




Topic	3D TEXT AND TIMING EVENTS	
Class Description	Students learn to use text in A-Frame. Students will learn about JavaScript timing events to show the timer clock in the virtual simulation scene.	
Class	C156	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> • Learn how to use 3D text in A-Frame. • Learn to use JavaScript timing events. 	
Resources Required	<ul style="list-style-type: none"> • Teacher Resources <ul style="list-style-type: none"> ○ Visual Studio Code Editor ○ laptop with internet connectivity ○ earphones with mic ○ notebook and pen • Student Resources <ul style="list-style-type: none"> ○ Visual Studio Code Editor ○ laptop with internet connectivity ○ earphones with mic ○ notebook and pen 	
Class structure	Warm-Up Teacher-led Activity Student-led Activity Wrap-Up	05 mins 15 mins 20 mins 05 mins
WARM-UP SESSION - 05 mins		
Teacher starts slideshow  from slides 1 to 12		

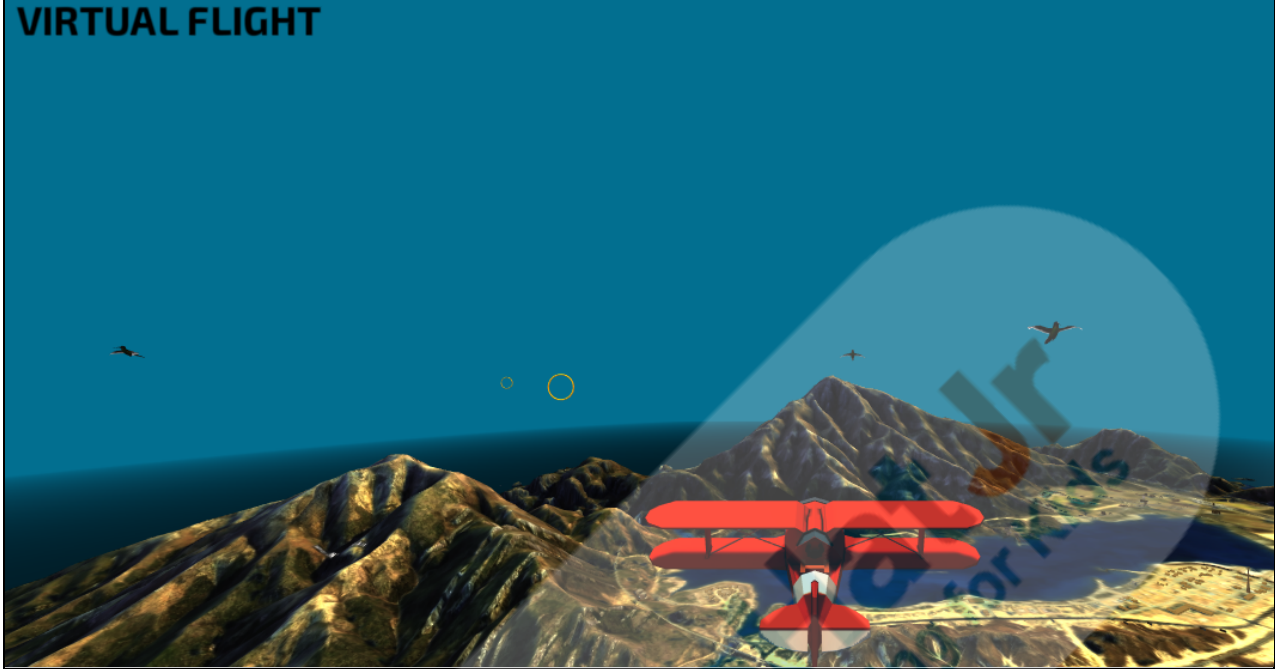
Refer to speaker notes and follow the instructions on each slide.	
Activity details	Solution/Guidelines
<p>Hi, how have you been? Are you excited to learn something new?</p> <p>Run the presentation from slide 1 to slide 4.</p> <p>The following are the warm-up session deliverables:</p> <ul style="list-style-type: none"> Reconnect with previous class topics. Warm-Up quiz session. 	<p>ESR: Varied Response.</p> <p>Click on the slide show tab and present the slides.</p>
Q&A Session	
Question	Answer
<p>In A-Frame, friction and restitution properties can be configured using:</p> <p>A. matter component B. engine component C. physics component D. world Component</p>	<p>C</p>
<p>How can we set the static-body component of the rings and birds?</p> <p>A. using setAttribute() method B. using setComponent() method C. both A & B D. using resetAttribute() method</p>	<p>A</p>
Continue the WARM-UP session	
Activity details	Solution/Guidelines

