32-Vue caméra

> Remarque: certaines techniques décrites dans ce didacticiel sont obsolètes. Il est recommandé aux utilisateurs d'utiliser l'objet *jit.gl.camera* au lieu des messages *jit.gl.render* ***camera*** et ***lookat*** décrits ci-dessous.

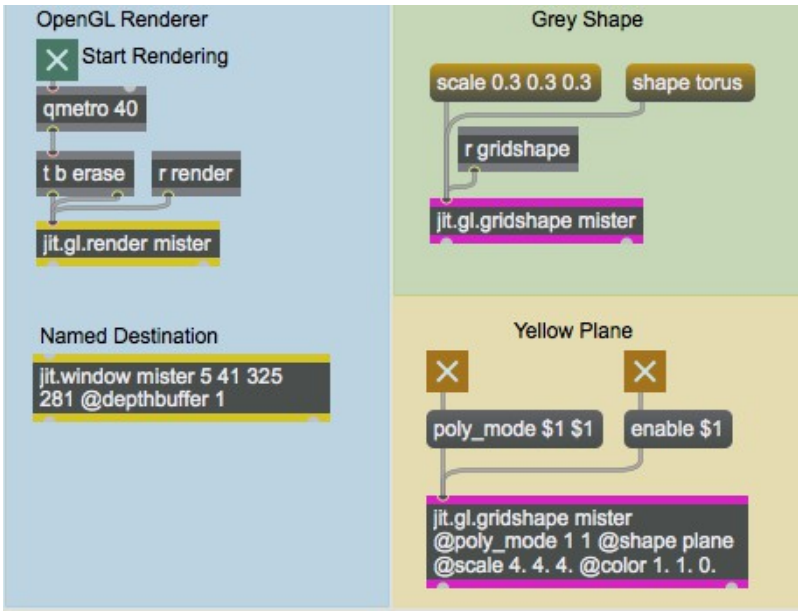
Ce didacticiel vous montre comment configurer la vue caméra et comment positionner et faire pivoter les objets GL à l'aide de *jit.gl.handle*. Il couvrira le groupe de composants qui, ensemble, constituent la ***vue caméra***: la position de la caméra, le point vers lequel la caméra regarde, le vecteur "up", le type de projection, l'angle de l'objectif et les plans de détourage.

Dans le coin inférieur gauche du patch, il y a un objet *jit.window* nommé **mister**. Cette fenêtre sera la destination de notre dessin OpenGL. Vous remarquerez que l'objet *jit.window* a un argument d'attribut **@depthbuffer 1** pour spécifier la création d'un tampon de profondeur. Un tampon de profondeur permet au moteur de rendu OpenGL de déterminer quelle commande de dessin est visible à un pixel donné en fonction de la proximité de la géométrie par rapport à la caméra. Sans tampon de profondeur, OpenGL utilise ce qui est souvent appelé "l'algorithme du peintre" – c'est-à-dire que les résultats visibles des commandes de dessin correspondent à la séquence dans laquelle elles sont exécutées.

Le contexte GL

• Cliquez sur l'objet *toggle* intitulé ***Start Rendering***.

Nous voyons maintenant un grand cercle gris et quelques lignes jaunes. Ceux-ci sont dessinés par deux instances de l'objet *jit.gl.gridshape*. L'objet *jit.gl.gridshape* peut dessiner une variété de formes 3D, y compris des sphères, des tores, des cylindres, des cubes, des plans et des cercles. Le cercle gris que nous voyons dessiné dans la fenêtre est en fait une sphère et est dessiné par la section du patch intitulée ***Gray Shape***. Les lignes jaunes sont en fait un plan et sont dessinées par la section du patch intitulée ***Yellow Plane***. Le plan jaune est rendu avec **poly_mode 1 1**, ce qui signifie que la forme est dessinée avec des polygones de contour plutôt que des polygones de remplissage pour les faces avant et arrière du plan. Si vous désactivez l'objet *toggle* connecté à la boîte de *message* **poly_mode $ 1 $ 1**, vous pouvez voir le plan rendu avec des polygones remplis.

