


Topic	A-Frame COMPONENTS	
Class Description	Students learn how to write their own custom components in A-Frame.	
Class	C151	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> <li>Learn about the Entity Component System in A-Frame.</li> <li>Learn how to register custom components in A-Frame.</li> </ul>	
Resources Required	<ul style="list-style-type: none"> <li>Teacher Resources:               <ul style="list-style-type: none"> <li>Visual Studio Code Editor</li> <li>laptop with internet connectivity</li> <li>earphones with mic</li> <li>notebook and pen</li> </ul> </li> <li>Student Resources:               <ul style="list-style-type: none"> <li>Visual Studio Code Editor</li> <li>laptop with internet connectivity</li> <li>earphones with mic</li> <li>notebook and pen</li> </ul> </li> </ul>	
Class structure	Warm-Up Teacher-led Activity Student-led Activity Wrap-Up	05 mins 20 mins 15 mins 05 mins
WARM-UP SESSION - 05 mins		
<b><u>CONTEXT</u></b> <ul style="list-style-type: none"> <li>Introduce A-Frame components.</li> </ul>		



**Teacher starts slideshow** from slides 1 to 16  
 Refer to speaker notes and follow the instructions on each slide.

Activity details	Solution/Guidelines
<p><i>Hey &lt;student's name&gt;. How are you? It's great to see you!          Are you excited to learn something new today?</i></p> <p><b>Run the presentation from slide 1 to slide 3</b></p> <p><b>Following are the WARM-UP session deliverables:</b></p> <ul style="list-style-type: none"> <li>• Greet the student.</li> <li>• Revision of previous class activities.</li> <li>• Quizzes</li> </ul>	<p><b>ESR:</b> Hi, thanks, Yes I am excited about it!</p> <p>Click on the slide show tab and present the slides</p>
<b>Q&amp;A Session</b>	
Question	Answer
<p>How many curved arrows are visible once a shape is selected in Tinkercad?</p> <p>A. 1          B. 2          C. 3          D. 4</p>	<b>C</b>
<p>Which of the shapes can we use to create parabola?</p> <p>A. paraboloid          B. pyramid          C. polygon          D. ring</p>	<b>A</b>
<b>Continue the WARM-UP session</b>	
Activity details	Solution/Guidelines

<p><b>Run the presentation from slide 4 to slide 16 to set the problem statement.</b></p> <p><b>Following are the WARM-UP session deliverables:</b></p> <ul style="list-style-type: none"> <li>• Appreciate the student.</li> <li>• Introduce A-Frame.</li> </ul>		<p>Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.</p>
<p style="text-align: center;"><b>Teacher ends slideshow</b></p> 		
Class Steps	Teacher Action	Student Action
<p><b>Step 1:</b> <b>Warm-Up</b> <b>(5 mins)</b></p>	<p>Hi &lt;Student Name&gt;! How are you?</p> <p>That's great!</p> <p>Do you remember what we learned in the previous class? Let's recollect what we learned in the previous class.</p> <p>Great! Now you have an idea of how 3D models are created and used for virtual scenes. We will be learning more about how to create a virtual simulation.</p> <p>Virtual simulation is basically re-creating real-world environments on computer screens.</p>	<p><b>ESR:</b> I am good.</p> <p><b>ESR:</b> Yes, we created a 3D model using Tinkercad and exported it to A-Frame.</p>

	<p>For example, you can create a simulation for games or dance clubs, shopping mall simulations and so on.</p> <p>The beauty of creating virtual simulation is that you can virtually experience things as if they were in the real world without actually visiting there.</p> <p>Isn't that cool?</p> <p>We will soon be learning how to create a virtual flight simulation.</p> <p>But before we can do that, we need to learn a few more concepts in A-Frame.</p> <p>Are you excited?</p> <p>Great!</p>	<p><b>ESR:</b> Yes.</p> <p><b>ESR:</b> Yes.</p>
	Now, Let's get started.	
<b>TEACHER-LED ACTIVITY - 20 mins</b>		
<b>Teacher Initiates Screen Share</b>		
<p style="text-align: center;"><b><u>CHALLENGE</u></b></p> <ul style="list-style-type: none"> <li>• <b>Understand the entity-component-system in A-Frame.</b></li> <li>• <b>Understand types of components.</b></li> <li>• <b>Learn how to register custom components in A-Frame.</b></li> </ul>		

