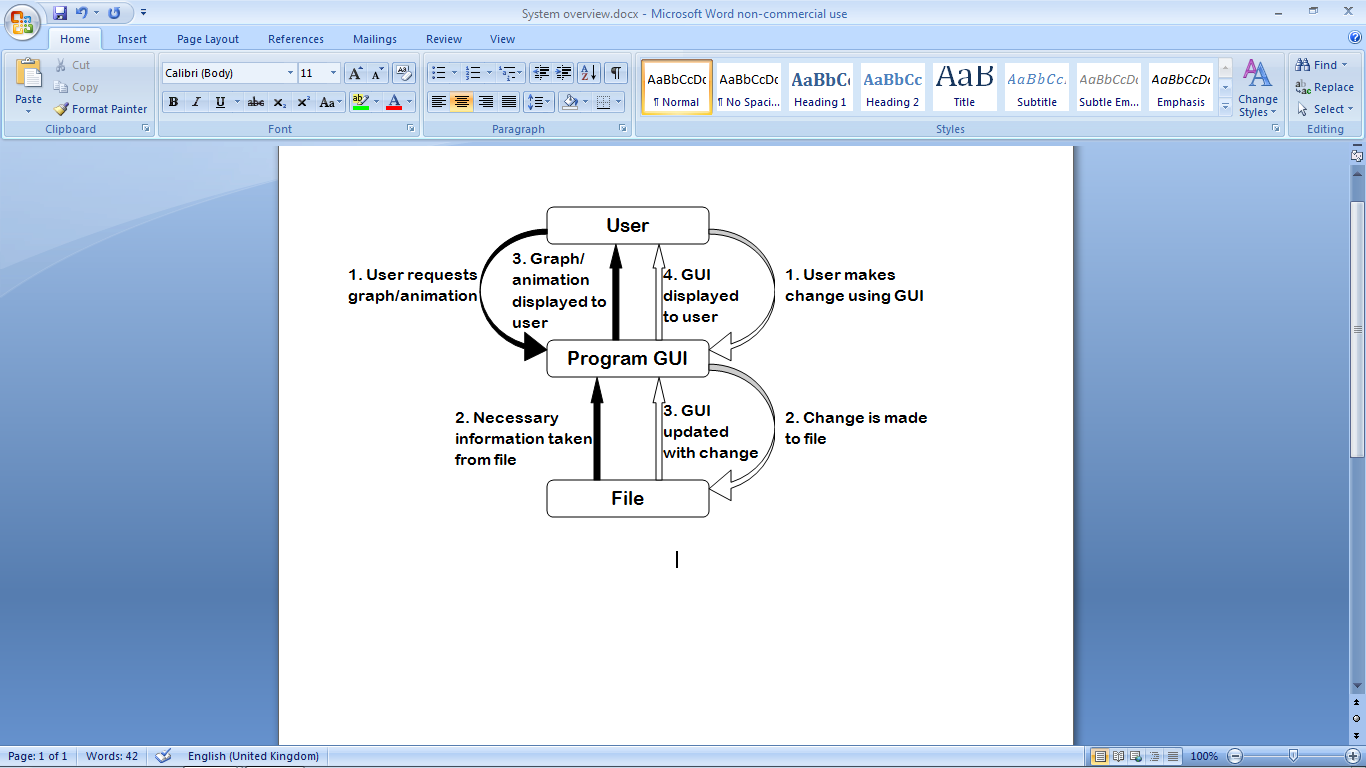
2. Design

2.1: System overview

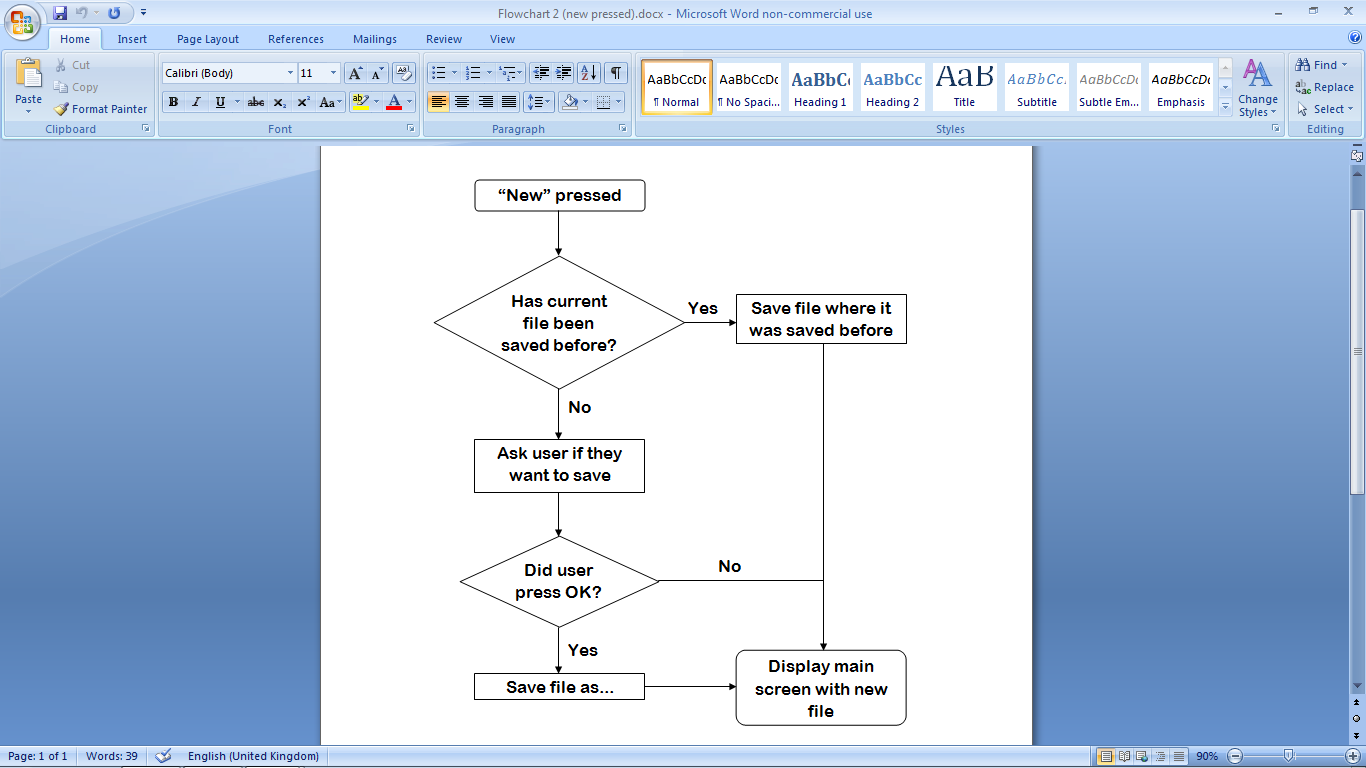
The design of the system will be an object-oriented, GUI-based application written in the Python programming language, using the PyQt application framework. The interface will be used to manipulate the current file’s object information (the conditions of the reactions). Whenever this information is manipulated, the changes will be  
reflected on the interface. The  
interface will also be used to produce graphs and animation, using some of the current file’s information. The relationship between the user, file and interface is demonstrated in figure 2.1. The program will be centred on a time-driven simulation of a reversible reaction, represented by the animation. The simulation should not be event-driven, because no events occur during the reaction. The main system processes are outlined below:

Figure 2.1: The entities which interact in the system at the top level.

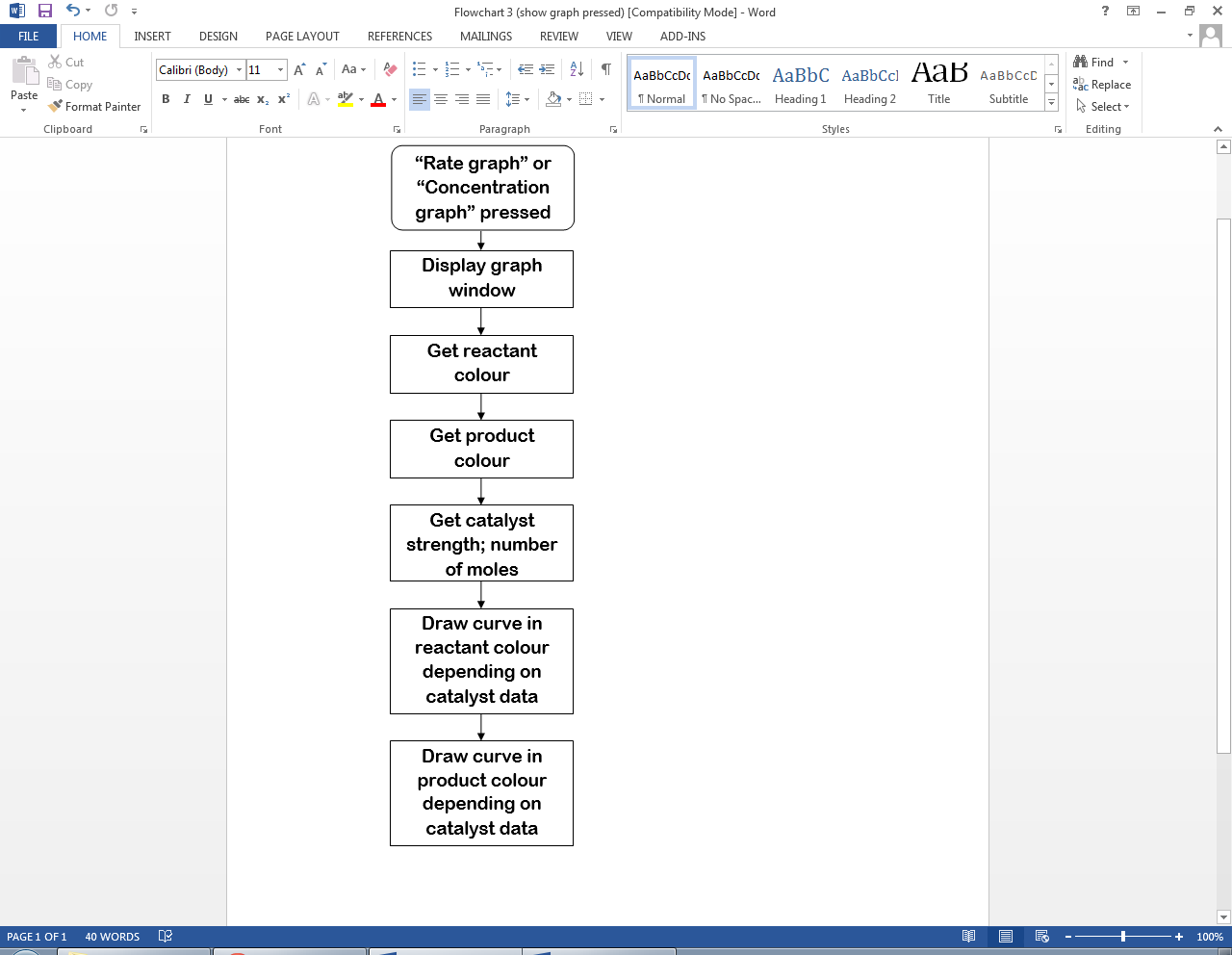
Program startup



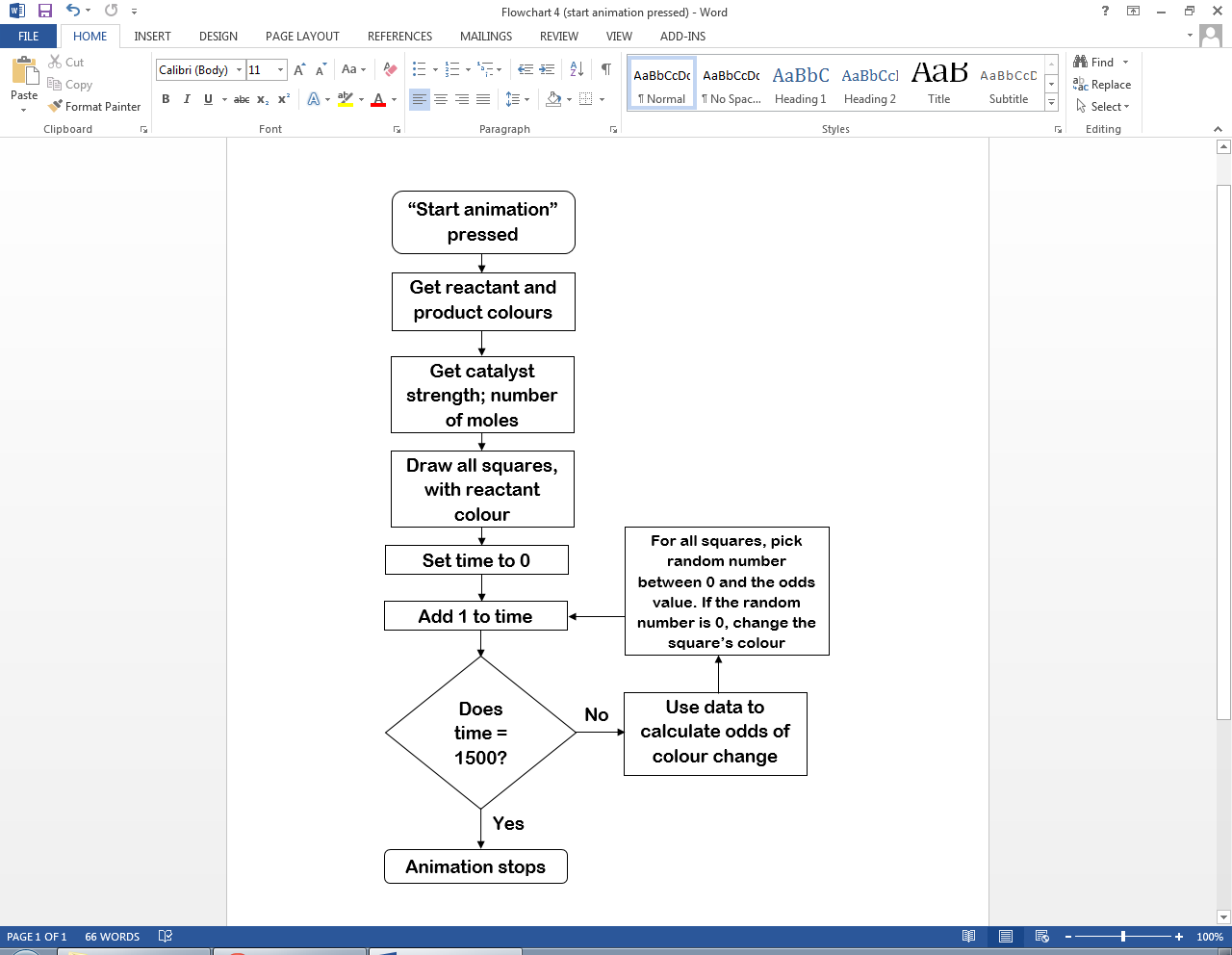
Opening a new file



Showing a graph



Starting an animation



2.2: Modular structure

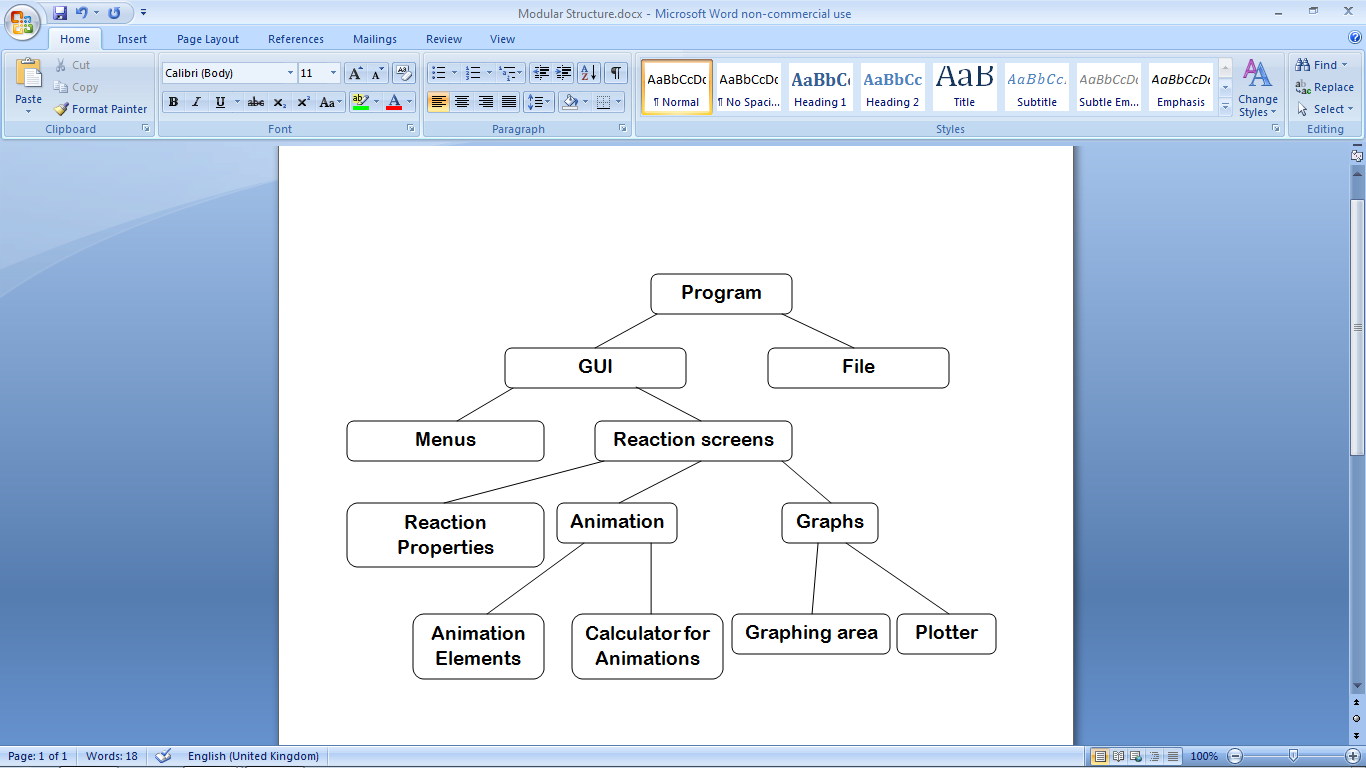
The modules making up the proposed system should group together as in figure 2.2.

Figure 2.2: The modular structure of the proposed system.

The modules “Calculator for Animations” and “Plotter” operate by reading data from the current file, and use the data to control “Animation Elements and the “Graphing Area” respectively. The “Reaction Properties” module reads data from, and writes data to, the file. Below is the hierarchy of the menu module.

