Déplacement 3D d'une entité avec PureBasic

par Comtois

Date de publication : 18/03/08

Dernière mise à jour :

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

Ce document est issu de http://www.developpez.com et reste la propriété exclusive de son auteur. La copie, modification et/ou distribution par quelque moyen que ce soit est soumise à l'obtention préalable de l'autorisation de l'auteur.

I - Introduction

Dans ce tutoriel nous aborderons :

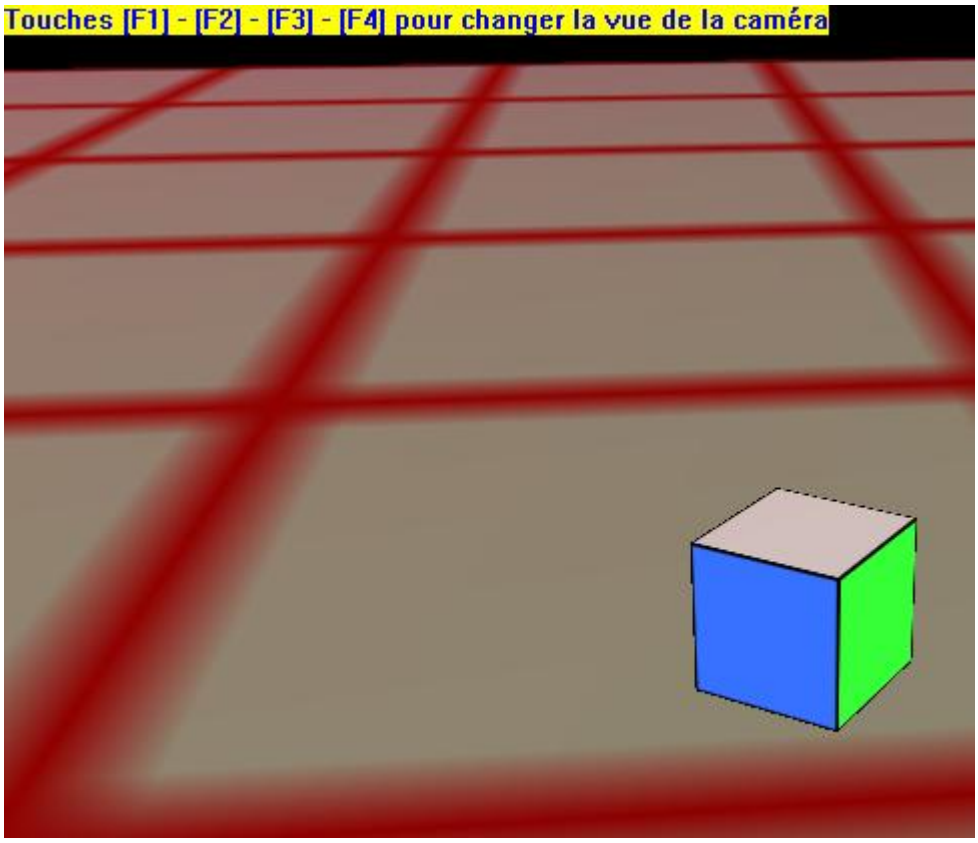
- Les lumières
- Le déplacement d'une entité avec le curseur
- Le suivi de la caméra

Reportez vous également aux articles ***construction d'un triangle 3D*** *et* ***construction d'un cube 3D****.*

Ce document est issu de http://www.developpez.com et reste la propriété exclusive de son auteur. La copie, modification et/ou distribution par quelque moyen que ce soit est soumise à l'obtention préalable de l'autorisation de l'auteur.

II - Mesh

En guise de démonstration, nous allons déplacer un cube sur un sol. Nous avons donc deux meshes à créer.



II-1 - Sol

Pour le sol j'avais prévu de créer un plan composé de deux triangles, finalement pour montrer qu'un mesh peut servir à la création de plusieurs entitys, le sol utilisera le même mesh que le cube.

