|  |
| --- |
| Laboratoire 8 A: WebGL - Les couleurs |

Théorie

1. **WebGL** et les couleurs

Jusqu’à date, nous avons tout dessiné en blanc. Cette couleur blanche provient du ***fragment shader*** car c’est lui qui colore chacun des pixels des objets 3D. Ici, le ***fragment shader*** colore en blanc opaque.

**void main(void) {**

**gl\_FragColor = vec4(1.0, 1.0, 1,0, 1.0);**

**}**

Nous aimerions dessiner dans d’autres couleurs que la couleur blanche. Pour dessiner en couleurs, il faut faire deux choses.

Tout d’abord, il faut spécifier au *vertex shader* quelle est la couleur du vertex (**vertexColor)**.

**attribute vec3 vertexPos;**

**attribute vec4 vertexColor;**

**uniform mat4 modelViewMatrix;**

**uniform mat4 projectionMatrix;**

**varying vec4 vColor;**

**void main(void) {**

**gl\_Position = projectionMatrix \* modelViewVMatrix \* vec4(vertexPos, 1.0);**

**vColor = vertexColor;**

**}**

Ici, la couleur du vertex (**vertexColor**) va devoir être transférée au ***vertex shader*** par le programme **Javascript** de la même manière que la position du vertex a été transférée au ***vertex shader*** par le programme **Javascript.**

Observez que le ***vertex shader*** place la couleur du vertex dans la variable **vColor**. Cette variable est par la suite récupérée par le ***fragment shader*** qui colore le pixel.

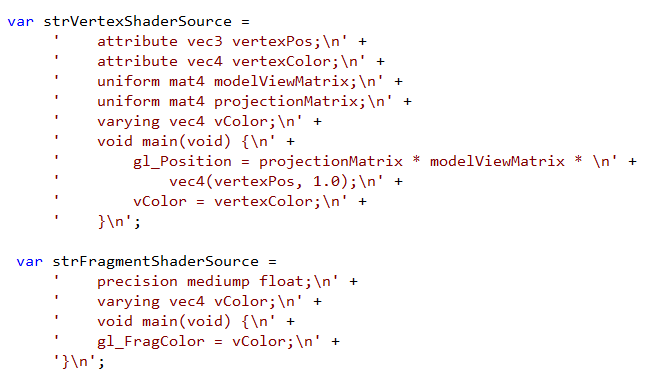
**precision medium float;**

**varying vec4 vColor;**

**void main(void) {**

**gl\_FragColor = vColor;**

**}**

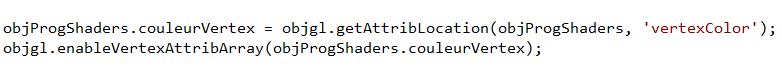


Pour les ***shaders***, le mot ***varying*** est très important. Cela signifie que la couleur du pixel va être interpolée par le ***fragment shader*** à partir des couleurs des vertex.

Le fichier **ShaderCouleurs.js** contient les modifications que l’on vient d’effectuer sur les ***shaders***.

Les codes sources des ***shaders*** ont été modifiés.

Dans l’initialisation des ***shaders***, nous avons ajouté la possibilité d’avoir accès à la variable **vertexColor** qui a été déclarée dans le ***vertex shader*** de la même manière que nous l’avons fait pour la position des vertex.

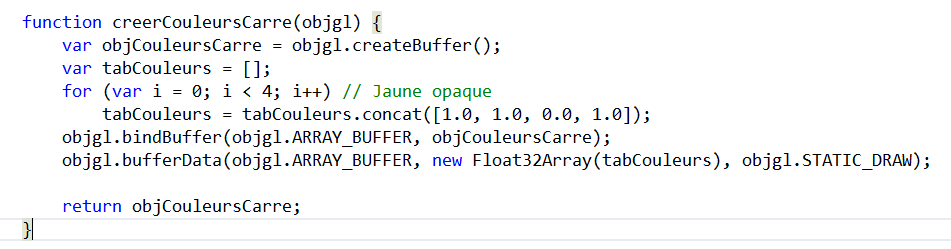


Ici, pour avoir accès à cette variable, nous allons utiliser **objProgShaders.couleurVertex**.

2. La coloration des objets

Ouvrez la page Web **8-A-1 Web GL Dessiner carré monochrome.html**. Cette page Web affiche un carré dans une seule couleur (en jaune opaque).