<p><b>Step 2:</b> <b>Teacher-led Activity</b> <b>(10 mins)</b></p>	<p>Do you remember what pattern/architecture A-Frame follows?</p> <p>A-Frame follows the <u>entity-component-system(ECS)</u> pattern. What do you think it is?</p> <p>Today we will be learning about the ECS in more detail.</p> <p>First, let's understand what an entity-component-system is.</p> <p>In simple terms, <b>Entity</b> can be considered an empty container in which multiple components can be attached. Entities can be referred to using <b>id</b>.</p> <p><b>Components</b> can be understood as the main building blocks in ECS, which contains <b>data</b> and where all logic is implemented by mixing, matching and configuring components. Once the component is defined, it can be <b>re-used</b> with different entities.</p> <p><b>System</b> globally manages the components and entities.</p> <p>Let's try to understand it with the help of an example.</p>	<p><b>ESR:</b> Varied.</p> <p><b>ESR:</b> Varied.</p>
--	--	---

	<p>Before that, can you tell me where else you have heard about components?</p> <p>Yes, great! And why do we create components?</p> <p>Amazing!</p> <p><u><a href="#">[Teacher Activity 1 Illustration]</a></u></p> <p><i>The teacher shows the illustration.</i></p> <p>Let's consider an entity as an empty square box container that does not do anything on its own and does not have any features or appearances.</p> <p>And components can be considered a piece of a puzzle that can be combined with an entity to define that entity's behavior and functionality.</p>	<p><b>ESR:</b> We have used components in React Native while making a mobile application.</p> <p><b>ESR:</b> That makes our code more readable and structured. And we can re-use those components wherever required.</p> <p><i>The student observes and asks questions.</i></p>
<p><u><a href="#">[Teacher Activity 1 Illustration]</a></u></p>		



**Entity**



**A component**



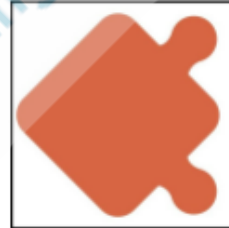
**Entity**

+



**A component**

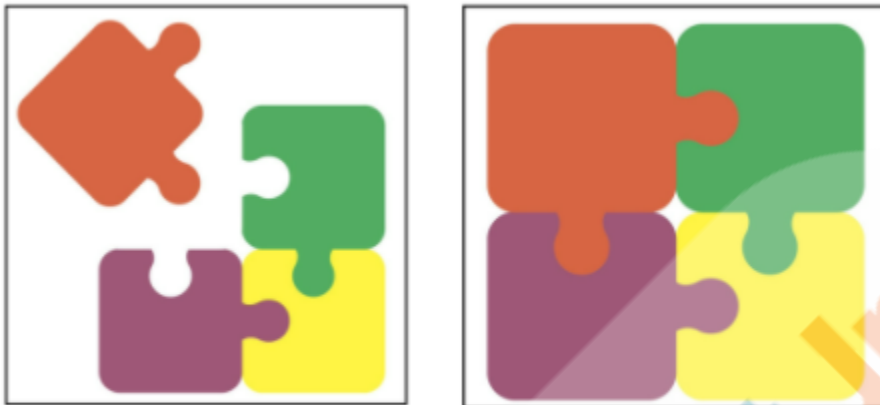
=



Also, multiple components can be attached to an entity like multiple pieces of puzzles are combined together.

