A Tiny World: Atom

CHUNG AN, CHEN, YANG, LIU, and LINXI, TAO

ACM Reference Format:
Chung An, Chen, Yang, Liu, and Linxi, Tao. 2018. A Tiny World: Atom. *ACM Trans. Graph.* 37, 4, Article 111 (August 2018), 1 page. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Prior to the electron cloud model, there are a lot of attempts on building a theoretical model for an atom. Most of them, for example, the commonly known Rutherford-Bohr model, perceive the electrons as if they are planets orbiting the sun — a non-changing centripetal movement. The electron cloud, however, models an atom consisting of a small, yet massive when put aside to its electrons, nucleus surrounded by a non-deterministic cloud of electrons.

In the electron cloud model, electrons can theoretically be found anywhere in the space. However, they can be found more often in some regions, which we usually call them orbitals despite no orbital motions of electrons are being held, than others. A Carbon atom has two orbitals, where the inner layer contains two electrons and the outer layer contains four, and electrons are less likely to be found in between the two layers.

This project aims to simulate and render a Carbon atom's presence in a three-dimensional space. The simulation will be grounded in the more accurate electron cloud model, shown in Figure 1 below. An electron cloud model depicts the occurence of an atom's electrons by a density map containing numerous sparse dots. In any region, the density of the dots draws a directly proportional relationship to the probability of an electron being present. To illustrate this model, We leverage a recently-built parallel programming language, namely **Taichi**, as our main development framework.

2 SOFTWARE

Taichi is a high-performance Domain-Specific Language for computer graphics applications. **Taichi** is designed towards performance, portability, spatially sparse computation, and differentiable programming. Despite inheriting most of its syntax from Python, **Taichi** does not carry over the downsides, for example, the slow computation speed, from Python.

Taichi is a higher level language such that it can be made to run on multiple mainstream rendering backends such as OpenGL, CUDA, and Metal. Using Pythonic decorators such as @ti.kernel and @ti.func brings the succeeding block of code into **Taichi**'s scope, where functions are naturally parallelized and differentiable on CPU or GPU devices. Two main data types in **Taichi** are Primitive Types and Compound Types. Primitive Types are numerical data types used by backends, while Compound Types are user-defined types composed of multiple members. To communicate between **Taichi**'s scope and Python's Scope, we use a global variables object called Field. They are commonly multi-dimensional array of elements of the following: a scalar, a vector, a matrix or a st, and can be either dense or sparse. Similar to a NumPy ndarray object, a field has a data type and a shape.

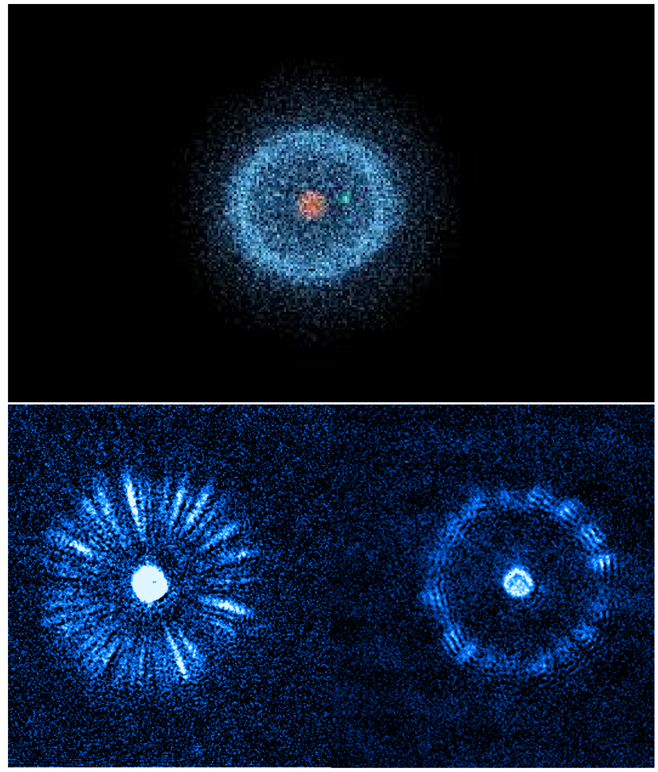

Fig. 1. Atoms and Their Nucleus and Electron Cloud

Authors' address: Chung An, Chen; Yang, Liu; Linxi, Tao.

0730-0301/2018/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456