Pour vous exercer, vous pouvez essayer de créer un mesh plan composé de deux triangles. Prenez modèle sur le mesh cube, chaque face du cube est en fait un plan composé de deux triangles.

II-2 - Entité en mouvement

Pour éviter de faire appel à des médias, tout sera construit dans le code. Nous allons utiliser le cube vu dans le tutoriel précédent.

Vous pouvez remplacer le cube par le robot fourni avec PureBasic (celui des démos d'Ogre).

Ce document est issu de http://www.developpez.com et reste la propriété exclusive de son auteur. La copie, modification et/ou distribution par quelque moyen que ce soit est soumise à l'obtention préalable de l'autorisation de l'auteur.

III - Macros

Les macros **NEW_X()** et **NEW_Z()** permettent de calculer la nouvelle position d'un point 3D en connaissant son origine, son angle et la distance de déplacement. Ces macros sont valables sur un terrain plat ou avec une pente légère. Sur un terrain accidenté, il faudrait tenir compte de la pente pour calculer la distance de déplacement.

```
#Deg2Rad = #Pi/180.0
Macro NEW_X(x, Angle, Distance)
  ((x) + Cos((Angle) * #Deg2Rad) * (Distance))
EndMacro

Macro NEW_Z(z, Angle, Distance)
  ((z) - Sin((Angle) * #Deg2Rad) * (Distance))
EndMacro
```

Sin() *et* ***Cos()*** *attendent un angle en radian , comme on utilise des degrés, la multiplication par* ***#Deg2Rad*** *permet de faire la conversion de degrés en radians.*

Cette macro permet d'afficher un peu d'aide à l'écran. C'est aussi pour montrer que l'on peut mélanger 3D et 2D. Placer tout ce qui concerne la 2D après la commande **RenderWorld()**. Il est aussi possible de rendre le fond 3D transparent avec la commande **CameraBackColor(#Camera, -1)**, ce qui permet d'afficher la 3D par dessus la 2D.

```
Macro AFFICHE_AIDE()
  StartDrawing(ScreenOutput())
    DrawText(0,0,"Touches [F1] - [F2] - [F3] - [F4] pour changer la vue de la caméra", $FF0000,
 $00FFFF)
  StopDrawing()
EndMacro
```


Ce document est issu de http://www.developpez.com et reste la propriété exclusive de son auteur. La copie, modification et/ou distribution par quelque moyen que ce soit est soumise à l'obtention préalable de l'autorisation de l'auteur.

IV - Procédures

```
Procedure.f CurveValue(actuelle.f, Cible.f, P.f)
   ;Calcule une valeur progressive allant de la valeur actuelle à la valeur cible
   Define.f Delta
   Delta = Cible - actuelle
   If P > 1000.0 : P = 1000.0 : EndIf
   ProcedureReturn  (actuelle + ( Delta * P / 1000.0))
EndProcedure
```

Cette procédure permet de calculer une valeur progressive allant de la valeur actuelle à la valeur cible à la vitesse P.

Elle est souvent utilisée pour rendre les mouvements de caméra plus fluides.

Vous pouvez par exemple essayer de changer de mode de vue avec les touches de fonction [F1] à [F4]. La caméra se placera progressivement à sa nouvelle position.

Ce document est issu de http://www.developpez.com et reste la propriété exclusive de son auteur. La copie, modification et/ou distribution par quelque moyen que ce soit est soumise à l'obtention préalable de l'autorisation de l'auteur.

V - Lumières

```
;-Light
AmbientColor(RGB(55,55,55)) ; Réduit la lumière ambiante pour mieux voir les lumières
#LightRouge = 0 : CreateLight(#LightRouge,RGB(255, 0, 0),    0, 500,    0)
#LightBleue = 1 : CreateLight(#LightBleue,RGB(0, 0, 255),    0, 500, 1000)
#LightVerte = 2 : CreateLight(#LightVerte,RGB(0, 255, 0), 1000, 500, 1000)
```

Comme nous vous l'avions déjà dit précédemment, il faut réduire l'éclairage ambiant pour mieux visualiser les autres lumières.

