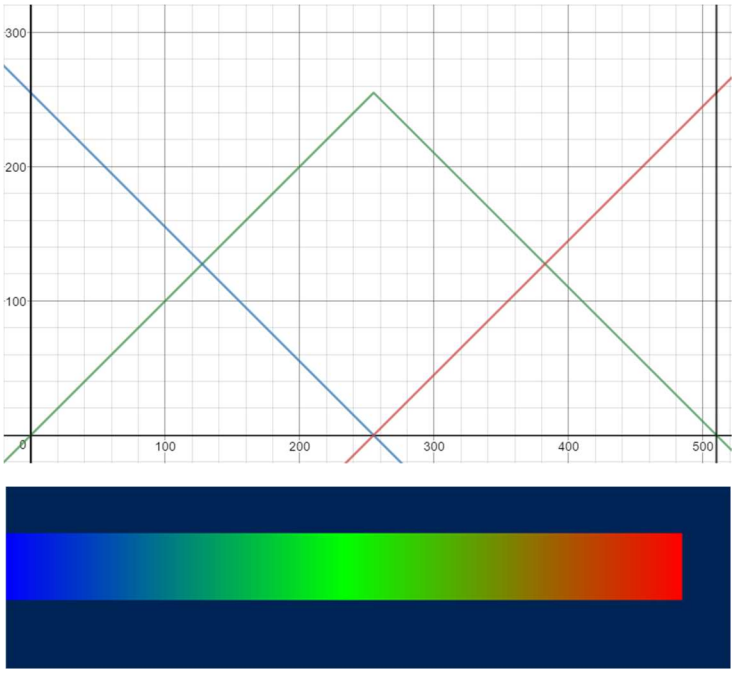
Filled rectangle representing the starting 'block' for our graphical representation



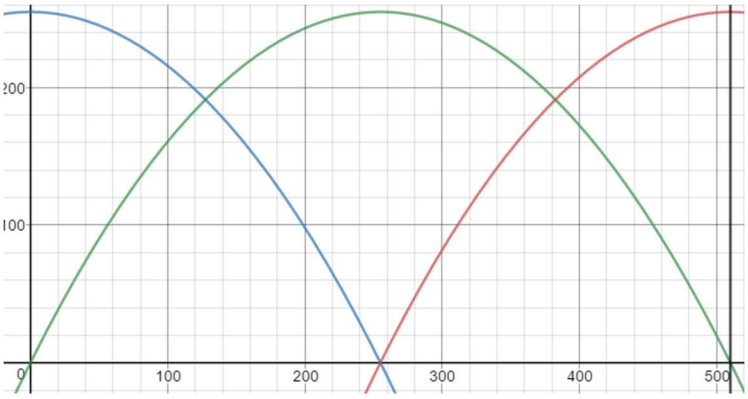
255-linear interpolation of two colors



1020-linear interpolation of three colors



510-linear interpolation of three colors



510-quadradic interpolation of three colors

Discussion and Conclusion

Throughout the design process, the layout and functionality of the program consistently changed. The UML diagrams helped to keep the project on track and our members on the same page. The importance of proper documentation, planning, and high level diagrams will only increase as we are exposed to bigger projects in the real world.

GitHub was used extensively throughout the development of the project. This was both of our group members first experience using GitHub as a collaboration tool and it proved to be a rewarding experience. The use of frequent commits lead to a workflow that kept both members working on up-to-date code and allowed us to roll back the code in the event a bug was introduced.

GitHub activity

This project pushed our knowledge of C++ programming to the limits. By the end, our project evolved from a simple idea into a solid foundation for future development by the Formula SAE team. Numerous features can be layered over our program such as reading data from a database or real time telemetry. We've created the starting block for future engineering students to expand and create.

Code:

```
/*
    Marshall Lindsay
    Max Houck
    Formula SAE Tire Temperature Visualization
    ECE 3220 Final Project
    main.cpp
*/
#include<windows.h>
#include "tires.h"
#include "SerialClass.h"
#include "CarGraphics.h"

#define LOGGING_RATE .1
#define ACCEPTABLE_PERCENTAGE .05

void programStart(void);
void mainOptionMenu(void);
void dataVisualization(void);
void dataVisualizationWelcomeMessage(void);
void simulation(Car* car, CarGraphics*);
void realTimeDemo(void);
Car* carSetup(void);
void graphicsSetup(CarGraphics*);
void graphicsSetupOptionMenu(void);
void realTimeGraphicsSetup(RealTimeGraphics*);
void realTimeGraphicsSetupOptionMenu(void);


int main() {
    try{
        programStart();
    }
    catch(...){
        cout<<"\nProgram Failed. Catch all Main()"<<endl;
        return(1);
    }
}

void programStart(void){
    string userInput;
    for(int i = 0; i < 60; i++){
        cout<<"#";
    }
    cout<<"\n\nFSAE TIRE TEMPERATURE DATA VISUALIZATION SOFTWARE\n"<<endl;
    for(int i = 0; i < 60; i++){
        cout<<"#";
    }
    cout<<"\n\n\nWelcome to version 1 of the FSAE Tire Temperature Visualization
    Software!"<<endl;
    while(1){
        mainOptionMenu();
        getline(cin, userInput);

        if(userInput == "1"){
            realTimeDemo();
        }else if(userInput == "2"){
            dataVisualization();
        }else if(userInput == "3"){
            return;
        }else{
            cout<<"\n Invalid input!"<<endl;
        }
    }
}

void mainOptionMenu(void){
    cout<<"\n\nPlease select from the following options: "
        <<"\n   1)Real time data sensor visualization"
        <<"\n   2)Data visualization from file and video"
```

```
69          <<"\n   3)Quit"<<endl;
70  }
71
72  void realTimeDemo(void){
73      string userInput;
74      HWND myconsole = GetConsoleWindow();
75      HDC mydc = GetDC(myconsole);
76      RealTimeGraphics* realTime = new RealTimeGraphics();
77      realTimeGraphicsSetup(realTime);
78      cout<<"\n Real Time graphics setup complete... Press any key to begin"<<endl;
79      getline(cin, userInput);
80      Serial* SP;
81
82      while(1){
83          try{
84              SP = new Serial("COM4"); //We may need to adjust this as necessary
85              break;
86          }
87          catch(const int x){
88              if(x == 1){
89                  cout<<"\n Please connect the arduino!"<<endl;
90                  Sleep(3000);
91              }else if(x == 2){
92                  cout<<"\n Unknown error!"<<endl;
93              }
94          }
95          catch(...){
96              cout<<"\n Catch all realTimeDemo. Should not be here!"<<endl;
97          }
98      }
99
100     if (SP->IsConnected()){
101         cout<<"\n Ardiuno Connected!"<<endl;
102     }else{
103         cout<<"\n Arduino not connected?.. Terminating!"<<endl;
104     }
105     char* token;
106     string incomingDataString;
107     char incomingData[256] = "";
108     int dataLength = 255;
109     int readResult = 0;
110     int data1 = 0;
111     int data2 = 0;
112     int data3 = 0;
113     int bufferSize = 5;
114     int bufferHold1,bufferHold2;
115     int buf1[bufferSize], buf2[bufferSize], buf3[bufferSize];
116     int data1Hold, data2Hold, data3Hold;
117     int average1,average2,average3, i;
118
119     int slope1, slope2, slope3;
120     int estimatedPoint1, estimatedPoint2, estimatedPoint3;
121     double percentDiff1, percentDiff2, percentDiff3;
122     double averageDelta;
123     while(1){
124
125         data1Hold = data1;
126         data2Hold = data2;
127         data3Hold = data3;
128         readResult = SP->ReadData(incomingData,dataLength);
129
130         token = strtok(incomingData,","); //parsing serial output from arduino
131         data1 = atoi(token);
132         token = strtok(NULL, ",");
133         data2 = atoi(token);
134         token = strtok(NULL, ",");
135         data3 = atoi(token);
136
137         if(buf1[0] == 0) { //if buffers are empty
```

```
138             for(i=0;i<bufferSize;i++) {
139                 buf1[i] = data1; //fill them completely with first value
140                 buf2[i] = data2;
141                 buf3[i] = data3;
142             }
143         }
144         int sum = 0; //calculate averages of buffers and error check
145         /*
146         for(int i=0;i<bufferSize;i++){
147             sum+=buf1[i];
148         }
149         average1 = sum / bufferSize;
