Shaders y THREE.js

<a.soroa@ehu.eus>

EHU

Shaders y THREE.js

- `THREE.js` viene con una serie de *shader*s incorporados.
- Asociado a los materiales.
- Normalmente, cada tipo de material define sus propios shaders:
  - `THREE.MeshLambertMaterial`, `THREE.MeshPhongMaterial`,...
- Pero además, `THREE.js` da la opción de crear nuevos materiales con nuestros *shaders*: `THREE.ShaderMaterial`

THREE.ShaderMaterial

- Tipo de material asociado a un par de shaders *vertex*/*fragment*.
- Ejemplo:

```
var material = new THREE.ShaderMaterial( {
  uniforms: {
    time: { type: 'f', value: 1.0 },
    matColor: { type: 'c', value: new THREE.Color(0x808000) }
  },
  attributes: {
    displacement: { type: 'f', value: [1.0, 2.0, -1.0, ...] }
  },
  vertexShader: "...",
  fragmentShader: "..."
} );
```

Atributos de los vértices

- Al definir una geometría `THREE.js` (`THREE.CubeGeometry`, `THREE.SphereGeometry`, ...) se definen una serie de atributos implícitos:

Atributo	Tipo	Descripción
`positon`	`vec3`	Posición vértice
`normal`	`vec3`	Normal vértice
`uv`	`vec2`	Coord. texturas
`uv2`	`vec2`	Coord. texturas auxiliar

- El *vertex shader* puede utilizar directamente estas variables.

```
void main() {
  vec3 newpos = position + vec3(1.0, 1.0, 0.0);
  ...
}
```

Variables uniform

- `THREE.js` envía una serie de variables `uniform` al shader.

Variable	Tipo	Descripción
`modelMatrix`	`mat4`	Matriz de local al mundo
`modelViewMatrix`	`mat4`	Matriz de local a la cámara
`projectionMatrix`	`mat4`	Matriz de proyección
`normalMatrix`	`mat3`	Inversa traspuesta de `modelViewMatrix`
`cameraPosition`	`vec3`	Posición cámara en el mundo

- Los *shaders* puede utilizar directamente estas variables.

```
void main() {
  gl_Position = projectionMatrix * modelViewMatrix * vec4(position, 1.0);
}
```

Nuevos atributos/uniform

- Podemos definir nuevos atributos/variables uniform.
  - Se añaden a los atributos/uniform por defecto.

```
var material = new THREE.ShaderMaterial( {
  uniforms: {... }, // nuevos uniform
  attributes: { ...}, // nuevos atributos
  ...
} );
```

- Sintaxis:

```
nombre : { type: 'tipo', value: valor }
```

Donde:

- `nombre`: el nombre del atributo/uniform.
- `type`: el tipo del atributo/uniform.
- `value`: el valor inicial del atributo/uniform.

Tipo

- Un código para determinar el tipo del atributo/uniform.
- Valores posibles:

Atributo	Tipo shader	Tipo `THREE.js`
'i', '1i'	`int`	`Number`
'f', '1f	`float`	`Number`
'v2'	`vec2`	`THREE.Vector2`
'v3'	`vec3`	`THREE.Vector3`
'c'	`vec3`	`THREE.Color`
'v4'	`vec4`	`THREE.Vector4`
'm3'	`mat3`	`THREE.Matrix3`
'm4'	`mat4`	`THREE.Matrix4`
't'	`sampler2D`	`THREE.Texture`
't'	`samplerCube`	`THREE.CubeTexture`

Ejemplo

```
var material = new THREE.ShaderMaterial( {
  uniforms: {
    time: { type: 'f', value: 1.0 },
    matColor: { type: 'c', value: new THREE.Color(0x808000) }
  },
  attributes: {
    displacement: { type: 'f', value: [1.0, 2.0, -1.0, ...] }
  },
  vertexShader: "...",
  fragmentShader: "..."
} );
```

