AR CHEMICAL COMPOUNDS

What is our GOAL for this MODULE?

We learned to show the chemical compound formation in Augmented reality based on barcode markers.

What did we ACHIEVE in the class TODAY?

- Learn to find the distance between two markers present in the Augmented reality of A-Frame.
- Learn to show the compound formation of two or more atomic molecules when markers are at a certain distance in the Augmented reality scene.

Which CONCEPTS/CODING BLOCKS did we cover today?

- 3X3 Barcode Markers.
- Three.js function distanceTo().
- ngrok to run the application.

How did we DO the activities?

1. Add **compounds** object in **compoundList.json** with:

    - **compound_name**: name of the compound that can be formed with elements with one electron to give.
    - **elements**: array of elements involved in the compound formation; keep the array empty for elements which will not be used as the base to update the compound formation.

```
"3": {
  "element_name": "K",
  "barcode_value": 3,
  "number_of_electron": 1,
  "compounds": [
    {
      "compound_name": "KF",
      "elements": ["K", "F"]
    },
    {
      "compound_name": "KCl",
      "elements": ["K", "Cl"]
    },
    {
      "compound_name": "KBr",
      "elements": ["K", "Br"]
    },
    {
      "compound_name": "KI",
      "elements": ["K", "I"]
    }
  ]
},
"4": {
  "element_name": "F",
  "barcode_value": 4,
  "number_of_electron": 7,
  "compounds": []
},
```

2. Add **compound nucleus** (spheres & name of all elements combined together) and the **base card** (in **Compounds.js** file) to render it after compound formation.

```
// adding compounds
var compounds = element.compounds;
compounds.map(item => {
  var compound = document.createElement("a-entity");
  compound.setAttribute("id", `${item.compound_name}-${barcodeValue}`);
  compound.setAttribute("visible", false);
  marker.appendChild(compound);

  var compoundCard = document.createElement("a-entity");
  compoundCard.setAttribute("id", `compound-card-${item.compound_name}`);
  compoundCard.setAttribute("geometry", {
    primitive: "plane",
    width: 1.2,
    height: 1.7
  });

  compoundCard.setAttribute("material", {
    src: `./assets/compound_cards/card_${item.compound_name}.png`
  });
  compoundCard.setAttribute("position", { x: 0, y: 0, z: 0.2 });
  compoundCard.setAttribute("rotation", { x: -90, y: 0, z: 0 });

  compound.appendChild(compoundCard);

  var posX = 0;
  item.elements.map((m, index) => {
    var n = document.createElement("a-entity");
    n.setAttribute("id", `compound-nucleus-${m}`);
    n.setAttribute("geometry", {
      primitive: "sphere",
      radius: 0.2
    });
    n.setAttribute("material", "color", colors[m]);
    n.setAttribute("position", { x: posX, y: 1, z: 0 });
    posX += 0.35;

    compound.appendChild(n);

    var nuclesName = document.createElement("a-entity");
    nuclesName.setAttribute("id", `compound-nucleus-name-${m}`);
    nuclesName.setAttribute("position", { x: 0, y: 0.21, z: 0 });
    nuclesName.setAttribute("rotation", { x: -90, y: 0, z: 0 });
    nuclesName.setAttribute("text", {
      font: "monoid",
      width: 3,
      color: "black",
      align: "center",
      value: m
    });

    n.appendChild(nuclesName);
  });
```

3. Set **markerhandler component** attribute for marker entity (in **Compund.js** file).

```
//add marker
var marker = document.createElement("a-marker");

marker.setAttribute("id", `marker-${barcodeValue}`);
marker.setAttribute("type", "barcode");
marker.setAttribute("element_name", elementName);
marker.setAttribute("value", barcodeValue);
marker.setAttribute("markerhandler", {});

scene.appendChild(marker);
```

4. Create a global array variable, called **elementsArray**, to keep track of all the elements which are visible in the scene's camera view.

```
var elementsArray = [];
```

