Lab11

Thu Pham

Profs. Topham

CS116

1 Description

- In this lab, we created a car parking program.
- We moved the car with a path and make sure that the cars will not run into each other.
- There are different types of wehicle such as sedan, motorcycle, truck and sports.
- So I have different drived class on base class Vehicle.
- The drive cb function is a recursive function and keeps calling the drive function that depends on the car id.
- The drive cb function also help with sopping and resetting.

2 The Problem

For this lab, let's bypass the help of ***FLUID*** and directly enter the ***FLTK*** code to draw things on the "canvas".

We use *inheritance* to derive a class from a **Fl_Double_Window**, then override the **draw** function (polymorphism).

Here are some cars we can park using a ***recursive*** algorithm: cars.zip

Create a timer using **Fl::add_timeout (example: image.png)** that will read in one "location" at-a-time from a string that had been created using our recursive function . These locations are actually numbers from 0.5 - 9.5 that represent a percentage of *where to place the car* on the street (just a line for now!) We only consider 0.5 to 9.5 since the cars are 1.0 and this is middle of car position. e.g

- suppose location is 3.2 (middle of the parking spot)
  divide by 10 to scale 10's to percentage 100's : .32 or 32%
  times 400 (pixel width of "street") to scale from 1 - 10 parking spaces
  minus 20 since x coordinate is left corner of the rectangle
  and each car is 40 units wide. (1/10 of 400)

The string with those numbers is created by a recursive function named "**parkingSequence**".

This function takes a **left** and **right** value to indicate the parking spaces range on street.

(there are no parking space "lines" so cars can parallel park anywhere there is room)

It generates a random number in that range to consider parking there.

Then recursively checks if there is more room for cars to left or to right

The recursion ends when there is no more room to fit a car (e.g. Car size of 1.0)

The function collects those spots where it found room into a string and returns it.

e.g. One possible output is this:

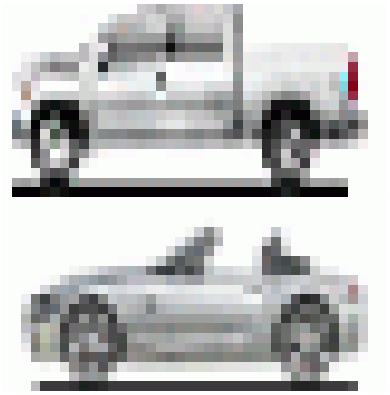
This time there was room for 8 cars.

*Submit your **pdf** with all src code and*

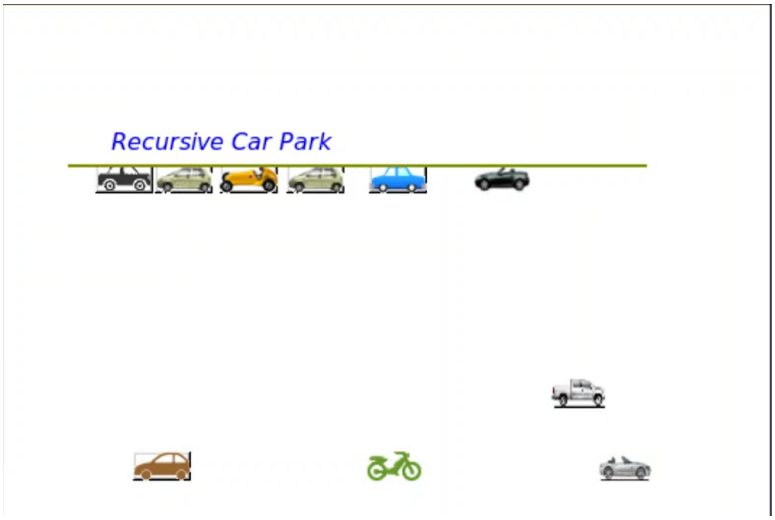
*images of several outputs and/or an mp4 **movie** and*

