Les tampons : Frame Buffer, Stencil Buffer, Depth Buffer, Accumulation Buffer et autres buffers

Marc Neveu

Rappel du Pipe-Line Open-GL

- **[Données]->[Evaluateurs]->[Opérations sur les sommets et assemblage de primitives]->[Discrétisation]->[Opérations sur les fragments]->[Image]**
- Les Evaluateurs produisent une description des objets à l'aide de sommets et de facettes.
- Les Opérations sur les sommets sont les transformations spatiales (rotations et translations) qui sont appliquées aux sommets.
- L' assemblage de primitives regroupe les opérations de clipping : élimination des primitives qui sont en dehors d'un certain espace et de transformation perspective .
- La discrétisation (Rasterization) est la transformation des primitives géométriques en fragments correspondant aux pixels de l'image.
- Les Opérations sur les fragments vont calculer chaque pixel de l'image en combinant les fragments qui se trouvent à l'emplacement du pixel (transparence, élimination des surfaces cachées…).

Rappels

- Le tampon servant à stocker la couleur de chaque pixel est appelé ***Tampon chromatique*** ou ***Frame-Buffer***.
- Le tampon servant à stocker la profondeur de chaque pixel est appelé ***Tampon de profondeur*** ou ***Z-Buffer***.
- Le tampon servant à restreindre le rendu à certaines portions de l'écran est appelé ***Tampon Stencil*** ou ***Stencil-Buffer***.
- Le tampon servant à accumuler une série d'images dans une image finale est appelé ***Tampon d'Accumulation*** ou ***Accumulation-Buffer***.

Stencil

- stencil buffer : tampon permettant de n'afficher que certaines portions de l'écran déterminées par l'utilisateur.
- Test du stencil : comparer une valeur de référence à celle contenue dans chacun des pixels du tampon.
- En fonction du résultat le tampon est modifié et le passage des fragments affecté.

- NB : ce test a lieu avant le test de profondeur (Z-test).

- initialisation du stencil buffer:
  - associer le stencil buffer à la fenêtre *GLUT (=>GLUT_STENCIL)* à la définition du mode d'affichage *void glutInitDisplayMode(unsigned int mode)*.
  - initialisation des valeurs contenues dans le tampon (*void glClearStencil(GLint s)) void glClear(GLbitfield mask)* avec au moins le paramètre *GL_STENCIL_BUFFER_BIT*.

- Par défaut le test du stencil n'est pas actif (tous les fragments passent.) Pour l'activer ou le désactiver
  - *void glEnable(GL_STENCIL_TEST)*
  - *void glDisable(GL_STENCIL_TEST)*.

```
... void display(void)
{
 glClear(       GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT |
GL_STENCIL_BUFFER_BIT);
 ...
}
 ...
void init(void){
... glClearStencil(0);
...
 }
...
 int main(int argc, char** argv)
{
...
 glutInitDisplayMode(        GLUT_DOUBLE |
                             GLUT_RGBA |
                             GLUT_DEPTH |
                             GLUT_STENCIL);
...
init();
 glutDisplayFunc(display);
...
}
...
```

Test du Stencil

- comparer une valeur de référence à celle contenue dans chaque pixel du tampon.
- fonction *void glStencilFunc(GLenum func, GLint ref, GLuint mask)*.
  - *ref* est la valeur de référence
  - *mask* représente un masque de bits qui sera appliqué à la valeur de référence et à celle du tampon.
  - test du stencil : (*ref* & *mask*) *func* (stencil & *mask*).
  - La variable *func* peut prendre ces différents états :
    - GL_NEVER : Le test est toujours faux;
    - GL_ALWAYS : le test est toujours vrai;
    - GL_LESS : le test est vrai si la valeur *ref* est strictement inférieure à celle du tampon;
    - GL_GREATER : le test est vrai si la valeur *ref* est strictement supérieure à celle du tampon;
    - GL_LEQUAL : le test est vrai si la valeur *ref* est inférieure ou égal à celle du tampon;
    - GL_GEQUAL : le test est vrai si la valeur *ref* est supérieure ou égale à celle du tampon;
    - GL_EQUAL : le test est vrai si la valeur *ref* est égale à celle du tampon;
    - GL_NOTEQUAL : le test est vrai si la valeur *ref* est différente de celle du tampon.

