

Aplicación de citas famosas

Amador Ortega Christian Amauri

Christian.amadoro@alumno.buap.mx

13 / 02 / 2023

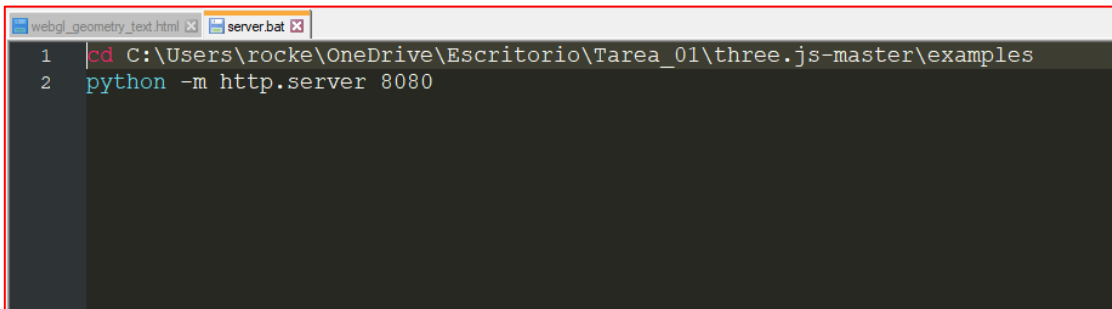
Resumen

Se presenta una aplicación multimedia en HTML y JavaScript que muestra aleatoriamente (*referencia [3]*) una de diez citas posibles. Se integra texto de modo gráfico (texto 3D), un sintetizador de voz que reproduce el texto (*referencia [1]*) y una pequeña animación utilizando la librería three.js (*referencia [2]*) de JavaScript.

Para la realización de esta aplicación, se trabajó directamente sobre uno de los ejemplos de three.js (`webgl_geometry_text.html`), modificándolo para cumplir con el propósito descrito con anterioridad.

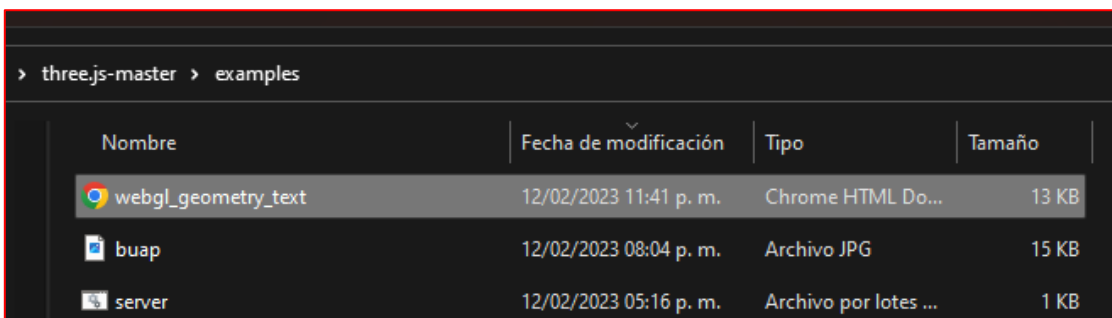
Para ejecutar localmente los ejemplos de esta librería, debemos descargarla de threejs.org (*referencia [4]*) y luego descomprimirla en alguna ruta de nuestro equipo. También tenemos que tener instalada en nuestro equipo, la versión 3.10 de Python. Una vez descargada e instalada, debemos emular un servidor, creando un archivo `.bat` y guardándolo en la carpeta donde descomprimimos la librería de three.js, así:




`server.bat` :



```
webgl_geometry_text.html  server.bat
1 cd C:\Users\rocke\OneDrive\Escritorio\Tarea_01\three.js-master\examples
2 python -m http.server 8080
```

`webgl_eometry_text.html` :



> three.js-master > examples			
Nombre	Fecha de modificación	Tipo	Tamaño
 webgl_geometry_text	12/02/2023 11:41 p. m.	Chrome HTML Do...	13 KB
 buap	12/02/2023 08:04 p. m.	Archivo JPG	15 KB
 server	12/02/2023 05:16 p. m.	Archivo por lotes ...	1 KB

```
webgl_geometry_text.html x server.bat x
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4 <title>three.js webgl - geometry - text</title>
5 <meta charset="utf-8">
6 <meta name="viewport" content="width=device-width, user-scalable=
7 <link type="text/css" rel="stylesheet" href="main.css">
8 </head>
9 <body>
10
11 <div id="info">
12
13 
14 <h1><b><i>Amador Ortega Christian Amauri <br> <br> 20192782
15 <hr width = "500">
16 <hr width = "380">
17
18 </div>
19
20
21 <!-- Import maps polyfill -->
22 <!-- Remove this when import maps will be widely supported -->
23 <script async src="https://unpkg.com/es-module-shims@1.3.6/dist
24
25 <script type="importmap">
26 {
27 "imports": {
28 "three": "../build/three.module.js",
29 "three/addons/": "../jsm/"
30 }
31 }
32 </script>
33
34 <script type="module"> // !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
35
36 import * as THREE from 'three';
```

