Canvas & WebGL

2D 그래픽과 간단한 그래픽 작업이 필요한 경우 Canvas API가 좋은 선택이며, 더 복잡하고 세부적인 3D 그래픽스와 고성능을 요구하는 작업에는 WebGL API가 더 적합합니다.

Canvas API는 간단하고 CPU 기반의 2D 그래픽 작업에 적합하고, WebGL API는 복잡하고 GPU 기반의 3D 그래픽 작업에 더 적합합니다.

Canvas

- Canvas API 특징

특징	설명
2D 그래픽	Canvas API는 주로 2D 그래픽을 위해 설계되었습니다.
HTML5 Canvas 요소 사용	`<canvas>` 태그를 사용하여 HTML 문서에 그래픽을 그립니다.
단순성과 접근성	Canvas는 JavaScript를 사용하여 간단한 2D 그림, 차트, 이미지 조작 등을 손쉽게 구현할 수 있게 해줍니다.
CPU 기반 렌더링	대부분의 Canvas 렌더링은 CPU를 사용하여 처리되며, 이는 복잡한 3D 그래픽스에는 적합하지 않을 수 있습니다.

- Canvas API 기능

기능	설명
2D 그래픽 렌더링	주로 2D 그래픽을 다루는 데 사용됩니다. 직선, 도형, 이미지, 텍스트 등을 그릴 수 있습니다.
이미지 그리기	이미지를 그릴 수 있습니다. 이미지 파일을 로드하거나, `<img>` 요소를 사용하여 이미지를 가져오거나 등이 가능합니다.
텍스트 렌더링	텍스트를 그릴 수 있습니다. 폰트, 크기, 색상 등을 지정하여 텍스트를 화면에 표시할 수 있습니다.
그라디언트 및 패턴	그라디언트와 패턴을 사용하여 그래픽을 그릴 수 있습니다. 선형 그라디언트, 이미지 패턴 등을 사용할 수 있습니다.
픽셀 조작	각 픽셀에 직접 접근하여 색상을 변경하거나 이미지 데이터를 조작할 수 있습니다.
변형 및 애니메이션	그림 요소를 변형하거나 애니메이션을 만들기 위해 2D 변형 함수 및 애니메이션 루프를 사용할 수 있습니다.
이벤트 처리	Canvas 요소에 이벤트 리스너를 추가하여 사용자 상호 작용을 처리할 수 있습니다.
기타	클리핑 영역 설정, 그림자 추가, 선 스타일 및 채우기 스타일 지정 등의 기능도 있습니다.

WebGL

- WebGL API 특징 및 기능

기능	설명
3D 그래픽 렌더링	웹에서 3D 그래픽을 렌더링하기 위한 API입니다.
OpenGL ES 기반	OpenGL ES (임베디드 시스템용 OpenGL)의 웹 버전으로, 하드웨어 가속된 3D 그래픽스 렌더링을 지원합니다.
고급 기능과 성능	더 복잡한 시각적 효과와 높은 성능을 요구하는 3D 애플리케이션을 만들기 위해 사용됩니다.
GPU 기반 렌더링	WebGL 렌더링은 GPU를 사용하여 처리되어, 복잡하고 세부적인 3D 그래픽 작업에 적합합니다.
셰이딩 및 프로그래밍	WebGL을 사용하여 셰이딩 언어를 통해 그래픽 객체를 조작하고, 그래픽 파이프라인을 커스터마이징할 수 있습니다.
버텍스 및 버퍼 조작	버텍스와 버퍼를 조작하여 3D 모델을 정의하고 렌더링할 수 있습니다.
텍스처 매핑	3D 모델에 텍스처를 매핑하여 더욱 현실적인 그래픽 효과를 구현할 수 있습니다.
삼각형 그리기	WebGL을 사용하여 삼각형을 그리고 복잡한 3D 모델을 구성할 수 있습니다.
깊이 및 스텐실 버퍼	깊이 및 스텐실 버퍼를 사용하여 그래픽 객체의 시야 표현 및 렌더링을 조절할 수 있습니다.
쉐이딩 언어	WebGL은 쉐이더 프로그래밍을 지원하여 더욱 다양하고 정교한 그래픽 효과를 구현할 수 있습니다.
풀 스크린 및 VR	WebGL을 사용하여 전체 화면 렌더링 및 가상 현실(VR) 애플리케이션을 개발할 수 있습니다.
특수 효과 및 필터	WebGL을 사용하여 특수 효과와 필터를 적용하여 그래픽을 수정하고 개선할 수 있습니다.
하드웨어 가속	WebGL은 하드웨어 가속을 지원하여 높은 성능을 제공하며, 복잡한 3D 시뮬레이션 및 게임을 구현하는 데 적합합니다.

