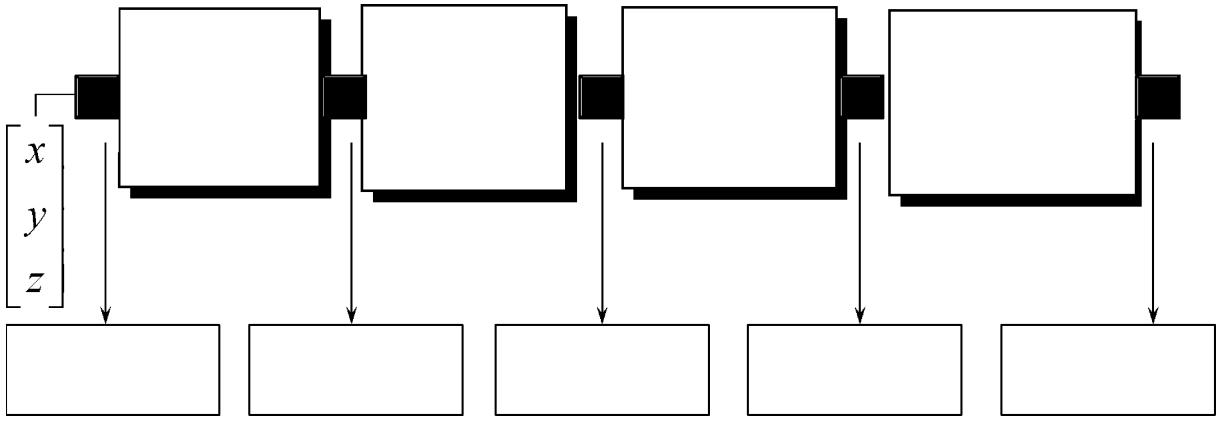
Question 1 Notions de base

a) Pour les transformations géométriques 3D, pourquoi utilise-t-on une matrice 4X4 au lieu de 3X3 ? Comment s'appelle le système de coordonnées ainsi créé ? **[2 points]**

b) Comparez les deux approches suivantes au niveau de leurs avantages et inconvénients respectifs : Mode indexé avec table de couleurs de 256 entrées (avec composantes R, G et B codées sur 8 bits) VS mode direct « trueColor » codé sur 24 bits. **[2 points]**

c) Remplissez correctement les cases vides de ce schéma du pipeline openGL de la transformation des sommets avec les neuf concepts suivants : coordonnées normalisées, matrice de projection, coordonnées de visualisation, transformation de clôture, coordonnées de découpage, transformation de normalisation, coordonnées d'objet, coordonnés d'affichage, matrice de modélisation. **[3 points]**



Question 2 Notions de base

a) À quoi sert l'algorithme de Bresenham ? **[2 points]**

b) Qu'est-ce qu'un fragment ? Comment sont créés les fragments ? Quelle est la différence entre un fragment et un pixel ? **[3 points]**

c) Qu'est-ce qu'un nuanceur ? À quoi servent les nuanceurs ? **[3 points]**

d) Nous avons vu la fonction servant à spécifier les propriétés de matériau des faces avant ou arrière des primitives graphiques. Par exemple : `glMaterialfv( GL_FRONT, GL_DIFFUSE, coeffs );`
Comment la librairie OpenGL détermine-t-elle la face avant d'une primitive ? **[2 points]**

e) Les applications OpenGL, comme celle que vous avez développée, utilisent généralement un tampon de profondeur.

i) Que contient ce tampon ?

ii) Quel est son rôle ?

iii) L'application peut-elle y accéder directement ?

[3 points]

Question 3 Notions de base

a) Une mise à l'échelle est fondamentalement une opération de *multiplication*, tandis qu'une translation est, au contraire, une opération d'*addition*. Pourtant, la librairie OpenGL cumule toutes ces opérations dans une seule matrice de transformation courante qui n'est utilisée, elle, que pour *multiplier* des coordonnées. Comment est-ce possible ? (Expliquez brièvement le fonctionnement.) **[2 points]**

b) Distinguez les concepts de sommet, de fragment et de pixel. **[3 points]**

c) Durant le cours, nous avons présenté trois modèles d'affichage.

i) Identifiez ces trois modèles d'affichage.

ii) Pour chacun, décrivez brièvement en quel format l'information graphique est conservée.
(Une image est décomposée en ...)

iii) Donnez un cas exemple où l'utilisation de chacun est appropriée.

[4 points]

d) Nous avons vu quelques modèles de couleur en classe.

i) Nommez trois modèles de couleur que nous avons vus.

ii) Pour chacun, décrivez les *trois* axes qui le définissent

iii) Dessinez le *volume* contenant les couleurs que le modèle définit.

[4 points]

e) Nous avons aussi parlé de modèles de couleur additif et soustractif.

i) Expliquez ce qu'est un modèle de couleur soustractif (par opposition à un modèle additif).

ii) Nommez une application courante de ce type de modèle.

[2 points]

f) Convertissez la couleur CMY (0.4, 0.75, 0.1) en couleur RGB. **[1 point]**

Question 4 Notions de base

a) Considérez les quatre sommets dans les schémas de droite. **Dessinez** le résultat visuel des énoncés de chaque sous-question **et dites** aussi si ce résultat était, *oui* ou *non*, celui le plus probablement attendu par le programmeur.

i)
```
glBegin( GL_QUAD_STRIP );
  glVertex2fv( S1 );                    S1 o     o S4
  glVertex2fv( S2 );
  glVertex2fv( S3 );
  glVertex2fv( S4 );                    S2 o     o S3
glEnd( );
```

ii)
```
glBegin( GL_TRIANGLE_STRIP );
  glVertex2fv( S1 );                    S1 o     o S4
  glVertex2fv( S2 );
  glVertex2fv( S3 );
  glVertex2fv( S4 );                    S2 o     o S3
glEnd( );
```

iii)
```
glBegin( GL_TRIANGLE_FAN );
  glVertex2fv( S1 );                    S1 o     o S4
  glVertex2fv( S2 );
  glVertex2fv( S3 );
  glVertex2fv( S4 );                    S2 o     o S3
glEnd( );
```

[3 points]

b) Soit une scène où l'on voit une sphère en projection perspective.

i) Combien de point(s) de fuite peut-on facilement identifier s'il n'y a que cette sphère ? Pourquoi ?

ii) Est-ce que votre réponse serait différente si la scène contenait plutôt un cylindre ? Pourquoi ?

[2 points]

c) À quoi sert l'algorithme du point milieu ? **[2 points]**

d) Qu'est-ce que GLSL ? **[2 points]**

e) Qu'est-ce qu'un anaglyphe ? **[2 points]**

f) Qu'est-ce que WebGL et quel est le lien avec OpenGL ? **[2 points]**

Question 5 Notions de base

a) Le langage PostScript est un langage mis au point par Adobe en 1985 et qui est utilisé aujourd'hui par une grande majorité d'imprimantes.

i) Dans ce langage, quelle est l'unité de mesure de base des coordonnées ?

ii) Ce langage possède la fonction `newpath`. Qu'est-ce qu'un « `path` » ?

[2 points]

b) Lors du tramage des primitives, les coordonnées (x,y,z) des sommets sont interpolées afin de produire la position de chaque fragment. Nommez deux autres attributs standards qui peuvent être spécifiés aux sommets et qui seront interpolés pour chaque fragment par la carte graphique lors du tramage. **[2 points]**

c) La fonction `void glDrawArrays( GLenum mode, GLint debut, GLsizei nombre );` peut être employée pour afficher 12 quadrilatères en 3D comme dans l'extrait ci-dessous :

```
glEnableClientState( GL_VERTEX_ARRAY );
GLdouble sommets[3*4*12] = { ... };   // 3 coo./sommet * 4 sommets/quad * 12 quads
glVertexPointer( NDim, GL_DOUBLE, 0, sommets );
glDrawArrays( GL_QUADS, 0, 4*12 );   // 4 sommets/quad * 12 quads
glDisableClientState( GL_VERTEX_ARRAY );
```

Toutefois, si on remplace `GL_QUADS` par `GL_QUAD_STRIP` dans l'appel à cette fonction, sans faire aucun autre changement, le résultat visuel obtenu n'est plus le même. Pourquoi ? **[1 point]**

d) Dans le cours, nous avons vu quelques modèles d'illumination qui permettent d'assigner des couleurs aux fragments d'une primitive. Nommez l'effet qui, avec un de ces modèles, peut exagérer de façon erronée la perception des variations d'intensité de la couleur aux frontières des primitives. **[1 point]**

e) Lorsqu'une source lumineuse d'OpenGL éclaire une surface, dans quelle direction est réfléchie la lumière diffuse ? **[1 point]**

f) Nommez la technique qui, à partir d'une texture principale, crée une série de textures de résolutions décroissantes afin d'améliorer le rendu. **[1 point]**

Question 6 Concepts théoriques [10 points]

a) Citez les avantages et les inconvénients des trois modèles d'affichage présentés durant le cours. [3 points]

b) Pourquoi est-il utile de séparer l'information géométrique (primitive graphique) du contexte graphique ? [1 point]

c) Justifiez l'intérêt de séparer un index en trois composantes dans la codification de la couleur RGB en mode indexé. [1 point]

d) Quelle est l'utilité d'une représentation en coordonnées homogènes des transformations géométriques ? [1 point]

e) Comment déterminer le nombre de points de fuite d'une projection en perspective à partir d'une connaissance à priori de la scène 3D et en examinant l'image obtenue ? [1 point]

f) Dans une projection en perspective, est ce que la silhouette de la projection d'une sphère est toujours un cercle ? [1 point]

g) Expliquez comment effectuer une sélection 3D en OpenGL. Indiquez toutes les étapes nécessaires, en portant une attention particulière à l'ordre des appels aux fonctions OpenGL ? [2 points]

Question 7 Concepts théoriques [10 points]

a) Décrivez brièvement les composantes d'une architecture MVC en interfaces usager graphiques [2 points]

b) Donnez l'algorithme de Cohen-Sutherland utilisé pour le découpage d'un segment par une région rectangulaire en 2D. Expliquez le système de codage utilisé. [2 points]

c) Une application a besoin d'afficher des images couleurs de résolution 512x512. Si on considère que 1024 couleurs simultanées sont suffisantes pour satisfaire les besoins de cette application quelles seraient les tailles minimales de la mémoire de trame (frame buffer) et de la table de couleurs, sachant que chacune des composantes R, G, et B est codée sur 8 bits (vous travaillez en mode RGB indexé) ? [2 points]

d) Dans une architecture client-serveur pour une application avec interface graphique, l'usager se retrouve nécessairement du coté client ou serveur ? [1 point]

e) Identifiez les deux matrices maintenues par OpenGL pour la visualisation 3D et dites laquelle de ces matrices la primitive gluPickMatrix() doit modifier pour réaliser une sélection 3D telle que vue en classe ? [1 point]

f) Décrivez les trois modèles d'illumination présentés durant le cours et citez les limites de chacun d'eux. [2 points]

Question 8 Concepts théoriques (Bresenham)

Considérez l'algorithme du point milieu pour tracer un cercle :

```
1   Fonction Cercle(rayon)
2      x = 0
3      y = rayon
4      h = 1 + rayon
5      AllumePixelEtSymétries(x, y)
6      Tant que y > x       // on reste dans le 2e octant
7         Si h < 0 alors    // Est
8            h = h + 2*(x-y) + 3
9         Sinon             // Sud-Est
10           h = h + 2*(x-y) + 5
11           y = y-1
12        Fin Si
13        x = x+1
14        AllumePixelsEtSymétries(x, y)
15     Fin Tant que
16  Fin Fonction
```

On veut adapter cet algorithme pour tracer une ellipse. L'équation implicite d'une ellipse centrée à l'origine est : $F(x, y) = b^2 * x^2 + a^2 * y^2 - a^2 * b^2 = 0$, a étant l'axe majeur et b l'axe mineur.

a) Indiquez quelles seraient alors les valeurs d'initialisation des variables x, y et h=d_0 (lignes 2, 3 et 4) dans le nouvel algorithme pour tracer une ellipse. Nous vous rappelons que, pour le cercle, le changement de variable $h_i = d_i$ - 0.25 a été effectué afin de travailler en arithmétique entière. Pour cette question, nous demandons uniquement l'évaluation de la variable d_0 dans le cas de l'ellipse. **[3 points]**

b) Évaluez l'incrément de la variable h, si on choisit de se déplacer vers l'Est après l'affichage du premier point (ligne 8). Il suffit d'évaluer la variable d_{i+1} présenté dans le cadre du cours. **[2 points]**

c) Évaluez l'incrément de la variable h, si on choisit de se déplacer vers le Sud-Est après l'affichage du premier point (ligne 10). **[2 points]**

d) Donnez les axes de symétrie ainsi que la stratégie à considérer pour tracer l'ellipse complète. Justifiez votre réponse. **[3 points]**

Question 9 Concepts théoriques

a) Dans le cadre de votre projet, vous avez utilisé un objet de la classe QGLWidget afin d'utiliser les fonctions graphiques d'OpenGL. (i) Identifiez les trois méthodes virtuelles de la classe QGLWidget que vous avez dû redéfinir dans la classe QGLWidget et (ii) décrivez les fonctionnalités qui sont implémentées dans chacune. **[3 points]**

b) Quelle est l'utilité d'une représentation en coordonnées homogènes dans les transformations géométriques ? **[2 points]**

c) En OpenGL, est-ce que les énoncés suivants définissent une source de lumière ponctuelle ou directionnelle ? **[1 point]**

```
GLfloat pos0[] = { 3.0, 2.0, 1.0, 0.0 };
glLightfv( GL_LIGHT0, GL_POSITION, pos0 );
```

Question 10 Concepts théoriques

a) Un carré dessiné avec un seul `GL_QUADS` est tel que deux sommets diagonalement opposés sont blancs et les deux autres sont noirs. Malheureusement le point central du carré n'est pas gris tel qu'on le souhaitait. Expliquez pourquoi et proposez une technique permettant que ce pixel soit affiché en gris ? **[2 points]**

b) On utilise `GL_QUADS` et 4 appels à `glVertex2f()` pour dessiner un carré. Si on remplace le `GL_QUADS` par un `GL_QUAD_STRIP` sans changer aucune autre ligne de code, le résultat obtenu n'est plus le même, pourquoi ? **[1 point]**

c) Lorsqu'on défini une lumière d'appoint, OpenGL ne permet pas de spécifier une valeur de `GL_SPOT_CUTOFF` supérieure à 90 degrés. Quel serait le problème, lors du calcul de l'intensité de la lumière d'appoint, si l'angle γ entre le vecteur de la lumière incidente et le vecteur de direction du *spot* était supérieur à 90 degrés ? **[1 point]**

d) Vous savez que la matrice suivante représente une transformation fenêtre-clôture :

$$\begin{bmatrix} 50 & 0 & 200 \\ 0 & 35 & 140 \\ 0 & 0 & 1 \end{bmatrix}$$

Croyez-vous que cette transformation respecte le rapport d'aspect ? (Indice, rappelez-vous la question 1.3 de votre devoir.) **[1 point]**

e) Montrez que l'inverse d'une matrice de rotation 3D par rapport à un axe quelconque passant par l'origine est égal à sa transposée. **[3 points]**

Question 11 Concepts théoriques

a) Durant le cours, nous avons présenté trois modèles d'affichage.

i) Identifiez ces trois modèles d'affichage

ii) Décrivez brièvement comment l'information est conservée dans chacun

iii) Donnez au moins un exemple d'utilisation appropriée pour chacun.

[3 points]

b) Durant le cours, nous avons aussi présenté le contexte graphique.

i) Quelle est l'utilité de définir un contexte graphique dans une application ?

ii) Donnez au moins un exemple où l'utilisation est clairement avantageuse.

[3 points]

c) Parmi les objets dessinés par une certaine application graphique, il y a un quelconque objet sphérique.

i) Si cette application utilise une projection orthograhique, est-ce que la silhouette de l'objet sphérique sera toujours un cercle ? Dites pourquoi.

ii) Si cette application utilise une projection perspective, est-ce que la silhouette de l'objet sphérique sera toujours un cercle ? Dites pourquoi.

[1 point]

Question 12 Notions de base

a) i) Pour les transformations géométriques 3D, pourquoi utilise-t-on une matrice 4X4 au lieu d'une 3X3 ? ii) Comment se nomme le système de coordonnées ainsi créé ? **[2 points]**

b) Si la matrice M ci-contre est utilisée comme matrice de transformation affine dans le pipeline graphique pour multiplier les sommets des primitives OpenGL, expliquez ce qui arrive si le terme $m16$ est nul ? **[2 points]**

$$M = \begin{bmatrix} m1 & m5 & m9 & m13 \\ m2 & m6 & m10 & m14 \\ m3 & m7 & m11 & m15 \\ m4 & m8 & m12 & m16 \end{bmatrix}$$

c) L'oeil humain est-il plus sensible aux variations d'intensité de la lumière ou aux variations de fréquence de la lumière ? Pourquoi ? **[2 points]**

d) Quelle est la différence entre un sommet et un fragment ? Comment sont créés les fragments ?
[3 points]

Question 13 Notions de base

a) En classe, nous avons vu l'algorithme du point milieu.

i) À quoi sert cet algorithme ?

ii) Au milieu de quoi est ce « point milieu » ?

[3 points]

b) Qu'est-ce qu'un anaglyphe ? **[2 points]**

c) L'oeil humain est-il plus sensible aux variations d'*intensité* de la lumière ou aux variations de *fréquence* de la lumière ? Pourquoi ? **[2 points]**

d) Nous avons vu quelques modèles de couleur en classe.

i) Identifiez *trois* modèles différents en nommant les axes du système de coordonnées de chaque modèle.

ii) Pour chacun, dessinez le *volume* représentant l'espace 3D des couleurs définies par le modèle en indiquant la position des axes.

[4 points]

Question 14 Concepts théoriques

a) Durant le cours, nous avons présenté trois modèles d'affichage.

i) Identifiez ces trois modèles d'affichage.

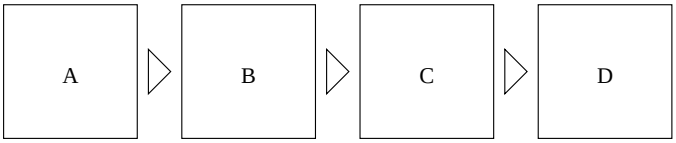
ii) Pour chacun, décrivez brièvement en quel format l'information graphique est conservée.
(Une image est décomposée en ...)

iii) Donnez un cas exemple où l'utilisation de chacun est appropriée.

[4 points]

b) Quelle est l'utilité d'une représentation en coordonnées homogènes des sommets dans les primitives graphiques ? **[2 points]**

c) Vous reconnaissez certainement le pipeline graphique d'OpenGL dans le schéma ci-dessous. Chaque case identifie une étape de transformation faite par OpenGL sur les sommets passent dans le pipeline.



i) Donnez les noms assignés à chaque étape (A-B-C-D) du pipeline.

ii) Nommez les 12 fonctions qui permettent de contrôler les transformations appliquées dans ce pipeline.

iii) Identifiez quelles fonctions s'appliquent à chaque étape du pipeline. (Notez que certaines fonctions peuvent s'appliquer à plus d'une étape !)

[6 points]

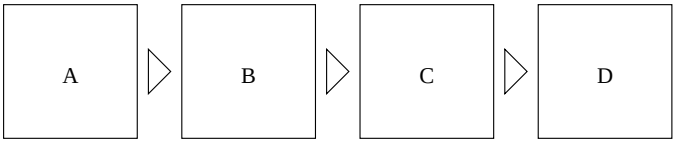
Question 15 Concepts théoriques

a) Dites à quoi sert chacune des fonctions OpenGL suivantes. Autant que possible, soyez bref et précis, tout en vous assurant de bien montrer l'utilité de la fonction. S'il y a lieu, faites référence aux arguments de cette fonction. Par exemple :
glPointSize ... spécifie le diamètre, en pixels, des points affichés à l'écran ;
glColor3f ... définit les nouvelles intensités de rouge, vert et bleu de la couleur courante.

i. glFrustum ...
ii. glGetFloatv ...
iii. glClear ...
iv. glIsEnabled ...
v. glMatrixMode ...
vi. glMultMatrixd ...
vii. glPolygonMode ...
viii. glPopMatrix ...
ix. glScalef ...
x. glPushAttrib ...
xi. gluLookAt ...
xii. glLoadIdentity ..
xiii. glVertex2d ...
xiv. gluOrtho2D ...
xv. glEnd ...

[15 points]

b) Vous reconnaissez le pipeline graphique d'OpenGL dans le schéma ci-dessous. Chaque case identifie une étape de transformation faite par OpenGL sur les sommets donnés au début du pipeline.



i) Donnez les noms assignés à chaque étape (A-B-C-D) du pipeline.

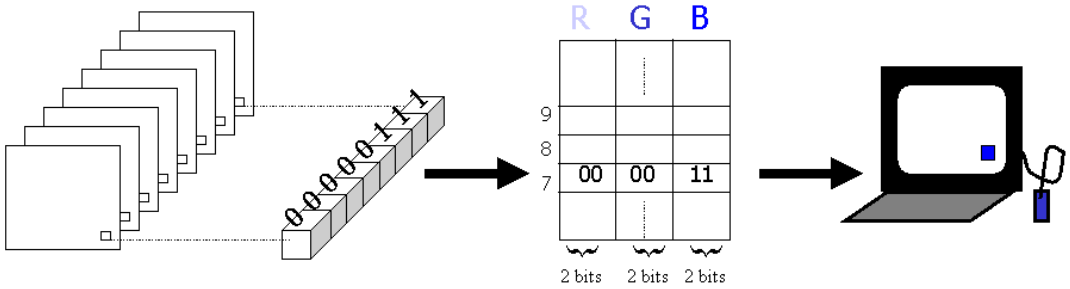
ii) Pour chacune des quatre étapes, nommez toutes les fonctions qui permettent de contrôler la transformation effectuée.

[6 points]

Question 16 Tables de couleurs [5 points]

a) Quelle est la différence entre le mode couleur direct et le mode couleur indexé ? OpenGL offre-t-il la possibilité d'utiliser les deux ? [2 points]

b) Supposons que vous disposez de la table de couleurs suivante, fonctionnant selon le monde indexé :



Expliquez pourquoi cette table de couleurs n'est pas utile en pratique. [3 points]

Question 17 Modèles de couleur

a) Combien de triplets du modèle HSV peuvent représenter chacune des couleurs suivantes ? **[3 points]**

rouge : _____ vert : _____ bleu : _____ noir : _____ blanc : _____

b) Donnez les coordonnées de la couleur magenta dans le modèle CMY et dans le modèle RGB. **[2 points]**

CMY : _____ RGB : _____

c) Dans une imprimante couleur, pourquoi y-a-t-il une cartouche d'encre noire en plus des cartouches d'encre cyan, d'encre magenta et d'encre jaune ? **[1 point]**

d) Quelle couleur est perçue par l'oeil lorsqu'on éclaire une feuille imprimée avec de l'encre jaune avec une lumière bleue ? Expliquez pourquoi. **[2 points]**

Question 18 Librairies graphiques et OpenGL

a) Nommez *deux* systèmes de librairies graphiques antérieurs à OpenGL. **[2 points]**
(Notez que Direct3D [1995] est plus récent qu'OpenGL [1992], ce n'est donc pas une réponse possible !)

b) Avec les librairies graphiques, on parle souvent de sommets, de fragments et de pixels.

i) Distinguez les concepts de sommet et de fragment.

ii) Comment sont créés les fragments ?

[3 points]

c) Une sphère pleine est tracée en OpenGL avec une projection perspective et elle est entièrement visible à l'écran.

i) Combien de point(s) de fuite peut-on facilement identifier s'il n'y a que cette sphère ? Pourquoi ?

ii) Quelle serait votre réponse si, au lieu d'une sphère, la scène contenait plutôt un cylindre ? Pourquoi ?

[2 points]

d) On affiche une scène 3D à l'écran en utilisation la fonction glFrustum d'OpenGL :

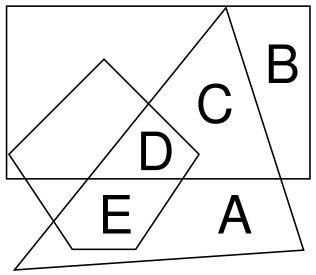
```
void glFrustum( GLdouble Gauche, GLdouble Droit, GLdouble Bas, GLdouble Haut,
                GLdouble PlanAvant, GLdouble PlanArrière );
```

i) Nommez deux conséquences d'une modification à la valeur du paramètre `PlanArrière` sur l'image vue à l'écran. **[1 point]**

ii) Nommez deux conséquences d'une modification à la valeur du paramètre `PlanAvant` sur l'image vue à l'écran. **[1 point]**

Question 19 Fragments

a) Trois primitives graphiques sont dessinées l'une à la suite de l'autre dans un programme utilisant les énoncés OpenGL ci-dessous pour tracer les primitives illustrées dans la figure ci-contre. La première primitive est un triangle, la seconde est un rectangle et la troisième est un pentagone. La fusion de couleurs OpenGL (`GL_BLEND`) étant activée, il faut déterminer, pour chaque région, quelle est la couleur source et quelle est la couleur destination afin de bien calculer la couleur résultante dans chaque région A, B, C, D, E dans le rendu final.




```
glPolygonMode( GL_FRONT_AND_BACK, GL_FILL );
glDisable( GL_DEPTH_TEST );
glEnable( GL_BLEND );
glBlendFunc( GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA ); // les valeurs de défaut!
glClearColor( 0.2, 0.2, 0.2, 1.0 );
glClear( GL_COLOR_BUFFER_BIT );
glColor4f( 0.7, 0.6, 0.6, 1.0 );
glCallList( leTriangle );  // la liste ne contient aucun appel à glColor()
glColor4f( 1.0, 0.2, 0.2, 0.5 );
glCallList( leRectangle );  // la liste ne contient aucun appel à glColor()
```

```
glCallList( lePentagone );  // la liste ne contient aucun appel à glColor()
```

i) Quelle sera la formule qui permet de calculer la couleur résultante selon ces énoncés ? **[2 points]**

Il faut utiliser la formule suivante pour la fusion de couleurs :
$RGBA_{final} = RGBA_{source} * A_{source} + RGBA_{destination} * (1 - A_{source})$

(Les valeurs de A doivent être calculées parce qu'elle pourrait être utilisées si on dessine autre chose par-dessus.)

ii) Lorsque la première primitive est tracée dans la région A, quelles sont les valeurs (R,G,B,A) de la couleur source, celles de la couleur destination et celles de la couleur résultante ? **[1 point]**

S : (0.7, 0.6, 0.6, 1.0) = la couleur du triangle
D : (0.2, 0.2, 0.2, 1.0) = la couleur du fond
*R : (0.7, 0.6, 0.6, 1.0) * (1.0) + (0.2, 0.2, 0.2, 1.0) * (1-1.0)*
*⇒ **(0.7, 0.6, 0.6, 1.0)***

iii) Lorsque la seconde primitive est tracée dans la région B quelles sont les valeurs (R,G,B,A) de la couleur source, celles de la couleur destination et celles de la couleur résultante ? **[2 points]**

S : (1.0, 0.2, 0.2, 0.5) = la couleur du rectangle
D : (0.2, 0.2, 0.2, 1.0) = la couleur du fond
*R : (1.0, 0.2, 0.2, 0.5) * (0.5) + (0.2, 0.2, 0.2, 1.0) * (1-0.5)*
= (0.5, 0.1, 0.1, 0.25) + (0.1, 0.1, 0.1, 0.5)
*⇒ **(0.6, 0.2, 0.2, 0.75)***

iv) Lorsque la seconde primitive est tracée dans la région C quelles sont les valeurs (R,G,B,A) de la couleur source, celles de la couleur destination et celles de la couleur résultante ? **[2 points]**

S : (1.0, 0.2, 0.2, 0.5) = la couleur du rectangle
D : (0.7, 0.6, 0.6, 1.0) = la couleur de A (ii)
*R : (1.0, 0.2, 0.2, 0.5) * (0.5) + (0.7, 0.6, 0.6, 1.0) * (1-0.5)*
= (0.5, 0.1, 0.1, 0.25) + (0.35, 0.3, 0.3, 0.5)
*⇒ **(0.85, 0.4, 0.4, 0.75)***

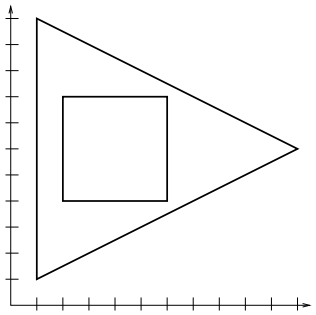
v) Lorsque la troisième primitive est tracée dans la région D quelles sont les valeurs (R,G,B,A) de la couleur source, celles de la couleur destination et celles de la couleur résultante ? **[2 points]**

S : (1.0, 0.2, 0.2, 0.5) = la couleur du pentagone
D : (0.85, 0.4, 0.4, 0.75) = la couleur de C (iv)
*R : (1.0, 0.2, 0.2, 0.5) * (0.5) + (0.85, 0.4, 0.4, 0.75) * (1-0.5)*
= (0.5, 0.1, 0.1, 0.25) + (0.425, 0.2, 0.2, 0.375)
*⇒ **(0.925, 0.3, 0.3, 0.625)***

vi) Lorsque la troisième primitive est tracée dans la région E quelles sont les valeurs (R,G,B,A) de la couleur source, celles de la couleur destination et celles de la couleur résultante ? **[1 point]**

S : (1.0, 0.2, 0.2, 0.5) = la couleur du pentagone
D : (0.7, 0.6, 0.6, 1.0) = la couleur de A (ii)
*R : (1.0, 0.2, 0.2, 0.5) * (0.5) + (0.7, 0.6, 0.6, 1.0) * (1-0.5)*
= (0.5, 0.1, 0.1, 0.25) + (0.35, 0.3, 0.3, 0.5)
*⇒ **(0.85, 0.4, 0.4, 0.75)***
La même couleur qu'en C (iv).

