

Lesson: Introduction to HTML5 Canvas

By the end of this lesson, students will have a basic understanding of HTML5 canvas and its capabilities, as well as the basics of drawing and animating on the canvas.

Lesson Outline:

Part 1: Overview of HTML5 Canvas (20 minutes):

- Introduce HTML5 canvas and how it differs from other HTML elements
- Explain the capabilities and limitations of canvas
- Demonstrate some examples of 2D games developed with canvas

Part 2: Understanding the Canvas Coordinate System (10 minutes):

- Introduce the concept of the canvas coordinate system
- Explain how coordinates work on the canvas
- Demonstrate drawing basic shapes on the canvas using coordinates

Part 3: Animation Basics with the Canvas Element (25 minutes):

- Introduce the concept of animation in game development
- Explain the basics of animation, such as frame rate and tweening
- Introduce the requestAnimationFrame function and how to use it for smooth animation on the canvas
- Demonstrate basic animation techniques, such as sprite sheet animation and keyframe animation

Part 4: Conclusion (5 minutes):

- Recap the main points covered in the lesson
- Encourage students to explore and experiment with HTML5 canvas on their own
- Provide resources for further learning

Assessment:

- During the lesson, check for understanding by asking students to participate in class discussions and answer questions
- At the end of the lesson, provide a brief quiz or exercise to assess students' understanding of the topics covered

Note: The hands-on exercises mentioned in the submodule description can be incorporated into the lesson as appropriate, depending on the time and skill level of the students.

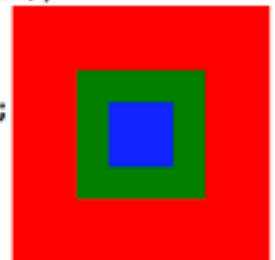
Part 1: Overview of HTML5 Canvas (20 minutes)

Introduction (5 minutes):

HTML5 canvas is a powerful tool for creating graphics on the web. It allows developers to create complex animations, games, and visualizations without the need for plugins or third-party software. One of the key advantages of canvas is its flexibility, allowing developers to create a wide range of 2D and 3D graphics, from simple shapes and lines to complex animations and visualizations.

However, there are also some limitations to using canvas. For example, canvas is not suitable for creating complex user interfaces or handling large amounts of text. Additionally, canvas does not provide built-in support for interactivity or event handling, which can make it difficult to create interactive applications.

```
<html>
<body>
  <canvas id="myCanvas" width="200" height="200" />
  <script>
    var canvas = document.getElementById("myCanvas");
    var ctx = canvas.getContext("2d");
    ctx.fillStyle = "red";
    ctx.fillRect(0, 0, 200, 200);
    ctx.fillStyle = "green";
    ctx.fillRect(50, 50, 100, 100);
    ctx.fillStyle = "blue";
    ctx.fillRect(75, 75, 50, 50);
  </script>
</body>
</html>
```



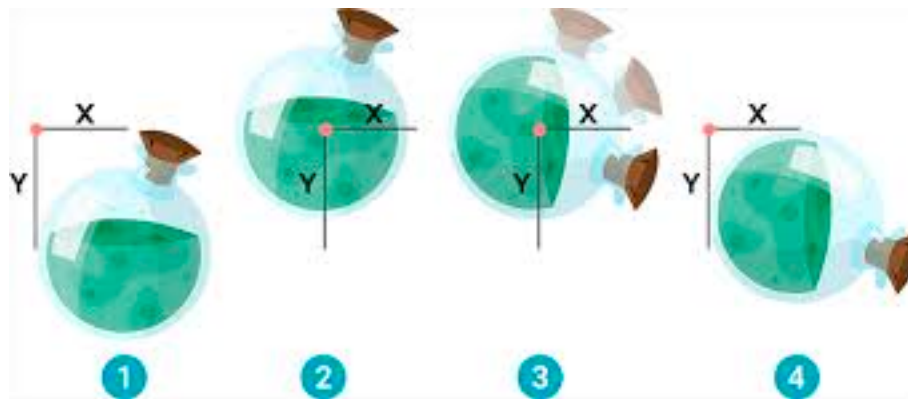
Despite these limitations, canvas remains a popular tool for creating graphics on the web. With the right skills and techniques, developers can use canvas to create engaging and interactive experiences for users. In the following sections, we will explore some of the key concepts and techniques involved in working with canvas, including drawing shapes and images, creating animations, and handling user input.