Modification du Stencil

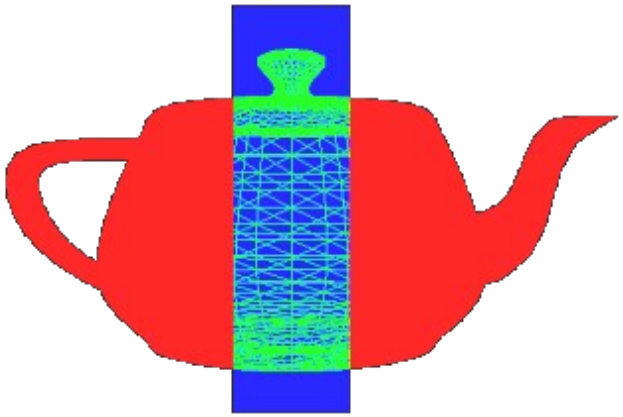
- La fonction *void glStencilOp(GLenum fail, GLenum zfail, GLenum pass)* permet de déterminer la manière dont le stencil buffer sera modifié en fonction du résultat du test.
- Le paramètre *fail* est utilisé si le test échoue.
- Le paramètre *zfail* est utilisé si le test stencil et le test de profondeur échouent.
- Sinon c'est le paramètre *pass* qui est utilisé.
- *fail*, *zfail* et *pass* peuvent prendre les valeurs suivante :
  - GL_KEEP : La valeur dans le tampon est conservée;
  - GL_ZERO : la valeur dans le tampon devient nulle;
  - GL_REPLACE : La valeur du tampon est remplacée par la valeur de référence;
  - GL_INCR : la valeur du tampon est incrémentée de un;
  - GL_DECR : la valeur du tampon est décrémentée de un;
  - GL_INVERT : les bits de la valeur du tampon sont inversés.

```
...
void display(void)
{
glClear(GL_COLOR_BUFFER_BIT |GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT); ...
glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_ALWAYS,1,1);
glStencilOp(GL_KEEP,GL_KEEP,GL_REPLACE);

glPushMatrix();
glColor3f(0.0f,0.0f,1.0f);
glTranslatef(pos,0.0f,0.0f);
glBegin(GL_QUADS);
        glVertex3f(-0.2f,0.7f,0.2f); glVertex3f(-0.2f,-0.7f,0.2f);
        glVertex3f(0.2f,-0.7f,0.2f); glVertex3f(0.2f,0.7f,0.2f);
glEnd();
glPopMatrix();

glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);

glStencilFunc(GL_EQUAL,1,1);
glStencilOp(GL_KEEP,GL_KEEP,GL_KEEP);
glColor3f(0.0f,1.0f,0.0f);
glutSolidTeapot(0.7);
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
glDisable(GL_STENCIL_TEST);
...
 }
...
```

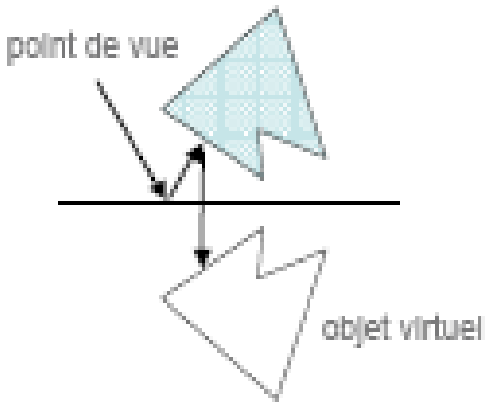


Applications :

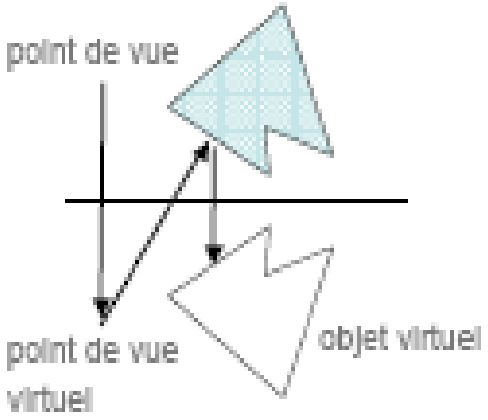
reflets et miroirs,
contours et silhouettes,…