b) Les énoncés OpenGL suivants sont utilisés pour dessiner un quadrilatère plein et un triangle plein (voir ci-contre). La caméra est placée en (0, 0, 10) et regarde vers l'origine.



```
glEnable( GL_DEPTH_TEST );
glDepthFunc( GL_LESS ); // la valeur de défaut!
glMatrixMode( GL_PROJECTION );
glLoadIdentity( );
glOrtho( -6.0, 6.0, -6.0, 6.0, 0.0, 20.0 );
glMatrixMode( GL_MODELVIEW );
glLoadIdentity( );
gluLookAt( 0.0, 0.0, 10.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0 );
glBegin(GL_TRIANGLES);
  glColor3f( 0.0, 0.5, 1.0 );
  glVertex3f(  1.0,  1.0, 8.0 );
  glVertex3f( 11.0,  6.0, 8.0 );
  glVertex3f(  1.0, 11.0, 8.0 );
glEnd();
glBegin(GL_QUADS);
  glColor3f( 1.0, 0.5, 0.0 );
  glVertex3f(  2.0,  4.0, 3.0 );
  glVertex3f(  6.0,  4.0, 3.0 );
  glVertex3f(  6.0,  8.0, 3.0 );
  glVertex3f(  2.0,  8.0, 3.0 );
glEnd();
```

i) Quelle est la valeur de Z (entre 0.0 et 1.0) des fragments du triangle après la transformation de projection ? Expliquez votre réponse. **[1 point]**

*La caméra se trouve en (0, 0, 10) et le volume de visualisation est profond de 20 unités. Le plan avant est à 10 et le plan arrière est à -10. Après la transformation de projection, la profondeur en Z du triangle est de **(10-8) / 20 = 2 / 20 = 0.1**.*

ii) Quelle est la valeur de Z (entre 0.0 et 1.0) des fragments du quadrilatère après la transformation de projection ? Expliquez votre réponse. **[1 point]**

*Pour sa part, la profondeur en Z du quadrilatère est de **(10-3) / 20 = 7 / 20 = 0.35**.*

iii) Quelle est la comparaison faite par le test de profondeur qui est exécuté lors de l'affichage du quadrilatère ? Est-ce que le quadrilatère sera visible dans le rendu final ? **[1 point]**

*Le test est : **0.35 < 0.1** ? ⇒ NON... donc rejet du quadrilatère.*

Question 20 Courbes paramétriques polynomiales 2D [8 points]

On désire construire des courbes d'approximation en utilisant un certain nombre de points de contrôle.

a) Classez d'abord les types de courbes d'approximation de la plus générale à la moins générale : Bézier, NURBS, B-Spline. [1 point]

b) Donnez le vecteur nodal uniforme ouvert, paramétré de $t = 0$ à $t = 1$, d'une courbe B-Spline d'ordre 2 définie en utilisant cinq points de contrôle. Donnez le nombre de segments de droite qui composent alors cette courbe. [2 points]

c) Si on utilise quatre points de contrôle, donnez l'ordre maximal d'une courbe B-Spline si elle doit passer par le premier et le dernier point. Donnez son vecteur nodal pour une paramétrisation variant de $t = 0$ à $t = 1$. [2 points]

d) Pour une courbe B-Spline d'ordre 3, donnez l'intervalle de validité du paramètre de la courbe si le vecteur nodal est (0 .1 .2 .4 .4 .5 .7 1). [1 point]

e) Donnez l'ordre de la courbe de Bézier pouvant être construite avec cinq points de contrôle. [1 point]

f) Quel est le type de courbes d'approximation le plus approprié pour tracer un quart d'ellipse. Justifiez votre réponse. [1 point]

Question 21 Courbes paramétriques polynomiales [9 points]

a) Soit le point en coordonnées homogènes (8, 6, 4, 2). Quelles sont ses coordonnées cartésiennes (euclidiennes) ? [1 point]

b) Que doit être la dimension de chaque point de contrôle pour représenter une courbe NURBS en 3D ? [1 point]

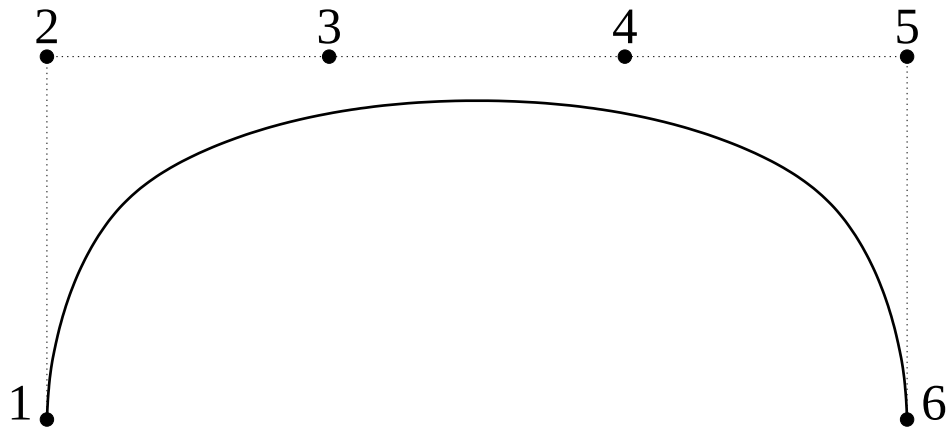
c) Quelle(s) restriction(s) doit-on imposer aux données d'une NURBS pour que la courbe devienne une B-Spline ? [2 points]

d) Quelle(s) restriction(s) doit-on imposer aux données d'une B-Spline pour que la courbe devienne une courbe de Bézier ? [2 points]

e) Combien faut-il de points de contrôle pour définir une courbe de Bézier de degré 5 (ordre 6) ? [1 point]

f) Combien de valeurs doivent être présentes dans le vecteur nodal pour construire une B-Spline d'ordre k=4 avec 8 points de contrôle ? [1 point]

g) Voici une B-Spline d'ordre k=6 utilisant 6 points de contrôle. Donnez son vecteur nodal, avec une *paramétrisation variant de* $t = 0$ *à* $t = 1$. [1 point]



Question 22 Courbes paramétriques polynomiales [12 points]

Spline cubique

Pour construire un segment d'une spline cubique tel que vu en cours, il est nécessaire de fournir deux points $\vec{P_1}$ et $\vec{P_2}$, de même que deux vecteurs $\vec{V_1}$ et $\vec{V_2}$. Si le paramètre t varie entre 0 et 1, l'équation paramétrique de la courbe et les polynômes d'Hermite de degré 3 utilisés sont alors :

$$\vec{P}(t) = [H_1(t)H_2(t)H_3(t)H_4(t)] \begin{bmatrix} \vec{P_1} \\ \vec{P_2} \\ \vec{V_1} \\ \vec{V_2} \end{bmatrix} \qquad \begin{array}{ll} H_1(t) = 2t^3 - 3t^2 + 1 & H_1(0.5) = +1/2 \\ H_2(t) = -2t^3 + 3t^2 & H_2(0.5) = +1/2 \\ H_3(t) = t^3 - 2t^2 + t & H_3(0.5) = +1/8 \\ H_4(t) = t^3 - t^2 & H_4(0.5) = -1/8 \end{array}$$

a) (i) Expliquez la différence entre une spline d'interpolation (collocation) et une spline d'approximation et (ii) dites de quel type est la spline cubique décrite ci-dessus. [1 point]

b) Si $\vec{P_1} = (3, 0)$, $\vec{P_2} = (6, 3)$, $\vec{V_1} = \vec{V_2} = (1, 1)$, (i) dessinez la courbe et (ii) donnez la valeur de $\vec{P}(0)$, $\vec{P}(0.5)$ et $\vec{P}(1)$. [1 point]

c) Avec les mêmes valeurs de $\vec{P}_1$ et $\vec{P}_2$, quelle est la valeur de $\vec{P}(0.5)$ si $\vec{V}_1 = (1,1)$ et $\vec{V}_2 = (5,5)$? Quelle est la valeur de $\vec{P}(0.5)$ si $\vec{V}_1 = (1,1)$ et $\vec{V}_2 = (9,9)$? Nous avons vu en cours que les deux vecteurs définissent les tangentes de la courbe aux extrémités. En observant la position de $\vec{P}(0.5)$, quelle autre information géométrique définissent-ils aussi ? [1 point]

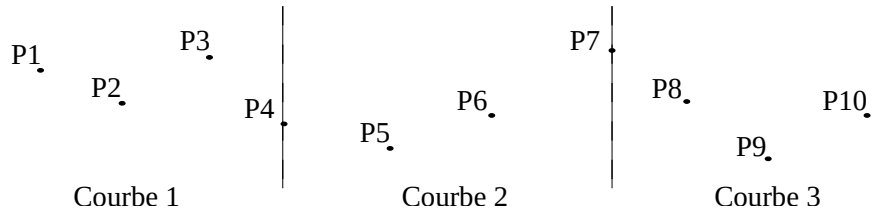
Bézier

Une spline cubique nécessite l'utilisation de deux vecteurs $\vec{V}_i$ pour construire la courbe. La forme mathématique d'une courbe de Bézier est très semblable à celui d'une spline cubique, mais n'utilise pourtant seulement que l'information de ses points de contrôle $\vec{P}_i$.

d) (i) Dites si une courbe de Bézier est une spline d'interpolation (collocation) ou une spline d'approximation et (ii) expliquez pourquoi une courbe de Bézier, contrairement à une spline cubique, n'a pas besoin de la définition de vecteurs $\vec{V}_i$. [1 point]

e) Quel est le degré d'une courbe de Bézier construite à l'aide de 10 points de contrôle ?
(ordre = degré + 1) [1 point]

Jugeant que le degré de la courbe de Bézier formée avec 10 points est trop élevé, nous désirons la séparer en une séquence de 3 (sous-)courbes utilisant chacune 4 points de contrôle consécutifs : $\vec{P}_1$ à $\vec{P}_4$ pour la courbe 1, $\vec{P}_4$ à $\vec{P}_7$ pour la courbe 2, $\vec{P}_7$ à $\vec{P}_{10}$ pour la courbe 3, tel qu'illustré ci-dessous.



f) Quel sera le degré de chacune de ces 3 courbes ? La nouvelle courbe ainsi formée par cette séquence de 3 courbes aura-t-elle la même forme que la courbe originale ? Pourquoi ? [1 point]

g) Quels que soient les points $\vec{P}_1 ... \vec{P}_{10}$ utilisés, est-il certain que les segments de courbe se joignent ? Pourquoi ? [1 point]

h) Si les points $\vec{P}_1 ... \vec{P}_{10}$ sont tels que les courbes se joignent, est-il certain que les dérivées premières sont continues aux points de jonction ? Si oui, dites pourquoi, sinon dites dans quel cas particulier nous avons la continuité des dérivés premières. [1 point]

i) Outre la réduction de degré, dites quel autre gain géométrique dans la manipulation de la courbe nous avons obtenu ? (Observez, par exemple, ce qui se passe si l'on bouge $\vec{P}_2$ avant et après la segmentation.) [1 point]

B-Spline + NURBS

j) Pour les trois différents cas suivants, dites quel est le problème dans la définition du vecteur nodal u de la B-Spline, sachant que n représente le nombre de points de contrôle de la courbe et que k représente l'ordre de la courbe. (ordre = degré + 1) [1 point]

$u = [0, 0, 0, 1, 2, 7, 7, 7]$, $n = 8$, $k = 4$
$u = [0, 0, 0, 1, 3, 2, 3, 3, 3]$, $n = 6$, $k = 3$
$u = [0, 0, 0, 0, 1, 2, 4, 5, 5, 5, 5]$, uniforme, ouvert

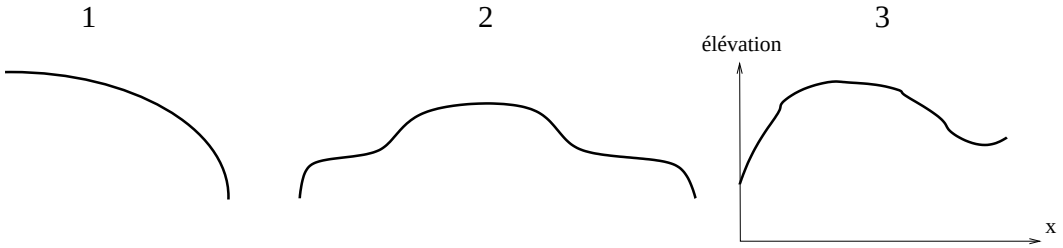
k) Étant donné 3 points de contrôle (n=3), donnez le vecteur nodal u de la B-Spline pour obtenir une courbe de Bézier ayant comme intervalle de validité [0,1]. [1 point]

l) Pour construire une NURBS, il est nécessaire d'ajouter une coordonnée homogène aux points de contrôle. Dans quel cas la NURBS ainsi obtenue est-elle une B-Spline ? [1 point]

Question 23 Courbes paramétriques polynomiales [10 points]

a) Parmi les types de splines vus en classe (spline cubique, B-Spline, Bézier, NURBS), donnez le type le plus approprié de courbe (et justifier votre réponse) pour tracer les formes suivantes : [6 points]

1. un quart d'ellipse ?
2. une courbe approximant le profil 2D de la carrosserie d'une automobile ?
3. une courbe reliant des mesures d'élévation de terrain acquises avec très bonne précision ?



b) Donnez l'ordre k de la courbe de Bézier pouvant être construite avec 12 points de contrôle. [1 point]

c) Donnez le vecteur nodal uniforme ouvert, paramétré de t = 0.0 à t = 1.0, d'une courbe B-Spline d'ordre k = 2 définie en utilisant 5 points de contrôle. Donnez aussi le nombre de segments qui composent alors cette courbe. [3 points]

Question 24 Courbes paramétriques polynomiales

Pour approximer une forme quelconque (par exemple un bonhomme de neige !) dans un logiciel de dessin en 2D, une courbe B-Spline d'ordre k = 3 est construite en utilisant n = 6 points de contrôle et un vecteur nodal uniforme ouvert.

a) Donnez le nombre de valeurs dans le vecteur nodal. **[1 point]**

b) Donnez son vecteur nodal, en utilisant une *paramétrisation variant de* $t = 0$ *à* $t = 1$. **[1 point]**

c) Si on utilise plutôt une B-Spline d'ordre k = 6 avec les mêmes points de contrôle, donnez son vecteur nodal ouvert, en utilisant une *paramétrisation uniforme variant de* $t = 0$ *à* $t = 1$. **[1 point]**

d) Si on utilisait plutôt une courbe de Bézier construite avec ces mêmes 6 points de contrôle, quel serait alors le degré de cette courbe ? **[1 point]**

Question 25 Courbes paramétriques polynomiales

Pour les questions suivantes, on veut simuler la trajectoire d'un objet en utilisant des splines. Pour chaque question, on vous demande :

i) D'indiquer le type de spline le plus simple que vous pouvez utiliser. Vous ne devez utiliser qu'une seule courbe par trajectoire (pas de courbes mises bout à bout). La liste donnée plus bas indique les choix possibles, du plus simple au plus complexe.

ii) De faire un dessin indiquant approximativement la position des points de contrôle. Si vous répétez des points de contrôle, indiquez le clairement.

iii) En fonction du type de spline choisi, vous devez aussi fournir les informations supplémentaires indiquées :
- Courbe de Bézier :
  Donnez le polynôme multipliant chaque point de contrôle.
- B-Spline :
  Donnez l'ordre de la spline. (ordre = degré + 1)
  Donnez un exemple de vecteur de valeur nodale valide.
- NURBS :
  Donnez l'ordre de la spline. (ordre = degré + 1)
  Donnez un exemple de vecteur de valeur nodale valide.
  Pour chaque point de contrôle, dites si la coordonnée homogène est inférieure à 1, égale à 1 ou supérieure à 1.

a) Un objet attaché par une corde décrit un quart de cercle. **[3 points]**

b) Un javelot est lancé avec un angle de 45^o par rapport à l'horizontale. Le javelot suit une trajectoire parabolique et plante dans le sol à son arrivée. **[3 points]**

c) Une balle de tennis est lancée horizontalement à partir du toit d'un édifice. La balle rebondit une fois sur le sol avant de terminer sa course dans un lac. Comme on considère le frottement de l'air, la trajectoire est une courbe polynomiale de degré 3. **[3 points]**

Question 26 Courbes paramétriques polynomiales

a) Une NURBS et une B-Spline sont deux types de splines, l'une des deux étant plus générale que l'autre. i) Dites laquelle des deux est la plus générale et ii) identifiez sous quelles conditions la plus générale devient une spline de l'autre type. **[2 points]**

b) Une Bézier et une NURBS sont deux types de splines, l'une des deux étant plus générale que l'autre. i) Dites laquelle des deux est la plus générale et ii) identifiez sous quelles conditions la plus générale devient une spline de l'autre type. **[2 points]**

c) Quelle(s) restriction(s) doit-on imposer aux points de contrôle d'une B-Spline afin que cette courbe 2D soit toujours contenue dans un carré $-1 < x < 1$ et $-1 < y < 1$? Pourquoi ? **[2 points]**

d) Les splines sont toutes définies par la relation $\sum_{i=1}^{n} N_{i,k}(t)G_i$ où la forme des fonctions de base diffère d'une spline à l'autre. Associez le nom de la spline à ses fonctions de base. **[3 points]**

1) B-Spline
2) Bézier
3) NURBS
4) Spline cubique

A) $N_{i,k=4}(t) = H_i(t)$ où $H_1(t) = 2t^3 - 3t^2 + 1$
$H_2(t) = -2t^3 + 3t^2$
$H_3(t) = t^3 - 2t^2 + t$
$H_4(t) = t^3 - t^2$

B) $N_{i,k=n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$

C) $N_{i,1}(t) = 1$ si $u_i \leq t \leq u_{i+1}$, 0 sinon
$N_{i,k}(t) = \frac{t-u_i}{u_{i+k-1}-u_i} N_{i,k-1}(t) + \frac{u_{i+k}-t}{u_{i+k}-u_{i+1}} N_{i+1,k-1}(t)$

D) $N^*_{i,1}(t) = 1$ si $u_i \leq t \leq u_{i+1}$, 0 sinon
$N^*_{i,k}(t) = \frac{t-u_i}{u_{i+k-1}-u_i} N^*_{i,k-1}(t) + \frac{u_{i+k}-t}{u_{i+k}-u_{i+1}} N^*_{i+1,k-1}(t)$
$N_{i,k}(t) = \frac{h_i N^*_{i,k}(t)}{\sum_{j=1}^{n} h_j N^*_{j,k}(t)}$

Question 27 Courbes paramétriques polynomiales

a) Quel est le degré maximal d'une B-Spline contenant 12 points de contrôle ? **[1 point]**

b) Soit une B-Spline avec $n = k = 6$ et construite en utilisant un vecteur nodal ouvert et normalisé, donnez les équations des 6 fonctions de base $N_{1,6}(t), N_{2,6}(t), \ldots, N_{6,6}(t)$ de cette B-Spline. (Indice : vous n'avez pas besoin d'utiliser la formule de Cox-DeBoor.) **[2 points]**

c) Écrivez les vecteurs nodeaux de B-Spline correspondant aux descriptions suivantes :

- Un vecteur nodal uniforme et normalisé pour une B-Spline d'ordre 3 contenant 6 points de contrôle et passant par le premier et le dernier point de contrôle. **[1 point]**

- Un vecteur nodal uniforme périodique pour une B-Spline d'ordre 4 contenant 7 points de contrôle et dont la plage de validité du paramètre t va de -1 à 1. **[1 point]**

- Un vecteur nodal périodique uniforme normalisé pour une B-Spline d'ordre 5 formant une boucle. La boucle est représentée par 11 points de contrôle **différents** (les répétitions potentielles ne sont pas comptées dans ces 11 points). **[1 point]**

- Un vecteur nodal ouvert et normalisé pour une B-Spline linéaire contenant 6 points de contrôle. Cette B-Spline doit servir de trajectoire pour un objet qui se déplace à peu près à vitesse constante. Le point de contrôle i se trouve à la position $\vec{P}_i$. **[1 point]**

d) Soit une B-Spline avec $k = 2$ composée de deux points de contrôle $\vec{P}_1 = (0, 5)$ et $\vec{P}_2 = (5, 0)$ et utilisant le vecteur nodal $[0, 1, 2, 3]$. Utilisez la formulation de Cox-DeBoor pour évaluer $N_{1,2}(0.5)$ et $N_{2,2}(0.5)$. Évaluez la B-Spline en $t = 0.5$. Tracez la courbe qui serait obtenue si on dessinait la B-Spline en faisant varier t de 0 à 3. **[3 points]**

Question 28 Courbes paramétriques polynomiales

a) Nous avons vu quelques catégories de courbes en classe dont : Bézier, NURBS et B-Spline. Parmi ces trois, quelle est celle qui est la plus générale ? Quelle est celle qui est la moins générale ? **[1 point]**

b) Soit une *B-Spline* avec $n = k = 5$ et construite en utilisant un vecteur nodal ouvert et normalisé. Cette courbe s'exprime par la relation $C(t) = \sum_{i=1}^{n} N_{i,k}(t)G_i$. Écrivez les équations des cinq fonctions de base $N_{1,5}(t), N_{2,5}(t), \ldots, N_{5,5}(t)$ de cette B-Spline. **[2 points]**

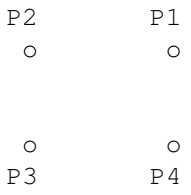
c) Écrivez les vecteurs nodaux correspondant aux descriptions suivantes :

i) Un vecteur nodal uniforme et normalisé pour une B-Spline d'ordre 3 contenant 6 points de contrôle et passant par son premier et son dernier point de contrôle. **[2 points]**

ii) Un vecteur nodal uniforme et périodique pour une B-Spline d'ordre 4 contenant 7 points de contrôle et dont la plage de validité du paramètre t va de -1 à 1. **[2 points]**

iii) Un vecteur nodal ouvert et normalisé pour une B-Spline linéaire (ordre 2) contenant 6 points de contrôle. Cette B-Spline servira de trajectoire pour un objet qui doit se déplacer à peu près à vitesse constante. Le point de contrôle i se trouve à la position $\vec{P}_i$. ($l_i = |\vec{P}_i - \vec{P}_{i-1}|$ et $L_i = \sum_{j=2}^{i} l_j$.) **[2 points]**

d) On définit une courbe NURBS en utilisant le vecteur nodal [0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45], en utilisant successivement les points de contrôle `P1`, `P2`, `P3`, `P4`, `P1`, `P2` et en posant une coordonnée homogène égale à 1 à chaque `Pi`.



Indiquez l'ordre maximal de la courbe NURBS pouvant être construite ainsi **et** tracez approximativement cette courbe. **[2 points]**

Question 29 Courbes paramétriques polynomiales

a) Nous avons vu quelques catégories de courbes en classe dont les NURBS et splines cubiques. Identifiez deux différences fondamentales entre ces deux types de courbes. **[2 points]**

b) Nous avons aussi vu en classe les courbes de Bézier.

i) Une courbe de Bézier est-elle une spline d'interpolation (collocation) ou une spline d'approximation ?

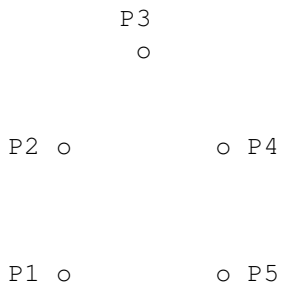
ii) Expliquez pourquoi une courbe de Bézier, contrairement à une spline cubique, n'a pas besoin de la définition de vecteurs $\vec{V}_i$.

[2 points]

c) Soit une B-Spline d'ordre $k = n = 5$ et construite en utilisant un vecteur nodal ouvert et normalisé.

i) Écrivez son vecteur nodal.

ii) Tracez approximativement la courbe si on utilise, dans l'ordre, ces $n = 5$ points de contrôle :



[2 points]

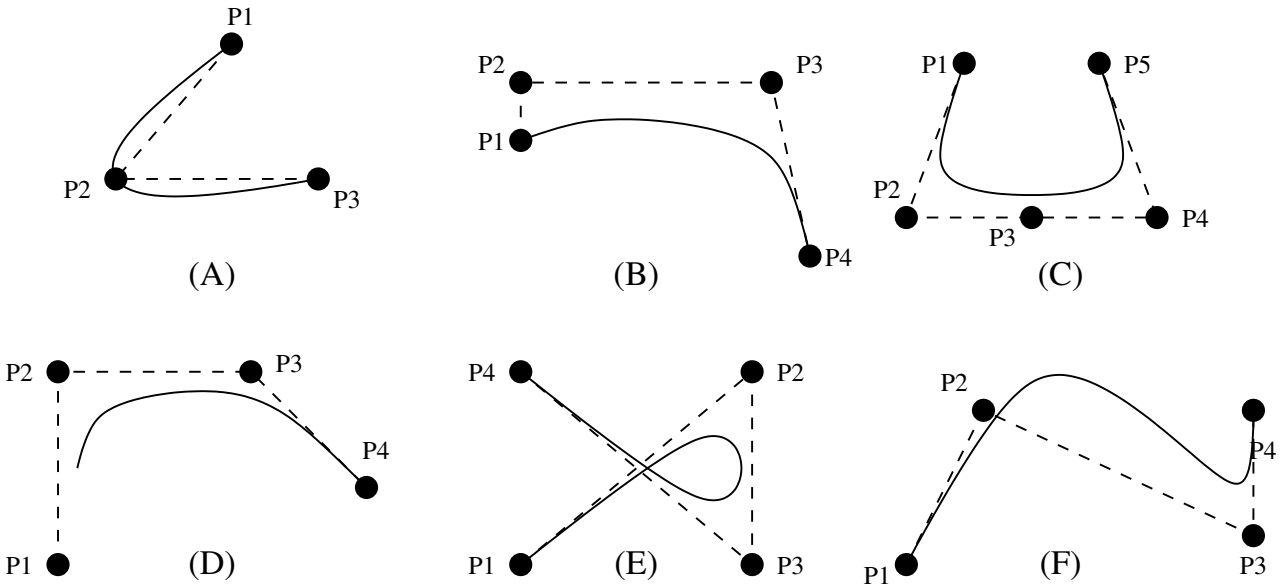
d) Quelle(s) restriction(s) doit-on imposer aux points de contrôle d'une B-Spline en 2D afin qu'elle soit toujours contenue dans un carré $-1 \leq x \leq 1$ et $-1 \leq y \leq 1$? Justifiez votre réponse. **[2 points]**

Question 30 Courbes paramétriques polynomiales

a) Nous avons vu quelques catégories de courbes en classe dont les splines cubiques et les courbes de Bézier. Identifiez *deux* différences fondamentales entre ces deux types de courbes. **[3 points]**

b) Afin de modéliser la trajectoire parabolique d'une boulette de papier que vous lancez au panier de recyclage, vous décidez d'utiliser une B-Spline d'ordre k = 3 avec 7 points de contrôle et un vecteur nodal *uniforme ouvert* afin que la courbe passe par le premier point et le dernier point.
Écrivez le vecteur nodal de cette B-Spline, en faisant en sorte que le paramètre t varie de 0 à 1. Expliquez succinctement comment vous arrivez à votre réponse. **[3 points]**

c) Considérez ces six courbes polynomiales et leurs points de contrôle :

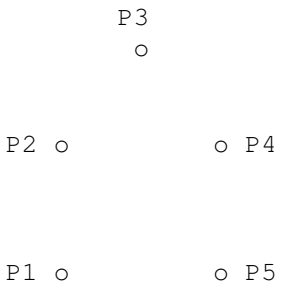


Pour chaque courbe, dites si *oui* ou *non* elle peut être une B-Spline. Si vous dites que la courbe n'est pas une B-Spline, *justifiez votre réponse* en donnant la raison à la base de votre affirmation.

[4 points]

Question 31 Courbes paramétriques polynomiales 2D [8 points]

On désire construire une courbe en utilisant, dans l'ordre, les cinq points de contrôle illustrés ci-dessous.



a) Donnez l'ordre minimal et l'ordre maximal d'une courbe de Bézier pouvant être construite avec ces points de contrôle. [2 points]

5 et 5.

b) Donnez l'ordre minimal d'une B-Spline si elle doit passer par le premier et le dernier point. Donnez son vecteur nodal pour une paramétrisation uniforme variant de t=0 à t=1. [1 point]

2, [0 0 .25 .5 .75 1 1]

c) Donnez l'ordre maximal d'une B-Spline si elle doit passer par le premier et le dernier point. Donnez son vecteur nodal pour une paramétrisation uniforme variant de t=0 à t=1. [1 point]

5, [0 0 0 0 0 1 1 1 1 1]

d) Pour une B-Spline d'ordre 3 à paramétrisation non uniforme, déterminez quels points de contrôle sont des points de collocation si on utilise le vecteur nodal (0 0 1 1 3 4 4 4) [2 points]

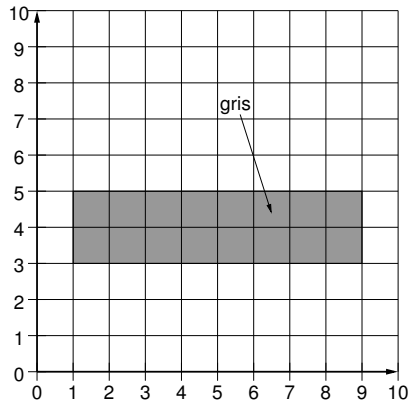
Les points 2 et 5. (La paramétrisation de la courbe est non uniforme, mais la courbe passe par ces points.)

e) Si on utilise successivement les points de contrôle P1, P2, P3, P4, P5 et à nouveau P1, avec le vecteur nodal (.1 .2 .3 .4 .5 .6 .7 .8), indiquez l'ordre maximal de la courbe pouvant être créée et tracez approximativement cette courbe. [2 points]

Ordre 2. La courbe décrit le contour d'une maisonnette en passant par chacun des points de contrôle.

