SCIENCE NORTH SCIENCE NORD
SUDBURY, ONTARIO, CANADA

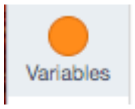
Electron configuration in Scratch	Grade 12 Chemistry
Handout	

The way Scratch is structured is that scripts are attached to sprites—individual actors within the game. We will thus require a button-sprite to increase the atomic number (N+1), to decrease it (N-1), and 92 individual electron sprites. (Fortunately, Scratch allows one to duplicate sprites, so this is not quite as much tedium as it sounds.) The actual action of the program is simple: as the atomic number increases or decreases, electrons appear or disappear, colour-coded by orbital, in their proper order according to the Aufbau principle.

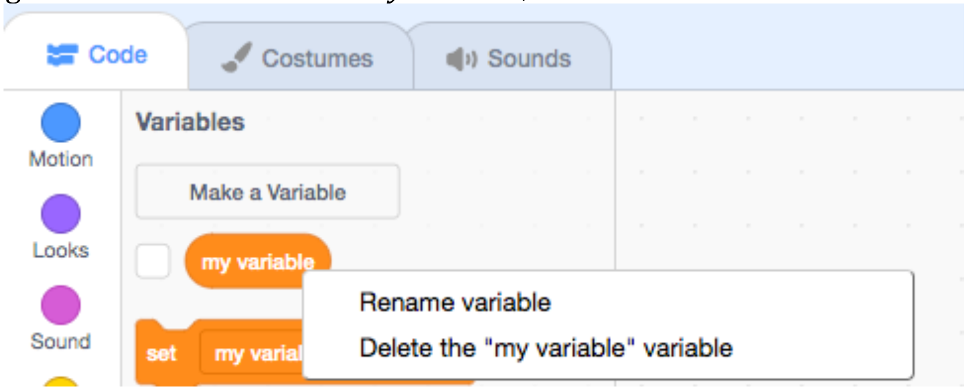
The Scratch project is available here: https://scratch.mit.edu/projects/450593118

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.

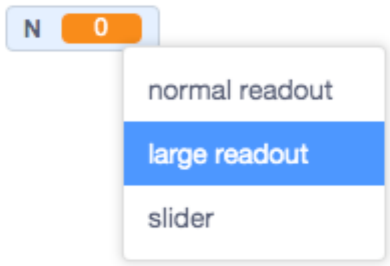
Rename Variable:
Click the orange circle on the left-hand side of the scratch window that says 'variables'.



Right click the block labeled *my variable*, and select *rename variable*. Name it *N*.



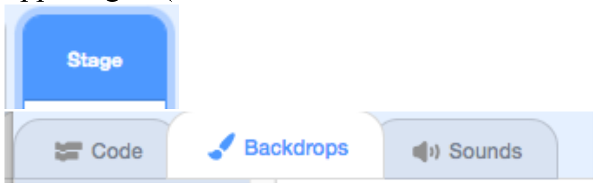
Click the check box next to N to set the variable N as visible to the player. Now in the play window in the upper right, you should see . This will remain visible to the player. To change it into the form seen in the demo game, right click that box and select large format:



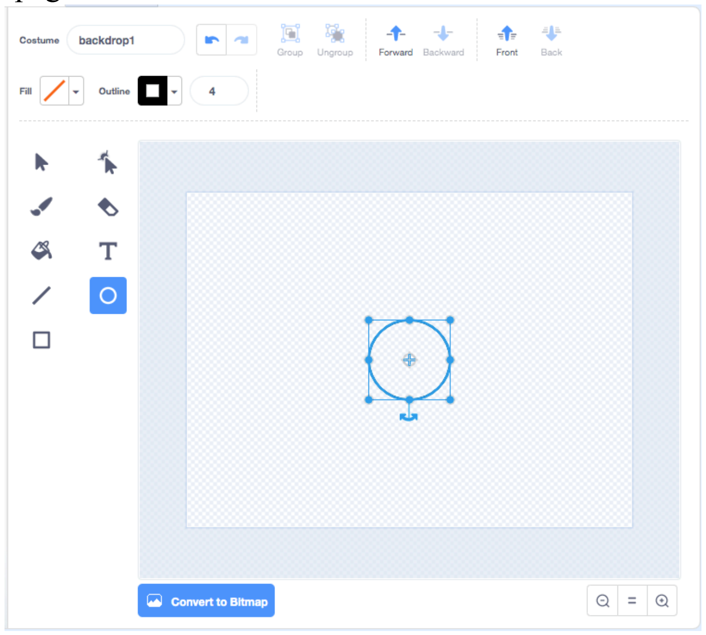
With variables defined, it is time to set the stage for our game.

