Virtual Periodic Table

Elliott Gray
School of Engineering
Deakin University
Melton, Australia
ehgray@deakin.edu.au

Abelaash Rajagugan
School of Engineering
Deakin University
Geelong, Australia
arajagugan@deakin.edu.au

Induka Punarjeewa
School of Engineering
Deakin University
Geelong, Australia
pkulathungaarac@deakin.edu.au

***Abstract*— This paper focuses on delivering the various method used on developing a Virtual Periodic Table to present the various properties of the elements and to perform various tests to deliver knowledge on how these elements react, this experience is aimed at lower secondary students. The requirements such as the hardware and software are discussed along-side the technical requirements and the features in Unreal Engine 5.03 the individual must learn and investigate in-order to develop the environment. The conclusion involves the discussion of what was achieved.**

I. Introduction

The Virtual Periodic Table experience is a virtual reality experience designed to showcase the properties of the 118 chemical elements. The experience is aimed at lower secondary students such as year 7s or 8s and is intended to be a fun, interactive way for them to learn about the properties of elements in the periodic table.
It also delivers knowledge about basic chemistry, such as how Group 1 elements react with water at different intensities as well as how Group 1 elements cause the change of flame color whilst exposed to a Bunsen burner flame. Since it is aimed to be developed in a virtual environment, different types of fictional and non-fictional tools will be implemented to deliver further knowledge on atomic mass, element density and radiation of different elements where applicable. The intended experience is as such that a virtual periodic table will be designed from which the player is able to pick up various elements and direct them through various tests to determine their properties.

II. Hardware Requirements

The hardware requirements to run the experience smoothly include a high-performance PC, VR headset and controller combo. As the unreal project is heavy a disk space of 12.9 GB will be required, the VR experience will need to run in optimum conditions ensuring that the user is focused on the overall learning outcomes and experience, hence a high-performance PC will be required. Quad-core Intel or AMD at 2.5Ghz or higher, RAM 16GB, DirectX 11 & 12 compatible graphics card, with the latest drivers. The recommended VR headset for this project is the Oculus Quest 2.

III. Software Requirements

The software upon which the project is built is the Unreal Engine 5.03 game development environment. The power-full software tool allows the user to create immersive virtual 3D environments and interfaces. The water plugin and Niagara particle system plugins are utilized in order to achieve the effects of the Bunsen burner, Group 1 elements in water, Atomic Mass Analyzer effects and the Hydrogen balloon explosion effect. Microsoft excel was used to generate an element property database based on existing real-world data for the elements. Blender was used in the generation of 3-D models and assets. To run the experience in Virtual Reality the oculus app o PC was used to connect the Quest 2 VR Headset.

IV. Artistic Requirements

A. *Visual Style*

To create an engaging visual experience throughout the project a consistent visual style was chosen; this style was that of a Science Fiction laboratory. This style allowed for the addition of several artistic liberties with creating fictional tools used to measure the various elemental properties, something that is done through use of multiple measurements and calculations in real life.

B. *3D modelling in Blender*

Several custom-made 3D meshes needed to be made for this project, with several tools, the elements, and the periodic table itself being created. In order to create these the 3D modelling software Blender was used. There were a great number of requirements and issues throughout the creation of these models, with the need to learn this new program and all its navigation and features.

The first asset to be modelled was that of the gas canister for use in each of the gaseous elements. This asset took several iterations to be adequate for use in game. The first steps were that of sketching a concept for the canister, the sketch referenced existing gas tanks, including that of propane and argon tanks used in welding. Once this sketch was complete modelling could begin, a cylindrical base mesh was imported using Blenders in built in meshes. This cylinder could then be scaled to a correct size, before using the face select tool and inset function to create rings on the top and bottom surfaces, these rings were then extruded and utilizing the bevel function they had extra details added. In order to create a smoother model a subdivision surface modifier was added to the object; this modifier creates a new surface mesh that rounds all the edges of the model, creating a surface that does not have jagged curves. This change in smoothness can be seen in the below figures.