Si une entity ne doit pas être affectée par les lumières, vous pouvez utiliser la commande **DisableMaterialLighting()**. Par exemple pour que le sol ne soit pas affecté par les lumières, sa matière sera écrite ainsi:

```
#MatiereSol = 1
CreateMaterial(#MatiereSol, TextureID(#TextureSol))
DisableMaterialLighting(#MatiereSol, 1)
```

Dans ce cas changez la couleur des faces du cube.

Modifiez la face du haut pour comparer le résultat.

```
;Dessus 0 à 3
Data.f -0.5,0.5,-0.5
Data.f 0,1,0
Data.l $FFFFFF
Data.f 0,0

Data.f 0.5,0.5,-0.5
Data.f 0,1,0
Data.l $FFFFFF
Data.f 0,1

Data.f 0.5,0.5,0.5
Data.f 0,1,0
Data.l $FFFFFF
Data.f 1,1

Data.f -0.5,0.5,0.5
Data.f 0,1,0
Data.l $FFFFFF
Data.f 1,0
```


Ce document est issu de http://www.developpez.com et reste la propriété exclusive de son auteur. La copie, modification et/ou distribution par quelque moyen que ce soit est soumise à l'obtention préalable de l'autorisation de l'auteur.

VI - Déplacement de l'entité

```
      If KeyboardPushed(#PB_Key_Left)
        Angle + 1
        RotateEntity(#Entity, Angle, 0, 0)
      ElseIf KeyboardPushed(#PB_Key_Right)
        Angle - 1
        RotateEntity(#Entity, Angle, 0, 0)
      EndIf

      If KeyboardPushed(#PB_Key_Up)
        MoveEntity(#Entity, NEW_X(0, Angle , Vitesse), 0, NEW_Z(0, Angle, Vitesse))
      ElseIf KeyboardPushed(#PB_Key_Down)
        MoveEntity(#Entity, NEW_X(0, Angle , -Vitesse), 0, NEW_Z(0, Angle, -Vitesse))
      EndIf
```

Les touches gauche et droite du curseur permettent de faire tourner l'entity.

Depuis la V4 la commande RotateEntity() effectue une rotation absolue de l'entity en fonction des angles x,y et z indiqués. Dans les versions précédentes la rotation était relative aux angles indiqués , donc ne soyez pas surpris si vous étudiez des codes des versions antérieures à la V4.

Autre surprise, les axes sont inversés dans la commande RotateEntity(), en effet pour faire une rotation sur l'axe des y, il faut renseigner l'axe des x, comme dans le code ci-dessus.

C'est un bug connu de Fred (l'auteur de PureBasic), il devrait le corriger dans une prochaine version (normalement pour la 4.30). Donc là aussi vous êtes prévenus, attendez-vous à voir des codes différents avec les prochaines versions.

Les touches haut et bas du curseur permettent d'avancer ou reculer l'entity.

Cette fois ci j'utilise la commande MoveEntity() qui effectue un déplacement relatif par rapport à la position précédente de l'entity.

Il est parfaitement possible d'effectuer un déplacement absolu en utilisant la commande EntityLocate() :

```
  If KeyboardPushed(#PB_Key_Up)
    EntityLocate(#Entity, NEW_X(EntityX(#Entity), Angle , Vitesse), EntityY(#Entity),
   NEW_Z(EntityZ(#Entity), Angle, Vitesse))
  ElseIf KeyboardPushed(#PB_Key_Down)
    EntityLocate(#Entity, NEW_X(EntityX(#Entity), Angle , -Vitesse), EntityY(#Entity),
   NEW_Z(EntityZ(#Entity), Angle, -Vitesse))
  EndIf
```


Ce document est issu de http://www.developpez.com et reste la propriété exclusive de son auteur. La copie, modification et/ou distribution par quelque moyen que ce soit est soumise à l'obtention préalable de l'autorisation de l'auteur.