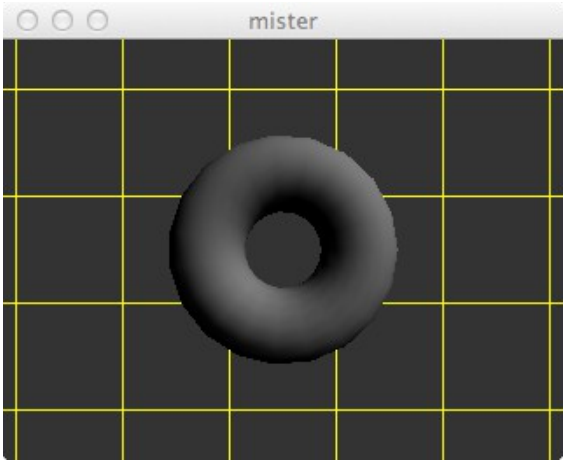


Le contexte du dessin **mister**.

• Dans la section ***Grey Shape*** du patch, cliquez sur la boîte de *message* **scale 0.3 0.3 0.3**, puis cliquez sur la boîte de *message* contenant **shape torus**. Vous devriez maintenant voir ce qui ressemble à un beignet gris.

• Cliquez sur l'objet *toggle* connecté à la boîte de *message* **lighting_enable $ 1**, puis cliquez sur l'objet *toggle* connecté à la boîte de *message* **smooth_shading $ 1**. Nous sommes maintenant en présence d'un tore gris 3D éclairé et à l'ombrage régulier.

> Par défaut, les objets GL de Jitter ont l'éclairage et l'ombrage lisse désactivés, il est donc nécessaire de les activer. L'éclairage sera traité en détail dans le *didacticiel 35*.

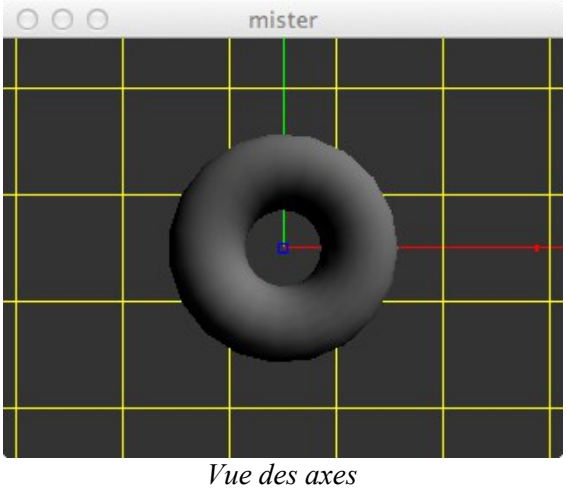


Rendus de formes avec un ombrage doux et un éclairage activé.

Position de la caméra

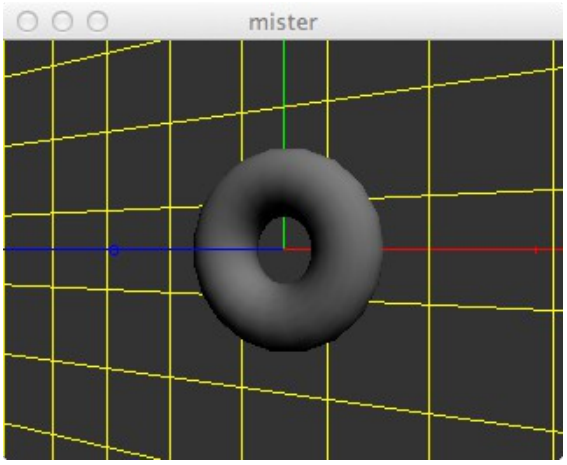
• Ouvrez le sub-patch ***Camera View*** et cliquez sur l'objet *toggle* intitulé *jit.gl.render axes*.