<p><i>Run the presentation from slide 5 to slide 12 to set the problem statement.</i></p> <p>The following are the warm-up session deliverables:</p> <ul style="list-style-type: none"> • Review code from the last class. • Text in A-Frame VR. • JavaScript timing events. 	
<p>Teacher ends slideshow </p>	
<p>Teacher Initiates Screen Share</p>	
<p>TEACHER-LED ACTIVITY - 15 mins</p>	
<p><u>CHALLENGE</u></p> <ul style="list-style-type: none"> • Use A-Frame text. • Use Javascript timing events. 	
<p>Teacher starts slideshow  for slide 13 to 18</p>	

Step 2: Teacher-led Activity (15 min)	<p><i><The teacher opens the code from the previous class.></i></p> <p>To begin with, we will use the A-Frame text component in the scene.</p> <p>The text component has many properties, and we will be learning to use a few of them today-</p> <ul style="list-style-type: none"> • align: this property is for the alignment of text; • width: width of the text; • height: height of the text; • value: actual content of the text. To add a new line in the text content we can use \n; • font: font of the text; and • color: color of the text. <p>Let's add an <code><a-entity></code> to make the header for all the text entities required for the timer and score entity.</p> <p><i><The teacher codes to add <a-entity> tag sets id and position.></i></p>	
<pre> <!--Game Timer and Score--> <a-entity id="header" position="1 -4 0"> </a-entity> </pre>		

	<p>Now let's add the title of the simulation using A-Frame text in the scene.</p> <p><i>< The teacher codes to add one more <a-entity> for the title as the child of the header entity and sets its position and text component.></i></p> <p><i><The teacher sets the font, align, width, color and value properties of the text component and shows the output.></i></p>	
<pre> <!--Game Timer and Score--> <a-entity id="header" position="1 -4 0"> <!-- Title --> <a-entity position="-28.5 21.5 0" text="font: exo2bold; align: center; width: 40; color: black; value: VIRTUAL FLIGHT" ></a-entity> </a-entity> </pre>		

VIRTUAL FLIGHT



Now we have got the text at the top left corner, let's set the text below to represent the time remaining to collect all the target rings.

<The teacher codes to add one more <a-entity> for the timer as the child of the header entity and sets its position and text component.>

<The teacher sets the font, align, width, color and value properties of the text component and shows the output.>

```
<!--Game Timer and Score-->
<a-entity id="header" position="1 -4 0">
  <!-- Title -->
  <a-entity
    position="-28.5 21.5 0"
    text="font: exo2bold; align: center; width: 40; color: black; value: VIRTUAL FLIGHT"
  ></a-entity>

  <!-- Timer -->
  <a-entity position="-23.5 19.3 0">
    <a-entity
      position="-1.5 0.07 0"
      text="font: aileronsemibold; width: 12; color: black; value: TIME\nREMAINING"
    ></a-entity>
    <a-entity
      id="timer"
      position="-9.8 0.3 0"
      text="font: exo2bold; align: center; width: 30; color: black; value: 00:00"
    ></a-entity>
  </a-entity>
</a-entity>
```



Now what should be done next?

Great!

ESR: We should start the timer clock when the scene loads.