Objetivo:



Introducción

> Por qué una aplicación de citas famosas?

Es un buen punto de partida para comenzar a crear aplicaciones multimedia, pues integra elementos muy básicos y didácticos.

Elementos:

> sintetizador:

JavaScript cuenta con diversas funciones multimedia, entre ellas está el sintetizador de voz que utilizaremos, a este se le puede definir el volumen de salida, el tono de voz, velocidad de voz, e incluso podemos seleccionar entre varias voces (aunque en este ejemplo simplemente usaremos la voz estándar).

> three.js:

Otra de las potentes librerías que tiene JavaScript es Three.js, esta sirve para crear e interactuar con modelos 3D. Incluye desde cosas muy básicas como cubos, hasta complejas como representaciones de armaduras, planetas, luces, sombras, texturas y demás...

> el diccionario de citas:

La forma en la que guardaremos nuestras citas es relacionándolas con su autor en un diccionario, y a los autores los relacionaremos en otro diccionario con un índice. Más adelante explicaremos esto a detalle...

Diseño

> Información:

Para esta pequeña sección de información, simplemente agregamos una etiqueta de imagen (nótese que la imagen debe tener formato .jpg, llamarse "buap" y tiene un tamaño de 347 x 121 pixeles) , una de texto y 2 <HR>'s como se muestra a continuación:

```
7      <link type="text/css" rel="stylesheet" href="main.css" />
8      </head>
9      <body>
10
11      <div id="info">
12
13          <IMG SRC="buap.jpg" ALIGN = "LEFT" BORDER=0 ALT="NOMBRE">
14          <H1><B><I>Amador Ortega Christian Amauri <BR> <BR> 201927821 <I><B></H1>
15          <HR width = "500">
16          <HR width = "380">
17
18      </div>
19
20
```

El código de color del fondo es: 0x113f67

> el diccionario de citas:

- ❖ Las citas se guardaron mediante un diccionario que relaciona_ autor-cita
- ❖ Los autores se guardaron en otro diccionario que relaciona: índice-autor

Para que al momento de obtener una cita aleatoria, se obtenga primero al autor mediante Math.random(); y luego la cita relacionada con el autor relacionado con el índice que se obtuvo:

```
42
43      //Mis frases célebres...////////////////////
44      const indices = new Map();
45      indices.set(0,"Gaston Berger");      indices.set(1,"Henry Ford");
46      indices.set(2,"Philip Roth");         indices.set(3,"Steve Jobs");
47      indices.set(4,"Lao Tse");             indices.set(5,"Peter Drucker");
48      indices.set(6,"Francis Bacon");       indices.set(7,"Ralph Waldo");
49      indices.set(8,"Gerard Duelo");        indices.set(9,"Jeffrey Eugenides");
50
51      const citas = new Map();
52      citas.set("Gaston Berger","El jefe de empresa debe ser un filosofo en accion");
53      citas.set("Henry Ford" ,"No encuentres la falta; encuentra el remedio");
54      citas.set("Philip Roth" ,"Deja de preocuparte por envejecer y piensa en crecer");
55      citas.set("Steve Jobs" ,"Hagamos una muesca en el universo");
56      citas.set("Lao Tse" ,"Para dirigir personas, camina detras de ellas");
57      citas.set("Peter Drucker","El resultado de un buen negocio es un cliente satisfecho");
58      citas.set("Francis Bacon","La ocasion hay que crearla, no esperar a que llegue");
59      citas.set("Ralph Waldo" ,"Unicamente la obediencia tiene derecho al mando");
60      citas.set("Gerard Duelo" ,"Los buenos asesores son caros, pero los malos aun mas");
61      citas.set("Jeffrey Eugenides","Lo peor de la religion, era la gente religiosa");
62
```

> *sintetizador:*

La creación del texto y el sintetizador se activará cuando se oprima una tecla, por lo que el código que reproduce la voz debe estar dentro de una función evento (PressKey).