150         sum = 0;
151         for(int i = 0; i < bufferSize; i++){
152             sum += buf1[i] - average1;
153         }
154         averageDelta = sum / buffersize;
155         percentDiff1 = (data1 - average1) / averageDelta;
156         if(percentDiff1 > ACCEPTABLE_PERCENTAGE && percentDiff1 <
            (ACCEPTABLE_PERCENTAGE + .01)
157             || percentDiff1 < ACCEPTABLE_PERCENTAGE  && percentDiff1 >
                (ACCEPTABLE_PERCENTAGE - .01)){
158             data1 *= .9;
159         }
160         if(percentDiff1 > (ACCEPTABLE_PERCENTAGE + .01) && percentDiff1 <
            (ACCEPTABLE_PERCENTAGE + .05)
161             || percentDiff1 < (ACCEPTABLE_PERCENTAGE - .01)  && percentDiff1 >
                (ACCEPTABLE_PERCENTAGE - .05)){
162             data1 *= .8;
163         }
164         if(percentDiff1 > (ACCEPTABLE_PERCENTAGE + .05) && percentDiff1 <
            (ACCEPTABLE_PERCENTAGE + .1)
165             || percentDiff1 < (ACCEPTABLE_PERCENTAGE - .05)  && percentDiff1 >
                (ACCEPTABLE_PERCENTAGE - .1)){
166             data1 *= .6;
167         }
168         if(percentDiff1 > (ACCEPTABLE_PERCENTAGE + .1) && percentDiff1 <
            (ACCEPTABLE_PERCENTAGE + .2)
169             || percentDiff1 < (ACCEPTABLE_PERCENTAGE - .1)  && percentDiff1 >
                (ACCEPTABLE_PERCENTAGE - .2)){
170             data1 *= .;
171         }
172         */
173         sum = 0;
174         for(int i=0;i<bufferSize;i++)
175             sum+=buf1[i];
176         average1 = sum / bufferSize;
177         if (data1 > average1 + 10 | data1 < average1 - 10)
178             data1 = average1;
179         sum = 0;
180         for(int i=0;i<bufferSize;i++)
181             sum+=buf2[i];
182         average2 = sum / bufferSize;
183         if (data2 > average2 + 10 | data2 < average2 - 10)
184             data2 = average2;
185         sum = 0;
186         for(int i=0;i<bufferSize;i++)
187             sum+=buf3[i];
188         average3 = sum / bufferSize;
189         if (data3 > average3 + 10 | data3 < average3 - 10)
190             data3 = average3;
191
192
193          //rotate it in the new values to each buffer
194         bufferHold1 = buf1[0];
195         buf1[0] = data1;
196         for(int i=1;i<bufferSize;i++) {
197             bufferHold2 = buf1[i];
198             buf1[i] = bufferHold1;
```

```
199             bufferHold1 = bufferHold2;
200         }
201         bufferHold1 = buf2[0];
202         buf2[0] = data2;
203         for(int i=1;i<bufferSize;i++) {
204             bufferHold2 = buf2[i];
205             buf2[i] = bufferHold1;
206             bufferHold1 = bufferHold2;
207         }
208         bufferHold1 = buf3[0];
209         buf3[0] = data3;
210         for(int i =1;i<bufferSize;i++) {
211             bufferHold2 = buf3[i];
212             buf3[i] = bufferHold1;
213             bufferHold1 = bufferHold2;
214         }
215 
216         realTime->setData(data1, data2, data3); //output to graphics
217         realTime->draw(&mydc);
218         Sleep(150);
219         }
220     delete realTime;
221 }
222 
223 
224 
225 void realTimeGraphicsSetup(RealTimeGraphics* graphics){
226     string userInput;
227     HWND myconsole = GetConsoleWindow();
228     HDC mydc = GetDC(myconsole);
229     int value;
230     while(1){
231 
232         realTimeGraphicsSetupOptionMenu();
233         getline(cin,userInput);
234         if(userInput == "1"){
235             graphics->draw(&mydc);
236             getline(cin, userInput);
237         }else if(userInput == "2"){
238             int value2;
239             cout<<"\n Please enter the new X value: "<<endl;
240             cin>>value;
241             cout<<"\n Please enter the new Y value: "<<endl;
242             cin>>value2;
243             graphics->move(value, value2);
244         }else if(userInput == "3"){
245             cout<<"\n Please enter the value you wish to vertically shift the graph
                by:"<<endl;
246             cin>>value;
247             graphics->verticalShift(value);
248         }else if(userInput == "4"){
249             cout<<"\n Please enter the value you wish to horizontally shift the graph
                by:"<<endl;
250             cin>>value;
251             graphics->horizontalShift(value);
252         }else if(userInput == "5"){
253             int value2;
254             cout<<"\n Please enter the new length:"<<endl;
255             cin>>value;
256             cout<<"\n Please enter the new width:"<<endl;
257             cin>>value2;
258             graphics->resize(value, value2);
259         }else if(userInput == "6"){
260             int value2;
261             cout<<"\n Please enter the lowest temperature:"<<endl;
262             cin>>value;
263             cout<<"\n Please enter the highest temperature:"<<endl;
264             cin>>value2;
265             graphics->setRange(value, value2);
```

```
266         }else if(userInput == "7"){
267             break;
268         }else{
269             cout<<"\n Invalid Input!"<<endl;
270         }
271
272         fflush(stdin);
273     }
274     return;
275 }
276
277 void realTimeGraphicsSetupOptionMenu(void){
278     cout<<"\n Graphics Setup Options:"
279         <<"\n 1)Print"
280         <<"\n 2)Move"
281         <<"\n 3)Vertical Shift"
282         <<"\n 4)Horizontal Shift"
283         <<"\n 5)Resize graph"
284         <<"\n 6)Set temperature range"
285         <<"\n 7)Finish"<<endl;
286 }
287
288 void dataVisualization(void){
289     string userInput;
290     CarGraphics* graphicsCar = new CarGraphics();
291     graphicsSetup(graphicsCar);
292     Car* car = carSetup();
293
294     cout<<"\n\n\n\n Car creation and GUI setup complete.\nPress any key to
        continue"<<endl;
295     getline(cin, userInput);
296     simulation(car, graphicsCar);
297     cout<<"\n Simulation complete!"<<endl;
298     while(1){
299
300             cout<<"\n What would you like to do?"
