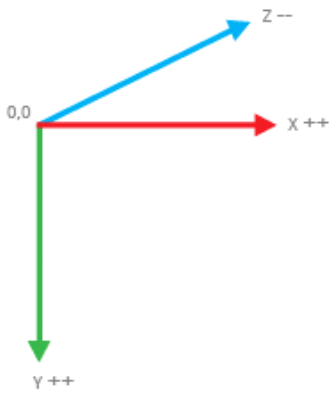
3D avancée

lundi 24 mars 2014 22:18

// Rappel

Nous avons vu précédemment que l'essentiel de la 3 Dimension repose sur l'ajout d'un axe Z. Ce dernier étant décroisant vers l'horizon et croisant vers nous



Par ailleurs nous avons vu comment, à l'aide de méthodes de transformations de matrices, nous pouvions déplacer l'origine de cet espace de coordonnées afin de placer/déplacer un objet dans l'espace.

Pour ce faire nous utilisons la méthodes simple de déplacement de matrices en ajoutant l'axe Z

```
pushMatrix();
translate(x, y, z);
popMatrix();
```

Par ailleurs nous pouvons ajouter à ces méthodes des méthodes de rotations spécifique à l'axe X, Y et Z au sein d'un déplacement de matrice

```
pushMatrix();
translate(x, y, z);
rotateX(radians);
    Ou
rotateY(radians);
    Ou
rotateZ(radians);
popMatrix();
```

Enfin nous avons vu différentes formes 3D primitive à savoir

```
box(width, height, depth);

sphere(rayon);
sphereDetail(res); //sphereDetails(resLongitude, resLatitude);

line(x0, y0, z0, x1, y1, z1);
point(x, y, z);
vertex(x, y, z);
text(string, x, y, z);
```

// Perspective et orthogonal

Une image 3D est une image calculé en sur 3dimension dont l'ensemble du dessin est projeté sur le plan 2D de l'ordinateur permettant ainsi de donner une notion de profondeur à notre image.

Il est possible de changer les paramètres de cette projection afin de changer le rendu de perspective de notre programme comme nous le ferions à travers une focal d'appareil photo.

Processing nous offre ainsi la possibilité de changer les paramètre de perspective à l'aide de la méthode suivante

perspective(fov, aspectRatio, zNear, zFar);
Ou de manière automatique
perspective();

ou :
Fov = l'angle vertical (focal) de notre perspective)
aspectRatio = le ratio de notre image (width/height en règle général)
zNear = le plan z le plus proche visible
zFar = le plan z le plus loin visible (horizon visible)

Il est également possible de travailler dans un espace orthogonal, c'est-à-dire privé de tout point de fuite et donc de perspective. Pour ce faire nous utilisons les méthodes suivantes :

ortho(left, right, bottom, top, near, far);
Ou de manière automatique
ortho();

où :
Left = plan gauche
Right = plan droite
Bottom = plan bas
Top = plan haut
Near = distance Z min
Far = distance Z max

// Camera

La particularité d'un espace 3D est l'ajout de la notion de point de vue. Il s'agit du point depuis lequel le programme voit le dessin. Par défaut celui-ci se trouve en width/2, height/2 et 0

Nous avons vu comment déplacer un objet à l'aide des déplacements de matrice afin de la rendre visible pour notre point de vue mais il est également possible de contrôler et donc déplacer ce point de vue.

Pour ce faire nous devons créer une caméra qui deviendra alors notre point de vue. Nous pourrons alors déplacer cette caméra à notre guise de sorte à nous déplacer dans l'environnement ou autour de nos objets.

Dans processing nous pouvons créer et gérer une camera à l'aide de la méthode

camera(eyeX, eyeY, eyeZ, lookAtX, lookAtY, lookAtZ, upX, upY, upZ);

Où
eyeX, eyeY et eyeZ = position x,y,z de l'œil de notre caméra
lookAtX, lookAtY, lookAtZ = la position x, y, z de direction vers laquelle nous regardons
upX, upY, upZ = défini l'axe tourné vers le haut (compris entre -1 et 1) - par défaut on utilise upX =0, upY = 1 et upZ = 0

// Lumières

Lorsque nous travaillons dans un espace 3D nous avons la possibilité de gérer les sources lumineuses ainsi que les paramètres de ces lumières.

L'utilisation de lumière permet ainsi de faire ressortir les volume par un jeu d'ombre et d'éclairage sur les surface. Dans processing, par défaut, notre espace ne possède aucune lumière mais il est possible d'en rajouter et de maitriser différents paramètre afin de contrôler l'ambiance lumineuse de notre scène. Pour ce faire nous pouvons utiliser diverse méthodes dont :

light(); défini une lumière globale par défaut
directionalLight(r,v,b, xAxis, yAxis, zAxis)
Où :
r,v,b (ou h,s,b) = couleur de la lumière
xAxis, yAxis et zAxis définissent l'orientation de la lumière comprise en -1 et 1
ambientLight(r,v,b, x, y, z);
Où :
r,v,b (ou h,s,b) = couleur de la lumière
x, y, z définissent la position de la lumière
pointLight(r,v,b, x,y,z);
Où :
r,v,b (ou h,s,b) = couleur de la lumière
x, y, z définissent la position de la lumière
spotLight(r,v,b, x,y,z, xAxis, yAxis, zAxis, angle, concentration);
Où :
r,v,b (ou h,s,b) = couleur de la lumière
x, y, z définissent la position de la lumière
xAxis, yAxis et zAxis définissent l'orientation de la lumière comprise en -1 et 1
Angle définie l'angle du cône de lumière
Concentration définie la longueur du cône
lightSpecular(r,v,b); //s'utilise en combinaison des méthodes de materiaux specular et shininess
Où :
r,v,b (ou h,s,b) = couleur de la lumière spéculaire

// Materials

Dans un espace 3D, chaque objet possède des propriétés de matériaux définissant la manière dont il vont réfléchir ou émettre une lumière. Ainsi nous pouvons définir pour nos forme 3d les paramètres suivant:

specular(r,v,b); Définie le niveau de couleur spéculaire du matériel
ambient(r,v,b); Définie la couleur ambiante de réflexion du matériel
emissive(r,v,b); Définie la couleur emissive du materiel
shininess(float); Définie le niveau de réflexion diffuse de 0 à infini (0 = 0 grande diffusion)

NB : les méthodes de matériaux s'emploient accompagné de la méthode lightSpecular et ambientLight permettant de définir la couleur de la lumière Speculaire et Lumière ambient

// Vertex 3D astuce "Les coordonnées sphérique"

Il est souvent utile de pouvoir positionner un point sur un sphère. Pour ce faire nous utilisons les bases de trigonométrique sphérique permettant de définir un point xyz sur un sphère de la manière suivante :

Soit phi un angle compris entre 0 et PI (180)
Soit theta un angle compris entre 0 et TWO_PI (360)
Soit radius le rayon

X = sin(alpha)*cos(theta)*radius;
Y = sin(alpha)*sin(theta)*radius;
Z = cos(alpha)*radius;