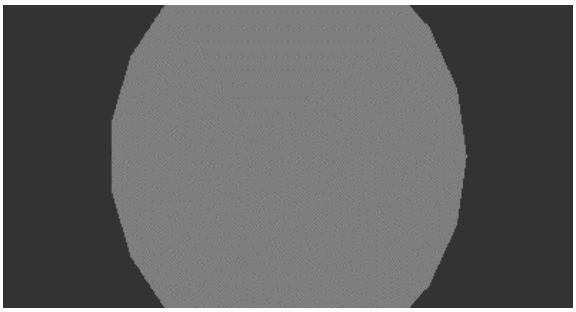
33-Modes polygonaux, couleurs et mélange

Dans le didacticiel précédent, vous avez vu comment positionner la caméra et les objets du groupe GL pour construire une scène OpenGL dans Jitter. Après avoir compris ce didacticiel, vous serez en mesure de masquer les polygones sélectionnés d'un objet OpenGL en fonction de leurs orientations spatiales, de dessiner les polygones sélectionnés en modes plein et filaire, et d'ajouter les résultats au tampon de dessin à l'aide de l'anticrénelage et le mélange.

• Ouvrez le patch du didacticiel. Cliquez sur la boîte *toggle* intitulée «Start Rendering».

Vous devriez voir une sphère grise dans la fenêtre *jit.pwindow* du patch de tutoriel. Elle est dessinée par un objet *jit.gl.gridshape*, relié à un objet *jit.gl.handle* qui vous permet de contrôler sa rotation. L'attribut **auto_rotate** de l'objet *jit.gl.handle* est activé, donc une fois que vous l'avez fait tourner, la sphère continuera à tourner le long de l'axe que vous avez défini. Si vous le souhaitez, vous pouvez la faire tourner.

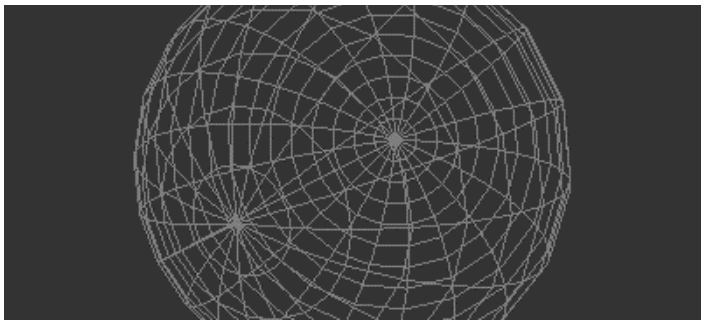


La sphère grise.

Mode filaire et sélection de faces

Juste en dessous de l'étiquette «OpenGL Objects to Render», dans le patch d'exemple se trouve un objet *pak* **poly_mode 1 1**. Cet objet génère des messages qui définissent l'attribut mode polygone de l'objet *jit.gl.gridshape*.

• Cliquez sur les deux objets *toggle* au-dessus de l'objet *pak* **poly_mode**. Vous devriez voir la sphère grise en mode filaire.

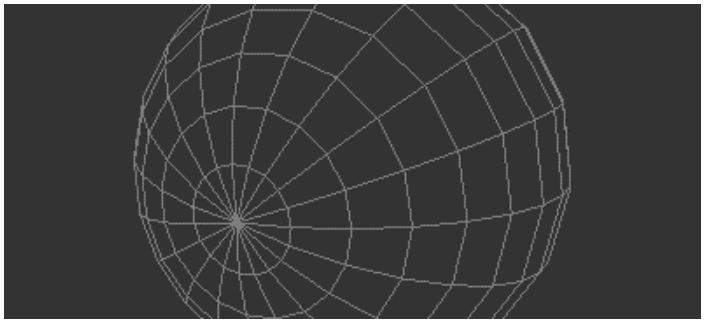


La sphère en mode filaire.