réflexion / plan

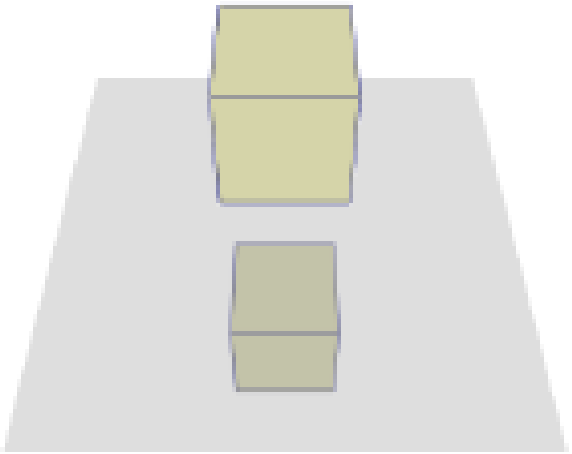
Symétrie de l'objet / plan



Autre point de vue



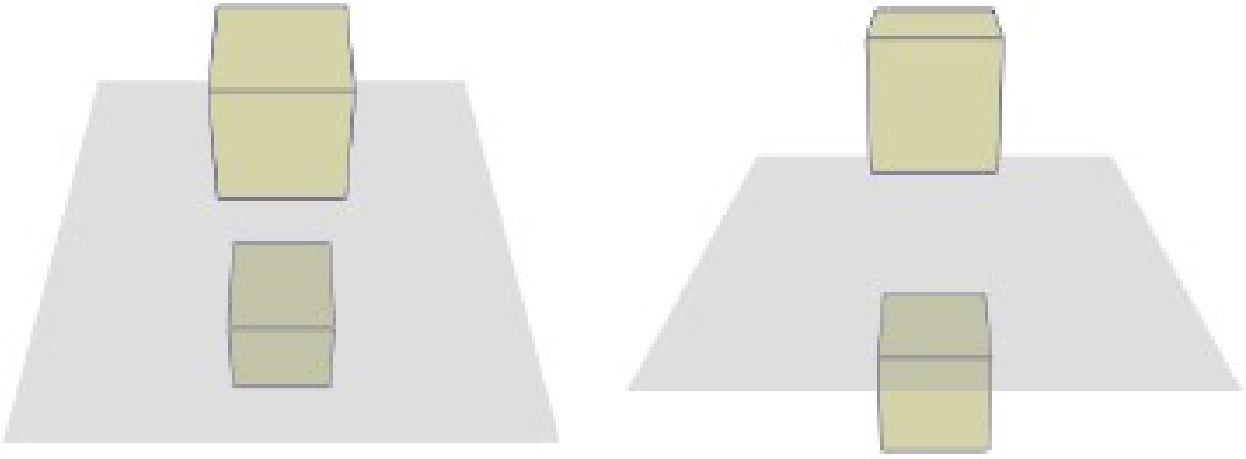
réflexion / plan



On dessine la réflexion de l'objet : glScalef(1,-1,1);

On dessine le véritable objet et le miroir

Gasp !

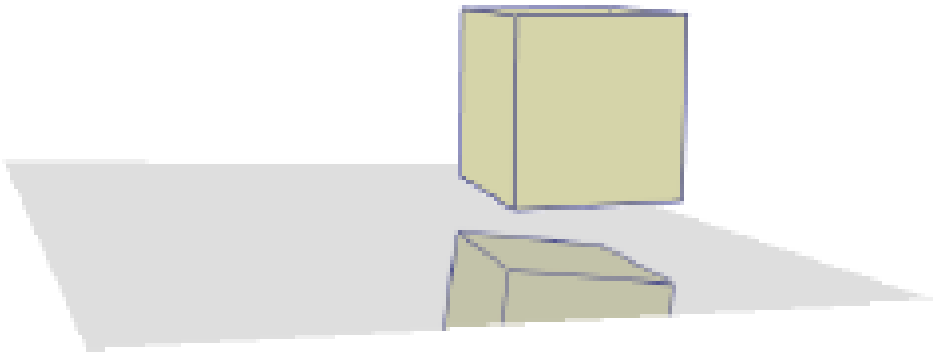


Pb : limiter la réflexion au miroir

Stencil Test

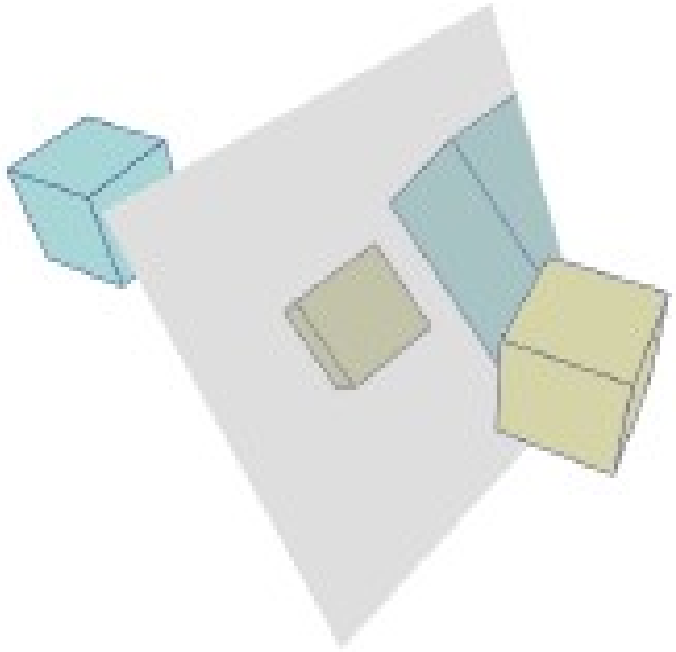
On écrit 1 dans le stencil buffer dans la partie correspondant au miroir et 0 ailleurs

On dessine la réflexion avec le stencil test actif



Re Gasp !