Vous devriez voir une ligne rouge du centre de la fenêtre vers la droite de la fenêtre, une ligne verte du centre de la fenêtre vers le haut de la fenêtre. Ce sont les axes x et y, respectivement. Ils nous aident à déterminer l'origine de notre scène. Puisque la position par défaut de la caméra est [0., 0., 2.], et que la position par défaut de **lookat** est [0., 0., 0.], la caméra regarde directement l'origine de notre scène le long de l'axe z . Par conséquent, nous ne voyons pas la ligne bleue qui représente l'axe z le long duquel la caméra regarde.



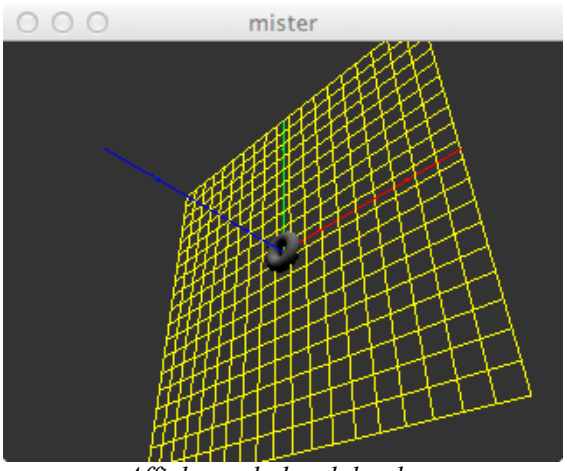
Vue des axes

• Sous l'étiquette ***camera position***, définissez la valeur x comme étant **1**. Maintenant, la caméra est à la position [1., 0., 2.], elle regarde toujours la position [0., 0., 0.], et la ligne bleue de l'axe z est devenue visible.



Les axes avec une position de visualisation différente.

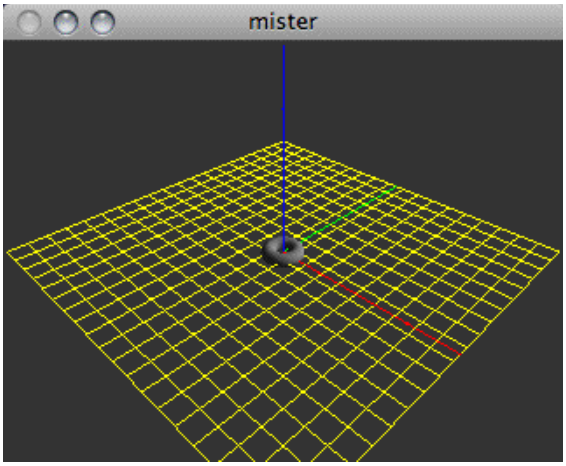
• Maintenant, définissons la valeur x de la **camera position** avec la valeur de 6., la valeur y à -6. et la valeur z à 6. de sorte que la **camera position** soit [6, –6,6]. Vous pouvez voir que le plan jaune et les axes sont, en fait, finis.



Affichage du bord du plan.

Jusqu'à présent, l'axe y a toujours été orienté vers le haut par rapport à la vue de la caméra. En effet, le vecteur "up" par défaut est [0., 1., 0.] - c'est-à-dire. le vecteur y unitaire.

• Sous l'étiquette ***up vector***, réglons la valeur y à 0. et la valeur z à 1. Nous voyons que la vue a pivoté, et que la ligne bleue de l'axe z pointe maintenant vers le haut.

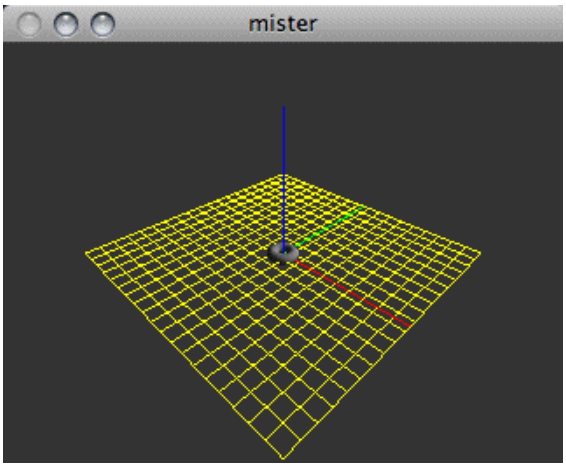


Utilisation d'un vecteur «up» différent.