La fonction qui crée la couleur pour le carré se nomme **creerCouleursCarre**.



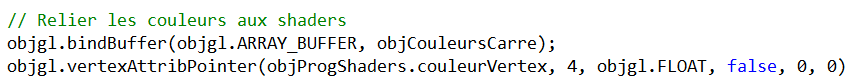
La fonction qui crée les couleurs pour le carré a été programmée exactement de la même manière que la fonction qui a créé les vertex du carré. Ici, ce qu’on place dans le tampon, ce ne sont pas des positions mais des couleurs.

Il est très important qu’il y ait une correspondance une à une entre un vertex et sa couleur. Le carré a 4 vertex. Par conséquent, le carré doit avoir 4 couleurs. Ici, les vertex ont tous la même couleur (le jaune opaque). C’est pour cette raison que le carré possède une couleur uniforme.

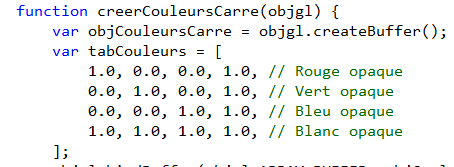


Dans la méthode **dessiner,** nous passons, en paramètre, le tampon de couleurs.

Puis, avant de dessiner, nous relions ce tampon de couleurs aux ***shaders*** comme nous l’avons fait pour les vertex. Ici, 4 est le nombre d’éléments qu’il y a à l’intérieur d’une couleur.

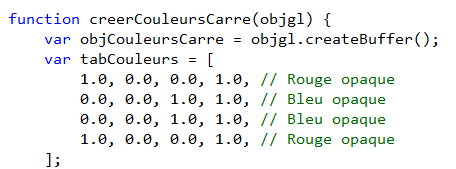


Ouvrez la page Web **8-A-2 Web GL Dessiner carré polychrome.html**. Cette fois-ci, il y a un dégradé de couleurs entre chaque vertex.

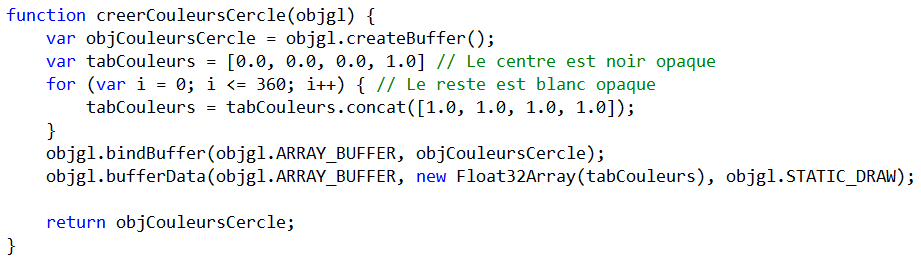


La raison est bien simple. C’est que chaque vertex possède sa propre couleur.

***Le fragment shader*** a interpolé la couleur des pixels qui sont situés entre les vertex (donc sur la droite qui relie les vertex).

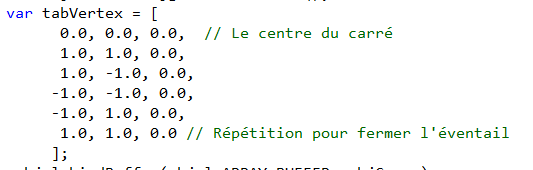
Ouvrez la page Web **8-A-3 Web GL Dessiner carré plein dégradé linéaire.html**. Le carré plein a un dégradé linéaire en diagonale. Cela s’explique par le fait que les vertex opposés ont la même couleur.

Ouvrez la page Web **8-A-4 Web GL Dessiner cercle plein dégradé radial.html**. Le cercle plein a un dégradé radial. Cela s’explique par le fait que le centre du cercle est noir et que les points sur la circonférence du cercle sont blanches.



Cela est rendu possible à cause qu’on utilise des triangles en éventail pour dessiner le cercle plein.

Nous obtenons également un dégradé radial dans le carré dessiné par **8-A-5 Web GL Dessiner carré plein dégradé radial.html.**

Pour pouvoir utiliser des triangles en éventail, nous avons dû créer 6 vertex. Le premier vertex pour le centre du carré et le sixième vertex ferme l’éventail.