	<p>How do you think we will update the time?</p> <p>We will be taking this text value from the scene and will update the value while the program is running.</p> <p>Let's see how we can achieve this.</p> <p>Suppose we want to set the timer clock to 2 mins.</p> <p>For this, let's take the variable "duration" and keep the value as 120, representing the number of seconds in 2 mins.</p> <p>Since we want to update the timer entity element in the scene, we can select that using the querySelector() method.</p> <p>Let's write our own function "startTimer" to update the "timer" entity's text value.</p> <p><i><Teacher codes to add the startTimer function with the duration and the timer element as parameters.></i></p>	<p>ESR: Varied.</p>
--	--	----------------------------


```
init: function () {
  var duration = 120;
  const timerEl = document.querySelector("#timer");
  this.startTimer(duration, timerEl);
},

startTimer: function (duration, timerEl) {
```

Now the timer should keep on decreasing second by second, right?

For this, we will be using, JavaScript **timings events** method- **setInterval()**.

The setInterval() method calls a function at specified intervals (in milliseconds). It is written as ***setInterval(function, millisecond)***.

For example, if we want the function to keep on calling every 1 sec, we will set milliseconds' value as 1000.

Then after 1 sec function will be again, and after 1 sec again and so on.

Note: 1 sec = 1000 milliseconds.

ESR: Yes.

```
startTimer: function (duration, timerEl) {  
  
    setInterval( ()=> { } ,1000)  
  
},
```

Now, since we want to show the timer clock on the screen, let's update the function part, `()=>{ //code }`, in the `setInterval()` method.

For this we will keep on decreasing the time duration, till the value of the duration becomes 0.

The teachers adds the condition:

```
if(duration>0){  
....  
duration-=1;  
}
```

Then, we will take two variables to represent the minutes and seconds of the time.

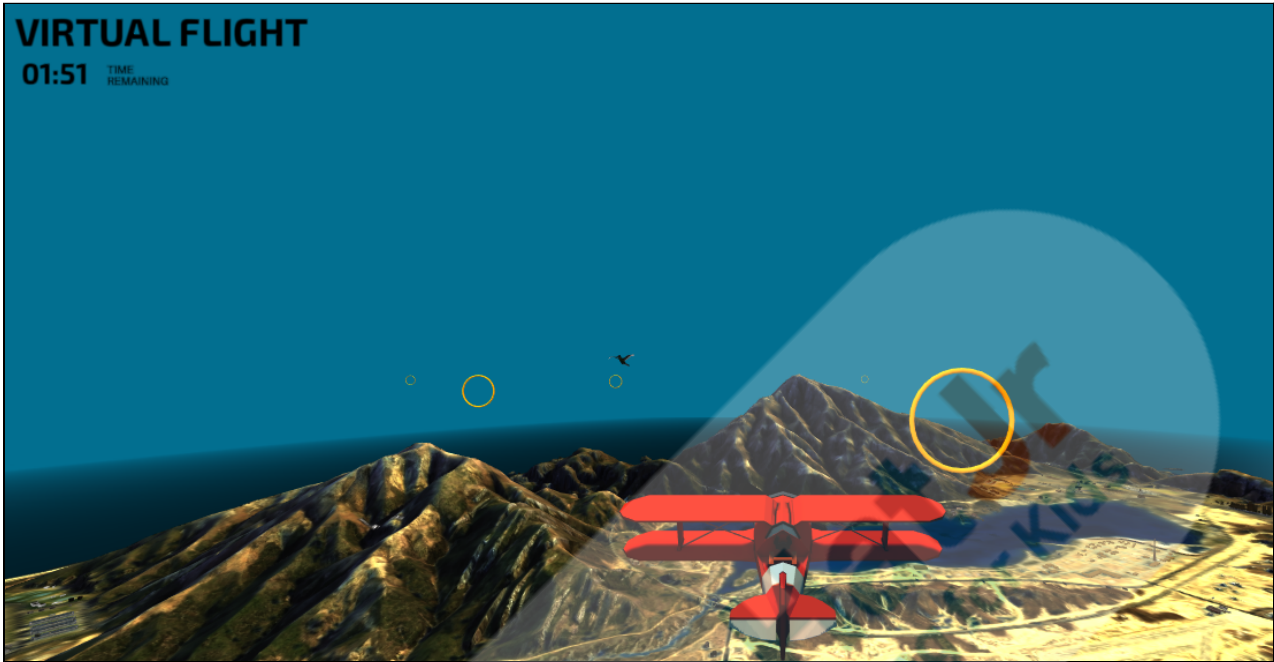
Now, since the duration is 120 second:

- **To get the minutes:** we can divide the number by 60.
- **To get the seconds:** we can take the modulus of the number by 60.