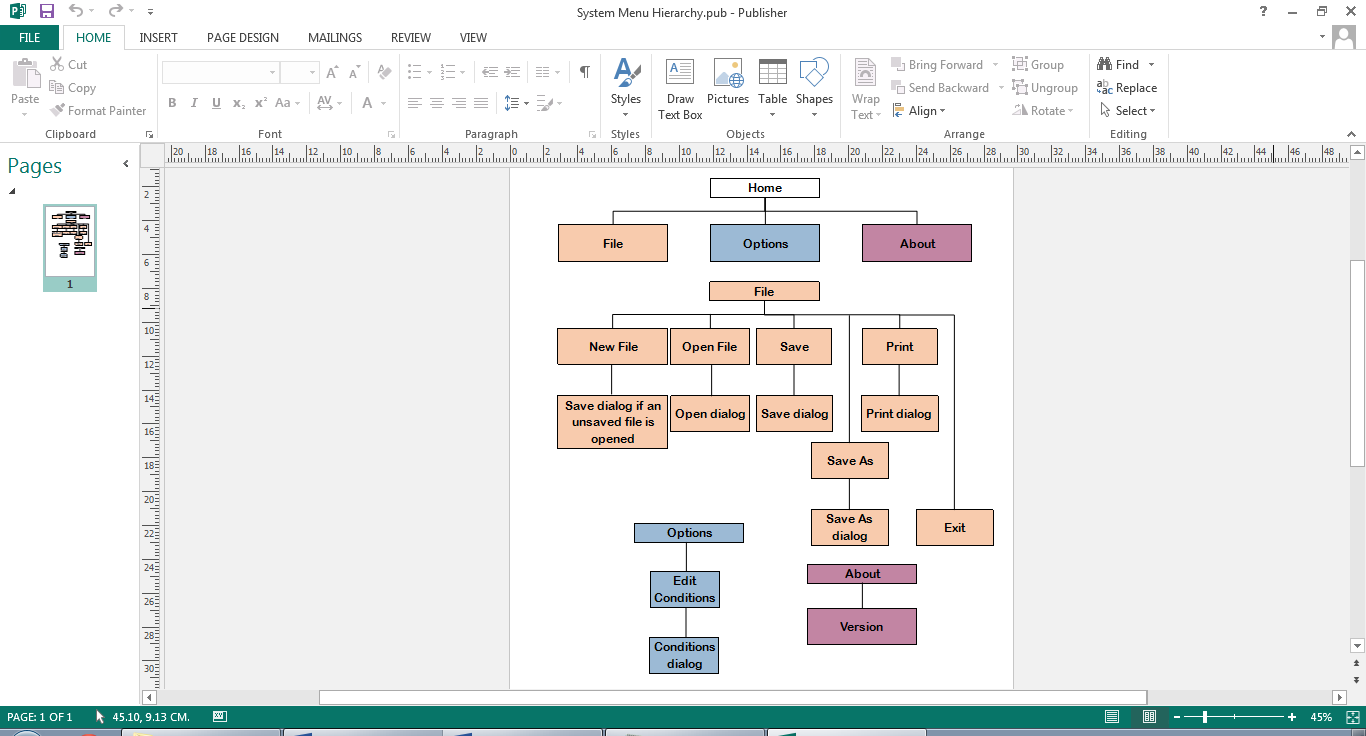


Figure 2.3: The menu hierarchy of the proposed system.

2.3: Data dictionary of proposed system

2.3.1: Variables to be used

PyQt’s text input system only supports plain text, so to enter subscript and superscript text, users must enter \_ and ^, as exemplified by the data dictionary.

Long-term stored properties

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Field Purpose | Field Type | Maximum Field Size | Example Data | Validation |
| Formula | Identifies a reactant, catalyst or product by the number of atoms of each element | String | 25 charact-ers | “H\_2O\_2” | Validated by regular expression |
| Moles | The amount of a substance, measured in moles, when the reaction is at equilibrium | Float | 4 digits, including 2 decimal places | 3.25 | Must be between 0.00 and 100.00 exclusive |
| Volume | The amount of space the system takes up, measured in cubic decimetres (dm3). One mole of gas occupies 24 dm3. | Float | 4 digits, including 2 decimal places | 18.50 | Must be between 00.00 and 100.00 exclusive |
| Temperature | The average kinetic energy of the particles in the system, measured in Kelvin | Integer | 3 digits | 273 | Must be more than 000 and less than 1000 |
| Endothermic | Describes whether, during production of products, energy is taken in overall | Boolean | - | True | - |
| Stoichiometric Ratio | The ratio of a molecule to other molecules required for them to react | Integer | 1 digit | 3 | Must be between 1 and 9 inclusive |
| Efficacy | Factor by which the catalyst alters reaction rate | Float | 10 | 4 | - |

Short-term local variables

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Purpose | Type | Maximum size | Example data | Validation |
| reactants | All reactants selected to be used in the reaction | List (Reactant) | 3 items | - | Valid if all data in each used reactant object are valid |
| products | All products selected to be used in the reaction | List (Product) | 3 items | - | Valid if all data in each used product object are valid |
| rvalues | All numerical values to do with the reactants. Items with even indices are concentrat-ions; items with odd indices are stoichiomet-ric values | List (float) | 6 items | [14.5, 3.00, 24.24, 5.00, 22.25, 4.00] | Valid if the formula and number of moles of each used reactant, and the reaction volume, are valid |
| pvalues | All numerical values to do with the products. Items with even indices are concentrat-ions; items with odd indices are stoichiomet-ric values | List (float) | 6 items | [0.4, 2.00, 1.6, 8.00, 1.8, 9.00] | Valid if the formula and number of moles of each used product, and the reaction volume, are valid |
| kcvalue | Shows the calculation needed to obtain a value for Kc. | String | 70 characte-rs | “Value of Kc = (0.4)^2 (1.6)^8 (1.8)^9 / (14.5)^3 (24.24)^5 (22.25)^4” | Valid if all items in rvalues and pvalues are valid |
| concentrat-ion | Amount of a reactant, product or catalyst per unit volume, measured in moles per dm3 | Float | 4 digits, including 2 decimal places | 10.04 | Valid if number of moles and reaction volume are valid |
| rproducts-um | Total of the stoichiomet-ric values of each used reactant | Integer | 27 | 14 | Valid if all odd-numbered items in rvalues are valid |
| pproducts-um | Total of the stoichiomet-ric values of each used product | Integer | 27 | 8 | Valid if all odd-numbered items in pvalues are valid |
| kc | Equilibrium constant of a reversible reaction at a constant temperature | Float | Between 1010 and 10-10 | 0.24 | Valid if concentrat-ions of products and reactants are valid |
| numerator | Product of concentrat-ions of each product to the power of its stoichiomet-ric value | Float | (99.999)3 = 9.97e+53 | 242.88 | Valid if all values in pvalues are valid |
| denominat-or | Product of concentrate-ions of each reactant to the power of its stoichiomet-ric value | Float | 9.97e+53 | 359.66 | Valid if all values in rvalues are valid |
| value | Value of Kc, without units | Float | (99.999)3 / (0.019)3 = 9.97e+107 (reversib-le reactions actually never have values this high) | 20.38 | Valid if numerator and denominat-or are valid |
| power | The power of the moles units. This multiplied by -3 is the power of the volume units. The result of pproducts-um minus rproducts-um | Integer | 2 | -1 | Valid if pproducts-um and rproducts-um are valid |
| newformula | HTML version of formula string, using tags instead of special characters | String | 105 characters | “CaCl <sub>2 </sub>” | Valid if formula passed in is valid |
| change | Factor that the graph changes by | Float | 10 | 0.5 | Valid if all catalyst data is valid |
| eqmpoint (in DrawCurve method) | X coordinate where the graph becomes level | Integer | 300 | 145 | Valid if less than 301 and more than 0 |
| newSquare | The next Square object to be added to the animation’s array of them | Square | - | - | All Square objects instantiated in the algorithm so must be valid |
| eqmpoint (in PlayAnimat-ion method) | Number of updates until chance of a colour change is equal for both colours | Integer | 750 | 624 | Valid if less than 751 and more than 0 |
| time | Number of updates since the animation was started | Integer | 1499 | 345 | Valid if less than 1500 and more than 0 |
| chance | Odds of a square changing colour (1 in 5 if chance = 5) | Integer | 200 | 130 | Valid if more than 0 and less than 201 |

2.3.2: Input validation checks

Checks need to be performed for the values in the first table of section 2.3.1. If these values are valid, the values in the second table will also be valid. If values are not selected from a list of valid values, they are entered as strings in text boxes, so instead of checking the data type entered, the string is validated and converted into the appropriate data type off-screen.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Field Name** | **Validation Checks** | **Description** | **Error message** | | Formula | If used: Presence; Length; Regex | Must be 1-25 characters long and fit the regex | “The formula of [chemical] must be 1-25 characters long inclusive.”/”The formula of [chemical] is invalid.” | | Moles | If used: Presence; Range; Regex | Must be between 0.01 and 99.99 inclusive and fit the regex | “The number of moles of [chemical] must be between 0.01 and 99.99 inclusive, with up to 2 decimal places.” | | Volume | Presence; Range; Regex | Must be between 0.01 and 99.99 inclusive and fit the regex | “The reaction volume must be between 0.01 and 99.99 inclusive, with up to 2 decimal places.” | | Temperature | Presence; Range | Must be between 1 and 999 inclusive | “The temperature must be a whole number between 1 and 999 inclusive.” | | Endothermic | Lookup(Endothermic, Exothermic; default setting is Endothermic) | If Endothermic is selected, set to true. If not, set to false | - | | Efficacy | Lookup(Good, Poor, Inhibitor; default setting is Good) | If used: if Good selected, Efficacy is set to 10; Poor, set to 4; Inhibitor, set to 0.5. If not used, set to 1 | - | | Stoichiometric Ratio (when entering conditions) | Range | Must be between 2 and 9. If not entered, set to 1. If larger than 9, Formula will be invalid anyway | - | | Stoichiometric Ratio (when attempting to balance equation) | Lookup (1-9; default value is 1) | - | - | |  |
|  |  |
|  |  |
|  |  |
|  |  |

2.4: Program and data storage; file organisation; system specification

The system will be compiled as a standalone binary application, installed by an installer file which will be distributed via mastered DVD-ROM. This is because the distribution medium should only have one purpose, which is to provide the installer that needs to be run. Being mastered means the installer cannot be edited, and malicious files cannot be added. A DVD-ROM is useful because they have a relatively small amount of memory, which suits the relatively small size of the installer. The DVD-ROM is also reusable, in case there is an error with the installation. The end user specifies an installation directory on their system when they run the installer.

Each file will consist of a list of reaction objects, and an integer index of the last reaction in the file that had been viewed. Each file will be saved using a binary dump with serial storage provided by Python’s pickle module, as they are not required to be read by any other programs. No images or videos will be saved, so file sizes should not exceed 5 KB. The user will be able to choose where to save files via a “Save As...” dialog.

The resulting standalone application will be compatible with any computer which runs a Linux, Windows NT (or later) or Mac OSX (or later) operating system. Two applications will be developed; one compatible with 32-bit systems and the other with 64-bit systems. The only peripherals required will be a standard keyboard, a standard mouse and a display unit which supports resolutions of at least 1024x768. All school computers fit these criteria, as they are mostly 32-bit systems, and all of them have monitors supporting 1024x768 resolution and run on Windows 7 Professional edition. The program should require no more than 128MB of disk space for storage and 50MB of RAM while running – all school computers easily meet this requirement.

2.5: Proposed algorithms

Calculating Kc

The ShowWorking() function is in the Reaction class and uses its properties. It ouputs a worked calculation of the value and units of the given reaction’s equilibrium constant. The process is as follows:

1. The reactants and products lists are initialized as empty.
2. For each reactant and product in the reaction, the algorithm checks if they are used. If so they are appended to the corresponding list. From here on “reactants” and “products” only refers to the ones in these lists.
3. The formula for concentration is printed.
4. The reaction’s volume is printed.
5. The pvalues and rvalues lists are initialized as empty.
6. For each product in turn, its concentration is printed, its concentration is appended to pvalues and its stoichiometric value is appended to pvalues.
7. For each reactant in turn, its concentration is printed, its concentration is appended to rvalues and its stoichiometric value is appended to rvalues.
8. A string kcvalue is initialized as “Value of Kc =” and a stepper x is initialized as 0.
9. Using the stepper and the two lists, the kcvalue string gradually has the whole of the right-hand side of the equation appended to it; so that it resembles the string “Value of Kc = (<concentration)^<stoichiometric value> / (<concentration)^<stoichiometric value>”
10. The total of the stoichiometric ratios for each product is calculated.
11. The total of the stoichiometric ratios for each reactant is calculated.
12. Concentration is measured in mol dm-3. The overall units are found by dividing the product of the units of concentration of the products by the product of the units of concentration of the reactants. In the algorithm, the power of the moles is the total of the ratios of the products minus that of the reactants. The power of the volume is this multiplied by -3. Calculating the units the second way yields the same result and is more efficient. The calculation of the units is printed out.
13. The algorithm now calculates Kc itself. For each product, the algorithm takes its concentration and multiplies it to the power of its stoichiometric ratio. The Numerator variable is the product of all these values. The Denominator is the result of the same process with each reactant. The units are calculated the same way as earlier. The value and units of Kc are then printed out.

The end user may sometimes need to display working with the answer, and sometimes only the answer, so this function may be split into a ShowWorking() and a GetKc() function. In this case, the GetKc() function is independent and ShowWorking() would call GetKc().

FUNCTION ShowWorking

reactants 🡨 []

products 🡨 []

FOR x 🡨 0 TO self.Reactants.length - 1 DO

IF self.Reactants[x].GetUsed() THEN

reactants.append(self.Reactants[x])

ENDIF

ENDFOR

FOR x 🡨 0 TO self.Products.length - 1 DO

IF self.Products[x].GetUsed() THEN

products.append(self.Products[x])

ENDIF

ENDFOR

OUTPUT "Concentration = moles / volume"

OUTPUT "Volume = "+str(self.Volume)+" dm^3"

pvalues 🡨 []

rvalues 🡨 []

FOR x 🡨 0 TO products.length – 1 DO

OUTPUT "Concentration of " + products[x].GetFormula() + " = " + str(products[x].GetConcentration(self.Volume))+" mol dm^-3"

pvalues.append(products[x].GetConcentration(self.Volume))

pvalues.append(products[x].GetSRatio())

ENDFOR

FOR x 🡨 0 TO reactants.length – 1 DO

OUTPUT “Concentration of " + reactants[x].GetFormula()+" = "+str(reactants[x].GetConcentration(self.Volume))+" mol dm^-3"

rvalues.append(reactants[x].GetConcentration(self.Volume))

rvalues.append(reactants[x].GetSRatio())

ENDFOR

kcvalue 🡨 "Value of Kc ="

x 🡨 0

WHILE x < pvalues.length DO

kcvalue 🡨 kcvalue + " ("+str(pvalues[x])+")^"

x 🡨 x + 1

kcvalue 🡨 str(pvalues[x])

x 🡨 x + 1

ENDWHILE

kcvalue 🡨 kcvalue + " /"

x 🡨 0

WHILE x < len(rvalues) DO

kcvalue 🡨 kcvalue + " ("+str(rvalues[x])+")^"

x 🡨 x + 1

kcvalue 🡨 kcvalue + str(rvalues[x])

x 🡨 x + 1

ENDWHILE

OUPUT kcvalue

rproductsum 🡨 0

FOR x 🡨 0 TO reactants.length - 1 DO

rproductsum += reactants[x].GetSRatio()

ENDFOR

pproductsum 🡨 0

FOR x 🡨 0 TO products.length – 1 DO

pproductsum 🡨 pproductsum + products[x].GetSRatio()

ENDFOR

OUTPUT “There are "+str(len(reactants))+" reactants, with units mol dm^-3."

OUTPUT "The product of these units is mol^"+str(rproductsum)+" dm^"+str(rproductsum \* -3)+"."

OUTPUT “There are "+str(len(products))+" products, with units mol dm^-3."

OUTPUT “The product of these units is mol^"+str(pproductsum)+" dm^"+str(pproductsum \* -3)+"."

OUTPUT "The product units must be divided by the reactant units, so"

numerator 🡨 1

FOR x 🡨 0 TO products.length – 1 DO

numerator 🡨 numerator \* (products[x].GetConcentration(self.Volume) \*\* products[x].GetSRatio())

ENDFOR

denominator 🡨 1

FOR x 🡨 0 TO reactants.length – 1 DO

denominator 🡨 denominator \* (reactants[x].GetConcentration(self.Volume) \*\* reactants[x].GetSRatio())

ENDFOR

value 🡨 numerator / denominator

power 🡨 pproductsum - rproductsum

OUTPUT "Kc = "+str(value)+" mol^"+str(power)+" dm^"+str(power \* -3)

END

The GetSRatio function is part of the ReactingSpecies class and uses its properties. It returns the reacting species’ value in the equation’s stoichiometric ratio – the smallest integer ratio of each molecule to the other molecules for a reaction to occur. The algorithm uses error handling to obtain this value. In effect, if a number is specified at the beginning of the formula, this will be the value. If no number is specified, the value is taken to be 1. This value is hardly ever above 9 at this stage so only the first character is examined.

FUNCTION GetSRatio

TRY

RETURN int(self.Formula[0])

EXCEPT

RETURN 1

ENDTRY

END

The GetKcChange function is in the Reaction class. It is used to get an abstract change in Kc when any condtions are adjusted. Kc is only constant for a particular reaction at a particular temperature, so an abstract change is useful as the user avoids having to entire a whole new set of reaction data.

Kc only changes in a reaction when the temperature is changed, which is why the temperature difference is the only parameter. The temperature difference is equal to the temperature of the “new” left-hand side reaction minus that of the “old” right-hand side reaction.

FUNCTION GetKcChange(TemperatureChange : int)

IF TemperatureChange == 0 THEN

RETURN ‘not changed’

ELSE

IF TemperatureChange < 0 THEN

IF Endothermic == FALSE THEN

RETURN ‘increased’

ELSE

RETURN ‘decreased’

ENDIF

ELSE

IF Endothermic == FALSE THEN

RETURN ‘decreased’

ELSE

RETURN ‘increased’

ENDIF

ENDIF

ENDIF

END

The GetFormulaForLabels function is in the Species class. It returns the species’ formula (which was entered by the user as plain text) in a HTML-based format so that it can be displayed conventionally to the end user. It does this by making a copy of the formula. It replaces every instance of “\_x” with “<sub>x</sub> and every instance of “^x” with “<sup>x</sup>” where x is an integer. This results in the formula having superscript and subscript characters in the right places.

FUNCTION GetFormulaForLabels

x 🡨 0

newformula 🡨 ""

WHILE x < self.Formula.length DO

IF self.Formula[x] == "\_" THEN

newformula 🡨 newformula + "<sub>"

x 🡨 x + 1

newformula 🡨 newformula + self.Formula[x]

newformula 🡨 newformula + "</sub>"

ELSE

IF self.Formula[x] == "^" THEN

newformula 🡨 newformula + "<sup>"

x 🡨 x + 1

newformula 🡨 newformula + self.Formula[x]

newformula 🡨 newformula + "</sup>"

ELSE

newformula 🡨 newformula + self.Formula[x]

ENDIF

ENDIF

x 🡨 x + 1

ENDWHILE

RETURN newformula

END

The ValidateVolume function uses regular expressions and exception handling to validate the volume string entered by the user. Because the acceptable values for the number of moles are the same as those for the volume, this function can be applied to the number of moles as well.

Note: when implemented, rather than being a separate function this was written as part of a larger algorithm to validate all input data.

FUNCTION ValidateVolume(volume : str)

TRY

IF float(volume) < 0.01 OR float(volume) > 99.99 THEN

OUTPUT "Volume must be between 0.01 and 99.99 inclusive."

ELSE

TRY

IF volume DOES NOT MATCH "[0-9]?[0-9]\.[0-9][0-9]?" THEN

OUTPUT "Volume must be between 0.01 and 99.99 inclusive, and must have a maximum of 2 decimal places."