Código usado para el momento de reproducir voz:

```
function onDocumentKeyPress( event ) { // C I T A   A L E A T O R I A . . . . . (mediante
    const keyCode = event.which;

    // tecla de borrado
    if ( keyCode == 8 ) {
        event.preventDefault();
    } else {
        var aux = Math.floor( Math.random() * (10 - 0) + 0); // variable aleatoria
        text = citas.get(indices.get(aux)); // obtenemos cita, mediante el autor
        autor = "- "+indices.get(aux); // obtenemos autor mediante su índice
        speechSynthesis.speak(new SpeechSynthesisUtterance(citas.get(indices.get(aux)))); //
        speechSynthesis.speak(new SpeechSynthesisUtterance("dicho por"+autor)); // autor
        refreshText(); // crea texto con frase y autor
        refresh_animation(); // crea esferas
    }
}
```

Como podemos observar, primero obtenemos un número entero entre 1 y 10 mediante Math.random(), luego usamos ese número para obtener al autor seleccionado, luego usamos ese autor para obtener su cita relacionada. Luego mandamos ambos datos de tipo String a la función de sintetización de voz.

> Creación de texto 3D:

Para la creación de texto 3D, simplemente mandamos la frase obtenida a la función ya creada en el código "createText()", y para mostrar también al autor (mediante otro texto geométrico), duplicamos esta porción de código ya antes creada pero adaptándola al nuevo valor del string (el autor). Así:

```
264
265
266 // PARA CREAR LA CITA
267 textGeo = new TextGeometry( text, [
268     font: font,
269     size: size,
270     height: height,
271     curveSegments: curveSegments,
272     bevelThickness: bevelThickness,
273     bevelSize: bevelSize,
274     bevelEnabled: bevelEnabled
275 ] );
276
277 textGeo.computeBoundingBox();
278 const centerOffset = - 0.5 * ( textGeo.boundingBox.max.x - textGeo.boundingBox.min.x ); //CENTRO
279 textMesh1 = new THREE.Mesh( textGeo, materials );
280 textMesh1.position.x = centerOffset;
281 textMesh1.position.y = hover;
282 textMesh1.position.z = 0;
283 textMesh1.rotation.x = 0;
284 textMesh1.rotation.y = Math.PI * 2;
285 group.add( textMesh1 );
286
287 //PARA CREAR AL AUTOR
288 textGeo2 = new TextGeometry( autor, [
289     font: font,
290     size: size,
291     height: height,
292     curveSegments: curveSegments,
293     bevelThickness: bevelThickness,
294     bevelSize: bevelSize,
295     bevelEnabled: bevelEnabled
296 ] );
297
298 textGeo2.computeBoundingBox();
299 const centerOffset2 = - 0.5 * ( textGeo2.boundingBox.max.x - textGeo2.boundingBox.min.x );
300 textMesh3 = new THREE.Mesh( textGeo2, materials );
301 textMesh3.position.x = centerOffset2;
302 textMesh3.position.y = 25;
303 textMesh3.position.z = 0;
304 textMesh3.rotation.x = 0;
305 textMesh3.rotation.y = Math.PI * 2;
306 group.add( textMesh3 );
307
308 textMesh3.rotation.x = 0;
309 textMesh3.rotation.y = Math.PI * 2;
310 group.add( textMesh3 );
311
312 if ( mirror ) {
313     textMesh2 = new THREE.Mesh( textGeo, materials );
314     textMesh2.position.x = centerOffset;
315     textMesh2.position.y = - hover;
316     textMesh2.position.z = height;
317     textMesh2.rotation.x = Math.PI;
318     textMesh2.rotation.y = Math.PI * 2;
319     group.add( textMesh2 );
320
321     textMesh4 = new THREE.Mesh( textGeo2, materials );
322     textMesh4.position.x = centerOffset2;
323     textMesh4.position.y = - 25;
324     textMesh4.position.z = height;
325     textMesh4.rotation.x = Math.PI;
326     textMesh4.rotation.y = Math.PI * 2;
327     group.add( textMesh4 );
328
329 }
330
331 } // fin de la función createText();
332
```

(las partes seleccionadas son las partes duplicadas para mostrar el nombre del autor)