Question 32 Opérations sur les fragments

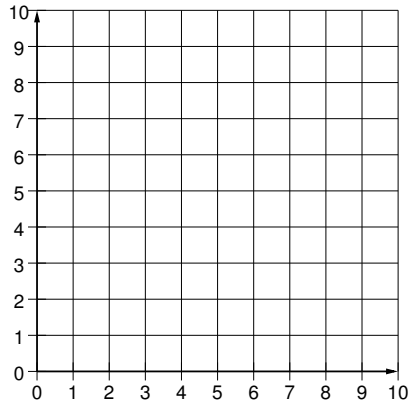
On vous demande de dessiner le contenu de la mémoire de trame et du stencil aux différents jalons identifiés (a), (b), (c), (d) et (e) dans le programme OpenGL présenté à l'annexe A. (L'effet des énoncés est évidemment cumulatif.) Ainsi, à la ligne notée « EXEMPLE ... », la mémoire de trame contient ceci :



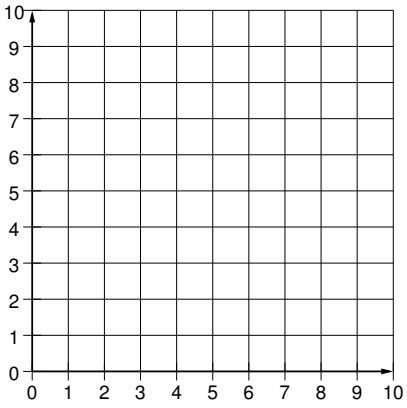
mémoire de trame à la ligne EXEMPLE

- L'affichage est de bonne résolution et on peut donc considérer les trames comme des régions continues.
- Dans vos réponses pour la mémoire de trame, assurez-vous d'indiquer clairement la couleur de chaque région.
- Dans vos réponses pour le stencil, inscrivez la valeur stockée dans le stencil pour chaque région.
- À moins d'avis contraire, le *brouillon* tout à droite est un brouillon qui ne sera pas corrigé.

a) Dessinez le contenu de la mémoire de trame. **[1 point]**

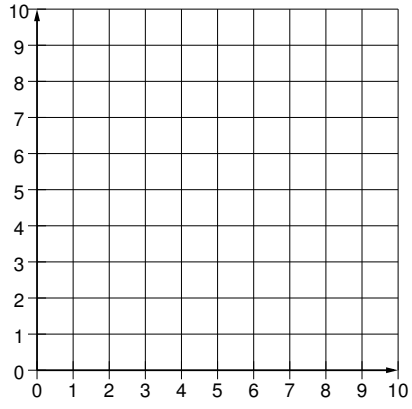


mémoire de trame à ligne (**a)**

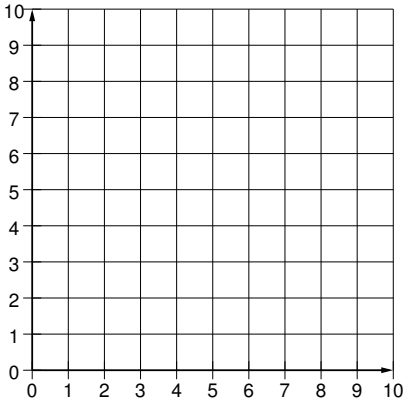


(brouillon)

b) Dessinez le contenu de la mémoire de trame. **[3 points]**

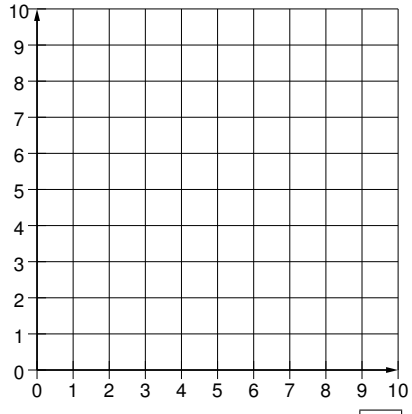


mémoire de trame à ligne (**b)**

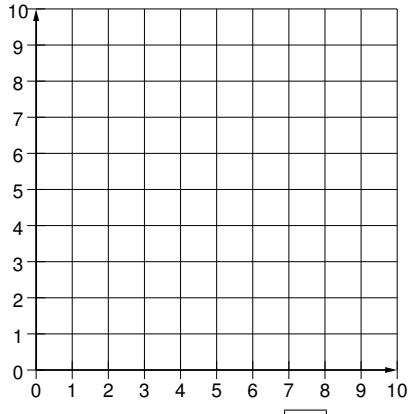


(brouillon)

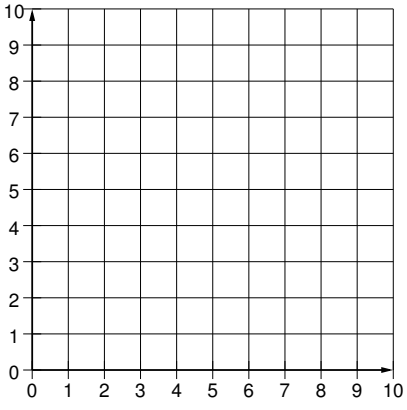
c) Dessinez le contenu de la mémoire de trame et le contenu du stencil. **[3 points]**



mémoire de trame à ligne (**c)**

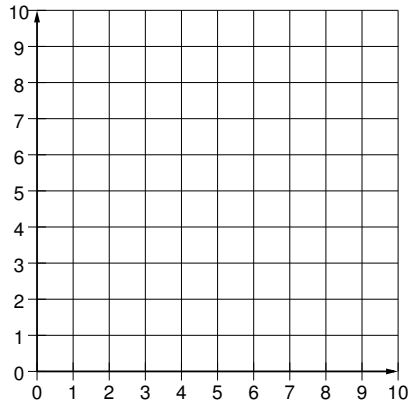


stencil à ligne (**c)**

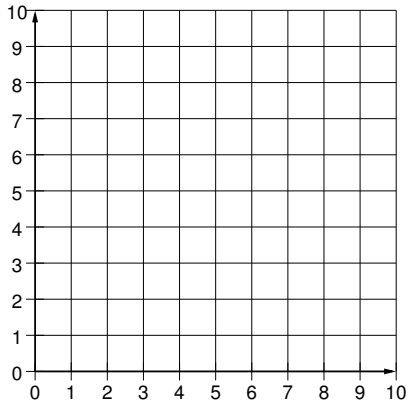


(brouillon)

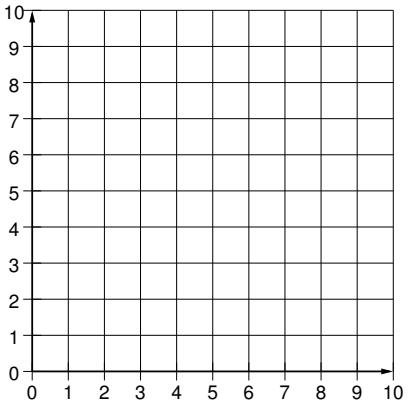
d) Dessinez le contenu de la mémoire de trame et le contenu du stencil. **[3 points]**



mémoire de trame à ligne (**d)**

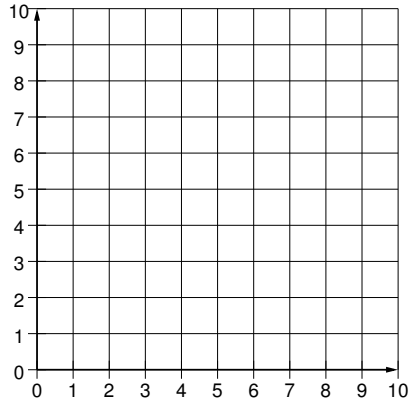


stencil à ligne (**d)**

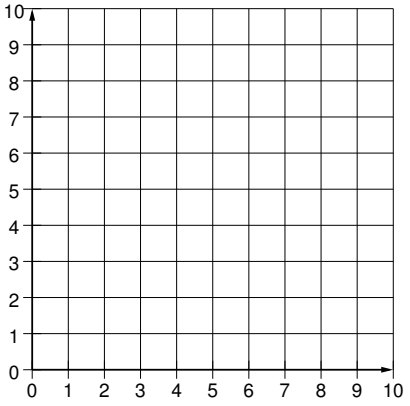


(brouillon)

e) Dessinez le contenu de la mémoire de trame. **[3 points]**



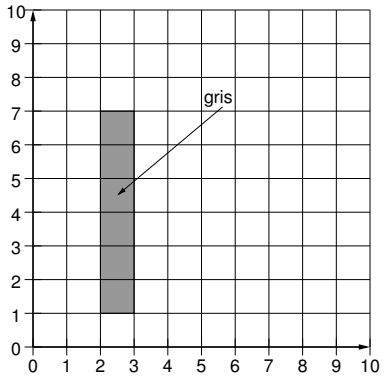
mémoire de trame à ligne (**e)**



(brouillon)

Question 33 Opérations sur les fragments

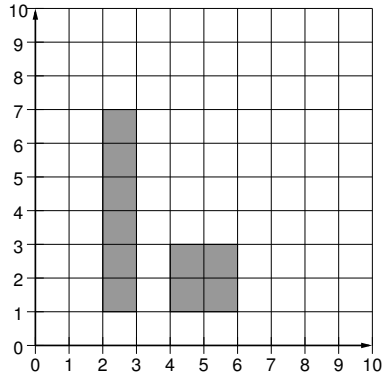
Considérez la méthode OpenGL présentée à l'annexe A. On vous demande ici dessiner le contenu de la mémoire de trame et du stencil aux différents jalons identifiés (a), (b), (c), (d) et (e) dans cette méthode. Par exemple, à la ligne notée « EXEMPLE ... », la mémoire de trame contient ceci :



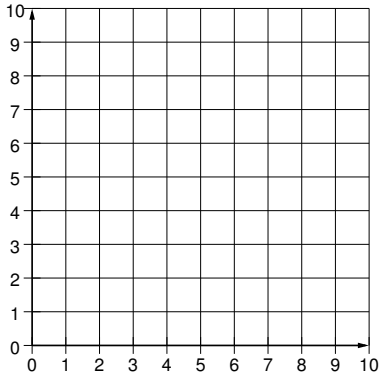
mémoire de trame à la ligne EXEMPLE

Notes :
- L'affichage est de bonne résolution et on peut donc considérer les trames comme des régions continues.
- Dans vos réponses pour la mémoire de trame, assurez-vous d'indiquer la couleur de chaque région.
- Dans vos réponses pour le stencil, inscrivez la valeur stockée dans le stencil pour chaque région.
- À moins d'avis contraire, le *brouillon* à droite est un brouillon qui ne sera pas corrigé.

a) Dessinez le contenu de la mémoire de trame. **[2 points]**

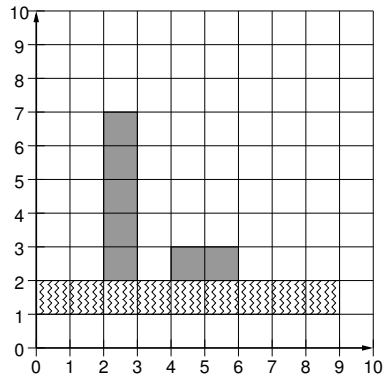


mémoire de trame à ligne (**a)**

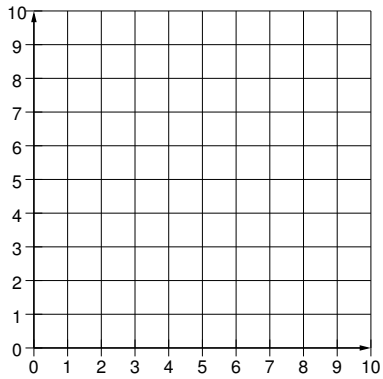


(brouillon)

b) Dessinez le contenu de la mémoire de trame. **[2 points]**

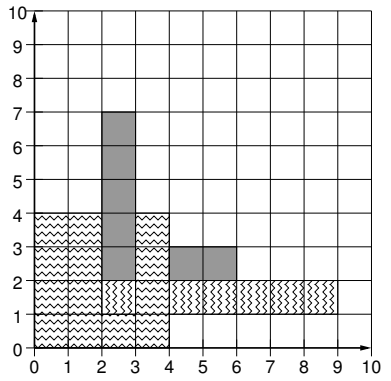


mémoire de trame à ligne (**b)**

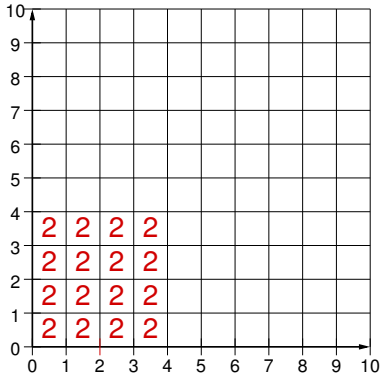


(brouillon)

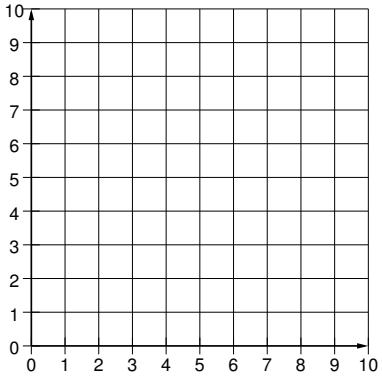
c) Dessinez le contenu de la mémoire de trame et le contenu du stencil. **[3 points]**



mémoire de trame à ligne (**c)**

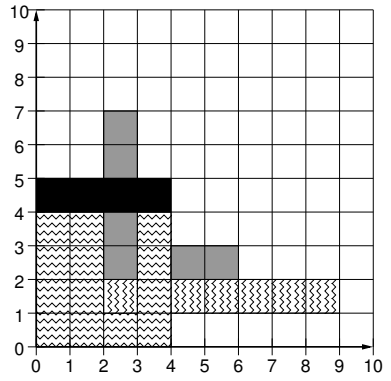


stencil à ligne (**c)**

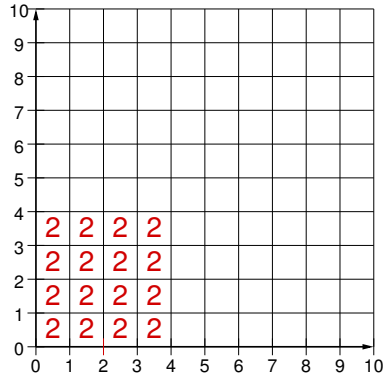


(brouillon)

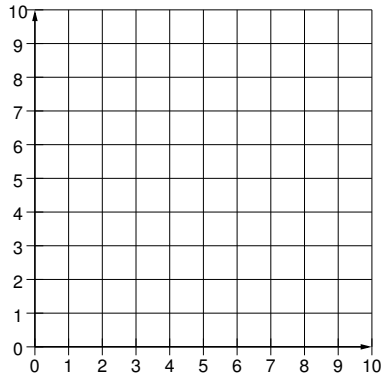
d) Dessinez le contenu de la mémoire de trame et le contenu du stencil. **[3 points]**



mémoire de trame à ligne (**d)**

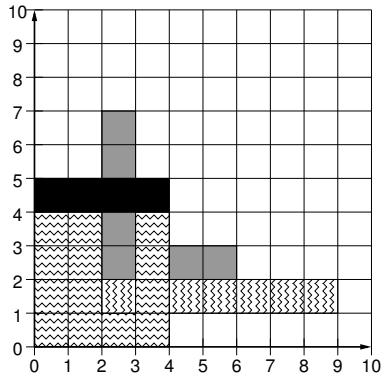


stencil à ligne (**d)**

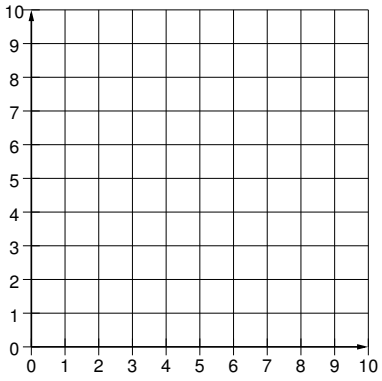


(brouillon)

e) Dessinez le contenu de la mémoire de trame. **[3 points]**



mémoire de trame à ligne (**e)**

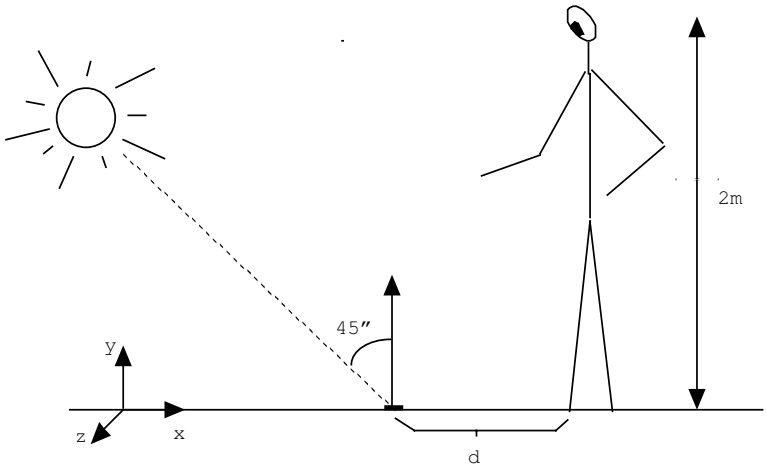


(brouillon)

Question 34 Modèle d'illumination [12 points]

Soit un ingénieur cyclope marchant dans la rue (aujourd'hui, tout est possible). L'oeil de ce cyclope est situé à 2 mètres du sol. À un moment donné, son regard est attiré par un reflet brillant sur le sol. Il s'agit d'une pièce de 10 cents abandonnée qui luit au soleil ! (La pièce est parfaitement spéculaire c.à.d. son coefficient spéculaire est de 1 et on considère aussi que le soleil est une source de lumière située à l'infini)

a) En supposant que cette scène soit éclairée par le soleil dont les rayons font un angle de 45 degrés avec le sol à cette heure-là de la journée, à quelle distance au sol d de la pièce pouvait bien se trouver le cyclope pour que son oeil ait pu intercepter ce reflet ? (On suppose que la taille de la pièce est négligeable par rapport au reste et que l'on peut donc l'approximer par un point.) [2 points]



b) Si l'on néglige tout facteur d'atténuation et que l'on suppose que le soleil ce jour-là ait une intensité Is, quelle est l'intensité spéculaire perçue par le cyclope ? Si le matériau de la pièce est gris et que le soleil émet une couleur rouge (!), quelle est la couleur du reflet perçu ? [3 points]

c) On suppose que le sol se trouve à x = 0, que l'axe des y pointe vers le haut et que la profondeur p (coordonnée z) de la pièce et du cyclope soit la même. La pièce se trouve à la position (4, 0, p) dans le repère du monde ainsi défini. Si vous vouliez modéliser la position de l'oeil du cyclope à l'aide du modèle de caméra synthétique vu en classe, quels seraient les paramètres d'orientation de la caméra (la position de l'observateur, la normale au plan de projection et le vecteur up) ? [3 points]

d) Quelle commande OpenGL utiliseriez-vous pour spécifier ce référentiel et quels arguments lui donneriez vous (l'ordre des paramètres n'a pas d'importance mais vous devez les identifier correctement) ? Quelle matrice de OpenGL est affectée par cette commande ? [3 points]

e) Pourquoi les acteurs se poudrent-ils le visage avant d'entrer en scène ? (reliez vos explications au contenu du cours, plus spécifiquement aux propriétés de matériau). [1 point]

Question 35 Modèles d'illumination [12 points]

a) Quelle est la différence entre le modèle de Lambert et celui de Gouraud ? Quelles sont les fonctions OpenGL pour activer chaque mode ? [2 points]

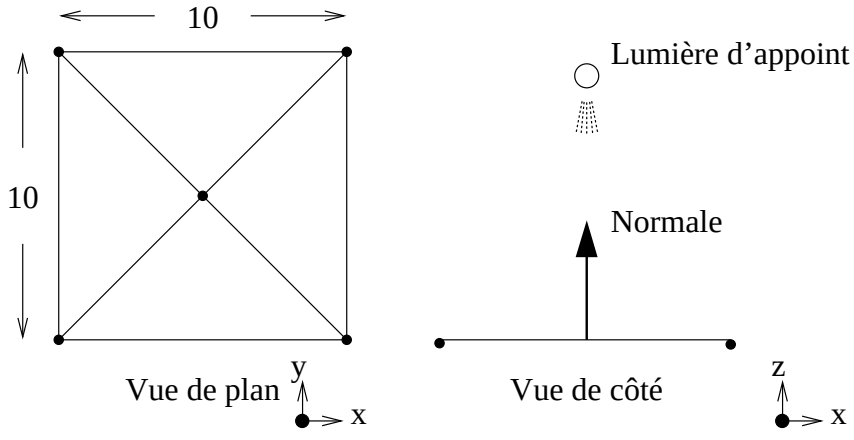
Supposons que la position d'un point d'un polygone se situe à $(1, 0, 0)$ et que la normale définie en ce point soit le vecteur $(0, 1, 0)$. De plus, nous disposons de deux lumières de couleur blanche $(I_r = I_g = I_b = 1)$, la première étant à $(1, 10, 0)$ et la seconde étant à $(11, 0, 0)$. La composante diffuse de la propriété de matériau du polygone est $K_d = (0.2, 0.5, 0.2)$. À moins d'avis contraire, on suppose aussi qu'il n'y a pas d'atténuation.

b) Donnez les composantes (r,g,b) de la couleur calculée en ce point par OpenGL due seulement aux composantes diffuses des lumières. [2 points]

c) Si la deuxième lumière est maintenant positionnée à $(11, 10, 0)$, quelle sera cette couleur ? [3 points]

d) Toujours avec la deuxième lumière en $(11, 10, 0)$, mais avec un facteur d'atténuation constant de 2, quelle sera cette couleur ? [2 points]

Soit un polygone carré de 10 x 10 unités dans le plan xy, formé de 4 triangles qui partagent un sommet au centre du carré. Soit aussi une lumière d'appoint ("spot") de couleur blanche $(I_r = I_g = I_b = 1)$ située le long de la normale à ce polygone, au-dessus du centre du carré, à une position $(0, 0, z)$ et qui éclaire dans une direction perpendiculaire à la surface du polygone. L'angle d'ouverture du cône de lumière de cette lumière d'appoint est tel que la forme de ce cône sur la surface est un cercle de rayon 1.



e) Décrivez ce qui est observé dans l'affichage par OpenGL en vue de plan si la composante diffuse de la propriété du matériau est $K_d = (1, 1, 1)$ à chacun des points du polygone et que le modèle de Gouraud est utilisé. [3 points]

Question 36 Calcul d'illumination

Votre application OpenGL utilise présentement des appels à `glColor3f` pour colorer les triangles qui composent la scène. Vous souhaitez plutôt utiliser le modèle d'illumination fourni par OpenGL.

Vous souhaitez créer une source lumineuse directionelle de couleur blanche et dont les intensités diffuse et spéculaire sont de 0.7. Le vecteur pointant **vers** cette source de lumière est $(1, 1, 1)$. Vous désirez aussi avoir une source de lumière ambiante globale blanche d'intensité 0.3.

Pour ne pas trop vous compliquer la vie, vous comptez remplacer tous les appels à `glColor3f( R, V, B )` par des appels à la fonction `materielDeCouleur( R, V, B )`. Cette dernière fonction, que vous comptez écrire, devra configurer un matériel non-spéculaire dont les coefficients diffus sont 10 fois plus grands que les coefficients ambiants. De plus, les coefficients diffus doivent correspondre aux paramètres reçus par la fonction.

a) Écrivez le code OpenGL configurant la source lumineuse directionelle et la source de lumière ambiante globale. **[3 points]**

b) Écrivez la fonction `materielDeCouleur( R, V, B )`. **[3 points]**

c) Quelle information supplémentaire, qui n'était pas requise lorsqu'on utilisait `glColor3f`, doit maintenant être spécifiée pour chaque sommet ? Quelle fonction OpenGL permet de spécifier cette information ? **[1 point]**

d) La scène obtenue manque beaucoup de réalisme car toutes les facettes triangulaires des objets sont distinctement visibles, ceci même pour les objets courbes. Vous avez pourtant pris soin de représenter ces derniers avec un très grand nombre de triangles. D'après-vous, d'où peut venir ce problème ? Comment le réglez-vous ? **[1 point]**

e) Si on configure un matériel avec l'appel `materielDeCouleur( 0.3, 0, 0.3 )`, quelle sera la couleur calculée par OpenGL en un sommet si la normale à ce sommet est de $(0, 1, 0)$? **[3 points]**

f) Pourquoi, à la question précédente, n'a-t-on pas besoin de la position du sommet ? Pourquoi n'a-t-on pas besoin de la position de l'observateur ? **[1 point]**

Question 37 Modèles d'illumination

a) Dites quelle est la couleur perçue en un point par un observateur dans chacun des cas suivants. Expliquez votre réponse. (Notez que : jaune = rouge + vert ; magenta = rouge + bleu ; cyan = vert + bleu.)

i) une lumière magenta éclaire un point d'une surface blanche.

ii) une lumière jaune éclaire un point d'une surface verte.

iii) une lumière bleue éclaire un point d'une surface jaune.

iv) une lumière cyan éclaire un point d'une surface magenta.

[4 points]

b) Le modèle couramment utilisé en infographie pour calculer la *réflexion de la lumière en un point* définit trois types distincts de réflexion, ce qui donne lieu à des termes différents dans l'équation d'illumination par une source de lumière. Identifiez et distinguez chacun des trois types de réflexion (nom + courte description). **[4 points]**

c) Nous avons vu trois *modèles d'illumination* utilisés en infographie pour calculer la couleur des pixels d'une facette d'un objet affiché à l'écran. L'un d'eux est le modèle de Gouraud.

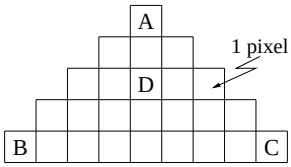
i) Nommez les deux autres.

ii) Différenciez les trois modèles.

iii) Dites comment on active chacun des trois modèles avec la librairie OpenGL.

[4 points]

d) Supposez que le modèle de réflexion de la lumière en un point a permis de calculer, dans le petit triangle ci-contre, que la couleur au pixel A est $(1.0, 1.0, 1.0)$, celle au pixel B est $(0.8, 0.5, 0.4)$ et celle au pixel C est $(0.6, 0.9, 0.6)$. Si on utilise le modèle d'illumination de Gouraud pour calculer la couleur des autres pixels du triangle, quelle couleur sera calculée au pixel D ? Expliquez clairement vos calculs. **[3 points]**



e) Considérez les énoncés OpenGL suivants qui définissent quatre sources de lumière. Pour chacune, dites si c'est une source de lumière ponctuelle ou directionnelle. Justifiez votre réponse. **[2 points]**

```
GLfloat lumpos1[] = { 3.0, 2.0, 1.0, 0.0 };
GLfloat lumpos2[] = { 3.0, 2.0, 1.0, 0.5 };
GLfloat lumpos3[] = { 6.0, 4.0, 2.0, 0.0 };
GLfloat lumpos4[] = { 6.0, 4.0, 2.0, 1.0 };
glLightfv( GL_LIGHT1, GL_POSITION, lumpos1 );
glLightfv( GL_LIGHT2, GL_POSITION, lumpos2 );
glLightfv( GL_LIGHT3, GL_POSITION, lumpos3 );
glLightfv( GL_LIGHT4, GL_POSITION, lumpos4 );
```

f) Dans les quatre définitions de sources de lumière à la sous-question précédente, est-ce que certaines définitions sont équivalentes ? Justifiez votre réponse. **[1 point]**

Question 38 Illumination

a) Nous avons vu en classe le modèle de *réflexion de la lumière en un point* qui est le plus couramment utilisé en infographie. Ce modèle définit trois types distincts de réflexion, donnant lieu à différents termes

dans l'équation d'illumination par une source lumineuse. (Pour les sous-questions suivantes traitant du calcul de la réflexion de la lumière en un point, supposez une seule source de lumière monochrome et sans atténuation.)

i) Qui a mis au point ce modèle qui sépare les réflexions en trois types ? **[1 point]**

ii) Écrivez l'expression mathématique servant au calcul du terme *ambiant*. Dites la signification de chaque variable de cette expression. **[1 point]**

iii) Écrivez l'expression servant au calcul du terme *diffus*. Dites la signification de chaque variable de cette expression. **[2 points]**

iv) Écrivez l'expression servant au calcul du terme *spéculaire*. Dites la signification de chaque variable de cette expression. **[2 points]**

v) Quel(s) terme(s) (*ambiant*, *diffus*, *spéculaire*) dépend(ent) de la position de l'observateur ? **[1 point]**

vi) Quel(s) terme(s) (*ambiant*, *diffus*, *spéculaire*) dépend(ent) de la position de la source lumineuse ? **[1 point]**

b) On peut calculer la couleur moyenne d'une image en faisant une moyenne des couleurs de tous les pixels qui composent cette image :

$$coul_{moy} = \sum_{i=1}^{larg} \sum_{j=1}^{haut} coul_{i,j} \; / \; (larg * haut)$$

Supposons maintenant une scène rendue à l'écran où les divers objets de la scène remplissent la fenêtre et occupent tous les pixels de la clôture. Si on diminue de moitié les coefficients de réflexion diffuse et de réflexion spéculaire des propriétés de matériau des objets et qu'on ne change rien d'autre, quel est l'impact sur la nouvelle couleur moyenne de l'image et comment peut-on le quantifier ? Justifiez votre réponse en donnant, s'il y a lieu, les conditions nécessaires pour que cette conclusion tienne *dans tous les cas*. **[3 points]**

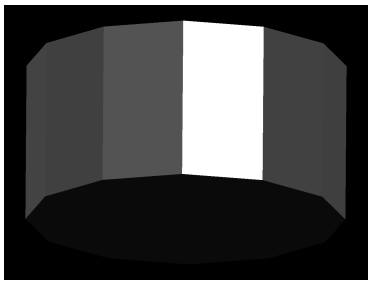
c) Nous avons vu qu'on peut définir des lumières directionnelles ou des lumières positionnelles.