EXCEPT

OUTPUT “Volume must be between 0.01 and 99.99 inclusive, and must have a maximum of 2 decimal places."

ENDIF

EXCEPT

OUTPUT "Volume must be a decimal number."

ENDTRY

END

The DrawCurve function is responsible for drawing the concentration or reaction rate graphs. All of these graphs have roughly the same shape: two curves which start at either end of the y axis and flatten out either in the middle (rate) or near the middle (concentration). However, the catalyst data causes the steepness of both curves to change: more of a stronger catalyst makes the curves steeper; more of a weaker catalyst makes the curves less steep. The algorithm uses the catalyst strength and number of moles to calculate eqmpoint (the point on the x axis where the curves flatten out). Then the PyQt pen system will be used to draw the axes, then an arc from x = 0 to x = eqmpoint, then from x = eqmpoint to x = 300 where the graph ends, for both curves.

FUNCTION DrawCurve

IF self.Reaction.GetCatalyst().GetUsed() THEN

change 🡨 self.\_reaction.GetCatalyst().GetEfficacy()

ELSE

change 🡨 1

ENDIF

eqmpoint 🡨 150 / change

IF change > 1 THEN

eqmpoint -= 0.1 \* self.Reaction.GetCatalyst().GetInitialMoles()

ELSE

IF change < 1 THEN

eqmpoint += 0.1 \* self.Reaction.GetCatalyst().GetInitialMoles()

ENDIF

self.Plotter.Pen.setPenColour(0, 0, 0)

self.Plotter.Pen.drawLine(0, 0, 0, 200)

self.Plotter.Pen.drawLine(0, 200, 300, 200)

self.Plotter.Pen.setPenColour (self.Reaction.GetReactantColour())

self.Plotter.Pen.moveTo(0, 0)

self.Plotter.Pen.drawArcTo(0, -100 + self.Plotter.GetFinalY(), eqmpoint \* 2, 200 - (2 \* self.Plotter.GetFinalY()), 180, 90)

self.Plotter.Pen.drawLineTo(300, 100 - self.Plotter.GetFinalY())

self.Plotter.Pen.setPenColour (self.Reaction.GetProductColour()))

self.Plotter.Pen.moveTo(0, 200)

self.Plotter.Pen.drawArcTo(0, 100 + self.Plotter.GetFinalY(), eqmpoint \* 2, 200 - (2 \* self.Plotter.GetFinalY()), 180, -90)

self.Plotter.Pen.drawLineTo(400, 120 + self.Plotter.GetFinalY())

END

The PlayAnimation function sets up the elements of the animation and performs the calculations required to display each frame. First, the 100 squares are constructed and drawn, laid out in a 10x10 grid. Then an eqmpoint value is calculated, very similarly to the last algorithm. The value determines the number of the frame in the animation where the chance of any square changing colour is equal. Then, each frame, each square’s Draw() function is called. When a square’s Draw() function is called, whether it changes colour depends on the chance parameter – see the function beneath PlayAnimation. Until equilibrium this chance is lower for squares with the product colour than squares with the reactant colour, which is consistent with the shape of the rate graph.

FUNCTION PlayAnimation

FOR x 🡨 0 TO 99 DO

newSquare 🡨 Square(200+((x % 10) \* 20), 120 + ((x // 10) \* 20)

self.Squares.append(newSquare)

newSquare.Draw(painter, -1)

ENDFOR

eqmpoint 🡨 750

IF self.Reaction.GetCatalyst().GetUsed() THEN

efficacy 🡨 self.Reaction.GetCatalyst().GetEfficacy()

eqmpoint 🡨 eqmpoint / efficacy

IF efficacy > 1 THEN

eqmpoint 🡨 eqmpoint - 0.5 \* self.Reaction.GetCatalyst().GetMoles()

ELSE

IF efficacy < 1 THEN

eqmpoint 🡨 eqmpoint + 0.5 \* self.Reaction.GetCatalyst().GetMoles()

ENDIF

ENDIF

ENDIF

FOR time 🡨 0 TO 1499 DO

chance 🡨 100

IF time < eqmpoint THEN

chance 🡨 chance + int(100 - ((time / eqmpoint) \* 100))

ENDIF

FOR x 🡨 0 TO 99 DO

self.Squares[x].Draw(chance)

ENDFOR

ENDFOR

END

FUNCTION Draw(chance : int)

IF chance > 0 THEN

IF (self.CurrentColour == self.ReactantColour AND randrange(chance) == 0) OR (self.CurrentColour == self.ProductColour AND randrange(201 - chance) == 0) THEN

IF self.CurrentColour == self.ReactantColour THEN

self.CurrentColour 🡨 self.ProductColour

ELSE

self.CurrentColour 🡨 self.ReactantColour

ENDIF

ENDIF

ENDIF

self.DrawSelf()

END

2.6: Classes used

2.6.1: Class diagrams

These are the classes which will be used to implement the chemical concepts:

**Reaction**

Public: ReactingSpeciesLimit : int

Private:

Temperature : int

Reactants : list (Reactant)

Products : list (Product)

Catalyst : Catalyst

Volume : float

Endothermic : boolean

ReactantColour : QColor

ProductColour : QColor

Public:

GetKc

GetKcChange

GetTemperature

SetTemperature

GetReactants

SetReactants

GetProducts

SetProducts

GetCatalyst

SetCatalyst

GetVolume

SetVolume

GetEndothermic

SetEndothermic

GetReactantColour

SetReactantColour

GetProductColour

SetProductColour

Species

Private:

Formula : String

Moles : float

Used : boolean

Public:

GetConcentration

GetFormula

SetFormula

GetFormulaForLabels

GetMoles

SetMoles

GetUsed

SetUsed

Catalyst

Private:

Efficacy

Public:

GetEfficacy

SetEfficacy

GetEfficacyAsString

ReactingSpecies

Public:

GetSRatio

Product

Reactant

2.6.2: Brief field and method descriptions

Most get and set methods are not described here as they simply return or set their corresponding property.

Reaction

* ReactingSpeciesLimit: Maximum number of reactants, and maximum number of products, per reaction. Will be set to 3
* Temperature: the temperature the reaction takes place at, measured in K
* Reactants: All reactants in the Reaction object, including ones that aren’t selected to be used
* Products: All products in the Reaction object, including ones that aren’t selected to be used
* Catalyst: The reaction’s catalyst
* Volume: The reaction vessel capacity, measured in dm3
* Endothermic: If true, the reaction takes in heat when products are formed. If false, it gives out heat
* ReactantColour: The colour that represents the reactants
* ProductColour: The colour that represents the products
* GetKc: Returns the reaction’s equilibrium constant
* GetKcChange: When conditions are changed, returns whether Kc has increased, decreased or not changed when conditions are changed.

Species

* Formula: The species’ chemical formula
* Moles: the number of moles at equilibrium
* Used: Whether or not the species is selected for use in the reaction
* GetConcentration: Returns the species’ concentration (number of moles divided by vessel volume)
* GetFormulaForLabels: Returns HTML-friendly version of formula, for proper display

ReactingSpecies

* GetSRatio: Returns the reacting species’ stoichiometric value

Catalyst

* Efficacy: The factor by which the reaction speeds up
* GetEfficacyAsString: Verbal description of Efficacy (10 = good; 4 = poor; etc.)

2.7: Planned user interfaces

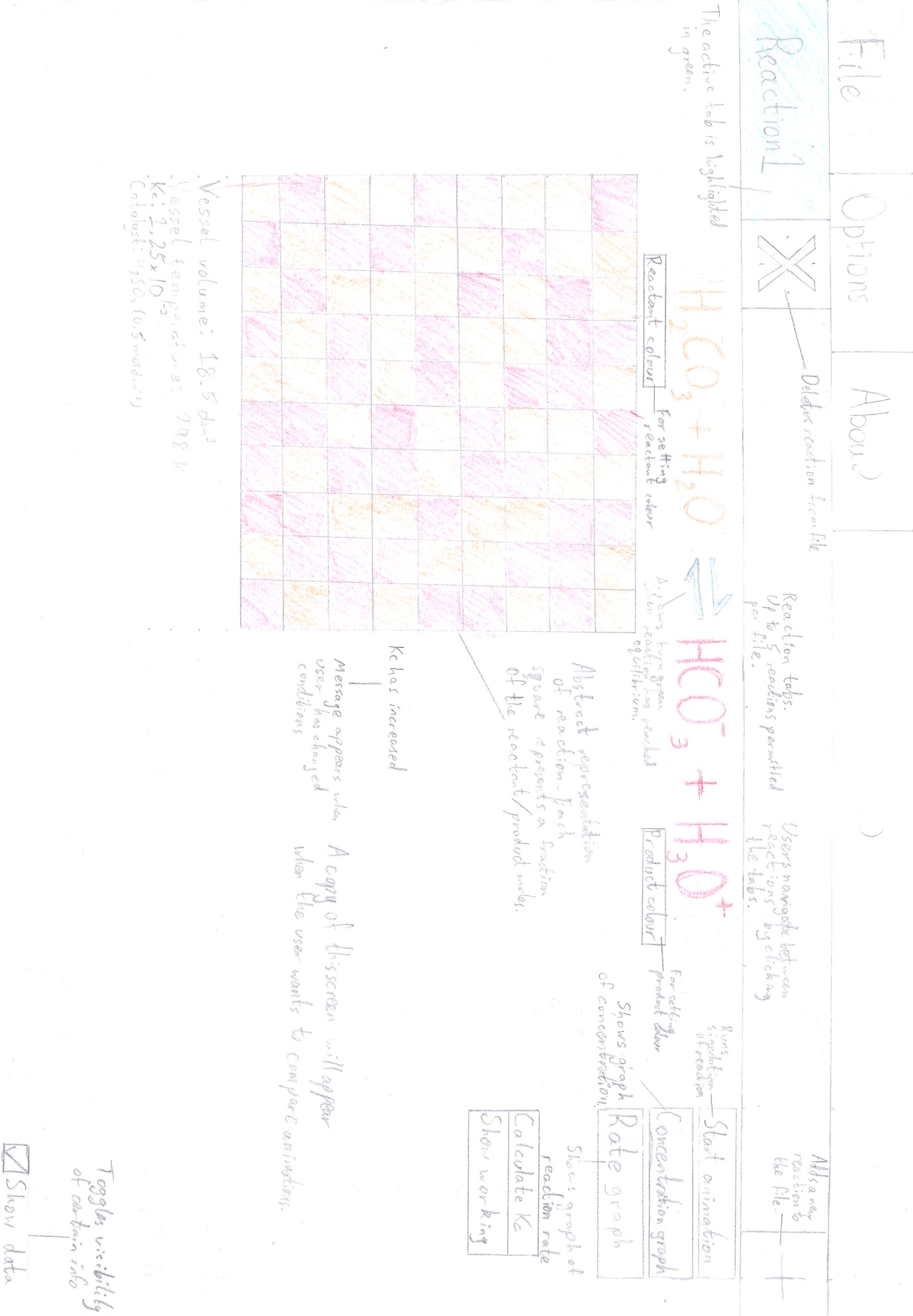
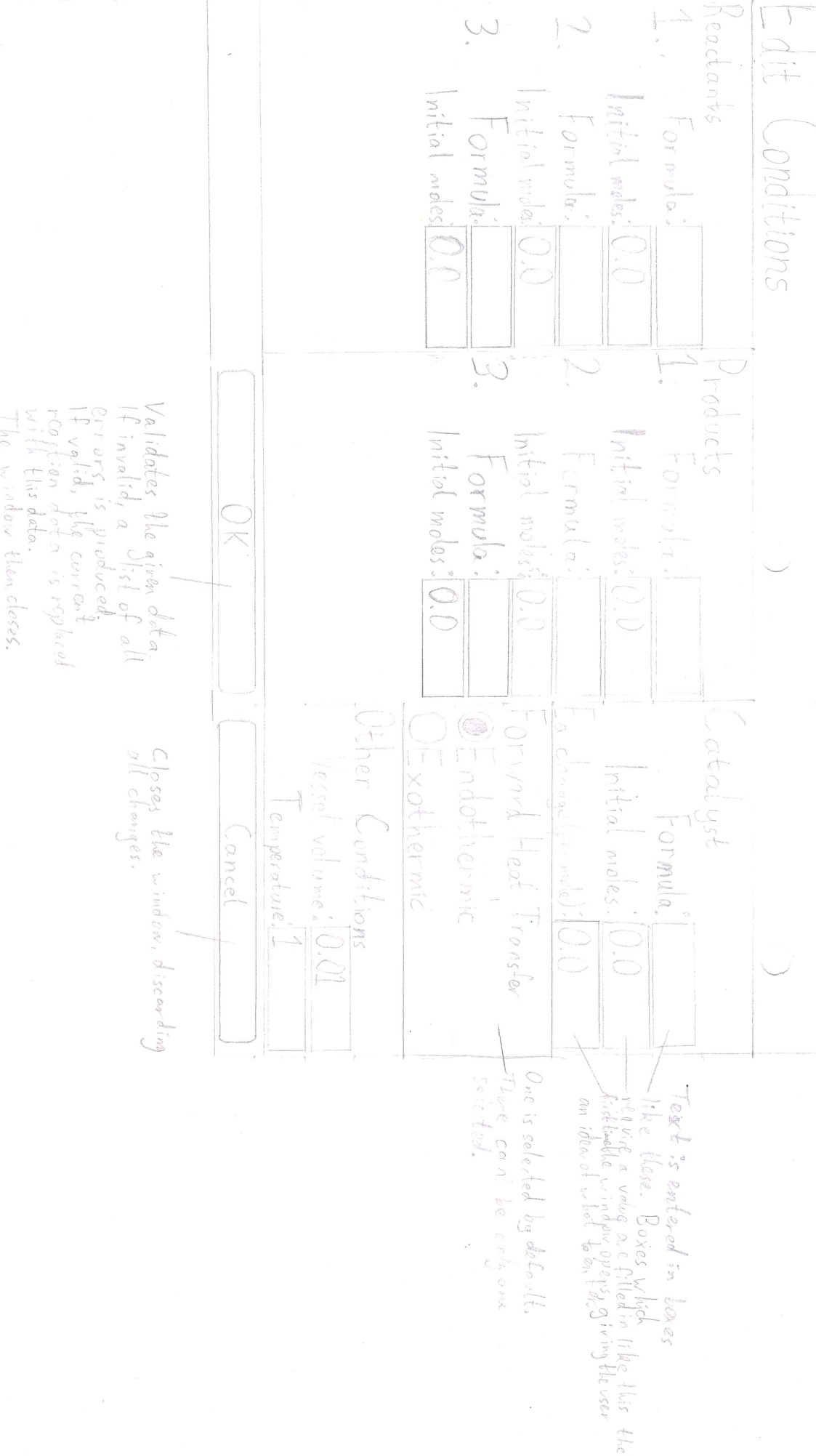
   


Figure 2.4: The planned reaction window screen.

Figure 2.5: The planned conditions dialog screen.

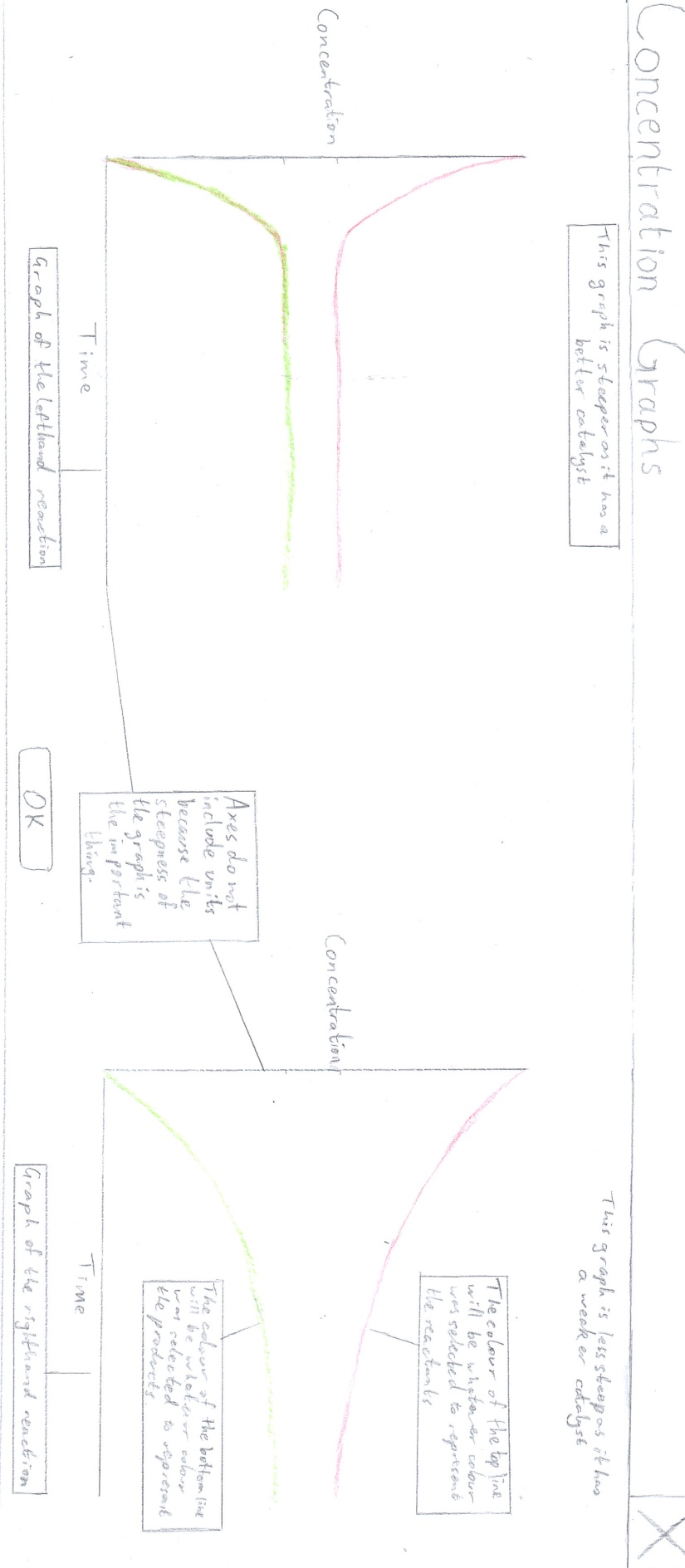


Figure 2.6: The planned screen for concentration graphs. The same screen is shown for rate graphs, except “Concentration” is replaced with “Rate” and the graphs meet in the middle.