	<p>parseInt() is the JavaScript function which can be used to convert any string to integer numbers.</p> <p>With the help of this method we can keep the value as whole integer numbers without decimal.</p> <p>Also, once the number is less than 10, it will be a single digit, right?</p> <p>For example: 9 seconds is less than 10. Now, "9" can be written as "09"</p> <p>What can be done to show "0" at the beginning of the number?</p> <p>Amazing!</p> <p>Let's write the condition to join "0" at the beginning of the minutes and seconds.</p> <p>Do you remember what we can use to join any two strings?</p> <p>Yes, we can use + operator to join any two strings in JavaScript.</p> <p>Now, let's update the "value" attribute of timer text in the A-Frame scene with the setAttribute() method.</p>	<p>ESR: Yes!</p> <p>ESR: Use string concatenation to join two strings. We can join "0" and "9".</p> <p>ESR: The + operator.</p>
--	---	--

<The teacher codes and shows the output.>

```
startTimer: function (duration, timerEl) {  
  var minutes;  
  var seconds;  
  
  setInterval(()=> {  
    if (duration >= 0) {  
      minutes = parseInt(duration / 60);  
      seconds = parseInt(duration % 60);  
  
      if (minutes < 10) {  
        minutes = "0" + minutes;  
      }  
      if (seconds < 10) {  
        seconds = "0" + seconds;  
      }  
  
      timerEl.setAttribute("text", {  
        value: minutes + ":" + seconds,  
      });  
  
      duration -= 1;  
    }  
  }, 1000)  
},
```

		
	<p>Now that we have learned how to add text in 3D scenes, you will be adding the text to show the remaining targets and the score. Also, update the values of the score and the number of targets left with the help of functions.</p> <p>You will also be adding a function for game over.</p> <p>Are you excited?</p>	<p>ESR: Yes!</p>
<p>Teacher Stops Screen Share</p>		
	<p>Now it's your turn. Please share your screen with me.</p>	
<p>STUDENT-LED ACTIVITY - 20 mins</p>		
<ul style="list-style-type: none"> • Ask the student to press the ESC key to come back to the panel. 		

- Guide the student to start screen share.
- Teacher gets into fullscreen.

ACTIVITY

- Set the attributes of the gLTF models to static and dynamic bodies.
- Write a component to detect a collision between entities.



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

Step 3: Student-Led Activity (20 mins)

The teacher guides the student to open the code from the previous class.

[Student Activity 1]

Now, we will show the number of targets remaining based on the number of rings that collide with the flight.

We will also be updating the score every time the flight collides with the ring.

Guide the student to add the text for the number of targets and score as a child entity of the header entity.

```
<!-- Target Status -->
<a-entity position="-18 19.3 0">
  <a-entity
    position="-0.6 0.13 0"
    text="font: aileronsemibold; width: 12; color: black; value: TRAGETS\nREMAINING"
  ></a-entity>
  <a-entity
    id="targets"
    position="-8 0.3 0"
    text="font: exo2bold; align: center; width: 30; color: black; value: 20"
  ></a-entity>
</a-entity>
```

```
<!-- Score -->
<a-entity position="30.5 21.5 0">
  <a-entity
    position=""
    text="font: exo2bold; align: center; width: 40; color: black; value: SCORE"
  ></a-entity>
  <a-entity
    id="score"
    position="0 -2 0"
    text="font: exo2bold; align: center; width: 40; color: black; value: 0"
  ></a-entity>
</a-entity>
</a-entity>
```



