Specification

The program, **Car Designer Pro**, allows the user to design a car by choosing a brand, painting it, and adding upgrades to see what the cost and weight are. The program starts by asking the user to choose the car model they want to customize or see their last three creations.

After the car is chosen, the car will show up with a default yellow paint job and drive off screen to the “auto shop”. The user will then proceed to menu with buttons to paint the car. Then the user presses a button to choose the upgrade they want. The car will then drive out of the shop with the paint job that the user chose. The program will then display the characteristics of the vehicle: the weight, the color you chose, the brand, the optimizations, and the total cost of the car. The user will then have the option to select a button to choose another car to design, view their last three creations, or quit the program.

Specification (cont.)

If the user chooses to view their cars, a window opens that shows the last three cars in a neat list that also displays the weight of the car, the price of the car, and with the color and upgrades you chose for each car. As said, the window only displays the last three cars so say if a fourth car is created it displays that one at the top of the list replacing the first created. For example, if the user created two cars, the window would display these.

Car 1: Red Lamborghini, 3,600 kg, Engine Upgrade ⟶ $400,000

Car 2: Yellow Ferrari, 3,525 kg, Seat Upgrade ⟶ $375000

Car 3: Empty

Analysis

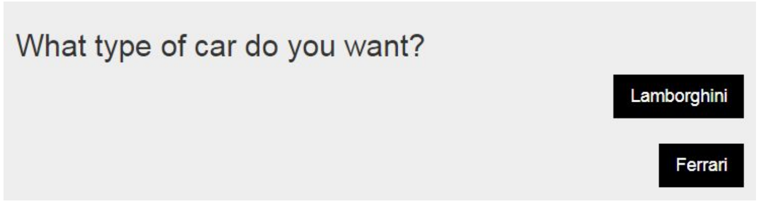
- Inputs:
    - External images of the same model cars in different colors
    - User must choose brand car they would like to design
    - User must choose a color they would like to “paint” the vehicle
    - User chooses any upgrade they would like for the vehicle
        - (e.g. engines, seats, transmission)
- Process:
    - PaintCar ⟶ "paints" the vehicle by swapping the image
    - AddCar ⟶ adds to the total costs and weight depending on user optimizations chosen
    - MoveCar ⟶ slides the image to and from the auto shop to imply motion

Analysis (cont.)

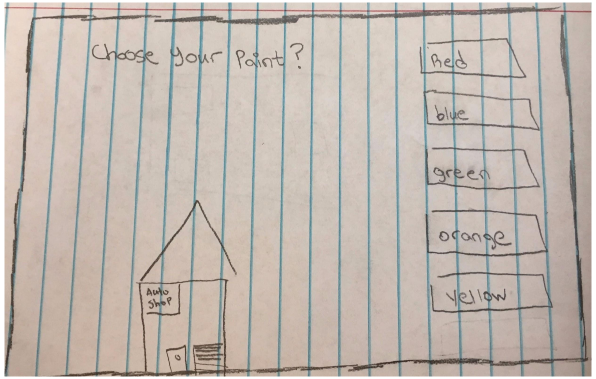
- Outputs:
  - Image of car with new color
  - Weight of the vehicle
  - Brand name of vehicle
  - Cost of vehicle
  - Upgrades to vehicle

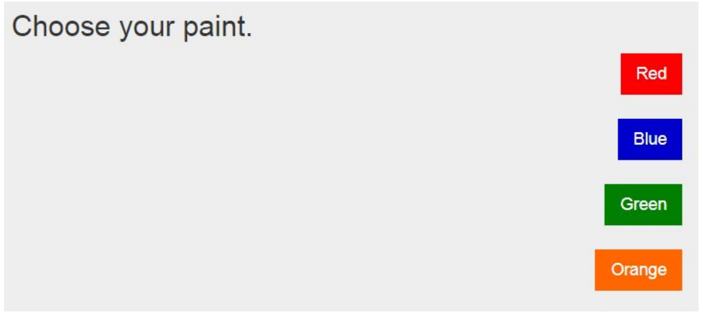
Analysis (cont.)