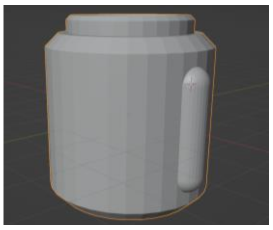
Figure 1 : Canister model Prior to smoothing

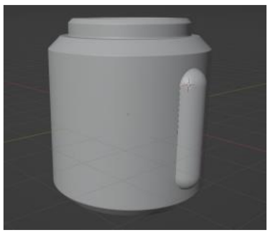
Figure 2: Canister model After Smoothing

The other elements used similar features to create a basic metal bar, a rounded uneven cube for the soft metals such as the alkalis, a jagged rocky model for use in the nonmetallic solids, and a torus and sphere for elements that do not have a known physical state. The metal bar makes use of the bevel tools, the soft metal cube was created by importing a cube and using the bevel tool along random edges before using the subdivision surface to smooth it out. The jagged rock model was created by first importing a cube mesh then scaling it to be rectangular, the entire cube was then selected and subdivided to create a greater number of vertices, these vertices could then be selected, and have their locations randomized creating a randomly jagged surface that was adequately rock-like.

The tools were created using the same tools within Blender, however they made use of much more compounding tools, as well as many extrusions. The Geiger counter was the first tool created, it strongly made use of the face inset and extrusion tools. Once the basic shape was blocked in with insets and extrusions the bevel tool could be used to create rounded corners for the detector end as well as the body of the tool and the handle section. The density and atomic mass tools were made in much the same way.

C. *Texturing and Materials*

The materials used in this project were a mix of premade assets found on the unreal marketplace, and custom-made textured materials primarily used on the tools and gaseous elements. When first creating these textures on the tools they were severely stretched and deformed, it was found that to correctly display materials on the models they first must have the UV maps of the objects cleaned, these UV maps act as an unfolding of the object allowing a flat texture to be painted onto the surface. Once a texture was created it could then be imported to unreal engine and using the material blueprint system turned into a material, this material could then be altered using various parameters to create an instance that is applied to the mesh with the editor. For each of the gaseous elements a separate material instance was used with each instance altering the texture parameter to include the periodic symbol for each element. The example of an unwrapped UV painted with the desired texture image can be seen below.

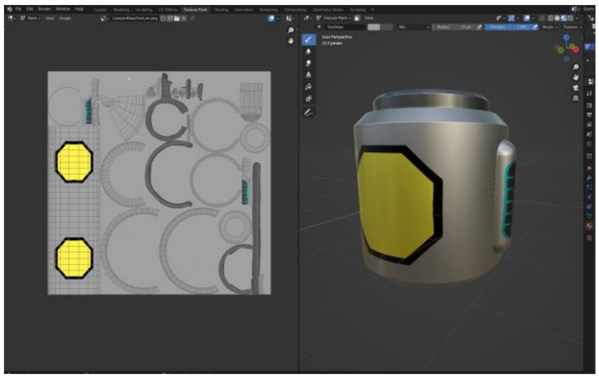

Figure 3: UV unwrap, and painted texture applied to canister model

D. *Environment Design*

To create the physical environment assets from the unreal marketplace were used, as these assets were of a high quality and allowed the focus of the 3D modelling to be on specialized assets. The environment also had to be scaled to comfortably allow for play when in a VR headset. This scale was found by changing the size of a floor panel in the editor and then checking to see what it looked like in VR. Another aspect of the environment was the lighting, as the room is intended to be a laboratory, a clean bright light was wanted. Initially Rect lights were used, however these were found to have too little effect on the ceiling of the environment, causing it to be pitch black. Instead using point lights allowed for a more realistic form of lighting. These point lights were also placed strategically in front of models of fluorescent light fixtures as real-life fluorescent tube lights tend to have bright spots, meaning that the lights seemed quite realistic.

