```cpp
/*
    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
```

```
    }
}

/*****************************************************************************
                            Real Time Graphics
*****************************************************************************/
RealTimeGraphics::RealTimeGraphics() : BaseGraphics(){
    this->length = 300;
    this->width = 100;

    this->outsideDataPoint = 0;
    this->middleDataPoint = 0;
    this->insideDataPoint = 0;

    this->tempLow = 0;
    this->tempHigh = 110;

    updateRectangle();
    updateGradient();

}

void RealTimeGraphics::setData(double value1, double value2, double value3){
    if(value1 > this->tempHigh){
        value1 = this->tempHigh;
    }
    if(value2 > this->tempHigh){
        value2 = this->tempHigh;
    }
    if(value3 > this->tempHigh){
        value3 = this->tempHigh;
    }
    if(value1 < this->tempLow){
        value1 = this->tempLow;
    }
    if(value2 < this->tempLow){
        value2 = this->tempLow;
    }
    if(value3 < this->tempLow){
        value3 = this->tempLow;
    }
    this->outsideDataPoint = value1 - this->tempLow;
    this->middleDataPoint = value2 - this->tempLow;
    this->insideDataPoint = value3 - this->tempLow;
    updateGradient();
    updateRectangle();
}

vector<double> RealTimeGraphics::calculateGradient(double point1, double point2, double
point3, int size){
    double slope1, slope2, slope3, slope4;
    double quaterRect = size / 4;
    double halfRect = size / 2;
    double threeQuaterRect = size *((double)(3) / (double)(4));
    vector<double> slopes;
    slope1 = (0 - point1) / (0 - quaterRect);
    slope2 = (point1 - point2) / (quaterRect - halfRect);
    slope3 = (point2 - point3) / (halfRect - threeQuaterRect);
    slope4 = (point3 - 0) / (threeQuaterRect - size);

    slopes.push_back(slope1);
    slopes.push_back(slope2);
    slopes.push_back(slope3);
    slopes.push_back(slope4);
    return(slopes);
}

void RealTimeGraphics::updateRectangle(void){
    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 
```