301             <<"\n 1)Run the simulation"
302             <<"\n 2)Change car data"
303             <<"\n 3)Change graphics settings"
304             <<"\n 4)Quit to main menu"<<endl;
305         getline(cin,userInput);
306
307         if(userInput == "1"){
308             simulation(car, graphicsCar);
309         }else if(userInput == "2"){
310             delete car;
311             car = carSetup();
312         }else if(userInput == "3"){
313             graphicsSetup(graphicsCar);
314         }else if(userInput == "4"){
315             break;
316         }
317     }
318     delete car;
319     delete graphicsCar;
320
321 }
322
323 void graphicsSetup(CarGraphics* car){
324     string userInput;
325     HWND myconsole = GetConsoleWindow();
326     HDC mydc = GetDC(myconsole);
327     int value;
328     while(1){
329
330         graphicsSetupOptionMenu();
331         getline(cin,userInput);
332         if(userInput == "1"){
333             car->drawGraphics(&mydc);
```

```
334             car->drawGraphics(&mydc);
335             getline(cin, userInput);
336         }else if(userInput == "2"){
337             int x, y;
338             cout<<"\n Please enter the new X value:"<<endl;
339             cin>>x;
340             cout<<"\n Please enter the new Y value:"<<endl;
341             cin>>y;
342             car->move(x, y);
343         }else if(userInput == "3"){
344             cout<<"\n Please enter the value you wish to shift the object vertically
                by:"<<endl;
345             cin>>value;
346             car->verticalShift(value);
347         }else if(userInput == "4"){
348             cout<<"\n Please enter the value you wish to shift the object horizontally
                by:"<<endl;
349             cin>>value;
350             car->horizontalShift(value);
351         }else if(userInput == "5"){
352             cout<<"\n Please enter the new size for the tires:"<<endl;
353             cin>>value;
354             car->resize(value);
355         }else if(userInput == "6"){
356             cout<<"\n Please enter the new size for the front left tire:"<<endl;
357             cin>>value;
358             car->flResize(value);
359         }else if(userInput == "7"){
360             cout<<"\n Please enter the new size for the front right tire:"<<endl;
361             cin>>value;
362             car->frResize(value);
363         }else if(userInput == "8"){
364             cout<<"\n Please enter the new size for the rear left tire:"<<endl;
365             cin>>value;
366             car->rlResize(value);
367         }else if(userInput == "9"){
368             cout<<"\n Please enter the new size for the rear right tire:"<<endl;
369             cin>>value;
370             car->rrResize(value);
371         }else if(userInput == "10"){
372             cout<<"\n Please enter the new offset value:"<<endl;
373             cin>>value;
374             car->updateOffset(value, value);
375         }else if(userInput == "11"){
376             int value2;
377             cout<<"\n Please enter the low temperature:"<<endl;
378             cin>>value;
379             cout<<"\n Please enter the high temperature:"<<endl;
380             cin>>value2;
381             car->setRange(value, value2);
382         }else if(userInput == "12"){
383             break;
384         }else{
385             cout<<"\nInvalid Input!"<<endl;
386         }
387         fflush(stdin);
388     }
389 
390     return;
391 }
392 
393 void graphicsSetupOptionMenu(void){
394     cout<<"\n Graphics Setup Options:"
395         <<"\n 1)Print"
396         <<"\n 2)Move"
397         <<"\n 3)Vertical Shift"
398         <<"\n 4)Horizontal Shift"
399         <<"\n 5)Resize Every Tire"
400         <<"\n 6)Resize Front Left Tire"
```

```
401             <<"\n 7)Resize Front Right Tire"
402             <<"\n 8)Resize Rear Left Tire"
403             <<"\n 9)Resize Rear Right Tire"
404             <<"\n 10)Increase Offset From MidPoint"
405             <<"\n 11)Set temperature range"
406             <<"\n 12)Finish"<<endl;
407     }
408
409
410     void simulation(Car* car, CarGraphics* graphicsCar){
411         HWND myconsole = GetConsoleWindow();
412         HDC mydc = GetDC(myconsole);
413         graphicsCar->setRange(0, 110);
414         for(int i = 0; i < car->numDataPoints(); i++){
415             graphicsCar->setDataFrontLeft(car->getTemperature(frontLeft, outer, i),
416                                           car->getTemperature(frontLeft, middle, i),
417                                           car->getTemperature(frontLeft, inner, i));
418
419             graphicsCar->setDataFrontRight(car->getTemperature(frontRight, outer, i),
420                                           car->getTemperature(frontRight, middle, i),
421                                           car->getTemperature(frontRight, inner, i));
422
423             graphicsCar->setDataRearLeft(car->getTemperature(rearLeft, outer, i),
424                                           car->getTemperature(rearLeft, middle, i),
425                                           car->getTemperature(rearLeft, inner, i));
426
427             graphicsCar->setDataRearRight(car->getTemperature(rearRight, outer, i),
428                                           car->getTemperature(rearRight, middle, i),
429                                           car->getTemperature(rearRight, inner, i));
430             graphicsCar->drawGraphics(&mydc);
431             Sleep(20);
432
433
434         }
435     }
436
437     void dataVisualizationWelcomeMessage(void){
438         string input;
439         for(int i = 0; i < 60; i++){
440             cout<<"-";
441         }
442         cout<<"\n Setup Wizard"<<endl;
443         cout<<"\n This wizard will guide you through the steps to set up your data for"
444             <<"\n visualization. Please follow the steps carefully."
445             <<"\n Press any key to continue"<<endl;
446
447         getline(cin,input);
448     }
449
450     Car* carSetup(void){
451         string userInput;
452         dataVisualizationWelcomeMessage();
453         ifstream dataFile1, dataFile2, dataFile3, dataFile4;
454
455         cout<<"\n Please enter the name of the car : "<<endl;
456         getline(cin, userInput);
457
458         Car* car = new Car(userInput);
459         try{
460
461             while(1){
462                 cout<<"\n Please enter the name of the data file for the Front Left Tire:
                    "<<endl;
463                 getline(cin, userInput);
464                 if(userInput == "quit"){
465                     throw(1);
466                 }
467                 dataFile1.open((userInput.c_str()), ios::in);
468                 if(!dataFile1.is_open()){
```

```
469                 cout<<"\n Could not open the file! Please try again or enter 'quit' to
                    exit"<<endl;
470             }else{
471                 break;
472             }
473         }
474
475         while(1){
476             cout<<"\n Please enter the name of the data file for the Front Right Tire:
                "<<endl;
477             getline(cin, userInput);
478             if(userInput == "quit"){
479                 dataFile1.close();
480                 throw(1);
481             }
482             dataFile2.open((userInput.c_str()), ios::in);
483             if(!dataFile2.is_open()){
484                 cout<<"\n Could not open the file! Please try again or enter 'quit' to
                    exit"<<endl;
485             }else{
486                 break;
487             }
488         }
489
490         while(1){
491             cout<<"\n Please enter the name of the data file for the Rear Left Tire:
                "<<endl;
492             getline(cin, userInput);
493                 if(userInput == "quit"){
494                     dataFile1.close();
495                     dataFile2.close();
496                     throw(1);
497                 }
498             dataFile3.open((userInput.c_str()), ios::in);
499             if(!dataFile3.is_open()){
500                 cout<<"\n Could not open the file! Please try again or enter 'quit' to
                    exit"<<endl;
501             }else{
502                 break;
503             }
504         }
505
506         while(1){
507             cout<<"\n Please enter the name of the data file for the Rear Right Tire:
                "<<endl;
508             getline(cin, userInput);
509                 if(userInput == "quit"){
510                     dataFile3.close();
511                     throw(1);
512                 }
513             dataFile4.open((userInput.c_str()), ios::in);
514             if(!dataFile4.is_open()){
515                 cout<<"\n Could not open the file! Please try again or enter 'quit' to
                    exit"<<endl;
516             }else{
517                 break;
518             }
519         }
520         cout<<"\n Please wait while the Car is created and the data imported..."<<endl;
521         string line;
522         char* token;
523
524         //Front left tire creation
525         Tire FL(frontLeft, "FrontLeft");
526         Sensor FL_Outer("FL_Outer", outer, 0x5A);
527         Sensor FL_Middle("FL_Middle", middle, 0x5A);
528         Sensor FL_Inner("FL_Inner", inner, 0x5A);
529         //Data for the sensor objects.