- Web GPU

Web GPU는 웹 브라우저에서 고성능 그래픽 작업을 수행하기 위한 새로운 웹 표준입니다. 이것은 웹 개발자가 복잡한 그래픽 작업을 수행하는 데 필요한 기능을 제공하는 것을 목표로 하고 있습니다. WebGL의 진화된 버전으로 볼 수 있지만, 좀 더 직관적이고 높은 수준의 추상화를 제공하여 개발자가 더 효율적으로 그래픽 애플리케이션을 작성할 수 있도록 합니다.

특징	설명
저수준 API	Web GPU는 OpenGL과 비슷한 저수준 그래픽 API를 제공합니다. 이는 개발자가 그래픽 하드웨어에 직접 접근하여 더 높은 성능을 얻을 수 있도록 합니다.
병렬 처리 및 비동기 작업	Web GPU는 병렬 처리와 비동기 작업을 통해 다중 코어 및 스레드를 활용하여 작업을 처리합니다. 이는 복잡한 그래픽 작업을 효율적으로 처리할 수 있도록 도와줍니다.

특징	설명
성능 최적화	Web GPU는 하드웨어 가속을 통해 그래픽 작업의 성능을 최적화합니다. 이는 더 빠르고 부드러운 그래픽 애니메이션 및 렌더링을 가능하게 합니다.
크로스 플랫폼 호환성	Web GPU는 여러 플랫폼 및 장치에서 동일한 코드를 실행할 수 있도록 지원합니다. 이는 개발자가 웹 애플리케이션을 다양한 디바이스 및 환경에서 쉽게 배포할 수 있도록 합니다.
개발자 편의성	Web GPU는 간단하고 직관적인 API를 제공하여 개발자가 빠르게 그래픽 애플리케이션을 작성할 수 있도록 합니다. 또한 높은 수준의 추상화를 제공하여 복잡한 작업을 간단하게 처리할 수 있도록 합니다.

- webgl 라이브러리

종류	설명
Three.js	WebGL을 사용하기 위한 가장 인기 있는 고수준 라이브러리 중 하나입니다. Three.js는 복잡한 3D 그래픽을 쉽게 만들고 조작할 수 있게 해줍니다.
Babylon.js	게임 및 인터랙티브 3D 응용 프로그램을 개발하기 위한 강력한 프레임워크입니다. Babylon.js는 Three.js와 유사하게 사용하기 쉬운 API를 제공합니다.
PixiJS	주로 2D 웹 그래픽을 위한 라이브러리로, 성능이 매우 뛰어나며 모바일과 데스크톱에서 훌륭한 성능을 발휘합니다.
PlayCanvas	웹 기반 게임 및 3D 응용 프로그램 개발을 위한 오픈 소스 엔진이자 툴셋입니다. PlayCanvas는 클라우드 기반의 IDE도 제공합니다.
Regl	Three.js나 Babylon.js와는 다르게, Regl은 WebGL을 좀 더 직접적으로 다루며, 좀 더 유연하지만 사용하기 어려울 수 있습니다.
A-Frame	웹 기반 가상 현실(VR) 응용 프로그램을 만들기 위한 프레임워크입니다. A-Frame은 Three.js 위에서 작동하며, VR 콘텐츠 개발을 쉽게 해줍니다.