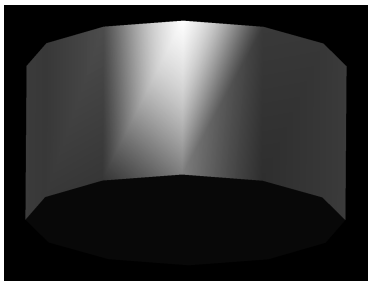
i) Quelle est la différence entre ces deux types de source de lumière ?

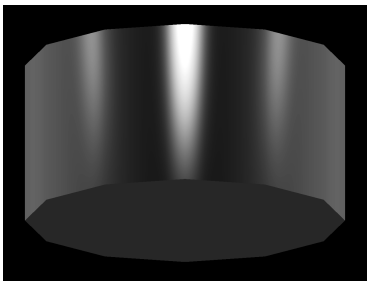
ii) En regardant les énoncés OpenGL et leurs paramètres lors de la définition d'une source de lumière, comment peut-on savoir de quel type de source de lumière il s'agit ?

[2 points]

d) Nous avons vu trois modèles d'illumination utilisés en infographie pour calculer la couleur des pixels d'une facette d'un même objet. Pour chaque sous-question suivante, donnez le *nom* du modèle d'illumination qui s'accorde le mieux avec l'image ou la description donnée. **[6 points]**
(+0.5 point par bonne réponse ; 0 si aucune réponse ; -0.25 point par réponse erronée.)

i) 

ii) 

iii) 

iv) utilise une normale par fragment.

v) utilise une normale par face.

vi) interpole les couleurs.

vii) interpole les normales.

viii) demande le moins de calcul pour colorer toute une facette.

ix) peut produire des bandes de Mach.

x) peut produire des discontinuités de la couleur entre deux faces contiguës.

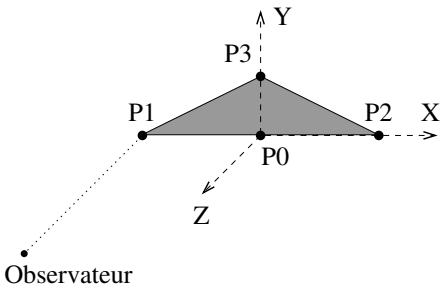
xi) est la meilleure méthode qui peut être supportée dans un nuanceur de *sommets*.

xii) est la meilleure méthode qui peut être supportée dans un nuanceur de *fragments*.

Question 39 Illumination

a) Le modèle de réflexion de la lumière en un point que nous avons vu en classe est celui communément utilisé en infographie. Il définit trois types distincts de réflexion (ambiante, diffuse, spéculaire) qui sont

calculés par trois termes qui s'additionnent dans l'équation proposée par le modèle de réflexion afin de déterminer la valeur de la réflexion en un point.



Considérez le triangle plein ci-contre, défini dans le plan XY, par les sommets P1, P2 et P3. Ce triangle est affiché dans une scène définie avec ces paramètres :

- Les sommets du triangle sont situés à P1 $= (-20, 0, 0)$, P2 $= (20, 0, 0)$ et P3 $= (0, 10, 0)$.
- Partout sur ce triangle, le vecteur normal est $\vec{N} = (0, 0, 1)$.
- L'observateur O voit tout le triangle et est positionné à $(-20, 0, 40)$.
- L'unique source lumineuse ponctuelle qui éclaire la scène est aussi positionnée à $(-20, 0, 40)$.
- La source lumineuse ponctuelle produit de la lumière ambiante, diffuse et spéculaire. Ses propriétés sont : $I_a = (1, 1, 1, 1)$, $I_d = (1, 1, 1, 1)$, $I_s = (1, 1, 1, 1)$.
- Le matériau utilisé pour afficher une certaine primitive ne réfléchit pas la lumière ambiante, mais réfléchit la lumière diffuse et la lumière spéculaire, n'est pas très brillant et n'émet aucune autre lumière. Ses propriétés sont : $k_a = (0, 0, 0, 0)$, $k_d = (1, 1, 1, 1)$, $k_s = (1, 1, 1, 1)$, $n_{brillance} = 2$.

Considérant ce qui précède, on vous demande de déterminer l'intensité de la lumière diffuse ou spéculaire à certains points. Pour chaque sous-question qui suit, **1)** faites un schéma montrant *tous les vecteurs utiles* au calcul à ce point, **2)** écrivez les valeurs *normalisées* de ces vecteurs et **3)** calculez l'intensité ...

i) ... de la lumière *diffuse* réfléchie au point P1.

ii) ... de la lumière *diffuse* réfléchie au point P2.

iii) ... de la lumière *spéculaire* selon *Phong* réfléchie au point P1.

iv) ... de la lumière *spéculaire* selon *Blinn* réfléchie au point P1.

v) ... de la lumière *spéculaire* selon *Phong* réfléchie au point P2.

vi) ... de la lumière *spéculaire* selon *Blinn* réfléchie au point P2.

vii) ... de la lumière *spéculaire* selon *Phong* réfléchie au point P0 situé au milieu entre P1 et P2.

viii) ... de la lumière *spéculaire* selon *Blinn* réfléchie au point P0 situé au milieu entre P1 et P2.

[12 points]

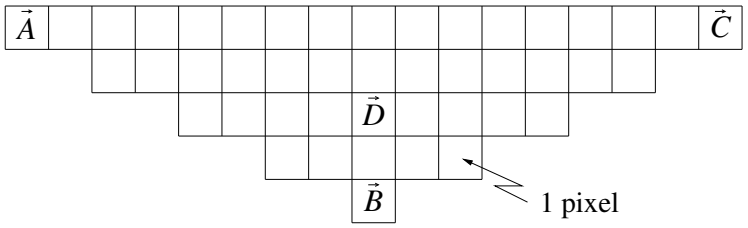
b) Nous avons vu que le modèle d'illumination d'OpenGL permet de définir des lumières directionnelles et des lumières positionnelles.

i) Quelle est la différence entre ces deux types de source de lumière ?

ii) En regardant les énoncés OpenGL et leurs paramètres lors de la définition d'une source de lumière, comment peut-on savoir de quel type de source de lumière il s'agit ?

[2 points]

c) Supposez que le modèle de réflexion de la lumière d'OpenGL en un point a permis de calculer, pour le triangle ci-dessous, les couleurs *rgba* pour les trois sommets correspondant aux pixels des extrémités du triangle ci-dessous :



- le vecteur $\vec{A}$ représente la couleur au sommet $\vec{P_A}$,
- le vecteur $\vec{B}$ représente la couleur au sommet $\vec{P_B}$,
- le vecteur $\vec{C}$ représente la couleur au sommet $\vec{P_C}$,

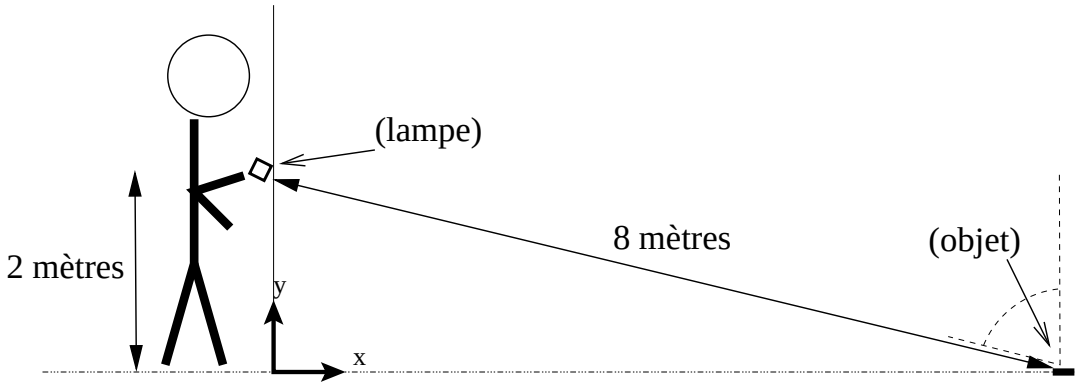
(Les couleurs sont des « `vec4` ».)

Si le modèle d'illumination défini en OpenGL par `glShadeModel(GL_SMOOTH)` est utilisé lors du tramage de la primitive, quelle est l'expression qui permet de calculer la couleur $\vec{D}$ au sommet $\vec{P_D}$ en fonction des couleurs $\vec{A}$, $\vec{B}$, $\vec{C}$? **[3 points]**

Question 40 Modèle d'illumination

Par une nuit glaciale et totalement noire, un ingénieur est dans une maison non éclairée (mais chauffée !). Avec une lampe de poche, il éclaire l'extérieur au travers d'un carreau tout propre d'une haute fenêtre située à 2 m du sol et voit, à 8 m de lui, un petit objet carré, plat et opaque, dont la taille est d'environ 1cm x 1cm. La lampe émet seulement de la lumière diffuse, de couleur plutôt jaunâtre, mais réussit tout de même à éclairer cet objet en totalité. Avant de s'aventurer à l'extérieur, l'ingénieur souhaite savoir avec quel matériau est fabriqué cet objet.

Si on considère que le modèle d'illumination d'OpenGL est celui qui régit le monde et en négligeant tout facteur d'atténuation, la scène se décrit ainsi :




```
// nuit totalement noire:
GLfloat  modAmbi[]  = { 0.0, 0.0, 0.0, 1.0 };
glLightModelfv( GL_LIGHT_MODEL_AMBIENT, modAmbi );
// lampe faiblissante:
GLfloat  lampAmbi[] = { 0.0, 0.0, 0.0, 1.0 };
GLfloat  lampDiff[] = { 0.6, 0.6, 0.2, 1.0 };
GLfloat  lampSpec[] = { 0.0, 0.0, 0.0, 1.0 };
glLightfv( GL_LIGHT0, GL_AMBIENT, lampAmbi );
glLightfv( GL_LIGHT0, GL_DIFFUSE, lampDiff );
glLightfv( GL_LIGHT0, GL_SPECULAR, lampSpec );
// matériau inconnu:
GLfloat  matAmbi[] = { ???, ???, ???, 1.0 };      // propriété 1
GLfloat  matDiff[] = { ???, ???, ???, 1.0 };      // propriété 2
GLfloat  matSpec[] = { ???, ???, ???, 1.0 };      // propriété 3
GLfloat  matEmis[] = { ???, ???, ???, 1.0 };      // propriété 4
GLfloat  matExpo   = ???;                         // propriété 5
glMaterialfv( GL_FRONT_AND_BACK, GL_AMBIENT, matAmbi );
glMaterialfv( GL_FRONT_AND_BACK, GL_DIFFUSE, matDiff );
glMaterialfv( GL_FRONT_AND_BACK, GL_SPECULAR, matSpec );
glMaterialfv( GL_FRONT_AND_BACK, GL_EMISSION, matEmis );
glMaterialf( GL_FRONT_AND_BACK, GL_EXPONENT, matExpo );
```

Les propriétés recherchées sont :

```
GLfloat matAmbi[] = { 0.1, 0.2, 0.1, 1.0 };
GLfloat matDiff[] = { 0.4, 0.8, 0.6, 1.0 };
GLfloat matSpec[] = { 0.2, 0.2, 0.2, 1.0 };
GLfloat matEmis[] = { 0.0, 0.1, 0.0, 1.0 };
GLfloat matExpo ≥ 100.0;
```

a) D'abord, il éteint sa lampe et constate que l'objet émet une faible lumière verdâtre d'intensité (0.0, 0.1, 0.0, 1.0). En allumant à nouveau sa lampe, il constate que la couleur qu'il perçoit alors est (0.06, 0.22, 0.03, 1.0). De plus, il est convaincu que c'est un objet très brillant. Mais il sait toutefois qu'il ne pourra pas déterminer ainsi toutes les propriétés du matériau. Dites à quelles trois propriétés du matériau inconnu il peut déjà donner des valeurs **et** donnez ces valeurs. **[6 points]**

matExpo ≥ 100.0

matEmis = (0.0, 0.1, 0.0, 1.0)

*couleur = matEmis + modAmbi + lampAmbi * matAmbi + $(L \cdot N)$ * lampDiff * matDiff*
*couleur = matEmis + $(L \cdot N)$ * lampDiff * matDiff*

*matDiff = (couleur - matEmis) / ($(L \cdot N)$ * lampDiff)*
*matDiff = ((0.06, 0.22, 0.03, 1.0) - (0.0, 0.1, 0.0, 1.0)) / ((2m/8m) * (0.6, 0.6, 0.2, 1.0))*
*matDiff = ((0.06, 0.12, 0.03, 1.0)) / (0.25 * (0.6, 0.6, 0.2, 1.0))*
matDiff = (0.06, 0.12, 0.03, 1.0) / (0.15, 0.15, 0.05, 1.0)
matDiff = (0.4, 0.8, 0.6, 1.0)

b) Après quelques minutes, le ciel s'éclaircit et une magnifique pleine lune apparaît directement au-dessus de la scène. La lune émet une lumière bleutée et l'ingénieur connaît les paramètres de cette nouvelle source de lumière (voir les paramètres de la lune ci-dessous). Avec sa lampe allumée, il constate que la couleur perçue du petit objet, maintenant éclairé par sa lampe et la lune, est maintenant (0.27, 0.64, 0.52, 1.0). En utilisant les résultats précédents et cette nouvelle observation, il sait alors qu'il peut déterminer une quatrième propriété du matériau ! Dites à quelle propriété du matériau inconnu il peut donner des valeurs **et** donnez ces valeurs. **[6 points]**

```
// lune bleutée:
GLfloat  luneAmbi[] = { 0.1, 0.1, 0.1, 1.0 };
GLfloat  luneDiff[] = { 0.5, 0.5, 0.8, 1.0 };
GLfloat  luneSpec[] = { 0.0, 0.0, 0.0, 1.0 };
glLightfv( GL_LIGHT1, GL_AMBIENT, luneAmbi );
glLightfv( GL_LIGHT1, GL_DIFFUSE, luneDiff );
glLightfv( GL_LIGHT1, GL_SPECULAR, luneSpec );
```

*couleur = matEmis + modAmbi + (lampAmbi + luneAmbi) * matAmbi + (($L_{lamp} \cdot N$) * lampDiff + ($L_{lune} \cdot N$) * luneDiff) * matDiff*
*couleur = matEmis + luneAmbi * matAmbi + (($L_{lamp} \cdot N$) * lampDiff + ($L_{lune} \cdot N$) * luneDiff) * matDiff*
*matAmbi = (couleur - matEmis - (($L_{lamp} \cdot N$) * lampDiff + ($L_{lune} \cdot N$) * luneDiff) * matDiff) / luneAmbi*
*matAmbi = (couleur - matEmis - ((2m/8m) * lampDiff + 1.0 * luneDiff) * matDiff) / luneAmbi*
*matAmbi = (couleur - matEmis - (0.25 * lampDiff + luneDiff) * matDiff) / luneAmbi*
*matAmbi = ((0.27, 0.54, 0.52, 1.0) - (0.25 * (0.6, 0.6, 0.2, 1.0) + (0.5, 0.5, 0.8, 1.0)) * (0.4, 0.8, 0.6, 1.0)) / (0.1, 0.1, 0.1, 1.0)*
*matAmbi = ((0.27, 0.54, 0.52, 1.0) - ((0.15, 0.15, 0.05, 1.0) + (0.5, 0.5, 0.8, 1.0)) * (0.4, 0.8, 0.6, 1.0)) / (0.1, 0.1, 0.1, 1.0)*
*matAmbi = ((0.27, 0.54, 0.52, 1.0) - (0.65, 0.65, 0.85, 1.0) * (0.4, 0.8, 0.6, 1.0)) / (0.1, 0.1, 0.1, 1.0)*
matAmbi = ((0.27, 0.54, 0.52, 1.0) - (0.26, 0.52, 0.51, 1.0)) / (0.1, 0.1, 0.1, 1.0)
matAmbi = (0.01, 0.02, 0.01, 1.0) / (0.1, 0.1, 0.1, 1.0)
matAmbi = (0.1, 0.2, 0.1, 1.0)

ou encore

*luneAmbi * matAmbi + ($L_{lune} \cdot N$) * luneDiff * matDiff = (0.27, 0.64, 0.52, 1.0) - (0.06, 0.22, 0.03, 1.0)*

*luneAmbi * matAmbi + luneDiff * matDiff = (0.21, 0.42, 0.49, 1.0)*

*matAmbi = ((0.21, 0.42, 0.49, 1.0) - luneDiff * matDiff) / luneAmbi*

*matAmbi = ((0.21, 0.42, 0.49, 1.0) - (0.5, 0.5, 0.8, 1.0) * (0.4, 0.8, 0.6, 1.0)) / (0.1, 0.1, 0.1, 1.0)*

matAmbi = ((0.21, 0.42, 0.49, 1.0) - (0.2, 0.4, 0.48, 1.0)) / (0.1, 0.1, 0.1, 1.0)

matAmbi = (0.01, 0.02, 0.01, 1.0) / (0.1, 0.1, 0.1, 1.0)

matAmbi = (0.1, 0.2, 0.1, 1.0)

c) Notre ingénieur déductif connaît maintenant toutes les propriétés du matériau sauf une. La chance étant avec lui, la lune se cache à nouveau, mais surtout sa collègue revient d'une promenade extérieure. La source de lumière qu'elle porte au front, plus forte (voir ses paramètres ci-dessous), éclaire l'objet qui se situe alors directement entre elle et lui. (L'objet inconnu et les deux collègues sont ainsi tous dans le même plan Z=0.) Dites quelle est la dernière propriété manquante qu'il peut déterminer **et** donnez l'expression *(sans faire les calculs)* qui permettrait à l'ingénieur investigateur de la déterminer. **[4 points]**

```
// lumière supplémentaire:
GLfloat  lumiAmbi[] = { 0.0, 0.0, 0.0, 1.0 };
GLfloat  lumiDiff[] = { 0.9, 0.9, 0.8, 1.0 };
GLfloat  lumiSpec[] = { 0.2, 0.2, 0.1, 1.0 };
glLightfv( GL_LIGHT2, GL_AMBIENT, lumiAmbi );
glLightfv( GL_LIGHT2, GL_DIFFUSE, lumiDiff );
glLightfv( GL_LIGHT2, GL_SPECULAR, lumiSpec );
```

L'ingénieur mesure la couleur perçue lorsque l'ingénieure seulement *éclaire l'objet. On peut alors déduire matSpec avec :*

*couleur = matEmis + modAmbi + (lampAmbi + lumiAmbi) * matAmbi*
*+ ($(L_{lamp} \cdot N)$ * lampDiff + $(L_{lumi} \cdot N)$ * lumiDiff) * matDiff*
*+ ($(V \cdot N)^{matExpo}$ * lampSpec + $(V \cdot N)^{matExpo}$ * lumiSpec) * matSpec*

Puisque modAmbi = lampAmbi = lumiAmbi = lampSpec = (0.0, 0.0, 0.0), on simplifie un peu :

*couleur = matEmis + ($(L_{lamp} \cdot N)$ * lampDiff + $(L_{lumi} \cdot N)$ * lumiDiff) * matDiff + $(V \cdot N)^{matExpo}$ * lumiSpec * matSpec*

et

*matSpec = (couleur - matEmis - ($(L_{lamp} \cdot N)$ *lampDiff + $(L_{lumi} \cdot N)$ * lumiDiff)*matDiff)/($(V \cdot N)^{matExpo}$ * lumiSpec)*
*matSpec = (couleur - matEmis - ((2m/8m) * lampDiff + (2m/8m) * lumiDiff) * matDiff) / ($(2m/8m)^{matExpo}$ * lumiSpec)*
*matSpec = (couleur - matEmis - 0.25 * (lampDiff + lumiDiff) * matDiff) / ($0.25^{matExpo}$ * lumiSpec)*
matSpec = ...

d) En vue d'écrire un programme pour modéliser la scène, comment doit-on exprimer les positions respectives (en mètres, telles que décrites en (a) et (b)) de la lampe de l'ingénieur et de la lune pour décrire la situation en OpenGL (dans le plan Z=0 et en considérant que l'origine est telle qu'illustrée dans la figure ci-desssus) ? **[2 points]**

```
GLfloat  lampPosition[] = { ? };
GLfloat  lunePosition[] = { ? };
glLightfv( GL_LIGHT0, GL_POSITION, lampPosition );
glLightfv( GL_LIGHT1, GL_POSITION, lunePosition );
```

GLfloat lampPosition = (0, 2, 0, 1) ; // lumière positionnelle
GLfloat lunePosition = (0, 1, 0, 0) ; // lumière directionnelle

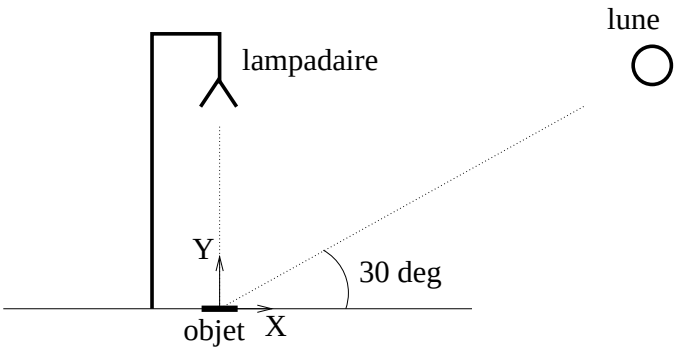
Question 41 Modèle d'illumination

Votre collègue a déniché un programme intéressant qui possède de belles sources de lumière : une lune et un lampadaire. Les caractéristiques des sources de lumière et des propriétés de matériau sont donnés dans un fichier de configuration :

```
Global: // coefficient ambiant global
 Ambi = { 0.0, 0.0, 0.0, 1.0 }
Lune: // position, ambiant, diffuse, spéculaire
 Position = { 0.866, 0.5, 0.0, 0.0 }
 Ambi = { 0.1, 0.1, 0.1, 1.0 }
```

```
 Diff = { 0.3, 0.3, 0.5, 1.0 }
 Spec = { 0.1, 0.1, 0.2, 1.0 }
Lampadaire: // position, ambiant, diffuse, spéculaire, att.
 Position = { 0.0, 3.0, 0.0, 1.0 }
 Atténuation = 1/(1+d)
 Ambi = { 0.0, 0.0, 0.0, 1.0 }
 Diff = { 0.8, 0.6, 0.6, 1.0 }
 Spec = { 0.1, 0.3, 0.0, 1.0 }
Matériau: // émission, ambiant, diffuse, spéculaire, brill.
 Emis = { 0.0, 0.0, 0.0, 1.0 }
 Ambi = { 0.2, 0.1, 0.1, 1.0 }
 Diff = { 0.8, 0.4, 0.2, 1.0 }
 Spec = { 0.0, 0.0, 0.0, 1.0 }
 Expo = 2
```

Le diagramme suivant illustre la scène. Un objet très mince avec les propriétés de matériau du fichier de configuration est placé au sol à l'origine. La lune et un lampadaire éclairent la scène. La lumière de la lune n'est pas atténuée, tandis que celle du lampadaire l'est par un facteur de `1/(1+d)` et le modèle d'illumination de Lambert (GL_FLAT) est utilisé.



a) Quel est la couleur perçu du pixel au centre de la face supérieure de l'objet si seulement la lune éclaire la scène ? **[2 points]**

*coul1 = matAmbi * luneAmbi + matDiff * luneDiff * $(L \cdot N)$*
*coul1 = (0.2, 0.1, 0.1) * (0.1, 0.1, 0.1) + (0.8, 0.4, 0.2) * (0.3, 0.3, 0.5) * cos(60)*
*coul1 = (0.02, 0.01, 0.01) + (0.24, 0.12, 0.10) * 0.5*
coul1 = (0.02, 0.01, 0.01) + (0.12, 0.06, 0.05)
coul1 = (0.14, 0.07, 0.06)

b) Quel est la couleur perçu du pixel au centre de la face supérieure de l'objet si la lune et le lampadaire éclairent la scène ? **[3 points]**

*coul2 = coul1 + (matAmbi * lampAmbi + matDiff * lampDiff * $(L \cdot N)$)/(1+d)*
*coul2 = (0.14, 0.07, 0.06) + ((0.8, 0.4, 0.2) * (0.8, 0.6, 0.6) * 1)/(1+3)*
coul2 = (0.14, 0.07, 0.06) + (0.64, 0.24, 0.12)/(4)
coul2 = (0.14, 0.07, 0.06) + (0.16, 0.06, 0.03)
coul2 = (0.30, 0.13, 0.09)

c) Quel est l'autre modèle d'illumination qui aurait pu être utilisé avec OpenGL ? **[1 point]**

Le modèle de Gouraud, GL_SMOOTH. (Le modèle de Phong n'est pas directement supporté par OpenGL.)

Question 42 Modèle d'illumination [7 points]

Une salle avec un plancher en or (le plan y=0) est éclairée par une ampoule sphérique située au plafond à la position $(0, 5\sqrt{3}, 0)$. Un observateur, dont l'oeil est situé à la position $(5(\sqrt{3}+1), 5, 0)$, regarde un point situé sur le plancher à la position $(5, 0, 0)$. Sachant que les propriétés de réflexion de l'or sont les suivantes :

Coefficient de réflexion ambiante : (0.3, 0.1, 0.1)
Coefficient de réflexion diffuse : (0.8, 0.7, 0.2)
Coefficient de réflexion spéculaire : (0.4, 0.3, 0.1)
Exposant de brillance : 2

que l'ampoule émet de la lumière dont le spectre correspond à la couleur RGB (1.0, 1.0, 0.3), qu'il existe dans la pièce une lumière ambiante dont le niveau RGB est (0.2, 0.2, 0.2) et que nous utilisons le modèle d'illumination vu en classe.

a) Quelle est la couleur perçue du plancher, en ne tenant compte que de la lumière ambiante ? [2 points]

C'est le résultat du produit scalaire de la "couleur" ambiante et des coefficients de réflexion ambiante : $(0.2, 0.2, 0.2) * (0.3, 0.1, 0.1) = (0.06, 0.02, 0.02)$.

b) Quelle est la couleur obtenue par réflexion diffuse de l'ampoule au point regardé par l'observateur, en négligeant le facteur d'atténuation ? [2 points]

Intensité de l'ampoule x coefficient de réflexion diffuse x cosinus de l'angle entre la normale du plancher et le rayon incident :
$(1.0, 1.0, 0.3) * (0.8, 0.7, 0.2) *$ *côté opposé* $(= 5)/$ *hypothénuse* $(= 10) = (0.4, 0.35, 0.03)$.

c) Quel serait le résultat de (b) si on utilisait un facteur d'atténuation de la forme $1/(0.1 * d^2)$? [1 point]

La distance de l'ampoule au point est de 10 unités. Par conséquent, il faut multiplier par $1/(0.1*10^2) = 0.1$, *ce qui donne* $(0.04, 0.035, 0.003)$.

d) Si nous ajoutons une bille d'argent sur notre plancher, modélisée à l'aide d'une douzaine de triangles, devrions-nous utiliser le modèle de Lambert ou celui de Gouraud ? [1 point]

Gouraud

e) En OpenGL, si on fixe l'attribut GL_POSITION de GL_LIGHT0 à (3,2,1,0), a-t-on créé une source de lumière ponctuelle ou directionnelle ? [1 point]

Directionnelle, puisque les coordonnées cartésiennes de cette source sont $(3/0, 2/0, 1/0)$*, une position à l'infini.*

Question 43 Modèle d'illumination [6 points]

Un petit cube de jade (<$1cm^3$) est déposé à plat sur le plancher lisse d'une grotte et un observateur debout regarde ce cube. La distance mesurée entre sa tête et le cube est de 5m et sa tête se trouve à 2.5m du sol. L'observateur a une lampe frontale à la tête qui éclaire la totalité de cet objet. Sachant qu'on utilise le modèle standard de lumière d'OpenGL avec une lumière ambiante d'intensité (0.2, 0.2, 0.2), que l'ampoule n'émet que de la lumière diffuse orangée d'intensité (1.0, 0.5, 0.1) et que les propriétés du jade sont les suivantes :

Coefficient de réflexion ambiante :	(0.1, 0.2, 0.2)
Coefficient de réflexion diffuse :	(0.5, 1.0, 0.5)
Coefficient de réflexion spéculaire :	(0.3, 0.3, 0.3)
Exposant de brillance :	10

a) Quelle est la couleur perçue par l'observateur de la face supérieure du cube, en ne tenant compte que de la composante ambiante et sans facteur d'atténuation ? [1 point]

*ambiantModel * ambiantMateriel =*
*(0.2, 0.2, 0.2) * (0.1, 0.2, 0.2) = (0.02, 0.04, 0.04)*

b) Quelle est la couleur perçue par l'observateur de la face supérieure du cube, en ne tenant compte que de la composante diffuse et sans facteur d'atténuation ? [2 points]

*$(L \cdot N)$ * diffuseLumiere * diffuseMateriel =*
*cos(60) * (1.0, 0.5, 0.1) * (0.5, 1.0, 0.5) =*
*0.5 * (0.5, 0.5, 0.05) = (0.25, 0.25, 0.025)*

c) Quelle serait la couleur perçue par l'observateur de cet objet, si on utilisait un facteur d'atténuation linéaire de la forme 1.0 / (1.0 x dist) et tenant compte de toutes les composantes du modèle de lumière ? [2 points]

*ambiantModel * ambiantMateriel +*
*($(L \cdot N)$ * diffuseLumiere * diffuseMateriel) * (1/(1 * 5)) =*
(0.02, 0.04, 0.04) + (0.25, 0.25, 0.025) / 5 =
(0.02, 0.04, 0.04) + (0.05, 0.05, 0.005) = (0.07, 0.09, 0.045)

d) Quelles valeurs devons-nous mettre dans le tableau `lumPos[]` dans l'exemple qui suit afin de définir une source de lumière provenant de la position (0,2,3) ? [1 point]

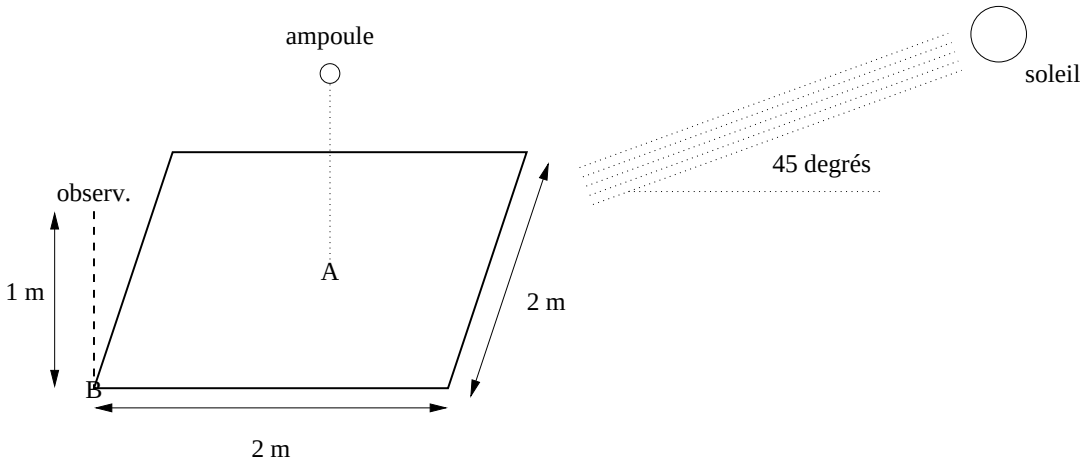
```
GLfloat lumPos[4] = { ???? };
glLightfv( GL_LIGHT0, GL_POSITION, lumPos );
```

{ 0, 2, 3, 1 }

Question 44 Modèle d'illumination [10 points]

Une surface carrée horizontale de 2 mètres de côté est recouverte d'un matériau uniforme parfaitement diffus (coefficients diffus = $(1, 1, 1)$ et spéculaire = $(0, 0, 0)$). La surface est éclairée par le soleil dont les rayons font un angle de 45 degrés avec l'horizon (le soleil est considéré comme une lumière située à l'infini et on suppose qu'il n'y a aucune lumière ambiante ni atténuation avec la distance). L'observateur est placé à 1 mètre au-dessus de la surface, et il se trouve directement au-dessus d'un des sommets du carré.