530         while(getline(dataFile1, line)){
```

```
            token = strtok((char*)line.c_str(),",");
            FL_Outer.addTemperature(atoi(token));
            token = strtok(NULL, ",");
            FL_Middle.addTemperature(atoi(token));
            token = strtok(NULL, ",");
            FL_Inner.addTemperature(atoi(token));
        }
        dataFile1.close();
        //Add the sensors to the tire
        FL.addSensor(FL_Outer);
        FL.addSensor(FL_Middle);
        FL.addSensor(FL_Inner);

        //Front right tire creation
        Tire FR(frontRight, "FrontRight");
        Sensor FR_Outer("FR_Outer", outer, 0x5A);
        Sensor FR_Middle("FR_Middle", middle, 0x5A);
        Sensor FR_Inner("FR_Inner", inner, 0x5A);
        while(getline(dataFile2, line)){
            token = strtok((char*)line.c_str(),",");
            FR_Outer.addTemperature(atoi(token));
            token = strtok(NULL, ",");
            FR_Middle.addTemperature(atoi(token));
            token = strtok(NULL, ",");
            FR_Inner.addTemperature(atoi(token));
        }
        dataFile2.close();

        //Add the sensors to the tire
        FR.addSensor(FR_Outer);
        FR.addSensor(FR_Middle);
        FR.addSensor(FR_Inner);

        //Rear left tire creation
        Tire RL(rearLeft, "RearLeft");
        Sensor RL_Outer("RL_Outer", outer, 0x5A);
        Sensor RL_Middle("RL_Middle", middle, 0x5A);
        Sensor RL_Inner("RL_Inner", inner, 0x5A);
        while(getline(dataFile3, line)){
            token = strtok((char*)line.c_str(),",");
            RL_Outer.addTemperature(atoi(token));
            token = strtok(NULL, ",");
            RL_Middle.addTemperature(atoi(token));
            token = strtok(NULL, ",");
            RL_Inner.addTemperature(atoi(token));
        }
        dataFile3.close();

        //Add the sensors to the tire
        RL.addSensor(RL_Outer);
        RL.addSensor(RL_Middle);
        RL.addSensor(RL_Inner);

        //Rear right tire creation
        Tire RR(rearRight, "RearRight");
        Sensor RR_Outer("RR_Outer", outer, 0x5A);
        Sensor RR_Middle("RR_Middle", middle, 0x5A);
        Sensor RR_Inner("RR_Inner", inner, 0x5A);
        while(getline(dataFile4, line)){
            token = strtok((char*)line.c_str(),",");
            RR_Outer.addTemperature(atoi(token));
            token = strtok(NULL, ",");
            RR_Middle.addTemperature(atoi(token));
            token = strtok(NULL, ",");
            RR_Inner.addTemperature(atoi(token));
        }
        dataFile4.close();
        //Add the sensors to the tire
        RR.addSensor(RR_Outer);
```

```
600             RR.addSensor(RR_Middle);
601             RR.addSensor(RR_Inner);
602 
603             //Add all of the tires to the car
604             car->addTire(FL);
605             car->addTire(FR);
606             car->addTire(RL);
607             car->addTire(RR);
608 
609             return(car);
610         }
611         catch(...){
612             return(NULL);
613         }
614     }
615 
```

```
/*
    Marshall Lindsay
    Max Houck
    Formula SAE Tire Temperature Visualization
    ECE 3220 Final Project
    CarGraphics.cpp
*/

#include "CarGraphics.h"
#include <iostream>
#include <vector>
using namespace std;

/*****************************************************************************

*****************************************************************************/

/*****************************************************************************
                            BaseGraphics
*****************************************************************************/
BaseGraphics::BaseGraphics(){
    //Set a random starting point. If I knew more about consoles and
    //graphics in general I would grab the midpoint of their window.
    this->x = 500;
    this->y = 500;
}

BaseGraphics::BaseGraphics(int _x, int _y){
    this->x = _x;
    this->y = _y;
}

void BaseGraphics::move(int _x, int _y){
    this->x = _x;
    this->y = _y;
}

/*****************************************************************************
                            CarGraphics
*****************************************************************************/
CarGraphics::CarGraphics() : BaseGraphics(){
    this->flRectSize = 50;
    this->frRectSize = 50;
    this->rlRectSize = 50;
    this->rrRectSize = 50;
    this->xCord_offsetStart = 200;
    this->yCord_offsetStart = 200;

    this->setDataFrontLeft(0,0,0);
    this->setDataFrontRight(0,0,0);
    this->setDataRearLeft(0,0,0);
    this->setDataRearRight(0,0,0);

    this->updateGradients();
    this->updateRectangles();
    this->tempLow = 0;
    this->tempHigh = 110;

}

CarGraphics::CarGraphics(int x, int y, int size, int xOffset, int yOffset) :
BaseGraphics(x, y){
    this->flRectSize = size;
    this->frRectSize = size;
    this->rlRectSize = size;
    this->rrRectSize = size;

    this->xCord_offsetStart = xOffset;
    this->yCord_offsetStart = yOffset;
```

```
69
70      this->setDataFrontLeft(0,0,0);
71      this->setDataFrontRight(0,0,0);
72      this->setDataRearLeft(0,0,0);
73      this->setDataRearRight(0,0,0);
74
75
76      this->updateGradients();
77      this->updateRectangles();
78      this->tempLow = 0;
79      this->tempHigh = 110;
80  }
81
82  void CarGraphics::setDataFrontLeft(double value1, double value2, double value3){
83      if(value1 > this->tempHigh){
84          value1 = this->tempHigh;
85      }
86      if(value2 > this->tempHigh){
87          value2 = this->tempHigh;
88      }
89      if(value3 > this->tempHigh){
90          value3 = this->tempHigh;
91      }
92      if(value1 < this->tempLow){
93          value1 = this->tempLow;
94      }
95      if(value2 < this->tempLow){
96          value2 = this->tempLow;
97      }
98      if(value3 < this->tempLow){
99          value3 = this->tempLow;
100     }
101     this->flDataPointOutside = value1 - this->tempLow;
102     this->flDataPointMiddle = value2 - this->tempLow;
103     this->flDataPointInside = value3 - this->tempLow;
104     this->updateGradients();
105 }
106
107 void CarGraphics::setDataFrontRight(double value1, double value2, double value3){
108     if(value1 > this->tempHigh){
109         value1 = this->tempHigh;
110     }
111     if(value2 > this->tempHigh){
112         value2 = this->tempHigh;
113     }
114     if(value3 > this->tempHigh){
115         value3 = this->tempHigh;
116     }
117     if(value1 < this->tempLow){
118         value1 = this->tempLow;
119     }
120     if(value2 < this->tempLow){
121         value2 = this->tempLow;
122     }
123     if(value3 < this->tempLow){
124         value3 = this->tempLow;
125     }
126     this->frDataPointOutside = value1 - this->tempLow;
127     this->frDataPointMiddle = value2 - this->tempLow;
128     this->frDataPointInside = value3 - this->tempLow;
129     this->updateGradients();
130 }
131
132 void CarGraphics::setDataRearLeft(double value1, double value2, double value3){
133     if(value1 > this->tempHigh){
134         value1 = this->tempHigh;
135     }
136     if(value2 > this->tempHigh){
137         value2 = this->tempHigh;
```

```
138         }
139         if(value3 > this->tempHigh){
140             value3 = this->tempHigh;
141         }
142         if(value1 < this->tempLow){
143             value1 = this->tempLow;
144         }
145         if(value2 < this->tempLow){
146             value2 = this->tempLow;
147         }
148         if(value3 < this->tempLow){
149             value3 = this->tempLow;
150         }
151         this->rlDataPointOutside = value1 - this->tempLow;
152         this->rlDataPointMiddle = value2 - this->tempLow;
153         this->rlDataPointInside = value3 - this->tempLow;
154         this->updateGradients();
155     }
156     void CarGraphics::setDataRearRight(double value1, double value2, double value3){
157         if(value1 > this->tempHigh){
158             value1 = this->tempHigh;
159         }
160         if(value2 > this->tempHigh){
161             value2 = this->tempHigh;
162         }
163         if(value3 > this->tempHigh){
164             value3 = this->tempHigh;
165         }
166         if(value1 < this->tempLow){
167             value1 = this->tempLow;
168         }
169         if(value2 < this->tempLow){
170             value2 = this->tempLow;
171         }
172         if(value3 < this->tempLow){
173             value3 = this->tempLow;
174         }
175         this->rrDataPointOutside = value1 - this->tempLow;
176         this->rrDataPointMiddle = value2 - this->tempLow;
177         this->rrDataPointInside = value3 - this->tempLow;
178         this->updateGradients();
179     }
180
181     void CarGraphics::move(int nx, int ny){
182         this->x = nx;
183         this->y = ny;
184         this->updateGradients();
185         this->updateRectangles();
186     }
187
188     vector<double> CarGraphics::calculateGradient(double point1, double point2, double
    point3, int rectSize){
189         double slope1, slope2, slope3, slope4;
190         double quaterRect = rectSize / 4;
191         double halfRect = rectSize / 2;
192         double threeQuaterRect = rectSize *((double)(3) / (double)(4));
193         vector<double> slopes;