**All components together can define how the entity will look and how the entity will behave.**

[Teacher Activity 1 Illustration]



**Multiple components with different behaviour and functionality can be attached to one entity.**

Now, let's move on to the ECS pattern in A-Frame.  
Do you remember how the entity is represented?

Great!

And we have been using components inside `<a-entity>` as an HTML attribute, right?

Can you give me some examples of attributes that we have been using till now?

Yes, and the value in the position attribute represents component data.  
We have a position component where the logic is written to control the position of any entity.

**ESR:** Yes, Using `<a-entity>` tag.

**ESR:** Yes!

**ESR:** We have been using position, rotation, animation, and light.



	<p>And this component can be used with any number of entities, which saves us from re-writing the same logic repeatedly.</p> <p>As a programmer, you must always remember the DRY (Don't Repeat Yourself) principle.</p> <p>We will be learning how to write our own components soon, but before that, let's understand the two types of components and data associated with them that we have used till now.</p> <p><b>Single-property component:</b> A single property component consists of the <u>data with single values</u>.</p> <p>For example, the position is the name of the component and the "-10 3 5" is the data of the component.</p> <p><b>Multi-property component:</b> A multi-property component consists of <u>data with multiple properties and their values</u></p> <p>For example, in the light component, data of the component is type, intensity, color, etc.</p>	
--	---	--

	<p>And these data are called properties of the components, and their values are set as -</p> <p>"type: spot, intensity:1 , color: yellow"</p>	
<p>Single-property Component</p> <pre>&lt;a-entity position="-10 3 5" &gt;&lt;/a-entity&gt;</pre> <p>Multi-property component</p> <pre>&lt;a-entity light="type: directional; intensity: 1; color:yellow"&gt; &lt;/a-entity&gt;</pre>		
	<p>Now, since we have learned how entities and components are represented in A-Frame, can you tell me how a <b>system</b> is represented in A-Frame?</p> <p><u>&lt;a-scene&gt; represents the system in A-Frame which globally manages all entities and its components.</u></p> <p>Okay, now let's move on to how we can write our own components i.e., custom components.</p> <p>We can begin with writing a very basic <b>log component</b> which will display a message in the console log once it is attached to the entity.</p>	<p>ESR: Varied.</p>

	<p>For this, we will need a JavaScript file.</p> <p><i>The teacher opens Visual Studio Code Editor, adds the JavaScript file 'Log.js', and includes that in index.html.</i></p> <p>To use the registered component, it must be used before &lt;a-scene&gt; in the index.html file.</p>	
	<pre> &lt;head&gt;   &lt;script src="https://aframe.io/releases/1.0.4/aframe.min.js"&gt;&lt;/script&gt;    &lt;script src="Log.js"&gt;&lt;/script&gt; &lt;/head&gt;  &lt;body&gt;    &lt;a-scene background="color:#000000"&gt;    &lt;/a-scene&gt; </pre>	
	<p>To write custom components, we use:</p> <p><b>AFRAME.registerComponent</b> <b>(name, definition)</b></p> <p><b>name:</b> is the component name; it is of string data type. Here 'box' is the name of the component.</p> <p><b>definition:</b> contains the component definition. It is a JavaScript object</p>	

	<p>which has <b>schema</b> and <b>lifecycle handler methods</b>(init, update, tick, remove, etc).</p> <p>Let's take a quick overview of the basic structure of the component and terminology used here.</p> <p><b>schema:</b> is an object that defines the property names, their types, and default values. <u>The schema defines the shape of the <b>data</b>.</u></p> <p><b>Lifecycle Handler Methods:</b></p> <p><b>init:</b> This is used to set up the initial state. It is called once when the component is initialized.</p> <p><b>update:</b> This is used to modify the entity.</p> <p><b>remove:</b> This is used to undo all previous modifications to the entity.</p> <p><b>tick:</b> This is used for checking continuous changes. It is called on every render loop of the scene.</p>	
--	--	--

```

AFRAME.registerComponent('name', {
  schema: {
    //data
  },

  init: function () {
    // Do something when component first attached.
  },

  update: function () {
    // Do something when component's data is updated.
  },

  remove: function () {
    // Do something the component or its entity is detached.
  },

  tick: function () {
    // Do something on every scene tick or frame.
  }
});

```

Now that we are familiar with the terminology let's start by using schema to set the properties needed to create a log component.