Nuevos atributos

- Nuevos atributos con información adicional de los vértices. Por ejemplo:

El programa:

```
var geometry = new THREE.CubeGeometry( 1 );
var material = new THREE.ShaderMaterial( {
  attributes: {
    displacement: {   type: 'f', value: [1.0, 2.0, -1.0, ...] }
  }, ...});
var obj = new THREE.Mesh(geometry, material);
scene.add(obj);
```

El *shader*:

```
  attribute float displacement;
  void main() {
    vec3 pos = position + vec3(displacement);
    ...
  }
}
```

Nuevos uniform

- Misma sintaxis que los atributos:

El programa:

```
  var geometry = new THREE.CubeGeometry( 1 );
  var material = new THREE.ShaderMaterial( {
    uniforms: {
      matColor: { type: 'c', value: new THREE.Color(0x808000) }
    }, ...});
  var obj = new THREE.Mesh(geometry, material);
  scene.add(obj);
```

El *shader*:

```
  uniform vec3 matColor;
  varying vec3 vertexColor;
  void main() {
    vertexColor = matColor;
    ...
  }
}
```

Nuevos uniform

- Normalmente cambiaremos el valor de los uniform en cada renderizado:

```
var material; // El material es una variable global

function createScene() {
  var geometry = new THREE.CubeGeometry( 1 );
  material = new THREE.ShaderMaterial( {
    uniforms: {
      matColor: { type: 'c', value: new THREE.Color(0x808000) },
      factor: { type: 'f', value: 1 }
    }, ...});
  var obj = new THREE.Mesh(geometry, material);
  scene.add(obj);
 }

function render() {
  material.uniforms.factor.value = 1.5; // asignar nuevo valor a 'factor'
  var myColor = new THREE.Color(...); // un color diferente
  material.uniforms.matColor.value.copy(myColor); // asignar 'matColor'
}
```

Fichero .js y .html

- `prog.js`: El JavaScript con la aplicación.
- `prog.html`: La hoja html que hace lo siguiente:
  - Carga los scripts y las dependencias.
  - Crea un *canvas* para renderizar el resultado de la aplicación.
  - Ejecuta la función principal.

Sin shaders

```
<!doctype html>
<html lang="en">
<head>
  ... <!-- metainformation -->
</head>
<div id="container">
</div>
<body>
  <script src="lib/jquery-1.11.3.min.js"></script>
  ... <!-- carga mas librerias -->
  <script src="proj.js"></script> <!-- script principal -->
</body>
<br>
</html>
```

```
  // RENDERER
  renderer = new THREE.WebGLRenderer( { antialias: true } );
  renderer.setSize( canvasWidth, canvasHeight );
  renderer.setClearColor( 0xAAAAAA, 1.0 );
  // Add to DOM
  var container = document.getElementById('container');
  container.appendChild( renderer.domElement );
```

Con shaders

- Escribimos el shader en la página HTML.
  - Conviene editar el programa en un editor y después copiarlo al fichero HTML.
- Desde el script javascript, cargamos el shader.

Fichero html

```
<!doctype html>
<body>
   ... <!-- carga librerias -->

  <!-- el vertex shader -->
  <script type="text/x-glsl" id="vertexShader">
varying vec3 vColor;
void main() {
  vColor = vec3(1.0, 1.0, 1.0);
  gl_Position = projectionMatrix * modelViewMatrix * vec4( position, 1.0 );
}
  </script>
  <!-- el fragment shader -->
  <script type="text/x-glsl" id="fragmentShader">
varying vec3 vColor;
void main() {
  gl_FragColor = vec4(vColor, 1.0);
}
  </script>
  <script src="proj.js"></script> <!-- script principal -->
</body>
```

Fichero js

```
var material = new THREE.ShaderMaterial( {
      uniforms: {...},
      attributes: {...},
      vertexShader: document.getElementById( 'vertexShader' ).textContent,
      fragmentShader: document.getElementById( 'fragmentShader' ).textContent
  } );
```