	<p>Now let's write the function to update the number of targets and score.</p> <p>For this we need to get the text element with the help of <code>querySelector()</code> and passing the id.</p> <p><u>Now to get the particular attribute of a multi-value component, we can specify the attribute name after dot while using <code>getAttribute()</code>.</u></p> <p>To get the "value" attribute of the text component, we can use the <code>getAttribute().value</code>.</p> <p>Once we get the count of the number of targets, we need to decrease it by one once the flight collides with the ring.</p> <p>To decrease it by 1, we need to first convert it into integer numbers using <code>parseInt()</code>.</p> <p>Then we can update the text element with the help of <code>setAttribute()</code> function.</p> <p>Similarly, we can write the function to update the score value by +50 points, everytime the flight touches the ring!</p> <p><i>Guide the student to write two user-defined functions "updateTargets" and "updateScore"</i></p>	
--	---	--

	to update the value of the text to show the score targets remaining in the scene.	
<pre>updateTargets: function () { const element = document.querySelector("#targets"); var count=element.getAttribute("text").value let currentTargets = parseInt(count); currentTargets -= 1; element.setAttribute("text", { value: currentTargets, }); }, updateScore: function () { const element = document.querySelector("#score"); var count=element.getAttribute("text").value let currentScore = parseInt(count); currentScore += 50; element.setAttribute("text", { value: currentScore, }); },</pre>		
	<p>Now, where should we call these functions?</p> <p>Yes. Great!</p> <p>Let's make the rings invisible by setting the visible attribute to false and call these functions.</p> <p><i>Guide the student to update the "isCollided" function.</i></p>	<p>ESR: We can call these two functions in the "collide" event.</p>

```
isCollided: function (elemntId) {
  const element = document.querySelector(elemntId);
  element.addEventListener("collide", (e) => {
    if (elemntId.includes("#ring")) {
      element.setAttribute("visible", false);
      this.updateScore();
      this.updateTargets();
    }
  });
},
```

Now, when do you think the game will be over?

ESR: Once the flight collides with the bird or if the time is up, the game should be over.

Yes, and what can we do about this?

ESR: Varied.

We can add the "Game Over" text and make it visible only when the time is up and it collides with the bird.

Guide the student to add the "Game Over" text and keep the visible attribute value as false.

Guide the student to write the "gameOver" function.

What can we do to crash the flight once the game is over?

ESR: Varied.

Since the flight is a "dynamic-body", and we kept the mass as 0 to avoid falling, we can update the flight's mass.

	<p><i>Guide the student to select the text element and set its visible attribute to true.</i></p> <p><i>Guide the student to select the flight element and set the mass attribute to 1.</i></p>	
<pre> <!-------Game Over-----> <a-entity id="game_over_text" position="0 7.5 0" text="font: exo2bold; align: center; width: 100; color: white;value: Game Over!!!" visible="false" ></a-entity> </pre> <pre> gameOver: function () { var planeEl = document.querySelector("#plane_model"); var element = document.querySelector("#game_over_text"); element.setAttribute("visible", true); planeEl.setAttribute("dynamic-body", { mass: 1 }); }, </pre>		
	<p>Now let's call this function.</p> <p><i>Guide the student to add the else condition for game over in the "startTimer" and "isCollided" function.</i></p>	

```
isCollided: function (elemntId) {
  var element = document.querySelector(elemntId);
  element.addEventListener("collide", (e) => {
    if (elemntId.includes("#ring")) {
      element.setAttribute("visible", false);
      this.updateScore();
      this.updateTargets();
    }
    else {
      this.gameOver();
    }
  });
},
```


```
setInterval(()=> {
  if (duration >=0) {
    minutes = parseInt(duration / 60);
    seconds = parseInt(duration % 60);




    if (minutes < 10) {
      minutes = "0" + minutes;
    }
    if (seconds < 10) {
      seconds = "0" + seconds;
    }



    timerEl.setAttribute("text", {
      value: minutes + ":" + seconds,
    });

    duration -= 1;
  }
  else {
    this.gameOver();
  }
},1000)
```

Now let's test the output.

