Tarea 1.1 – Investigación de una Librería Grafica para Web

Babylon.js

Descripción general

Babylon.js es un motor 3D de código abierto para la web, creado inicialmente por Microsoft en 2013. Está desarrollado en TypeScript, pero se utiliza principalmente con JavaScript. Permite generar gráficos 3D y experiencias interactivas directamente en el navegador sin necesidad de instalar complementos adicionales, gracias a su soporte para WebGL y WebGPU.

Características clave

01 Compatibilidad

Babylon es compatible con navegadores modernos por qué funciona con WebGL 1, 2 y WebGPU

02 PBR

Soporta materiales PBR para superficies realistas, iluminación física, sombras dinámicas y efectos de post-procesamiento (Bloom, SSAO, Motion Blur).

03 Animación

Ofrece animaciones esqueléticas, morph targets, blending de animaciones y un sistema de timeline para controlar escenas.

04 Fisicas

Incluye integración con motores como Havok, Cannon.js o Ammo.js, permitiendo simular colisiones, gravedad y ragdolls.

Características clave

05 Modelos 3D

Soporte de modelos 3D estándar como glTF, GLB, OBJ y STL.

06 GUI

Proporciona controles 2D (botones, paneles, HUDs) directamente en el canvas 3D.

07 Realidad virtual y aumentada

Se puede crear experiencias inmersivas en VR/AR desde el navegador.

08 Herramientas

- Playground online para probar código
- Inspector integrado para depuración visual
- "Lightweight viewer" para mostrar modelos rápidamente

¿Cuando usar Babylon.js?

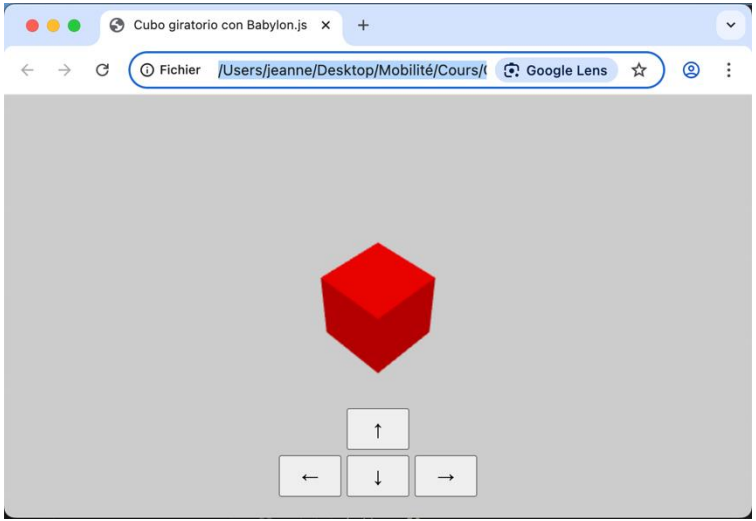
Es ideal para aplicaciones y juegos 3D en navegador que requieren calidad visual y un flujo de trabajo rápido.

El Playground facilita la experimentación y la documentación es extensa.

Si necesita PBR, herramientas integradas y compatibilidad amplia de navegadores, es una excelente elección.

Enlace del código: https://github.com/jeannelgb/Graficas-por-Computadora/blob/main/index.html

Documentación



El programa propuesto es un programa permitiendo de hacer rotaciones a un cubo con ayuda del teclado y de botones.
Con el panel táctil se puede también hacer zoom del cubo.

Enlace del código: https://github.com/jeannelgb/Graficas-por-Computadora/blob/main/index.html

Documentación

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Cubo giratorio con Babylon.js</title>
  <script src="https://cdn.babylonjs.com/babylon.js"></script>
  <style>
    html, body {
      width: 100%;
      height: 100%;
      margin: 0;
      overflow: hidden;
    }
    canvas {
      width: 100%;
      height: 100%;
      display: block;
    }

    /* Botones */
    .controls {
```

El código es escrito en HTML, con la codificación UTF-8 para permitir la visualización correcta de los acentos. Hay también un script para incluir la librería Babylon.js.

Añadiendo la etiqueta <style> se pueden definir los estilos gráficos de los elementos de la página html.

Enlace del código: https://github.com/jeannelgb/Graficas-por-Computadora/blob/main/index.html

Documentación