Activer le mode filaire vous permet de voir clairement que l'objet *jit.gl.gridshape* se rapproche d'une sphère à l'aide de polygones - dans ce cas, des quadrilatères. Chaque polygone dessiné dans OpenGL a un côté frontal, défini comme le côté à partir duquel ses sommets semblent s'enrouler dans le sens des aiguilles d'une montre. Chaque polygone d'une scène peut donc être classé comme orienté vers l'avant ou vers l'arrière, selon que sa face avant est orientée vers ou loin de la caméra.

OpenGL peut automatiquement masquer les polygones orientés vers l'avant ou vers l'arrière, ce qui permet d'accélérer considérablement le dessin ou de mettre en évidence certains aspects des données visualisées. Dans Jitter, vous pouvez contrôler la visibilité des polygones sur une base objet par objet à l'aide de l'attribut **cull_face**.

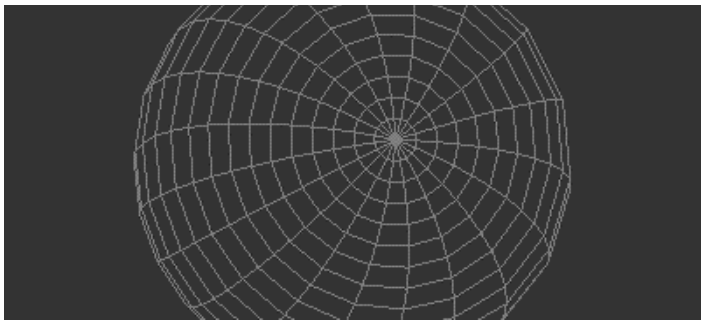
• Définissez la boîte de *nombre* au-dessus de l'objet *prepend* **cull_face** sur **1**.



Les polygones orientés vers l'avant de la sphère.

L'attribut **cull_face** d'un objet GL group peut être défini sur **0**, **1** ou **2**. Une valeur de **0** affiche tous les polygones de l'objet. Une valeur de **1** masque les polygones orientés vers l'arrière. La valeur **2** permet de masquer les polygones orientés vers l'avant. Avec le réglage actuel de **1**, la sphère filaire semble solide, car ***les lignes cachées*** - les bords des polygones qui ne seraient pas visibles si la sphère était faite d'un matériau solide - ne sont pas dessinées. La rotation (l'avez-vous fait tourner?) représente de manière convaincante celle d'un objet solide du monde réel.

• Définissez la boîte de *nombre* au-dessus de l'objet *prepend* **cull_face** sur **2**.

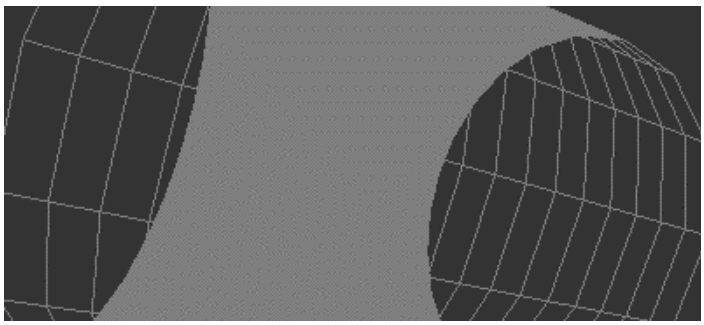


Les polygones orientés vers l'arrière de la sphère.

Maintenant, les polygones orientés vers l'avant sont cachés. C'est comme si vous regardiez à travers la sphère et ne voyiez que la partie intérieure tournée vers vous. Vous pouvez voir sur cette image que la perspective est quelque peu étrange, mais en regardant la rotation dans le patch, il est vraiment évident que la scène n'est pas dessinée «normalement».

En général, le paramètre **cull_face 1** permet de supprimer les polygones qui devraient être cachés pour les objets solides et convexes tels que la sphère. Mais pour les objets qui ne sont pas solides, une combinaison des polygones de la face avant et de la face arrière peut être visible.

• Définissez la boîte de *nombre* au-dessus de l'objet *prepend* **cull_face** sur **0** pour afficher tous les polygones. Définissez la case de gauche sur **0** (désactivée). Réglez l'*umenu* au-dessus de l'objet *prepend* **shape** sur «opencylinder».



Cylindre ouvert, avec des polygones pleins orientés vers l'avant et des polygones filaires orientés vers l'arrière.