- theejs 와 바빌론

기능	바빌론 (Babylon.js)	Three.js
렌더러	WebGL을 직접 조작하는 저수준 API를 사용	Three.js의 추상화된 렌더링 API를 사용하여 웹 그래픽을 구현
커뮤니티 및 생태계	비교적 작고, 그러나 활발한 커뮤니티와 생태계	Three.js는 매우 크고 활성화된 커뮤니티를 가짐
성능 및 최적화	높은 수준의 성능 최적화 및 퍼포먼스	Three.js보다 상대적으로 최적화된 성능
문서화 및 학습 곡선	문서화가 잘 되어 있으며 상대적으로 쉬운 학습 곡선	문서화와 학습 리소스가 풍부하지만 학습 곡선이 다소 가팔름
기능 및 모듈화	다양한 내장 기능 및 모듈화된 아키텍처	유연한 아키텍처를 가지고 있으며 필요한 기능을 선택적으로 사용 가능

기능	바빌론 (Babylon.js)	Three.js
확장성 및 유연성	상대적으로 쉽게 확장 가능하고 유연한 아키텍처	유연성과 확장성이 높은 아키텍처를 가짐
지원하는 파일 형식	glTF, Babylon Scene 파일, OBJ, STL 등 다양한 파일 형식	OBJ, STL, glTF, FBX 등 다양한 파일 형식을 지원
GUI 및 에디터	Babylon.js Editor 및 GUI 라이브러리를 제공	Three.js에는 비공식적인 GUI 및 에디터가 있지만, 공식적인 것은 없음

```
<!DOCTYPE html>
<html>
<head>
  <title>Canvas API Example</title>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css2?
family=Noto+Sans+KR:wght@400&display=swap">
  <style>
    body {
      font-family: 'Noto Sans KR', sans-serif;
      margin: 0; padding: 0;
    }
  </style>
</head>
<body>
  <div>
    <p>1. Canvas</p>
    <canvas id="myCanvas" width="200" height="200" style="border:1px solid
#000000;"></canvas>
  </div>
  <script>
    let canvas2d = document.getElementById("myCanvas");
    let ctx = canvas2d.getContext("2d");

    // 원 그리기
    ctx.beginPath();
    ctx.arc(100, 70, 50, 0, 2 * Math.PI);
    ctx.fillStyle = "blue";
    ctx.fill();
    ctx.stroke();

    // 한글 텍스트 그리기
    ctx.font = "20px 'Noto Sans KR', sans-serif";
    ctx.fillText("CanvasAPI", 50, 150);
  </script>
  <div>
    <p>2. Canvas(p5js)</p>
    <div id="p5-container"></div>
  </div>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.4.0/p5.js">
</script>
  <script>
```

```
    function setup() {
      let cnv = createCanvas(400, 400);
      cnv.parent('p5-container'); // p5.js 캔버스를 지정된 요소에 추가
      background(220);
      fill(0, 0, 255);
      noStroke();
      ellipse(width / 2, height / 2, 200, 200); // 중심에 원 그리기

      fill(0);
      textSize(20);
      textAlign(CENTER, CENTER);
      text("p5js!", width / 2, height / 2); // 텍스트 그리기
    }
    setup();
  </script>
  <div>
    <p>2. WebGL</p>
    <canvas id="glCanvas"></canvas>
  </div>
  <script>
    let canvas3d = document.getElementById("glCanvas");
    let gl = canvas3d.getContext("webgl");

    // 캔버스의 배경 색상 설정
    gl.clearColor(0.0, 0.0, 0.0, 1.0);
    gl.clear(gl.COLOR_BUFFER_BIT);
  </script>
  <div>
    <p>4. WebGL (babylonjs)</p>
  </div>
  <!-- Import Babylon.js -->
  <script src="https://cdn.babylonjs.com/babylon.max.js"></script>
  <!-- Import WebGPU adapter for Babylon.js -->
  <script
src="https://preview.babylonjs.com/webgpuAdapter/babylonjs.webgpu.min.js">
</script>
  <canvas id="renderCanvas"></canvas>
    <script>
        // Create Babylon.js engine
        let engine = new
BABYLON.Engine(document.getElementById("renderCanvas"), true);

        // Create scene
        let scene = new BABYLON.Scene(engine);

        // Create camera
        let camera = new BABYLON.ArcRotateCamera("camera", -Math.PI / 2,
Math.PI / 2, 5, new BABYLON.Vector3(0, 0, 0), scene);
        camera.attachControl(true);