Utilisation d'un vecteur "haut" différent

Vous avez peut-être remarqué, en éloignant la caméra de l'origine, que le tore, le plan et les axes sont devenus plus petits. Cela est dû au fait que le mode de visualisation par défaut utilise une projection en perspective, qui est similaire à la façon dont nous voyons les choses dans le monde tridimensionnel que nous habitons. Si vous êtes familier avec les objectifs d'un l'appareil photo, vous savez peut-être aussi qu'en fonction de l'angle de l'objectif, les objets seront plus petits à mesure que l'angle de l'objectif augmente pour permettre un plus grand champ de vision. De même, nous pouvons modifier l'angle de l'objectif de notre transformation en perspective pour augmenter le champ de vision et, ce faisant, les objets deviendront encore plus petits.

• L'angle d'objectif par défaut est de 45 degrés. Changeons-le en quelque chose comme 60 degrés en modifiant la boîte de *nombres* connectée à la boîte de *message* **lens_angle $ 1**.



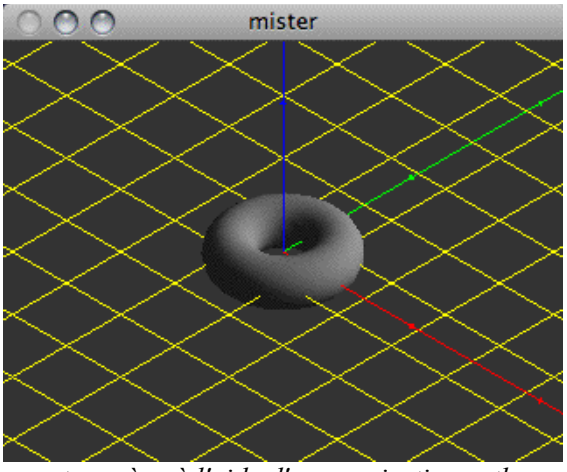
Utilisation d'un angle d'objectif de 60 degrés.

Projection orthographique

Un autre type de projection supporté par OpenGL est la projection orthographique. Ce type de projection ne diminue pas la taille des objets en fonction de la position de la caméra. La projection orthographique est commune aux logiciels de CAO 3D utilisés pour des tâches telles que l'ingénierie mécanique. De nombreux premiers jeux vidéo comme Q-Bert utilisaient également une projection orthographique. Vous pouvez basculer entre la projection en perspective et la projection orthographique en cliquant sur la boîte *toggle* intitulée ***projection orthographique***. Le message **ortho 1** active la projection orthographique. Si vous essayez de déplacer la caméra avec la

projection orthographique activée, vous ne devriez pas voir les objets devenir plus petits.

Cependant, le changement de l'angle de l'objectif modifie le champ de vision et la taille des objets par rapport à la vue.



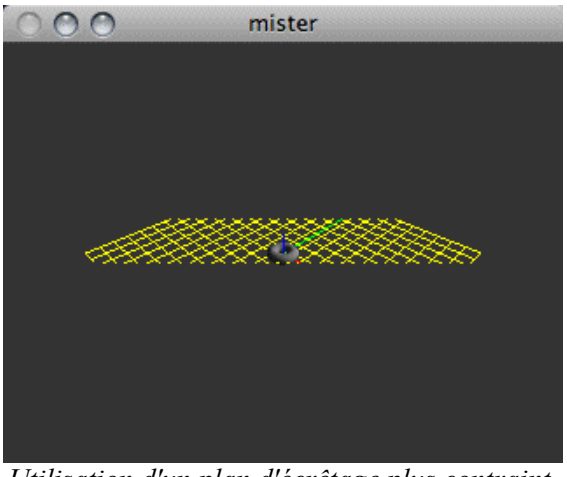
Visualiser notre scène à l'aide d'une projection orthographique.