V. Technical Requirements and Programming Skills

A. *Unreal Engine Requirements*

The utilization of a blueprint visual scripting system is a skill that must be learnt in order to add functionality to the actors or any object in game. This technique utilizes node-based interface to create events within Unreal Engine. Connecting nodes, events, functions, and variables with wire will allow the user to create complex gameplay scenarios.
Widget Blueprints are used to create a gameplay HUD with various elements included these elements can be set up to read variables from other blueprints. This is needed to create a layer of guidance and navigation; it would also be useful to the end user for information and methodology.[1]
Unreal Engine also has various plug-ins that can be used to achieve various effects and functions such as custom water bodies, converting cascade systems to a Niagara system etc... It contains a collection of code and data from which the user can enable or disable in the Editor, this plugin feature can be used to add runtime gameplay functionality, upgrade, or edit existing built-in features in Unreal Engine and improve the potential of the Engine. Since the Virtual periodic table along with other types of equipment is built upon a sci-fi laboratory environment, water bodies can be developed using the water system plugin for various experimental needs, also the Cascade to Niagara System plugin can be used to convert Cascade particle system to a Niagara system.[2]

B. *Element Data Table*

A CSV file which was built using Microsoft Excel containing the properties of each of the elements in the order of their Atomic Number was imported into the Unreal Editor as a Data Table. The import required the creation of a defining Structure blueprint which should mirror the headings of the properties in each of the columns of the CSV file, the data type of each property column was also defined in the blueprint. The Get Data Table Row function with the reference to the created Data table and required Row was used inside the target blueprint to extract the property information using a Break structure node. This enabled the extraction of the properties based on the element of interest.[3, 4]

C. *Periodic Table Blueprints*

To ensure that the experience was populated with an adequate representation of the periodic table a blueprint was created that could quickly and easily create a representation of all 118 elements. In order to accomplish this a single blueprint was created, by utilizing public variables each instance of this blueprint was able to have an individual atomic number variable applied to it. Using this public variable, the elemental data table can have a particular row broken down into its individual variables for that specific atomic number. Once broken down the code then checks for the “MeshType” of the element, “HardMetal”, “SoftMetal”, “Gas”, “Rocky”, “AtomicMetal”, and “Unknown”. The blueprint the uses a switch statement to determine what type of the element is, based on the switch statement a particular static mesh is set as the mesh for the instance. The code then checks the name of the element in another switch statement to determine what material should be applied to the mesh. Each element type has a separate switch for determining the material to be applied. By using this code to populate the periodic table it is incredibly easy to add each element and if needed in future set a unique mesh and material for each individual element.

The periodic table blueprint is also used to choose what element is selected for use within the experience. When the user overlaps the held VR controller with one of the elements it checks what element was selected and subsequently sets each of the relevant variables from the data table to the generic element blueprint, essentially saving that data within the generic element until a different element is selected.

D. *Generic Element Blueprint*

The generic element object is what the user interacts with when in the experience. By grabbing this object, the user can then take it to any of the tools or reaction experiments and complete these actions. The generic element works similarly to the periodic table blueprint, however instead of being multiple instances with a single variable changed the generic element is sent the data from the selected element in the period table selector. The generic element checks for changes in its variables every tick or frame of the game, when the mesh type and atomic number variables are changed the system changes the mesh and material of the generic element using a function consisting of the same base code that sets each of the elements in the periodic table itself.

The generic element also contains the code that activates the tool interfaces for each of the tools implemented. As the element has all the variables set to the selected element it is able to become any element and when read by the tools it will correctly output the properties of the selected element.

E. *Niagara Effect System*

The Niagara system is utilized to create various VFX (visual effects) for different events occurring in-game. This VFX system can be used to also create real-time fluid effects such as water, gas, smoke, and fire whilst also creating GPU ray-tracing collisions, but this feature is still experimental in Unreal Engine. The Niagara system also lets the user create custom modules and the Niagara script editor can be used to set your own logical flow. Since the demonstration of various chemical reactions is needed, the Niagara system will be utilized to create the various types of VFX needed in order to make the VR experience more immersive and enjoyable.[5]