        // Create light
        let light = new BABYLON.HemisphericLight("light", new
BABYLON.Vector3(0, 1, 0), scene);
```

```
        // Create sphere
        let sphere = BABYLON.MeshBuilder.CreateSphere("sphere", {
diameter: 1 }, scene);

        // Apply materials
        let material = new BABYLON.StandardMaterial("material", scene);
        material.diffuseColor = new BABYLON.Color3(1, 0, 0); // Red color
        sphere.material = material;

        // Run render loop
        engine.runRenderLoop(function () {
            scene.render();
        });

        // Resize canvas with window
        window.addEventListener("resize", function () {
            engine.resize();
        });
    </script>
  <div>
    <p>3. WebGL (Threejs)</p>
  </div>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r128/three.min.js">
</script>
  <script>
    window.onload = function() {
      const scene = new THREE.Scene();
      const camera = new THREE.PerspectiveCamera(75, window.innerWidth /
window.innerHeight, 0.1, 1000);
      const renderer = new THREE.WebGLRenderer({ antialias: true });
      renderer.setSize(window.innerWidth, window.innerHeight);
      document.body.appendChild(renderer.domElement);

      // 카메라를 z축 방향으로 이동시킵니다.
      camera.position.z = 300;

      let textMesh; // 텍스트 메시 변수를 전역 변수로 선언합니다.

      const fontLoader = new THREE.FontLoader();
      fontLoader.load('./Do_Hyeon_Regular.json', function(font) {
        const textGeometry = new THREE.TextGeometry('Hello! 하이!', {
          font: font,
          size: 46,
          height: 5,
          curveSegments: 9,
          bevelEnabled: true,
          bevelThickness: 1,
          bevelSize: 1,
          bevelOffset: 0,
          bevelSegments: 5
        });

        const textMaterial = new THREE.MeshBasicMaterial({ color: 0x00ff00
```

```
});
        textMesh = new THREE.Mesh(textGeometry, textMaterial); // 전역 변수에 할당합니다.

        scene.add(textMesh);

        textMesh.position.x = -100;
        textMesh.position.y = 10;
        textMesh.position.z = -100;

        animate(); // 폰트가 로드된 후에 애니메이션을 시작합니다.
      });

      // 마우스 이벤트 처리
      let isDragging = false;
      let previousMousePosition = {
        x: 0,
        y: 0
      };

      document.addEventListener('mousedown', (event) => {
        isDragging = true;
      });

      document.addEventListener('mousemove', (event) => {
        if (!isDragging) return;

        let deltaMove = {
          x: event.offsetX - previousMousePosition.x,
          y: event.offsetY - previousMousePosition.y
        };

        if (deltaMove.x != 0) {
          textMesh.rotation.y += deltaMove.x * 0.01;
        }

        previousMousePosition = {
          x: event.offsetX,
          y: event.offsetY
        };
      });

      document.addEventListener('mouseup', (event) => {
        isDragging = false;
      });

      function animate() {
        requestAnimationFrame(animate);
        renderer.render(scene, camera);
      }
    };
  </script>
</body>
</html>
```