

Atom Simulator

Interactive Graphics

Ibis Prevedello

November 13, 2018

1 Introduction

The goal of this project is to create an atom simulator, where the user can see and interact with any element of the periodic table, with its correct number of protons, neutrons and electrons. Also, for any element chosen, it also presents its relevant information, as shown in figure 1. The idea is to develop a fun and easy platform that can be used by students while learning chemistry, once that what are usually shown during classes are very poor 2D drawings.

This project was developed using three.js, which is a cross-browser JavaScript library and Application Programming Interface (API) used to create and display animated 3D computer graphics in a web browser and uses WebGL.

The source code of this project can be found on GitHub and it can be tested in this link. The project was also sent to Experiments with Google and is still under evaluation.

2 Hierarchical Model

The model implemented is not complicated and thus the hierarchical model is simple. The model elements are basically all centered at the origin, with exception of protons and neutrons that are translated inside the nucleus.

Below is presented the hierarchy used, the vector of shells is taken as parent and the rest of the elements taken as children.

- Shells
 - Nucleus Elements

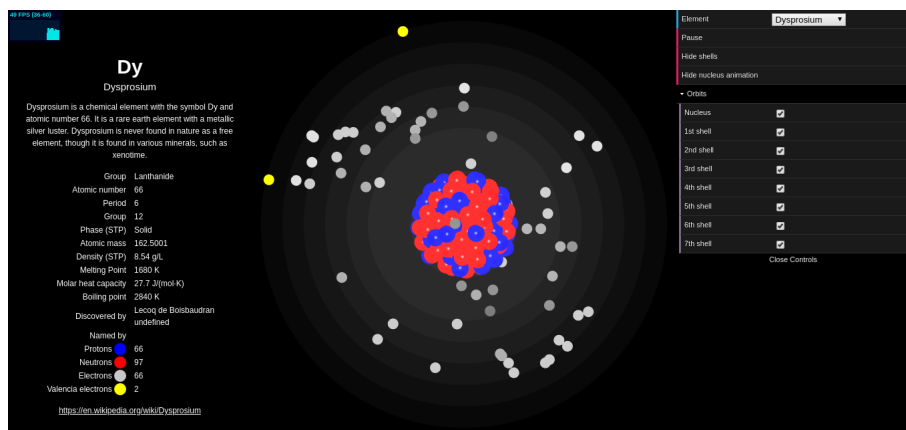


Figure 1: Atom simulator running

- * Protons
- * Neutrons
- Electrons

3 Implementation

Below is presented in detail all the features developed for the project, including the nucleus generation, electron orbits, user controls and information card.

3.1 Nucleus generation

This was by far the most difficult task of the project, at the end the solution achieved was really simple, however the process was very challenging. There is not a formula or a simple algorithm to position n spheres with a maximum radius inside another sphere with a fixed radius R without overlapping.

This problem is known as 'sphere packing problem' and there are some recursive algorithms to achieve the best result, however, because this was not the main focus of the project and something more easy and straight forward was needed, it was decided to implement a simpler version of the sphere packing algorithm.

The implementation chosen does not achieve a perfect arrangement and the spheres end up overlapping a little bit of the edges. For this implementation the internal spheres start with a radius equivalent to fill 1% of the total volume of the container and increase its radius continuously and rearrange itself, avoiding overlapping with each other and with the container till a point where the overlapping achieves a maximum threshold, this process is presented in figure 2.

Also, while they are expanding, they are also doing random walk in order to achieve the rearrangement.

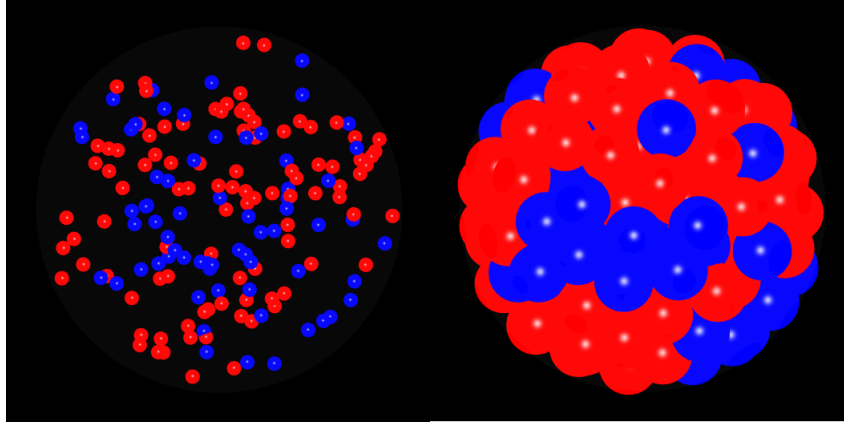


Figure 2: Before and after of nucleus generation process

3.2 Electron orbits

The electrons are placed in its correct orbit obeying the electron configuration, presented in figure 3. The color and the speed are also changing gradually as the orbit is expanding.