The Niagara System in Unreal Engine was utilized to achieve several effects that aided to visually represent the player on understanding various events occurring in-game, the effects of fire, smoke and sparks that are produced during the reaction of Group 1 elements and water through this system. Initially, a texture material blueprint was created to develop a material that represents fire and smoke with the help adjusting the parameters (such as opacity, particle color, density of the material visually) of two texture samples blended to help develop a fire looking texture, a texture material blueprint for smoke was also develop using the same method by adjusting the parameters of a texture sample to visually represent smoke. These texture materials were then imported to a new Niagara system asset, four particle emitter systems were created inside the Niagara system and the texture materials of fire, smoke and sparks were applied to each of the emitter systems by adjusting their parameters to visually represent an explosion with sparks, fire and smoke.
The created effect was combined with a trigger box to activate the explosion effect when the user dropped the element into the basin of water. The level blueprint of the trigger box was created in such that when an actor in game is overlapped with a mesh asset with the Group 1 element tags created on them to represent different types of elements in the Group 1 of the periodic table. The Niagara system created above is then spawned using the Spawn System at Location function and then selecting the Niagara system asset that was previously built into the function. A temporary sound of an explosion effect is also played to the user through a sound cue using a spawn sound at location function.

An effect for lightning sparks was created by the Niagara system for miscellaneous uses, a translucent white blueprint material was created, then it was imported into a new Niagara system, the sprite renderer was changed to a ribbon renderer giving a ribbon beam like effect to the material. The parameters in the ribbon renderer were adjusted to visually represent sparks, to give the effect more depth and reality, four ribbon renderers were created with different parameters to create an effect so that it looks like multiple sparks fired at randomly.
A water body was needed to represent water for the group 1 elements in the periodic table to react with, hence the two plugins water and the water extras were added into the engine. Through the water system a custom water body was created, and the mesh was overridden with a spherical mesh. The custom waterbody was then scaled and morphed into the shape that was suitable to fit the water-basin and the water attenuation was adjusted to limit the wave distance from the wave source.
The Bunsen burner required a Niagara particle system in order to simulate an effective flame. The first step in the process involved the creation of a Niagara Emitter, the sprite renderer creates particles at a defined spawn rate, the mesh of which is configured to use a generic flame texture. The Niagara system is created next utilizing the emitter, the

parameters such as spawn rate, lifetime and forces were adjusted again to create an effective simulation of a flame. A user-defined variable is bound to the color parameter of the flame. A blueprint for the Niagara system is created with a box overlay, which when triggered by an actor compares the name to pre-set names, if a successful element is found to overlap, the color parameter of the Niagara component is adjusted to simulate the color changing effect for the element. The hydrogen balloon experiment is only available when the chosen element is hydrogen, the Toggle visibility node is utilized to hide it from gameplay. When a collision event is detected by the sphere overlay, an explosion Niagara system is spawned at the actor location of sphere center, the Spawn System at Location node is used to achieve this, an explosion sound is played using the Play Sound 2D node, and the destroy actor node is used to remove the balloon asset from gameplay.