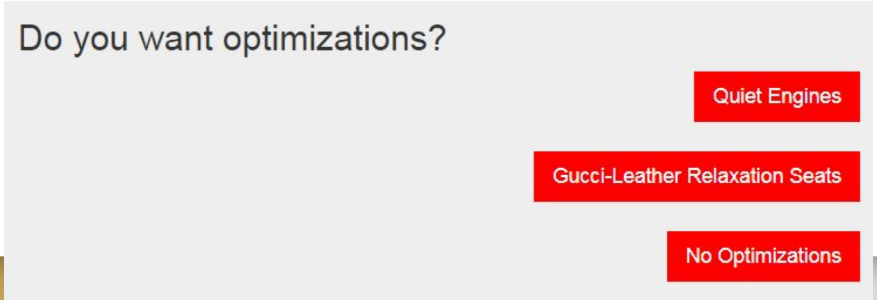
Once the program starts, the user will be prompted whether they would like to paint a ferrari and/or lamborghini model vehicle


What type of car do you want?

After the user selects the car model, the vehicle will enter the auto shop and the program will prompt the user to choose the color they would like to paint the vehicle. →




Choose your paint.


Do you want optimizations?

Design

car.cpp ⟶ Car class represents a car object. Defines functions that modify variables declared in car.h

car.h ⟶ Declares variables and functions that modify variables for car.cc

main.cpp ⟶ Functions create all the FLTK widgets used in the program, animate the vehicle movement, and modify values in an array that holds instances of the Car class

Function: main

- Creates window
- Calls InitButton to create buttons
- Calls InitGfx to create labels and graphics
- Runs the FLTK loop and returns the integer based on the return of the Fl::run function

Function: InitButton

- Sets Width/Height
- Aligns buttons
- Uses dynamic memory allocation to allocate buttons
- Sets Callbacks after button creation
- Hides unused buttons

Function: InitGfx

- Initializes labels
- Creates initial invisible car for animation
- Creates the outputs that appears near the end of the program and hides this initially

Function: CallbackCarSelect

- Sets default image to a yellow car, uses function AddCar to initialize the car type variable in the class instance that is stored an array
- Starts animation to move the vehicle off the screen into the auto shop

Function: MoveCarShop

- Creates delay timer that repeats itself until the car image reaches a certain number of pixels
  - While the car is moving, the label will change informing the user their car is being transported to the auto shop

Function: MenuSelectColor

- Hides the Ferrari and Lamborghini buttons replacing them with the paint color option buttons
- Changes label asking the user to choose what color they would like their vehicle modified

Function: CallbackColorSelect

These functions are going to take the car struct array and set the color of the car depending on the color the user chose using the PaintCar function then moves to the next menu using MenuSelectMod

Function: PaintCar

When the vehicle moves off the screen the image of the vehicle changes to the color the user selected from pressing the button

Function: MenuSelectMod

Hides the color buttons and instead changes them into modification buttons.

Changes label asking user to choose one modification they would like applied to their vehicle

Function: CallbackModSelect

Calls AddMod function which adds the mod to the vehicle and adds the cost and weight into the class member variables

Calls the animation function MoveCarDisplay which moves the car

Function: MoveCarDisplay

Animates the car off of the screen and back on using a delay timer

Changes the label prompting the user the modifications are complete

Opens new menu using MenuResult

Function: MenuResult

Hides unnecessary buttons and displays buttons prompting user to create a new car or view previously created vehicles, or quit the program.

Displays output boxes of the car the user created.

Label changes to let the user know their car is complete.