Backdrops
The 'stage' upon which our game plays out is called a 'backdrop'. You can think of the backdrops as a sort of visual variable: which backdrop is in use is certainly treated by the code as a variable, so we would like to define them first.

We wish to have 7 backdrops: one for each energy level in the atom. You can create these backdrops in whatever program you'd like and upload from your computer, or use the 'paint' tool to create a simple one. Do this by clicking on the 'Stage, in the lower-left, and then selecting the backdrops tab in the upper right. (or use the button in the absolute lower-left to pick a backdrop).

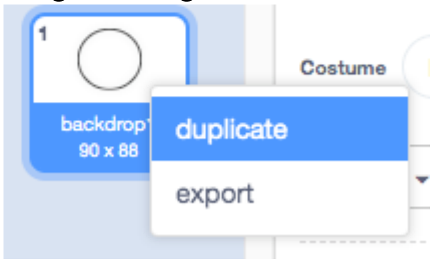


Using the circle tool, for this example we'll make the first backdrop. Simply select transparent (the red slash) under the fill colour, click the circle tool, and make a small circle. Drag it so it is centered on the 'page'.

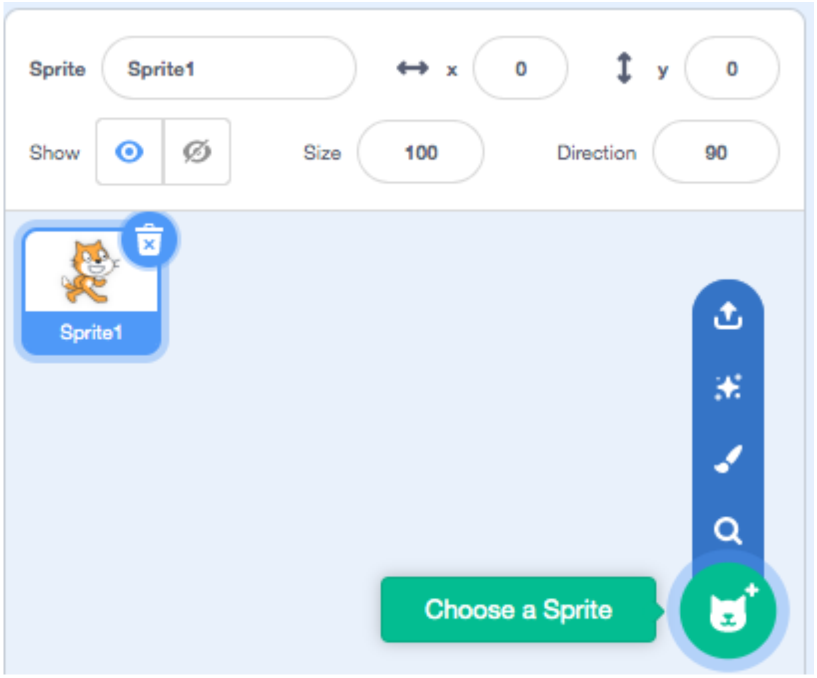


Now to create the second backdrop, right-click backdrop1 in the left-hand column, and select duplicate. Create a slightly larger circle, centered on the first. Repeat this process 7 times until you have 7 circles (or ovals) totally filling the screen.

Drag the orange box of the N variable into the center of the 'atom' thus created by the backdrop.

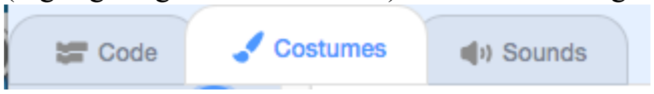


Sprites:
We will create three new sprites: an electron, and 2 buttons. To create a sprite, click in the bottom-left corner and select 'chose a sprite'. This example used the 'ball' sprite for electrons, and a button sprite for buttons, but feel free to get creative. We can delete the cat with the garbage can icon.