En utilisant cette nouvelle forme, nous pouvons voir la distinction entre les polygones faisant face à l'avant et à l'arrière. Le message **poly_mode a b**, étant donné deux entiers **a** et **b**, définit les polygones faisant face à l'avant en mode filaire si **a** est égal à **1**, et les polygones faisant face à l'arrière en mode filaire si **b** est égal à **1**.

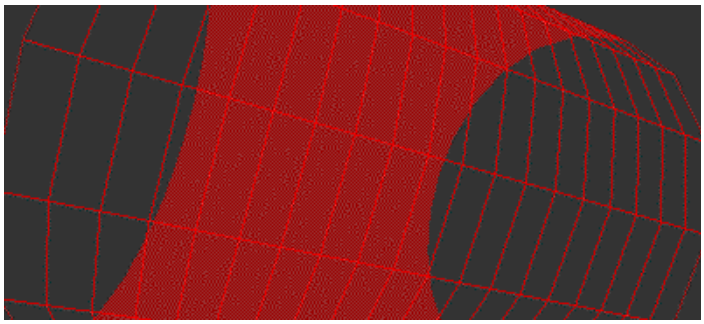
Couleurs RGBA

Les couleurs des objets du groupe GL sont spécifiées à l'aide du message **color R G B A**, où R est la composante rouge de la couleur, B est le bleu, G est le vert et A est la composante alpha ou opacité. Toutes les valeurs sont comprises entre **0** à **1**.

• Définissez les objets de la boîte de *nombre* au-dessus de l'objet *pak* **color**… sur les valeurs **1., 0., 0., 0.5**. Cela spécifie un rouge pur avec une opacité de 50%.

Vous verrez la couleur du cylindre devenir rouge, mais l'opacité n'est pas visible. C'est est dû au fait que le ***mélange,*** c'est-à-dire le mélange des pixels avec ceux qui se trouvaient précédemment dans le tampon de dessin, est désactivé par défaut.

• Cliquez sur la boîte *toggle* au-dessus de la boîte de *message* **blend_enable $ 1** pour activer le mélange pour l'objet *jit.gl.gridshape*. Vous devriez voir quelque chose comme ceci:



Le cylindre rouge avec le mélange activé.

ARGB vs RGBA Si vous avez utilisé les objets de manipulation vidéo de Jitter, vous savez que les couleurs de ces objets sont stockées dans des plans et spécifiées dans des arguments dans l'ordre A, R, G, B. Dans le groupe d'objets GL, l'ordre est RGBA, avec alpha en dernier, comme nous le voyons ici. Cela peut sembler étrange, on doit donc vous fournir quelques explications. Les objets Jitter sont liés aussi étroitement que possible aux formats natifs dans les domaines OpenGL et vidéo, afin de permettre un traitement aussi rapide que possible. OpenGL stocke les couleurs des objets et des sommets au format RGBA, et QuickTime stocke ses images au format ARGB. Les objets Jitter reflètent donc cela. Si vous souhaitez combiner le traitement des matrices OpenGL et vidéo dans votre patch, les objets

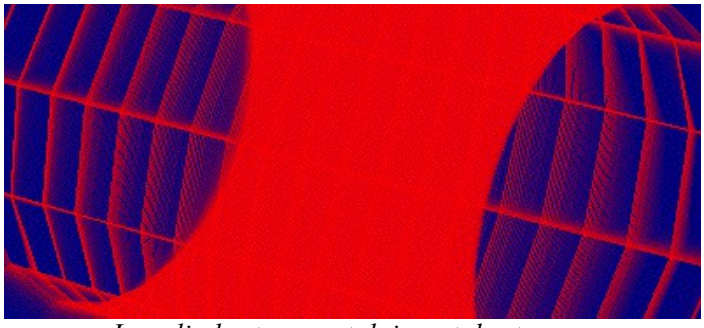
pack, *unpack*, *jit.pack* et *jit.unpack* fournissent un moyen facile de conversion entre les deux systèmes. Vous pouvez également convertir une matrice de valeurs de caractères d'ARGB en RGBA en envoyant la matrice à travers un objet *jit.matrix* avec l'attribut **planemap** défini sur **1 2 3 0** (décalant efficacement tous les plans d'une unité). Le *didacticiel 6* montre d'autres exemples d'utilisation de l'attribut **planemap** de l'objet *jit.matrix*.