***tgz** with all files needed to build and run program.*

3 Photo of Cars



4 The Result



5 Source Code

5.1 Lab11b.cxx Source Code

```
1  #include <string>
2  #include <time.h>
3  #include "lab11b.h"
4  #include <unordered_set>
5  #include <bits/stdc++.h>
6  using namespace std;
7  Vehicle *cars[CARS];
8  vector<int> *carSelections;
9  int programRunTimes = 0;
10
11 void reset_all_car_position()
12 {
13    for (int i = 0; i < CARS; i++)
14    {
15       cars[i]->position(i * CARWIDTH + 50, 308);
16       cars[i]->setParked(false);
17    }
18 }
19 //randomly selected cars number from garage
20 void randomlyPick(int numberOfCarsParking)
21 {
22    unordered_set<int> selectedCars;
23    srand(time(NULL));
24    while (selectedCars.size() < numberOfCarsParking)
25    {
26       selectedCars.insert(random() % 10);
27    }
28    carSelections = new vector<int>(selectedCars.begin(), selectedCars.end());
29
30 }
31
32 void drive_cb(void *w)
```

```
33 {
34    static int i = 0;
35    Street *street = (Street *) w;
36 
37    int numOfCarsParking = street->getNumberOfParkingCars();
38 
39    cars[(*carSelections)[i]]->drive(street->getParkingLocationX(i)-10, street->getStreet_Y()+
20);
40 
41    if (cars[(*carSelections)[i]]->isParked())
42    {
43       i = (i + 1) % CARS;  // park next car
44    }
45    //let program runs three times
46    if (programRunTimes == 3)
47    {
48       //reset cars
49       Fl::remove_timeout(drive_cb, w);
50       return;
51    }
52    //terminate timer
53 
54    if (i == numOfCarsParking)
55    {
56    //let program run three times
57    //otherwise reset the cars position and i
58       reset_all_car_position();
59       randomlyPick(numOfCarsParking);
60       i = 0;
61       programRunTimes++;
62    }
63    street->redraw();
64 
65    Fl::repeat_timeout(0.1, drive_cb, w);
66 }
67 
68 int main()
69 {
70    cout << "Program␣Start" << endl;
71    Street *w = new Street(530, 355);
72    cout << "Successfully␣created␣Street" << endl;
73    //make window white
74    w->color((Fl_Color)55);
75 
76    //picking cars for parking
77    randomlyPick(w->getNumberOfParkingCars());
78 
79    //loop 10 times and created diffenet types of cars
80    for (int i = 0; i < CARS; i++)
81    {
82       switch (i)
83       {
84       case 0:
```

```
85          case 1:
86          case 2:
87          case 4:
88          case 6:
89             cars[i] = new Sedan(i * CARWIDTH + 50, 308, CARWIDTH, CARHEIGHT, i);
90             break;
91          case 3:
92          case 7:
93          case 9:
94             cars[i] = new Sports(i * CARWIDTH + 50, 308, CARWIDTH, CARHEIGHT, i);
95             break;
96          case 5:
97             cars[i] = new Motorcycle(i * CARWIDTH + 50, 308, CARWIDTH, CARHEIGHT, i);
98             break;
99          case 8:
100            cars[i] = new Truck(i * CARWIDTH + 50, 308, CARWIDTH, CARHEIGHT, i);
101            break;
102         }
103
104      cars[i]->box(FL_UP_FRAME);
105      string imageFile = "car" + to_string(i) + ".png";
106      cars[i]->image(new Fl_PNG_Image(imageFile.c_str()));
107
108      }
109   w->end();
110
111   w->show();
112
113
114
115   Fl::add_timeout(0.1, drive_cb, w);
116   return Fl::run();
117   }
```

5.2 Lab11b.h Source Code

```
1  #ifndef LAB11B_H
2  #define LAB11B_H
3
4  #include <FL/Fl.H>
5  #include "street.h"
6  #include "vehicle.h"
7  #include "sedan.h"
8  #include "truck.h"
9  #include "sports.h"
10 #include "motor.h"
11
12 const int CARWIDTH = 40;
13 const int CARHEIGHT = 20;
14
15 extern Vehicle *car[CARS];
16
17 #endif
```

