30-Dessiner du texte en 3D

Ce didacticiel vous montre comment dessiner et positionner du texte 3D dans un objet *jit.window* en utilisant les objets *jit.gl.text3d* et *jit.gl.render*. En cours de route, nous aborderons les bases du dessin de graphiques OpenGL à l'aide de l'objet *jit.gl.render*.

L'objet *jit.gl.text3d* est l'un des nombreux objets de dessin OpenGL de Jitter qui fonctionnent conjointement avec l'objet *jit.gl.render*. OpenGL est un standard multiplateforme pour le dessin de graphiques 2D et 3D, conçu pour décrire des images afin qu'elles puissent être dessinées par des coprocesseurs graphiques. Ces coprocesseurs, également connus sous le nom d'unités de traitement graphiques ou GPU, accélèrent énormément les opérations de dessin, ce qui permet d'animer en temps réel des scènes complexes constituées de polygones texturés. Les graphiques OpenGL peuvent vous aider à créer des affichages ou des interfaces visuelles plus rapides sans alourdir le processeur de votre ordinateur.

Dans le coin inférieur gauche du patch, il y a un objet *jit.window* nommé **hello**. Cette fenêtre sera la destination de notre dessin OpenGL.

• Cliquez sur le *toggle* intitulé ***Start Rendering***.

Le *toggle* démarre l'objet *qmetro*, qui envoie des messages **bang** à un objet *trigger*. Pour chaque **bang** reçu à son entrée, l'objet *trigger* envoie le message **erase** par sa sortie droite, puis un message **bang** par sa sortie gauche. Ces messages sont envoyés à l'objet *jit.gl.render* **hello**.

Création d'un contexte de dessin

Tous les graphiques OpenGL dans Jitter sont dessinés à l'aide d'objets *jit.gl.render*. Chaque objet *jit.gl.render* doit faire référence à une destination nommée pour le dessin. Vous pouvez spécifier cette destination en utilisant un argument initial à *jit.gl.render*. Donc dans ce cas, *jit.gl.render* **hello** crée un objet *jit.gl.render* qui dessinera dans la fenêtre de destination que nous avons nommée **hello**.

> Il est important de noter que **hello** n'est pas le nom de l'objet *jit.gl.render*, mais uniquement sa destination.

Nous faisons référence à cette combinaison d'un objet *jit.gl.render* et d'une destination nommée comme un contexte de dessin. Un contexte de dessin est nécessaire pour le rendu OpenGL. Les objets sur le côté gauche de ce patch sont suffisants pour créer un contexte de dessin valide, et donc lorsque vous cliquez sur le *toggle*, le message **jit.gl.render: building GL on window 'hello'** apparaît dans la console Max. Cela vous indique que le contexte est en cours de création.

Un objet *jit.gl.render* et une destination nommée créent un contexte de dessin.

Remarque: Max version 7 a introduit un nouvel objet - *jit.world* qui contient un contexte de dessin complet: *qmetro*, *jit.gl.render* et *jit.window* avec la touche escape liée à l'affichage plein écran. Tout ce dont vous avez besoin est un *toggle* et un nom.

Objets GL dans le contexte

• Cliquez sur la boîte de *message* indiquant **Hello \, Jitter !**.

Détails techniques: l'objet *qmetro* est nécessaire dans ce patch car, contrairement à la plupart des patchs du didacticiel précédent, aucun objet *jit.matrix* n'est présent. Dans les patchs complexes, le dessin ou les calculs matriciels déclenchés de manière répétée par un objet *metro* peuvent ne pas se terminer avant que le **bang** suivant ne soit programmé. Dans cet exemple, cette situation se produirait si le texte prenait plus de 40 millisecondes pour être rendu. Normalement, le planificateur Max devrait placer ce **bang** dans sa file d'attente - une liste de messages en attente. Max n'est normalement pas autorisé à supprimer des messages de la file d'attente. Donc, dans ce patch, si chaque **bang** génére une séquence d'événements qui durait plus de 40 millisecondes, et qu'aucune suppression de messages n'est autorisée, la file d'attente finirait par déborder (***overflow***) ou par manquer d'espace pour stocker des messages en attente supplémentaires. Le programmateur s'arrêterait alors, tout comme votre show! Ce n'est probablement pas ce que vous aviez en tête. L'objet *qmetro* (qui n'est en fait qu'une combinaison d'un objet *metro* et de l'objet *jit.qball* - voir *Tutoriel 16*) vous aide à éviter cette situation en abandonnant tous les messages **bang** qui sont encore en attente pendant un calcul. Si un **bang** est programmé pour se produire pendant que le reste du patch est en cours de rendu, il sera placé dans une file d'attente. Si ce **bang** n'a pas été transmis au reste du patch au moment où le **bang** suivant se produit, le premier **bang** sera écrasé par le second, qui sera placé ensuite dans la file d'attente. Etc.

