Gas Law Calculator	Grade 11 Chemistry

Handout

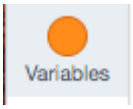
This handout describes in part how to create a calculator app that applies the gas laws learned in gr. 11, using MIT's Scratch as a coding environment.

The way Scratch is structured is that scripts are attached to sprites—individual actors within the game. So we have a sprite for each gas law, which will summon forth sprites associated with the variables associated with that law, and a sprite for a 'calculate' button that will hold the number-crunching script.

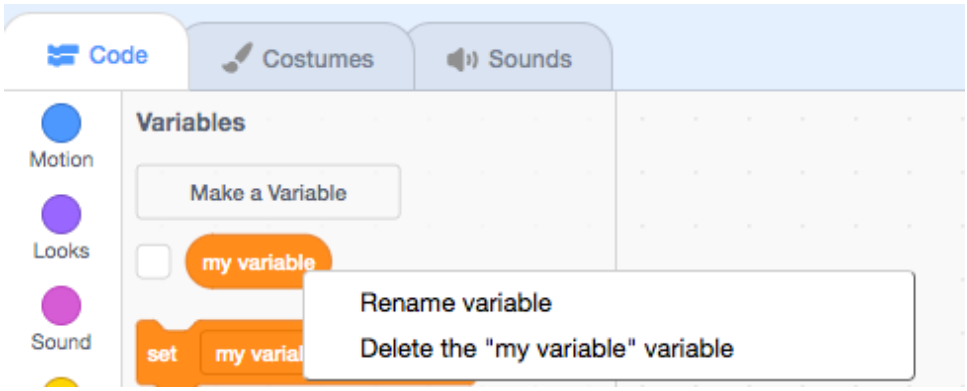
The game can be accessed on Scratch here: https://scratch.mit.edu/projects/451122922

We need only one true variable for this script: the atomic number, N. Every scratch script starts with one sprite (a happy cat) and one variable (called my variable). Let us begin by renaming that variable.

Rename and Create Variables:
It is always good practice, when programming, to define variables before doing anything else. For Boyle's, Gay-Lussac's and Charles' laws, we will need 6: V1, V2, P1, P2, T1, and T2.
Click the orange circle on the left-hand side of the scratch window that says 'variables'.



One variable exists by default. We will rename it. Right click the block labeled *my variable*, and select *rename variable*. Name it *V1*.

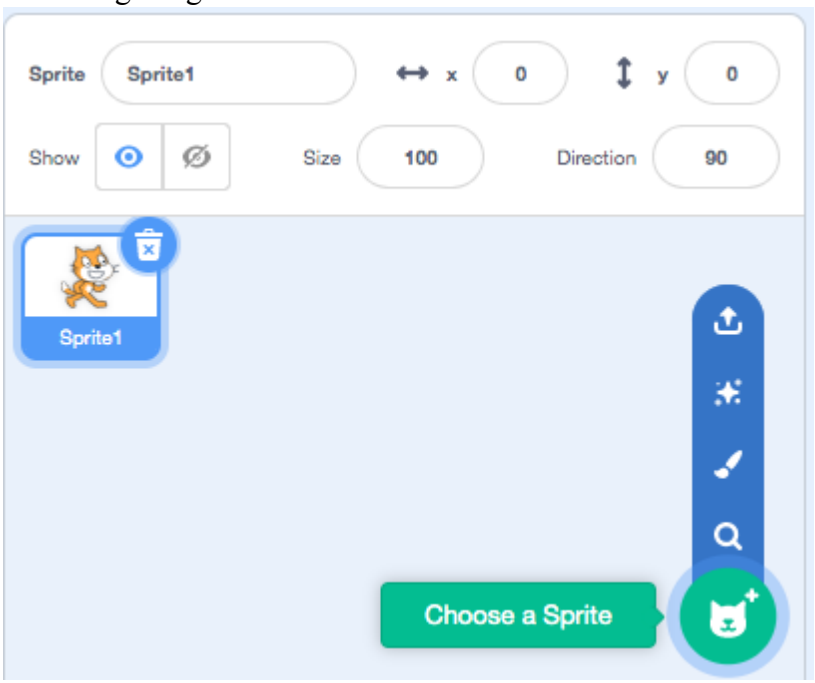


Then click the *Make a Variable* box and create the remaining variables.

With variables defined, it is time to set the stage for our game.

Sprites:

We need to create a number of sprites. There is a default sprite, the Scratch Cat; we can delete the cat with the garbage can icon.

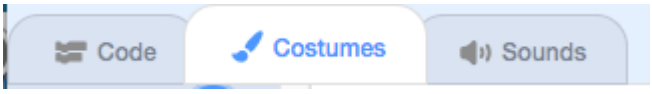


Each sprite will have its own code associated with it, controlling both that sprite and the behaviour of the global variables.

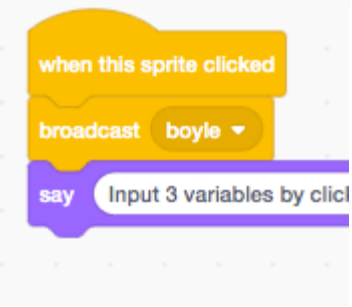
The Gas Law Button:

Each of the 'gas law' buttons are incredibly similar, so we will only deal with Boyle's law here. It can be duplicated to create buttons for Charles' law and Guy-Lussac's law.