Couleur d'effacement et tracés

Actuellement, chaque fois que l'objet *jit.gl.render* reçoit le message **erase**, le tampon de dessin est rempli avec le gris foncé qui est la couleur d'effacement par défaut. Vous pouvez définir une couleur d'effacement différente à l'aide des boîtes de *nombres* RGBA au-dessus de l'objet de rendu.

• Définissez les boîtes de *nombres* au-dessus de l'objet *jit.gl.render* sur les valeurs **0, 0, 0,5 et 0,1**.

Le fond devient bleu foncé (rouge = 0, vert = 0, bleu = 0,5). Si le cylindre tourne, vous verrez également des traînées dans l'image. Cela est dû au fait que l'arrière-plan est effacé avec une opacité de 0,1. Lorsque les pixels bleu foncé sont dessinés par dessus les pixels existants dans le tampon de dessin, ils sont superposés de telle sorte que le résultat est un dixième de bleu foncé et neuf dixièmes de la couleur qui se trouvait précédemment sur chaque pixel. Par conséquent, les images précédentes persistent pendant un certain temps avant d'être entièrement effacées. Notez que si l'attribut **blend_enable** doit être défini pour voir les couleurs de dessin avec une opacité partielle, il n'est pas nécessaire pour effacer avec une opacité partielle.



Le cylindre tournant, laissant des traces.

Modes de fusion

Lorsque l'attribut **blend_enable** d'un objet du groupe GL est activé, chaque pixel est appliqué au tampon de dessin à l'aide d'une fonction de ***fusion***. La fonction de fusion est l'opération de base de la composition d'images. Elle contrôle l'application de nouveaux pixels, la source, sur des pixels existants, la destination. La fonction comprend deux parties: un facteur de mélange de la source et un facteur de mélange de destination. Le facteur de mélange de la source spécifie comment la contribution de la source à l'image finie doit être calculée. Le facteur de mélange de destination spécifie la contribution de la destination.

Jitter reconnaît onze modes possibles pour les facteurs de mélange. Certains peuvent être appliqués uniquement à la source, d'autres uniquement à la destination, et d'autres encore aux deux. Chaque mode spécifie un ensemble différent de valeurs multiplicatrices pour le rouge, le vert, le bleu et l'alpha. Le message **blend_mode** [**src_factor**] [**dest_factor**] vous permet de spécifier les deux facteurs pour tous les objets de dessin du groupe GL.

Les modes de fusion.
Les facteurs de fusion source et destination sont des quadruplets RGBA qui sont multipliés par composant par les valeurs RGBA des pixels source et destination, respectivement. Les composants correspondants de la source et de la destination sont ajoutées et ensuite fixées à la plage [**0, 1**] pour produire le pixel de sortie.

Ce tableau montre comment les facteurs de fusion sont calculés. La colonne «Mode» indique le nombre transmis dans le message Jitter **blend_mode** [**src_factor**] [**dest_factor**]. La colonne «Nom OpenGL» indique le nom du mode. La colonne «Relevant for» indique si le mode peut s'appliquer au facteur source, au facteur de destination ou aux deux. Enfin, la colonne «Blend Factor Equation» indique la formule réelle utilisée pour calculer le pixel. Les indices s et d font référence aux composants source et destination, respectivement. Par exemple, R_s fait référence à la composante rouge du pixel source.