VII - Suivi de la caméra

```
Procedure GestionCamera(Mode.l)
  Define.f Px, Py, Pz, Pv
  Static AngleCamera.f

  Pv = 25

  Select Mode

    Case #VueDessus
      AngleCamera = CurveValue(AngleCamera, Angle + 180, Pv)
      Px = CurveValue(CameraX(#Camera), NEW_X(EntityX(#Entity), AngleCamera, 40), Pv)
      Py = CurveValue(CameraY(#Camera), EntityY(#Entity) + 140, Pv)
      Pz = CurveValue(CameraZ(#Camera), NEW_Z(EntityZ(#Entity), AngleCamera, 40), Pv)

    Case #VueArriere
      AngleCamera = CurveValue(AngleCamera, Angle + 180, Pv)
      Px = CurveValue(CameraX(#Camera), NEW_X(EntityX(#Entity), AngleCamera, 80), Pv)
      Py = CurveValue(CameraY(#Camera), EntityY(#Entity) + 40, Pv)
      Pz = CurveValue(CameraZ(#Camera), NEW_Z(EntityZ(#Entity), AngleCamera, 80), Pv)

    Case #VueCote
      AngleCamera = CurveValue(AngleCamera, Angle + 120, Pv)
      Px = CurveValue(CameraX(#Camera), NEW_X(EntityX(#Entity), AngleCamera, 80), Pv)
      Py = CurveValue(CameraY(#Camera), EntityY(#Entity) + 40, Pv)
      Pz = CurveValue(CameraZ(#Camera), NEW_Z(EntityZ(#Entity), AngleCamera, 80), Pv)

    Case #VueAvant
      AngleCamera = CurveValue(AngleCamera, Angle, Pv)
      Px = CurveValue(CameraX(#Camera), NEW_X(EntityX(#Entity), AngleCamera, 80), Pv)
      Py = CurveValue(CameraY(#Camera), EntityY(#Entity) + 40, Pv)
      Pz = CurveValue(CameraZ(#Camera), NEW_Z(EntityZ(#Entity), AngleCamera, 80), Pv)

   EndSelect

   CameraLocate(#Camera, Px, Py, Pz)
   CameraLookAt(#Camera, EntityX(#Entity), EntityY(#Entity), EntityZ(#Entity))
EndProcedure
```

En guise de démonstration cette procédure comporte 4 modes :

- Vue de dessus
- Vue à la troisième personne
- Vue de côté (Amusez vous à changer l'angle 120°)
- Vue avant

le principe est toujours le même, il consiste à calculer la position de la caméra dans Px,Py et Pz, et ensuite d'orienter la caméra vers l'entité souhaitée avec la commande **CameraLookAt()**.

CurveValue() permet de passer en douceur d'une position à l'autre.

Ce document est issu de http://www.developpez.com et reste la propriété exclusive de son auteur. La copie, modification et/ou distribution par quelque moyen que ce soit est soumise à l'obtention préalable de l'autorisation de l'auteur.

VIII - Source

Source **Déplacement d'une entité 3D**.

Ce document est issu de http://www.developpez.com et reste la propriété exclusive de son auteur. La copie, modification et/ou distribution par quelque moyen que ce soit est soumise à l'obtention préalable de l'autorisation de l'auteur.

IX - Remerciements

Ce document est issu de http://www.developpez.com et reste la propriété exclusive de son auteur. La copie, modification et/ou distribution par quelque moyen que ce soit est soumise à l'obtention préalable de l'autorisation de l'auteur.

Ce document est issu de http://www.developpez.com et reste la propriété exclusive de son auteur. La copie, modification et/ou distribution par quelque moyen que ce soit est soumise à l'obtention préalable de l'autorisation de l'auteur.