5. Update **markerFound** event in the .init() function (in **markerhandler.js** file), if barcode marker is in the scene's camera view:
    - Get the element name and barcode value and push the data in **elementsArray**.
    - Keep the atomic structure visible and compound structure invisible.

```
this.el.addEventListener("markerFound", () => {
  var elementName = this.el.getAttribute("element_name");
  var barcodeValue = this.el.getAttribute("value");
  elementsArray.push({ element_name: elementName, barcode_value: barcodeValue });

  // Changing Compound Visiblity
  compounds[barcodeValue]["compounds"].map(item => {
    var compound = document.querySelector(`#${item.compound_name}-${barcodeValue}`);
    compound.setAttribute("visible", false);
  });

  // Changing atom Visiblity
  var atom = document.querySelector(`#${elementName}-${barcodeValue}`);
  atom.setAttribute("visible", true);
});
```

6. Update **markerLost** event in the .init() function (in **markerhandler.js** file), if the marker is not visible in the scene's camera view:

    - Find the index of the element whose marker is lost from the scene using **findIndex()** method of the array.

    - Remove the atomic structure of that element from the elementsArray using the **splice()** method.

```
this.el.addEventListener("markerLost", () => {
  var elementName = this.el.getAttribute("element_name");
  var index = elementsArray.findIndex(x => x.element_name === elementName);
  if (index > -1) {
    elementsArray.splice(index, 1);
  }
});
```

7. Take two arrays (**A** & **B**) for element names such that elements of A (having 1 free electron) can make compounds with elements of B (having 7 free electrons) only.

```
var A = ["H", "Li", "Na", "K"];
var B = ["F", "Cl", "Br", "I"];
```

8. Write a function **getCompund()** which will return the name of the compound and the barcode value of the marker on which we will show the compound:
    - Trace the array **A**, keep the **name** of the element in a variable called **compound**.
    - Trace the array **B**, **join the name** of the element to the same variable, **compound**.
    - Return the **final compound name** and barcode value of the first element.

```
getCompound: function () {
  for (var el of elementsArray) {
    if (A.includes(el.element_name)) {
      var compound = el.element_name;
      for (var i of elementsArray) {
        if (B.includes(i.element_name)) {
          compound += i.element_name;
          return { name: compound, value: el.barcode_value };
        }
      }
    }
  }
},
```

9. Write a function to calculate the distance between the two Three.js position vectors using distanceTo():
    - Take **elA** and **elB** as the parameters to pass the A-Frame marker entity element's position.
    - Access elA and elB as Three.js object to find distance using **distanceTo()**.

```
getDistance: function (elA, elB) {
  return elA.object3D.position.distanceTo(elB.object3D.position);
},
```

10. Write a function to show the compound:
    - Set the visible attributes of the **elements as false**.
    - Set the visible attributes of the **compounds as true**.

```
showCompound: function (compound) {
  elementsArray.map(item => {
    var el = document.querySelector(`#${item.element_name}-${item.barcode_value}`);
    el.setAttribute("visible", false);
  });

  // show Compound
  var compound = document.querySelector(`#${compound.name}-${compound.value}`);
  compound.setAttribute("visible", true);
},
```

11. Add the text entity to show when the compounds cannot be formed.

```
<a-plane
  id="message-text"
  position="0 0 -3.2"
  rotation="0 0 0"
  width="1.6"
  height="0.15"
  color="#FFFFFF"
  visible="false"
>
  <a-entity
    text="font:monoid;value:Compound will not form...;align:center;width:2.1;color:#c2185b;"
  ></a-entity>
</a-plane>
```

12. Write in the .**tick()** function:
    - Select the marker, **marker1** and **marker2** element to find the distance.
    - Call **getDistance()** by passing maker1 and marker2.
    - Call **showCompound()** if distance is less than 1.25.
    - Else make the text message visible.

```
tick: function () {
  if (elementsArray.length > 1) {

    var messageText = document.querySelector("#message-text");

    var length = elementsArray.length;
    var distance = null;

    var compound = this.getCompound();

    if (length === 2) {
      var marker1 = document.querySelector(`#marker-${elementsArray[0].barcode_value}`);
      var marker2 = document.querySelector(`#marker-${elementsArray[1].barcode_value}`);

      distance = this.getDistance(marker1, marker2);

      if (distance < 1.25) {
        if (compound !== undefined) {
          this.showCompound(compound);
        } else {
          messageText.setAttribute("visible", true);
        }
      } else {
        messageText.setAttribute("visible", false);
      }
    }
  }
},
```

