

Construction d'un triangle 3D avec PureBasic

par Comtois

Date de publication : 07/03/08

Dernière mise à jour :

Ce tutoriel est une introduction à la 3D avec PureBasic.

- I - Avant-propos
- II - Introduction
- III - Mesh
- IV - Texture
- V - Matière
- VI - Entity
- VII - Eclairage
- VIII - Caméra
- IX - Affichage de la scène 3D
- X - Sources
- XI - Remerciements

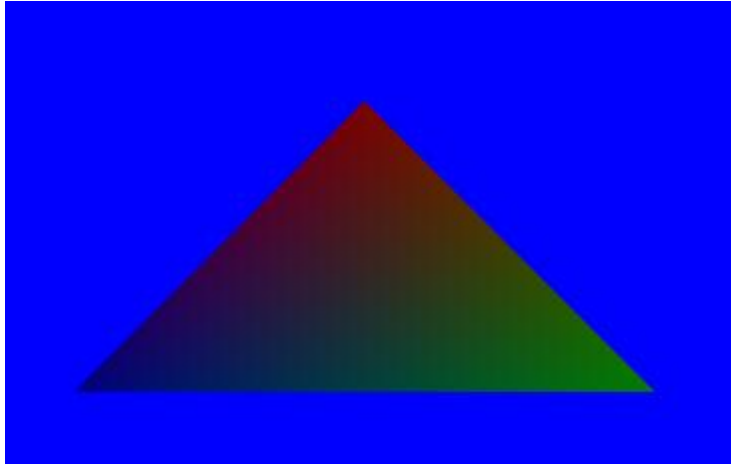
I - Avant-propos

Pour faire de la programmation 3D, PureBasic dispose d'un accès simplifié au moteur 3D open source très performant

 **Ogre.**

II - Introduction

Pour débiter nous allons créer un triangle et l'afficher à l'écran.



Copie d'écran de ce fameux triangle

Ce premier tutoriel sur la 3D nous permettra d'aborder :

- La création d'un mesh
- La création d'une texture
- La création d'une matière
- La création d'une entity
- L'éclairage ambiant
- La création d'une caméra
- L'affichage d'une scène 3D

Et le tout formera votre [Source](#) **premier programme 3D** avec PureBasic.

III - Mesh

Ce que dit la documentation PureBasic :

Les Meshs sont des objets 3D composés de triangles qui sont reliés entre eux pour donner la forme finale. Une Mesh peut posséder un squelette si elle est animée, permettant des animations réalistes et de qualité. Une Mesh ne peut pas être affichée directement dans le monde 3D, elle doit être encapsulée dans une entity.

Un triangle est défini par 3 sommets (en anglais : Vertex au singulier, vertices au pluriel).

Un sommet est défini par :

- Une position en x, y et z (#PB_Mesh_Vertex)
- Une normale (#PB_Mesh_Normal)
- Une couleur (#PB_Mesh_Color)
- Des coordonnées u et v d'une texture. (#PB_Mesh_UVCoordinate)

Seule la position du sommet est obligatoire. La normale, la couleur et les coordonnées UV sont facultatives. Dans ce tutoriel nous allons créer un mesh en définissant la position et la couleur des sommets. Les coordonnées de la position se donnent par rapport aux repères du mesh.

```
;Création d'un mesh
#Mesh = 0
CreateMesh(#Mesh, 200) ; 200 indique la dimension maxi du mesh


;Définition des sommets
SetMeshData(#Mesh, #PB_Mesh_Vertex | #PB_Mesh_Color , ?SommetsTriangles, 3) ; Indiquez ici le
nombre de sommets

;Définition des triangles
SetMeshData(#Mesh, #PB_Mesh_Face, ?IndexTriangles, 1) ; indiquez ici le nombre de triangles
```

?SommetsTriangles : Indique l'adresse à laquelle se trouvent les données correspondant à la définition des sommets. Dans le premier exemple on utilise des datas, mais il est aussi possible d'utiliser un tableau , ou d'allouer un bloc mémoire, voir les exemples ci dessous.

?IndexTriangles : Indique l'adresse à laquelle se trouvent les données correspondant à la définition du ou des triangles, dans notre exemple il n'y a qu'un triangle.

```
IndexTriangles:
Data.w 2,1,0
```

 **Notez le .w pour les datas, vous devez utiliser des word**

Pour former un triangle il faut indiquer l'index des sommets dans un ordre précis, dans le sens inverse des aiguilles d'une montre pour que la face soit visible, l'autre face du triangle sera invisible. Essayez de croiser 0 et 2 :

```
IndexTriangles:  
Data.w 0,1,2
```

En changeant l'ordre des sommets, le triangle n'est plus affiché. Faites l'essai !

Pour le voir à nouveau il faudrait placer la caméra derrière.

```
CameraLocate(#Camera, 0, 0, -500) ; Positionne la caméra
```

Si on voulait effectuer des rotations à notre triangle il faudrait donc doubler les faces, pour que notre triangle soit toujours visible quelque soit la position de la caméra.

```
IndexTriangles:  
Data.w 2,1,0  
Data.w 0,1,2
```

Et bien sûr, n'oubliez pas de changer le nombre de faces :

```
;Définition des triangles  
SetMeshData(#Mesh, #PB_Mesh_Face, ?IndexTriangles, 2) ; indiquez ici le nombre de triangles
```

IV - Texture

Ce que dit la documentation PureBasic :

Les textures permettent aux objets 3D d'avoir un aspect réaliste. En effet, sans texture les objets 3D seraient d'une seule couleur unie. PureBasic offre la possibilité de créer des textures directement à l'aide des outils 2D de base (bibliothèque 2DDrawing) ou de les charger à partir de fichiers.