194         slope1 = (0 - point1) / (0 - quaterRect);
195         slope2 = (point1 - point2) / (quaterRect - halfRect);
196         slope3 = (point2 - point3) / (halfRect - threeQuaterRect);
197         slope4 = (point3 - 0) / (threeQuaterRect - rectSize);
198
199         slopes.push_back(slope1);
200         slopes.push_back(slope2);
201         slopes.push_back(slope3);
202         slopes.push_back(slope4);
203         return(slopes);
204     }
205
```

```
206  void CarGraphics::updateRectangles(void){
207      this->frontLeft.left = this->x - this->xCord_offsetStart;
208      this->frontLeft.right = this->frontLeft.left + 1;
209      this->frontLeft.top = this->y - this->yCord_offsetStart;
210      this->frontLeft.bottom = this->frontLeft.top + this->flRectSize;
211
212      this->frontRight.left = this->x + this->xCord_offsetStart;
213      this->frontRight.right = this->frontRight.left + 1;
214      this->frontRight.top = this->y - this->yCord_offsetStart;
215      this->frontRight.bottom = this->frontRight.top + this->frRectSize;
216
217      this->rearLeft.left = this->x - this->xCord_offsetStart;
218      this->rearLeft.right = this->rearLeft.left + 1;
219      this->rearLeft.top = this->y + this->yCord_offsetStart;
220      this->rearLeft.bottom = this->rearLeft.top + this->rlRectSize;
221
222      this->rearRight.left = this->x + this->xCord_offsetStart;
223      this->rearRight.right = this->rearRight.left + 1;
224      this->rearRight.top = this->y + this->yCord_offsetStart;
225      this->rearRight.bottom = this->rearRight.top + this->rrRectSize;
226  }
227
228  void CarGraphics::updateGradients(void){
229      this->flGradient = this->calculateGradient(this->flDataPointOutside,
230                                                 this->flDataPointMiddle,
231                                                 this->flDataPointInside,
232                                                 this->flRectSize);
233
234      this->frGradient = this->calculateGradient(this->frDataPointOutside,
235                                                 this->frDataPointMiddle,
236                                                 this->frDataPointInside,
237                                                 this->frRectSize);
238
239      this->rlGradient = this->calculateGradient(this->rlDataPointOutside,
240                                                 this->rlDataPointMiddle,
241                                                 this->rlDataPointInside,
242                                                 this->rlRectSize);
243
244      this->rrGradient = this->calculateGradient(this->rrDataPointOutside,
245                                                 this->rrDataPointMiddle,
246                                                 this->rrDataPointInside,
247                                                 this->rrRectSize);
248  }
249
250  void CarGraphics::resize(int newSize){
251      this->flRectSize = newSize;
252      this->frRectSize = newSize;
253      this->rlRectSize = newSize;
254      this->rrRectSize = newSize;
255
256      this->updateGradients();
257      this->updateRectangles();
258  }
259
260  void CarGraphics::flResize(int newSize){
261      this->flRectSize = newSize;
262      this->updateGradients();
263      this->updateRectangles();
264  }
265
266  void CarGraphics::frResize(int newSize){
267      this->frRectSize = newSize;
268      this->updateGradients();
269      this->updateRectangles();
270  }
271
272  void CarGraphics::rlResize(int newSize){
273      this->rlRectSize = newSize;
274      this->updateGradients();
```

```
275         this->updateRectangles();
276     }
277
278     void CarGraphics::rrResize(int newSize){
279         this->rrRectSize = newSize;
280         this->updateGradients();
281         this->updateRectangles();
282     }
283
284     void CarGraphics::horizontalShift(int dx){
285         this->x = this->x + dx;
286         this->updateRectangles();
287     }
288
289     void CarGraphics::verticalShift(int dy){
290         this->y = this->y + dy;
291         this->updateRectangles();
292     }
293
294     void CarGraphics::setRange(int low, int high){
295         this->tempLow = low;
296         this->tempHigh = high;
297         this->updateGradients();
298     }
299
300     void CarGraphics::updateOffset(int nx, int ny){
301         this->xCord_offsetStart = nx;
302         this->yCord_offsetStart = ny;
303         this->updateGradients();
304         this->updateRectangles();
305     }
306     //Need to implement a moving of rectangles.*******************************************
307
308
309     void CarGraphics::drawGraphics(HDC* mydc){
310         HBRUSH myBrush;
311         RECT tempRect;
312
313         double flQuaterRect = this->flRectSize / 4;
314         double flHalfRect = this->flRectSize / 2;
315         double flThreeQuaterRect = this->flRectSize *((double)(3) / (double)(4));
316
317         double frQuaterRect = this->frRectSize / 4;
318         double frHalfRect = this->frRectSize / 2;
319         double frThreeQuaterRect = this->frRectSize *((double)(3) / (double)(4));
320
321         double rlQuaterRect = this->rlRectSize / 4;
322         double rlHalfRect = this->rlRectSize / 2;
323         double rlThreeQuaterRect = this->rlRectSize *((double)(3) / (double)(4));
324
325         double rrQuaterRect = this->rrRectSize / 4;
326         double rrHalfRect = this->rrRectSize / 2;
327         double rrThreeQuaterRect = this->rrRectSize *((double)(3) / (double)(4));
328
329         int red, green, blue;
330         int RGB;
331         int tempRange = this->tempHigh - this->tempLow;
332         for(double i = 0; i < this->flRectSize; i++){
333
334             //FrontLeft
335             tempRect.left = i + this->frontLeft.left;
336             tempRect.top = this->frontLeft.top;
337             tempRect.right = tempRect.left + 1;
338             tempRect.bottom = this->frontLeft.bottom;
339
340             if(i <= flQuaterRect){
341                 RGB = (510/tempRange)*(0 +(this->flGradient[0] * i));
342             }
343             else if(i > flQuaterRect && i <= flHalfRect){
```

```
344             RGB = (510/tempRange)*(this->flDataPointOutside + (this->flGradient[1] *(i
                - flQuaterRect)));
345         }
346         else if(i > flHalfRect && i <= flThreeQuaterRect){
347             RGB = (510/tempRange)*(this->flDataPointMiddle + (this->flGradient[2] *(i -
                flHalfRect)));
348         }
349         else if(i > flThreeQuaterRect){
350             RGB = (510/tempRange)*(this->flDataPointInside + (this->flGradient[3] *(i -
                flThreeQuaterRect)));
351         }
352
353         //cout<<"\n "<<i<<" "<<RGB;
354         if(RGB <= 255){
355             blue = (-RGB) + 255;
356             red = 0;
357             green = (RGB);
358         }else if(RGB >= 255){
359             blue = 0;
360             green = (-RGB) + 510;
361             red = (RGB) - 255;
362         }
363         myBrush = CreateSolidBrush(RGB(red, green, blue));
364         FillRect(*mydc, &tempRect, myBrush);
365         DeleteObject(myBrush);
366     }
367     for(double i = 0; i < this->frRectSize; i++){
368         //Front Right
369         tempRect.left = i + this->frontRight.left;
370         tempRect.top = this->frontRight.top;
371         tempRect.right = tempRect.left + 1;
372         tempRect.bottom = this->frontRight.bottom;
373
374         if(i <= frQuaterRect){
375             RGB = (510/tempRange)*(0 +(this->frGradient[0] * i));
376         }
377         else if(i > frQuaterRect && i <= frHalfRect){
378             RGB = (510/tempRange)*(this->frDataPointOutside + (this->frGradient[1] *(i
                - frQuaterRect)));
379         }
380         else if(i > frHalfRect && i <= frThreeQuaterRect){
381             RGB = (510/tempRange)*(this->frDataPointMiddle + (this->frGradient[2] *(i -
                frHalfRect)));
382         }
383         else if(i > frThreeQuaterRect){
384             RGB = (510/tempRange)*(this->frDataPointInside + (this->frGradient[3] *(i -
                frThreeQuaterRect)));
385         }
386
387         //cout<<"\n "<<i<<" "<<RGB;
388         if(RGB <= 255){
389             blue = (-RGB) + 255;
390             red = 0;
391             green = (RGB);
392         }else if(RGB >= 255){
393             blue = 0;
394             green = (-RGB) + 510;
395             red = (RGB) - 255;
396         }
397         myBrush = CreateSolidBrush(RGB(red, green, blue));
398         FillRect(*mydc, &tempRect, myBrush);
399         DeleteObject(myBrush);
400     }
401     for(double i = 0; i < this->rlRectSize; i++){
402
403         //Rear Left
404         tempRect.left = i + this->rearLeft.left;
405         tempRect.top = this->rearLeft.top;
406         tempRect.right = tempRect.left + 1;
```

```
407             tempRect.bottom = this->rearLeft.bottom;
408
409             if(i <= rlQuaterRect){
410                 RGB = (510/tempRange)*(0 +(this->rlGradient[0] * i));
411             }
412             else if(i > rlQuaterRect && i <= rlHalfRect){
413                 RGB = (510/tempRange)*(this->rlDataPointOutside + (this->rlGradient[1] *(i