Disons que la sortie d'un objet *metro* réglé pour émettre un **bang** toutes les 10 millisecondes est envoyée à un objet *jit.movie*, suivi par quelques effets dont le dessin prend 100 millisecondes par image. L'objet *jit.movie* recevra donc 10 messages **bang** pendant le temps de calcul des effets. L'objet *jit.movie* sait dans ce cas que le traitement n'est pas encore terminé, et au lieu d'envoyer dix messages *jit_matrix* la prochaine fois qu'il en a l'occasion, il les laisse tomber sauf un. Alors que les objets *jit.movie* et *jit.matrix* ont cette capacité intégrée, l'objet *jit.gl.render* ne l'a pas. Cela est dû au fait que l'objet *jit.gl.render* aura souvent besoin de groupes de messages correspondants afin de dessiner une image correctement. Dans cet exemple, le message **erase** est nécessaire pour effacer l'écran, et le message **bang** dessine le texte. Si un message **erase** était abandonné pour éviter de faire déborder (***overflow***) la file d'attente, plusieurs messages **bang** pourraient être traités à la suite, et plusieurs copies du texte seraient dessinées en même temps, ce qui n'est pas ce que nous voulons faire. Au fur et à mesure que les patchs deviennent plus complexes, différents types d'artefacts visuels peuvent apparaître. C'est pourquoi l'objet *qmetro* est fourni pour permettre au concepteur du patch de décider des messages à supprimer de la file d'attente. Dans cet exemple, comme dans la plupart des cas, le simple fait de substituer des objets *qmetro* à des objets *metro* garantit que le dessin aura toujours un aspect correct et que la file d'attente ne débordera jamais.

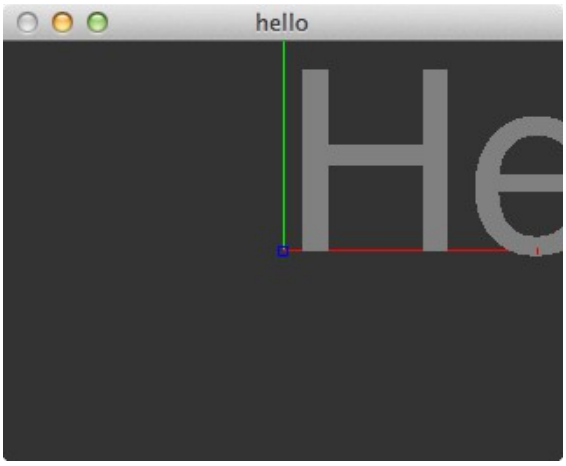
Attributs 3D communs

Tous les objets Jitter OpenGL (qui commencent tous par "jit.gl") partagent un ensemble commun d'attributs qui leur permettent d'être positionnés dans l'espace 3D, colorés et modifiés de toute autre manière - le ***groupe GL*** (ob3d). Dessiner des graphiques en 3D peut être une tâche assez complexe. Le groupe ob3d simplifie cela en garantissant que les messages que vous utilisez pour dessiner un

objet 3D fonctionneront avec tous les objets, dans la mesure du possible. Ces attributs communs sont entièrement documentés dans la section ***groupe GL*** de la ***référence d'objet*** de tout *objet jit.gl*. Pour introduire le groupe 3D ici, nous allons présenter les attributs position, rotation, échelle et axes en manipulant l'objet *jit.gl.text3d*.

Nous pouvons ajouter un ensemble d'axes spatiaux à l'objet *jit.gl.text3d* pour rendre nos manipulations spatiales plus faciles à voir et à comprendre - il est à la fois plus facile de voir comment l'objet texte est orienté, et également de voir des informations supplémentaires sur l'origine exacte de l'objet.

• Cliquez sur le *toggle* connecté à la boîte de *message* **reading axes $1** afin de voir les axes de l'objet 3D.