Notons A le point situé exactement au centre de la surface et B le point de la surface directement sous l'observateur.



a) Quel est le ratio entre l'intensité de lumière perçue par l'observateur au point A et au point B (lorsque la surface est éclairée par le soleil seulement) ? (intensité de A / intensité de B) [1 point]

1

b) Supposons que l'on éteigne(!) le soleil et que l'on allume une ampoule se trouvant à 1 mètre au-dessus de la surface, exactement au centre de celle-ci. Que devient le ratio précédent (lorsque la surface est éclairée par l'ampoule seulement) ? [2 points]

$cos(ThetaA) = 1$
$cos(ThetaB) = (0, 1, 0) \cdot (1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3}) = 1/\sqrt{3}$

intensité de A / intensité de B $= \sqrt{3}$

c) Dans la question précédente, si l'observateur s'éloigne verticalement jusqu'à 2 mètres au-dessus de la surface, que devient ce ratio ? [1 point]

Il ne change pas.

Soit une sphère de rayon 1 centrée à l'origine et recouverte d'un matériau uniforme parfaitement spéculaire (coefficients diffus = $(0,0,0)$ et spéculaire = $(1,1,1)$). Supposons que l'éclairage provienne d'une source située à l'infini d'intensité $(1,1,1)$ et dirigée dans la direction $(0,-1,0)$. Supposons aussi que l'observateur se situe à l'infini et regarde dans la direction $(0,0,-1)$.
(On considère qu'il n'y a aucune lumière ambiante ni atténuation avec la distance.)

d) Quelle est la direction du vecteur normal au point de la surface où l'intensité est maximale ? [2 points]

Vecteur bissecteur non renormalisé = $(0,1,0)+(0,0,1)=(0,1,1)$

Vecteur bissecteur renormalisé = $(0,1/\sqrt{2},1/\sqrt{2})$

L'intensité spéculaire est maximale quand le vecteur bissecteur égal la normale, donc :

N = $(0,1/\sqrt{2},1/\sqrt{2})$

e) Si l'exposant de brillance est n = 2, quelle est l'intensité de la lumière perçue au point dont les coordonnées sont $(1/\sqrt{3},1/\sqrt{3},1/\sqrt{3})$? [2 points]

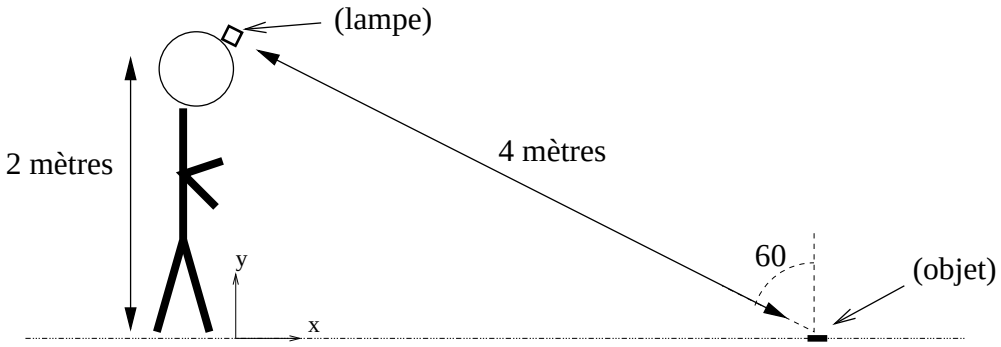
- *Calcul avec le vecteur de lumière réfléchi R.*
  N = $(1/\sqrt{3},1/\sqrt{3},1/\sqrt{3})$
  R = $2N(0,1,0).N-(0,1,0) = 2(1/3,1/3,1/3)-(0,1,0) = (2/3,-1/3,2/3)$
  I = $((2/3,-1/3,2/3)\cdot(0,0,1))^2 = 4/9$
- *Calcul avec le vecteur bissecteur B.*
  N = $(1/\sqrt{3},1/\sqrt{3},1/\sqrt{3})$
  B = $(0,1/\sqrt{2},1/\sqrt{2})$
  I = $((1/\sqrt{3},1/\sqrt{3},1/\sqrt{3})\cdot(0,1/\sqrt{2},1/\sqrt{2}))^2 = 4/6$

f) Pourquoi n'est-il pas recommandé de prendre de photo en utilisant un flash directement devant une fenêtre ? (Appuyez votre explication sur le contenu et la terminologie du cours.) [2 points]

Parce que le verre a un coefficient spéculaire non nul, donc la lumière intense du flash est réfléchie par la vitre. Si on se trouve directement devant la fenêtre, le vecteur de lumière réfléchie se confond avec le vecteur observateur et l'intensité spéculaire est grande, ce qui contribue à masquer le paysage derrière la vitre.

Question 45 Modèle d'illumination [10 points]

La nuit, sous un ciel couvert et sans autre éclairage que sa lampe frontale, un explorateur se promène dans un lieu désert. Les piles de sa lampe faiblissent et la lampe émet une lumière plutôt jaunâtre. Du haut de ses 2 m, en regardant le sol, il aperçoit quelque chose devant lui qui se trouve à une distance de 4 m de sa lampe frontale. Il s'arrête. C'est un petit objet carré, plat et opaque, dont la taille est d'environ 1cm x 1cm. Sans bouger, il éclaire cet objet en totalité avec la lueur de sa lampe.



Si on considére que le modèle d'illumination d'OpenGL est celui qui régit le monde et en négligeant tout facteur d'atténuation, cette situation hypothétique peut être décrite de la façon suivante :

```
GLfloat  modAmbiant[]     = { 0.2, 0.2, 0.2, 1.0 };
glLightModelfv( GL_LIGHT_MODEL_AMBIENT, modAmbiant );
GLfloat  lampAmbiant[]    = { 0.1, 0.1, 0.1, 1.0 };
GLfloat  lampDiffuse[]    = { 0.5, 0.5, 0.1, 1.0 };
GLfloat  lampSpeculaire[] = { 0.0, 0.0, 0.0, 1.0 };
glLightfv( GL_LIGHT0, GL_AMBIENT, lampAmbiant );
glLightfv( GL_LIGHT0, GL_DIFFUSE, lampDiffuse );
glLightfv( GL_LIGHT0, GL_SPECULAR, lampSpeculaire );
```

a) L'explorateur, étant bien équipé, peut mesurer séparément les couleurs perçues de cet objet dues à la lumière émise par sa lampe seulement. Il constate ainsi que l'objet ne réfléchit aucune lumière spéculaire et n'émet aucune lumière (GL_EMISSION = (0, 0, 0, 0)). Il mesure l'intensité des composantes ambiante et diffuse de cet objet (dues à sa lampe seulement) et celles-ci sont : coulAmbiant = (0.225, 0.22, 0.21, 1.0) et coulDiffuse = (0.175, 0.15, 0.01, 1.0). Quelle est la couleur résultante (r,g,b,a) de l'objet qui est perçue par l'explorateur selon le modèle d'illumination d'OpenGL ? [2 points]

couleur = coulAmbiant + coulDiffuse
couleur = (0.225, 0.22, 0.21, 1.0) + (0.175, 0.15, 0.01, 1.0)
couleur = (0.4, 0.37, 0.22, 1.0)

b) Avec les mêmes paramètres qu'en (a), quelles sont les propriétés ambiante et diffuse du matériau de cet objet ? [6 points]

```
GLfloat  matAmbiant[] = { ?, ?, ?, 1.0 };
GLfloat  matDiffuse[] = { ?, ?, ?, 1.0 };
glMaterialfv( GL_FRONT_AND_BACK, GL_AMBIENT, matAmbiant );
glMaterialfv( GL_FRONT_AND_BACK, GL_DIFFUSE, matDiffuse );
```

La réponse attendue :
*coulAmbiant = (modAmbiant + lampAmbiant) * matAmbiant*
matAmbiant = coulAmbiant / (modAmbiant + lampAmbiant)
matAmbiant = (0.225, 0.22, 0.21, 1.0) / ((0.2, 0.2, 0.2, 1.0) + (0.1, 0.1, 0.1, 1.0))
matAmbiant = (0.225, 0.22, 0.21, 1.0) / (0.3, 0.3, 0.3, 1.0)
matAmbiant = (0.75, 0.7333, 0.7, 1.0)

Mais, on acceptera aussi :
*coulAmbiant = lampAmbiant * matAmbiant*

matAmbiant = coulAmbiant / lampAmbiant
matAmbiant = (0.225, 0.22, 0.21, 1.0) / (0.1, 0.1, 0.1, 1.0)
matAmbiant = (2.25, 2.2, 2.1, 1.0)

coulDiffuse = $(L \cdot N)$ ** lampDiffuse * matDiffuse*
matDiffuse = coulDiffuse / ($(L \cdot N)$ ** lampDiffuse)*
*matDiffuse = coulDiffuse / (cos(60) * lampDiffuse)*
*matDiffuse = (0.175, 0.15, 0.01, 1.0) / (0.5 * (0.5, 0.5, 0.1, 1.0))*
matDiffuse = (0.175, 0.15, 0.01, 1.0) / (0.25, 0.25, 0.05, 1.0)
matDiffuse = (0.7, 0.6, 0.2, 1.0)

c) Comment doit-on exprimer la position (en mètres) de la lampe frontale en OpenGL pour décrire la situation ci-dessus (dans le plan z=0 et en considérant que l'origine est aux pieds de l'observateur, comme illustré dans la figure ci-desssus) ? [2 points]

```
GLfloat  lampPosition[] = { ? };
glLightfv( GL_LIGHT0, GL_POSITION, lampPosition );
```

lampPosition = (0, 2, 0, 1) ; $\Rightarrow$ *lumière positionnelle*

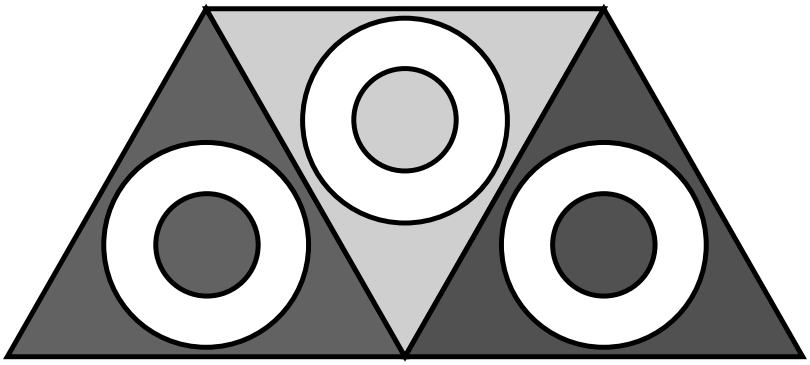
Question 46 Listes d'affichage [10 points]

La forme décrite ci-dessous est composée de trois triangles équilatéraux de différentes couleurs et chacun des triangles contient deux cercles concentriques. Les deux cercles concentriques délimitent une surface 2D colorée en blanc et sont centrés au centre de gravité de chaque triangle. Étant donné (x1, y1), (x2, y2) et (x3, y3) les coordonnées 2D des sommets d'un triangle, les coordonnées du centre de gravité du triangle sont données par xg = (x1 + x2 + x3)/3 et yg = (y1 + y2 + y3)/3.

a) Donnez les énoncés OpenGL qui permettent de créer la liste d'affichage qui trace deux cercles concentriques de rayons 1 cm et 2 cm respectivement. La surface délimitée par les deux cercles est pleine et colorée en blanc. Pour cette question, les deux cercles doivent être centrés à l'origine et doivent être tracés à l'aide de primitives OpenGL de base et non des primitives des librairies GLU ou GLUT. [4 points]

b) Donnez les énoncés OpenGL qui permettent de créer la liste d'affichage qui trace un triangle équilatéral plein, de couleur rouge, et qui fait appel à la liste d'affichage donnée dans a). Les cercles concentriques sont centrés au centre de gravité du triangle. On ne demande pas de calculer les valeurs numériques des coordonnées (xg, yg) du centre de gravité. Les paramètres des transformations nécessaires peuvent être exprimés en fonction de (xg, yg). [3 points]

c) Après avoir défini un système de coordonnées global, donnez les énoncés OpenGL qui décrivent une structure hiérarchique de listes d'affichage pour tracer la forme illustrée ci-dessous en utilisant les listes d'affichage données en (a) et (b). Les trois triangles sont respectivement rouge, vert et bleu. [3 points]

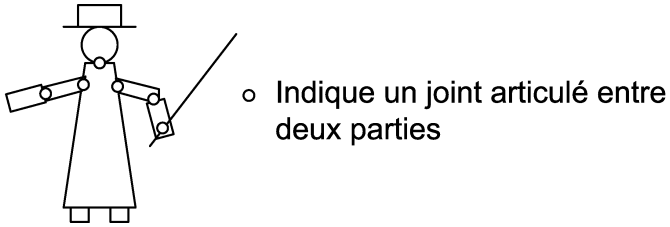


Question 47 Listes d'affichage

Durant un stage, vous travaillez chez *JeuRoleSoft*, la célèbre firme québécoise ayant mis au point le jeu *Le monde des artisans de la guerre*. Étant donné les connaissances acquises durant votre cours INF2700, vous êtes assigné au groupe *rendu et modélisation 3D*. Vous constatez que les membres du groupe n'utilisent pas les listes d'affichage. Un de vos nouveaux collègues vous mentionne que les listes d'affichage ne servent à rien.

a) Quels arguments (donnez en 2) allez vous lui donner pour le convaincre de l'utilité des listes d'affichage. Chaque argument doit être clairement expliqué. **[2 points]**

b) Comment utiliseriez-vous les listes d'affichage pour faire le rendu de façon optimale du personnage *Troll de la pleine lune* schématisé ci-dessous en 2D ? Supposez que chaque partie du corps du personnage est composée d'un très grand nombre de triangles. Donnez les éléments faisant partie de chaque liste d'affichage que vous utiliseriez. **[3 points]**



c) Donnez les commandes OpenGL pour créer une liste d'affichage composée d'une tête sphérique blanche de 10 pixels de diamètre. Utilisez `gluSphere(GLUquadricObj *qobj, GLdouble radius, GLint slices, GLint stacks)`. **[3 points]**

Question 48 Listes d'affichage et hiérarchie

À l'occasion du nouvel an chinois qui aura lieu le 7 février 2008 prochain, un ami vous demande de lui créer un dragon virtuel en openGL qu'il pourra intégrer dans son application le moment venu afin d'épater les enfants.

Pour ce faire, vous devez lui écrire les fonctions suivantes :

```
void afficherDragon(float theta1, float theta2, float theta3, float posX, float posY);
void compilerTete( );
void compilerCorps( );
void compilerQueue( );
```

La fonction `afficherDragon` se chargera d'afficher le dragon tout entier à la position virtuelle (posX, posY), qui correspond au milieu de la tête du dragon. La fonction devra *impérativement* faire usage des listes d'affichages et des transformations matricielles, et mettre de l'avant la hiérarchisation des listes d'affichage afin de dessiner le dragon de façon performante. Vous devez aussi utiliser dans votre code les arguments d'entrées `theta1`, `theta2`, `theta3`, `posX` et `posY`.

Les fonctions `compilerTete`, `compilerCorps` et `compilerQueue` se chargeront de compiler en listes d'affichage distinctes le graphisme des segments correspondants.

L'unique axe de rotation de la tête et de la queue se situe en (0,0), alors que le segment de corps possède deux axes de rotation en (0,0) et en (-4,0). Le graphisme des différentes parties du dragon vous est fourni en coordonnées d'objet dans la figure 1.

Tel q'indiqué sur la figure 2, le dragon est composé (dans l'ordre) d'une « tête » jaune, d'un premier segment de « corps » vert, d'un deuxième segment de « corps » bleu et d'une « queue » rouge, articulés entre eux selon les angles θ_1, θ_2 et θ_3. La position du dragon en coordonnées universelles se situe en (posX, posY), point qui correspond à la commissure des lèvres de la tête du dragon (voir figure 1).

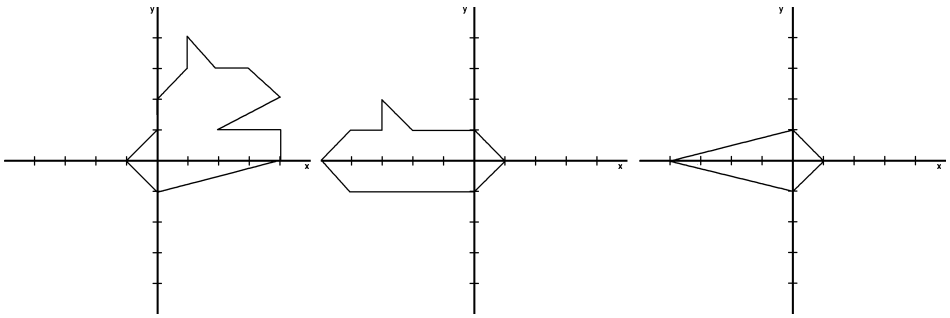


Figure 1 : Définition en coordonnées d'objet de la tête (gauche), d'un segment de corps (milieu) et de la queue (droite).

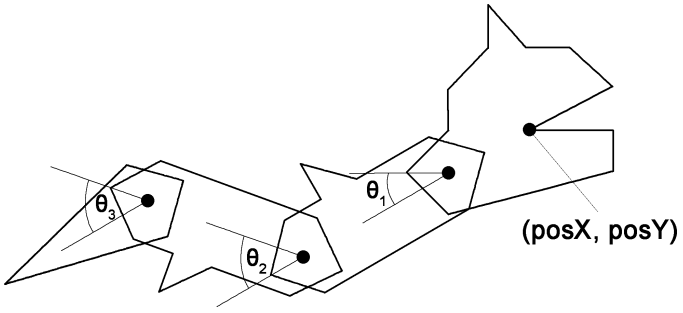
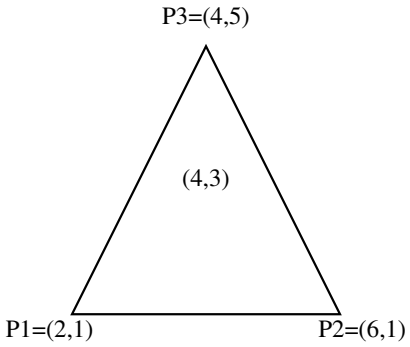


Figure 2 : Constitution du dragon. Le dragon est constitué d'une tête, de deux segments corps (l'un inversé, l'autre normal) et d'une queue.

Écrivez les quatre fonctions demandées. **[10 points]**

Question 49 Listes d'affichage



a) Considérez le triangle plein décrit par trois sommets dont les coordonnées sont : P1=(2,1), P2=(6,1) et P3=(4,5). Donnez tous les énoncés OpenGL nécessaires pour créer correctement la liste d'affichage qui affichera ce triangle aux coordonnées indiquées. (Si on appelle la liste d'affichage sans aucune autre transformation, le triangle s'affiche à ces coordonnées.) On stockera le numéro de la liste d'affichage dans la variable `triLA`. **[3 points]**

```
// Création de la liste d'affichage
GLuint triLA = ...
...
```

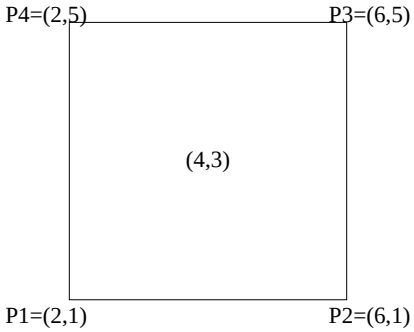
b) On souhaite maintenant créer une séquence d'animation où ce triangle tournera sur lui-même, autour du point (4,3), dans le plan Z=0. Pour tester l'animation, seulement ce triangle tournant sur lui-même sera d'abord tracé et animé. Le programme principal appellera la fonction `affiche` qui utilise la liste d'affichage créée précédemment pour afficher le triangle, en tournant à chaque fois le triangle d'un angle supplémentaire de 1 degré autour du point (4,3). Donnez les énoncés qu'il faut mettre dans cette fonction, sachant qu'elle reçoit en argument le numéro de la liste d'affichage et qu'elle appelle la liste précédemment créée en (a). **[4 points]**

```
// Affichage de la scène
static void affiche( const GLuint triLA )
{
  static GLfloat angleCourant = 0; // en degrés
  angleCourant += 1.0;
  glClear( GL_COLOR_BUFFER_BIT );
  ...
}
```

c) Dans une compagnie où vous venez d'être embauché, un de vos nouveaux collègues soutient que les listes d'affichage ne servent à rien. Pour le convaincre du contraire, donnez deux arguments militant en faveur de l'utilisation des listes d'affichage. Expliquez brièvement et clairement chacun. **[3 points]**

Question 50 Listes d'affichage

Considérez le rectangle décrit par quatre sommets dont les coordonnées sont : P1=(2,1), P2=(6,1), P3=(6,5) et P4=(2,5). On souhaite créer une séquence d'animation où ce rectangle doit tourner sur lui-même, autour de son centre de gravité, dans le plan 2D. Bien sûr, on voudra utiliser des listes d'affichage afin d'être efficace !



a) Donnez tous les énoncés OpenGL nécessaires pour créer la liste d'affichage qui contient ce rectangle. On stockera le numéro de la liste d'affichage dans la variable `rectN`. **[3 points]**

```
// Création de la liste d'affichage
GLuint rectN = ...
...
```

b) Pour tester l'animation, seulement ce rectangle tournant sur lui-même sera d'abord tracé et animé. Le programme principal appelera la fonction `affiche` qui utilise la liste d'affichage créée précédemment pour afficher le rectangle, en tournant à chaque fois le rectangle d'un angle supplémentaire de 1 degré autour de son centre de gravité situé en (4,3). Donnez les énoncés qu'il faut mettre dans cette fonction, sachant qu'elle reçoit en argument le numéro de la liste d'affichage. **[4 points]**

```
// Affichage de la scène
static void affiche( const GLuint rectN )
{
  static GLfloat angleCourant = 0; // en degrés
  angleCourant += 1.0;
  glClear( GL_COLOR_BUFFER_BIT );
  ...
}
```

c) Donnez deux arguments militant en faveur de l'utilisation des listes d'affichage. Expliquez brièvement et clairement chacun. **[3 points]**

Question 51 Listes d'affichage

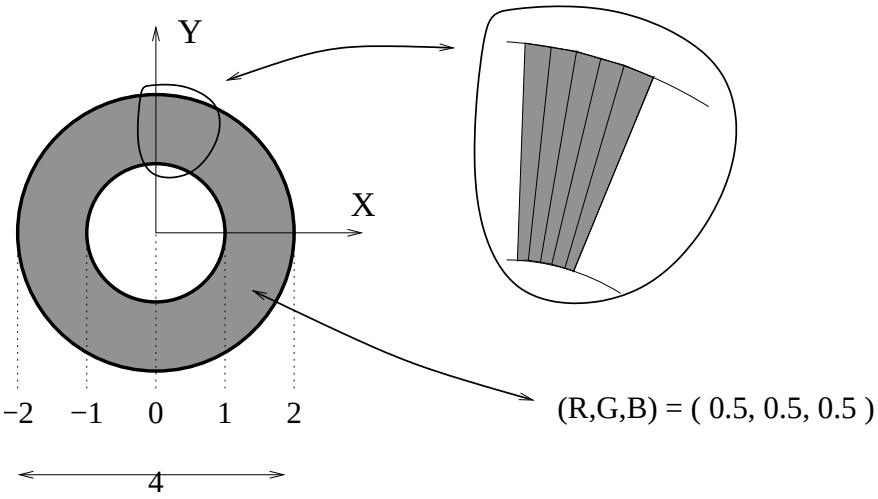


Figure 1

a) On souhaite tracer un anneau plein en approximant sa forme par des quadrilatères minces (voir Fig. 1). On utilisera une seule primitive OpenGL (une seule paire `glBegin() ... glEnd()` et aucune fonction des librairies GLU ou GLUT) et plusieurs `glVertex`.
Écrivez les énoncés C++ qui créent une liste d'affichage qui tracera cet anneau *centré à l'origine*, de la *taille* et de la *couleur* indiquées dans la figure. (On veut voir les appels aux fonctions OpenGL avec leurs arguments, de même que la boucle `for` ($n \approx 50$) qui calcule les coordonnées des sommets.) **[3 points]**

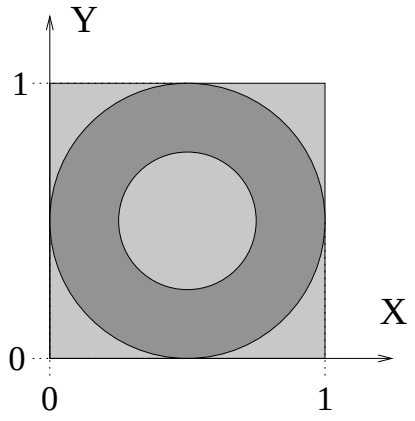


Figure 2

b) On souhaite maintenant tracer un carré unitaire plein (voir Fig. 2) où est inscrit l'anneau précédent. Notez la position de l'origine ! À nouveau, on utilisera une seule primitive OpenGL (une seule paire `glBegin() ... glEnd()` et aucune fonction des librairies GLU ou GLUT).
Écrivez les énoncés C++ qui créent une seconde liste d'affichage qui tracera cette figure à la *position* et de la *taille* indiquées dans la figure et en faisant appel à la liste d'affichage créée en (a) pour inscrire l'anneau dans le carré. (On veut voir les appels aux fonctions OpenGL avec leurs arguments, incluant l'appel à la liste d'affichage de l'anneau.) La couleur de fond du carré sera spécifiée par ailleurs. **[4 points]**

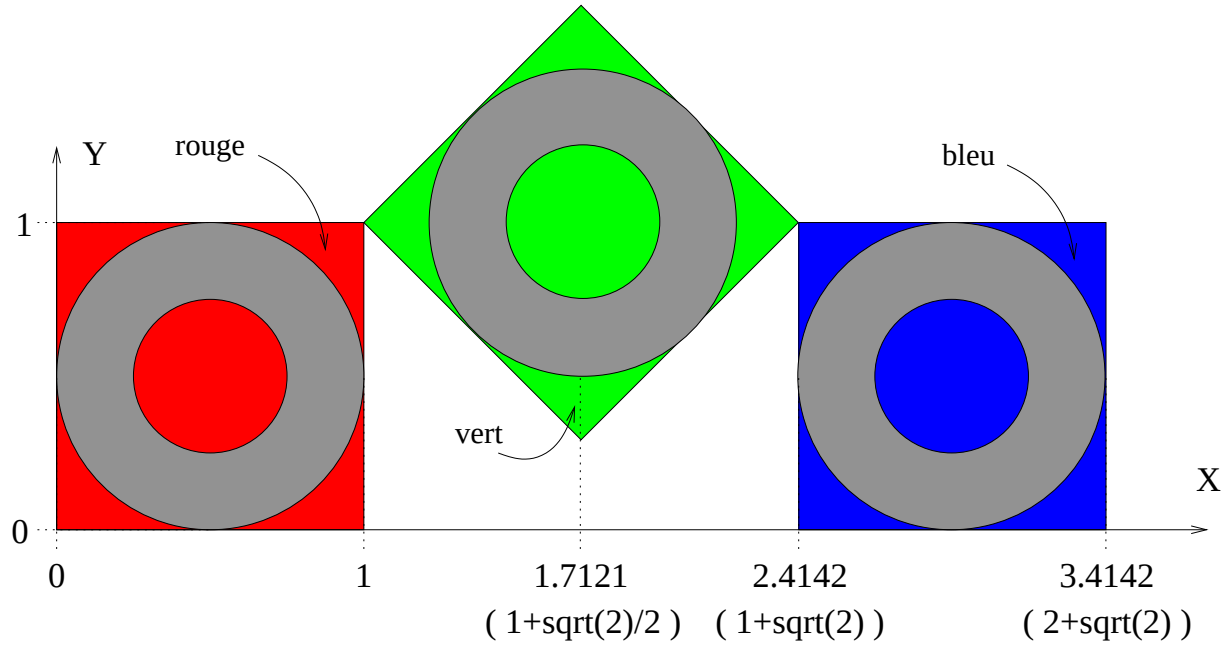


Figure 3

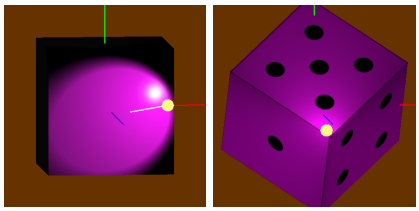
c) On souhaite maintenant utiliser la liste d'affichage définie en (b) pour tracer la forme illustrée ci-dessus (voir Fig. 3).
Écrivez les énoncés C++ qui tracent cette figure à la *position*, de la *taille* et de la *couleur* indiquées dans la figure et en faisant appel à la liste d'affichage créée en (b). (On veut voir les appels aux fonctions OpenGL avec leurs arguments, incluant la spécification des couleurs de fond des carrés.) **[4 points]**

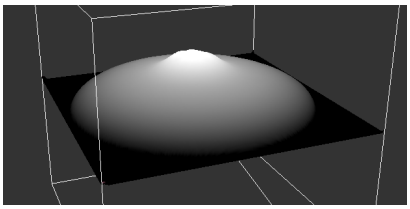
Question 52 Utilisation des nuanceurs en GLSL

a) Lors du TP07 portant sur l'illumination, vous avez commencé par subdiviser un côté du cube afin d'améliorer le rendu. Ensuite, comme meilleure solution, vous avez implanté le modèle d'illumination souhaité directement dans le nuanceur de fragments.