                    - rlQuaterRect)));
414             }
415             else if(i > rlHalfRect && i <= rlThreeQuaterRect){
416                 RGB = (510/tempRange)*(this->rlDataPointMiddle + (this->rlGradient[2] *(i -
                    rlHalfRect)));
417             }
418             else if(i > rlThreeQuaterRect){
419                 RGB = (510/tempRange)*(this->rlDataPointInside + (this->rlGradient[3] *(i -
                    rlThreeQuaterRect)));
420             }
421
422             //cout<<"\n "<<i<<" "<<RGB;
423             if(RGB <= 255){
424                 blue = (-RGB) + 255;
425                 red = 0;
426                 green = (RGB);
427             }else if(RGB >= 255){
428                 blue = 0;
429                 green = (-RGB) + 510;
430                 red = (RGB) - 255;
431             }
432             myBrush = CreateSolidBrush(RGB(red, green, blue));
433             FillRect(*mydc, &tempRect, myBrush);
434             DeleteObject(myBrush);
435         }
436         for(double i = 0; i < this->rrRectSize; i++){
437             //Rear right
438             tempRect.left = i + this->rearRight.left;
439             tempRect.top = this->rearRight.top;
440             tempRect.right = tempRect.left + 1;
441             tempRect.bottom = this->rearRight.bottom;
442
443             if(i <= rrQuaterRect){
444                 RGB = (510/tempRange)*(0 +(this->rrGradient[0] * i));
445             }
446             else if(i > rrQuaterRect && i <= rrHalfRect){
447                 RGB = (510/tempRange)*(this->rrDataPointOutside + (this->rrGradient[1] *(i
                    - rrQuaterRect)));
448             }
449             else if(i > rrHalfRect && i <= rrThreeQuaterRect){
450                 RGB = (510/tempRange)*(this->rrDataPointMiddle + (this->rrGradient[2] *(i -
                    rrHalfRect)));
451             }
452             else if(i > rrThreeQuaterRect){
453                 RGB = (510/tempRange)*(this->rrDataPointInside + (this->rrGradient[3] *(i -
                    rrThreeQuaterRect)));
454             }
455
456             //cout<<"\n "<<i<<" "<<RGB;
457             if(RGB <= 255){
458                 blue = (-RGB) + 255;
459                 red = 0;
460                 green = (RGB);
461             }else if(RGB >= 255){
462                 blue = 0;
463                 green = (-RGB) + 510;
464                 red = (RGB) - 255;
465             }
466             myBrush = CreateSolidBrush(RGB(red, green, blue));
467             FillRect(*mydc, &tempRect, myBrush);
468             DeleteObject(myBrush);
469
```

```
470         }
471     }
472
473     /*************************************************************************
474                                 Real Time Graphics
475     *************************************************************************/
476     RealTimeGraphics::RealTimeGraphics() : BaseGraphics(){
477         this->length = 300;
478         this->width = 100;
479
480         this->outsideDataPoint = 0;
481         this->middleDataPoint = 0;
482         this->insideDataPoint = 0;
483
484         this->tempLow = 0;
485         this->tempHigh = 110;
486
487         updateRectangle();
488         updateGradient();
489
490     }
491
492     void RealTimeGraphics::setData(double value1, double value2, double value3){
493         if(value1 > this->tempHigh){
494             value1 = this->tempHigh;
495         }
496         if(value2 > this->tempHigh){
497             value2 = this->tempHigh;
498         }
499         if(value3 > this->tempHigh){
500             value3 = this->tempHigh;
501         }
502         if(value1 < this->tempLow){
503             value1 = this->tempLow;
504         }
505         if(value2 < this->tempLow){
506             value2 = this->tempLow;
507         }
508         if(value3 < this->tempLow){
509             value3 = this->tempLow;
510         }
511         this->outsideDataPoint = value1 - this->tempLow;
512         this->middleDataPoint = value2 - this->tempLow;
513         this->insideDataPoint = value3 - this->tempLow;
514         updateGradient();
515         updateRectangle();
516     }
517
518     vector<double> RealTimeGraphics::calculateGradient(double point1, double point2, double
    point3, int size){
519         double slope1, slope2, slope3, slope4;
520         double quaterRect = size / 4;
521         double halfRect = size / 2;
522         double threeQuaterRect = size *((double)(3) / (double)(4));
523         vector<double> slopes;
524         slope1 = (0 - point1) / (0 - quaterRect);
525         slope2 = (point1 - point2) / (quaterRect - halfRect);
526         slope3 = (point2 - point3) / (halfRect - threeQuaterRect);
527         slope4 = (point3 - 0) / (threeQuaterRect - size);
528
529         slopes.push_back(slope1);
530         slopes.push_back(slope2);
531         slopes.push_back(slope3);
532         slopes.push_back(slope4);
533         return(slopes);
534     }
535
536     void RealTimeGraphics::updateRectangle(void){
537         this->rectangle.left = this->x;
```

```
538         this->rectangle.right = this->rectangle.left + 1;
539         this->rectangle.top = this->y;
540         this->rectangle.bottom = this->rectangle.top + this->width;
541     }
542
543     void RealTimeGraphics::updateGradient(void){
544         this->gradient = this->calculateGradient(this->outsideDataPoint,
545                                                  this->middleDataPoint,
546                                                  this->insideDataPoint,
547                                                  this->length);
548     }
549
550     void RealTimeGraphics::resize(int nLength, int nWidth){
551         this->length = nLength;
552         this->width = nWidth;
553         updateGradient();
554         updateRectangle();
555     }
556
557     void RealTimeGraphics::horizontalShift(int dx){
558         this->x = this->x + dx;
559         updateRectangle();
560     }
561
562     void RealTimeGraphics::verticalShift(int dy){
563         this->y = this->y + dy;
564         updateRectangle();
565     }
566
567     void RealTimeGraphics::setRange(int low, int high){
568         this->tempLow = low;
569         this->tempHigh = high;
570         updateGradient();
571     }
572
573     void RealTimeGraphics::move(int nx, int ny){
574         this->x = nx;
575         this->y = ny;
576         updateRectangle();
577         updateGradient();
578     }
579
580     void RealTimeGraphics::draw(HDC* mydc){
581         double quaterRect = this->length / 4;
582         double halfRect = this->length / 2;
583         double threeQuaterRect = this->length *((double)(3) / (double)(4));
584         int red, blue, green;
585         int RGB;
586         RECT tempRect = this->rectangle;
587         HBRUSH myBrush;
588         int tempRange = this->tempHigh - this->tempLow;
589
590         for(double i = 0; i < this->length; i++){
591             //Front Right
592             tempRect.left = i + this->rectangle.left;
593             tempRect.top = this->rectangle.top;
594             tempRect.right = tempRect.left + 1;
595             tempRect.bottom = this->rectangle.bottom;
596
597             if(i <= quaterRect){
598                 RGB = (510/tempRange)*(0 +(this->gradient[0] * i));
599             }
600             else if(i > quaterRect && i <= halfRect){
601                 RGB = (510/tempRange)*(this->outsideDataPoint + (this->gradient[1] *(i -
                    quaterRect)));
602             }
603             else if(i > halfRect && i <= threeQuaterRect){
604                 RGB = (510/tempRange)*(this->middleDataPoint + (this->gradient[2] *(i -
                    halfRect)));
```

```
605             }
606             else if(i > threeQuaterRect){
607                 RGB = (510/tempRange)*(this->insideDataPoint + (this->gradient[3] *(i -
                    threeQuaterRect)));
608             }
609
610             //cout<<"\n "<<i<<" "<<RGB;
611             if(RGB <= 255){
612                 blue = (-RGB) + 255;
613                 red = 0;
614                 green = (RGB);
615             }else if(RGB >= 255){
616                 blue = 0;
617                 green = (-RGB) + 510;
618                 red = (RGB) - 255;
619             }
620             myBrush = CreateSolidBrush(RGB(red, green, blue));
621             FillRect(*mydc, &tempRect, myBrush);
622             DeleteObject(myBrush);
623         }
624     }
625
```

```
/*
    Marshall Lindsay
    Max Houck
    Formula SAE Tire Temperature Visualization
    ECE 3220 Final Project
    tires.cpp
*/


#include <iostream>
#include <vector>
#include <fstream>
#include <string>
#include "tires.h"
using namespace std;

/*****************************************************************************
                          TIRE CLASS METHODS
*****************************************************************************/
/*****************************************************************************
    Tire class parametric constuctor. Takes the location of the new tire as
the tireLocation enum as well as the name of the new tire and sets the
corresponding values in the object.