Les objets derrière le miroir sont transformés par symétrie : ils passent devant

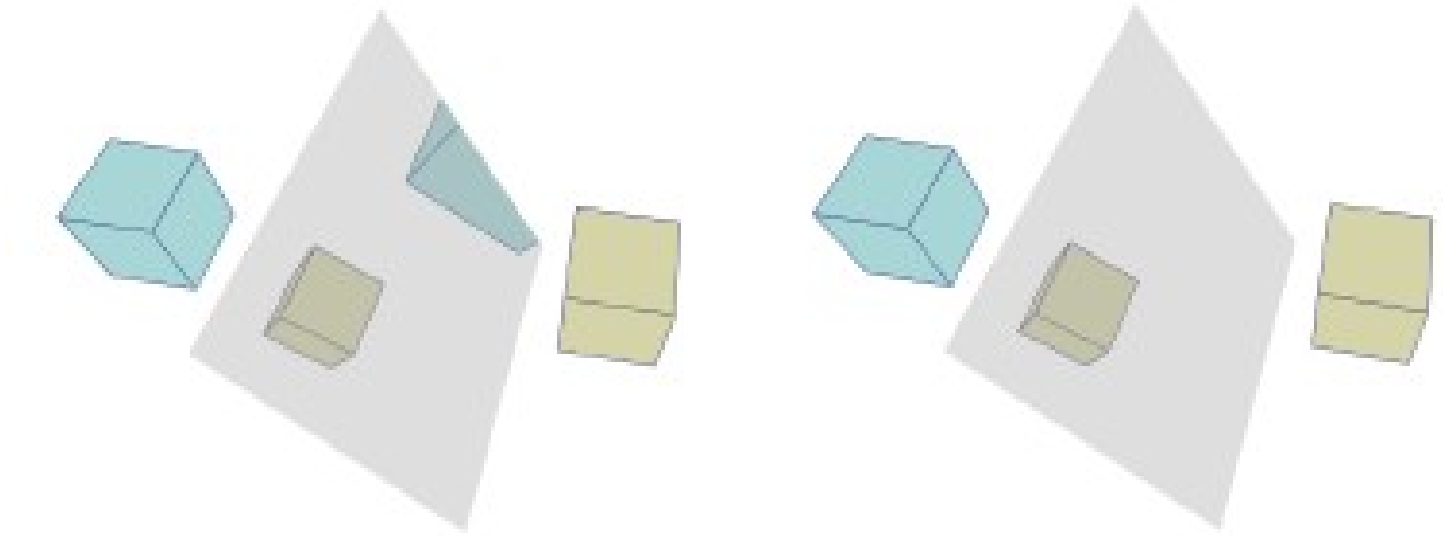


Supprimer la réflexion de ces objets

Plan de clipping

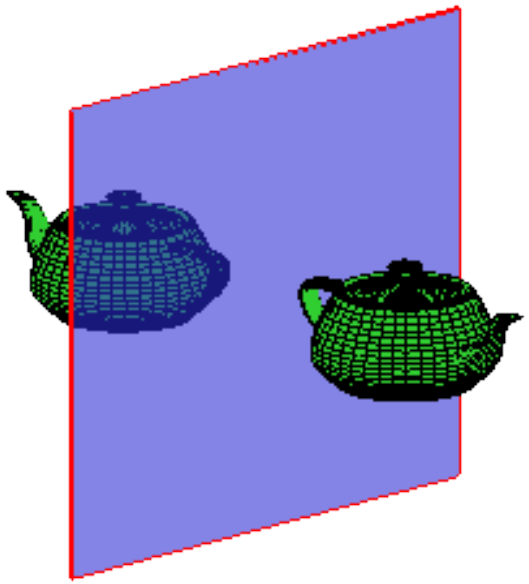
Plan de clipping additionnel (glClipPlane)

Plan choisi : celui du miroir



Exemple de miroir (E Bittar, Reims)

Utilisation du tampon stencil



Lorsque l'on dessine les reflets des objets, il faut s'assurer qu'ils ne seront visibles qu'à l'intérieur du miroir. Sinon on obtient un effet peu crédible, comme l'illustre l'image ci-contre

Stencil Buffer

C'est le Tampon Stencil qui permet de restreindre l'affichage des reflets à une certaine zone de l'image. Le Tampon Stencil est une zone mémoire de même taille que l'image, et qui va servir de masque : On y écrira des uns à l'intérieur du miroir et des zéros ailleurs.

Recette :

Réserver une zone mémoire correspondant au Tampon Stencil lors de l'initialisation : **glutInitDisplayMode**(GLUT_STENCIL)

Lors du rendu de la scène : Activer le test du Stencil **glEnable**(GL_STENCIL_TEST)

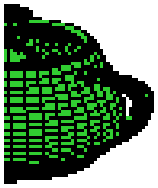
Vider le Tampon Stencil **glClear** (GL_STENCIL_BUFFER_BIT), ce qui initialise toutes les valeurs à zéro.

Paramétrer le test du Stencil avec les deux fonctions **glStencilFunc**(GL_ALWAYS, 1, 1) et **glStencilOp**(GL_REPLACE, GL_REPLACE, GL_REPLACE), de manière à remplacer les valeurs du Tampon Stencil par la valeur un pour tous les fragments qui seront calculés.

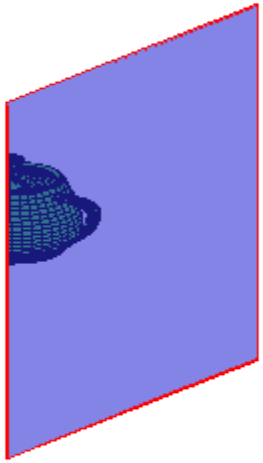
Désactiver l'écriture dans toutes les composantes du Tampon Chromatique, à l'aide d'un **glColorMask**(false, false, false, false) avant de dessiner la surface du miroir, puisqu'à ce stade, il ne s'agit pas encore de mettre à jour l'image.