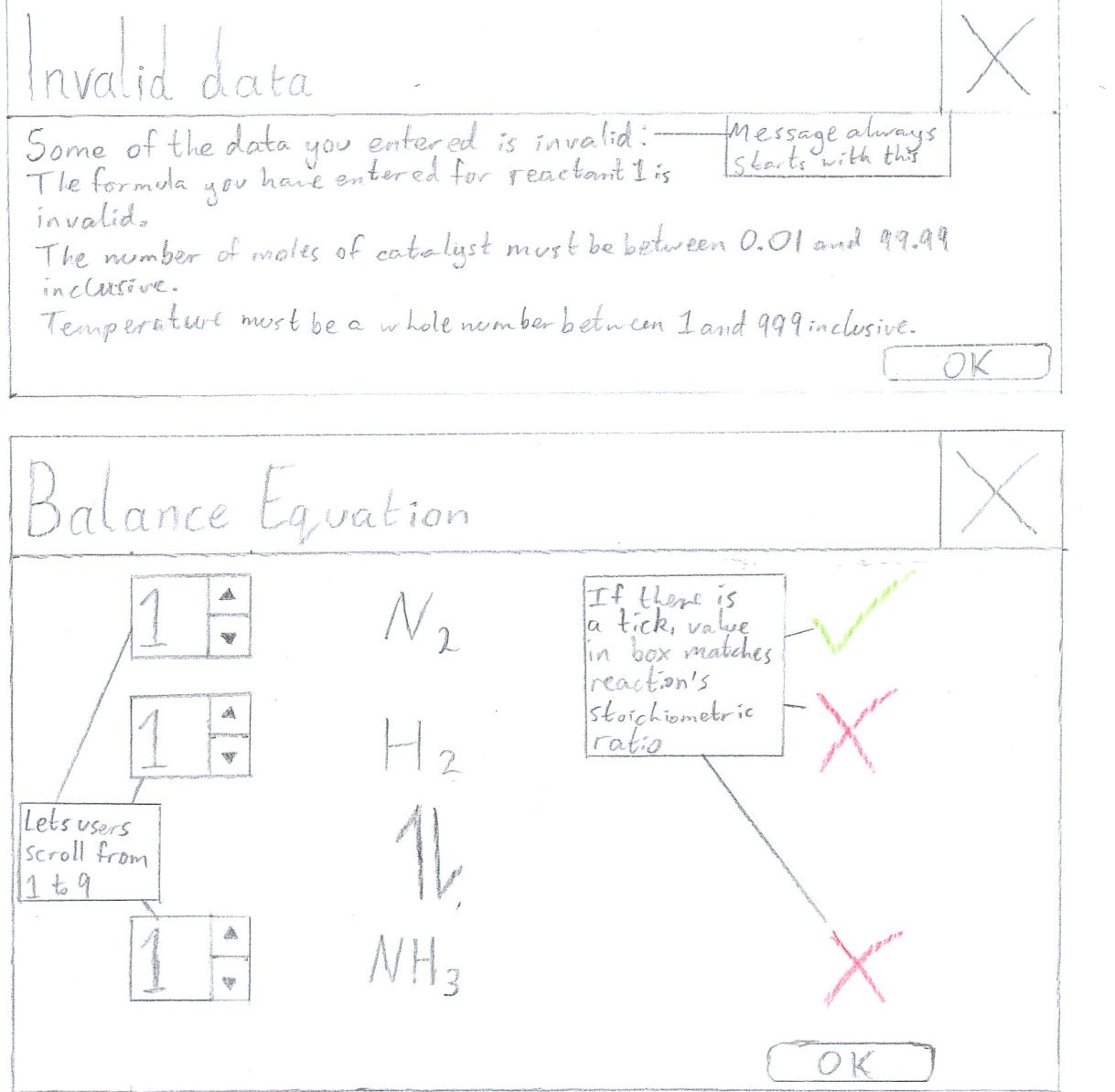


Figure 2.7: The planned error box for when the user enters invalid data. With every item of invalid data, the list of error messages is added to. The window should resize to fit the entire list onto it.

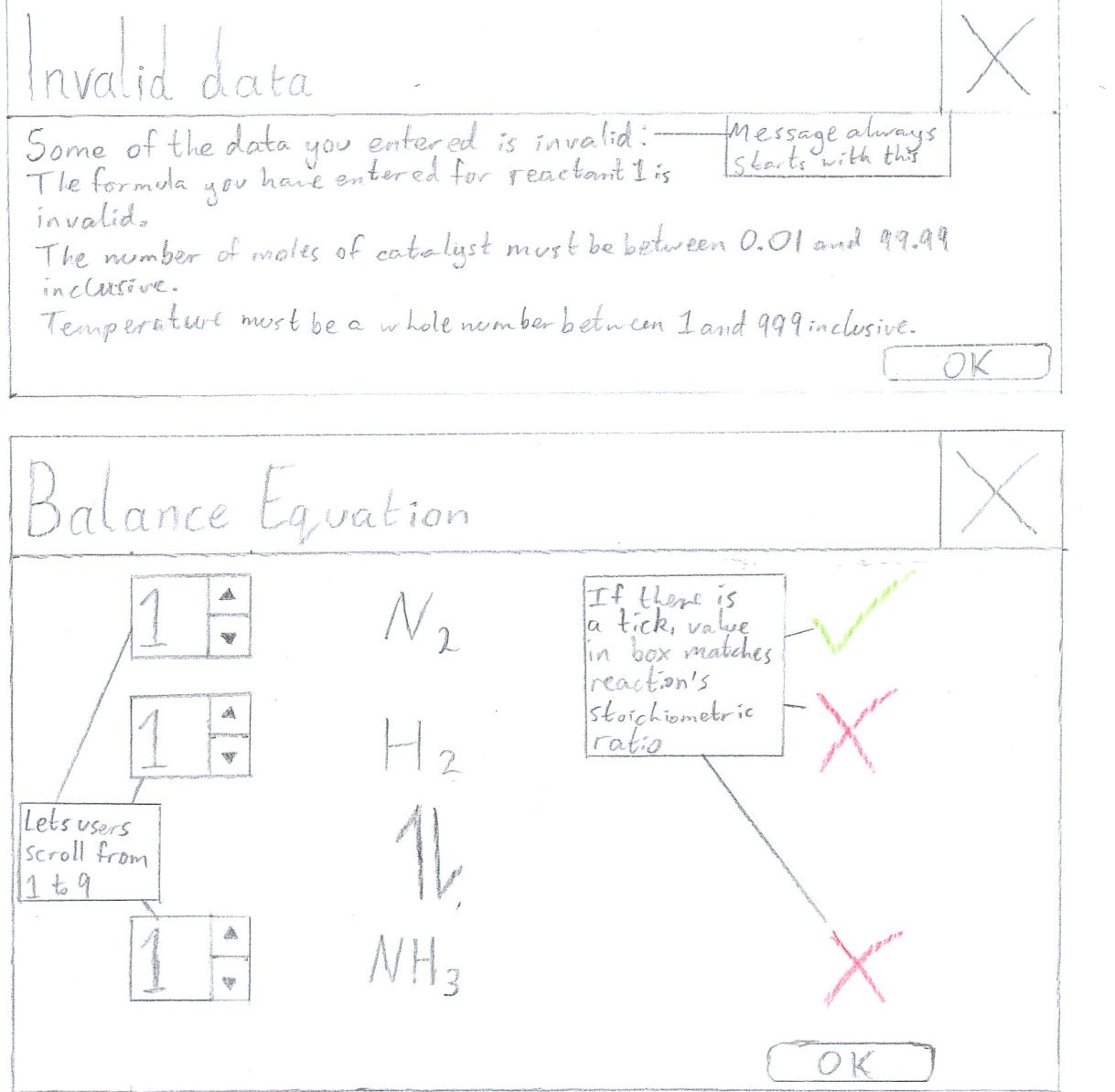


Figure 2.8: The planned screen for when users attempt to balance the reaction equation.

2.8: Security

The program will not need to store any personal or otherwise sensitive data, so no file encryption will be used. However, before distribution the program will be compiled to binary to avoid the source code being edited, which could make the program unusable.

2.9: Test plan

The test strategy for validation will be comparing the result to a dry run of each algorithm, because the algorithms used are easy for humans to perform and check. Navigation requires black-box testing because it will be handled by the application framework, the source code of which is unavailable. Logic must also be tested with a black-box method because the simulation involves thousands of calculations being performed each time, as well as random number generation, so white-box and dry-run testing are practically impossible.

2.9.1: Validation

All user input required to produce each simulation will be entered in text boxes in the Conditions dialog (see Figure 2.4). These text boxes are the only areas where users cannot select from predetermined valid values, so these are the only areas where input validation will take place. Data entered as a string may be converted internally to other data types. Invalid input data will prevent the window from closing when the “OK” button is pressed.

|  |  |  |  |
| --- | --- | --- | --- |
| Test number | Description | Test type; input data | Expected result |
| 1 | The formula of each reactant, product and the catalyst must be 1-25 characters long inclusive, and must fit R1 | N; “2NaCl” | If all other data valid, window closed, reaction updated and user can continue |
| 2 | E; “2N@C&” | Message explaining that the formula is invalid |
| 3 | E; “” | Message explaining the accepted formula length |
| 4 | B; “H” | If all other data valid, window closed, reaction updated and user can continue |
| 5 | B; “CH\_3COO(CH\_2)\_4C\_1\_0H\_2\_1” | If all other data valid, window closed, reaction updated and user can continue |
| 6 | Number of moles must be between 0.01 and 99.99 inclusive, and in the float format. There must be 1-2 digits before and after the decimal point, i.e. it must fit R2 | N; “24.53” | If all other data valid, window closed, reaction updated and user can continue |
| 7 | E; “101.42” | Message explaining the accepted range |
| 8 | E; “-4.24” | Message explaining the accepted range |
| 9 | E; “” | Message explaining that a decimal number must be entered |
| 10 | E; “6” | Message explaining that a decimal number must be entered |
| 11 | E; “4. | Message explaining that a decimal number must be entered |
| 12 | E; “3.14159” | Message explaining that there should be a maximum of two decimal places |
| 13 | E; “sixteen” | Message explaining that a decimal number must be entered |
| 14 | B; “99.99” | If all other data valid, window closed, reaction updated and user can continue |
| 15 | B; “0.01” | If all other data valid, window closed, reaction updated and user can continue |
| 16 | Volume must be between 0.01 and 99.99 inclusive, and in the float format. There must be 1-2 digits before and after the decimal point, i.e. it must fit R2 | N; “13.37” | If all other data valid, window closed, reaction updated and user can continue |
| 17 | E; “133.75” | For a value larger than the range, a message explaining the accepted range |
| 18 | E; “-42.69” | For a value smaller than the range, a message explaining the accepted range |
| 19 | E; “” | Message explaining that a decimal number must be entered |
| 20 | E; “72” | Message explaining that a decimal number must be entered |
| 21 | E; “72.” | Message explaining that a decimal number must be entered |
| 22 | E; “1.234” | Message explaining that there should be a maximum of two decimal places |
| 23 | E; “fourteen” | Message explaining that a decimal number must be entered |
| 24 | B; “99.99” | If all other data valid, window closed, reaction updated and user can continue |
| 25 | B; “0.01” | If all other data valid, window closed, reaction updated and user can continue |
| 26 | Temperature must be an integer between 0 and 1000 exclusive | N; “450” | If all other data valid, window closed, reaction updated and user can continue |
| 27 | E; “2000” | Message explaining accepted range |
| 28 | E; “-200” | Message explaining accepted range |
| 29 | E; “” | Message explaining that an integer must be entered |
| 30 | E; “Boiling” | Message explaining that an integer must be entered |
| 31 | B; “1” | Window closed, reaction updated and user can continue |
| 32 | B; “999” | Window closed, reaction updated and user can continue |

2.9.2: Navigation

Navigation will be done via buttons and menus, using the left mouse button.

The “File” menu

|  |  |  |
| --- | --- | --- |
| Test number | Description | Expected navigation |
| 1 | On clicking “New” | Popup box asking user if they want to save |
| 1a) | On clicking “Yes” in popup box | “Save As...” dialog if not saved before; none if saved before. Popup box closed; left reaction window reset |
| 1b) | On clicking “No” in popup box | Popup box closed; left reaction window reset |
| 2 | On clicking “Open” | “Open...” dialog |
| 3 | On clicking “Save” | None |
| 4 | On clicking “Save as” | “Save As...” dialog |
| 5 | On clicking “Print” | “Print” dialog |
| 6 | On clicking “Exit” | Closes program |

The “Options” menu

|  |  |  |
| --- | --- | --- |
| Test number | Description | Expected navigation |
| 7 | On clicking “Edit Conditions” | “Edit Conditions” dialog |
| 7a) | On clicking “Cancel” in “Edit Conditions” dialog | Closes dialog window (no changes made) |
| 7b) | On clicking “OK” in “Edit Conditions” dialog (with invalid data) (“Students attempt to balance equation” unchecked) | “Invalid Data” dialog |
| 7b)i) | On clicking “OK” in “Invalid Data” dialog | Closes dialog window |
| 7c) | On clicking “OK” in “Edit Conditions” dialog (with invalid data) (“Students attempt to balance equation” checked) | “Invalid Data” dialog |
| 7d) | On clicking “OK” in “Edit Conditions” dialog (with valid data) (“Students attempt to balance equation” unchecked) | Closes dialog window; reaction window updated |
| 7e) | On clicking “OK” in “Edit Conditions” dialog (with valid data) (“Students attempt to balance equation” checked) | “Balance Equation” dialog |
| 7e)i) | On clicking “OK” in “Balance Equation” dialog | Closes dialog window and “Edit Conditions” dialog |

The “About” menu

|  |  |  |
| --- | --- | --- |
| Test number | Description | Expected navigation |
| 8 | On clicking “Version [version of program]” | None |

The reaction windows

The buttons which appear for the left and right reaction viewing widgets function exactly the same way, so only the results for the buttons in the left widget are included.

|  |  |  |
| --- | --- | --- |
| Test number | Description | Expected navigation |
| 9 | On clicking “Set reactant colour” | “Select Color” dialog |
| 10 | On clicking “Set product colour” | “Select Color” dialog |
| 11 | On clicking “Concentration Graphs” | “Compare Graphs” window |
| 12 | On clicking “Rate Graphs” | “Compare Graphs” window |
| 13 | On clicking “Close” in “Compare Graphs” window | Closes window |
| 14 | On clicking “Calculate Kc” | None (displays text when “Show data” is checked) |
| 15 | On clicking “Show working” | A small box showing working in plain text |
| 16 | On clicking “Start animation” | None |
| 17 | On clicking “Compare reaction” | None (right reaction window updated) |
| 18 | On clicking “+” (open tab) button | None |
| 19 | On clicking “X” (close tab) button when only 1 reaction in file | None |
| 20 | On checking / unchecking “Show data” box | None |

2.9.3: Logic

Animation

Note: “Time taken” refers to the time taken from the first frame with at least 1 square with the product colour to the first frame where at least 50 squares have the product colour.

|  |  |  |
| --- | --- | --- |
| Test number | Description | Expected outcome |
| 1 | Comparing good catalyst at 25 and 75 moles | Less time taken at 75 moles |
| 2 | Comparing poor catalyst at  25 and 75 moles | Less time taken at 75 moles |
| 3 | Comparing unused catalyst at 25 and 75 moles | No effect on animation |
| 4 | Comparing inhibitor catalyst at 25 and 75 moles | Less time taken at 25 moles |
| 5 | Comparing time taken by animation with each catalyst at 50 moles | In order of least to most time taken: good; poor; unused; inhibitor |

Graphs

|  |  |  |
| --- | --- | --- |
| Test number | Description | Expected curve steepness |
| 6 | Comparing good catalyst at 25 and 75 moles | Steeper at 75 moles |
| 7 | Comparing poor catalyst at 25 and 75 moles | Steeper at 75 moles |
| 8 | Comparing unused catalyst at 25 and 75 moles | No effect on curve |
| 9 | Comparing inhibitor catalyst at 25 and 75 moles | Steeper at 25 moles |
| 10 | Good catalyst  Poor catalyst  No catalyst  Inhibitor catalyst | In order of steepest to least steep curve: good; poor; unused; inhibitor |
| 11 | Testing if graph plots alongside animation | Point where curves flatten out should appear at the time reaction reaches equilibrium |

Calculations

|  |  |  |
| --- | --- | --- |
| Test number | Description | Expected outcome |
| 12 | Showing working for Kc | For the inputs in the cross-reference, final Kc value of 228.921 mol-2 dm6 (not calculated precisely) |
| 13a) | Current reaction has lower temperature than reaction being compared (endothermic reaction) | Message saying Kc has decreased |
| 13b) | Current reaction has lower temperature than reaction being compared (exothermic reaction) | Message saying Kc has increased |
| 14a) | Current reaction has higher temperature than reaction being compared (endothermic reaction) | Message saying Kc has increased |
| 14b) | Current reaction has higher temperature than reaction being compared (exothermic reaction) | Message saying Kc has decreased |
| 15a) | Current reaction has same temperature as reaction being compared (endothermic reaction) | Message saying that Kc has not changed |
| 15b) | Current reaction has same temperature as reaction being compared (exothermic reaction) | Message saying that Kc has not changed |

2.10: Prototype

2.10.1: Changes made

The program prototype had very few features. These included file handling and a tabbed window for reactions.

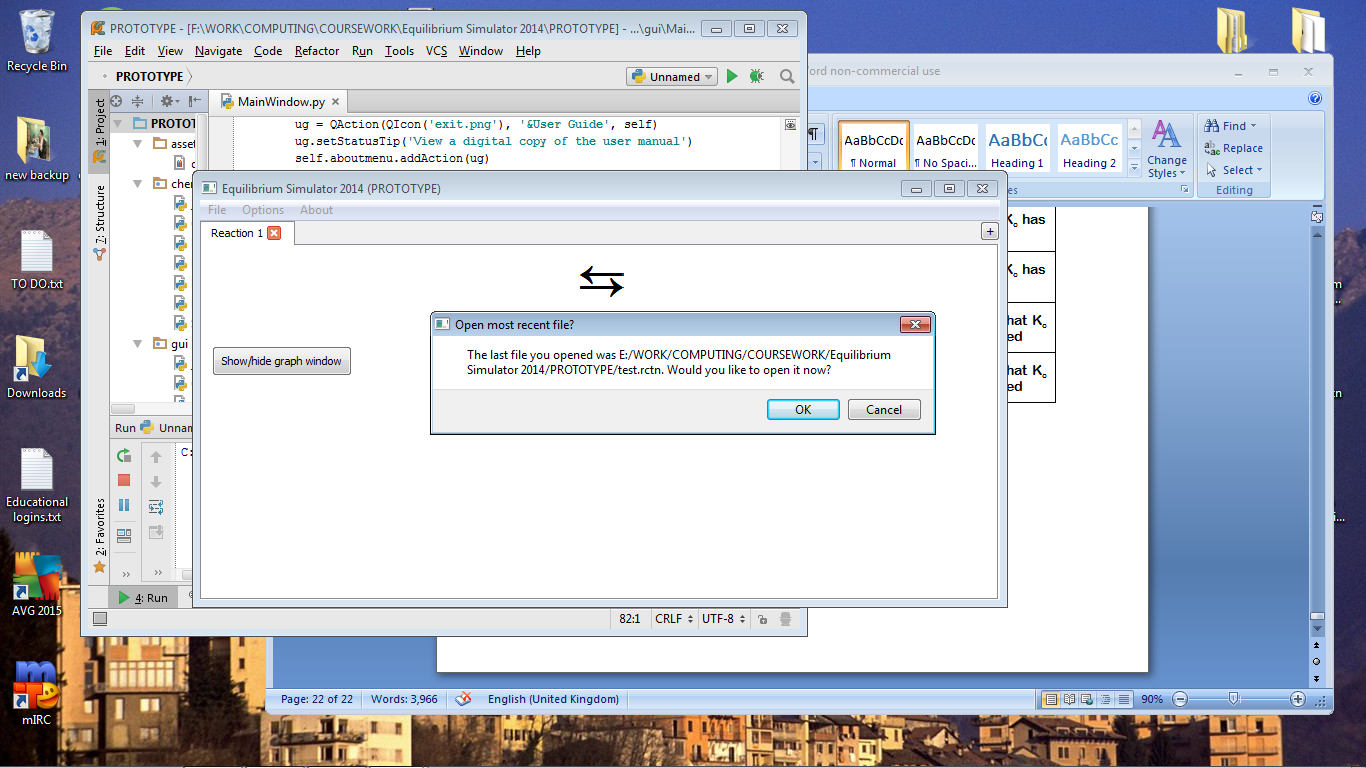
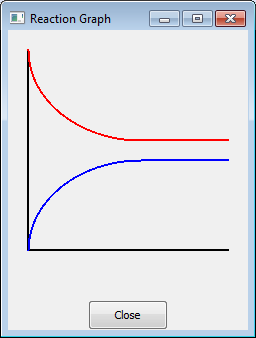


Figure 2.9: The prototype’s main screen.