*****************************************************************************/
Tire::Tire(tireLocation tireLoc, string newName){
    this->location = tireLoc;
    this->name = newName;
}

/*****************************************************************************
    Tire class addSensor method. Takes a Sensor object as an arguement
and adds it to the sensorArray vector.

*****************************************************************************/
void Tire::addSensor(Sensor newSensor){
    this->sensorArray.push_back(newSensor);
}


/*****************************************************************************
    Tire class removeSensor method. Accepts the location of the sensor
as a sensorLocation enum and removes the selected sensor from the
sensorArray vector. It also checks to make sure the senssorArray vector
is not empty. If the vector is empty, it will throw an const int equal to
1.

*****************************************************************************/

void Tire::removeSensor(sensorLocation sensorLoc) throw(const int){
    //If the sensor array is empty, throw the number 1
    if(this->sensorArray.size() <= 0){
        throw(1);
    }

    for(int i = 0; i < this->sensorArray.size(); i++){
        if(this->sensorArray[i].location == sensorLoc){
            this->sensorArray.erase(this->sensorArray.begin()+i);
            break;
        }
    }

}

/*****************************************************************************
    Tire class printInfo method. Prints all of the information about the
tire to the screen.
*****************************************************************************/
void Tire::printInfo(void){
```

```
70          cout<<"\n Name: "<<this->name
71              <<"\n Location: "<< this->location<<endl;
72
73
74          cout<<" Sensor vector contents: "<<endl;
75          if(this->sensorArray.size() <= 0){
76              cout<<"  Sensor array is empty!"<<endl;
77          }else{
78              for(int i = 0; i < this->sensorArray.size(); i++){
79                  sensorArray[i].printInfo();
80              }
81          }
82      }
83
84      /*************************************************************************
85                                SENSOR CLASS METHODS
86      *************************************************************************/
87      /*************************************************************************
88          Sensor class parametric constructor. Takes the name, the location as a
89      sensorLoaction enum, and the address as arguements. Sets the corresponding
90      fields to the given values.
91
92      *************************************************************************/
93      Sensor::Sensor(string newName, sensorLocation sensorLoc, int sensorAddress){
94          this->name = newName;
95          this->location = sensorLoc;
96          this->address = sensorAddress;
97          this->ambTemperature = 0;
98      }
99
100     /*************************************************************************
101         Sensor class addTemperature method. Takes a temperature as an integer
102     and adds it to the objTemperature vector.
103     *************************************************************************/
104     void Sensor::addTemperature(int temperature){
105         this->objTemperature.push_back(temperature);
106     }
107
108     /*************************************************************************
109         Sensor class getTemperature method. Returns the most recent
110     temperature from the objTemperature vector. Checks to make sure the
111     objTemperature vector is not empty. If the vector is empty, returns a
112     const int equal to 2.
113     *************************************************************************/
114     int Sensor::getTemperature(void) throw(const int){
115         if(this->objTemperature.size() <= 0){
116             throw(2);
117         }
118         return(this->objTemperature.back());
119     }
120
121     /*************************************************************************
122         Sensor class printEverything method. Used for debugging, this method
123     does exactly what the name implies. It will print everything contained in
124     the class to the screen.
125     *************************************************************************/
126     void Sensor::printInfo(void){
127         cout<<"\n  "<<this->name<<" information: "
128             <<"\n   Sensor location: "<<this->location
129             <<"\n   Ambient temperature: "<<this->ambTemperature
130             <<"\n   Address : "<<this->address<<endl;
131
132         cout<<"   Temperature vector contents: "<<endl;
133         if(this->objTemperature.size() <= 0){
134             cout<<"   The vector is empty!"<<endl;
135         }else{
136             for(int i = 0; i < this->objTemperature.size(); i++){
137                 cout<<"    "<<this->objTemperature[i];
138                 if(i % 3 == 0)
```

```
                cout<<"\n";
        }
        cout<<"\n------------------------------------------------------"<<endl;
    }
}

/*************************************************************************
                        CAR CLASS METHODS
*************************************************************************/
/*************************************************************************
    Car class parametric constructor. Takes the name of the car as a string
and sets the corresponding field.
*************************************************************************/
Car::Car(string newName){
    this->name = newName;
}

/*************************************************************************
    Car class printInfo method. Prints all of the information about the car
to the screen.
*************************************************************************/
void Car::printInfo(void){
        cout<<"\n"<<this->name<<endl;
        cout<<"Tire vector contents: "<<endl;
        if(this->tireArray.size() <= 0){
            cout<<"Tire array is empty!"<<endl;
        }else{
            for(int i = 0; i < this->tireArray.size(); i++){
                    cout<<"\n Tire "<<i<<endl;
                    tireArray[i].printInfo();
            }
        }
        cout<<"-----------------------------------"<<endl;
}

/*************************************************************************
    Car class getTemperature method. Takes the tire location,
sensor location (as tireLocation and sensorLocation respectively), and the
index of the temperature value. Returns the temperature at that index as
an integer.
*************************************************************************/
int Car::getTemperature(tireLocation tireLoc, sensorLocation sensorLoc, int index){
    return(this->tireArray[tireLoc-1].sensorArray[sensorLoc-1].objTemperature[index]);
}
/*************************************************************************
    Car class numDataPoints method. Returns the number of data points in
the temperature vectors as an integer.
*************************************************************************/
int Car::numDataPoints(void){
    return(this->tireArray[0].sensorArray[0].objTemperature.size());
}
/*************************************************************************
    Car class addTire method. Takes a Tire object as an arguement and
adds it to the tire vector.
*************************************************************************/
void Car::addTire(Tire newTire){
    this->tireArray.push_back(newTire);
}



//Parametric constructor. This will be the only way a badVector is created.
badVector::badVector(vector<int> vect, char tireLoc, char vectorLoc){
    this->size = vect.size();
    this->tireLocation = tireLoc;
    this->vectorLocation = vectorLoc;
}

```

```
208  //Default destructor.
209  badVector::~badVector(void){
210      //Nothing is needed here.
211  }
212
213  //Prints the message corresponding to the error.