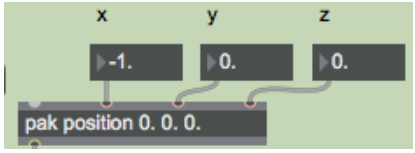
Le texte 3D à l'origine

L'axe x, dessiné en rouge, pointe vers la droite. Le zéro est au centre de l'écran et les valeurs x croissantes se déplacent en direction de la droite de l'écran. ***L'axe y***, dessiné en vert, pointe vers le haut, et ***l'axe z***, ***dessiné en bleu***, pointe hors de l'écran vers vous (comme il est pointé directement vers vous, vous ne verrez qu'un petit point bleu). Ces axes représentent le système de coordonnées local de l'objet lorsqu'il se déplace dans le monde. Lorsqu'un objet de groupe GL est créé pour la première fois, son système de coordonnées local n'est pas soumis à une rotation, ou une mise à l'échelle – par défaut, il possède le même système de coordonnées que le reste du monde 3D.

Le texte «Hello, Jitter!» est affiché dans la zone d'affichage de l'objet *jit.window*, mais il commence au centre de l'écran, dont la partie finale est coupée. Déplaçons le texte vers la gauche.

• Définissez la boîte de *nombre* intitulée x dans la section ***attributs 3D communs*** du patch sur la valeur **–1**.

La boîte de *nombre* envoie une valeur à virgule flottante à l'objet *pak*, qui envoie le message **position** suivi de trois chiffres à l'objet *jit.gl.text3d*. Lorsque vous modifiez la valeur dans la boîte de *nombre*, vous pouvez voir le texte glisser vers la gauche jusqu'à ce qu'il soit entièrement visible à l'écran.



Modification de l'attribut de position

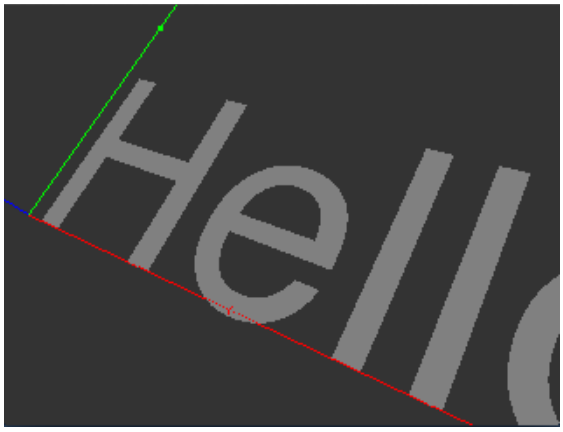
Nous venons de modifier l'attribut position de l'objet *jit.gl.text3d*. Le message **position** peut être suivi de trois arguments numériques qui définissent la position de l'objet en trois dimensions. Si moins de trois nombres suivent le message **position**, les axes sont remplis dans l'ordre [x, y, z] et la position de l'objet sur les axes non spécifiés est fixée à 0. Par exemple, l'envoi du message **position 5**. fixera la position d'un objet de groupe GL à l'emplacement [5, 0, 0].

L'opération consistant à modifier la position d'un objet est appelée ***translation***.

Maintenant, faisons pivoter le texte.

• Définissez la valeur **1** dans les boîtes de *nombres* intitulées x, y et z directement au-dessus de l'objet *pak* **rotation** dans la section **Attributs 3D communs** du patch. Cela définit l'axe de rotation.

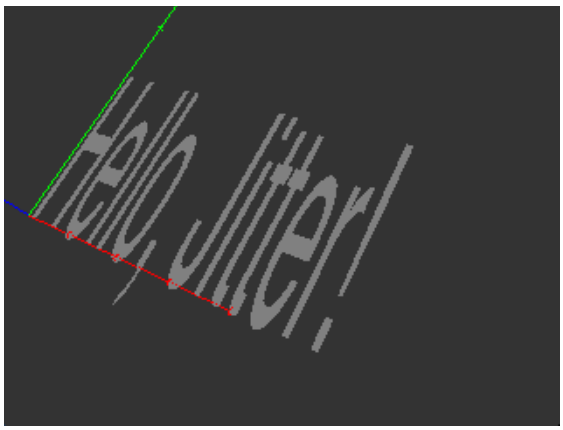
• Faites glisser la boîte de *nombre* intitulée **angle** jusqu'à la valeur **320**. Vous verrez le texte pivoter autour de l'axe dans la position indiquée dans cette capture d'écran.