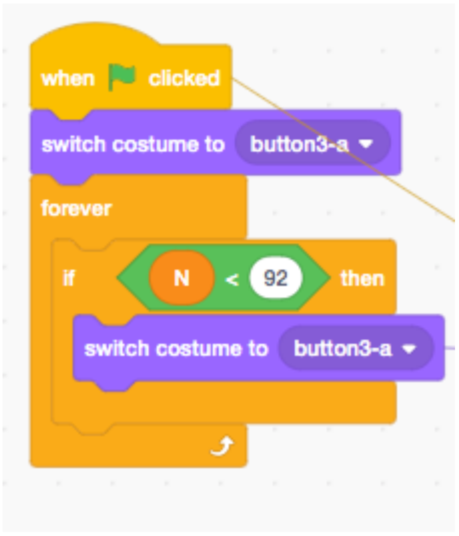


Each sprite will have its own code associated with it, controlling both that sprite and the behaviour of the global variable, N.

The N+1 Button:
First we will deal with the button that increases atomic number, N. Createa 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:

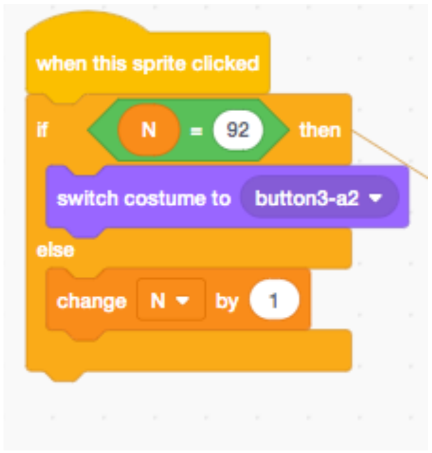


(this is where you had edited the stages, previous).
Use the text tool to add text to your button in the font of your choice. We also wish to create another *costume* for our sprite (similar to the backdrop of a stage) for when the button is disabled (when we run out of room for more electrons, for example.) Again, simply duplicate the existing costume, just as we did with the backdrops.



```
when [flag] clicked
switch costume to (button3-a ▾)
forever
    if <(N) < (92)> then
        switch costume to (button3-a ▾)
    ↺
```

Programming the button is fairly simple. We will set a maximum value of N=92, as this will allow us to cover all of the naturally-occurring elements. There are two portions to the code. The first is a loop, which engages when the program starts. It begins by switching costumes to our default 'N+1' (here we left it named button3-a) and then, on a loop, checks if N is ever less than 92. If it is, the costume is reset.
A second script is created for when the button is pressed. This one has no loops, but a simple if statement—if N=92, then change the costume to the second, disabled version. Otherwise, change N by +1. Simple enough.

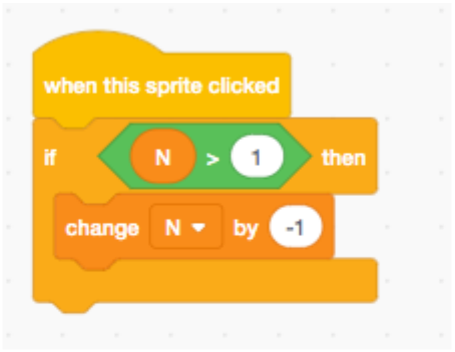


```
when this sprite clicked
if <(N) = (92)> then
    switch costume to (button3-a2 ▾)
else
    change (N ▾) by (1)
```

Note that we could put a script controlling the backdrops (to show and hide the rings representing energy levels) on this button. That it is on the other, N-1 button in this example is wholly arbitrary.

The N-1 Button:

Create a new button sprite to decrease atomic number, and label it as desired. (In our example, it is *N-1,* and was created in exactly the same manner described above for the N+1 button.)