214  void badVector::badVectorMsg_InvalidSize(void){
215          string tireName;
216          string vectorLocation;
217          switch(this->tireLocation){
218              case 1:
219                  tireName = "frontLeft";
220                  break;
221              case 2:
222                  tireName = "frontRight";
223                  break;
224              case 3:
225                  tireName = "rearLeft";
226                  break;
227              case 4:
228                  tireName = "rearRight";
229                  break;
230              default:
231                  cout<<"Error in badVectorMsg_InvalidSize"<<endl;
232                  exit(2);
233                  break;
234
235          }//End switch
236          switch(this->vectorLocation){
237              case 'o':
238                  vectorLocation = "outer";
239                  break;
240              case 'm':
241                  vectorLocation = "middle";
242              case 'i':
243                  vectorLocation = "inner";
244              default:
245                  cout<<"Error in badVectorMsd_InvalidSize vectorLocation switch"<<endl;
246                  exit(3);
247                  break;
248
249          }//End switch
250          cout<<"Could not access "<< tireName<< " "<<vectorLocation<< " vector. This
             vectors size is "<< this->size<<endl;
251  }
```

```
/*
    Marshall Lindsay
    Max Houck
    Formula SAE Tire Temperature Visualization
    ECE 3220 Final Project
    serial.cpp
*/

#include "SerialClass.h"

Serial::Serial(const char *portName)
{
    //We're not yet connected
    this->connected = false;

    //Try to connect to the given port throuh CreateFile
    this->hSerial = CreateFile(portName,
            GENERIC_READ | GENERIC_WRITE,
            0,
            NULL,
            OPEN_EXISTING,
            FILE_ATTRIBUTE_NORMAL,
            NULL);

    //Check if the connection was successfull
    if(this->hSerial==INVALID_HANDLE_VALUE)
    {
        //If not success full display an Error
        if(GetLastError()==ERROR_FILE_NOT_FOUND){

            //Print Error if neccessary
            printf("ERROR: Handle was not attached. Reason: %s not available.\n",
            portName);
            throw(1);

        }
        else
        {
            printf("ERROR!!!");
            throw(2);
        }
    }
    else
    {
        //If connected we try to set the comm parameters
        DCB dcbSerialParams = {0};

        //Try to get the current
        if (!GetCommState(this->hSerial, &dcbSerialParams))
        {
            //If impossible, show an error
            printf("failed to get current serial parameters!");
        }
        else
        {
            //Define serial connection parameters for the arduino board
            dcbSerialParams.BaudRate=CBR_9600;
            dcbSerialParams.ByteSize=8;
            dcbSerialParams.StopBits=ONESTOPBIT;
            dcbSerialParams.Parity=NOPARITY;
            //Setting the DTR to Control_Enable ensures that the Arduino is properly
            //reset upon establishing a connection
            dcbSerialParams.fDtrControl = DTR_CONTROL_ENABLE;

             //Set the parameters and check for their proper application
             if(!SetCommState(hSerial, &dcbSerialParams))
             {
                printf("ALERT: Could not set Serial Port parameters");
             }
```

```
69              else
70              {
71                  //If everything went fine we're connected
72                  this->connected = true;
73                  //Flush any remaining characters in the buffers
74                  PurgeComm(this->hSerial, PURGE_RXCLEAR | PURGE_TXCLEAR);
75                  //We wait 2s as the arduino board will be reseting
76                  Sleep(ARDUINO_WAIT_TIME);
77              }
78          }
79      }
80
81  }
82
83  Serial::~Serial()
84  {
85      //Check if we are connected before trying to disconnect
86      if(this->connected)
87      {
88          //We're no longer connected
89          this->connected = false;
90          //Close the serial handler
91          CloseHandle(this->hSerial);
92      }
93  }
94
95  int Serial::ReadData(char *buffer, unsigned int nbChar)
96  {
97      //Number of bytes we'll have read
98      DWORD bytesRead;
99      //Number of bytes we'll really ask to read
100     unsigned int toRead;
101
102     //Use the ClearCommError function to get status info on the Serial port
103     ClearCommError(this->hSerial, &this->errors, &this->status);
104
105     //Check if there is something to read
106     if(this->status.cbInQue>0)
107     {
108         //If there is we check if there is enough data to read the required number
109         //of characters, if not we'll read only the available characters to prevent
110         //locking of the application.
111         if(this->status.cbInQue>nbChar)
112         {
113             toRead = nbChar;
114         }
115         else
116         {
117             toRead = this->status.cbInQue;
118         }
119
120         //Try to read the require number of chars, and return the number of read bytes
            on success
121         if(ReadFile(this->hSerial, buffer, toRead, &bytesRead, NULL) )
122         {
123             return bytesRead;
124         }
125
126     }
127
128     //If nothing has been read, or that an error was detected return 0
129     return 0;
130
131 }
132
133
134 bool Serial::WriteData(const char *buffer, unsigned int nbChar)
135 {
136     DWORD bytesSend;
```

```
137
138         //Try to write the buffer on the Serial port
139         if(!WriteFile(this->hSerial, (void *)buffer, nbChar, &bytesSend, 0))
140         {
141             //In case it don't work get comm error and return false
142             ClearCommError(this->hSerial, &this->errors, &this->status);
143
144             return false;
145         }
146         else
147             return true;
148     }
149
150     bool Serial::IsConnected()
151     {
152         //Simply return the connection status
153         return this->connected;
154     }
155
```

```
/*****************************************************************************
MLX90614_Serial_Demo.ino
Serial output example for the MLX90614 Infrared Thermometer

This example reads from the MLX90614 and prints out ambient and object
temperatures every half-second or so. Open the serial monitor and set the
baud rate to 9600.

Hardware Hookup (if you're not using the eval board):
MLX90614 ------------- Arduino
  VDD ------------------ 3.3V
  VSS ------------------ GND
  SDA ------------------ SDA (A4 on older boards)
  SCL ------------------ SCL (A5 on older boards)

An LED can be attached to pin 8 to monitor for any read errors.

Jim Lindblom @ SparkFun Electronics
October 23, 2015
https://github.com/sparkfun/SparkFun_MLX90614_Arduino_Library

Development environment specifics:
Arduino 1.6.5
SparkFun IR Thermometer Evaluation Board - MLX90614
*****************************************************************************/

#include <Wire.h> // I2C library, required for MLX90614
#include <SparkFunMLX90614.h> // SparkFunMLX90614 Arduino library

IRTherm therm1; // Create an IRTherm object to interact with throughout
IRTherm therm2;
IRTherm therm3;

const byte LED_PIN = 8; // Optional LED attached to pin 8 (active low)

void setup()
{
  Serial.begin(9600); // Initialize Serial to log output
  therm1.begin(0x0A); // Initialize thermal IR sensor
  therm2.begin(0x0B);
  therm3.begin(0x0C);
  therm1.setUnit(TEMP_C); // Set the library's units to Farenheit
  therm2.setUnit(TEMP_C);
  therm3.setUnit(TEMP_C);
  // Alternatively, TEMP_F can be replaced with TEMP_C for Celsius or
  // TEMP_K for Kelvin.
```

```
  pinMode(LED_BUILTIN, OUTPUT); // LED pin as output
  setLED(LOW); // LED OFF
}

void loop()
{
  digitalWrite(LED_BUILTIN, HIGH); //LED on

  // Call therm.read() to read object and ambient temperatures from the sensor.
  if (therm1.read()) // On success, read() will return 1, on fail 0.
  {
    // Use the object() and ambient() functions to grab the object and ambient
  // temperatures.
  // They'll be floats, calculated out to the unit you set with setUnit().
   Serial.print(String(therm1.object(), 2));
   Serial.print(",");
  }
  if(therm2.read()){

   Serial.print(String(therm2.object(), 2));
   Serial.print(",");
  }

  if(therm3.read()){

   Serial.print(String(therm3.object(), 2));
   Serial.print(",");
  }
  Serial.print("\n");
  digitalWrite(LED_BUILTIN, LOW);
  delay(100);
}

void setLED(bool on)
{
  if (on)
    digitalWrite(LED_PIN, LOW);
  else
    digitalWrite(LED_PIN, HIGH);
}
```