Lors du TP08 portant sur le déplacement de surfaces, vous avez utilisé des textures pour déformer la surface affichée (un carré plat) selon l'intensité de la couleur dans ces textures. Même si aucun requis du TP ne l'imposait explicitement et même si les textures sont facilement utilisables dans le nuanceur de fragments, la déformation a été calculée et appliquée dans le nuanceur de sommets sur de petits quadrilatères plutôt que d'être implantée dans le nuanceur de fragments.

Quelle est la raison principale pour laquelle vous n'avez pas calculé et appliqué les déformations de la surface dans le nuanceur de fragments ? **[2 points]**

TP07 : 

TP08 : 

b) On vous donne ci-dessous deux nuanceurs simples qui permettent d'appliquer, sans illumination, une texture donnée à une surface :

Nuanceur de sommets :

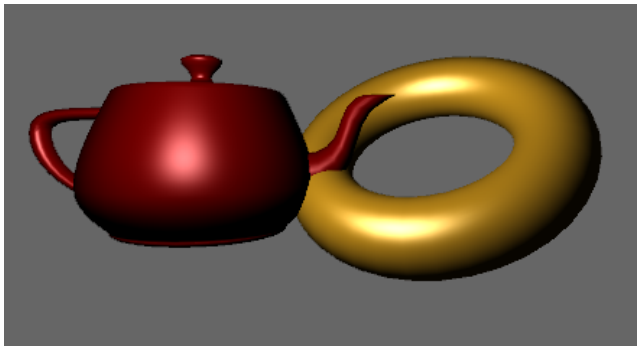
```
void main( void )
{
   gl_TexCoord[0] = gl_MultiTexCoord0;
   gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```

Nuanceur de fragments :

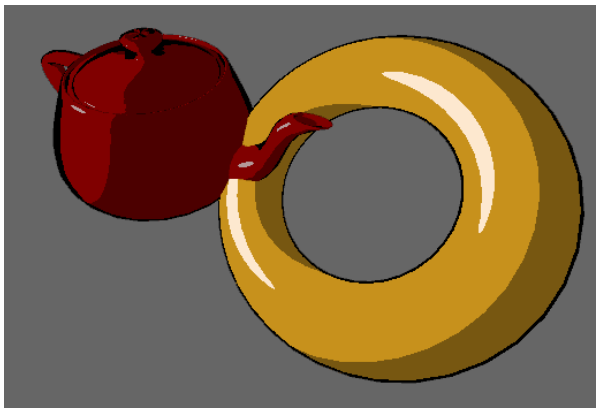
```
uniform sampler2D laTexture;
void main( void )
{
   gl_FragColor = texture2D( laTexture, gl_TexCoord[0].st );
}
```

Écrivez une nouvelle version du nuanceur de sommets qui déforme la surface dans la direction Z en utilisant la couleur de la texture. Puisque la texture choisie est très colorée, vous utiliserez la moyenne des composantes de la couleur (R,G,B) comme facteur de déformation. (Appliquez la même solution qu'au TP08, mais en tenant compte cette fois des trois composantes de couleur de la texture.) **[6 points]**

c) Vous avez décidé d'implanter un rendu celluloïd (« *toon shading* ») dans votre nuanceur de fragments. Ce type de rendu remplace les multiples intensités de couleur calculées avec un modèle d'illumination standard (image de gauche) par un rendu utilisant un très petit nombre de couleurs (image de droite).



Rendu standard avec Phong



Rendu celluloïd à trois intensités

Le nuanceur de sommets déjà existant calcule une valeur *albédo*, toujours comprise entre 0.0 et 1.0, qui est le rapport *« intensité de la lumière réfléchie / intensité de la lumière incidente »*.

Nuanceur de sommets :

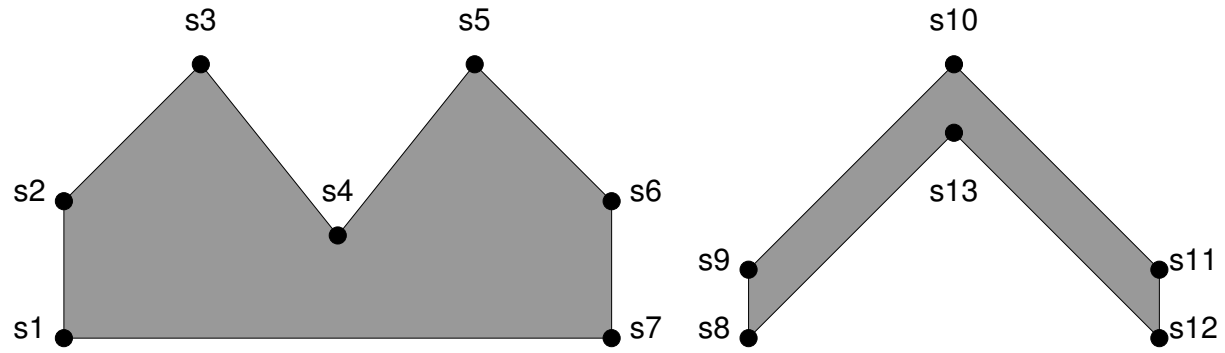
```
varying float albedo;
void main( void )
{
  gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
  vec3 pos_EC = vec3( gl_ModelViewMatrix * gl_Vertex );
  vec3 normale_EC = normalize( gl_NormalMatrix * gl_Normal );
  vec3 lumi_EC = normalize( gl_LightSource[0].position.xyz - pos_EC );
  albedo = dot( normale_EC, lumi_EC );
}
```

Écrivez le nuanceur de fragments qui pourra être utilisé avec ce nuanceur de sommets. Votre nuanceur de fragments assignera simplement la couleur rouge clair (0.9, 0.2, 0.2) si l'albédo est plus grand que 0.7 et la couleur rouge foncé (0.4, 0.0, 0.0) sinon. Votre nuanceur contiendra toutes les lignes nécessaires afin qu'il puisse être compilé sans erreur et produire un programme avec le nuanceur de sommets fourni. (Votre paire de nuanceurs à deux intensités pourrait être utilisée pour afficher une théière avec un rendu celluloïd semblable à celui dans l'image de droite ci-dessus.)

[6 points]

Question 53 Primitives OpenGL

Considérez ces formes pleines tracées par des primitives OpenGL :



Dans les sous-questions suivantes, on vous demande d'écrire les énoncés `glBegin`, `glVertex2iv` et `glEnd` nécessaires pour tracer exactement et correctement les formes ci-dessus en utilisant les primitives d'OpenGL. Les règles à respecter sont :

- vous ne devez pas utiliser d'autres sommets que ceux existants (s1 à s13) ;
- vous devez utiliser un nombre minimal de paires `glBegin() ... glEnd()`.
- vous ne devez pas remplir ou tracer par-dessus ce qui est déjà rempli ou tracé.

Exemple :
Quels sont les énoncés OpenGL si on utilise la primitive `GL_TRIANGLES` ?

```
glBegin( GL_TRIANGLES );
  S1 S7 S4   S1 S4 S2   S2 S4 S3   S7 S6 S4   S6 S5 S4
glEnd( );
glBegin( GL_TRIANGLES );
  S9 S8 S13   S9 S13 S10   S13 S12 S11   S13 S11 S10
glEnd( );
```

où, pour faire plus court dans la réponse, « `S1` » *remplace* « `glVertex2iv(s1);` ».

a) Quels sont les énoncés OpenGL si on utilise la primitive `GL_QUADS` ? **[2 points]**

b) Quels sont les énoncés OpenGL si on utilise la primitive `GL_QUAD_STRIP` ? **[2 points]**

c) Quels sont les énoncés OpenGL si on utilise la primitive `GL_TRIANGLE_STRIP` ? **[2 points]**

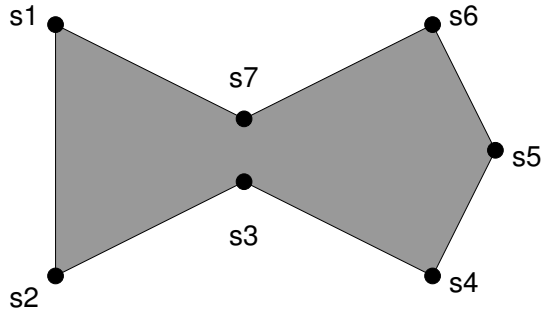
d) Quels sont les énoncés OpenGL si on utilise la primitive `GL_TRIANGLE_FAN` ? **[2 points]**

e) Quels sont les énoncés OpenGL si on utilise la primitive `GL_POLYGON` ? **[2 points]**

f) Pour l'affichage de ces formes, quelles sont les primitives les plus efficaces ? Pourquoi ? **[2 points]**

Question 54 Primitives OpenGL

Considérez cette forme pleine pouvant être tracée par différentes primitives OpenGL :

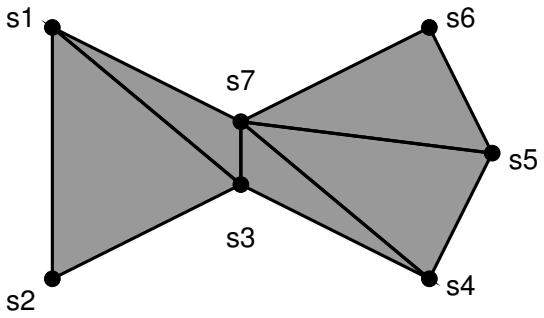


Dans les sous-questions suivantes, on vous demande d'écrire les énoncés `glBegin`, `glVertex2iv` et `glEnd` nécessaires pour tracer exactement et correctement la forme ci-dessus.
Pour faire plus court dans la réponse, utilisez « `S1` » au lieu de « `glVertex2iv(s1);` ».

Les règles à respecter sont :
- vous ne devez pas utiliser d'autres sommets que ceux existants (s1 à s7) ;
- vous devez utiliser un nombre minimal de paires `glBegin() ... glEnd()`.
- vous ne devez pas remplir ou tracer par-dessus ce qui est déjà rempli ou tracé.

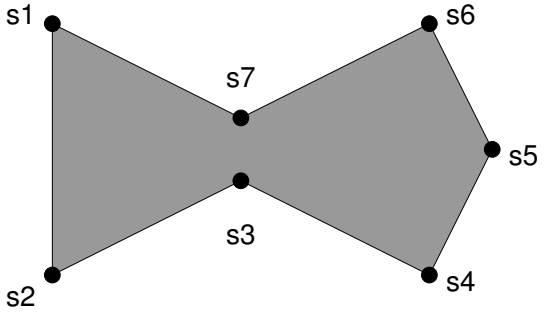
EXEMPLE :

i) Tracez les triangles et ii) donnez les énoncés OpenGL si on utilise la primitive `GL_TRIANGLES` ?

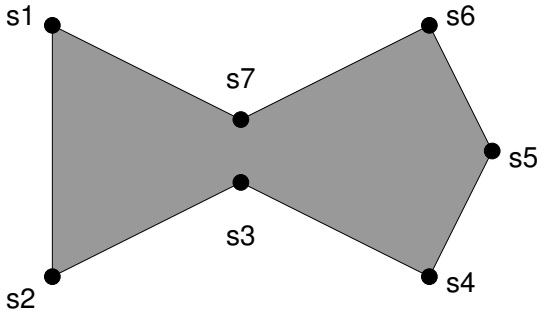



```
glBegin( GL_TRIANGLES );
  s1 s2 s3
  s7 s1 s3
  s3 s4 s7
  s7 s4 s5
  s5 s6 s7
glEnd( );
```

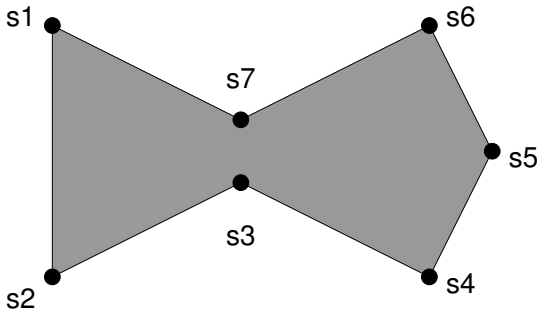
a) i) Tracez les quadrilatères et ii) donnez les énoncés OpenGL si on utilise la primitive `GL_QUADS` ? **[2 points]**



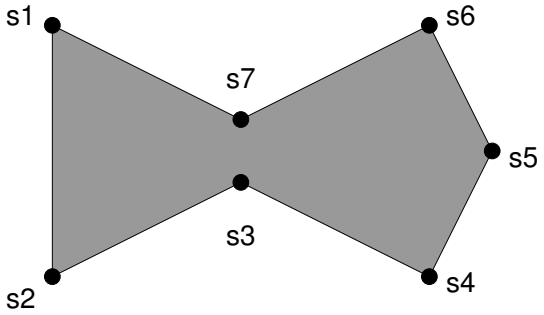
b) i) Tracez les quadrilatères et ii) donnez les énoncés OpenGL si on utilise la primitive `GL_QUAD_STRIP` ? **[2 points]**



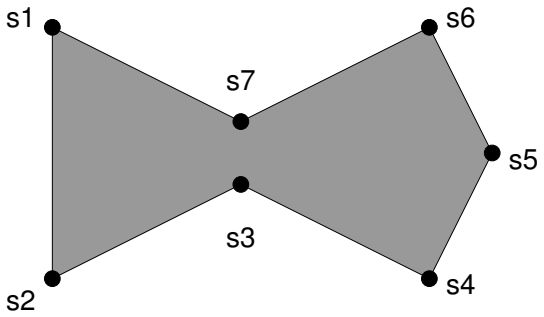
c) i) Tracez les triangles et ii) donnez les énoncés OpenGL si on utilise la primitive `GL_TRIANGLE_STRIP` ? **[2 points]**



d) i) Tracez les triangles et ii) donnez les énoncés OpenGL si on utilise la primitive `GL_TRIANGLE_FAN` ? **[2 points]**



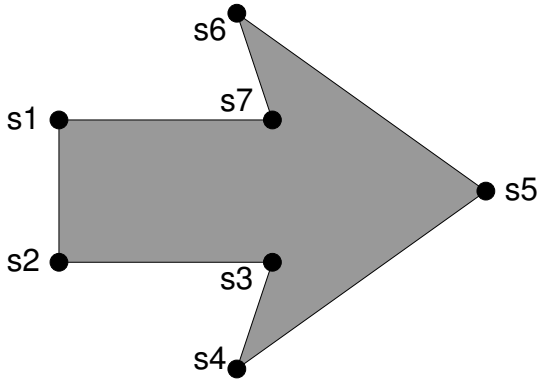
e) i) Tracez les polygones et ii) donnez les énoncés OpenGL si on utilise la primitive `GL_POLYGON` ? **[2 points]**



f) Pour l'affichage de cette forme, quelles sont les primitives les plus efficaces ? Pourquoi ? **[1 point]**

Question 55 Primitives OpenGL

Considérez cette forme pleine pouvant être tracée par différentes primitives OpenGL :

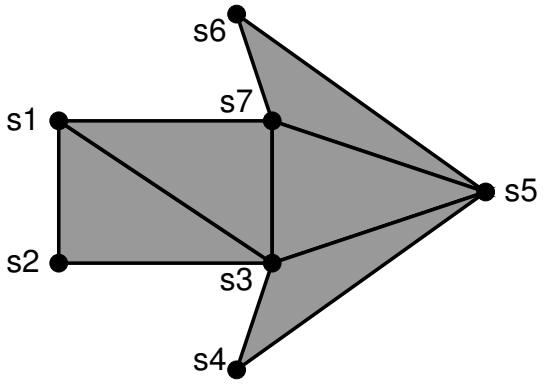


Dans les sous-questions suivantes, vous devez écrire les énoncés `glBegin()`, `glVertex*()` et `glEnd()` nécessaires pour tracer exactement et correctement la forme ci-dessus.

- Vous ne devez pas utiliser d'autres sommets que ceux existants (s1 à s7).
- Vous ne devez pas remplir ou tracer par-dessus ce qui est déjà rempli ou tracé.
- Vous devez utiliser un nombre minimal de paires `glBegin() ... glEnd()`, mais vous devez bien sûr les répéter lorsque nécessaire.
- Pour faire plus court dans la réponse, utilisez simplement « `S1` » au lieu de « `glVertex*(s1);` ».

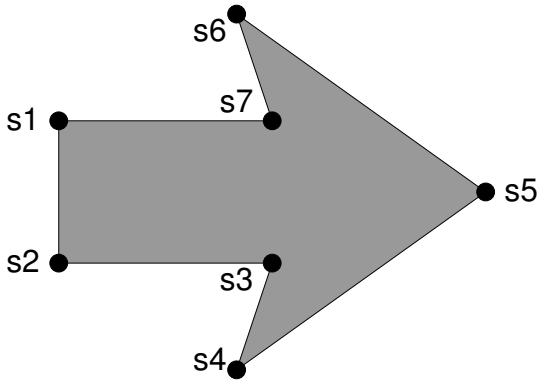
EXEMPLE :

i) **Tracez** les triangles et ii) **écrivez** les énoncés OpenGL si on utilise la primitive `GL_TRIANGLES`.

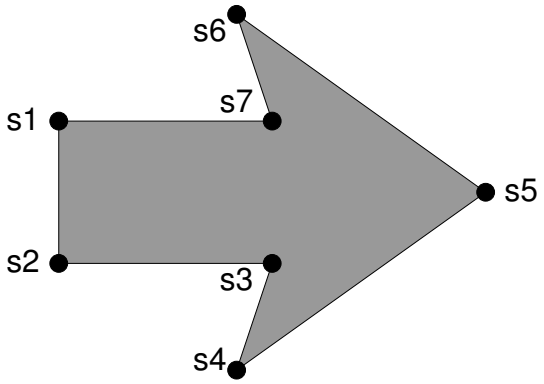



```
glBegin( GL_TRIANGLES );
  s1 s2 s3
  s1 s3 s7
  s3 s4 s5
  s3 s5 s7
  s5 s6 s7
glEnd( );
```

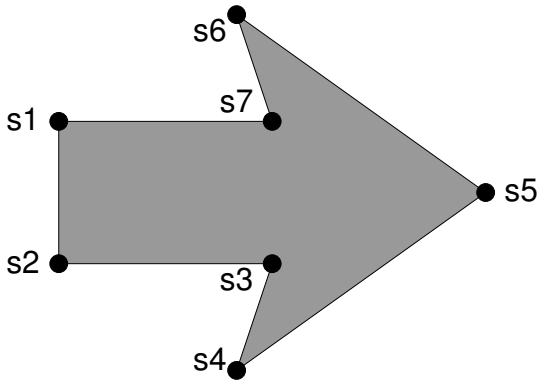
a) i) **Tracez** les quadrilatères et ii) **écrivez** les énoncés OpenGL si on utilise la primitive `GL_QUADS`. **[2 points]**



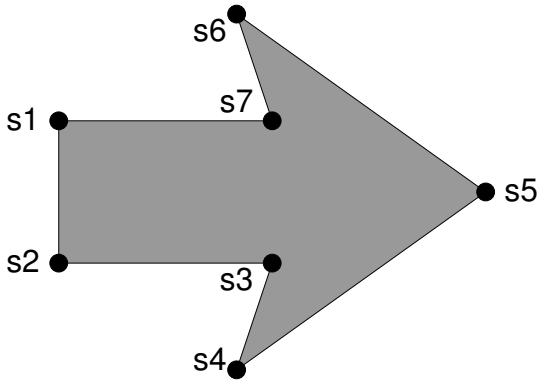
b) i) **Tracez** les quadrilatères et ii) **écrivez** les énoncés OpenGL si on utilise la primitive `GL_QUAD_STRIP`. **[2 points]**



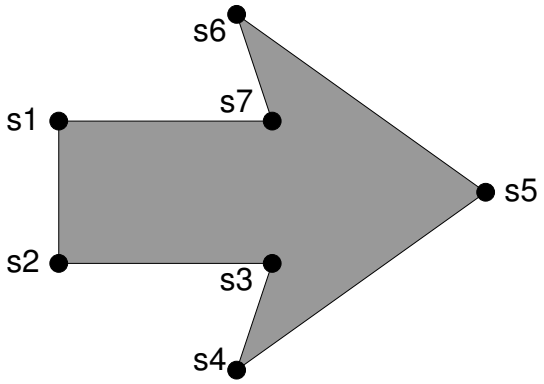
c) i) **Tracez** les triangles et ii) **écrivez** les énoncés OpenGL si on utilise la primitive `GL_TRIANGLE_STRIP`. **[2 points]**



d) i) **Tracez** les triangles et ii) **écrivez** les énoncés OpenGL si on utilise la primitive `GL_TRIANGLE_FAN`. **[2 points]**



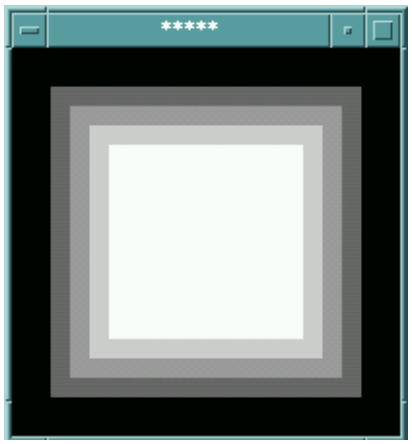
e) i) **Tracez** les polygones et ii) **écrivez** les énoncés OpenGL si on utilise la primitive `GL_POLYGON`. **[2 points]**



f) Pour l'affichage de cette forme, quelle est la primitive la plus efficace ? Pourquoi ? **[1 point]**

Question 56 Sélection 3D [12 points]

Considérez le programme donné à la page suivante. Ce programme crée une fenêtre, affiche un dessin en mode d'affichage, fait appel au mode de sélection d'OpenGL et montre finalement les résultats de la sélection à l'écran. La fonction `Affiche`, appelée deux fois par le programme principal, initialise le pipeline graphique et appelle la fonction `Dessine` pour dessiner 4 carrés imbriqués tel qu'illustré dans la figure ci-dessous.



a) Afin d'utiliser le mode de sélection, la fonction `glRenderMode` est appelée deux fois, une fois avec le paramètre GL_SELECT et une fois avec le paramètre GL_RENDER. À quoi correspond la valeur de retour de cette fonction lors du deuxième appel ? [1 point]

b) Dans ce programme, la fonction `gluPickMatrix` n'est pas utilisée et OpenGL sélectionne alors tout ce qui est visible dans la fenêtre. Notez que la fonction `Dessine` utilise un appel à la fonction `glPushName`, mais aucun à `glPopName`. Donnez (dans l'ordre) ce qui est écrit par les appels à `cout` à l'étape finale de ce programme. [5 points]

c) La fonction `gluPickMatrix` affecte le pipeline graphique en modifiant une matrice. Quelle matrice doit être modifiée ? [1 point]

d) La fonction `Pick` du programme utilise `gluPickMatrix` pour sélectionner une région carrée de 5 pixels autour d'un point (x,y) donné. Pour sélectionner l'objet ou les objets autour d'un point situé au centre de l'écran, c'est-à-dire au point (Largeur/2, Hauteur/2), on doit ajouter la ligne suivante dans la fonction `Affiche` :

```
if ( mode == GL_SELECT ) Pick( Largeur/2, Hauteur/2 );
```

Donnez les numéros de lignes dans la fonction Affiche entre lesquelles il faut ajouter cette ligne afin qu'elle remplisse sa tâche. [3 points]

e) Après insertion de la ligne tel qu'indiqué à la sous-question précédente, quel est alors le résultat des appels à `cout` à l'étape finale du programme ? [2 points]

```
static const unsigned int Largeur=200, Hauteur=200;

// Sélection d'une région carrée de 5 pixels autour d'un
// point (x,y) ( voir sous-question (d) )
static void Pick( GLdouble x, GLdouble y )
{
   GLint Viewport[4];
   glGetIntegerv( GL_VIEWPORT, Viewport );
   gluPickMatrix( x, y, 5, 5, Viewport );
}

// Dessin d'une série de carrés
static void Dessine( GLenum mode )
{
   for ( int i = 2 ; i <= 5 ; ++i ) // i varie de 2 à 5 incl.
   {
      glColor3f( .2*i, .2*i, .2*i ); // teinte de gris, de 0.4 à 1.0
      if ( mode == GL_SELECT ) glPushName( i );
      GLfloat x = 10 - i;
      glBegin( GL_QUADS );
      glVertex3f( -x, -x, -x );
      glVertex3f( -x, x, -x );
      glVertex3f( x, x, -x );
      glVertex3f( x, -x, -x );
      glEnd( );
      // "glPopName()" absent!
   }
   return;
}

// Initialisation du pipeline graphique et affichage
static void Affiche( GLenum mode )
{
   glViewport( 0, 0, Largeur, Hauteur );           // ligne 1
   glMatrixMode( GL_PROJECTION );                  // ligne 2
   glLoadIdentity( );                              // ligne 3
   glOrtho( -10, 10, -10, 10, 0, 20 );             // ligne 4
   glMatrixMode( GL_MODELVIEW );                   // ligne 5
   glLoadIdentity( );                              // ligne 6
   gluLookAt( 0, 0, 10, 0, 0, 0, 0, 1, 0 );        // ligne 7
   glClear( GL_COLOR_BUFFER_BIT );                 // ligne 8
   Dessine( mode );                                // ligne 9
   return;                                         // ligne 10
}


// Programme principal
int main( int argc, char **argv )
{
   // 1- Initialisation graphique
   initGraphique( ); // (ouvrir une fenêtre X)

   // 2- Affichage en mode normal
   Affiche( GL_RENDER );
```

```
   // 3- Sélection
   GLuint buf[500];
   glSelectBuffer( sizeof(buf), buf );
   glRenderMode( GL_SELECT );
   Affiche( GL_SELECT );
   GLint n = glRenderMode( GL_RENDER );

   // 4- Affichage du résultat de la sélection
   //        (le contenu de "buf")
   GLuint *ptr = buf;
   cout << "n=" << n << endl;                        // "cout"!
   for ( int i = 0 ; i < n ; ++i )
   {
      GLuint k = *ptr; ++ptr;
      cout << "k=" << k << ": ";                     // "cout"!
      ++ptr; ++ptr; // zmin et zmax sont ignorés
      for ( int j = 0 ; j < k ; ++j )
      { cout << " " << *ptr; ++ptr; }                // "cout"!
      cout << endl;                                  // "cout"!
   }
   return( 0 );
}
```

Question 57 Sélection 3D

Vous avez trouvé un programme qui affiche les chiffres d'un cadran. Une collègue veut modifier ce programme afin de pouvoir choisir l'heure de début et de fin d'un rendez-vous dans un agenda électronique affichant les chiffres de deux cadrans comme illustré dans la figure 1. Cette collègue a produit une nouvelle version du programme dans laquelle le mode de sélection d'OpenGL est déjà presque fonctionnel. Toutefois, les paramètres de visualisation ne sont pas très bien spécifiés et le programme, listé à la page suivante, affiche plutôt l'image de la figure 2. Malgré ce léger problème d'affichage, elle met en place le mode de sélection 3D d'OpenGL.

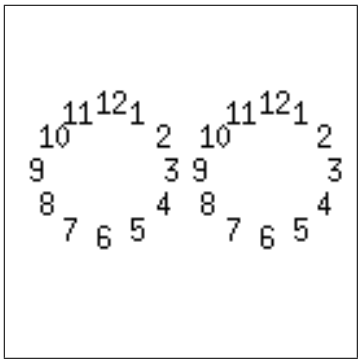


Figure 1 (ce qu'on souhaiterait voir)

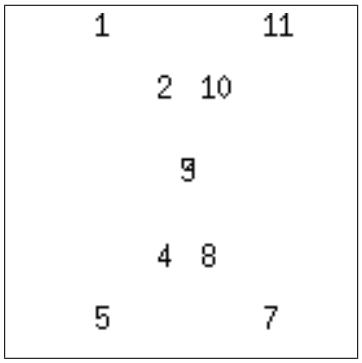


Figure 2 (ce que le programme affiche)

a) (i) Expliquez pourquoi elle doit appeler la fonction `glRenderMode` à deux reprises dans le programme et (ii) dites à quoi correspond la valeur de retour de cette fonction lors du second appel. **[2 points]**

b) Aucun appel à la fonction `gluPickMatrix` n'est utilisé pour le moment dans le programme et le programme sélectionne alors tout ce qui est visible dans la fenêtre (Figure 2). Donnez (dans l'ordre) ce qui est écrit par les appels à `cout` dans le programme principal. **[3 points]**

c) La fonction `gluPickMatrix` permet de définir une région rectangulaire autour d'un point donné. Quelle matrice du pipeline graphique doit être modifiée par cette fonction ? **[2 points]**

d) Pour sélectionner l'objet ou les objets dans une région de 3x3 pixels autour d'un point situé au centre de l'écran, on doit ajouter la ligne suivante :