Pour charger une texture utilisez la commande **LoadTexture()**.

Dans ce tutoriel nous n'utiliserons pas de média, nous devons donc créer notre texture.

```
;Création d'une texture
#Texture = 0
CreateTexture(#Texture, 64, 64)

;Remplissage de la texture en blanc pour visualiser les couleurs des sommets
StartDrawing(TextureOutput(#Texture))
  Box(0,0, TextureWidth(#Texture), TextureHeight(#Texture), RGB(255, 255, 255))
StopDrawing()
```

C'est un simple carré blanc. Amusez vous à changer la couleur de la texture et observez les changements à l'affichage.


V - Matière

Ce que dit la documentation PureBasic :

Les matières sont composées de une ou plusieurs textures et parfois de couleurs. Elles sont utilisées par les autres objets 3D tels que les 'Entity', 'Billboards' et les 'Particules' pour leur donner une matière (principalement une texture). Chaque matière regroupe un grand nombre de propriétés tel que l'éclairage, la couleur spéculaire et de réfraction, etc. pour permettre à une matière de ressembler à des revêtements complexes comme le verre, l'eau, le bois...

```
;Création d'une matière
#Matiere = 0
CreateMaterial(#Matiere, TextureID(#Texture))
MaterialAmbientColor(#Matiere, #PB_Material_AmbientColors)
```

Pour voir la couleur des sommets définis précédemment, nous devons utiliser la commande **MaterialAmbientColor()** avec la constante #PB_Material_AmbientColors. Vous pouvez essayer de mettre en commentaire cette commande et observez ce qui se passe. Il est aussi possible de donner une couleur à la matière (par exemple **MaterialAmbientColor(#Matiere, \$FF00FF)**), dans ce cas les couleurs des sommets ne seront plus utilisées.

La gestion des matières est un vaste sujet, surtout depuis que la V4 permet d'utiliser les scripts. On y reviendra peut-être dans un prochain tutoriel. Si vous êtes impatients vous pouvez toujours jeter un oeil sur la  **doc d'ogre**. Pour charger un fichier script reportez vous à la commande **Parse3DScripts()**.

VI - Entity

Ce que dit la documentation PureBasic :

Les 'Entity' sont des objets composés d'un mesh et d'une matière pouvant être déplacés et transformés en temps réel. Il est possible de partager la même Mesh entre plusieurs Entities tout en utilisant une matière différente pour chaque Entity, réduisant ainsi la consommation mémoire et l'utilisation du processeur.

```
;Création entity
#Entity = 0
CreateEntity(#Entity, MeshID(#Mesh), MaterialID(#Matiere))
```

VII - Eclairage

Si vous trouvez la scène 3D trop sombre, vous pouvez changer la couleur ambiante avec la commande :

```
AmbientColor(Rgb(255, 255, 255))
```

On verra dans un prochain tutoriel l'utilisation des lumières, mais sachez dès à présent qu'il faudra réduire la lumière ambiante pour que l'effet des lumières ajoutées soit perceptible. Par exemple une lumière grise foncée.

```
AmbientColor(Rgb(85, 85, 85))
```

VIII - Caméra

Ce que dit la documentation PureBasic :

Les caméras sont utilisées pour se déplacer dans le monde en 3D. Il est possible de les utiliser comme des caméras réelles avec lesquelles on peut se déplacer, effectuer des rotations, changer l'angle de vision etc. Au moins une caméra est nécessaire pour effectuer un rendu du monde 3D sur un écran, mais plusieurs caméras peuvent être utilisées en même temps pour afficher le monde sous des angles de vue différents (rétroviseurs , split-screen...).

Pour ce tutoriel on se contente d'une seule caméra

```
;Création d'une caméra, c'est indispensable pour voir quelque chose
#Camera = 0
CreateCamera(#Camera, 25, 25, 50, 50) ; Création d'une caméra
CameraBackColor(#Camera, $FF0000) ; Couleur de fond bleue
CameraLocate(#Camera,0,0,-500) ; Positionne la caméra
CameraLookAt(#Camera, EntityX(#Entity), EntityY(#Entity), EntityZ(#Entity)) ; Oriente la caméra
vers l'entity
```

Pour la démonstration, notre caméra occupe la moitié de l'écran aussi bien en largeur qu'en hauteur et pour la centrer, on la place à 25% à gauche , et 25 % en haut.

Pour une caméra qui occupe tout l'écran recopiez cette ligne :

```
CreateCamera(#Camera, 0, 0, 100, 100) ; Création d'une caméra
```

IX - Affichage de la scène 3D

Pour afficher la scène 3D utilisez la commande **RenderWorld()**.

Il est possible de mélanger 3D et 2D :

- Pour que la 3D s'affiche par dessus la 2D utilisez la commande **CameraBackColor(#Camera, -1)**. Cette commande permet de changer la couleur de fond de la scène 3d, ou de rendre le fond transparent si le paramètre vaut -1.
- Pour que la 2D s'affiche par dessus la 3D placez vos commandes d'affichage 2D après la commande **RenderWorld()**

```
RenderWorld()  
  
; Affiche la 2D par dessus la 3D  
DisplaySprite(#Sprite, x, y)  
  
StartDrawing(ScreenOutput())  
DrawText(0, 0, "Un texte à l'écran")  
StopDrawing()  
  
FlipBuffer()
```

X - Sources

Source [Construction d'un triangle 3D en utilisant les datas](#)

Source [Construction d'un triangle 3D en utilisant des tableaux](#)

XI - Remerciements

Je tiens à remercier tout particulièrement **Dut** pour la relecture de l'article.