Le texte 3D après traduction et rotation

L'envoi du message **rotation** suivi d'un à quatre chiffres définit l'attribut de rotation d'un objet de groupe GL. Le premier nombre est une quantité de rotation en degrés, dans le sens inverse des aiguilles d'une montre autour de l'axe de rotation de l'objet. Les trois autres nombres spécifient cet axe de rotation sous la forme d'un vecteur [x, y, z]. Comme pour l'attribut **position**, certaines valeurs suivant l'attribut rotation peuvent être supprimées, et les valeurs par défaut seront incluses. Si le message de rotation n'est suivi que d'un seul chiffre, ce chiffre correspond à l'angle en degrés et l'axe de rotation est [0, 0, 1]. Cela fait pivoter l'objet autour de l'axe z, ou en d'autres termes, dans le plan x-y de l'écran. Si deux ou plus accompagnent le message de rotation, le premier spécifie toujours l'angle de rotation, et les suivants spécifient le vecteur de rotation dans l'ordre dans lequel ils apparaissent.

• Réglez la boîte de *nombre* étiquetée x directement au-dessus de l'objet *pak* **scale** sur **0,25**. Cela permet de mettre à l'échelle le texte 3D au quart de la taille le long de son axe x local.



Le texte 3D après traduction, rotation et mise à l'échelle

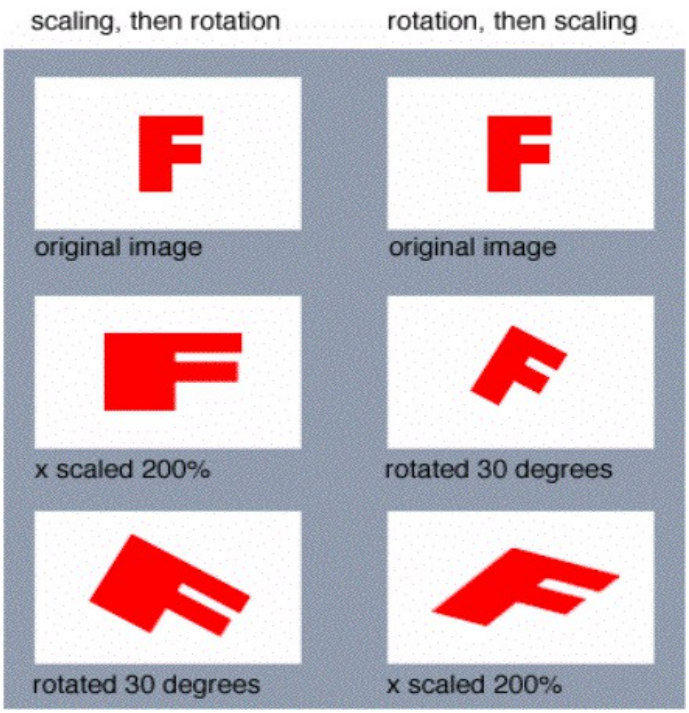
Notez que l'axe rouge est mis à l'échelle en même temps que l'objet. Les points situés le long de la ligne sont maintenant deux fois plus proches le uns des autres qu'avant l'opération de mise à l'échelle. Les axes sont toujours dessinés dans le système de coordonnées local du groupe GL, qui dans ce cas a été translaté, pivoté et mis à l'échelle par rapport au système de coordonnées universelles dans l'objet *jit.gl.render*.

Il est important de tenir compte de l'ordre des opérations lorsque vous effectuez des transformations géométriques. Vous pouvez définir les attributs de rotation, de position et d'échelle d'un objet indépendamment dans l'ordre que vous souhaitez. Mais à chaque fois qu'il est dessiné, l'objet est transformé dans l'ordre suivant:

1 - Mise à l'échelle
2 - Rotation
3 –Traduction

Il est essentiel de respecter cet ordre pour que les attributs se comportent de manière prévisible. Si la rotation se produisait avant la mise à l'échelle, par exemple, un message **scale 0,5** mettrait l'objet à l'échelle non seulement dans ses coordonnées x, mais dans une combinaison de x, y et z en fonction de la rotation de l'objet.

L'exemple suivant montre la différence que fait l'exécution d'opérations dans des ordres différents.



L'ordre des opérations fait toute la différence

Sommaire

Créer un contexte de dessin est la première étape pour utiliser des graphiques OpenGL dans Jitter. Un contexte de dessin consiste en une destination nommée, comme une fenêtre, et un objet *jit.gl.render* qui dessine vers cette destination.

Il existe une variété d'objets Jitter qui dessinent des graphiques OpenGL en coopération avec *jit.gl.render*; leurs noms commencent tous par «jit.gl.» L'objet *jit.gl.text3d* en est un exemple. Tous les objets OpenGL de Jitter partagent un ensemble commun d'attributs permettant de les déplacer dans l'espace 3D. Ce groupe d'objets est appelé ***groupe GL***.