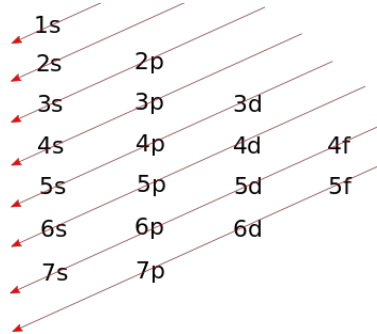


Figure 3: Electron configuration

For the last layer, the electrons are colored yellow, to show that these are the valence electrons, which are electrons in one of the outer shells of an atom that can participate in forming chemical bonds with other atoms (figure 4).

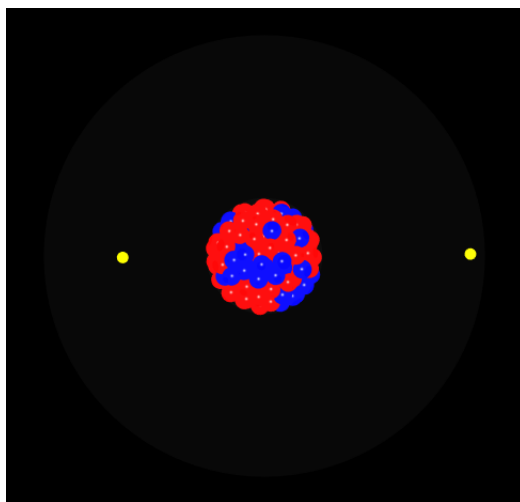


Figure 4: Valence electrons

3.3 User controls

The controls represent one way for the user to interact with the simulation, with the controls one can chose the element to be simulated, pause and continue animation, show or hide the shells, and show or hide the nucleus generation.

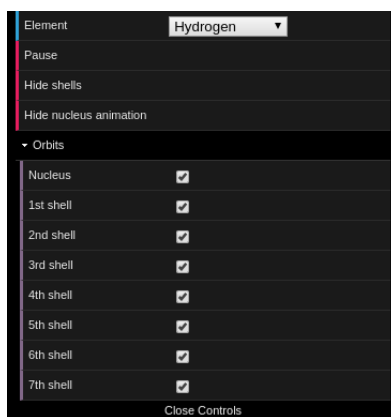


Figure 5: Controls

Moreover, it is also possible to select which of the layers to show, what makes easy to see how many electrons are orbiting each layer.

3.4 Information card

On the other side of the screen the information of each element is shown, as can be seen in figure 6. All the information is located in a json file and is the base of the application, once it is necessary to calculate the number of spheres of each element as well as the electron shells distribution.

Dy	
Dysprosium	
Dysprosium is a chemical element with the symbol Dy and atomic number 66. It is a rare earth element with a metallic silver luster. Dysprosium is never found in nature as a free element, though it is found in various minerals, such as xenotime.	
Group	Lanthanide
Atomic number	66
Period	6
Group	12
Phase (STP)	Solid
Atomic mass	162.5001
Density (STP)	8.54 g/L
Melting Point	1680 K
Molar heat capacity	27.7 J/(mol·K)
Boiling point	2840 K
Discovered by	Lecoq de Boisbaudran
Named by	undefined
Protons	66
Neutrons	97
Electrons	66
Valencia electrons	2
https://en.wikipedia.org/wiki/Dysprosium	

Figure 6: Information sheet

3.5 Libraries used

Below is the list of third-party libraries used for this project.

- **dat.gui.min.js**
- **OrbitControls.js**
- **stats.min.js**
- **three.js**
- **three.min.js**

4 Conclusion

Since the beginning, the idea of this project has been to implement an interactive and easy way to see and learn about chemical elements, and it is interesting how

much difference it can make in the learning process of a child with basically no effort. This project took me basically one week from start to finish, considering learning WebGL and revising some chemistry lessons.

There was basically one challenge of solving the sphere packing problem, which ended up taking more time than I intended, however the goal was achieved and the final result is very satisfactory.