Create a new sprite; we used button-3. To add text to our button, we first selected the sprite by clicking on it in the sprites pane (highlighting its frame in blue) and then clicking the costumes tab in the upper left:



Use the text tool to add text to your button in the font of your choice. You can also get more creative if you wish, even uploading an image from your computer.



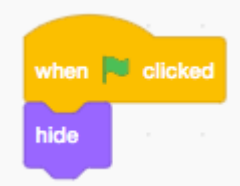
Programming the button is fairly simple. At the start of the program (denoted by when the green flag is clicked) we want to make sure all the variables associated with this gas law are hidden and set to some obviously non-physical value, like -1.
A second script is created for when the button is pressed (to scratch, *when this sprite is clicked*). It is to broadcast an *event message* that other sprites can use to trigger their behaviour. We have to create this event message using the drop down menu. We will also have the sprite *say* a reminder of what the user is to do: input 3 variables, by clicking on their buttons.
(Note: the only difference between this sprite and other gas laws would be the variables controlled and the name of the broadcast.)

The Variable Buttons

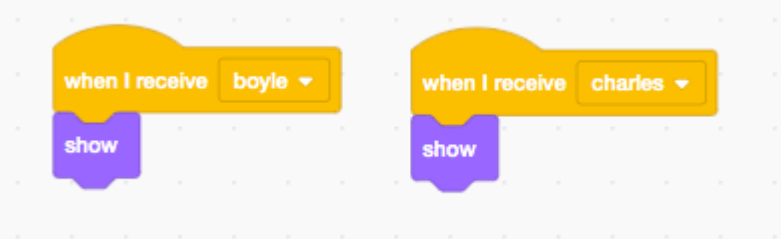
Again, because they are virtually identical, the buttons associated with each variable will not all be dealt with. We will only discuss the button for the variable V1.

As before, modify the sprite using the Costumes tab paint tool or upload your own.

The first thing is to make sure the button initializes as hidden:



And shows itself when either of the applicable gas law messages is sent out.



(Note that these are all separate, unconnected, and independent scripts!)

Finally, we require a script to control the behaviour of the button when it is pressed: we wish to

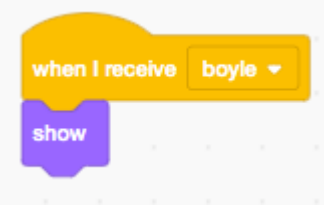
(note that one will want to place the now-showing variable somewhere prominent; in the example it is dragged so it sits overtop the button.)

The Calculate Buttons

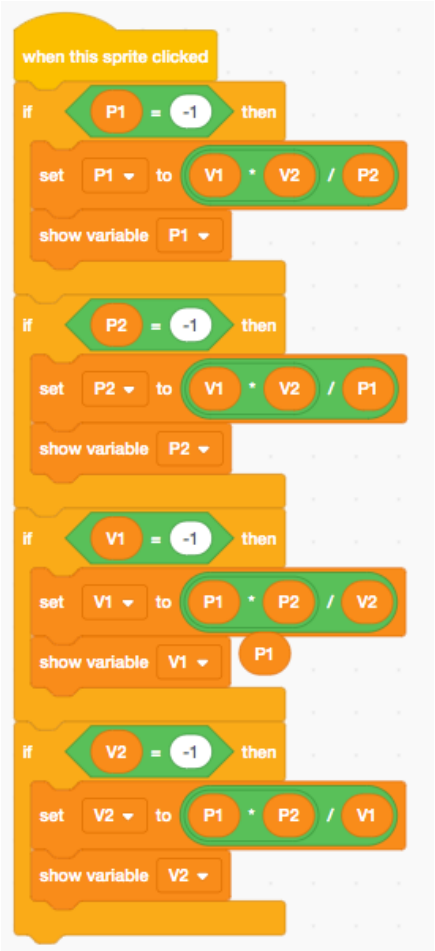
Though it is not visually apparent in the demo app, each gas law has its own sprite to provide a calculate button hold the scripts for its mathematics. Once again, we will treat with only Boyle's Law, as there are no major differences.

Again, we do not want the button to show when the program starts, so it is to hide at startup:

And show itself when it receives the message:



The calculations are not, of themselves, at all complicated. The only fly in the ointment is making sure the *right* calculations are made. That is why the program is set to initialize each variable at the value of –1: so we can pick the variable the user did not input a value for. The variable the user did not input a value for is the one we wish to calculate. Hence the code is a tower of IF statements, checking each variable in turn if it is –1. If, say, P1 =1, then it is calculated by Boyle's Law, and the variable is displayed. Otherwise the code checks P2, and so on.



Questions for those more interested in coding than chemistry would be:
What happens if the user types –1 as one or more input?
What happens if they type regular numbers for all 4 inputs?
Do you think that meets best programming practices, and if not, how would you change this code?