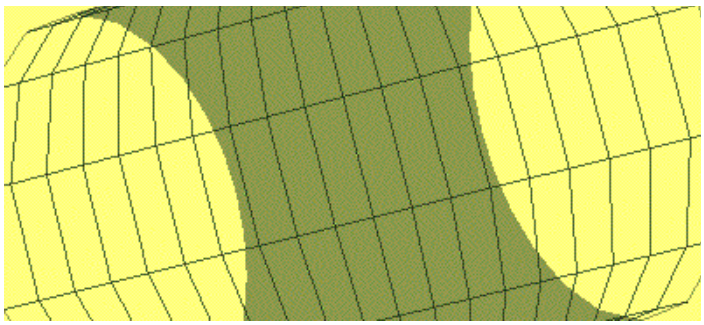
Mode	*Nom OpenGl*	*pertinent pour*	*l'équation du facteur de mélange*
0	GL_ZERO	les deux	(0, 0, 0, 0)
1	GL_ONE	les deux	(1, 1, 1, 1)
2	GL_DST_COLOR	source	(R_d, G_d, B_d, A_d)
3	GL_SRC_COLOR	destination	(R_s, G_s, B_s, A_s)
4	GL_ONE_MINUS_DST_COLOR	source	(1, 1, 1, 1) - (R_d, G_d, B_d, A_d)
5	GL_ONE_MINUS_SRC_COLOR	destination	(1, 1, 1, 1) - (A_s, A_s, A_s, A_s)
6	GL_SRC_ALPHA	les deux	(A_s, A_s, A_s, A_s)
7	GL_ONE_MINUS_ SRC_ALPHA	les deux	(1, 1, 1, 1) - (A_s, A_s, A_s, A_s)
8	GL_DST_ALPHA	les deux	(A_d, A_d, A_d, A_d)
9	GL_ONE_MINUS_ DST_ALPHA		(1, 1, 1, 1)- (A_d, A_d, A_d, A_d)
10	GL_SRC_ALPHA_SATURATE	source	(f, f, f, 1); f = min (A_s, 1A_d)

Les modes de fusion source et destination par défaut pour tous les objets du groupe GL sont respectivement **6** et **7**. Ils correspondent aux facteurs de fusion GL GL_SRC_ALPHA et GL_ONE_MINUS_SRC_ALPHA. Il s'agit d'une opération de fusion très couramment utilisée, et peut-être vous n'aurez jamais besoin d'une autre. Elle vous permet d'effectuer un fondu enchaîné intuitif entre la source et la destination en modifiant la valeur alpha de la source.

D'autres valeurs sont utiles pour simuler diverses situations d'éclairage du monde réel, ainsi que des effets spéciaux qui n'ont pas de contrepartie physique.

• Définissez les boîtes de *nombres* RGBA au-dessus de l'objet *jit.gl.render*, qui contrôlent l'attribut **erase_color**, sur les valeurs **1.0, 1.0, 0.5 et 1.0**. Cela définira la couleur de l'arrière-plan sur le jaune et supprimera les traînées, rendant l'effet de fusion plus visible.

• Définissez les boîtes de *nombres* gauche et droite au-dessus de l'objet *pak* **blend_mode** sur **0** et **7** respectivement. Ceci spécifie un facteur de fusion source de GL_ZERO et un facteur de fusion destination de GL_SRC_ALPHA.



*Le cylindre avec un réglage de **blend_mode 0** 7.*

Examinons comment ce **blend_mode** produit l'image que nous voyons ici. Le facteur source est GL_ZERO. Cela signifie que tous les composantes du pixel source sont multipliées par 0 - le pixel source n'a aucun effet. Vous pouvez le vérifier en essayant différentes valeurs RVB pour le cylindre. Elles produisent toutes les mêmes couleurs.

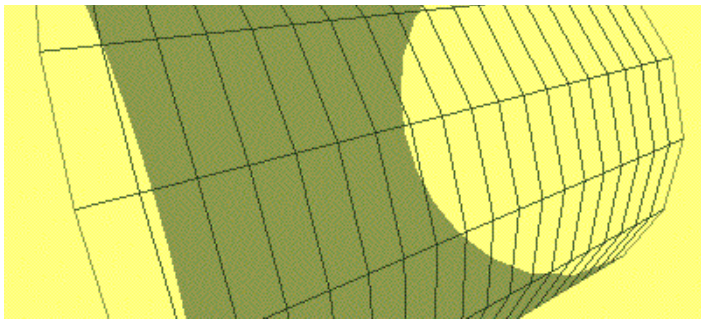
Le facteur de destination est GL_SRC_ALPHA. En regardant dans le tableau ci-dessus, nous pouvons trouver l'équation du facteur de mélange à laquelle cela correspond: (A_s, A_s, A_s, A_s). Chaque composante du pixel de destination est multipliée par l'alpha de la source, dans ce cas **0,5**, avant d'être ajoutée au pixel source multiplié par le facteur source, qui dans ce cas est **0**. Ainsi, chaque fois qu'un pixel est dessiné, sa luminosité est réduite de moitié.