The graphing system worked well, but side-by-side comparison had not been implemented, as well as the graph axes not being labelled.

In the “Edit Conditions” dialog, some changes had to be made. Firstly chemicals would not be used in reaction calculations if the number of moles was set to 0.0, but they would still appear in the equation and have their formulae validated. This would not have been satisfactory for demonstrating reactions with a smaller number of reactants or products, or no catalyst, so a checkbox solution was developed instead. Validation had also not yet been implemented.

Figure 2.10: The prototype graph window.

The end user pointed out that as the method of entering superscript and subscript characters was not intuitive, some brief instructions would be useful. Furthermore, teachers and students are highly unlikely to know the precise strength of a catalyst per mole, so it was suggested that users select catalyst strength from four possible options: good (speeds up reaction rate by a lot); poor (speeds up reaction rate but not by as much); and inhibitor (slows down reaction). The fourth option would be not to use a catalyst. Lastly, the double arrow symbol used did not fit the convention, so a different symbol was found.

At this point, the end user also requested that Undo and Redo functions be added to the program for when changes are made to a reaction.

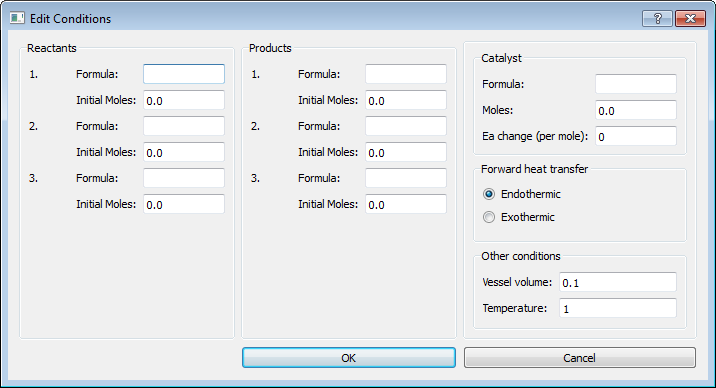
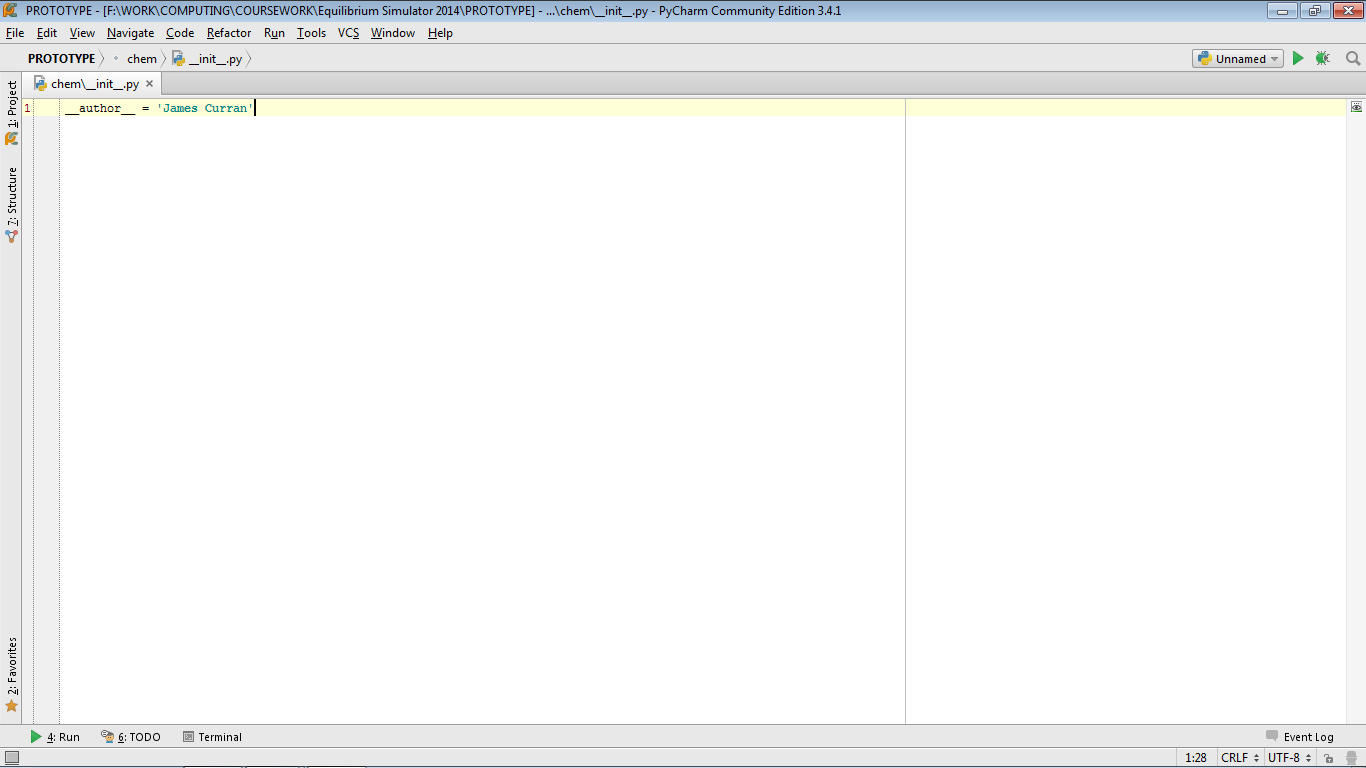


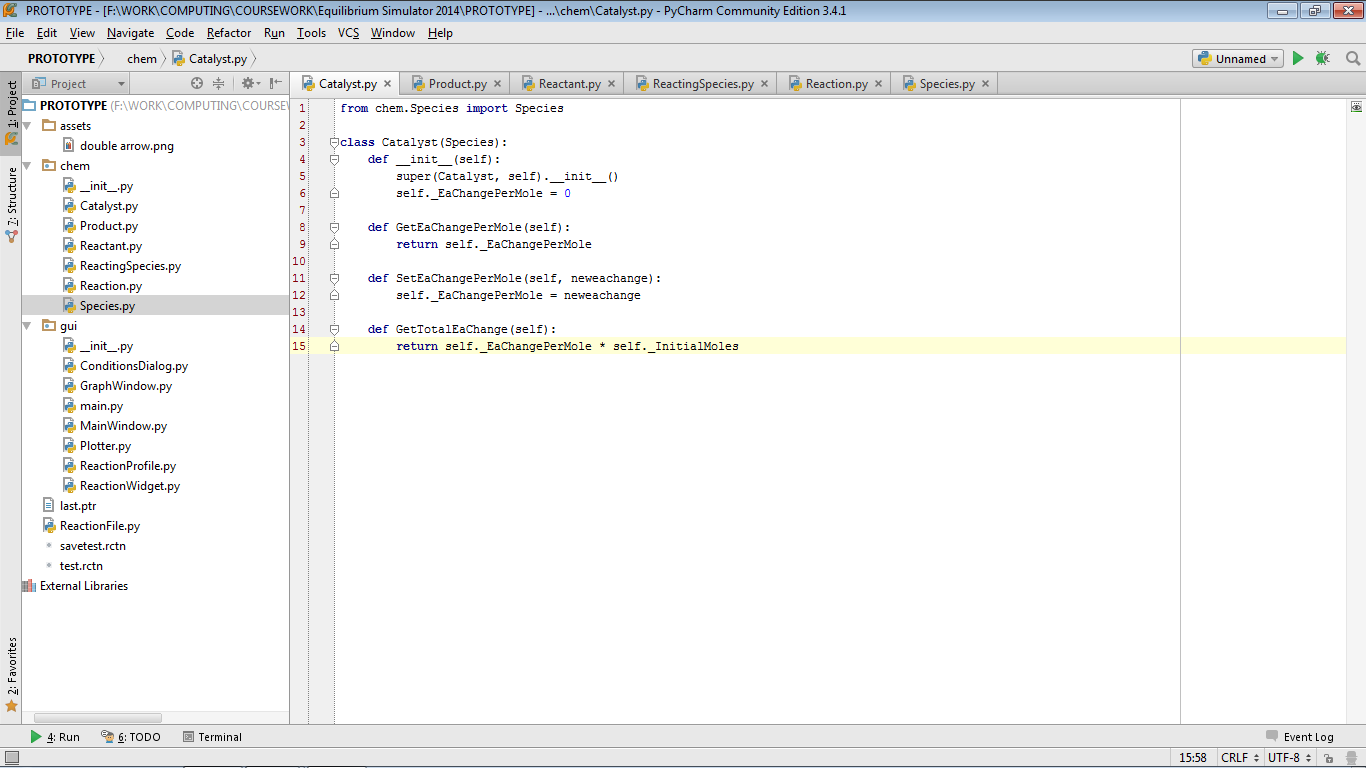
Figure 2.11: The prototype conditions dialog.