13. **Compound formations** between **three elements**:
    - Take a separate array **C** of elements that needs two elements from **A** to form compounds.
    - Write the function **countOccurrences()**.
    - Update the **getCompound()** to trace elements from array **C**.

```
var A = ["H", "Li", "Na", "K"];
var B = ["F", "Cl", "Br", "I"];

var C = ["O", "S", "Se"];

var elementsArray = [];
```

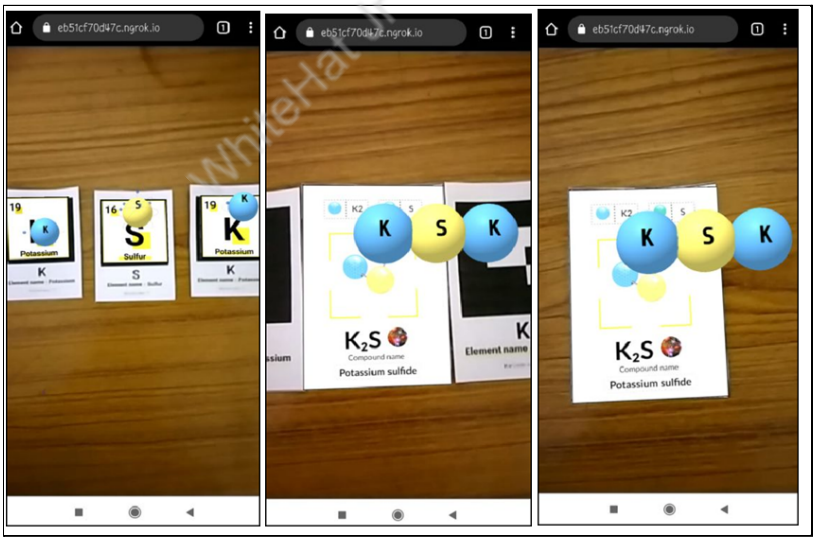
```
countOccurrences: function (arr, val) {
  return arr.reduce((a, v) => (v.element_name === val ? a + 1 : a), 0);
},
```

```
getCompound: function () {
  for (var el of elementsArray) {
    if (A.includes(el.element_name)) {
      var compound = el.element_name;
      for (var i of elementsArray) {
        if (B.includes(i.element_name)) {
          compound += i.element_name;
          return { name: compound, value: el.barcode_value };
        }

        if (C.includes(i.element_name)) {
          var count = this.countOccurrences(elementsArray, el.element_name);
          if (count > 1) {
            compound += count + i.element_name;
            return { name: compound, value: i.barcode_value };
          }
        }
      }
    }
  }
},
```

14. Update the **.tick()** function accordingly for 3 elements.

```
if (length === 3) {
  var marker1 = document.querySelector(`#marker-${elementsArray[0].barcode_value}`);

  var marker2 = document.querySelector(`#marker-${elementsArray[1].barcode_value}`);

  var marker3 = document.querySelector(`#marker-${elementsArray[2].barcode_value}`);

  var distance1 = this.getDistance(marker1, marker2);
  var distance2 = this.getDistance(marker1, marker3);

  if (distance1 < 1.25 && distance2 < 1.25) {

    if (compound !== undefined) {
      var barcodeValue = elementsArray[0].barcode_value;
      this.showCompound(compound, barcodeValue);
    } else {
      messageText.setAttribute("visible", true);
    }
  }
  else {
    messageText.setAttribute("visible", false);
  }
}
```

15. Test the code using ngrok:



We learned to show the chemical compound formation in Augmented reality based on barcode markers.

What's NEXT?

In the next class, we will learn about location based augmented reality.

EXTEND YOUR KNOWLEDGE:

- You can refer to the link below to explore more about A-Frame:
  A-Frame