What do you think are properties needed for the log?

Yes, great!

For our log component, let's define a message data property type via the schema. The 'message' property type will have a 'string' type and have a 'default' value of Hello, World!

**ESR:** Log message to be displayed in the console.

```
// Registering component in log-component.js
AFRAME.registerComponent('log', {
  schema: {
    message: {type: 'string', default: 'Hello, World!'}
  }
});
```

Now, this component will log a simple message once when the component's entity is attached using the `.init()` handler.

The component's property values defined in the schema can be accessed through **this.data**.

**this** points to the entity at which the component is attached.

So, let's log `this.data.message`!

*Defining .ini() handler method.*

```
// Registering component in log-component.js
AFRAME.registerComponent('log', {
  schema: {
    message: {type: 'string', default: 'Hello, World!'}
  },
  init: function () {
    console.log(this.data.message);
  }
});
```

*Attaching the component to the entity.*

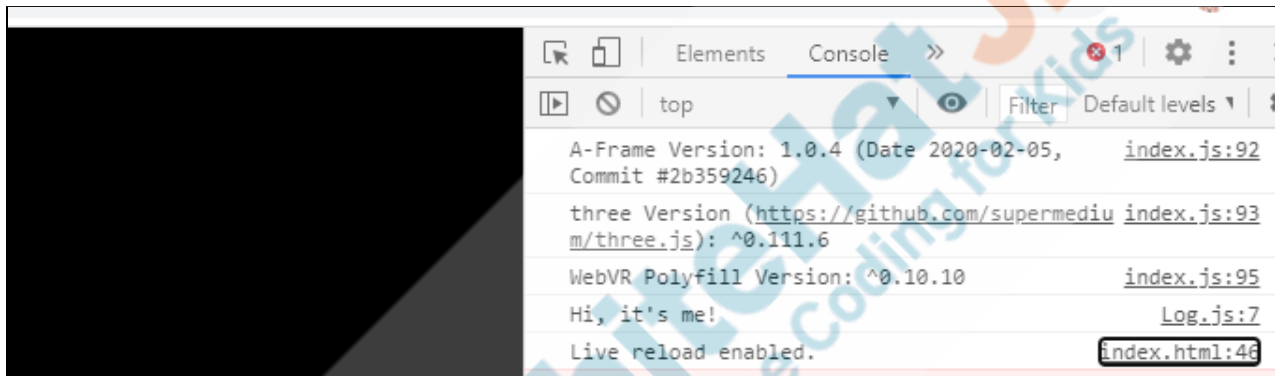
```
<body>

  <a-scene background="color:#000000">

    <a-entity log="message: Hi, it's me!"></a-entity>

  </a-scene>

</body>
```





Now that we have learned how to create custom components and use them in the A-Frame scene, it's your turn.

We have been using the animation component, which handles movements and rotation.

Now you will create a simple component that can handle a box's movement from one position to another in the x-direction.

Please share your screen with me.