Dessiner la surface du miroir : ce qui va mettre à jour à la fois le Tampon Stencil et le Tampon de profondeur, si celui-ci est activé.

Réactiver l'écriture dans le Tampon Chromatique : **glColorMask**(true, true, true, true)

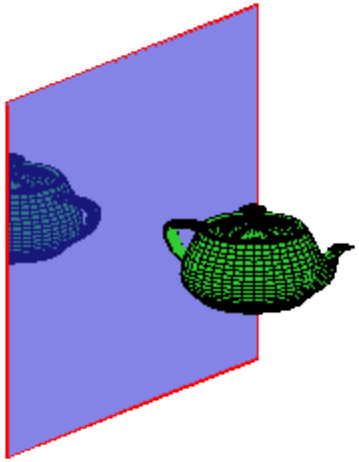


Dessiner dans le Tampon Chromatique les reflets des objets dans le miroir en utilisant le Tampon Stencil comme masque, avec **glStencilFunc**(GL_EQUAL, 1, 1) et **glStencilOp**(GL_KEEP, GL_KEEP, GL_KEEP) pour indiquer qu'on ne modifie pas la valeur du Tampon Stencil



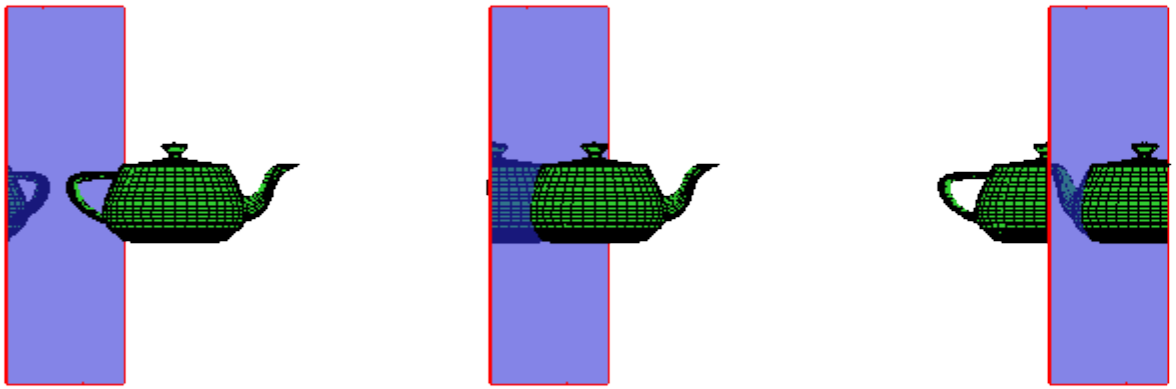
•Désactiver le test du Stencil avec **glDisable**(GL_STENCIL_TEST)

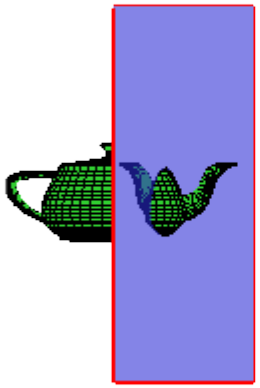
et Dessiner la surface du miroir dans le Tampon Chromatique, en utilisant la transparence (Blending) pour mélanger la couleur du miroir avec celle du reflet.



Dessiner les objets de la scène.

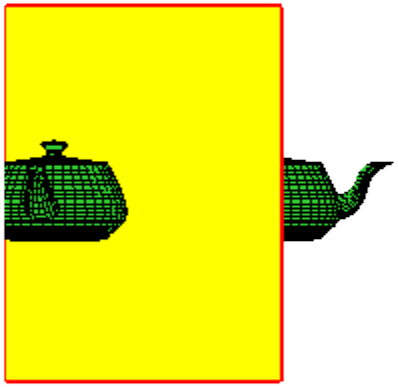
- Problème : si l'objet est derrière le miroir, le reflet apparaît devant



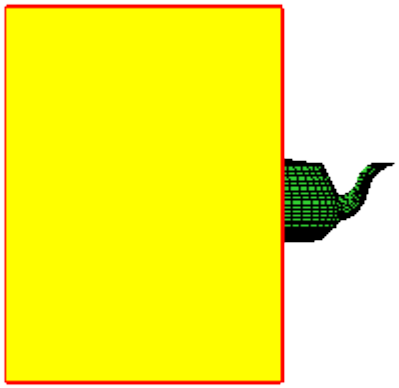


- positionner un plan de clipping.
  - Activer le plan de clipping **glEnable** (GL_CLIP_PLANE0);
  - Spécifier les 4 coefficients du plan (a, b, c, d)=eqn[0..3],
  - Placer le plan dans la scène **glClipPlane**(GL_CLIP_PLANE0, eqn);
  - Dessiner le reflet
  - Désactiver le plan de clipping **glDisable**(GL_CLIP_PLANE0)