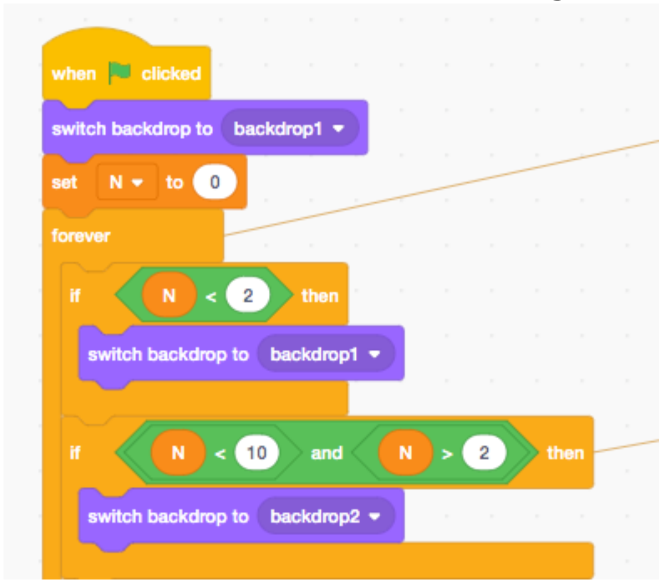


```
when this sprite clicked
if <(N) > (1)> then
    change (N ▾) by (-1)
```

There are two scripts on this button. The first is very simple: When the sprite is clicked, if N is greater than 1, we decrease N by 1.

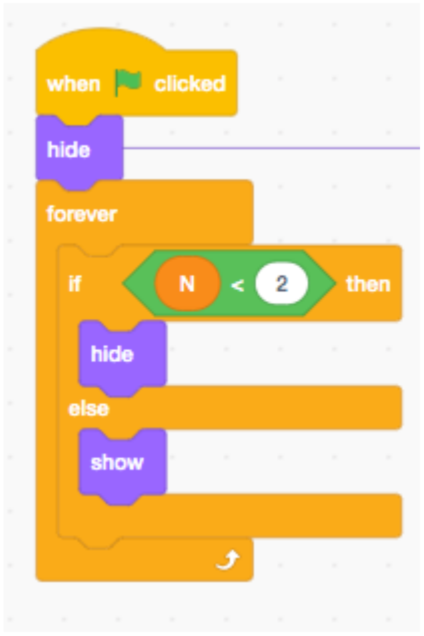
The second has to do rather a bit more heavy lifting. This script was chosen to initialize the program. Therefore, it is a 'when green flag clicked' script.
First of all, it switches the backdrop back to backdrop1—energy level 1, and sets N to 0, resetting whatever the conditions of the program were when it last ran. Then it enters a loop to control the backdrops. This takes the form of a series of if statements. If N<2, then we need backdrop 1. If N>2, then we need backdrop2 (the one with 2 circles) -- but past a certain point, that will not be enough! So, we need to set a value for N to be lesser than as well. The script is not reproduced here for all 7

energy levels, for brevity, and so as not to give the game away. Which N values are used here are a test of the student's scientific knowledge.



```
when [flag] clicked
switch backdrop to backdrop1
set N to 0
forever
    if N < 2 then
        switch backdrop to backdrop1
    if N < 10 and N > 2 then
        switch backdrop to backdrop2
```

The Electrons:



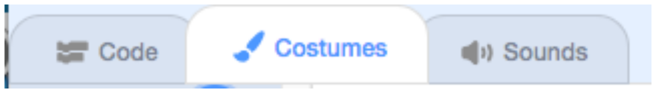
```
when [flag] clicked
hide
forever
    if N < 2 then
        hide
    else
        show
```

We need only program one electron-sprite, and then duplicate it endlessly and place it on the circles representing the different energy levels.

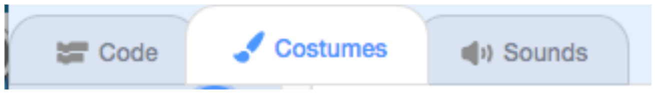
The code for this is fairly simple: when the flag is clicked to start the program, the electron-sprite hides. It then starts a forever-loop, checking the statement IF N<x. When true, the electron hides. When false, it shows itself. X will change for each electron you add to the atom. *1s2*, for example, is x=2. *3p5*, x=17. This is where you will demonstrate your scientific knowledge.

To duplicate a sprite, simply right-click it, and click *duplicate.* Rename the sprite so you can tell them apart. You can use any combination of quantum numbers you would like so long as you can keep track! Note that each newly-duplicated sprite appears at random and must be dragged to the circle representing its correct energy level.

To change the colour of the electron so you can differentiate the orbitals, you need only change the costume of the sprite. This is done with the costumes tab:



Legend
The last thing to do is to create a ‘legend’ for the *s, p, d,* and *f* orbitals. Add sprite that matches what you have used for electrons in each of these orbitals, and use the paint function under costumes in the upper left to add the lettering to each.


Code
Costumes
Sounds