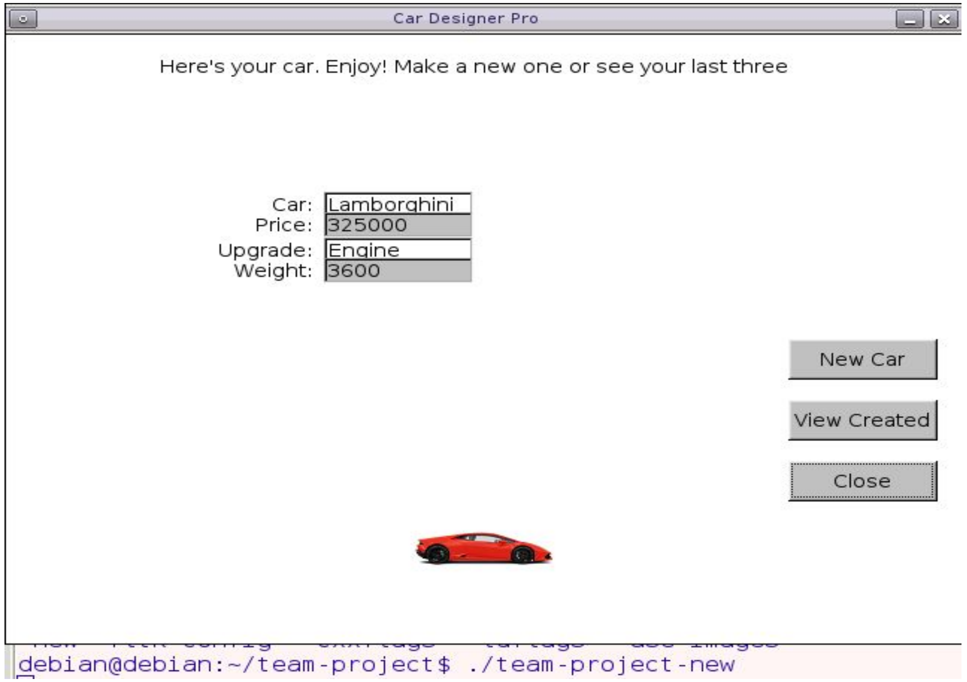
Function: CallbackView

Creates new window displaying the struct array elements for the last 3 vehicles the user creates. Also creates widgets to display.

Class: Car

Car is a class that represents a car. This class has variable members to represent many of the properties that you can want to know about a car (e.g. price, weight). When an instance of this class is created the object can be modified to symbolize any car that the programmer wishes.

Testcase 1: Lamborghini / Red / Engine

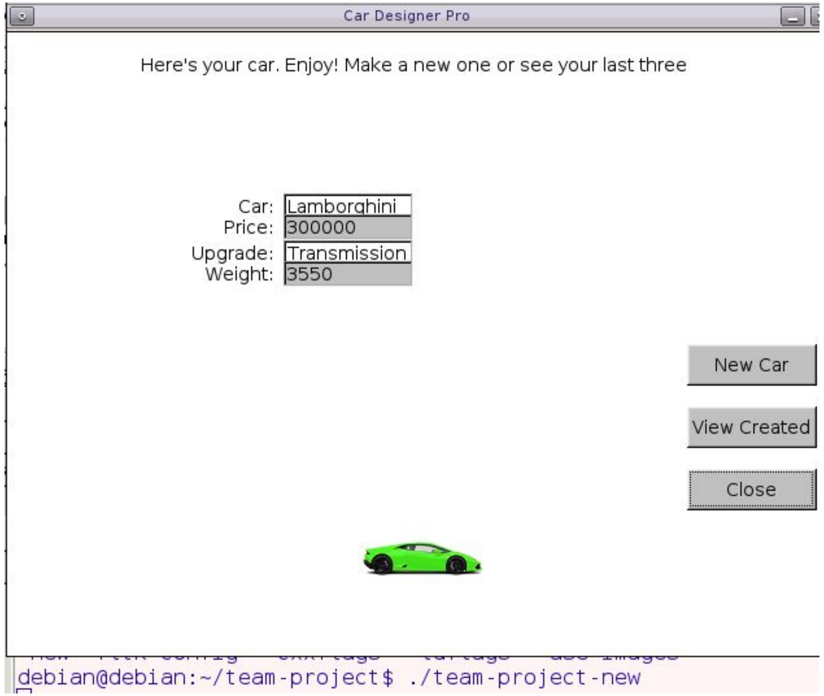

Car Designer Pro

Here's your car. Enjoy! Make a new one or see your last three

Car:	Lamborghini
Price:	325000
Upgrade:	Engine
Weight:	3600

```
debian@debian:~/team-project$ ./team-project-new
```