2.10.2: Prototype source code

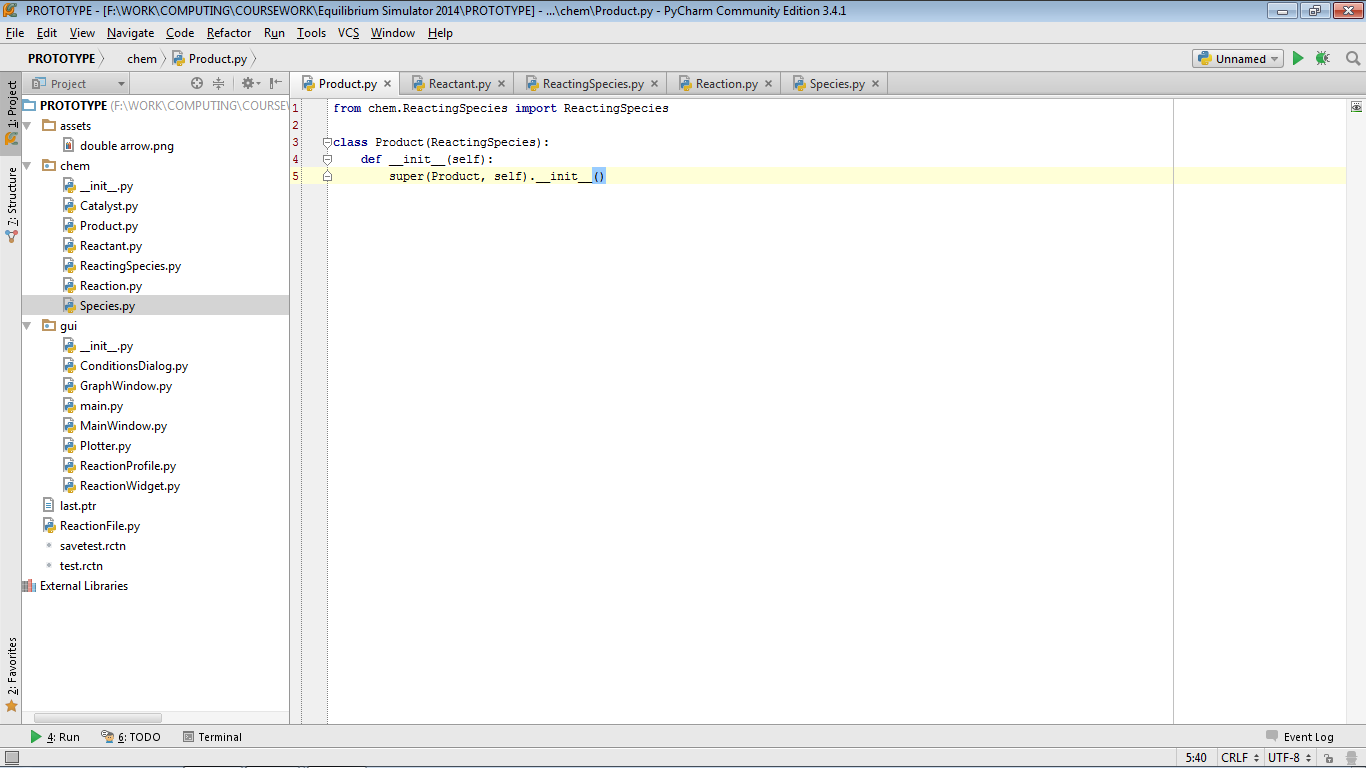
\_\_init\_\_.py



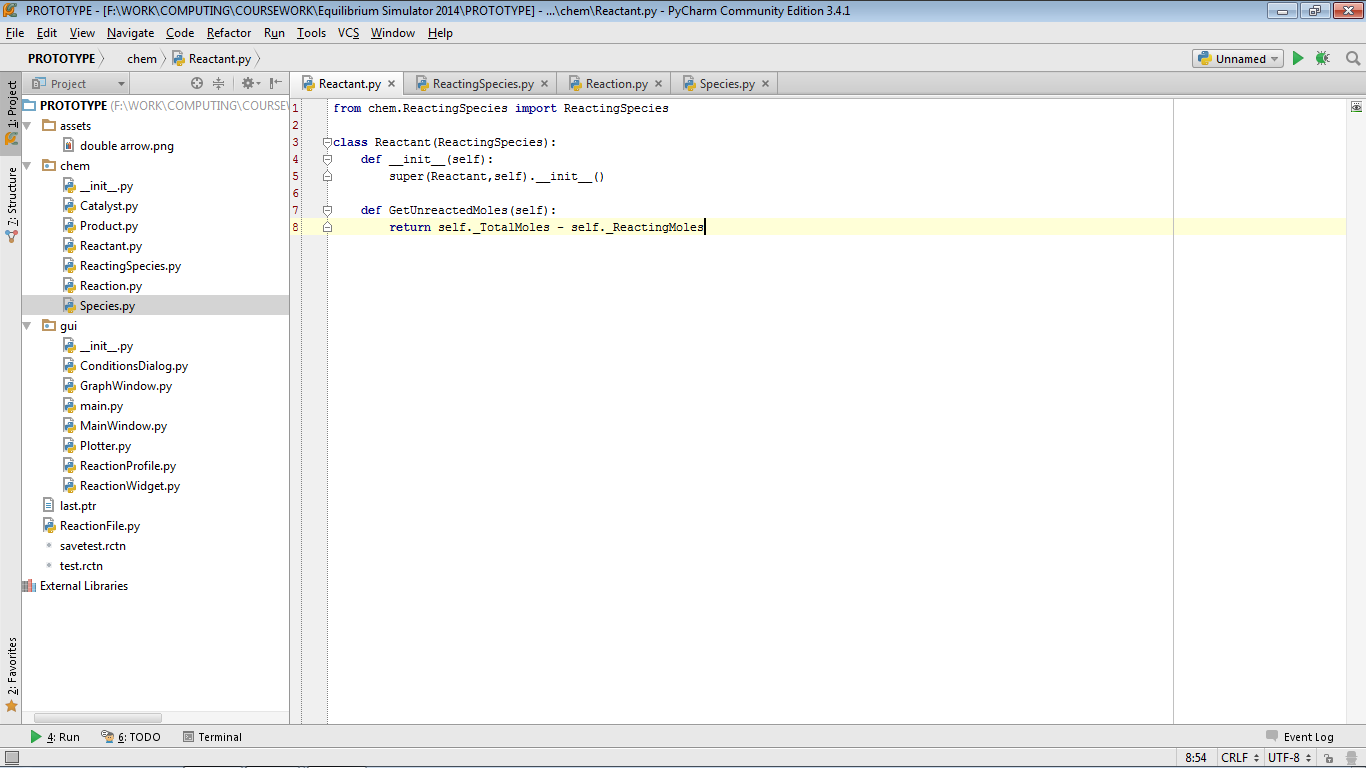
Catalyst.py



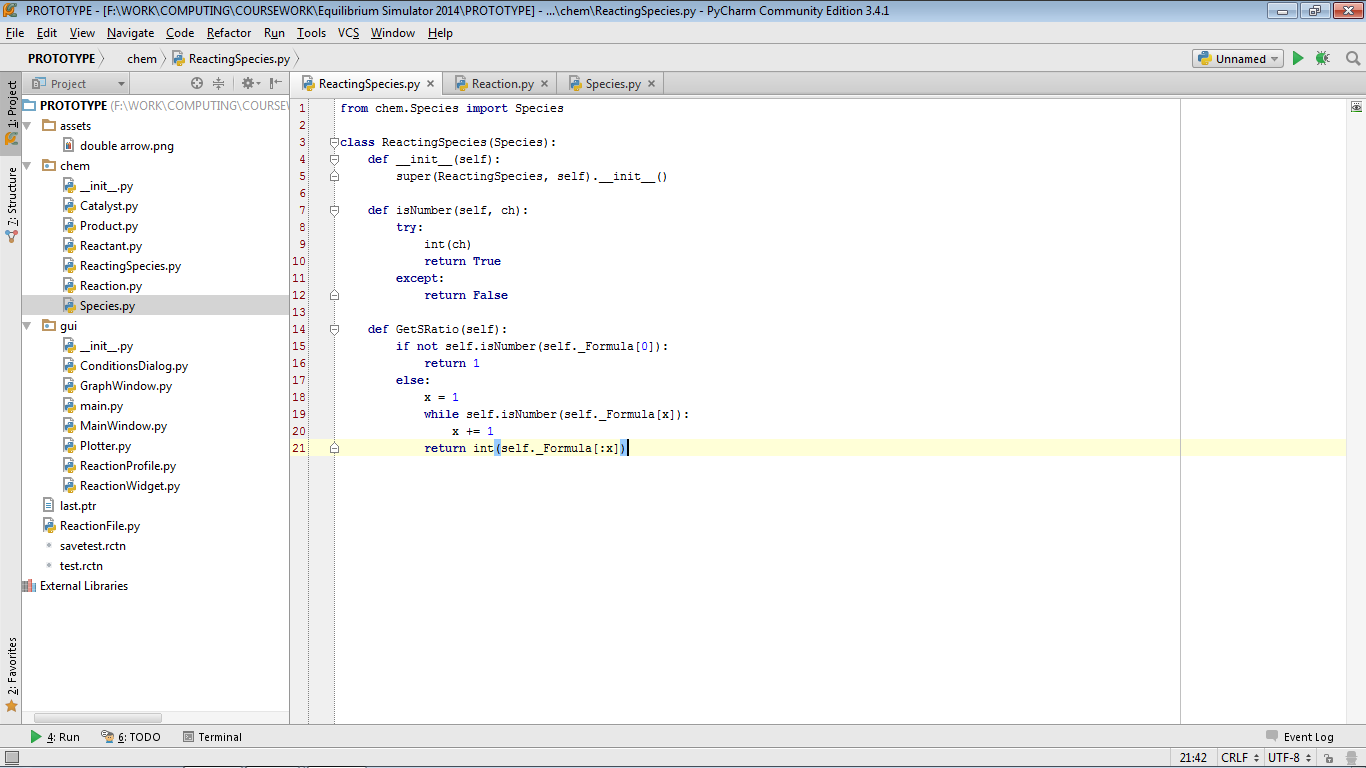
Product.py



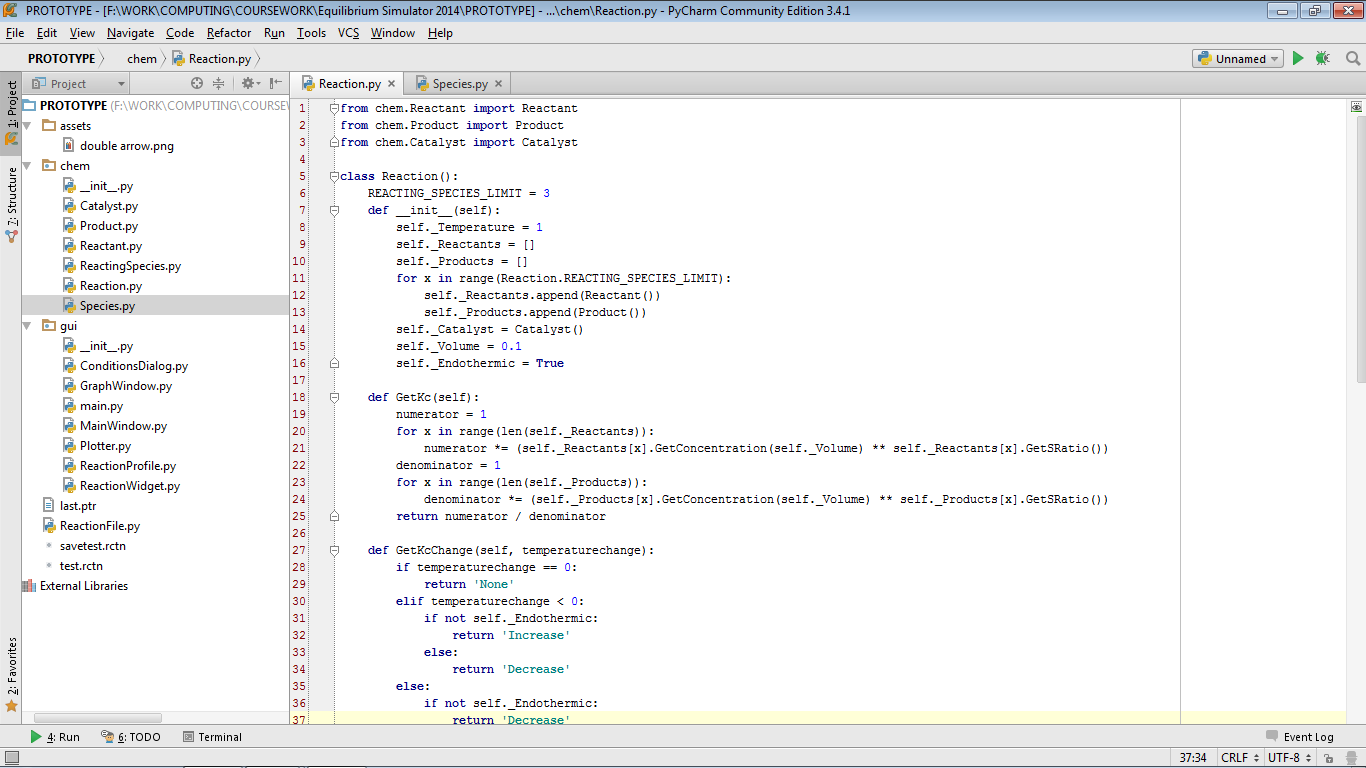
Reactant.py



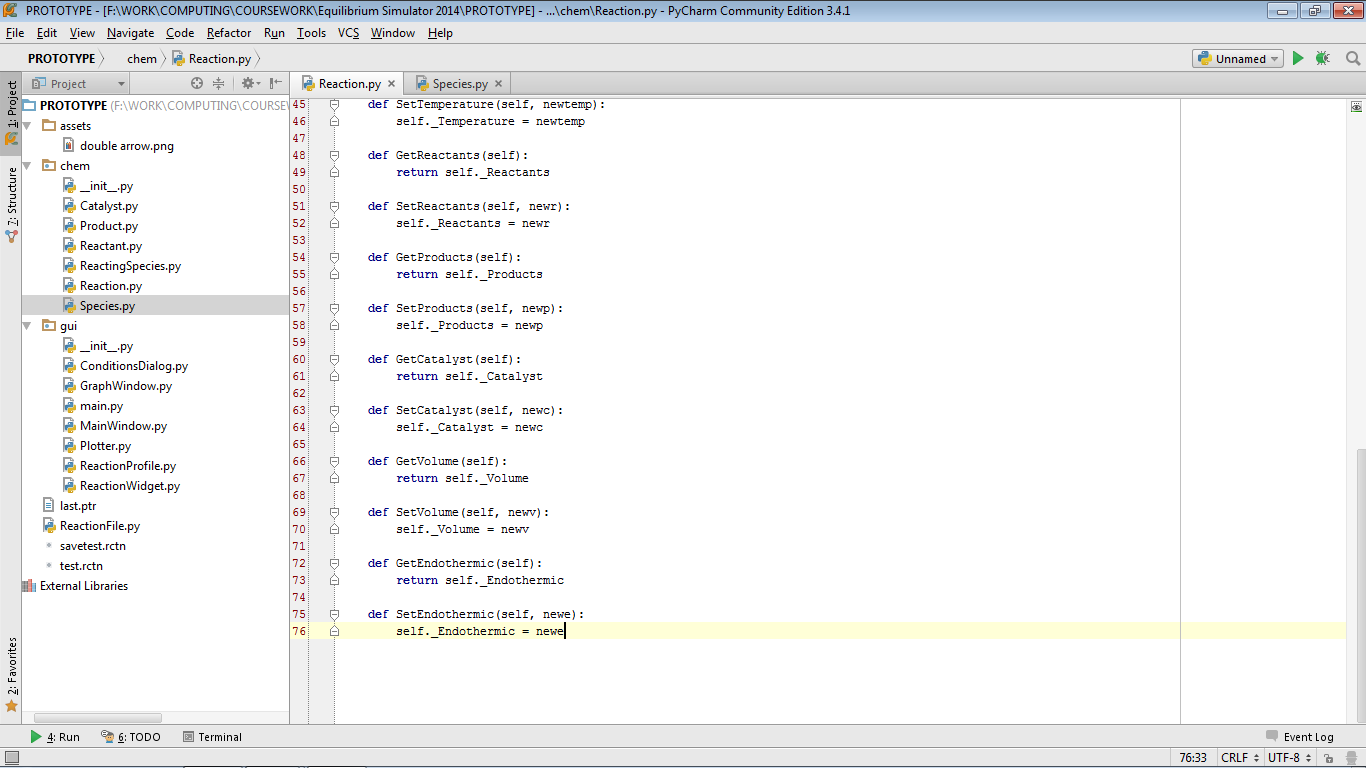
ReactingSpecies.py



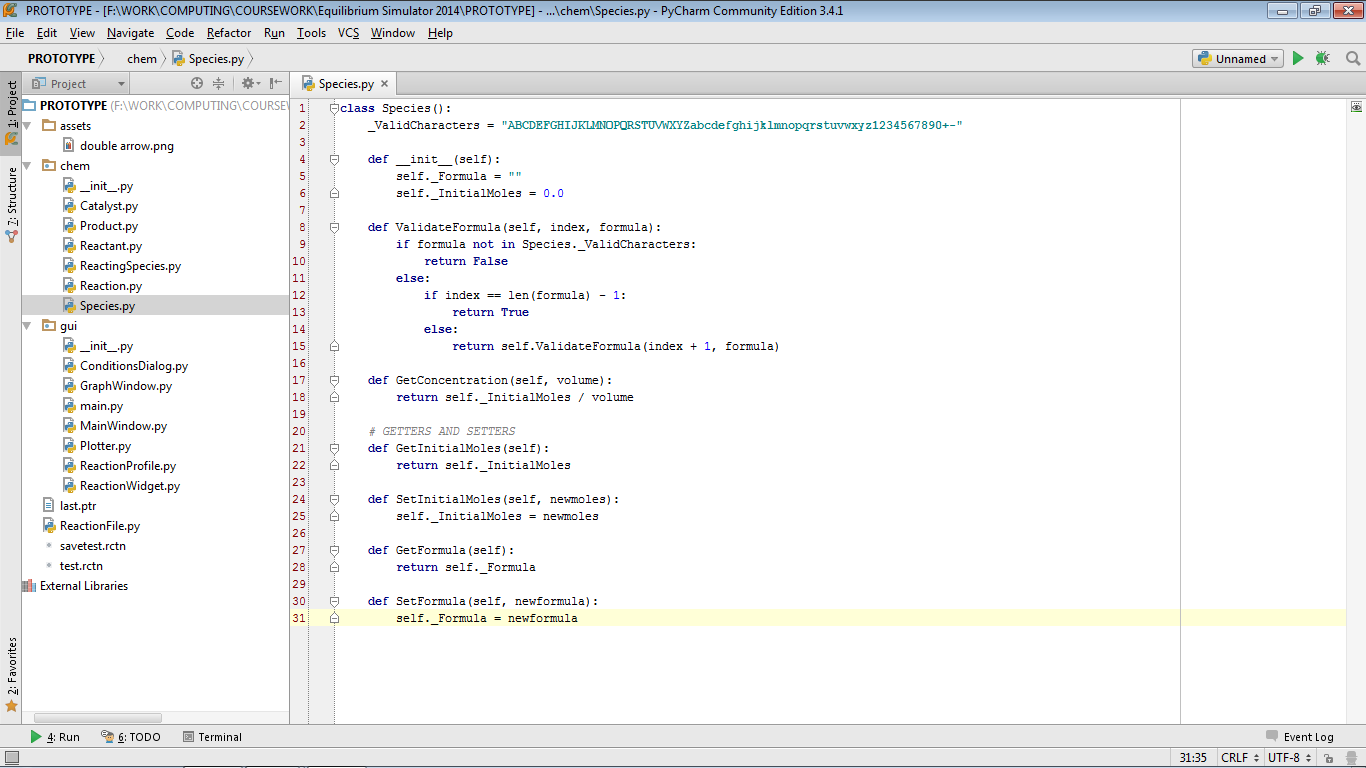
Reaction.py (part 1)

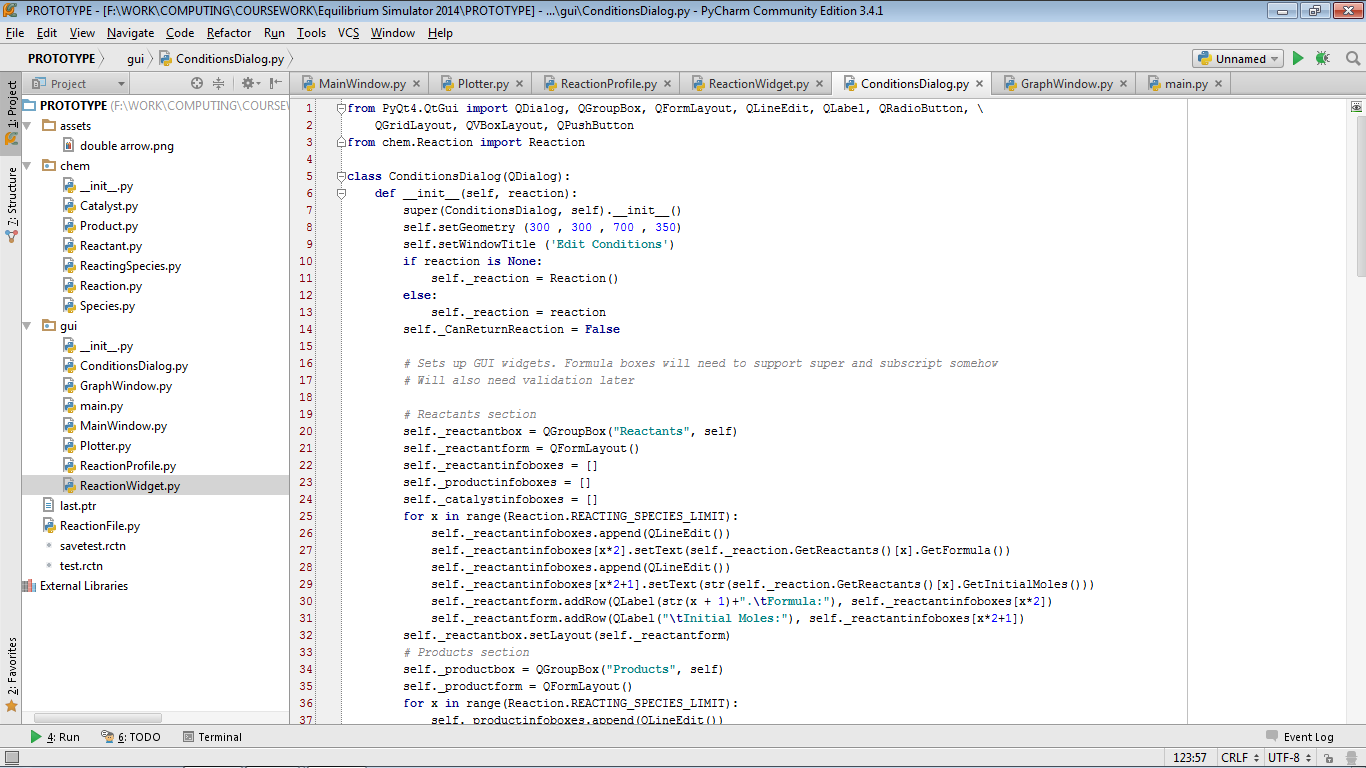
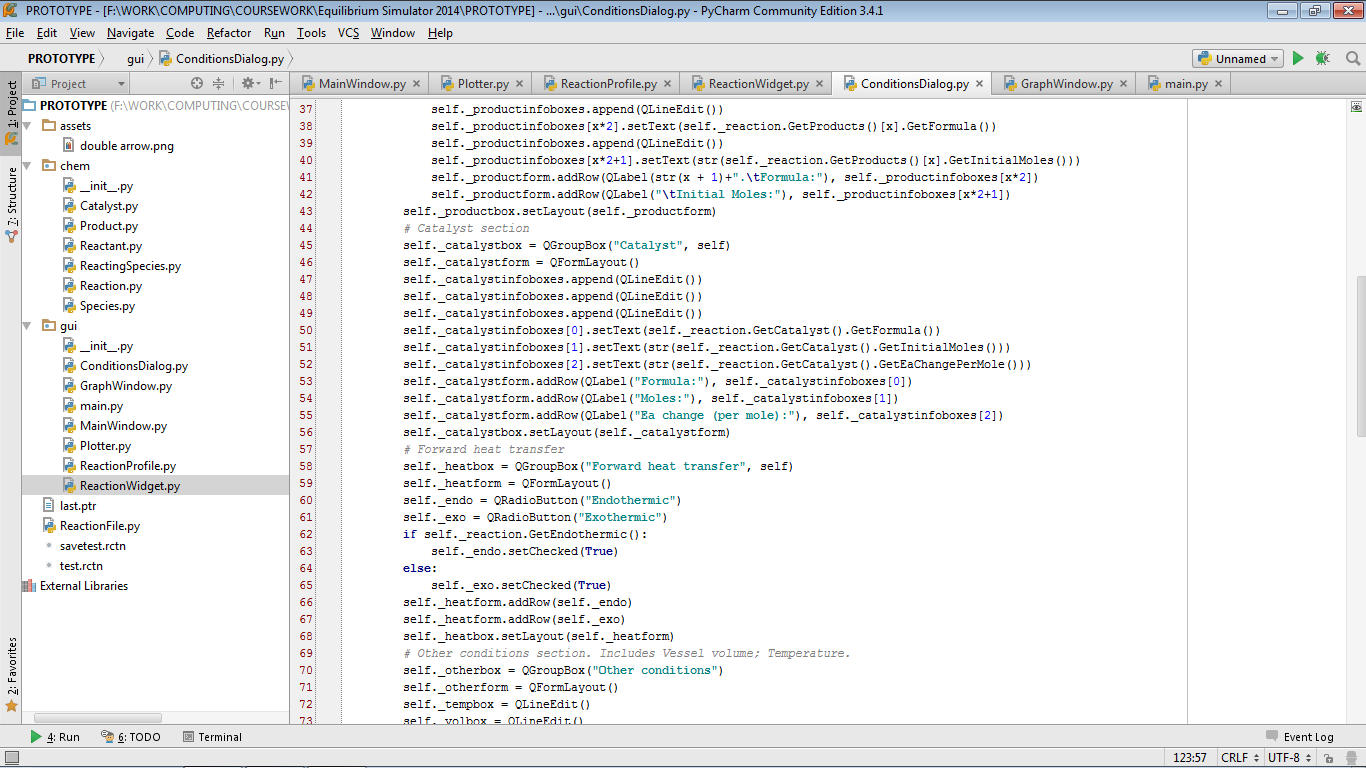

Reaction.py (part 2)