**Teacher Stops Screen Share**

<div>  </div> <p><b>Teacher starts slideshow</b> from slides 17 to 18</p> <p>Refer to speaker notes and follow the instructions on each slide.</p>		
	Now it's your turn. Please share your screen with me.	
<div>  </div> <p><b>Teacher ends slideshow</b></p>		
<b>STUDENT-LED ACTIVITY - 15 mins</b>		
<ul style="list-style-type: none"> <li>Ask the student to press the ESC key to come back to the panel.</li> <li>Guide the student to start a screen share.</li> <li>The teacher gets into fullscreen.</li> </ul>		
<p><u><b>ACTIVITY</b></u></p> <ul style="list-style-type: none"> <li>Learn how to write custom components to move a box.</li> <li>Learn how to attach components to the entity.</li> </ul>		
<p><b>Step 3:</b> <b>Student-Led Activity</b> <b>(20 mins)</b></p>	<p><i>Guide the student to download the code from <a href="#">Student Activity 1</a>.</i></p> <p><i>C1: Guide the student to open the Visual Studio Code Editor and create the Box.js file.</i></p> <p><i>Include the file in index.html before &lt;a-scene&gt;.</i></p>	<p><i>The student writes the code to create a JavaScript file and include that in index.html.</i></p>
<pre>&lt;head&gt;   &lt;script src="https://aframe.io/releases/1.0.4/aframe.min.js"&gt;&lt;/script&gt;    &lt;script src="Box.js"&gt;&lt;/script&gt;  &lt;/head&gt;</pre>		



	<p>What is next?</p> <p>And how can we do that?</p> <p>Yes. Great!</p> <p><i>C2: Guide the student to register a component and define a variable in the schema which will be used to update the position in the x-direction.</i></p>	<p><b>ESR:</b> Registering component.</p> <p><b>ESR:</b> By using <b>AFRAME.registerComponent()</b>.</p>
<pre>// Registering component in box-component.js AFRAME.registerComponent("move-box", {   schema: {     moveX: { type: "number", default: 1 },   }, });</pre>		
	<p>Now to update the position continuously we can use the tick handler method which can update the values continuously.</p> <p>Let's update the data value of this variable. How would you do that?</p> <p>Amazing!</p> <p><i>C3: Guide the student to update the variable's value by some amount in the tick handle method.</i></p>	<p><b>ESR:</b> By using <b>this.data</b>.</p>

```
// Registering component in box-component.js
AFRAME.registerComponent("move-box", {
  schema: {
    moveX: { type: "number", default: 1 },
  },

  tick: function () {

    this.data.moveX = this.data.moveX + 0.01;

  }
});
```

Now, what's next?

ESR: Varied.

We should **update the value of the position attribute of <an-entity>** where <a-box> is present.

To do this, we use:

**this.el.getAttribute()**: to get the current values of the position attribute.

**this.el.setAttribute()**: to set the updated value of the position attribute.

**this.el** gives reference to the entity as an HTML element.

*C4: Guide the student to update the value of the position attribute of the entity.*

	<p>Take a “<b>pos</b>” variable to get the attribute value.</p> <p>Now update the x-axis value of position attribute- “<b>pos.x</b>”.</p>	
<pre>// Registering component in box-component.js AFRAME.registerComponent("move-box", {   schema: {     moveX: { type: "number", default: 1 },   },   tick: function () {      this.data.moveX = this.data.moveX + 0.01;      var pos = this.el.getAttribute("position");      pos.x = this.data.moveX;      this.el.setAttribute("position", {x: pos.x, y: pos.y, z: pos.z});   } });</pre>		
	<p>Now, we should attach this component to the entity to move the box.</p> <p><i>C5: Guide the student to attach the “move-box” component.</i></p>	

```
<a-entity position="0 0 0" move-box>
  <a-box>
    </a-box>
  </a-entity>
```

Amazing! You can see the box moving.

You have done a great job of writing your own component!

### Teacher Guides Student to Stop Screen Share

#### FEEDBACK

- Compliment the student for her/his effort in the class.
- Encourage the student to think and come up with their own solutions.



Teacher starts slideshow from slide 19 to slide 30

#### Activity details

#### Solution/Guidelines

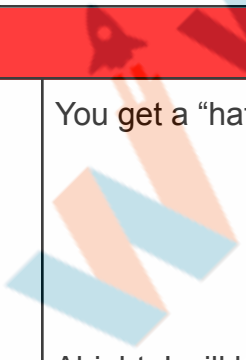



Run the presentation from slide 19 to slide 30

Following are the wrap-up session deliverables:

- Explain the facts and trivias
- Next class challenge
- Project for the day
- Additional Activity

Guide the student to develop the project and share with us.

