Les informations sur tous les sommets d’un objet sont contenues dans le vertex Buffer

Chaque sommet d’un objet doit contenir la même quantité d’information et dans le même ordre.

Dans le code il est possible de définir 4 informations différentes :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Information : | Position | Couleur | textureMapping | Normales |
| Nb de valeurs possibles | 2/3 | 0/3/4 | 0/2 | 0/3 |
| Explication : | Définit la position dans l’espace du point. En 2D la troisième valeur est 0.  C’est la seule information obligatoire. | Permet de définir une couleur différente pour chaque sommet. | Coordonnées 2D de l’image qui sera mappé sur ce points.  -> Pour afficher une texture | Définit la normale a chaque sommet  -> Pour le smoothShading |
| Valeurs possibles : | Flottant [-∞ ; +∞] | Flottant [-1, 1] | Flottant [-1, 1] | Flottant [-1, 1] |
| Eléments compatible : | Tous (Point, Ligne, Face, Texte) | Tous(Point, Ligne, Face, Texte) | Face, Texte | ElementFace en 3D |

La coloration d’un objet peut varier en fonction de la valeur de typeColoration :

* ‘U’ : couleur unie sur tout l’objet (par défaut).
  + Passage des données : un vec4 par uniform
* ‘C’ : Chaque points a une couleur.
  + Passage des données : lu dans le vertex Buffer
  + Nécessite : Avoir définir une couleur par sommet avec **AddColor()**
* ‘T’ : L’objet est entouré par une texture.
  + Passage des données : Mapping lu dans VertexBuffer et image en uniform
  + Nécessite : Avoir définit le mapping avec **AddMapping()**. Avoir généré la texture a partir d’un fichier image avec **useTexture()**

La luminosité peut être calculer pour les éléments Face. Cela revient a multiplié la couleur qui devrait être afficher par une variable comprise entre 0 et 1. C’est-à-dire que l’on va assombrir certaines zones de l’objet (donc jamais l’éclairer !).

Les valeurs possibles pour typeShading sont

* ‘S’ : pas de Shading
* ‘D’ : Dur, La lumosité est la même pour la facette entière car le calcul est basé sur la normale a la surface.
  + Nécessite : être un élément Face, calcul automatique des normales dans le geometry shader
* ‘L’ : Lisse, la luminosité est calculé a partir des normales en tout point.
  + Nécessite : être un élément face, avoir ajouter des normales avec **AddNormales()** OU avoir générer les normales avec **generateNormales()** (moyenne des normales de chaque face pondérées par l’angle).

**Valeurs possibles pour typeColoration**

Une image contenant capture d’écran, texte, conception

Description générée automatiquement

**Valeurs possibles pour typeShading**

Une image contenant capture d’écran, texte, cercle, nombre

Description générée automatiquement

**Tous les cas et les aspects :**

|  |  |
| --- | --- |
| U + S |  |
| U + D |  |
| U + L |  |
| C + S |  |
| C + D |  |
| C + L |  |
| T + S |  |
| T + D |  |
| T + L |  |