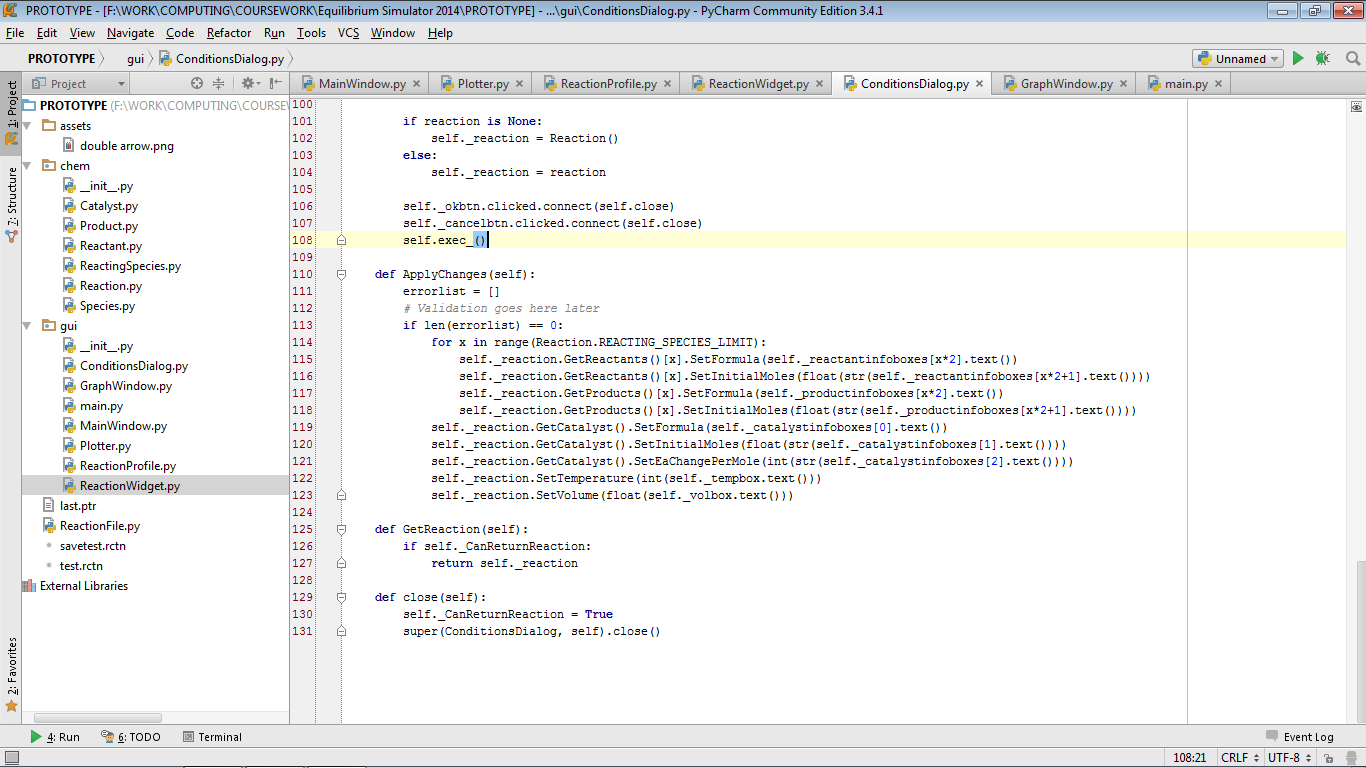
Species.py



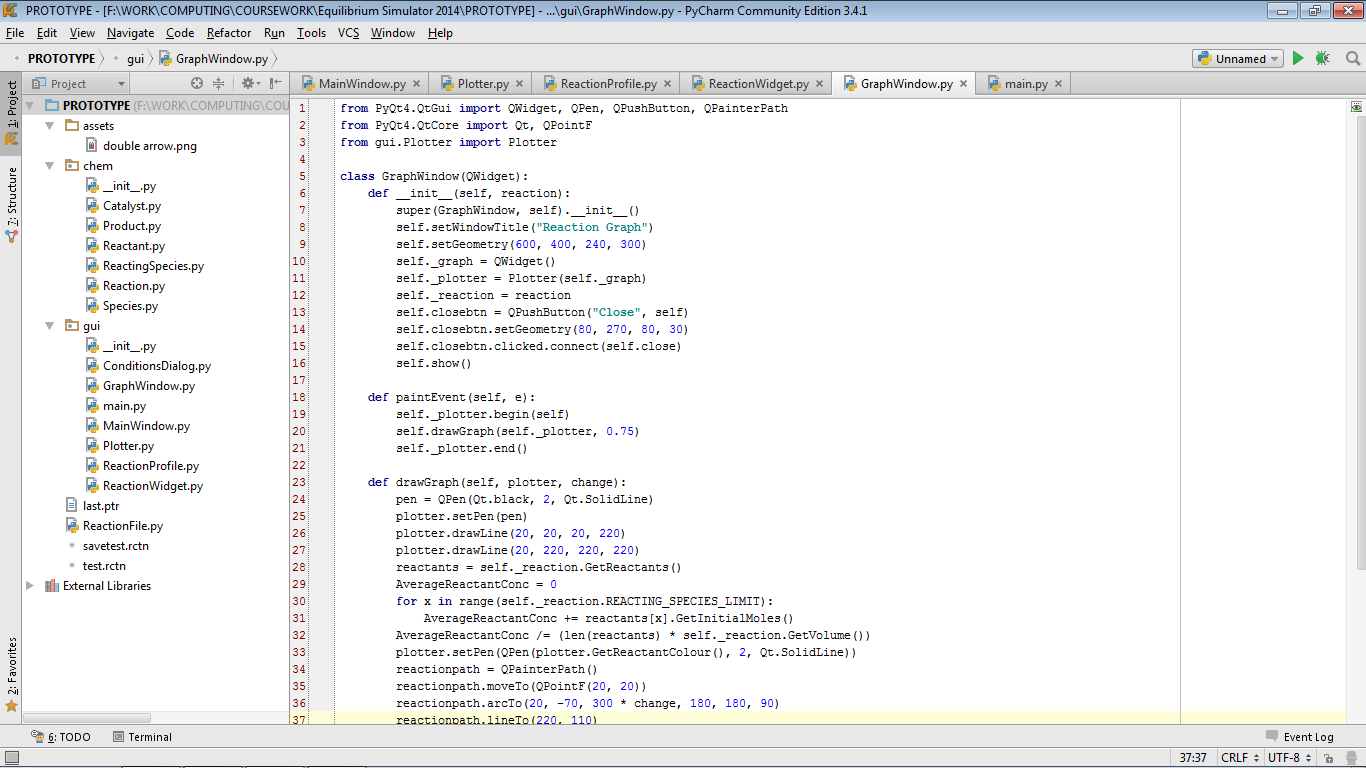
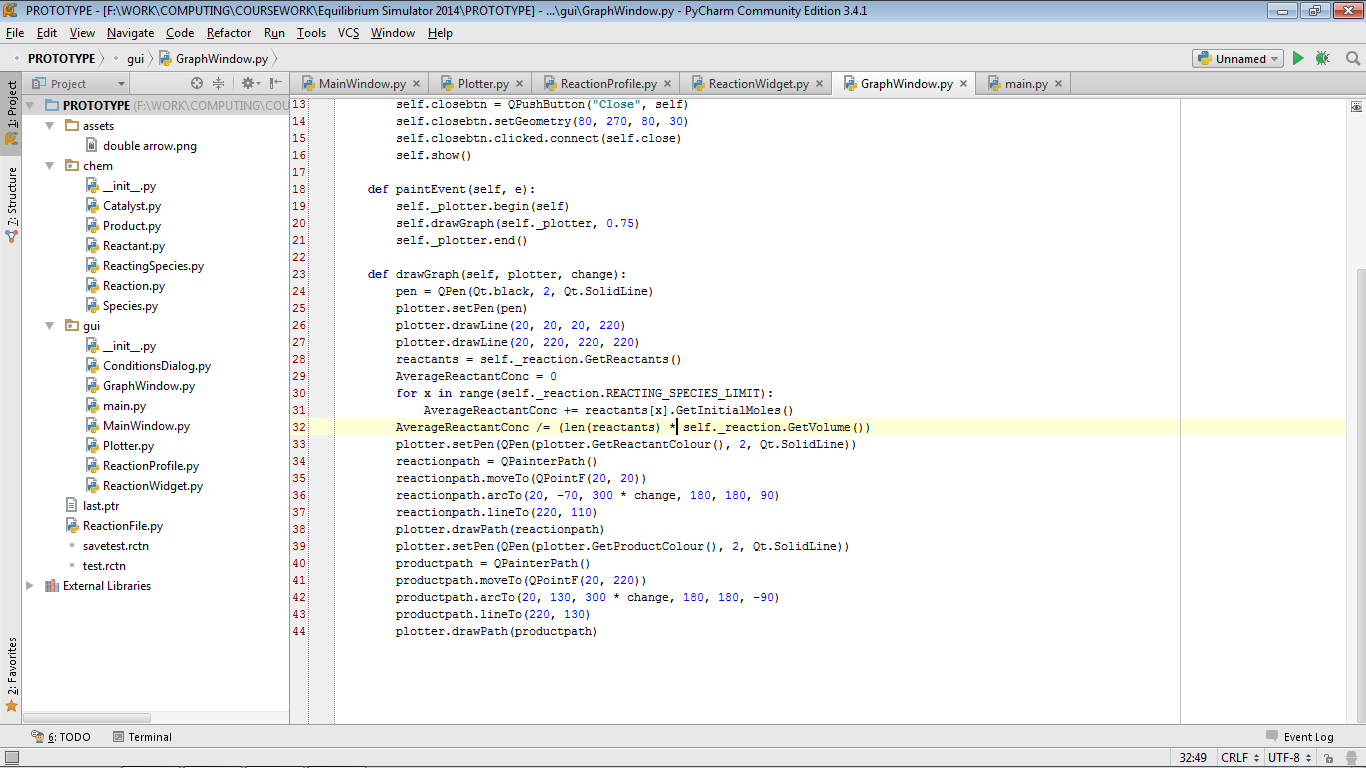
ConditionsDialog.py (part 1)

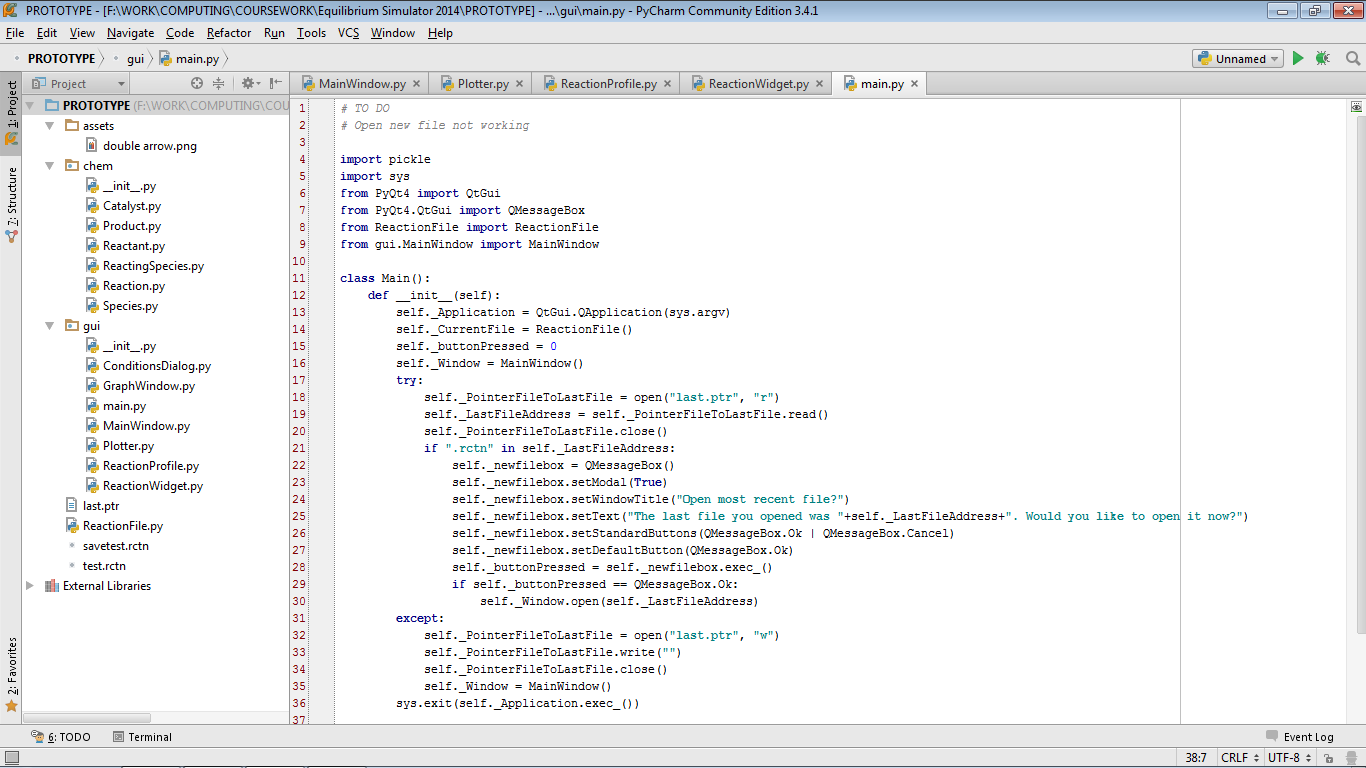
  


ConditionsDialog.py (part 2)