Testcase 2: Lamborghini / Green / Transmission

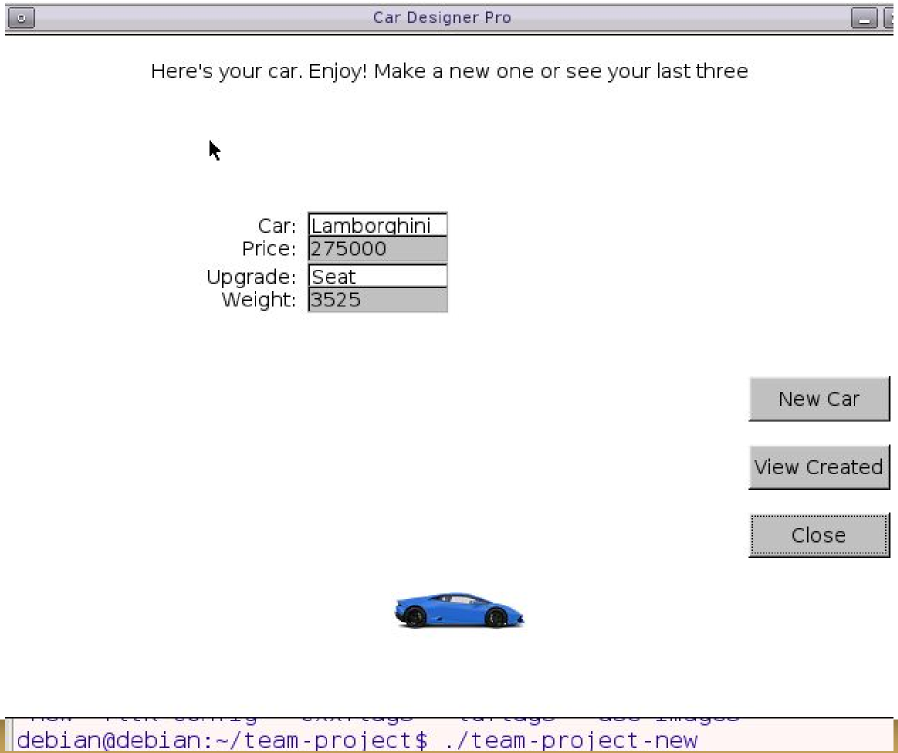


Here's your car. Enjoy! Make a new one or see your last three

Car:	Lamborghini
Price:	300000
Upgrade:	Transmission
Weight:	3550

```
debian@debian:~/team-project$ ./team-project-new
```

Testcase 3: Lamborghini / Blue / Seat

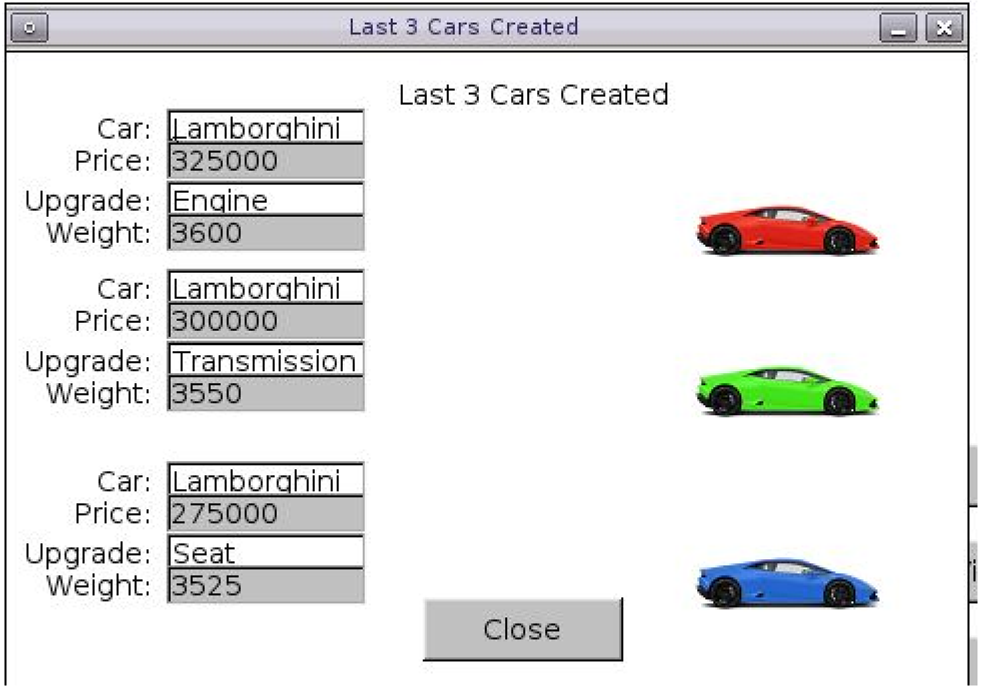


Here's your car. Enjoy! Make a new one or see your last three

Car:	Lamborghini
Price:	275000
Upgrade:	Seat
Weight:	3525

```
debian@debian:~/team-project$ ./team-project-new
```