• Cliquez à nouveau sur le *toggle* pour désactiver la projection orthographique avec un message **ortho 0**.

Plans d'écrêtage

Examinons les plans d'écrêtage qui déterminent l'étendue de la vue de la caméra qui sera rendue. OpenGL a un plan d'écrêtage proche et un plan d'écrêtage éloigné, et seule la géométrie qui se trouve entre ces deux plans sera rendue. Ces plans d'écrêtage sont spécifiés en unités de distance de la caméra le long du vecteur de visualisation à l'aide des messages **clip_near** et **clip_far**. Par défaut, le plan d'écrêtage proche est défini à 0,1 et le plan d'écrêtage éloigné à 100.

• Essayez d'augmenter le plan d'écrêtage proche à 10 et de réduire le plan d'écrêtage lointain à 12. Vous devriez voir disparaître les bords proches et éloignés du plan jaune qui se trouvent en dehors des plans d'écrêtage.



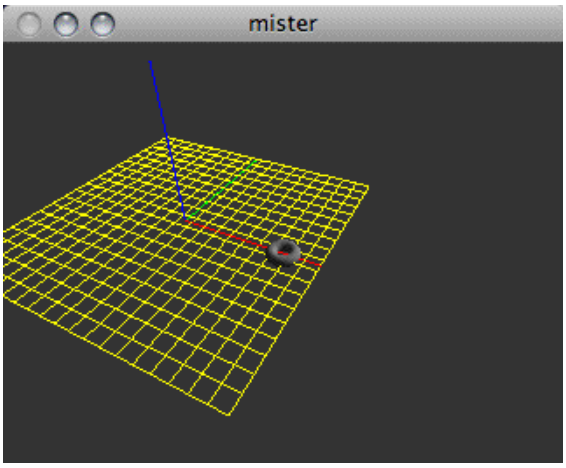
Utilisation d'un plan d'écrêtage plus contraint.

• Rétablir le plan d'écrêtage proche à la valeur par défaut de **0,1** et le plan d'écrêtage éloigné à la valeur par défaut de **100**.

Jusqu'à présent, la caméra a toujours regardé l'origine [0., 0., 0.]. Si nous modifions la valeur x du **lookat position** à 3., la caméra regarde maintenant [3., 0., 0.].

Manipulations

• Déplaçons le tore à la position [3., 0., 0.], en cliquant sur la boîte de *message* contenant **position 3. 0. 0.** dans le sub-patch intitulé ***UI Rotation and Position Control***. Le tore est maintenant à nouveau situé au centre de la vue, [3., 0., 0.].



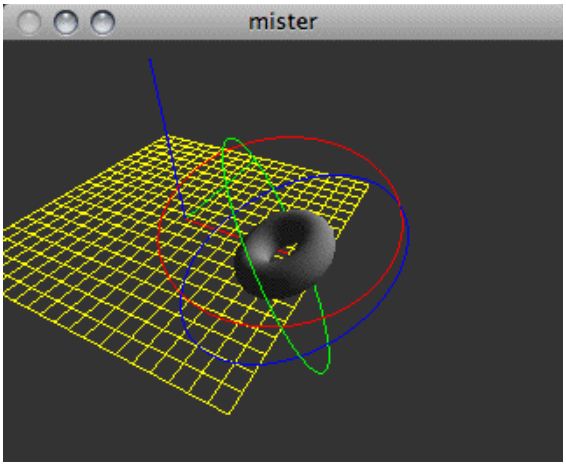
Modification de la position de visualisation et de la position de la forme.