```
if ( mode == GL_SELECT ) gluPickMatrix( Largeur/2, Hauteur/2, 3, 3, vp );
```

Donnez les numéros de lignes dans la fonction `Paint` entre lesquelles il faut ajouter cet appel à `gluPickMatrix`. **[2 points]**

```
static const unsigned int Largeur=130, Hauteur=130;

// Ouvrir une fenêtre et faire diverses initialisations
static void initGraphique( void );

// Afficher le texte donné à la position (x,y,z)
static void Heure( const char *texte, GLfloat x, GLfloat y, GLfloat z );

static void Cadran( GLfloat x, GLfloat y, GLfloat z )
{
   const int N=12;
   const GLfloat dAngle = 2*M_PI / N;
   char *c[N] = { "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12" };
   glPushMatrix( );
   glTranslatef( x, y, z );
   for ( int i = 1 ; i <= N ; ++i )
   {
      glPushName( i );
      Heure( c[i-1], 10*sin(i*dAngle), 10*cos(i*dAngle), 0 );
      glPopName( );
   }
   glPopMatrix( );
}

static void Paint( GLenum mode )
{
   GLint vp[4] = { 0, 0, Largeur, Hauteur };            // ligne 1
   glViewport( vp[0], vp[1], vp[2], vp[3] );            // ligne 2
   glMatrixMode( GL_PROJECTION );                       // ligne 3
   glLoadIdentity( );                                   // ligne 4
   gluOrtho2D( -4, 4, -4, 4 );                          // ligne 5
   glMatrixMode( GL_MODELVIEW );                        // ligne 6
   glLoadIdentity( );                                   // ligne 7
   gluLookAt( 0, 0, 1, 0, 0, 0, 0, 1, 0 );             // ligne 8
   glInitNames( );                                      // ligne 9
   glPushName( 100 );                                   // ligne 10
   glLoadName( 10 ); Cadran( -10, 0, 0 ); glPopName( ); // ligne 11
   glPushName( 20 ); Cadran(  10, 0, 0 ); glPopName( ); // ligne 12
   return;                                              // ligne 13
}

int main( int argc, char **argv )
{
   initGraphique( );
   glClearColor( 1, 1, 1, 1 );
   glClear( GL_COLOR_BUFFER_BIT );
   Paint( GL_RENDER );

   GLuint buf[500], *p = buf;
   glSelectBuffer( sizeof(buf), buf );
   glRenderMode( GL_SELECT );                   // premier appel à glRenderMode
   Paint( GL_SELECT );

   GLint n1 = glRenderMode( GL_RENDER );        // second appel à glRenderMode

   cout << "n1 = " << n1 << endl;                                  // "cout"!
   for ( int i = 0 ; i < n1 ; ++i )
   {
      GLuint n2 = *p; ++p;
      GLfloat zmin = *p / (GLfloat)0xffffffff; ++p;
      GLfloat zmax = *p / (GLfloat)0xffffffff; ++p;
      for ( int j = 0 ; j < n2 ; ++j ) { cout << " " << *p; ++p; } // "cout"!
      cout << endl;                                               // "cout"!
   }
   return( 0 );
}
```

Question 58 Sélection 3D

Durant votre stage chez *JeuRoleSoft*, vous êtes en charge de la sélection des objets dans l'interface usager de la version 2 du populaire jeu *Le monde des artisans de la guerre*. Malheureusement, suite à quelques problèmes techniques causés par une intrusion dans le système informatique de la compagnie, le code source de la première version pour la sélection est complètement disparu. Vous devez donc vous servir de vos connaissances acquises dans le cours INF2700 pour mettre au point le nouveau code de sélection. Décrivez tous les étapes requises pour implanter la sélection 3D dans ce jeu. **[3 points]**

Question 59 Sélection graphique

a) En cours d'implémentation de la sélection OpenGL dans votre application, vous souhaitez mieux comprendre ce qui se passe dans votre programme. Vous savez que 3 objets ont été sélectionnés par OpenGL et vous imprimez les 25 premières valeurs du tampon de sélection. Vous obtenez ce qui suit :

3, 3, 6, 41, 104, 4, 1, 2, 3, 5, 2, 1, 2, 23, 250, 5, 2, 9, 4, 8, 19, 22, 3, 2, 11, 7, 21, 31, ...

Quels sont les *noms* qui permettent d'identifier chacun des trois objets sélectionnés ? **[3 points]**

b) Considérez l'extrait suivant d'un petit programme utilisant la sélection graphique d'OpenGL. La fonction `Select` est appelée par le programme principal lorsqu'il s'agit de tracer la scène en mode sélection.

```
void Select( GLenum mode, GLdouble x, GLdouble y )  // ligne  1
{                                                   // ligne  2
   GLint cloture[4];                                // ligne  3
   glGetIntegerv( GL_VIEWPORT, cloture );           // ligne  4
   glMatrixMode( GL_MODELVIEW );                    // ligne  5
   glLoadIdentity( );                               // ligne  6
   gluLookAt( 0, 0, 1,  0, 0, 0,  0, 1, 0 );       // ligne  7
   glMatrixMode( GL_PROJECTION );                   // ligne  8
   glLoadIdentity( );                               // ligne  9
   gluPerspective( 35, 1, 1, 90 );                  // ligne 10
   gluPickMatrix( x, y, 3, 3, cloture );            // ligne 11
   glMatrixMode( GL_MODELVIEW );                    // ligne 12
   scene->dessine( GL_SELECT );                     // ligne 13
   return;                                          // ligne 14
}                                                   // ligne 15
```

Dans ces énoncés, est-ce que la fonction `gluPickMatrix` est placée au bon endroit ? Sinon, après quelle ligne devrait-on la mettre ? Justifiez votre réponse. **[2 points]**

c) Expliquez ce qui se passe si on omet l'appel à la fonction `gluPickMatrix` dans cette fonction `Select` lorsqu'on est en mode sélection ? **[2 points]**

d) Tel que vu en classe, on se rappelle que le prototype de la fonction `gluPickMatrix` est :

```
void gluPickMatrix( GLdouble x, GLdouble y,
                    GLdouble largeur, GLdouble hauteur, GLint cloture[4] );
```

La fonction `gluPickMatrix` est normalement appelée avec les coordonnées (x,y) du centre du rayon de sélection. Dans quel système de coordonnées (en quelles unités) sont exprimées (x,y) ? **[1 point]**

Question 60 Sélection graphique

Considérez la fonction ci-dessous qui fait partie d'un programme qui utilise la sélection graphique d'OpenGL. Cette fonction est appelée avec les coordonnées du point cliqué afin de déterminer les objets sélectionnés. Toutefois, vous constaterez certainement que les lignes numérotées sont dans le désordre !

```
void Selection( GLdouble x, GLdouble y )
{
      GLint cloture[4];
      glGetIntegerv( GL_VIEWPORT, cloture );
      scene->passerEnModeSelection();  // passer au mode sélection

   1  glFrustum( ... );
   2  glMatrixMode( GL_MODELVIEW );
   3  glMatrixMode( GL_PROJECTION );
   4  glLoadIdentity( );
   5  glPushMatrix( );
   6  glLoadIdentity( );
   7  glPushMatrix( );
   8  scene->consulterTampon( );  // déterminer les objets sélectionnés
   9  scene->dessinerSelect( );  // tracer la scène en donnant des noms aux objets
  10  gluPickMatrix( ... );
  11  gluLookAt( ... );
  12  scene->passerEnModeNormal();  // revenir du mode sélection

      glMatrixMode( GL_PROJECTION );
      glPopMatrix( );
      glMatrixMode( GL_MODELVIEW );
      glPopMatrix( );
      return;
}
```

a) Replacez les lignes 1 à 12 dans le bon ordre, sachant que les autres lignes sont correctement placées. *(Pour répondre, écrivez simplement les numéros de ligne dans le bon ordre.)* **[5 points]**

b) Écrivez les énoncés OpenGL de la méthode `passerEnModeSelection`. **[1 point]**

```
void Scene::passerEnModeSelection( void )
{
   ... // écrivez les énoncés
}
```

c) Écrivez les énoncés OpenGL de la méthode `passerEnModeNormal`. **[1 point]**

```
void Scene::passerEnModeNormal( void )
{
   ... // écrivez les énoncés
}
```

d) La méthode `consulterTampon` sert à déterminer les objets sélectionnés. Comment fait-on pour connaître le nombre d'objets sélectionnés ? **[1 point]**

e) Supposons que vous savez que trois objets ont été sélectionnés. Pour comprendre quoi faire dans la méthode `consulterTampon`, vous imprimez les vingt-cinq premières valeurs du tampon de sélection en vous disant que c'est sûrement assez pour voir toutes les valeurs pertinentes. Vous obtenez ces valeurs :

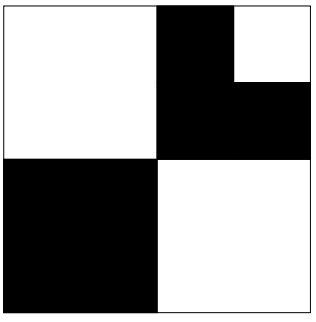
tampon = [3, 3, 57, 64, 0, 4, 2, 5, 33, 77, 5, 1, 0, 52, 12, 5, 0, 9, 14, 8, 19, 22, 3, 2, 11]

Quels sont les noms qui permettent d'identifier chacun des trois objets sélectionnés ? *Expliquez bien votre raisonnement.* **[3 points]**

Question 61 Texture

Considérez les énoncés ci-dessous qui chargent en mémoire l'image de droite afin de définir une texture. (Cette texture sera ensuite appliquée sur les primitives définies en (c) et (d).)

```
GLuint texid = ChargerImage( "Figure.png" );
glBindTexture( GL_TEXTURE_2D, texId );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP );
glEnable( GL_TEXTURE_2D );
```

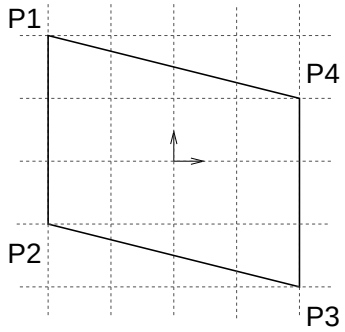


a) Expliquez l'effet de chacun des deux énoncés `glTexParameteri` présents dans l'extrait ci-dessus. **[2 points]**

b) Dessinez l'*espace de texture* associé à l'image chargée en mémoire selon ces énoncés. **[2 points]**

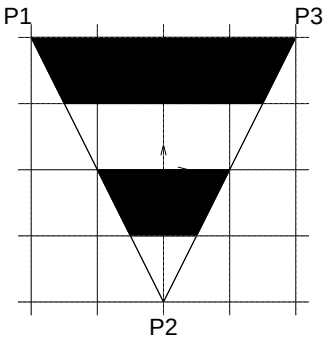
c) En utilisant les énoncés ci-dessous, on applique maintenant la texture définie auparavant à un quadrilatère. Dessinez le résultat visuel obtenu **et** expliquez comment vous arrivez à ce résultat. **[3 points]** (Annotez votre dessin au besoin afin d'être bien compris.)

```
glBegin( GL_QUADS );
glTexCoord2f(  3.0, 2.0 ); glVertex2f( -2.0,  2.0 ); // P1
glTexCoord2f(  0.0, 2.0 ); glVertex2f( -2.0, -1.0 ); // P2
glTexCoord2f(  0.0, 0.0 ); glVertex2f(  2.0, -2.0 ); // P3
glTexCoord2f(  3.0, 0.0 ); glVertex2f(  2.0,  1.0 ); // P4
glEnd( );
```



d) Quelles sont les coordonnées de texture à utiliser dans chacun des trois appels à `glTexCoord2f` afin de produire le triangle texturé de droite ? Commentez vos choix. **[3 points]**

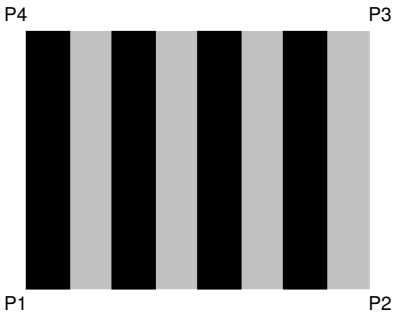
```
glBegin( GL_TRIANGLES );
glTexCoord2f( ???, ??? ); glVertex2f( -2.0,  2.0 );  // P1
glTexCoord2f( ???, ??? ); glVertex2f(  0.0, -2.0 );  // P2
glTexCoord2f( ???, ??? ); glVertex2f(  2.0,  2.0 );  // P3
glEnd( );
```



Question 62 Texture

Considérez les énoncés et l'image ci-dessous. Ces énoncés chargent en mémoire une texture "Image.png", initialisent certains paramètres d'OpenGL et affichent un seul quadrilatère sur lequel la texture est appliquée pour produire le résultat visuel montré.

```
GLuint texid = ChargerTexture( "Image.png" );
glBindTexture( GL_TEXTURE_2D, texId );
glEnable( GL_TEXTURE_2D );

glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, ??? );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, ??? );

glBegin( GL_QUADS );
glTexCoord2f( ???, ??? ); glVertex2f( -2.0, -2.0 ); // P1
glTexCoord2f( ???, ??? ); glVertex2f(  2.0, -2.0 ); // P2
glTexCoord2f( ???, ??? ); glVertex2f(  2.0,  2.0 ); // P3
glTexCoord2f( ???, ??? ); glVertex2f( -2.0,  2.0 ); // P4
glEnd( );
```



a) Le fichier "Image.png" contient l'image utilisée comme texture.

i) Dessinez l'image qui serait *la plus petite image possible* utilisable avec ces énoncés.

ii) Quelle est la taille (largeur x hauteur) en pixels de cette image ?

[3 points]

b) Que doit-on alors utiliser à la place des « `???` » dans chacun des deux appels à `glTexParameteri` présents dans les énoncés ci-dessus. *Justifiez votre réponse.* **[2 points]**

c) Quelles coordonnées de texture doit-on utiliser à la place des « `???` » dans chacun des quatre appels à `glTexCoord2f` afin de produire le résultat visuel qui est montré ci-dessus ? *Justifiez votre réponse.* **[4 points]**

Question 63 Texture

a) Les énoncés ci-dessous chargent en mémoire l'image de droite afin de créer une texture.

```
GLuint texid = LireEtCreerTexture( "ABCD.png" );
```

```
glBindTexture( GL_TEXTURE_2D, texId );
glEnable( GL_TEXTURE_2D );
```

Dessinez l'*espace de texture* associé à cette texture ...

i) ... si on utilise ces énoncés :

```
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT );
```

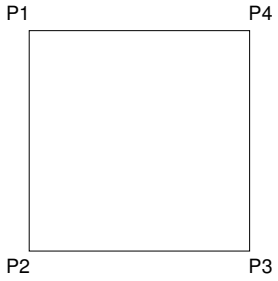
ii) ... si on utilise ces énoncés :

```
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP );
```

[2 points]

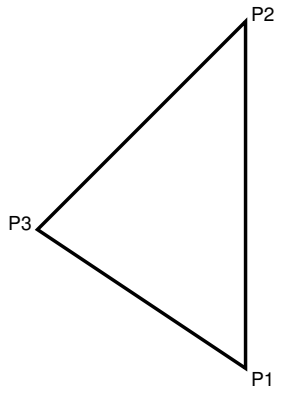
b) Dessinez le résultat visuel du rectangle texturé avec l'image chargée à la sous-question (a) lorsqu'on utilise les énoncés ci-dessous. **[3 points]**

```
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT );

GLfloat sommets[] = { -5.0,  5.0,   // P1
                      -5.0, -5.0,   // P2
                       5.0, -5.0,   // P3
                       5.0,  5.0    // P4
                    };
GLfloat texcoords[] = { -1.5, 1.5,
                        -1.5, 0.5,
                         0.5, 0.5,
                         0.5, 1.5
                      };

glVertexPointer( 2, GL_FLOAT, 0, sommets );
glTexCoordPointer( 2, GL_FLOAT, 0, texcoords );
glDrawArrays( GL_QUADS, 0, 4 );
```



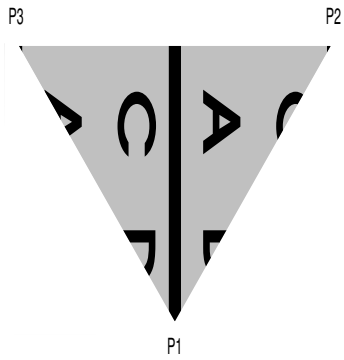
c) Dessinez le résultat visuel du triangle texturé avec l'image chargée à la sous-question (a) lorsqu'on utilise les énoncés ci-dessous. **[3 points]**

```
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP );

GLfloat sommets[] = {  4.0,  1.0,   // P1
                       4.0,  6.0,   // P2
                       1.0,  3.0    // P3
                    };
GLfloat texcoords[] = {  1.0, 0.0,
                         0.0, 0.0,
                         0.6, 0.6
                       };

glVertexPointer( 2, GL_FLOAT, 0, sommets );
glTexCoordPointer( 2, GL_FLOAT, 0, texcoords );
glDrawArrays( GL_TRIANGLES, 0, 3 );
```



d) Quelles sont les 6 valeurs à utiliser dans le tableau `texcoords` ci-dessous afin de produire ce triangle texturé avec l'image chargée à la sous-question (a) ? **[3 points]**

```
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT );

GLfloat sommets[] = {  0.0, -5.0,   // P1
                       5.0,  5.0,   // P2
                      -5.0,  5.0    // P3
                    };
GLfloat texcoords[] = { ???, ???,
                        ???, ???,
                        ???, ???
                      };

glVertexPointer( 2, GL_FLOAT, 0, sommets );
glTexCoordPointer( 2, GL_FLOAT, 0, texcoords );
glDrawArrays( GL_TRIANGLES, 0, 3 );
```



Question 64 Transformations 2D [6 points]

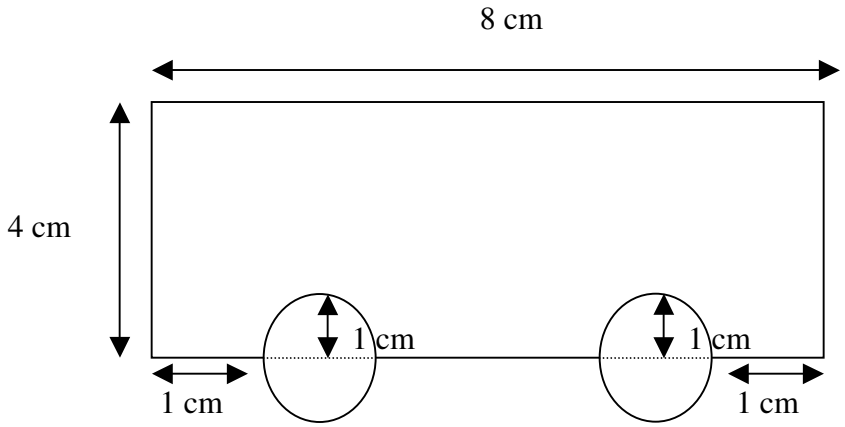
Soit un rectangle décrit par quatre sommets dont les coordonnées cartésiennes sont : P1 = (2, 1), P2 = (5, 1), P3 = (5, 5) et P4 = (2, 5). Ce rectangle tourne en 2D autour de son centre de gravité à une vitesse d'un quart de tour par seconde. Pour afficher une animation vidéo, il est souhaitable de générer au moins 30 images par seconde et, en conséquence, on cherchera à être le plus efficace possible dans l'affichage des images.

a) Décrivez la série de transformations géométriques, la valeur de leurs paramètres ainsi que les matrices correspondantes qui permettent de réaliser cette animation. [4 points]

b) Proposez une implémentation efficace (en temps d'affichage) dans OpenGL pour afficher une séquence d'animation. [2 points]

Question 65 Transformations 2D [6 points]

La représentation 2D, à une échelle réduite, d'un véhicule simple se compose d'un rectangle de dimensions 4cmx8cm (pour représenter la carrosserie) et de deux cercles de 2cm de diamètre (pour représenter deux roues) qui sont positionnées sur la carrosserie selon la figure ci-dessous.

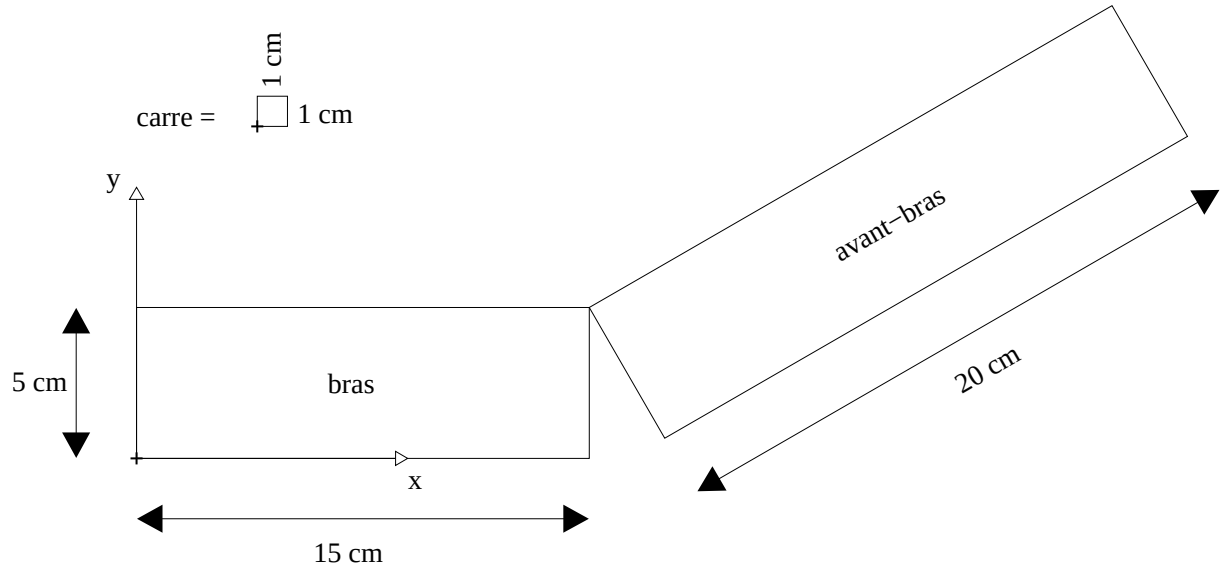


a) En considérant que l'origine du repère se trouve au coin inférieur gauche de la carrosserie donnez le code OpenGL qui décrit une structure hiérarchique de liste d'affichage pour afficher ce véhicule de façon efficace. Donnez aussi le code OpenGL pour les différentes sous-listes d'affichage ainsi que les fonctions utilisées. [3 points]

b) Étant donné une droite d'équation y = - x + 2, décrivez la série de transformations (directes et inverses) géométriques, la valeur de leurs paramètres ainsi que les matrices correspondantes qui permettent d'obtenir la réflexion du véhicule par rapport à cette droite. On ne veut pas les appels aux fonctions OpenGL. [3 points]

Question 66 Transformations 2D [12 points]

La représentation 2D simplifiée d'un bras de robot se compose d'un rectangle de 5cm x 15cm (pour représenter le bras) et d'un rectangle de 5cm x 20cm (pour représenter l'avant-bras) comme l'indique la figure ci-dessous. Le bras forme un angle de 30 degrés avec l'avant-bras.



Dans le programme listé à la page suivante, la fonction "main" initialise d'abord le point de vue, puis construit une liste d'affichage "carre" et appelle finalement la fonction Dessine. Dans les sous-questions qui suivent, on vous demande les énoncés OpenGL qui doivent être mis dans la fonction Dessine. Vous devez donner tous les appels à la liste d'affichage "carre" (glCallList), toutes les transformations de modélisation (glTranslate, glScale, glRotate) et toutes les sauvegardes de matrice (glPushMatrix et glPopMatrix). Notez que l'origine du repère se situe au coin inférieur gauche du bras du robot et que les transformations de modélisation d'une partie de la fonction Dessine peuvent affecter une partie subséquente.

a) Donnez les énoncés de la première partie de la fonction Dessine qui trace *le bras seulement*.
[4 points]

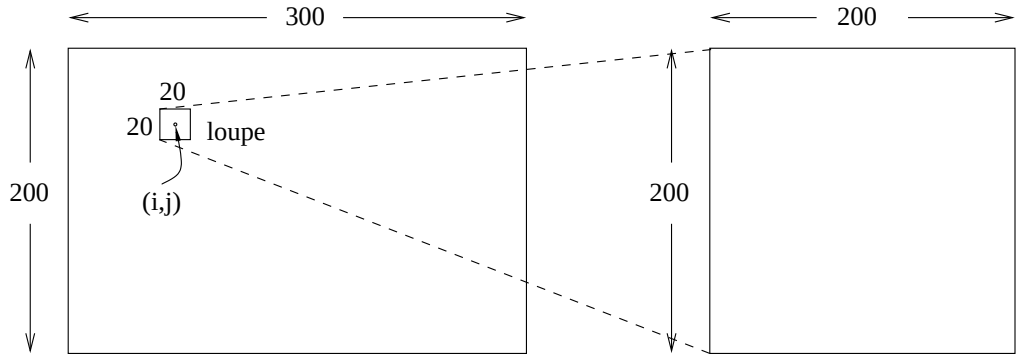
b) Donnez les énoncés de la deuxième partie de la fonction Dessine qui fait *la rotation*.
[4 points]

c) Donnez les énoncés de la troisième partie de la fonction Dessine qui trace *l'avant-bras*.
[4 points]

(Vous *devez* utiliser la liste d'affichage "carre" pour tracer les objets graphiques.)

```
void Dessine( GLuint carre, GLfloat angle )
{
   // a) afficher le bras
   ...

   // b) faire la rotation
   ...