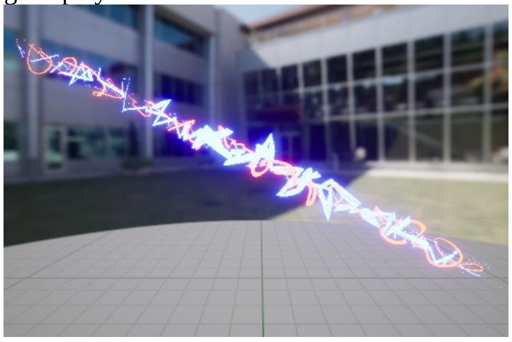

Figure 4: Lightning Effect created By Niagara System

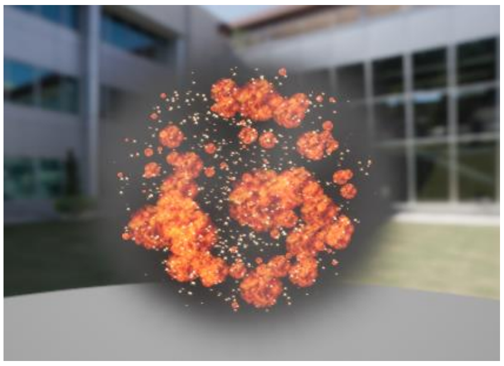

Figure 5: Explosion Effect created By Niagara System

F. Tool Interfaces and Blueprints

Interface blueprints were utilized extensively in the tool-element interactions to generate a start event only on the target blueprint when a successful interaction occurs at the tool blueprint, the blueprint interface allows to create a collection of user configurable functions, the functions may be a simple start event or may be configured have an input value associated with them, which could be later used in the target blueprint for further manipulation of the data. The Atomic Mass tool was simple to implement utilizing interfaces, as a simple box overlap event is used to trigger the blueprint and, the other actor is directly fed into the atomic tool function in the interface to generate a start event at the valid target element.[6, 7]

The Density tool required the use of Ray casting (called Line Trace in Unreal) in order to achieve the effect of measurement based on the trigger event. Line tracing casts a line from a defined start to a defined end location, the start location utilized the position of the nozzle location in world coordinates, and the end location utilized the rotation of the nozzle in world coordinates in order to calculate a vector coordinate that is in front of the nozzle defined by a specified distance away from it. The Line Trace by channel node returns the hit result which could be broken down to obtain the actor reference, and a Boolean value if the trace was successful. An interface function for the Density tool is activated when a successful hit result is found. If the actor is a valid element, an event is triggered at the target blueprint.[8, 9]

The Geiger Counter tool introduced a different set of problems in order to create an effective simulation. It required a constant activation event when in range of a radioactive element. An Area of effect (AOE) was utilized to achieve this, and a new game state was created to define the Spawn AOE function so that all actors with radioactivity associated with them could trigger the event, as the initial plan of the game was to have multiple interactable elements together. A custom event was created in order to bind the interaction to an event delegate for the element this used as activation event when the player actor is within bounds of the AOE , the Sphere Overlap Actors node was utilized to create a circular radius around the spawn AOE actor, the custom event is only triggered if the overlap actor contains the Tool interface for the radiation tool , this event is then passed through to the radiation interface to trigger a measurement. The measurement sent out is dependent on the distance of the Geiger counter from the location of the element with radioactivity, the value is scaled based on this distance. The tool interface triggers an event in the Geiger counter blueprint, which is delayed based on the distance to reduce the number of measurements taken, essentially reducing the number of sounds clicks it produces.[10-13]

G. Widget Blueprints

Widgets have been used in various aspects of the gameplay to provide the user with useful information and feedback. All the measuring tools have been linked to a widget, to ensure that the readings are smoothly conveyed to the user, and as a means of defining what the tool is to be used for. The creation of the widget involved the process of designing a canvas panel, the UI is configurable in a multitude of ways, however, to keep it simple and exciting a simple text box and throbber elements were added. The throbber element provides a means of feedback to the user that the tool is currently active. A cast is done to the associated tool blueprint to retrieve the reading and then output to the text box. The placement of the widget on the tool asset adds to the experience, the process involved adding a widget class and placing it on the designated screen area, the widget class is linked to associated widget for the tool.[14, 15]

In-addition to the tool widgets, an overall gameplay widget conveyed as a digital screen is also implemented, where the generic element blueprint is accessed for the values through casting to actor, if the measurement for the designated property has been taken, then the reading is set onto the textbox, else the text box is set an empty value to hide the reading. A Widget is also implemented on the periodic table to provide the user feedback as to what element is currently chosen which follows the same methodology.[14, 15]