```
<script>
  const canvas = document.getElementById("canvas");
  const engine = new BABYLON.Engine(canvas, true);

  const createScene = () => {
    const scene = new BABYLON.Scene(engine);
    scene.clearColor = new BABYLON.Color3(0.8, 0.8, 0.8);

    // Camára
    const camera = new BABYLON.ArcRotateCamera("camera", Math.PI/4, Math.PI/4, 6, BABYLON.Vector3.Zero(), scene);
    camera.attachControl(canvas, true);

    // Luz
    const light = new BABYLON.HemisphericLight("light", new BABYLON.Vector3(0.5, 1, 0.5), scene);

    const box = BABYLON.MeshBuilder.CreateBox("box", { size: 2 }, scene);
    const mat = new BABYLON.StandardMaterial("mat", scene);
    mat.diffuseColor = new BABYLON.Color3(1, 0, 0);
    box.material = mat;

    const rotationStep = 0.05;

    // Deplazamiento con teclado
    window.addEventListener("keydown", (event) => {
      switch(event.key) {
        case "ArrowUp": box.rotation.x -= rotationStep; break;
        case "ArrowDown": box.rotation.x += rotationStep; break;
        case "ArrowLeft": box.rotation.y -= rotationStep; break;
        case "ArrowRight": box.rotation.y += rotationStep; break;
      }
```

Luego hay una etiqueta <script> donde el script para Babylon esta escrito.
En primer lugar, se cree el motor Babylon, que es responsable de render la escena en 3D y de gestionar las interacciones. El parámetro *true* activa el antialiasing para alisar los objetos.

Enlace del código: https://github.com/jeannelgb/Graficas-por-Computadora/blob/main/index.html

Documentación

```
<script>
  const canvas = document.getElementById("canvas");
  const engine = new BABYLON.Engine(canvas, true);

  const createScene = () => {
    const scene = new BABYLON.Scene(engine);
    scene.clearColor = new BABYLON.Color3(0.8, 0.8, 0.8);

    // Camára
    const camera = new BABYLON.ArcRotateCamera("camera", Math.PI/4, Math.PI/4, 6, BABYLON.Vector3.Zero(), scene);
    camera.attachControl(canvas, true);

    // Luz
    const light = new BABYLON.HemisphericLight("light", new BABYLON.Vector3(0.5, 1, 0.5), scene);

    const box = BABYLON.MeshBuilder.CreateBox("box", { size: 2 }, scene);
    const mat = new BABYLON.StandardMaterial("mat", scene);
    mat.diffuseColor = new BABYLON.Color3(1, 0, 0);
    box.material = mat;

    const rotationStep = 0.05;

    // Deplazamiento con teclado
    window.addEventListener("keydown", (event) => {
      switch(event.key) {
        case "ArrowUp": box.rotation.x -= rotationStep; break;
        case "ArrowDown": box.rotation.x += rotationStep; break;
        case "ArrowLeft": box.rotation.y -= rotationStep; break;
        case "ArrowRight": box.rotation.y += rotationStep; break;
      }
```

La función *createScene* permite configurar et crear la escena: un contenedor para todos los objetos 3D, la luz y la cámara esta creado, y se ha definido un color de fondo.
La escena usa una cámara orbital (*ArcRotateCamera*) para girar alrededor de un punto central. La función *camera.attachControl()* permite controlar la cámara con el ratón.

Enlace del código: https://github.com/jeannelgb/Graficas-por-Computadora/blob/main/index.html

Documentación

```
<script>
  const canvas = document.getElementById("canvas");
  const engine = new BABYLON.Engine(canvas, true);

  const createScene = () => {
    const scene = new BABYLON.Scene(engine);
    scene.clearColor = new BABYLON.Color3(0.8, 0.8, 0.8);

    // Camára
    const camera = new BABYLON.ArcRotateCamera("camera", Math.PI/4, Math.PI/4, 6, BABYLON.Vector3.Zero(), scene);
    camera.attachControl(canvas, true);