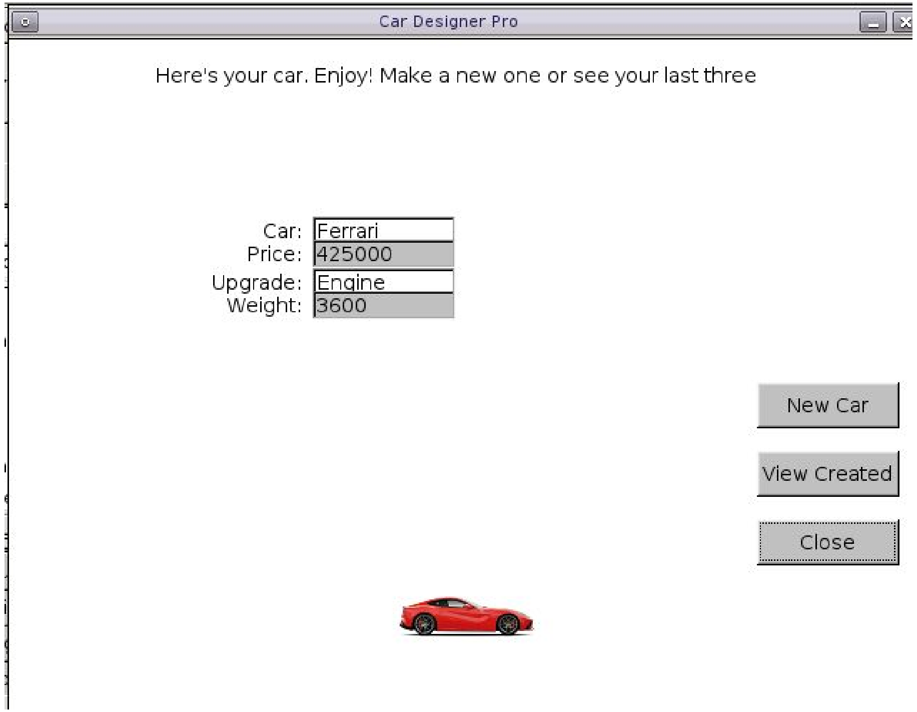
Testcase 4: View Last 3 Created



Last 3 Cars Created

Car:	Lamborghini
Price:	325000
Upgrade:	Engine
Weight:	3600
Car:	Lamborghini
Price:	300000
Upgrade:	Transmission
Weight:	3550
Car:	Lamborghini
Price:	275000
Upgrade:	Seat
Weight:	3525