HTML5 and Its Use in Web and Game Development:

HTML5 is a markup language that has become increasingly popular for creating web-based games. One of its most significant features is the HTML5 canvas element, which allows developers to create 2D and 3D graphics on a web page. This feature has enabled game developers to create complex games that run smoothly in a web browser without requiring plugins like Adobe Flash.

In addition to the canvas element, HTML5 also includes features such as audio and video support, offline storage, and cross-platform compatibility. These features have made it easier for game developers to create games that can be played on a variety of devices, including desktop computers, mobile devices, and tablets. Another advantage of using HTML5 for game development is that it is relatively easy to learn compared to other game development tools like Unity or Unreal Engine. With basic knowledge of HTML, CSS, and JavaScript, developers can quickly get started with creating simple games and gradually build up to more complex games.

Overall, HTML5 has revolutionized game development by making it easier and more accessible to a broader range of developers. With its powerful features and ease of use, HTML5 has become a popular choice for creating web-based games that can be played on a variety of devices.

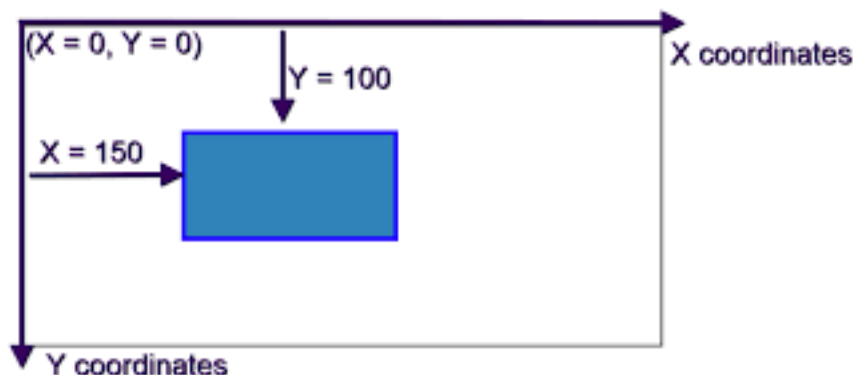


Purpose of HTML5 Canvas:

Canvas is a powerful tool that allows developers to create rich and interactive games and applications on the web. One of the main advantages of canvas is that it provides a platform for creating 2D and 3D graphics, which can enhance the user experience and engagement.

Canvas has several capabilities that make it a great choice for game development. For instance, developers can draw different shapes like lines, rectangles, arcs, and circles on the canvas. They can also add images to the canvas and manipulate them. Additionally, developers can create animations on the canvas by drawing different shapes and images at different intervals.

Despite its many capabilities, canvas has some limitations that developers should be aware of. For example, complex graphics can slow down the performance of the canvas. Therefore, developers need



to be mindful of the graphics they use to ensure optimal performance. Moreover, not all browsers support canvas, so developers should test their games or applications on different browsers to ensure compatibility.

There are several popular 2D games that have been developed using HTML5 canvas. For example, Flappy Bird, which is a popular mobile game, was developed using HTML5 canvas. Angry Birds is another popular game that was originally developed for mobile devices but has also been adapted for web browsers using HTML5 canvas. Cut the Rope is another popular game that was developed using HTML5 canvas.

In summary, HTML5 canvas is a powerful tool for game development on the web. It provides developers with the ability to create 2D and 3D graphics, add images, and create animations. While canvas has some limitations, it has been used successfully to develop many popular games, including Flappy Bird, Angry Birds, and Cut the Rope.

Demonstration (10 minutes):

To demonstrate 2D games developed with canvas, you can show some examples of popular games that were built with this technology, such as Flappy Bird, Angry Birds, and Cut the Rope, as mentioned earlier. You can also demonstrate how canvas is used to create graphics and animation in these games.



For example, you can show how Flappy Bird uses canvas to draw the game's background, obstacles, and the bird character. You can also show how the game uses animation to make the bird flap its wings and move through the obstacles.

Similarly, you can demonstrate how Angry Birds uses canvas to draw the game's background, characters, and objects, such as the birds and pigs. You can also show how the game uses physics to create realistic movement and collision detection.



Finally, you can demonstrate how Cut the Rope uses canvas to draw the game's background, objects, and the character. You can also show how the game uses animation to create movement and interaction between the character and the objects.

By demonstrating these examples, you can help students understand how canvas is used in game development and inspire them to experiment with canvas to create their own games or interactive projects.