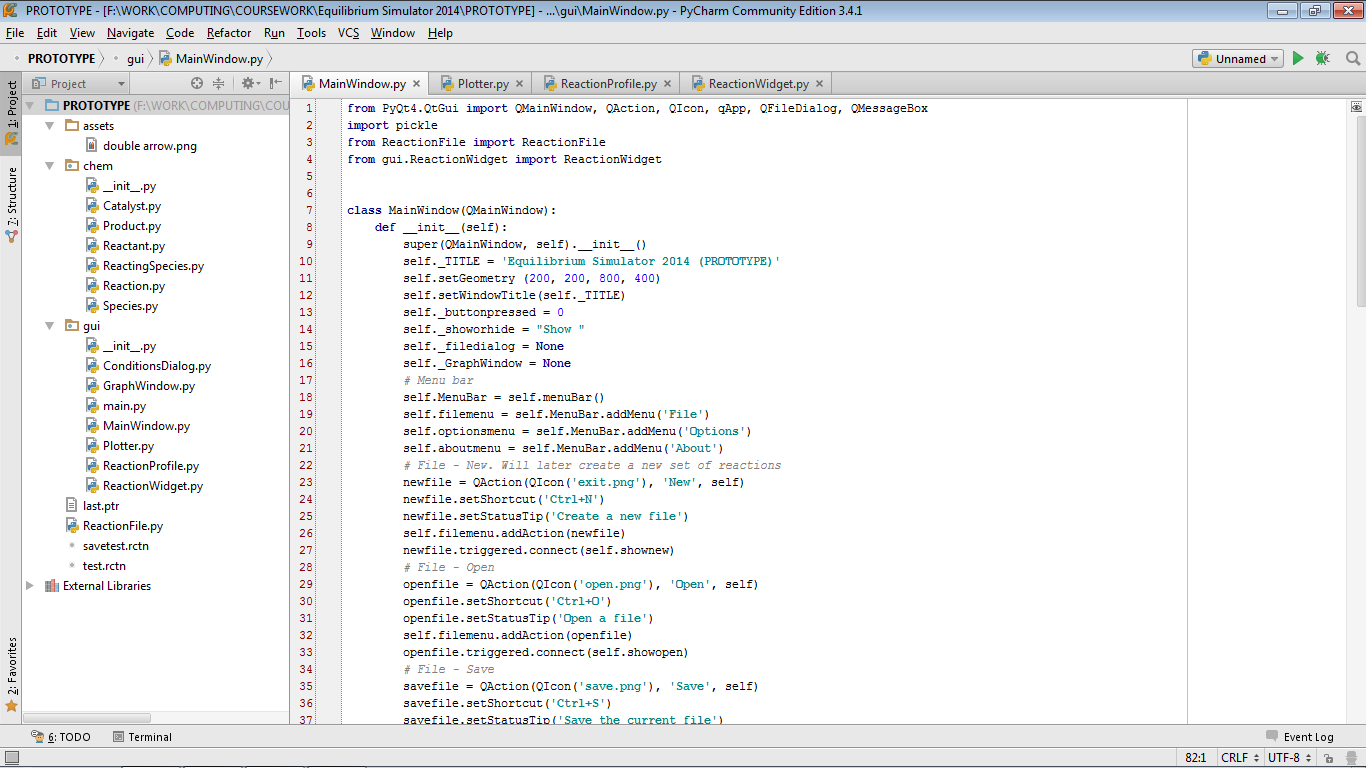
  


GraphWindow.py

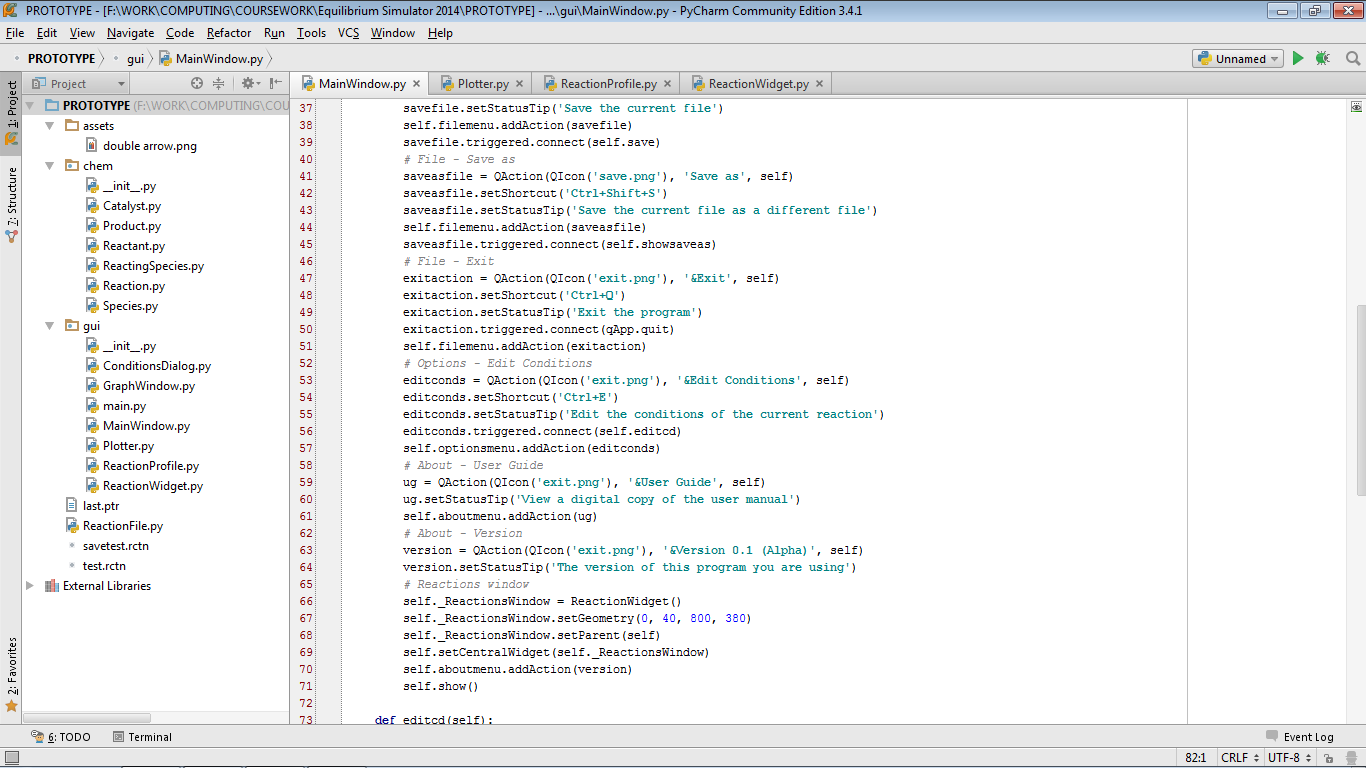
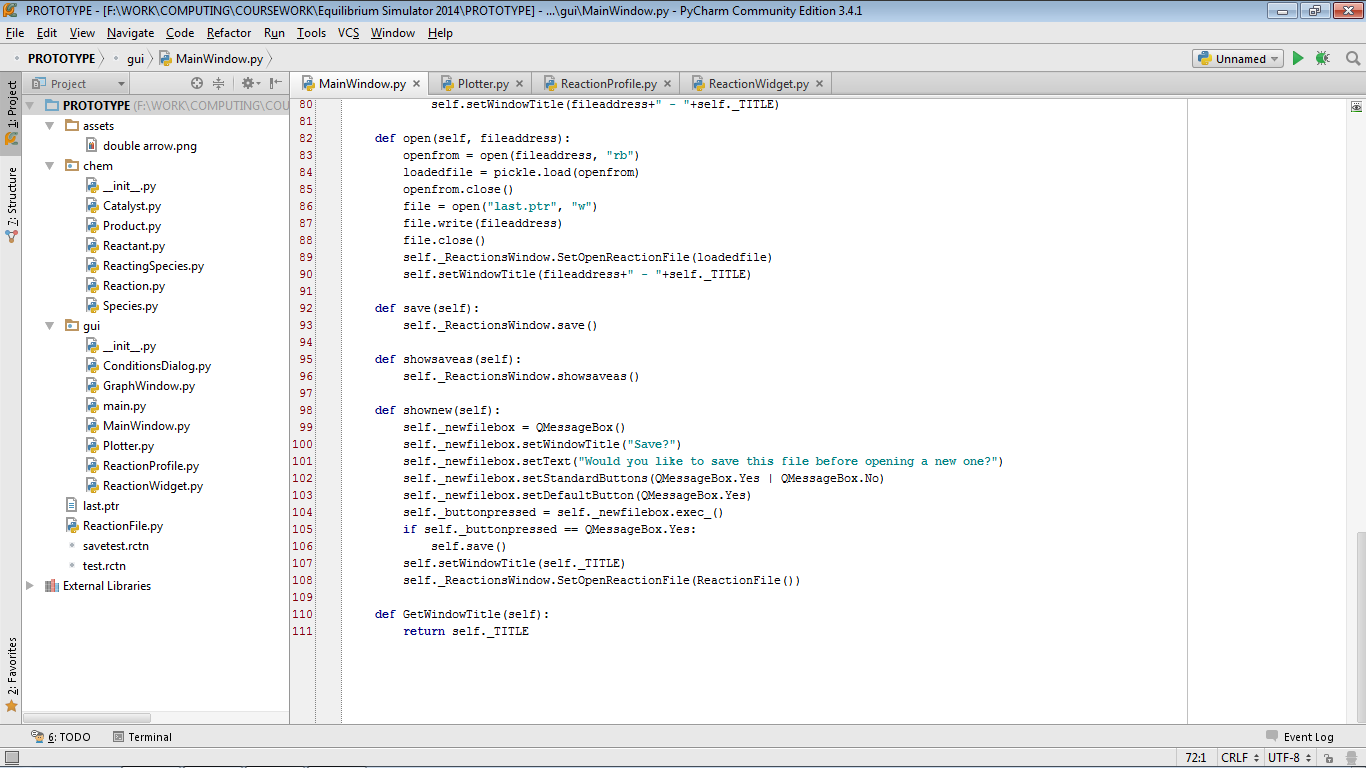
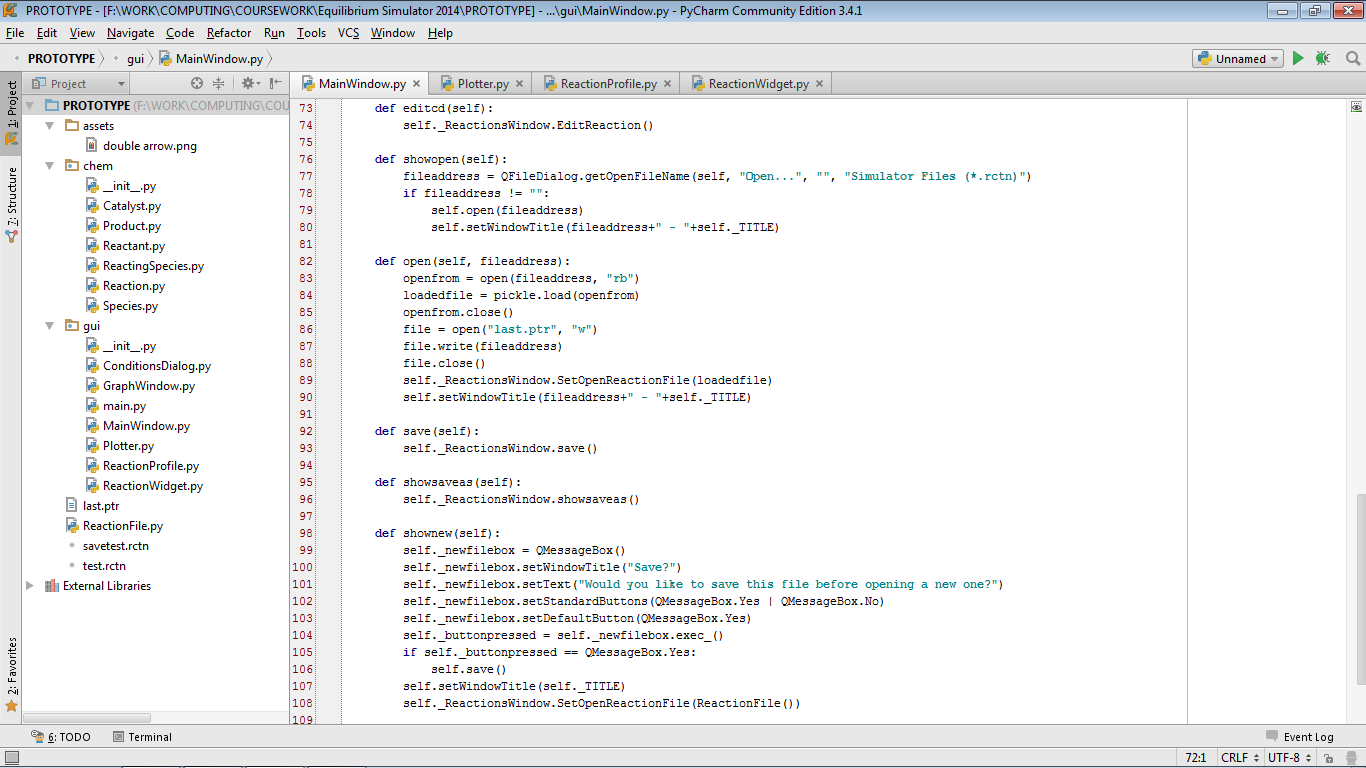
  


main.py  
  

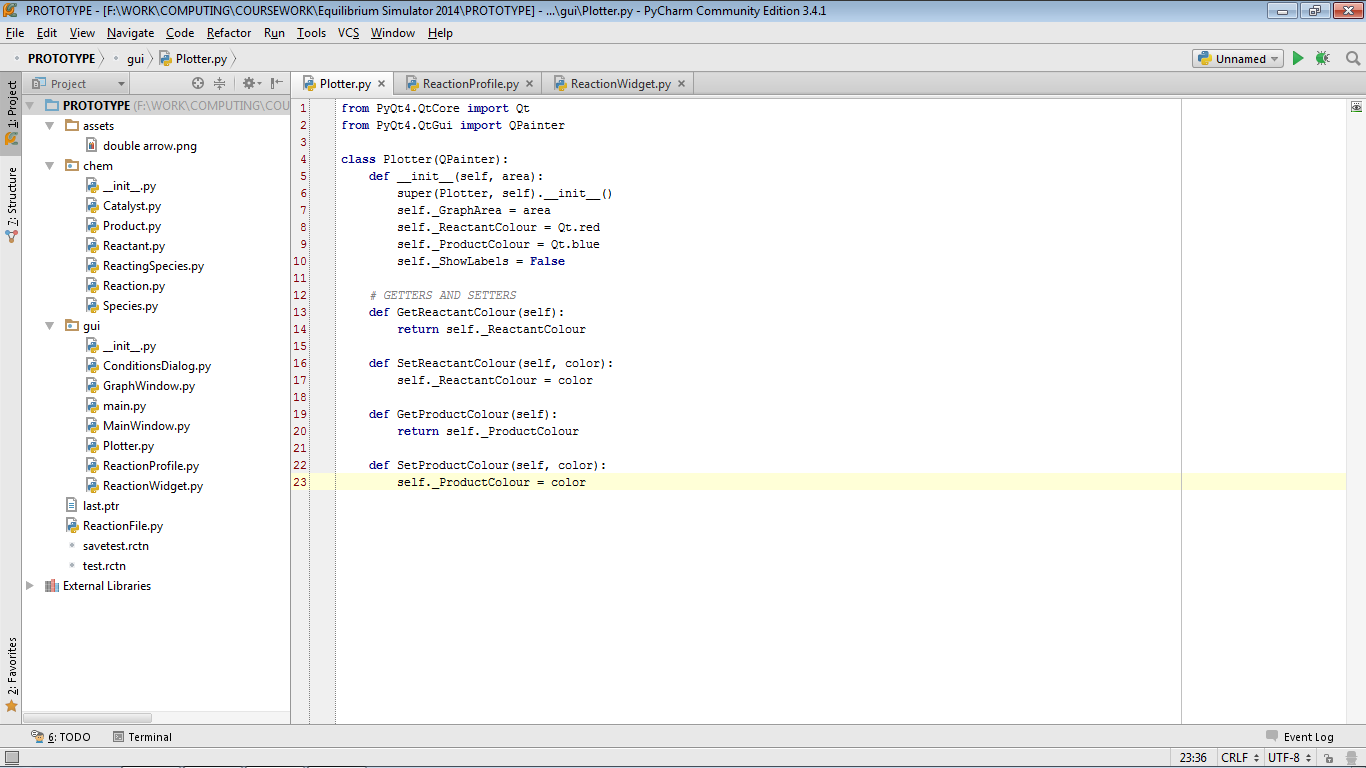

MainWindow.py (part 1)



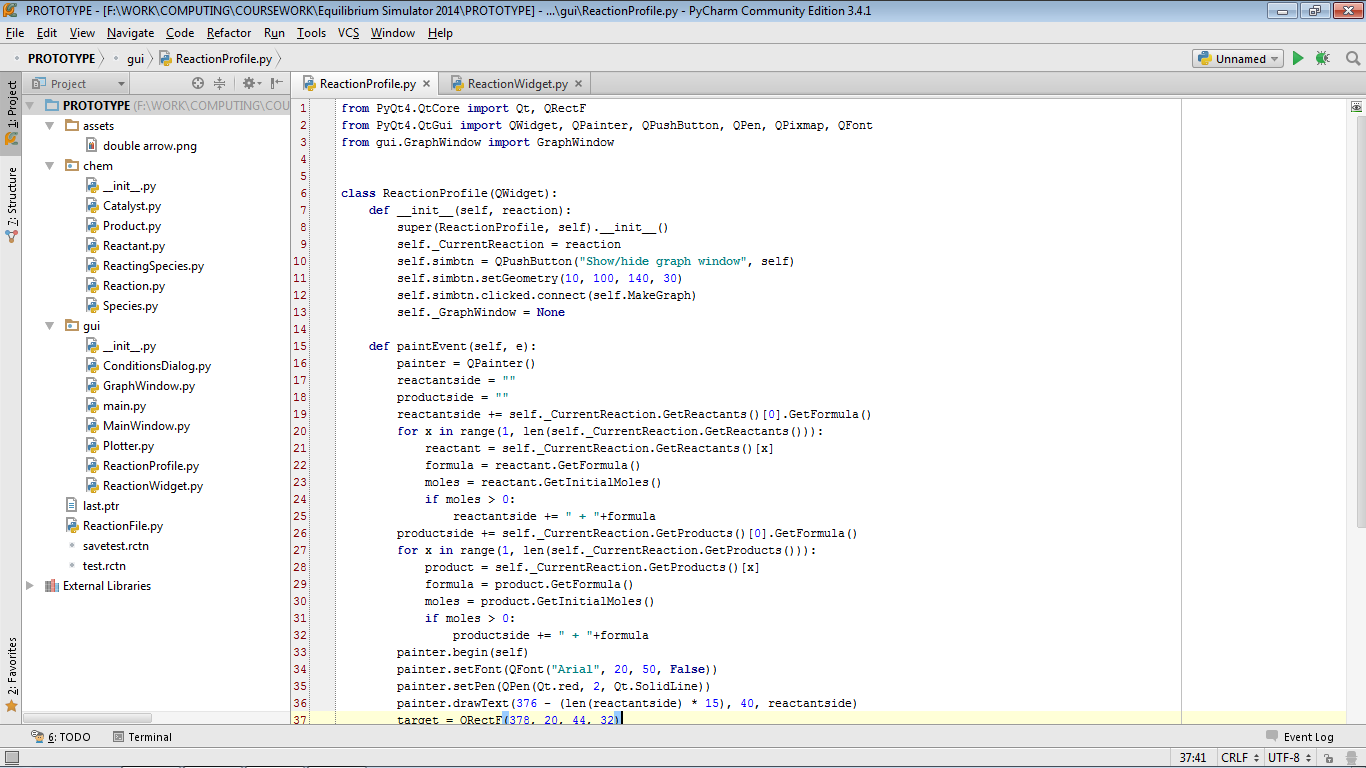
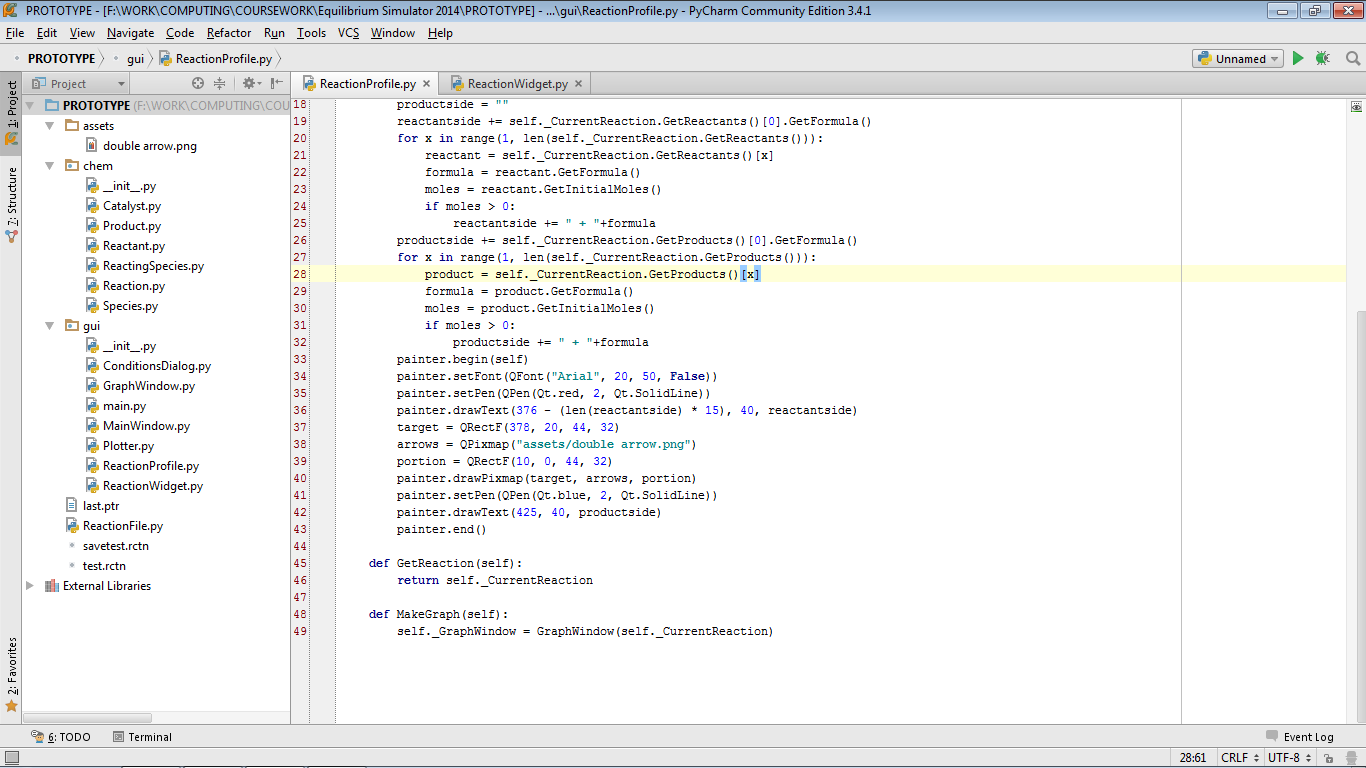
MainWindow.py (part 2)

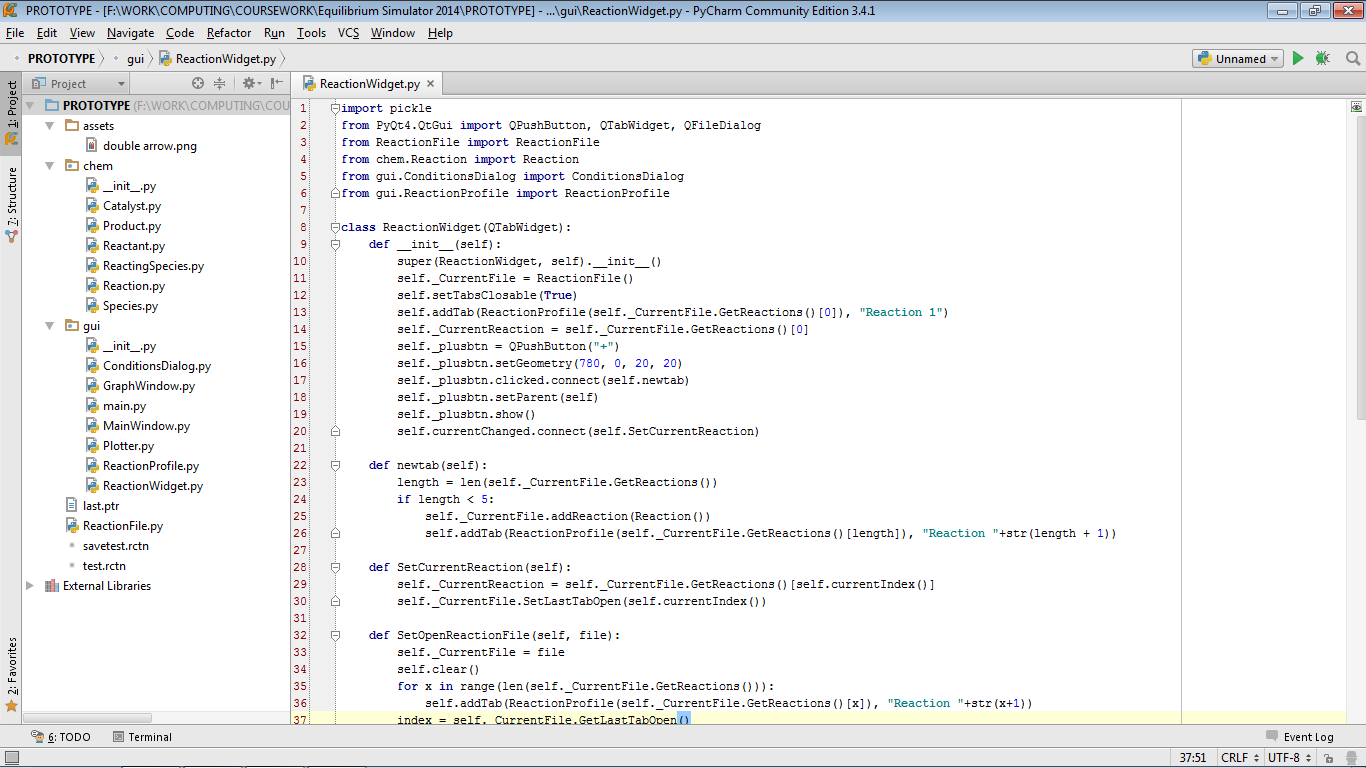
Plotter.py (part 3)



ReactionProfile.py

ReactionWidget.py


ReactionFile.py