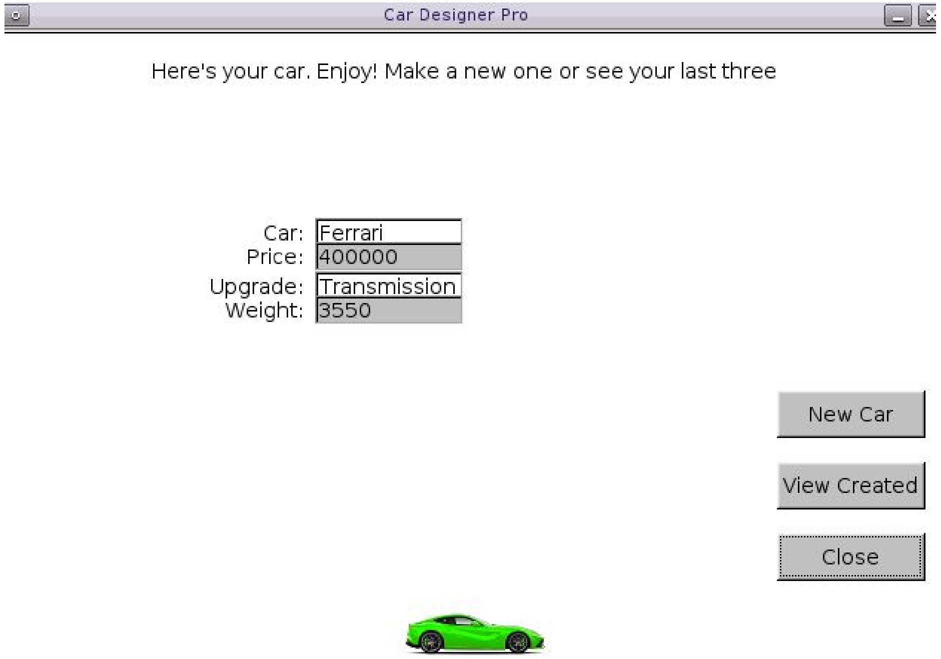
Testcase 5: Ferrari / Red / Engine



Here's your car. Enjoy! Make a new one or see your last three

Car:	Ferrari
Price:	425000
Upgrade:	Engine
Weight:	3600

Testcase 6: Ferrari / Green / Transmission

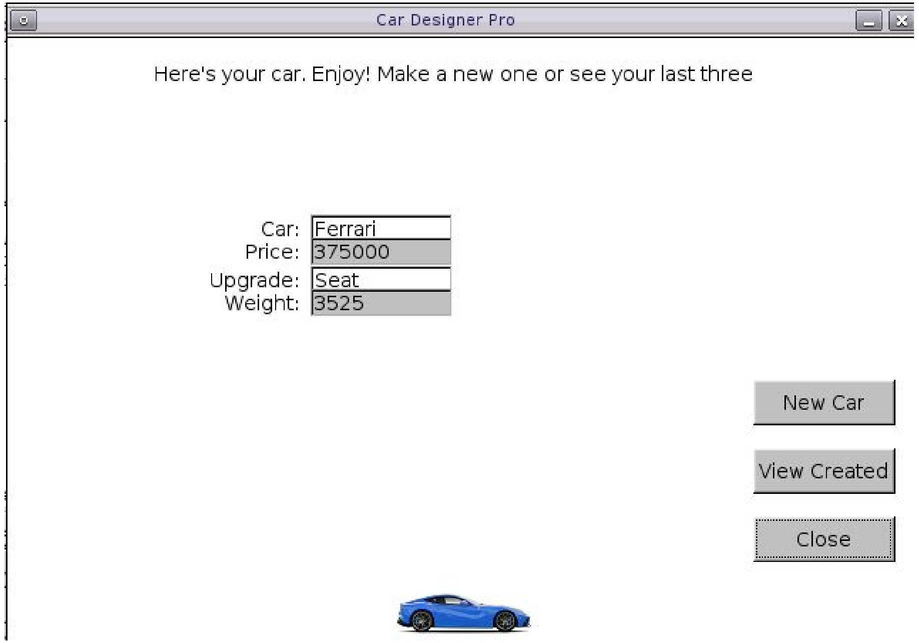


Car Designer Pro

Here's your car. Enjoy! Make a new one or see your last three

Car: Ferrari
Price: 400000
Upgrade: Transmission
Weight: 3550

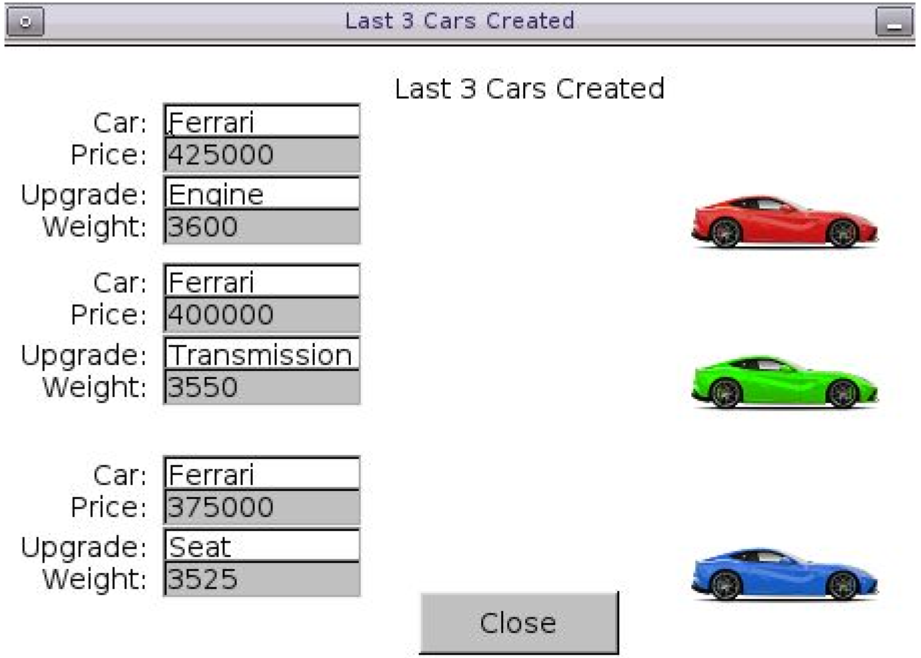
Testcase 7: Ferrari / Blue / Seat



Here's your car. Enjoy! Make a new one or see your last three

Car:	Ferrari
Price:	375000
Upgrade:	Seat
Weight:	3525

Testcase 8: View Last 3 Created



Last 3 Cars Created

Car:	Ferrari
Price:	425000
Upgrade:	Engine
Weight:	3600
Car:	Ferrari
Price:	400000
Upgrade:	Transmission
Weight:	3550
Car:	Ferrari
Price:	375000
Upgrade:	Seat
Weight:	3525