Conclusion (5 minutes):

To sum up, in this lesson, we learned that HTML5 canvas is a useful element that allows developers to create 2D and 3D graphics on a web page. We saw how canvas offers several capabilities, such as drawing shapes, adding images, and creating animations. However, canvas also has some limitations, such as performance issues and browser compatibility. We also saw some examples of popular 2D games developed with canvas, such as Flappy Bird, Angry Birds, and Cut the Rope.

In the next section, we will explore the canvas coordinate system and learn how to draw basic shapes on the canvas. Understanding the canvas coordinate system is essential for positioning shapes accurately on the canvas. Drawing basic shapes on the canvas is the foundation for creating more complex graphics and animations.

Part 2: Understanding the Canvas Coordinate System

Introducing the canvas coordinate system is essential for building games with HTML5 canvas. In this section, we'll discuss how coordinates work on the canvas and demonstrate how to draw basic shapes.

Introduction to the Canvas Coordinate System (2 minutes)

The canvas coordinate system is essential to understand when creating games or other graphics using HTML5 canvas. It is a grid that spans the entire canvas area, with the x-axis running horizontally and the y-axis running vertically.

The origin, or starting point, of the canvas coordinate system is located at the top-left corner of the canvas, where both the x and y coordinates are zero (0,0). The x-coordinate increases as you move to the right, and the y-coordinate increases as you move down the canvas.

For example, if you want to draw a shape at the bottom-right corner of the canvas, you would need to set the x-coordinate to the maximum width of the canvas, and the y-coordinate to the maximum height of the canvas.

The coordinate system can be visualized as a graph with x and y axes, with the origin (0,0) located at the bottom-left corner. This can be different from what students may have learned in mathematics, where the origin is usually located at the center of a graph.

It's important to keep in mind that the coordinate system is relative to the canvas itself, and not to the browser window or the webpage. This means that if the canvas is resized, the coordinate system will also adjust accordingly.

Understanding the canvas coordinate system is crucial for creating accurate and precise graphics on the canvas. It allows developers to control the positioning and movement of shapes and objects with precision, making it easier to create complex game mechanics and interactions.

Explaining How Coordinates Work on the Canvas (3 minutes)

To draw on the canvas, we need to know the x and y coordinates of the location where we want to place our shapes. For instance, to draw a rectangle at position (100, 50), we would use the `fillRect()` method and pass in the x and y coordinates, as well as the width and height:

Demonstrating Drawing Basic Shapes Using Coordinates (5 minutes)

Now, let's try drawing some basic shapes using coordinates. We'll start with a rectangle, which we'll create using the `fillRect()` method. We'll pass in the x and y coordinates for the top-left corner of the rectangle and the width and height of the rectangle:

```

// Get the canvas element from the HTML
const canvas = document.getElementById('my-canvas');
// Get the 2D context of the canvas
const ctx = canvas.getContext('2d');

let x = 0; // Define the initial position of a shape

// Define the animation function
function animate() {

    x += 1; // Update the position of the shape

    // Clear the canvas
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    // Draw the shape at the new position
    ctx.fillRect(x, 50, 50, 50);

    // Request the next frame of the animation
    requestAnimationFrame(animate);
}

// Call the animation function to start the animation
requestAnimationFrame(animate);

```

We can also draw circles using the `arc()` method. The `arc()` method takes four parameters: the x and y coordinates of the center of the circle, the radius of the circle, and the starting and ending angles (in radians).

Conclusion (2 minutes)

The canvas coordinate system is the foundation of all shapes and objects in HTML5 canvas. The coordinate system defines a rectangular grid, where the x-coordinate is horizontal, and the y-coordinate is vertical. The top-left corner of the canvas is the origin (0,0), and the x and y coordinates increase as we move to the right and down, respectively.

In this section, we learned how to draw basic shapes on the canvas using coordinates. We used the `fillRect()`, `strokeRect()`, and `clearRect()` methods to draw rectangles and erase parts of the canvas. We also learned how to set the `fillStyle` and `strokeStyle` properties to change the color of our shapes.

Understanding the canvas coordinate system and how to draw basic shapes is crucial for building games with HTML5 canvas. You'll use these concepts to create complex graphics, animations, and interactive content. In the next section, we'll explore how to draw more complex shapes on the canvas, such as circles, arcs, and curves.

