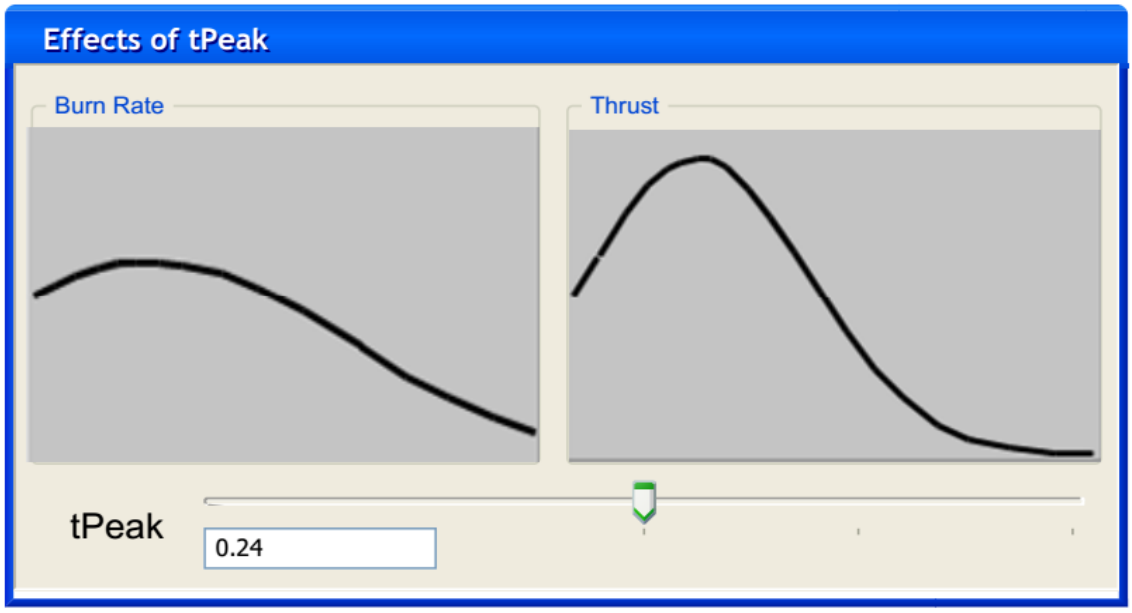
Practical Object oriented design (Mediator & Command)

Solve the following problems.

Problem1: Turbo Engine Ignition

You will need now to build for Audi an application (like the one shown in the figure below) that will give the possibility to the turbo ignition engine engineers to experiment visually with parameters that determine the relationships between the car thrust and the surface area of the ignited fuel.



In the new turbo engine that is under development when the engine ignites, the part of its fuel that is exposed to air burns producing thrust. From ignition to maximum burn rate, the burn area increases from the initial ignition area to the full surface area of the fuel. This maximum rate occurs at time tPeak . As fuel burns off, the surface area reduces again until the fuel is consumed. Suppose that the burn rate and thrust equations are:

$$rate = 25^{-(t-t_{peak})^2} \quad thrust = 1.7\left(\frac{rate}{0.6}\right)^{\frac{1}{0.3}}$$

The application from the figure above shows how tPeak affects the burn rate and thrust of a car. As a user moves the slider, the value of tPeak changes, and the curves take on new shapes. The value for tPeak can be also entered numerically in the edit box below the slider. Design the application for above requirements?

Practical Object oriented design (Mediator & Command)

Problem2: Text Editor

Design a simple text editor that support only one open document(for an example, you can represent it as stringbuffer) and operations like insert, delete, cut, paste and copy operations. The editor must also support infinite undo for all the operations except copy operation. You might need to create singleton Clipboard object for cut and paste operations. The clipboard object needs a method to load a string and a method to get a copy of the string it holds. Here are the details of operations that need to be supported:

InsertText: Insert operation allows the user to insert text in a document. It will take cursor position and text as input parameters and inserts the text at cursor position inside document. For example, if the document contained the text "012345" and the parameters were 2 and "abc" the result should be "012abc345". Be careful about undo; the text I'm inserting may also be elsewhere in the buffer, so be sure you delete it from the same place it was inserted.

DeleteText: Delete operation allows the user to select a region of the text and delete it. It will take two integers which are the positions of the first and last characters to be deleted (assuming the positions are numbered from zero). For example, if my document contains "0123456789" and the parameters to DeleteText are 2 and 4, the document should end up containing "0156789". Be careful about undo to be sure that things get put back together correctly.

Cut: Cut operation allows the user to cut a selected region of text in a document and place it in the clipboard. It's parameters and its effect on the document are the same as DeleteText, but it also loads the text into the Clipboard so we can use it later.

Paste: Paste operation allows the user to get stuff out of the clipboard onto document. It's parameters and its effect on the document are the same as InsertText, but it should get the string to be inserted from the clipboard.

Copy: Copy operation allows the user to select a region of text and paste it another place. Like in most real systems there is nothing to undo for this operation.

Problem3: Race Replay

Suppose we have a game that allows us to race with a car. The user can perform the following actions: go faster, go slower, go left and go right. Provide an object oriented design for adding functionality to record a whole race such that we can replay the recorded race.

Practical Object oriented design (Mediator & Command)

Problem4: War games

Imagine a war zone where armed units are moving into the enemy's territory. Armed units can include soldier, tank, grenadier, and sniper units. The strategy being employed is that whenever one unit attacks, other units should stop attacking and take cover. To do so, the unit that is currently attacking needs to notify the other units. Provide object oriented design to implement the above functionality.