Finalmente, en la función refreshText() agregamos lo mismo que se aplica para el texto original, a nuestra copia para el autor (mesh3 y mesh4):

```
329 = function refreshText() {
330     group.remove( textMesh1 );
331     group.remove( textMesh3 );
332     if ( mirror ) group.remove( textMesh2 );
333     if ( mirror ) group.remove( textMesh4 );
334     if ( ! text ) return;
335     createText();
336 }
337
```

> Esferas

Para agregar la pequeña animación de las esferas, replicamos la idea de la creación de texto; definimos una función de creación de animación (create_animation()) y una función de actualización de animación (refresh_text()):

```
337
338 = function create_animation(){//      E S F E R A S . . . . .
339 =     for(var i = 0;i<num_esferas;i++){
340         esferas[i] = new THREE.Mesh(
341             new THREE.SphereGeometry(radius_esfera,20,20),
342             new THREE.MeshBasicMaterial({color: clr ,opacity:0.5,transparent:false}))
343         );
344         group.add(esferas[i]);
345     }
346
347     for(var i=0;i<=num_esferas;i++){
348         esferas[i].position.x = xmintext+5/*+jth*/+inc_x+10*i*inc_x    -10;
349         esferas[i].position.y =radio_espiral*Math.cos(jth*inc_theta+i*2*Math.PI/num_esferas)+50;
350         esferas[i].position.z =radio_espiral*Math.sin(jth*inc_theta+i*2*Math.PI/num_esferas);
351     }
352 }
353
354 = function refresh_animation(){//      E S F E R A S . . . . .
355     jth=jth+2;
356     for(var i=0;i<=num_esferas;i++) group.remove(esferas[i]);
357     create_animation();
358 }
359
```


La función de actualización de animación, llama a la de creación, y la de actualización se manda a llamar desde la función onPointerMove(); para que al mover el texto con el cursos, las esferas roten alrededor de él:

```
370 = function onPointerMove( event ) {
371     if ( event.isPrimary === false ) return;
372     pointerX = event.clientX - windowHalfX;
373     targetRotation = targetRotationOnPointerDown +
374     refresh_animation();
375 }
```

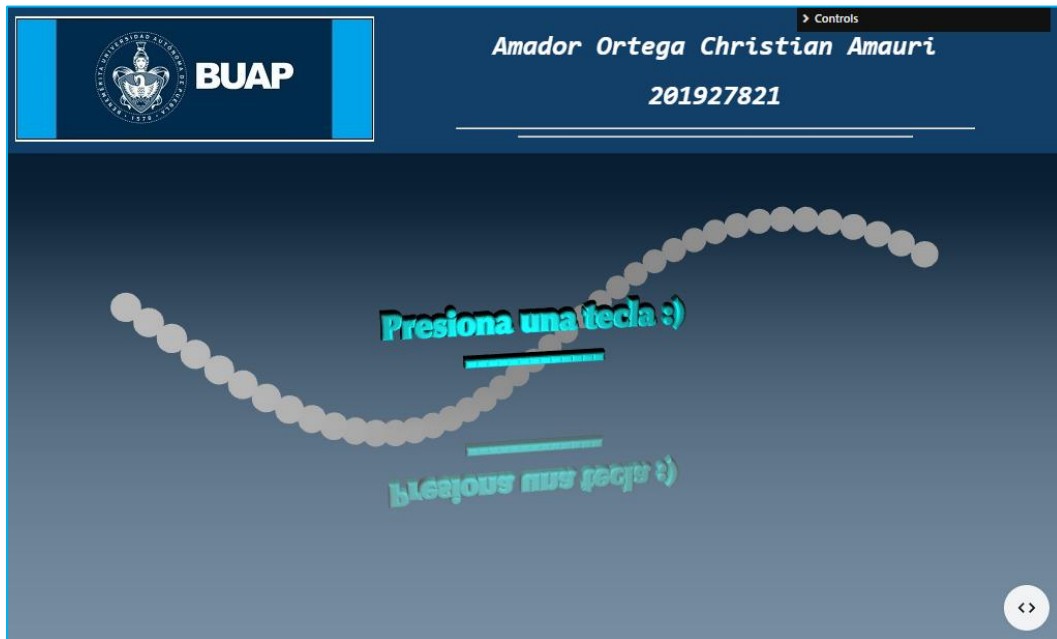
> Cambiar el color de las esferas:

Agregamos un botón que active esta acción mediante la creación de un GUI, en la única función que agregamos en él, hacemos rotar una variable mediante un incremento en 1, y una condición de igualar a 1 si esta llega a 12, por cada incremento, la variable que define el color de las esferas es cambiada por una más oscura. Así:

```
181 = const params = {
182 =     f1: function() {
183         bucle ++;
184         if(bucle == 1) clr = 0xffffffff;
185         if(bucle == 2) clr = 0xc7e8f7;
186         if(bucle == 3) clr = 0xaefff7;
187         if(bucle == 4) clr = 0x54f9ff;
188         if(bucle == 5) clr = 0x00f4e7;
189         if(bucle == 6) clr = 0x20dad9;
190         if(bucle == 7) clr = 0x17a9aa;
191         if(bucle == 8) clr = 0x127377;
192         if(bucle == 9) clr = 0x124650;
193         if(bucle == 10) clr = 0x122c32;
194         if(bucle == 11) clr = 0x000000;
195         if(bucle == 12) {bucle =1;  clr = 0xffffffff; }
196         refresh_animation();
197         jth++;
198     }
199 };
200
201
202 const gui = new GUI();
203 gui.add(params, 'f1').name('Cambiar color');
204 gui.open();
205
```

Resultado

Presentación de la aplicación:



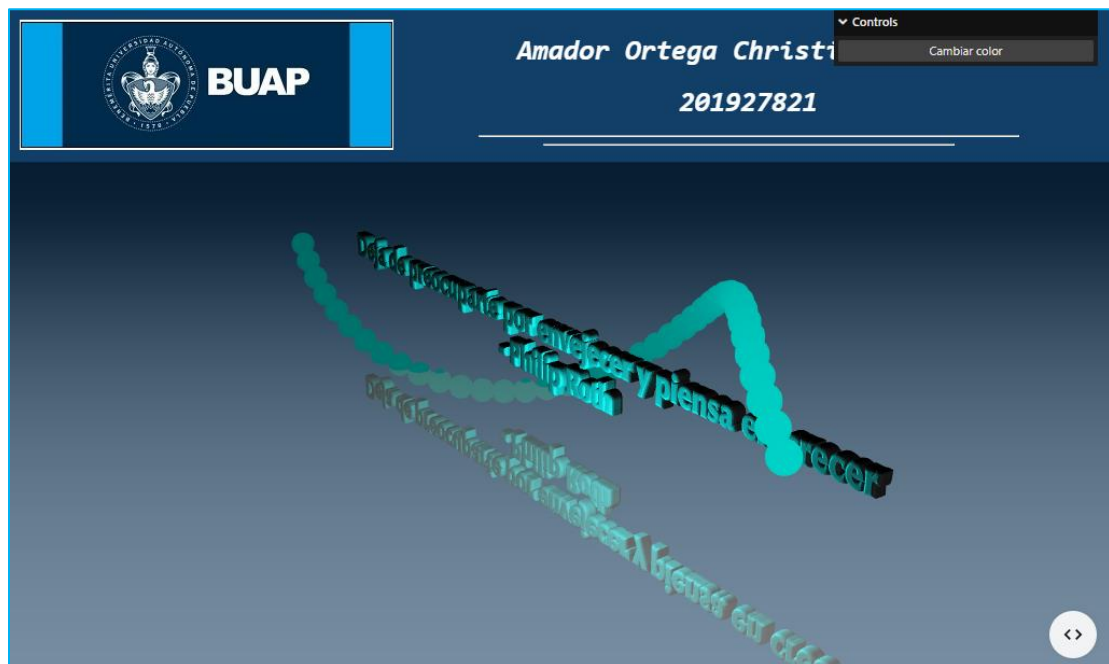
Si presionamos una tecla:



Podemos cambiar el tono de color de la animación de las esferas con el botón “cambiar color”:



Al rotar el texto, rotan las esferas alrededor de él:



Conclusiones

Como primer introducción a los sistemas multimedia, este proyecto es adecuado debido a que aunque conocer sobre programación te puede ayudar muchísimo, tampoco se requiere de gran cosa para acomodar bloques a nuestra conveniencia y construir las cosas que deseemos.

Aunque tampoco es demasiado fácil, hay que tener cuidado y prestar mucha atención a lo que estamos haciendo, simplemente debemos aprovechar el hecho de que ya han sido creadas muchas librerías y herramientas muy robustas y potentes precisamente para facilitar este desarrollo e incentivarlo.

Bibliografía

- [1] <https://developer.mozilla.org/en-US/docs/Web/API/SpeechSynthesis>
- [2] <https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene>
- [3] https://www.w3schools.com/js/js_random.asp
- [4] <https://threejs.org>