5.3 Street.cpp Source Code

```
#include "street.h"

#include <string>
#include <ctime>
#include <vector>

using namespace std;
//helper function prototype
double fRand(double fMin, double fMax);
string parkingSequence(double streetLeft, double streetRight);
static void Timer_CB(void *parkingCars);

Street::Street(int width, int height)
: Fl_Double_Window(width, height), parkingCarsLocation(),street_x(45), street_y(130), street_w(40
street_h(2)
{
   string parkingSpaces = parkingSequence(0.5, 9.5);
   string delimiter = "␣";
   int counter = 0;
   while ((counter = parkingSpaces.find(delimiter)) != string::npos)
   {

      string a = parkingSpaces.substr(0, counter);
      parkingCarsLocation.push_back(stod(a));
      parkingSpaces.erase(0, counter + delimiter.length());
   }
   if (parkingSpaces.length() > 0)
   {
      parkingCarsLocation.push_back(stod(parkingSpaces));
   }
}

void Street::draw()
{
   Fl_Double_Window::draw();
   int x, y;
   //display title
   fl_color(FL_BLUE);
   fl_font(FL_ITALIC, 16);
   std::string s = "Recursive␣Car␣Park";
   x = 1 * W_WIDTH / 8;
   y = 1 * W_HEIGHT / 5;
   fl_draw(s.c_str(), x, y);
   //display street

   fl_color(FL_DARK_YELLOW);
   fl_rectf(street_x, street_y + 20, street_w, street_h);
}

int Street::getStreet_Y()
{
   return street_y;
```

```
52
53 }
54
55 double Street::getParkingLocationX(int carNumber)
56 {
57    return parkingCarsLocation[carNumber] / CARS * street_w;
58 }
59
60 int Street::getNumberOfParkingCars()
61 {
62    return parkingCarsLocation.size();
63 }
64
65 //recursive helper function
66 string parkingSequence(double streetLeft, double streetRight)
67 {
68    string carPos = ""; //to store the midpoint of where a car park
69    if (streetRight - streetLeft <= 0)
70    {
71       return carPos;
72    }
73    else
74    {
75       double streetLeft2, streetRight2;
76       double parkWhere = fRand(streetLeft, streetRight);
77       carPos = to_string(parkWhere);
78       streetLeft2 = parkWhere + 1;
79       streetRight2 = parkWhere - 1;
80
81       string result_left = parkingSequence(streetLeft, streetRight2);
82       string result_right = parkingSequence(streetLeft2, streetRight);
83       if (result_left.length() > 0)
84       {
85          carPos = result_left + "␣" + carPos;
86       }
87       if (result_right.length() > 0)
88       {
89          carPos = carPos + "␣" + result_right;
90       }
91
92       return carPos;
93    }
94 }
95 //to generate a random number in a certain range
96 double fRand(double fMin, double fMax)
97 {
98    srand(time(NULL));
99    double f = (double)rand() / RAND_MAX;
100   return fMin + f * (fMax - fMin);
101 }
```

5.4 Street.h Source Code

```
#include <string>
#include <vector>
#include <FL/Fl_Double_Window.H>
#include <FL/Fl_Box.H>
#include <FL/fl_draw.H>
#include <FL/Fl_PNG_Image.H>
using namespace std;

const int W_WIDTH = 600;
const int W_HEIGHT = 400;
const int CARS = 10;

class Street : public Fl_Double_Window
{
public:
   Street(int width, int height);

   virtual void draw();
   //This causes compiler to create a virtual function
   //table which is an array of function pointers
   //At runtime, the RTTI (RunTime Type Information) is
   //used to figured out which of the pointers in
   //the array to call based on type of object that
   //invoked it.

   int getStreet_Y();
   //maybe boolean to stop moving if there is another car
   double getParkingLocationX(int carNumber);
   int getNumberOfParkingCars();
private:
   vector<double> parkingCarsLocation;
   int street_x;
   int street_y;
   int street_w;
   int street_h;
};
```