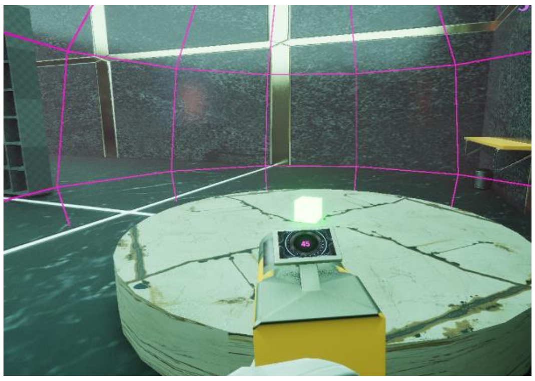

Figure 6 Geiger Counter at AOE event with widget interaction

VI. Summary Of Achievements

The project was able to create an effective VR simulation environment where the user would be able to pick up any of the elements from the periodic table, in-order to test them in basic experiments and utilize tools to measure the properties of the element. The Bunsen burner experiment provides the user a visual interactive platform for the user to gain basic chemistry knowledge about the flame test. The Hydrogen balloon experiment provides basic knowledge about the highly flammable gas. The Atomic tool, Geiger counter and the Density tool provide an interactive twist to gaining knowledge about the element. The use of effects and sounds allow the user to get form feedback on the tool being used. As the Density and Atomic mass of the element in the real world is determined based on several experiments, the project scope is narrowed down so that the user's focus is more towards the measured property rather than the process of measuring the property. The reactions of the group 1 elements with water were conveyed fairly-well with increasing reaction intensity simulated as you move down the Group 1 element table. Each of the radioactive elements were configured with a generic Radiation number as in the real world the Radiation of the element is based on the half-life, and its value may vary from sample to sample[12, 13], a generic Radiation number is associated with radioactive elements to provide an experience which would allow the user to experience varying levels of radiation depending on the distance from the element sample. Overall, this project has successfully completed the goal of creating an interactive environment, where users can learn about the basic properties of the elements in a fun and interactive way at their own pace, as promised in the design brief.

VII. Bibliography

[1] U. Engine. Types of Blueprints [Online] Available: https://docs.unrealengine.com/5.0/en-US/types-of-blueprints-in-unreal-engine/

[2] U. Engine. Plugins [Online] Available: https://docs.unrealengine.com/5.0/en-US/plugins-in-unreal-engine/

[3] U. Engine. Fill Data Table from CSV File [Online] Available: https://docs.unrealengine.com/4.26/en-US/BlueprintAPI/EditorScripting/DataTable/FillDataTablefromCSVFile/

[4] M. Wadstein, "HTF do I? Import a CSV File into a Data Table (UE4)," ed, 2017.

[5] U. Engine. Creating Visual Effects [Online] Available: https://docs.unrealengine.com/5.0/en-US/creating-visual-effects-in-niagara-for-unreal-engine/

[6] U. Engine. Blueprint Interface [Online] Available: https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/Blueprints/UserGuide/Types/Interface/#:~:text=A%20Blueprint%20Interface%20is%20a,the%20Blueprints%20that%20added%20it.

[7] GDXR, "How To Use Cast Nodes And Why You Actually Want To Use Interfaces Instead - UE4/UE5 Blueprints," ed, 2021.

[8] Darkfall, "Unreal Engine 4: Part 4 - Linetrace / Interaction (Interface)," ed, 2020.

[9] U. Engine. Using a Single Line Trace (Raycast) by Channel [Online] Available: https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/Tracing/HowTo/SingleLineTraceByChannel/

[10] R. Channel, "Unreal Engine - AOE Damage Tutorial (1/3)," ed, 2020.

[11] R. Channel, "Unreal Engine - AOE Damage Tutorial (3/3)," ed, 2020.

[12] C. L. TC Kaspar , MW Dilbert, "Evaluation of Uranium235 Measurement
Techniques," 2017. [Online]. Available: https://www.pnnl.gov/main/publications/external/technical_reports/PNNL-26490.pdf

[13] C. f. D. C. a. Prevention. Measuring Radiation [Online] Available: https://www.cdc.gov/nceh/radiation/measuring.html

[14] T. P. Pilgrimage, "Unreal Engine 4 Interactive Sci Fi Screens! (Widget Blueprints Tutorial)," ed, 2017.

[15] U. Engine. Widget Blueprints [Online] Available: https://docs.unrealengine.com/5.0/en-US/widget-blueprints-in-umg-for-unreal-engine/