le Culling



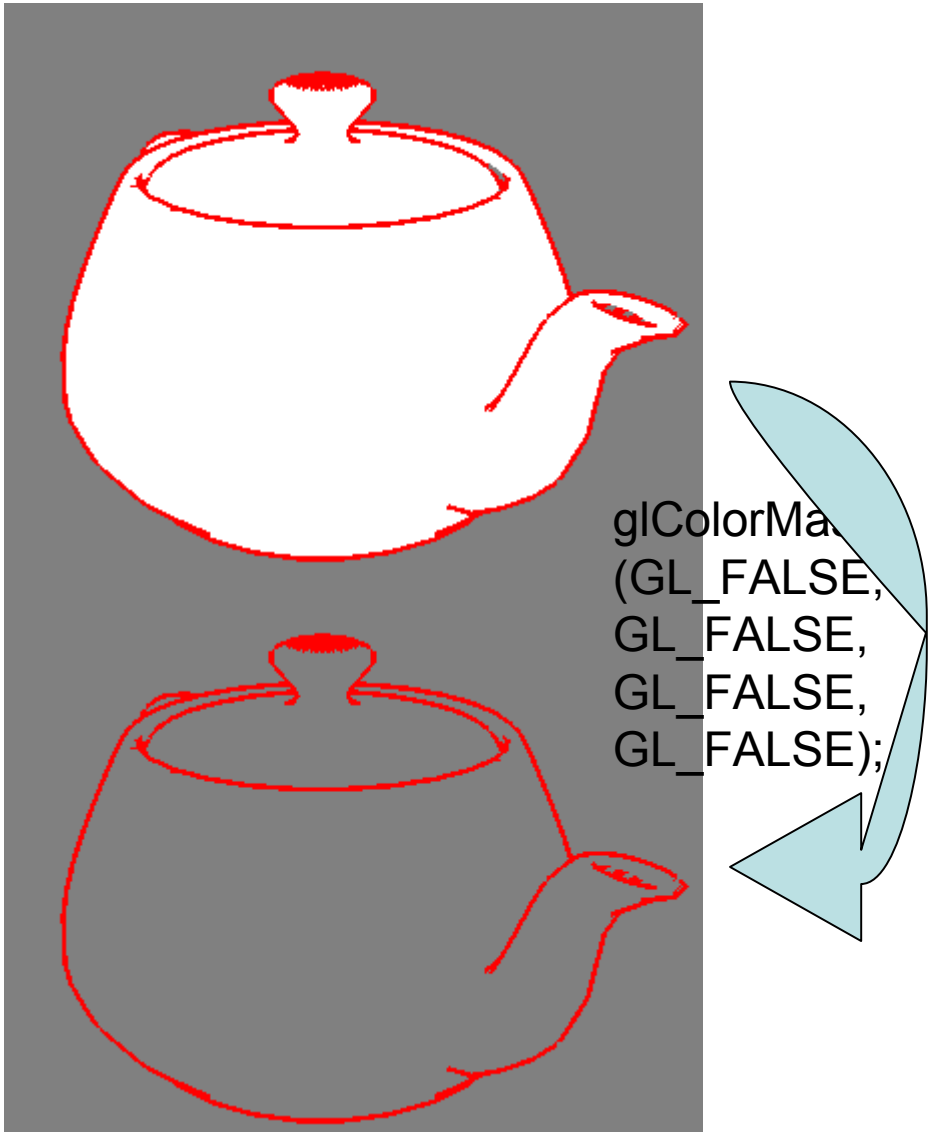
Et derrière le miroir?
la théière est derrière le miroir, son reflet est donc **devant** le miroir par rapport à l'utilisateur,
mais **derrière** par rapport au miroir : le plan de clipping ne le supprime donc pas...



lors de l'affichage du miroir dans le Tampon Stencil, on n'affiche que sa face avant:
glCullFace(GL_BACK);
glEnable(GL_CULL_FACE);
On dessine le miroir, et on réactive si l'on veut l'affichage des faces arrières :
glDisable(GL_CULL_FACE);

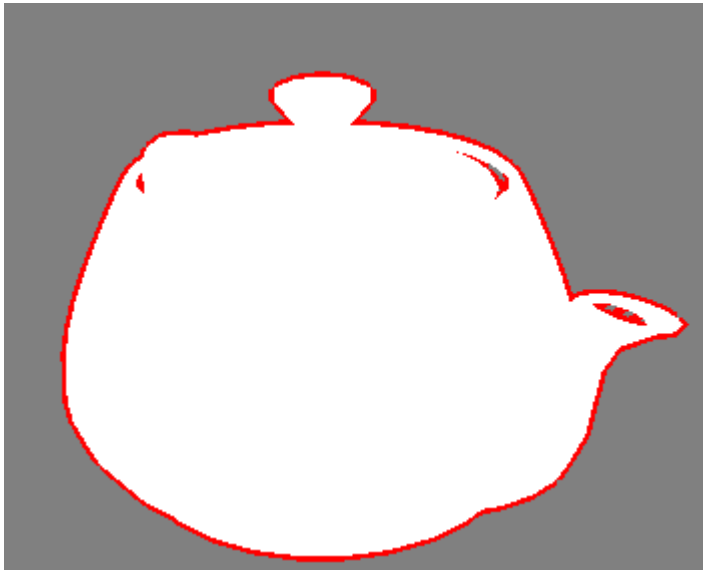
ainsi, le Tampon Stencil ne sera marqué qu'avec l'endroit du miroir !