6 Source Code of all derived class H file

6.1 sedan.h file

```
#ifndef SEDAN_H
#define SEDAN_H

#include "vehicle.h"
#include <cmath>

class Sedan : public Vehicle
{
public:
   Sedan(int x, int y, int w, int h, int id) : Vehicle(x, y, w, h, id) {}
   void drive(double destinationX, double destinationY)
```

```
12      {
13          int x1 = x();
14          int y1 = y();
15          if (y1 == static_cast<int>(destinationY) && x1 == static_cast<int>(destinationX))
16          {
17              setParked(true);
18              return;
19          }
20          //change y position
21          if (destinationY >= y1 - 6)
22          {
23              y1 = destinationY;
24          }
25          else
26          {
27              if (initialDrive)
28              {
29                  y1 -= 20;
30                  initialDrive = false;
31              }
32                  y1 -= 2;
33          }
34          //change x position
35          if (abs(destinationX - x1) <= 4)
36          {
37              x1 = destinationX;
38          }
39          else if (x1 > destinationX)
40          {
41              x1 -= 7;
42          }
43          else
44          x1 += 7;
45
46          position(x1, y1);
47      }
48
49  private:
50      bool initialDrive = true;
51  };
52  #endif
```

6.2 motor.h file

```
1   #ifndef MOTORCYCLE_H
2   #define MOTORCYCLE_H
3
4   #include "vehicle.h"
5   #include <cmath>
6
7   class Motorcycle : public Vehicle
8   {
9   public:
10      Motorcycle(int x, int y, int w, int h, int id) : Vehicle(x,y,w,h, id) {}
```

```
11      void drive(double destinationX, double destinationY)
12      {
13          int x1 = x();
14          int y1 = y();
15          if (y1 == static_cast<int>(destinationY) && x1 == static_cast<int>(destinationX))
16          {
17              setParked(true);
18              return;
19          }
20          //change y position
21          if (destinationY >= y1 - 10)
22          {
23              y1 = destinationY;
24          }
25          else
26          y1 -= 12;
27
28          //change x position
29          if (abs(destinationX - x1) <= 11)
30          {
31              x1 = destinationX;
32          }
33          else if (x1 > destinationX)
34          {
35              x1 -= 15;
36          }
37          else
38          x1 += 15;
39
40          position(x1, y1);
41      }
42
43  private:
44  };
45
46  #endif
```

6.3 sports.h file

```
1   #ifndef SPORTS_H
2   #define SPORTS_H
3
4   #include "vehicle.h"
5   #include <cmath>
6
7   class Sports : public Vehicle
8   {
9   public:
10      Sports(int x, int y, int w, int h, int id) : Vehicle(x, y, w, h, id) {}
11      void drive(double destinationX, double destinationY)
12      {
13          int x1 = x();
14          int y1 = y();
15          if (y1 == static_cast<int>(destinationY) && x1 == static_cast<int>(destinationX))
```

```
16      {
17         setParked(true);
18         return;
19      }
20      //change y position
21      if (destinationY >= y1 - 20)
22      {
23         y1 = destinationY;
24      }
25      else
26      {
27         if (initialDrive)
28         {
29            y1 -= 22;
30            initialDrive = false;
31         }
32         y1 -= 3;
33      }
34
35      //change x position
36      if (abs(destinationX - x1) <= 15)
37      {
38         x1 = destinationX;
39      }
40      else
41      {
42      if (y1 - 100 < destinationY)
43      {
44         if (x1 > destinationX)
45         {
46            x1 -= 15;
47         }
48      else
49         x1 += 15;
50      }
51      else
52
53      {
54         if (leftTurn)
55         {
56            x1 -= 15;
57            leftTurn = false;
58         }
59         else
60         {
61            x1 += 15;
62            leftTurn = true;
63         }
64      }
65   }
66
67   position(x1, y1);
68   }
```

```
69
70 private:
71    double x1;
72    double y1;
73    bool initialDrive = true;
74    bool leftTurn = true;
75 };
76
77 #endif
```