Non seulement cela a envoyé le message **position 3. 0. 0**. au tore, mais aussi à l'objet *jit.gl.handle*. L'objet *jit.gl.handle* est un objet de groupe GL qui utilise les informations de la souris pour déplacer et faire pivoter des objets dans la scène 3D. Comme l'objet *jit.gl.gridshape*, l'objet *jit.gl.handle* nécessite un contexte de dessin nommé dans lequel il peut dessiner. Contrairement à l'objet *jit.gl.gridshape*, il s'agit également d'un objet d'interface utilisateur qui traduit l'activité de la souris dans la destination du contexte de dessin en messages Max.

Dans ce patch, les messages de l'objet *jit.gl.handle* sont envoyés à l'objet *jit.gl.gridshape*. Ils sont également envoyés à l'objet *route* **rotate position** et formatés pour que vous puissiez voir exactement ce qui est envoyé. Ces messages sont la seule communication de l'objet *jit.gl.handle* - il ne se passe rien dans les coulisses.

Si vous cliquez sur le tore et faites glisser la souris, vous verrez le tore être tourné par l'objet *jit.gl.handle* comme s'il s'agissait d'une boule de commande virtuelle. Si vous maintenez la touche commande enfoncée tout en faisant glisser la souris, vous pouvez déplacer le tore vers la gauche, la droite, le haut et le bas. Si vous maintenez la touche option pendant que vous faites glisser la souris, vous pouvez déplacer le tore vers vous ou loin de vous. Si vous utilisez la touche shift en même temps que vous effectuez l'une des actions de souris décrites ci-dessus, l'action sera limitée à un seul axe.

• Essayez de manipuler l'orientation du tore en cliquant sur lui dans l'objet *jit.window*. Découvrez comment l'objet *jit.gl.handle* traduit les informations bidimensionnelles de la souris en informations de rotation tridimensionnelles.



Utilisation de l'objet jit.gl.handle pour manipuler la position de l'objet.

Comme avec les axes affichables de l'objet *jit.gl.render*, l'objet *jit.gl.handle* affiche des lignes colorées qui correspondent aux plans x (rouge), y (vert) et z (bleu) de l'objet en cours de rotation. Les lignes apparaissent comme des cercles autour de l'objet concerné qui est "manipulé". La souris contrôle les axes dont les cercles sont les plus proches de l'avant de votre champ de vision actuel. En manipulant l'image de manière à ce que ces cercles se déplacent vers l'arrière de l'objet, vous pouvez contrôler une autre paire d'axes avec le prochain clic de souris. Les touches de modification vous permettent de repositionner l'objet en le déplaçant sur les trois axes. L'objet *jit.gl.handle* émet les messages pertinents pour définir les attributs **rotate** et **position** de l'objet de groupe GL qui lui est attaché. Notez que si vous affichez un contexte GL dans une fenêtre *jit.p*, l'option **Help in Locked Patchers** de Max (que vous pouvez modifier dans le menu Options) doit être désactivée pour que le zoom fonctionne avec *jit.gl.handle*. Sinon, la touche d'option fera apparaître le patch d'aide pour *jit.pwindow* (!).

Sommaire

Nous avons examiné les différents constituants qui composent la vue caméra d'une scène OpenGL, et les attributs nécessaires de l'objet *jit.gl.render* qui les contrôlent. L'attribut **caméra** spécifie la position de la caméra; **up** spécifie le vecteur ascendant; **lookat** spécifie la position à laquelle la caméra regarde; **ortho** spécifie s'il faut utiliser une projection orthographique ou en perspective; et **near_clip** et **far_clip** spécifient les plans d'écrêtage. Les attributs d'éclairage et d'ombrage lisse peuvent être activés en définissant les attributs **lighting_enable** et **smooth_shading** de l'objet du groupe GL qui gère la géométrie (dans ce cas, l'objet *jit.gl.gridshape*).

L'objet *jit.gl.handle* nous permet de faire pivoter et de repositionner des objets OpenGL à l'aide de la souris et des touches de modification du clavier. L'objet *jit.gl.handle* prend le nom d'un contexte de dessin valide auquel il s'attache, et envoie des messages à tout objet connecté qui utilise également ce contexte, en définissant les attributs **rotate** et **position** de cet objet.