### Quiz Time - Click on In-Class Quiz

Question		Answer
What pattern/architecture A-Frame follows?  A. entity-component-system B. entire-component-system C. entity-command-system D. entire-command-system		<b>A</b>
_____ is recreating real-world environments on computer screens.  A. Visual simulation B. Virtual Simulation C. Vital simulation D. Verified Simulation		<b>B</b>
Which of the following can be considered as a piece of a puzzle that can be combined with an entity to define that entity's behavior and functionality?  A. entity B. box C. component D. All of the above		<b>A</b>
<div>        • End the quiz panel     </div>		
	You get a "hats-off".  Alright. I will look forward to seeing you create your own component.	<p><i>Make sure you have given at least 2 Hats Off during the class for:</i></p> <div> <div>Creatively Solved Activities  +10</div> <div>Great Question  +10</div> <div>Strong Concentration  +10</div> </div>

<p><b>Project Overview</b></p>	<p><b>CUSTOM CAR DESIGN</b></p> <p><b>Goal of the Project:</b></p> <p>In this project, you will be creating custom components.</p> <p><b>Story:</b></p> <p>Components in A-Frame are a way to manage, structure, and modify entities.</p> <p>Jess wants to showcase car models his company will be building to a client in a meeting. He needs to show multiple cars with a variety of sizes and colors.</p> <p>To manage a project like this, Jess is going to create the components for the cars so that it will be easy to manage the code. Help Jess in building the entity component system of the car models.</p> <p>Write a component for the car model design specifications.</p> <p>I am very excited to see how you will write your own components.</p> <p>Bye!</p>	
<div> <div>Teacher Clicks</div> <div>✕ End Class</div> </div>		

<b>Additional Activities</b>	<i>Guide the student to try updating the rotation attribute of the entity.</i>	
------------------------------	--	--

```
// Registering component in box-component.js
AFRAME.registerComponent("rotate-box", {
  schema: {
    moveX: { type: "number", default: 1 },
  },
  tick: function () {

    this.data.moveX = this.data.moveX + 5;

    var rotate = this.el.getAttribute("rotation");

    rotate.x = this.data.moveX;
    rotate.y= this.data.moveX;

    this.el.setAttribute("rotation", {x: rotate.x, y: rotate.y, z: rotate.z});
  }
});
```

Activity	Activity Name	Links
Teacher Activity 1	Entity Component System Illustration	<a href="https://s3-whjr-v2-prod-bucket.whjr.online/8db2ee9b-721f-4d43-a74b-3255e8fda634.pdf">https://s3-whjr-v2-prod-bucket.whjr.online/8db2ee9b-721f-4d43-a74b-3255e8fda634.pdf</a>
Teacher Activity 2	Teacher Reference Code	<a href="https://github.com/whitehatjr/PRO-C151-Teacher-Ref">https://github.com/whitehatjr/PRO-C151-Teacher-Ref</a>
Student Activity 1	Blank Activity	<a href="https://github.com/whitehatjr/PRO-C151-Student-Activity">https://github.com/whitehatjr/PRO-C151-Student-Activity</a>
Project Solution Link	Custom Car Design	<a href="https://github.com/whitehatjr/PRO-VR-C151">https://github.com/whitehatjr/PRO-VR-C151</a>
Teacher Ref. Visual Aid Link	Visual Aid link	<a href="https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/PRO_C151_withcues.html">https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/PRO_C151_withcues.html</a>

Teacher Ref. In-class Quiz	In-Class quiz	<a href="https://s3-whjr-curriculum-uploads.whjr.online/58702593-4bc1-46c5-a85c-6119ffc317c8.pdf">https://s3-whjr-curriculum-uploads.whjr.online/58702593-4bc1-46c5-a85c-6119ffc317c8.pdf</a>
----------------------------	---------------	---