6.4 truck.h file

```
1  #ifndef TRUCK_H
2  #define TRUCK_H
3
4  #include "vehicle.h"
5  #include <cmath>
6
7  class Truck : public Vehicle
8  {
9  public:
10    Truck(int x, int y, int w, int h, int id) : Vehicle(x, y, w, h, id) {}
11
12    void drive(double destinationX, double destinationY)
13    {
14       int x1 = x();
15       int y1 = y();
16       if (y1 == static_cast<int>(destinationY) && x1 == static_cast<int>(destinationX))
17       {
18          setParked(true);
19          return;
20       }
21       //change y position
22       if (destinationY >= y1 - 6)
23       {
24          y1 = destinationY;
25       }
26       else
27       {
28          if (initialDrive)
29          {
30             y1 -= 20;
31             initialDrive = false;
32          }
33          y1 -= 5;
34       }
35
36       //change x position
37       if (abs(destinationX - x1) <= 6)
38       {
39          x1 = destinationX;
40       }
41       else if (x1 > destinationX)
42       {
```

```
43          x1 -= 5;
44       }
45       else
46          x1 += 5;
47
48    position(x1, y1);
49    }
50
51 private:
52
53    double x1;
54    double y1;
55    bool initialDrive = true;
56 };
57
58 #endif
```

7 Tex File

```
1  \input opmac
2  \input ../hisyntax
3  \tit Lab11
4  \maketoc
5  Thu Pham
6
7  Profs. Topham
8
9  CS116
10
11 \sec Description
12 \begitems
13 * In this lab, we created a car parking program.
14 * We moved the car with a path and make sure
15 that the cars will not run into each other.
16 * There are different types of wehicle such as sedan,
17  motorcycle, truck and sports.
18 * So I have different drived class on base class Vehicle.
19 * The drive cb function is a recursive function and keeps
20 calling the drive function that depends on the car id.
21 * The drive cb function also help with sopping and resetting.
22 \enditems
23
24 \sec The Problem
25
26 \picw=7in
27
28 \inspic lab11.png
29
30 \sec Photo of Cars
31 \picw=2in
32
33 \inspic car0.png
34 \filbreak
```

```
\picw=2in

\inspic car1.png
\filbreak
\picw=2in

\inspic car2.png
\filbreak
\picw=2in

\inspic car3.png
\filbreak
\picw=2in

\inspic car4.png
\filbreak
\picw=2in

\inspic car5.png
\filbreak
\picw=2in

\inspic car6.png
\filbreak
\picw=2in

\inspic car7.png
\filbreak
\picw=2in

\inspic car8.png
\filbreak
\picw=2in

\inspic car9.png

\filbreak
\sec The Result
\filbreak
\picw=4in

\inspic r.png


\sec Source Code

\secc Lab11b.cxx Source Code
\hisyntax{C}
\verbinput (-) lab11b.cxx

\filbreak
\secc Lab11b.h Source Code
\hisyntax{C}
```

```
\verbinput (-) lab11b.h

\filbreak
\secc Street.cpp Source Code
\hisyntax{C}
\verbinput (-) street.cpp

\filbreak
\secc Street.h Source Code
\hisyntax{C}
\verbinput (-) street.h

\sec Source Code of all derived class H file
\secc sedan.h file
\hisyntax{C}
\verbinput (-) sedan.h
\secc motor.h file
\hisyntax{C}
\verbinput (-) motor.h
\secc sports.h file
\hisyntax{C}
\verbinput (-) sports.h
\secc truck.h file
\hisyntax{C}
\verbinput (-) truck.h

\sec Tex File
\hisyntax{C}
\verbinput (-) lab11.tex

\bye
```