Antialiasing

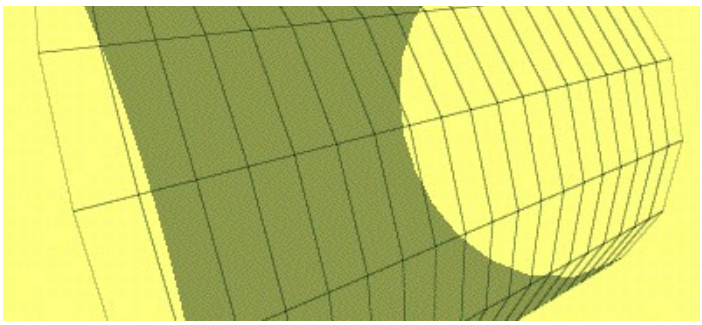
Quand OpenGL dessine des polygones et des lignes, il rapproche leurs formes géométriques idéales en remplissant des pixels d'une grille matricielle. Ce processus est sujet à des difficultés similaires à celles rencontrées lors de la reconstruction de formes d'onde idéales en audio numérique. Inévitablement, le crénelage (***aliasing***), c'est-à-dire des fréquences spatiales qui ne sont pas incluses dans l'image idéale, sera introduit une fois celle-ci reconstruite à partir de pixels discrets. Ce crénelage est visible sous la forme de ce que l'on appelle communément des «saccades», en particulier sur les lignes ou les bords presque horizontaux ou verticaux.

OpenGL dispose de ceraines techniques d'anticrénelage pour réduire les saccades. Nous les avons rendus disponibles dans Jitter à travers les attributs du groupe d'objets GL. En utilisant ces attributs, vous pouvez spécifier si un objet donné du groupe GL utilisera l'anticrénelage lorsqu'il sera dessiné.

• Activez la boîte *toggle* au-dessus de la boîte de *message* **antialiasing $ 1** pour envoyer le message **antialias 1** à l'objet *jit.gl.gridshape*.



Antialiasing désactivé



Antialiasing activé

Les lignes antialiasing ont une apparence plus lisse, mais aussi plus grasse. Elles peuvent aussi se dessiner plus lentement, donc si vous êtes préoccupé par la vitesse de dessin, vous devez décider si l'apparence améliorée vaut le temps supplémentaire.

Le comportement de l'antialiasing dans OpenGL dépend de l'implémentation. Cela signifie que les fabricants de matériel et de pilotes OpenGL ont une certaine marge de manoeuvre pour décider de ce qui se passe exactement lorsque vous demandez l'anticrénelage. Dans les images ci-dessus, par exemple, notez que si les irrégularités sur les lignes sont réduites, les bords du polygone en bas à gauche n'ont pas changé. Lorsque vous activez l'antialiasing, Jitter demande que les bords des polygones soient anti-crénelés. Mais l'implémentation OpenGL particulière qui a généré ces images (un accélérateur ATI Rage 128 avec les pilotes de la version 5.9.8) n'offre aucune aide à cet égard. Votre implémentation peut être différente.

Sommaire

Nous avons défini des polygones faisant face à l'avant et à l'arrière, et vu comment les dessiner en modes solide et filaire (en utilisant les attributs **poly_mode** et **cull_face**). L'attribut **erase_color** du rendu et son utilisation pour dessiner des traînées ont été introduits. Nous avons défini en détail ce qui se passe lorsqu'un pixel source est appliqué à une destination de tampon de dessin, en tenant compte de l'opacité et des modes de fusion. Enfin, lnous avons introduit la fonction d'anticrénelage d'OpenGL, pratique mais quelque peu imprévisible..