<div> <div> VIRTUAL FLIGHT 00:00 <small>TIME REMAINING</small> 16 <small>TRAJECTS REMAINING</small> </div> <div> Game Over!!! </div> </div>		SCORE 200
Teacher Guides Student to Stop Screen Share		
<u>WRAP-UP SESSION - 05 Mins</u>		
<div> <div>  </div> <div> Teacher starts slideshow </div> <div> from slide 20 to slide 29 </div> </div>		
Activity details		Solution/Guidelines
<p>Run the presentation from slide 20 to slide 29</p> <p>Following are the wrap-up session deliverables:</p> <ul style="list-style-type: none"> ● Explain the facts and trivias ● Next class challenge ● Project for the day ● Additional Activity 		<p>Guide the student to develop the project and share with us.</p>
Quiz Time - Click on In-Class Quiz		
Question		Answer
<p>Which of the following is NOT a property of the A-Frame text component?</p> <p>A. align</p> <p>B. value</p> <p>C. color</p>		<p>D</p>

D. background	
<p>How can we update the time?</p> <p>A. by taking the text value from the scene and updating the value while the program is running</p> <p>B. by taking the text value from the scene and updating the value before the program is running</p> <p>C. by taking the image from the scene and updating the value while the program is running</p> <p>D. by taking the image from the scene and updating the value before the program is running</p>	A
<p>To set the timer clock to 2 mins, what should the variable "duration" be set to?</p> <p>A. 2</p> <p>B. 120</p> <p>C. 0</p> <p>D. 60</p>	B
<p>● End the quiz panel</p>	
	<p>You get a "hats-off".</p> <p>Alright. See you in the next class.</p> <p><i>Make sure you have given at least 2 Hats Off during the class for:</i></p> <div data-bbox="1019 1266 1312 1367"> <p>Creatively Solved Activities  +10</p> </div> <div data-bbox="1019 1388 1312 1478"> <p>Great Question  +10</p> </div> <div data-bbox="1019 1499 1312 1598"> <p>Strong Concentration  +10</p> </div>

Project Overview	<p>COUNTDOWN TIMER</p> <p>Goal of the Project:</p> <p>In this project, you will add A-Frame text and update the text value using timing events in the project created in the previous class.</p> <p>Story:</p> <p>Your friend always wanted to go for the Scuba Dive, but he is really scared of water and high waves in the ocean. He always wished to travel under the ocean without going under water.</p> <p>Write an A-Frame program to add the A-Frame text and update the values of the text to show the timer clock, coins left to collect and the score in the scene.</p> <p>I am very excited to see how to create a virtual ocean game for your friend.</p> <p>Bye!</p>	
<div>  Teacher ends slideshow </div>		
<div> Teacher Clicks  </div>		

Additional Activities	<p><i>Encourage the student to write reflection notes in their reflection journal using markdown.</i></p> <p>Use these as guiding questions:</p> <ul style="list-style-type: none"> • What happened today? <ul style="list-style-type: none"> ◦ Describe what happened. ◦ The code I wrote. • How did I feel after the class? • What have I learned about programming and developing games? • What aspects of the class helped me? What did I find difficult? 	<p><i>The student uses the markdown editor to write their reflections in a reflection journal.</i></p>
------------------------------	--	--

Activity	Activity Name	Links
Teacher Activity 1	Teacher Reference Code	https://github.com/whitehatjr/PRO-C156-Teacher-Ref
Teacher Activity 2	Final Output Reference	https://curriculum.whitehatjr.com/PRO+Asset/ad61d920ed4c4b458a7c64f60f7f5331.mp4
Student Activity 1	Flight Simulation Stage 4	https://github.com/whitehatjr/PRO-C155-Student-Activity
Project Solution Link	Countdown Timer	https://github.com/whitehatjr/PRO-C156-Project
Teacher Ref. Visual	Visual Aid Link	https://curriculum.whitehatjr.com/Visual

Aid Link		+Project+Asset/PRO_VD/PRO_C156_withcues.html
Teacher Ref. In-Class Quiz	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/36b46cf2-c340-45b2-965d-d9e94fca7935.pdf