Contours et silhouettes



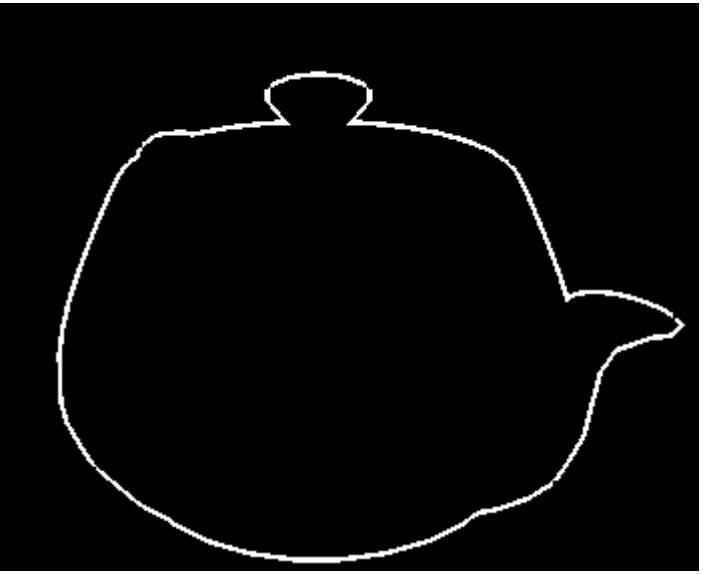
- Tracer les polygones avant en mode plein (avec illumination) et éliminer les faces arrière
- Tracer les polygones arrière en mode ligne (sans illumination) et éliminer les faces avant
- glDepthFunc (GL_LEQUAL);

Contours et silhouettes



- Tracer les polygones arrière en mode ligne (sans illumination) et éliminer les faces avant
- Tracer les polygones avant en mode plein (avec illumination) et éliminer les faces arrière
- Pas de test de profondeur

Contours et silhouettes



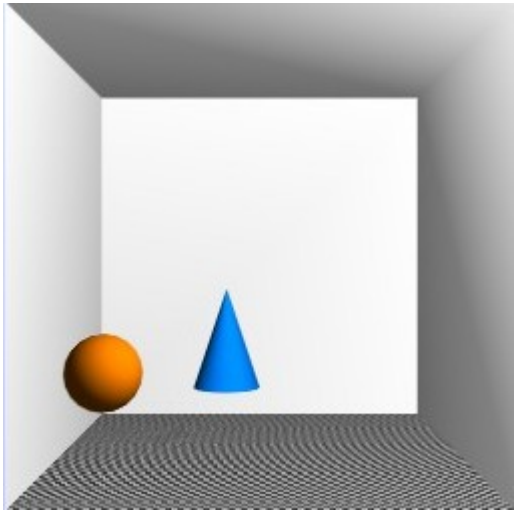
- On trace les faces avant en mode plein
  - mais en désactivant les tampons de couleur et de profondeur
  - Initialisation du stencil buffer : on trace toujours dedans
  - On trace l'objet plein pour faire un masque
- On trace les faces arrière en mode ligne
  - en rétablissant les tampons de couleur et de profondeur
  - Initialisation du stencil buffer : on ne trace pas dans le masque
  - Pas d'illumination
  - Mode ligne

Le tampon d'accumulation

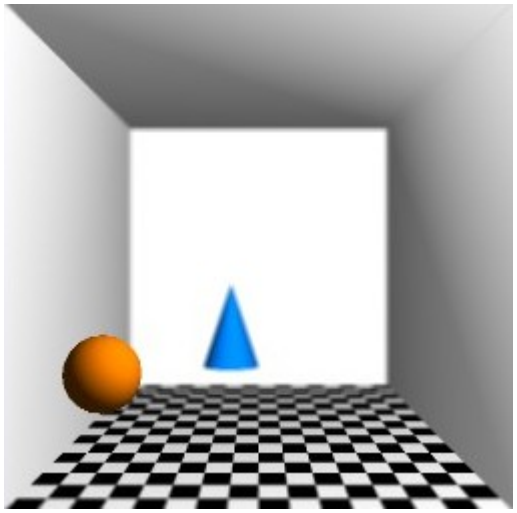
- Accumulatuion d'une série d'images dans le tampon d'accumulation.
- Lorsque l'accumulation est terminée, le résultat est copié dans un tampon chromatique pour affichage.

- Plusieurs restitutions d'une scène prennent plus de temps qu'une seule, mais le résultat est de meilleure qualité.
- void **glAccum**(GLenum op, GLfloat value); contrôle le tampon d'accumulation.
  Le paramètre *op* sélectionne l'opération, et peut avoir pour valeur GL_ACCUM, GL_LOAD, GL_RETURN, GL_ADD ou GL_MULT.
  value est un nombre à utiliser dans cette opération.
  - GL_ACCUM lit chaque pixel du tampon sélectionné avec **glReadBuffer**(), multiplie les valeurs RVBA par *value* et ajoute les valeurs obtenues au tampon d'accumulation.
  - GL_LOAD est semblable à GL_ACCUM, mais les valeurs calculées remplacent le contenu du tampon d'accumulation.
  - GL_RETURN prend les valeurs du tampon d'accumulation, les multiplie par *value*, et place le résultat dans le ou les tampons chromatiques activés en écriture.
  - GL_ADD ajoute (resp. GL_MULT multiplie) *value* à (resp. par) la valeur de chaque pixel du tampon d'accumulation et retourne le résultat dans le tampon d'accumulation. Pour GL_MULT, *value* est arrondi à l'intervalle [-1.0, 1.0], GL_ADD il n'est pas arrondi.