Part 4 - Animation Basics with the Canvas Element (25 minutes)

Introduction (5 minutes)

Animation is a crucial aspect of game development that helps create an immersive and engaging game world. Animations can include everything from character movement and object interactions to particle effects and

```

let img = new Image();
img.src = 'spritesheet.png';
let frameIndex = 0;
let framesPerRow = 4;
let numRows = 2;
let frameWidth = img.width / framesPerRow;
let frameHeight = img.height / numRows;

function draw() {
    context.clearRect(0, 0, canvas.width, canvas.height);
    context.drawImage(
        img,
        (frameIndex % framesPerRow) * frameWidth,
        Math.floor(frameIndex / framesPerRow) * frameHeight,
        frameWidth,
        frameHeight,
        100,
        100,
        frameWidth,
        frameHeight);
    frameIndex++;
    requestAnimationFrame(draw);
}

img.onload = function () {
    draw();
};

```


environmental changes. In HTML5 canvas game development, animations are typically created by drawing different frames of an animation in succession at a specific rate.

Understanding Animation Basics (10 minutes)

When it comes to the Canvas HTML element, animation is achieved by updating the positions, properties, or appearance of objects or shapes on the canvas over time. This process involves clearing the canvas, updating the objects' positions or properties, and then redrawing them on the canvas in their new state. This cycle is repeated over and over again, usually at a specific frame rate, to create the illusion of motion.

To create animation in Canvas, developers use the `requestAnimationFrame()` method, which schedules a function to be called before the browser performs the next repaint. The function that is passed to `requestAnimationFrame()` should include the code to update the objects' properties or positions and redraw them on the canvas. By calling `requestAnimationFrame()` repeatedly, developers can create an animation loop that updates and redraws the canvas at a specific frame rate.

Tweening is a popular technique used to create smooth and natural-looking animations in games. Instead of displaying abrupt changes between two frames, tweening fills in the gaps by creating intermediate frames to transition between two keyframes. There are two primary types of tweening: linear and eased. Linear tweening creates equal intervals between each intermediate frame, resulting in a consistent animation speed. Eased tweening, on the other hand, applies a mathematical function to the intervals between frames to create



```

let x = 100;
let y = 100;
let angle = 0;

function draw() {
  context.clearRect(0, 0, canvas.width, canvas.height);
  context.save();
  context.translate(x, y);
  context.rotate(angle);
  context.fillStyle = 'red';
  context.fillRect(-25, -25, 50, 50);
  context.restore();
  requestAnimationFrame(draw);
}

function animate() {
  let startTime = null;
  let duration = 1000;
  function step(timestamp) {
    if (!startTime) startTime = timestamp;
    let progress = timestamp - startTime;
    let t = Math.min(progress / duration, 1);
    x = 100 + t * 200;
    y = 100 + Math.sin(t * Math.PI) * 50;
    angle = t * Math.PI * 2;
    if (progress < duration) {
      requestAnimationFrame(step);
    }
  }
  requestAnimationFrame(step);
}

animate();

```

a more natural-looking animation. Eased tweening can be further customized with various easing functions that apply different levels of acceleration or deceleration to the animation.

Overall, animation is an important aspect of game development that can greatly enhance the gaming experience. Understanding the basics of animation, including frame rate and tweening, is essential for creating compelling and engaging games using HTML5 Canvas.

The requestAnimationFrame Function (5 minutes)

The requestAnimationFrame function is a built-in method in the window object that schedules a callback function to be executed before the next repaint of the browser. This allows for more efficient use of system resources and smoother animations.

To use `requestAnimationFrame` in a canvas animation, we can create a function that will be called each time a new frame needs to be rendered. This function should update the canvas with the new position of objects or any other changes that need to be made.

Here's an example of how to use `requestAnimationFrame` in a canvas animation:

In this example, we first get the canvas element and its 2D context. We then define the initial position of a shape and create the `animate` function that updates the position of the shape and redraws it on the canvas. Inside the `animate` function, we clear the canvas with `ctx.clearRect` and draw the shape at the new position with `ctx.fillRect`. Finally, we call `requestAnimationFrame` again to request the next frame of the animation.

By using `requestAnimationFrame`, we can create smoother and more efficient animations on the canvas.

Basic Animation Techniques (5 minutes)

Sprite sheet animation and keyframe animation are two common techniques used in game development to create animations on the canvas.

Sprite sheet animation is a technique where a single image file containing multiple frames of an animation is used. The game engine then displays each frame in sequence to create the illusion of movement. The sprite sheet can be a grid of frames or a strip of frames, and it's important to define the size of each frame and the number of frames in the sheet.

Here's an example of how to create a sprite sheet animation using the `drawImage()` method:

In this example, we define the `image`, `frameIndex`, `framesPerRow`, `numRows`, `frameWidth`, and `frameHeight` variables. We then use the `drawImage()` method to draw each frame of the sprite sheet in sequence, and we update the `frameIndex` variable to move to the next frame.