    // Luz
    const light = new BABYLON.HemisphericLight("light", new BABYLON.Vector3(0.5, 1, 0.5), scene);

    const box = BABYLON.MeshBuilder.CreateBox("box", { size: 2 }, scene);
    const mat = new BABYLON.StandardMaterial("mat", scene);
    mat.diffuseColor = new BABYLON.Color3(1, 0, 0);
    box.material = mat;

    const rotationStep = 0.05;

    // Deplazamiento con teclado
    window.addEventListener("keydown", (event) => {
      switch(event.key) {
        case "ArrowUp": box.rotation.x -= rotationStep; break;
        case "ArrowDown": box.rotation.x += rotationStep; break;
        case "ArrowLeft": box.rotation.y -= rotationStep; break;
        case "ArrowRight": box.rotation.y += rotationStep; break;
      }
```

Se ha añadido la luz para iluminar los objetos *(HemisphericLight)* y el cubo con *BABYLON.MeshBuilder.CreateBox("box", {size:2}) scene).* Así, un lado del cubo mide 2 unidades y se aplica un material y un color rojo sobre el cubo.
La variable rotationStep corresponde a la cantidad de rotación aplicada a cada clic en un botón.

Enlace del código: https://github.com/jeannelgb/Graficas-por-Computadora/blob/main/index.html

Documentación

```
<script>
  const canvas = document.getElementById("canvas");
  const engine = new BABYLON.Engine(canvas, true);

  const createScene = () => {
    const scene = new BABYLON.Scene(engine);
    scene.clearColor = new BABYLON.Color3(0.8, 0.8, 0.8);

    // Camára
    const camera = new BABYLON.ArcRotateCamera("camera", Math.PI/4, Math.PI/4, 6, BABYLON.Vector3.Zero(), scene);
    camera.attachControl(canvas, true);

    // Luz
    const light = new BABYLON.HemisphericLight("light", new BABYLON.Vector3(0.5, 1, 0.5), scene);

    const box = BABYLON.MeshBuilder.CreateBox("box", { size: 2 }, scene);
    const mat = new BABYLON.StandardMaterial("mat", scene);
    mat.diffuseColor = new BABYLON.Color3(1, 0, 0);
    box.material = mat;

    const rotationStep = 0.05;

    // Deplazamiento con teclado
    window.addEventListener("keydown", (event) => {
      switch(event.key) {
        case "ArrowUp": box.rotation.x -= rotationStep; break;
        case "ArrowDown": box.rotation.x += rotationStep; break;
        case "ArrowLeft": box.rotation.y -= rotationStep; break;
        case "ArrowRight": box.rotation.y += rotationStep; break;
      }
```

Para hacer la rotación con el teclado se necesita añadir *EventListener* a los botones afectados del teclado y cuando se pulsa un botón, se realiza una rotación en la buena dirección.

Enlace del código: https://github.com/jeannelgb/Graficas-por-Computadora/blob/main/index.html

Documentación

```
    // Deplazamiento con botones
    document.getElementById("up").addEventListener("click", () => box.rotation.x -= rotationStep);
    document.getElementById("down").addEventListener("click", () => box.rotation.x += rotationStep);
    document.getElementById("left").addEventListener("click", () => box.rotation.y -= rotationStep);
    document.getElementById("right").addEventListener("click", () => box.rotation.y += rotationStep);

    return scene;
  };

  const scene = createScene();
  engine.runRenderLoop(() => scene.render());
  window.addEventListener("resize", () => engine.resize());
</script>
</body>
</html>
```

Finalmente, se pone *EventIListener* también a los botones HTML para hacer el mismo que con el teclado.
La función *engine.runRenderLoop() => scene.render()* permite dibujar la escena de forma continua para que todas las rotaciones y movimientos se pueden ver.

Referencias

- **Wikipédia :** https://fr.wikipedia.org/wiki/Babylon.js
- **Babylon.js :** https://www.babylonjs.com
- **Documentación de Babylon :** https://doc.babylonjs.com/setup/support/webGPU
- **Medium :** https://babylonjs.medium.com/introducing-babylon-js-8-0-77644b31e2f9
- **Windows Blog :** https://blogs.windows.com/windowsdeveloper/2025/03/27/announcing-babylon-js-8-0/