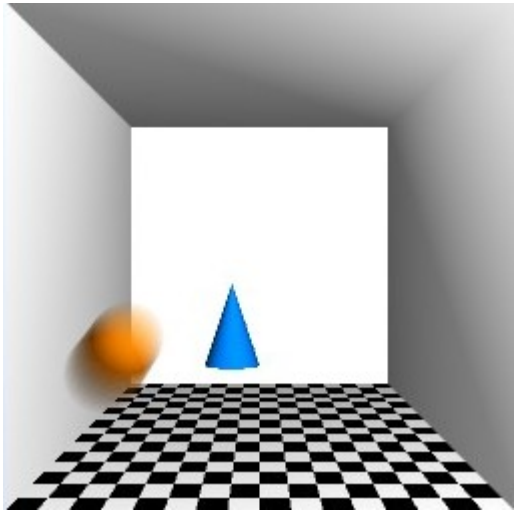
Tampon d’accumulation



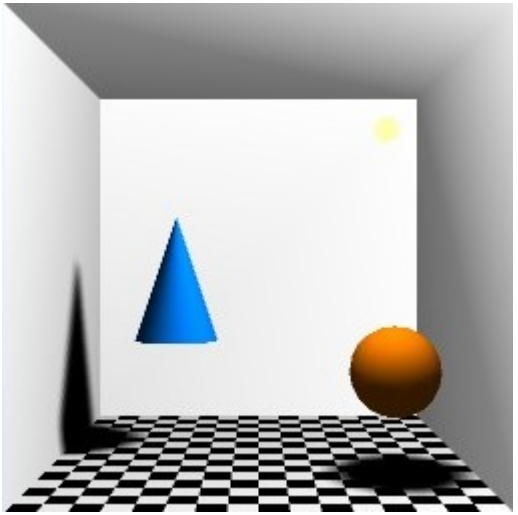
Antialiasing



Profondeur de champ

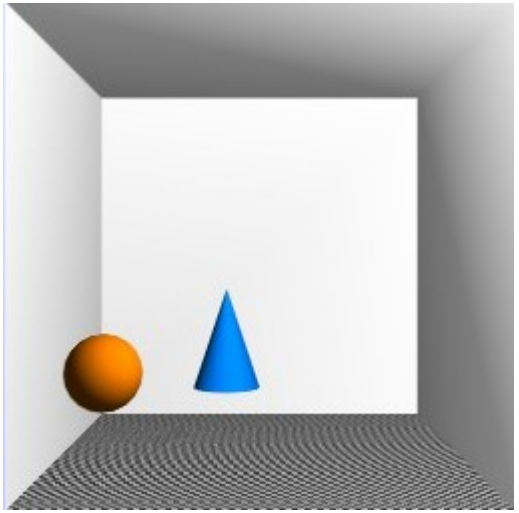


Flou (motion blur)



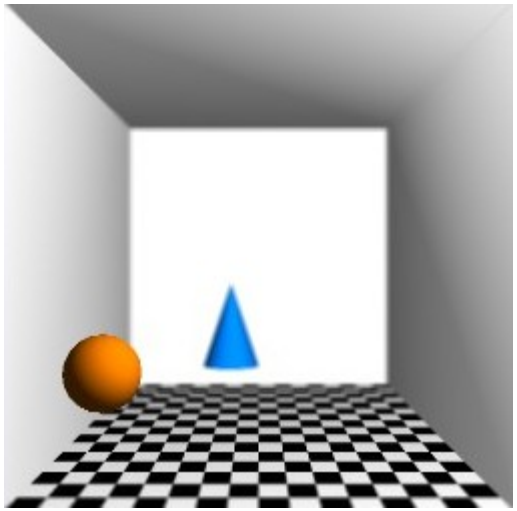
Ombres douces

antialiasing



- accumuler *n* versions de la même scène, légèrement décalées :
  - Commencer par vider le tampon d'accumulation
  - Accumuler chaque image par : glAccum(GL_ACCUM, 1.0/n);
  - Afficher l'image finale par glAccum(GL_RETURN, 1.0);
- Pour éviter que les *n* images intermédiaires de la scène ne soient affichées, les dessiner dans un tampon chromatique qui n'est pas affiché.

Profondeur de champ



- La mise au point d'une photographie n'est parfaite que pour les éléments se trouvant à une certaine distance de l'appareil photographique.
- Pour simuler cet effet avec OpenGL, le tampon d'accumulation va servir à accumuler différentes images dans lesquelles seul un plan de mise au point va rester stable.