Keyframe animation, on the other hand, involves defining the starting and ending positions of an object and creating a sequence of intermediate frames to transition between them. The intermediate frames are called keyframes, and they define the position, scale, and rotation of the object at specific points in time. The animation engine then fills in the gaps between the keyframes to create a smooth animation.

Here's an example of how to create a keyframe animation using the `animate()` method, seen below. In this example, we define the `x`, `y`, and `angle` variables, and we use the `draw()` function to draw a red square at the current position and rotation. We then define the `animate()` function, which uses the `requestAnimationFrame()` method to create a smooth animation by updating the position and rotation of the square over time. We use the `startTime` variable to track the starting time of the animation, and the `duration` variable to define how long the animation should last. We then use the `Math.min()` function to ensure that the animation stops after the duration has passed. Finally, we use the `requestAnimationFrame()` method to call the `step()` function recursively to create a smooth animation.

Demonstration (10 minutes)

In this section, we will delve into the details of implementing basic animation techniques using HTML5 canvas. Animation is an important aspect of game development as it brings life and excitement to games. With HTML5 canvas, it is possible to create simple animations using basic shapes, or more complex animations using sprites and keyframes.

First, we will start by setting up the canvas, which is the drawing area on which we will be creating the animations. We will use the HTML5 canvas element to create a canvas in our HTML file, and then use

JavaScript to access and manipulate the canvas. We will then draw basic shapes such as rectangles, circles, and lines on the canvas.

Next, we will create an animation loop using the `requestAnimationFrame` function. This function is a built-in method in JavaScript that schedules a function to run before the next repaint of the browser. This allows us to create smooth and efficient animations that do not consume too much of the user's device resources. After setting up the animation loop, we will implement two different animation techniques: sprite sheet animation and keyframe animation. Sprite sheet animation involves animating a sequence of images that are arranged in a grid or a sprite sheet. We will use the sprite sheet image to display different frames of the animation at different times to create a fluid animation.

Keyframe animation, on the other hand, involves defining different states or keyframes of the animation at specific times. We will use CSS to define these keyframes and then use JavaScript to control the animation. Keyframe animation can be used to create complex animations such as character movements and transitions. By the end of this section, learners will have a better understanding of how to create animations using HTML5 canvas, and will be able to apply these techniques to their game development projects.

Conclusion (5 minutes)

Sure, let's dive deeper into the different topics covered in this section:

1. Animation basics:
2. We learned that animation is the illusion of motion created by displaying a sequence of images, called frames, in quick succession. In game development, animation is a crucial aspect that adds life and interactivity to games. We also learned about frame rate, which is the number of frames per second that are displayed in an animation. A higher frame rate creates smoother animation, but it also requires more processing power.
3. `requestAnimationFrame` function:
4. We covered the `requestAnimationFrame` function, which is used to optimize the performance of animations on the HTML5 canvas. The function helps to sync the animation loop with the browser's repaint cycle, reducing unnecessary computations and minimizing the load on the CPU and GPU.
5. Basic animation techniques:
6. We demonstrated two basic animation techniques - sprite sheet animation and keyframe animation - and how to implement them using HTML5 canvas.
 - Sprite sheet animation involves using a single image that contains multiple frames of an animation. We learned how to use the `drawImage()` method to display individual frames of the sprite sheet at specific intervals to create the illusion of motion.
 - Keyframe animation involves specifying specific positions, angles, and sizes of an object at different points in time. We learned how to use the `CanvasRenderingContext2D` object's properties and methods, such as the `translate()`, `rotate()`, and `scale()` methods, to create a smooth animation transition between the keyframes.

By understanding and implementing these basic animation techniques, game developers can bring their game characters and environments to life, adding a new layer of interactivity and immersion to their games.

Part 5: Conclusion (5 minutes):

To recap, in this lesson, we covered the basics of HTML5 canvas and its use in game development. We discussed the purpose of HTML5 and its capabilities and limitations. We also learned about the canvas coordinate system and how to draw basic shapes using coordinates. Finally, we explored animation basics in game development, including frame rate, tweening, and using the `requestAnimationFrame` function for smooth animation on the canvas.

As a final point, we encourage you to explore and experiment with HTML5 canvas on your own. There are many resources available online to help you learn more about canvas and its potential in game development. Some useful resources include the Mozilla Developer Network's Canvas API documentation, w3schools.com, and various online tutorials and courses.

We hope you have found this lesson helpful and informative. Good luck with your HTML5 canvas game development endeavors!