   // c) afficher l'avant-bras
   ...
}

int main( int argc, char **argv )
{
   ...
   // Initialisation du pipeline graphique
   ( glViewport, glOrtho, gluLookAt, glClear, ...
     et autres fonctions qu'on ne vous demande pas d'écrire! )

   // Création de la liste d'affichage ``carre''
   GLuint carre = glGenLists( 1 );
   glNewList( carre, GL_COMPILE );
   glBegin( GL_LINE_LOOP );
   glVertex2f( 0, 0 );
   glVertex2f( 0, 1 );
   glVertex2f( 1, 1 );
   glVertex2f( 1, 0 );
   glEnd( );
   glEndList( );

   // Appel à ``Dessine''
   Dessine( carre, 30 );

   // C'est tout!
   glXSwapBuffers...;
   ...
   return( 0 );
}
```

Question 67 Transformations 2D



Ayant écrit avec succès un programme avec OpenGL qui dessine dans une fenêtre une certaine vue sur un monde virtuel, vous souhaitez maintenant ajouter la fonctionnalité d'une loupe à votre application, tel que la figure ci-dessus l'illustre. Vous avez défini la clôture et la projection de la fenêtre originale (celle de gauche) en utilisant les appels OpenGL suivants :

```
glViewport( ox=0, oy=0, largeur=300, hauteur=200 );
gluOrtho2D( gauche=10, droite=40, bas=30, haut=50 );
```

aux endroits appropriés dans le programme. Le point (i,j) sera déterminé interactivement par l'utilisateur dans la fenêtre originale (à gauche) et vous devrez alors afficher le contenu d'une petite région rectangulaire de 20 x 20 pixels autour de ce point dans la seconde fenêtre (à droite). Le point (i,j), exprimé en pixels, du centre de cette loupe (dans la fenêtre de gauche) sera ainsi tel que $10 < i < 290$ et $10 < j < 190$.

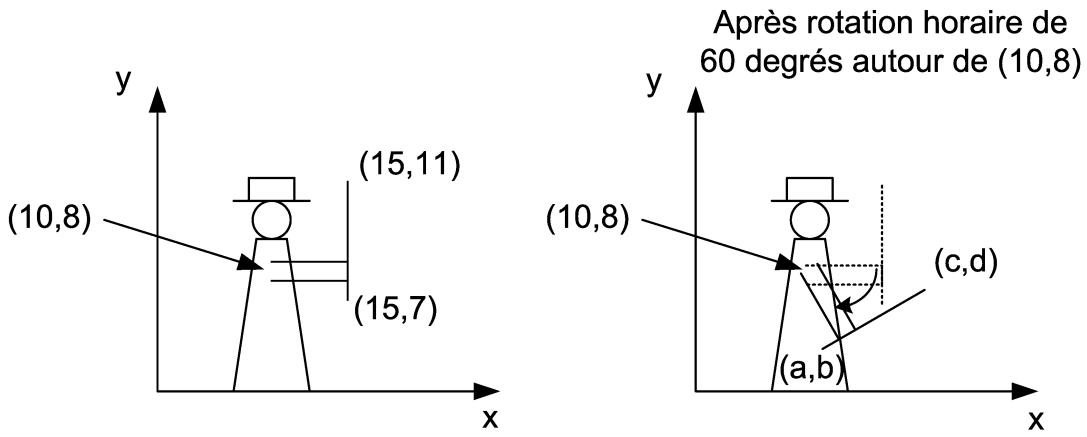
a) Dans le but de connaître le centre de la loupe dans le monde virtuel, donnez les fonctions qui permettent de connaître à quel point (x,y) du monde virtuel correspond le point (i,j) ? (Donnez la réponse en termes de i et j seulement, c'est-à-dire x(i,j) = ... et y(i,j) = ...) **[4 points]**

b) Les dimensions de la seconde fenêtre sont initialement de 200 x 200 pixels et lorsque vous affichez le contenu de la loupe dans cette seconde fenêtre, vous désirez occuper exactement tout l'espace de cette fenêtre. Bien sûr, il faut respecter le rapport d'aspect de la fenêtre originale et ne pas déformer l'image résultante dans la seconde fenêtre. Que doivent alors être les paramètres des deux fonctions `glViewport` et `gluOrtho2D` pour la seconde fenêtre ? (Donnez la réponse en termes de i et j : `glViewport( ?, ?, ?, ? );  gluOrtho2D( ?, ?, ?, ? );`) **[4 points]**

Question 68 Transformations 2D

Dans votre tout dernier jeu favori, *Le monde des artisans de la guerre* par *JeuRoleSoft*, vous désirez fracasser une porte en assenant un coup avec le bâton de votre personnage, *un troll de la pleine lune*. Le

coup est assené par une rotation du bras du personnage autour de son épaule située au point (10, 8). La position initiale du bâton est de (15, 7) pour une de ses extrémités et de (15, 11) pour l'autre (voir figure ci-dessous). Une rotation de 60 degrés dans le sens horaire est requise pour assener le coup.



a) Quels sont les valeurs (a,b,c,d) des points aux extrémités du bâton après la rotation ? **[4 points]**
Indice : $\begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$

b) Arrondissez à l'entier le plus près les coordonnés obtenues en a) et appliquez l'algorithme de Bresenham pour déterminer les pixels du bâton qui seront allumés. Pour chaque itération de l'algorithme de Bresenham, calculez la valeur de d et indiquez le pixel allumé. **[4 points]**
Indice :
initialisation : $d = 2 * dy - dx$,
pour l'EST : $d = d + 2 * dy$,
pour le NORD-EST : $d = d + 2 * (dy - dx)$

c) Est-ce que l'algorithme de Bresenham cause du crénelage ? Justifiez votre réponse. **[1 point]**

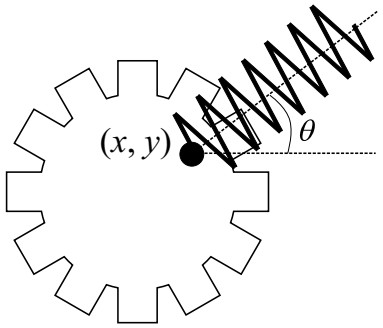
Question 69 Transformations 2D

a) La compagnie *Rolllex* désire écrire un logiciel OpenGL permettant de modéliser et d'animer l'intérieur d'une montre composée de plusieurs engrenages. Le stagiaire qui vous a précédé a déjà écrit la fonction `engrenage( int n )` qui fait tous les appels à OpenGL pour dessiner un engrenage de rayon 1 comptant n dents et centré à la position $(1, 1, 0)$. Pourquoi aurait-il été préférable de dessiner cet engrenange centré à l'origine ? **[1 point]**

b) Vos délais de production sont très serrés et le code du stagiaire est illisible ! Écrivez une fonction `engrenage2( int n, GLfloat r )` qui fait appel à la fonction du stagiaire mais qui dessine un engrenage de rayon r centré à l'origine. Vous pouvez supposer que la matrice courante est `GL_MODELVIEW`. **[2 points]**

c) Croyez-vous que les engrenages devraient être organisés hiérarchiquement de façon à ce que les transformations d'un engrenage parent affectent ses engrenages enfants ? Pourquoi ? **[1 point]**

d) Un ressort de longueur 1 est attaché à un engrenage de la montre. La fonction `ressort()` fait tous les appels à OpenGL permettant de dessiner un ressort ayant un point d'attache en $(0, 0, 0)$ et un autre en $(1, 0, 0)$. Écrivez la fonction `engrenageARessort(`n`,`r`,`x`,`y`,`θ`)` qui permet de dessiner un engrenage de rayon r à n dents et le ressort qui y est attaché. La position (x, y) indique le point d'ancrage du ressort dans le référentiel de l'engrenage, tandis que θ indique l'angle de rotation du ressort (voir la figure suivante). **[3 points]**



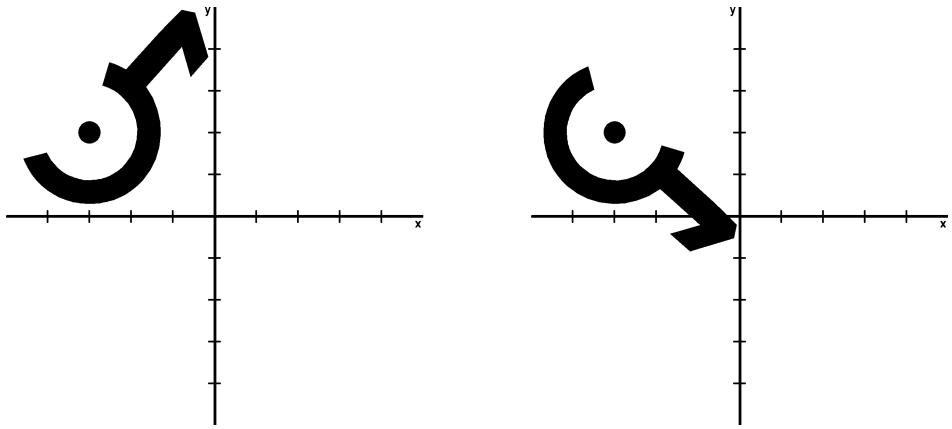
e) On veut plutôt utiliser `engrenageARessort(`n`,`r`,`x_1`,`y_1`,`x_2`,`y_2`)`. Ici, (x_1, y_1) et (x_2, y_2) indiquent la position des deux points d'ancrage du ressort dans le référentiel de l'engrenage. Le ressort ne sera donc plus toujours de longueur 1. Écrivez cette nouvelle fonction. **[3 points]**

Question 70 Transformations affines 2D

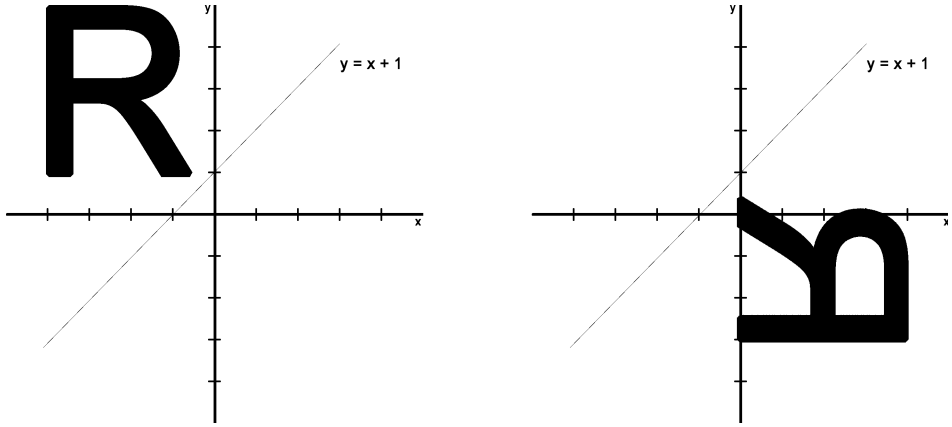
Voici un ensemble de 2 patrons définis en coordonnées d'objet dans l'image de gauche. Chacun de ces patrons ont subi une série de n transformations élémentaires affines 2D dont le résultat final est affiché dans l'image de droite. Ces transformations élémentaires peuvent être de type translation $T(d_x, d_y)$, rotation par rapport à l'origine $R(\theta)$ ou mise à l'échelle $S(s_x, s_y)$.

Pour chaque patron :
- Décrivez textuellement les n différentes étapes de transformation (en donnant également la valeur des paramètres) ;
- Écrivez les matrices 3x3 correspondantes à chacune des n étapes ; et
- Écrivez l'équation matricielle $p' = Mp$ en détaillant M en fonction des matrices $T(d_x, d_y)$, $R(\theta)$ et $S(s_x, s_y)$ de façon à faire ressortir leur ordre matriciel.

a) Patron 1 : $n = 3$



b) Patron 2 : $n = 5$



Question 71 Transformations affines

La librairie GLUT fournit la fonction `glutWireSphere(1.0,15,15)` qui permet de tracer une sphère de rayon 1.0 centrée en $(0, 0, 0)$ tel qu'illustré à la figure 1. En utilisant cette fonction avec ces arguments, ainsi que d'autres fonctions de base d'OpenGL comme `glRotatef()`, `glScalef()` et `glTranslatef()`, écrivez la méthode qui trace l'ellipsoïde de la figure 2 *à la position* et aussi *dans l'orientation* illustrées. Observez l'orientation des lignes en fil de fer !

Notez que le rayon de l'ellipsoïde selon l'axe Y est 2.0, tandis que le rayon selon les axes X et Z est 1.0. (N'utilisez pas de transformations inutiles. Ne faites pas compliqué lorsque vous pouvez faire simple !)

[4 points]

```
void traceEllipsoide( )
{
  ...
  glutWireSphere( 1.0, 15, 15 );
  ...
}
```

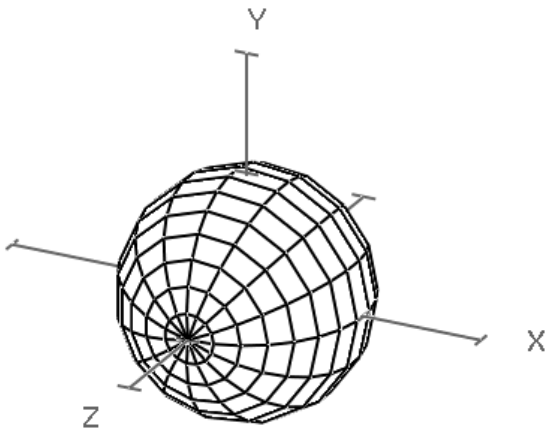


Figure 1 : sphère

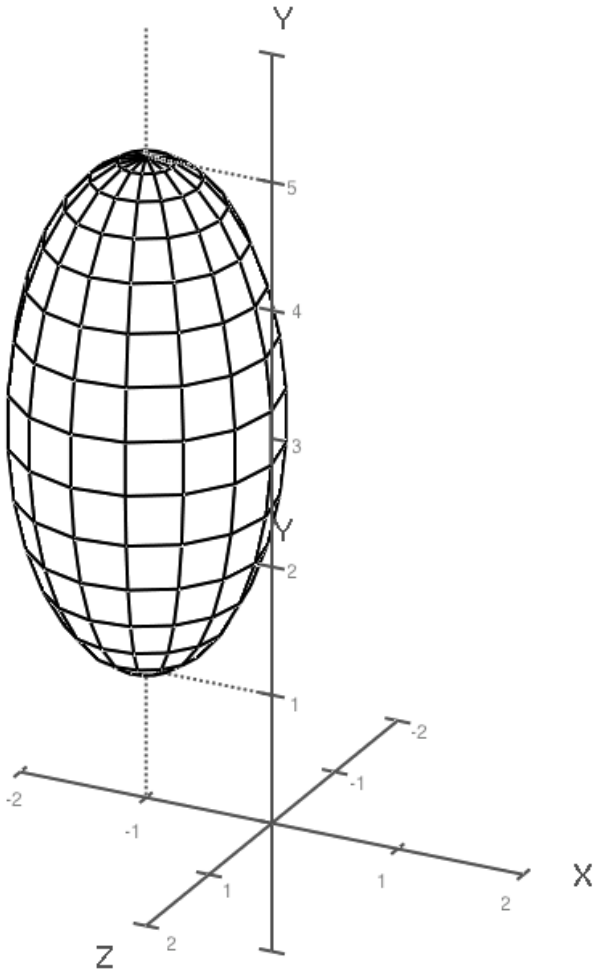
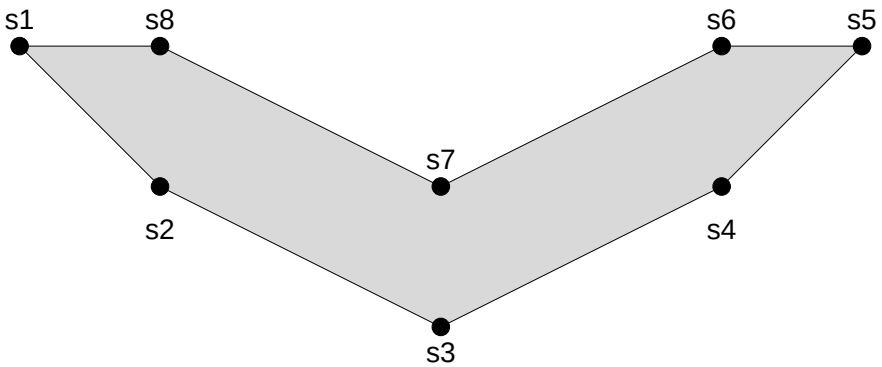


Figure 2 : ellipsoïde

Question 72 Primitives OpenGL

Considérez cette forme pleine tracée avec les énoncés OpenGL qui suivent :




```
glBegin( GL_TRIANGLES );
  glVertex2iv( s1 ); glVertex2iv( s2 ); glVertex2iv( s8 );
  glVertex2iv( s2 ); glVertex2iv( s7 ); glVertex2iv( s8 );
  glVertex2iv( s2 ); glVertex2iv( s3 ); glVertex2iv( s7 );
  glVertex2iv( s3 ); glVertex2iv( s4 ); glVertex2iv( s7 );
```

```
  glVertex2iv( s7 ); glVertex2iv( s4 ); glVertex2iv( s6 );
  glVertex2iv( s4 ); glVertex2iv( s5 ); glVertex2iv( s6 );
glEnd( );
```

Dans les sous-questions suivantes, on vous demande de donner les énoncés `glVertex2iv` nécessaires pour tracer exactement et correctement la même forme, mais en utilisant d'autres primitives OpenGL. Quelques règles à respecter :

- vous ne devez pas utiliser d'autres sommets que ceux existants (s1 à s8) ;
- vous devez utiliser le nombre minimal de paires `glBegin() ... glEnd()`.
- vous ne devez pas remplir ou tracer par-dessus ce qui est déjà rempli ou tracé.

Pour faire une réponse plus courte, plutôt que d'écrire l'énoncé `glVertex2iv(`s_i`)` au complet, vous pouvez n'indiquer que le numéro du sommet s_i. Ainsi, au lieu des énoncés donnés en exemple, on peut écrire :

```
glBegin( GL_TRIANGLES );
  s1 s2 s8   s2 s7 s8   s2 s3 s7   s3 s4 s7   s7 s4 s6   s4 s5 s6
glEnd( );
```

a) Quel doit être l'ordre des énoncés si on utilise la primitive `GL_QUADS` ? **[2 points]**

b) Quel doit être l'ordre des énoncés si on utilise la primitive `GL_QUAD_STRIP` ? **[2 points]**

c) Quel doit être l'ordre des énoncés si on utilise la primitive `GL_TRIANGLE_STRIP` ? **[2 points]**

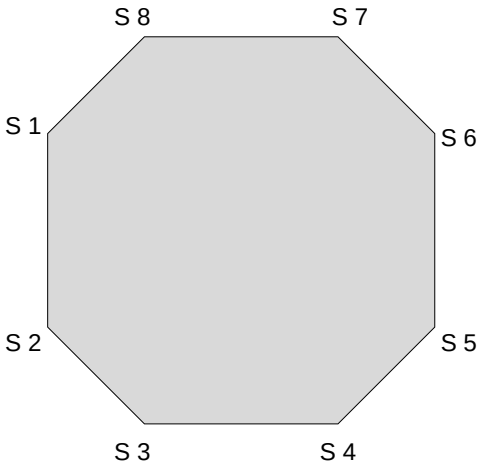
d) Quel doit être l'ordre des énoncés si on utilise la primitive `GL_TRIANGLE_FAN` ? **[2 points]**

e) Quel doit être l'ordre des énoncés si on utilise la primitive `GL_POLYGON` ? **[2 points]**

f) Pour l'affichage de cette forme, quelles sont les primitives les plus efficaces ? Pourquoi ? **[1 point]**

Question 73 Transformations affines et primitives OpenGL

Considérez les sommets de l'octogone illustré, qu'on peut tracer avec les énoncés OpenGL ci-dessous :




```
glBegin( GL_POLYGON );
  glVertex2iv( s1 );
  glVertex2iv( s2 );
  glVertex2iv( s3 );
  glVertex2iv( s4 );
  glVertex2iv( s5 );
  glVertex2iv( s6 );
  glVertex2iv( s7 );
  glVertex2iv( s8 );
glEnd( );
```

Dans les sous-questions suivantes, on vous demande de donner les énoncés `glVertex2iv` nécessaires pour tracer exactement et correctement la même surface, mais en utilisant d'autres primitives OpenGL. Quelques règles à respecter : i) vous ne devez pas utiliser d'autres sommets que s1 à s8 ; ii) vous ne devez pas remplir / tracer par-dessus ce qui est déjà rempli / tracé. Pour faire une réponse plus courte, vous pouvez n'indiquer que le numéro du sommet s_i plutôt que d'écrire l'énoncé `glVertex2iv(`s_i`)` au complet.

a) Que doit être l'ordre des énoncés si on utilise la primitive GL_QUADS ? **[2 points]**

b) Que doit être l'ordre des énoncés si on utilise la primitive GL_QUAD_STRIP ? **[2 points]**

c) Que doit être l'ordre des énoncés si on utilise la primitive GL_TRIANGLES ? **[2 points]**

d) Que doit être l'ordre des énoncés si on utilise la primitive GL_TRIANGLE_STRIP ? **[2 points]**

e) Que doit être l'ordre des énoncés si on utilise la primitive GL_TRIANGLE_FAN ? **[2 points]**

f) Pour l'affichage de cet octogone, quelle(s) primitive(s) semble(nt) être la(les) plus efficace(s) ? Pourquoi ? **[2 points]**

Question 74 Transformations 3D et projections [10 points]

Soit un dé à jouer traditionnel (la somme des faces opposées donne 7).

a) On veut transformer le dé (unitaire) de la figure 1 (6 en haut, 5 devant) pour qu il se retrouve comme indiqué sur la figure 2 (5 en haut, 6 devant). Décrivez les transformations à effectuer pour y arriver. Donnez également les instructions OpenGL correspondantes en supposant que vous avez une liste d'affichage Le_Dé qui vous dessine un dé à jouer (n'oubliez pas d'inclure les appels pour indiquer à OpenGL que vous désirez modifiez une de ses piles de matrice). [5 points]

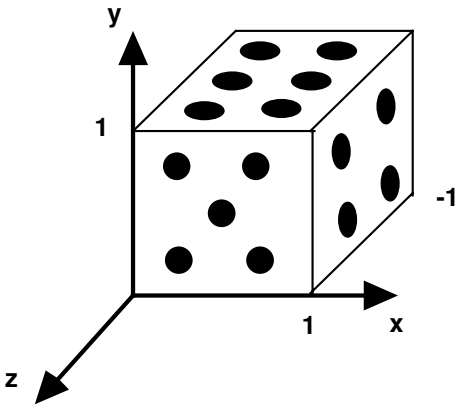


Figure 1

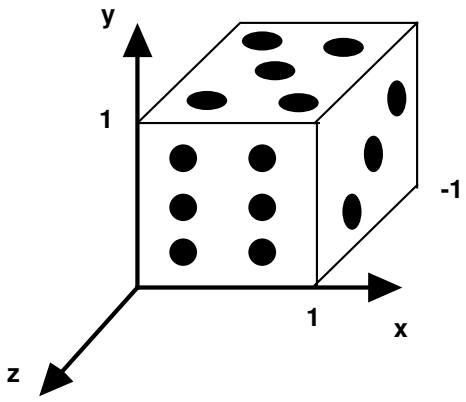


Figure 2

b) Qu'est ce qu'un "point de fuite" dans une image ? Une projection orthogonale contient-elle des points de fuite ? À quoi correspond le nombre de points de fuite d'une image (projection) ? [4 points]

c) On projette notre dé en utilisant une projection perspective. On obtient l'image de la figure 3. En observant cette image, pouvez-vous indiquer le nombre de points de fuite induits par la projection ? [1 point]

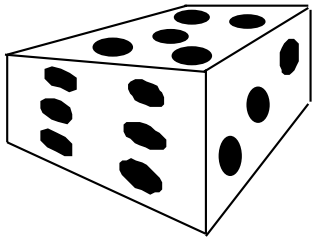
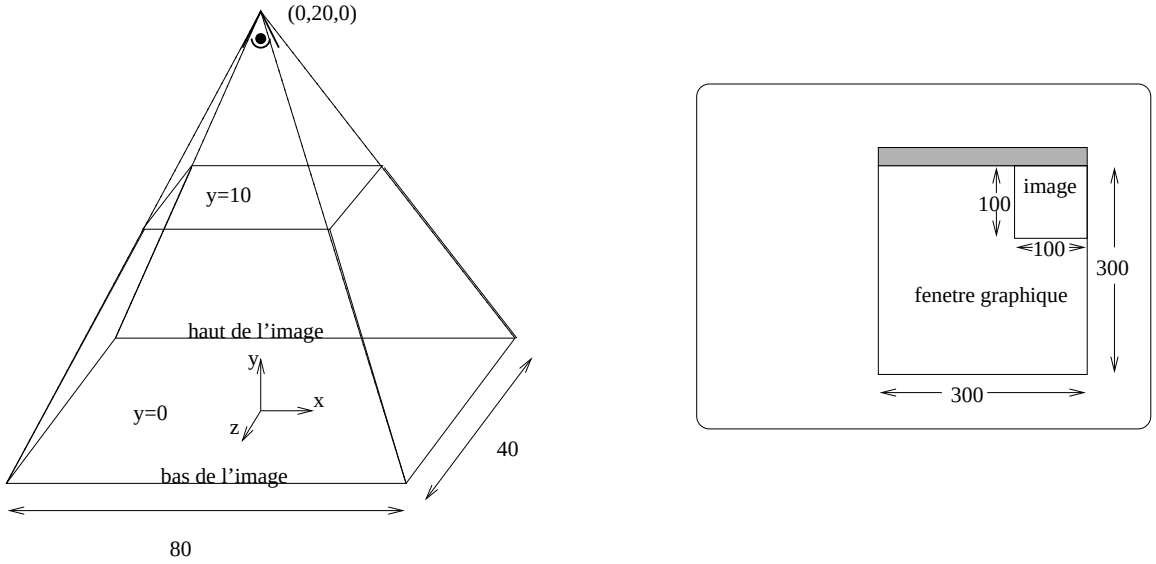


Figure 3

Question 75 Visualisation 3D [9 points]

Considérez la situation suivante où on souhaite obtenir une vue de plan d'une scène quelconque. Pour ce faire, on place un observateur aux coordonnées (0,20,0) qui regarde directement vers le bas (direction -y). Son volume de vision est délimité par les plans y=10, y=0 ainsi que les quatre plans passant par (0,20,0) et chacun des quatre côtés du rectangle de dimension 80x40 centré à l'origine. Le haut de l'image souhaitée ainsi que les paramètres ci-dessus sont indiqués à la figure ci-dessous.



Figure

Dans une fenêtre graphique (un objet de la classe QGLWidget) de dimension 300x300 de notre application Qt, nous souhaitons que l'image produite apparaisse au coin supérieur droit et soit de dimension 100x100 (voir la figure ci-dessus).

a) Utilisez certaines des fonctions glFrustum(), gluLookAt(), glOrtho(), gluPerspective() et glViewport() afin de produire l'image souhaitée. Pour cette sous-question nous ne voulons que les appels de fonction avec la valeur de leurs paramètres. Il n'est pas nécessaire de placer ces paramètres dans le bon ordre mais donnez la signification de chacun d'eux. [6 points]

b) Identifiez les deux matrices maintenues par OpenGL pour la visualisation 3D et dites pour chacune des fonctions utilisées en (a) laquelle de ces matrices est affectée. [2 points]

c) Avec les spécifications actuelles, la scène apparaîtra déformée à l'écran. Suggérez une modification à la réponse de (a) afin de préserver le rapport d'aspect. [1 point]

Question 76 Visualisation 3D [10 points]

Le modèle de la caméra synthétique est utilisé pour visualiser des scènes en 3D et nécessite la spécification de quelques paramètres. Certaines des fonctions `glFrustum()`, `glLoadIdentity()`, `glMatrixMode()`, `glOrtho()`, `glViewport()`, `gluLookAt()`, `gluOrtho2D()` et `gluPerspective()` sont utilisées pour définir ces paramètres.

a) Laquelle ou lesquelles de ces fonctions ser(ven)t à spécifier les dimensions de la photo ? [1 point]

b) Laquelle ou lesquelles de ces fonctions ser(ven)t à spécifier la position de la caméra ? [1 point]

c) Laquelle ou lesquelles de ces fonctions ser(ven)t à spécifier l'orientation de la caméra ? [1 point]

d) Laquelle ou lesquelles de ces fonctions ser(ven)t à travailler en vue perspective ? [1 point]

e) Laquelle ou lesquelles de ces fonctions ser(ven)t à délimiter le volume photographié ? [1 point]

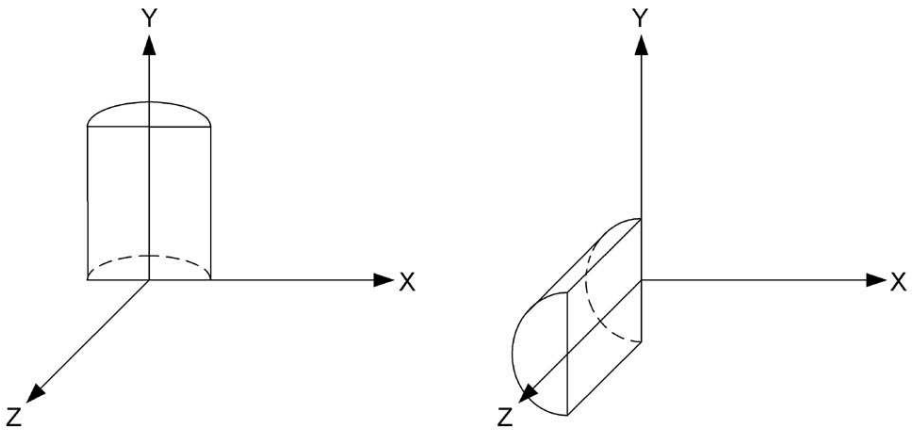
Pour les deux sous-questions suivantes, supposons qu'une certaine fonction `Dessine()` permet d'afficher une pièce dont les dimensions sont de 8m x 6m x 4m (largeur x profondeur x hauteur). Dans cette fonction, on considère que l'origine est au sol au centre de la pièce et que les largeur, profondeur et hauteur correspondent respectivement à des variations en X de -4 à 4 (de gauche à droite), en Y de -3 à 3 (d'avant à arrière) et en Z de 0 à 4 (de bas en haut). De plus, cette pièce possède une fenêtre de 2m x 2m au centre du mur du fond (Y=3). On vous demande d'écrire le bout de programme qui initialise tous les paramètres appropriés en utilisant seulement les fonctions énumérées plus haut. Assurez-vous d'inclure les appels aux fonctions qui choisissent la matrice à modifier. De plus, on considère que la clôture a déjà été établie et on ne tient pas compte des déformations dues à celle-ci. (Nous ne voulons que les appels de fonction avec la valeur de leurs paramètres. Il n'est pas nécessaire de placer les paramètres dans le bon ordre, mais vous devez donner la signification de chaque paramètre.)

f) Écrivez tous les appels aux fonctions pour obtenir, en projection orthogonale, une vue de plan de toute la pièce, mais seulement la pièce. [2 points]

g) Écrivez tous les appels aux fonctions pour obtenir, en projection perspective, une vue d'exactement tout le mur du fond d'une distance de 4m. On désire voir tout ce qu'il y a entre le centre de la pièce et le mur du fond. [2 points]

h) Écrivez seulement la ou les modification(s) à apporter à (g) pour voir seulement la fenêtre, sans déplacer la caméra. [1 point]

Question 77 Visualisation 3D et caméra synthétique [7 points]



On désire transformer le demi-cylindre de la figure de gauche pour qu'il corresponde à celui de la figure de droite.

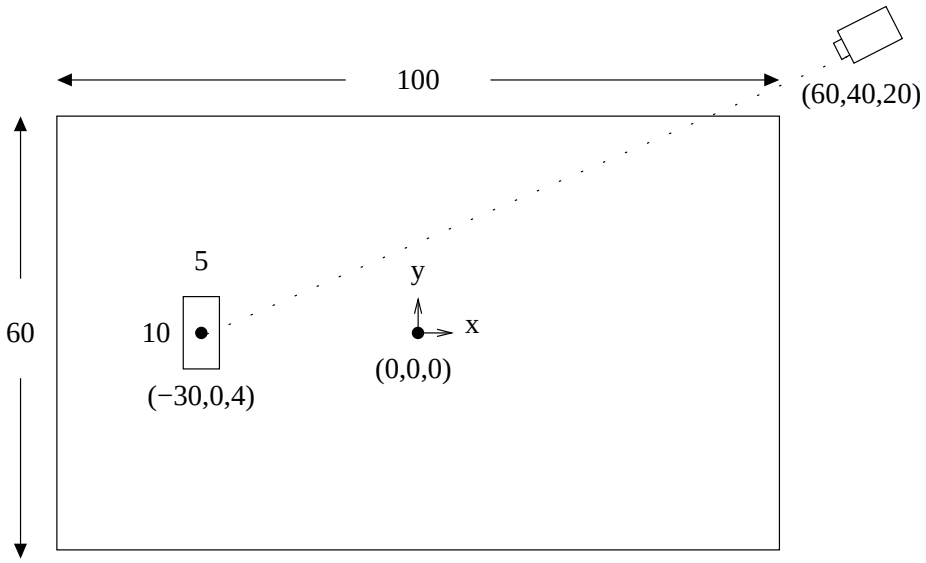
a) Donnez trois façons de parvenir à ce résultat en multipliant *deux* matrices de rotation autour des axes cartésiens en utilisant seulement des angles entre 0 et 360 degrés. Vous pouvez utiliser la notation Rx(theta), Ry(theta) ou Rz(theta) pour indiquer une matrice de rotation autour de x, y ou z. [3 points]

b) Sélectionnez une réponse précédente, donnez votre choix et donnez les énoncés OpenGL permettant de réaliser ces transformations. [2 points]
Rappel : glRotate(angle, x, y, z) effectue une rotation de "angle" degrés autour de l'axe (x,y,z).

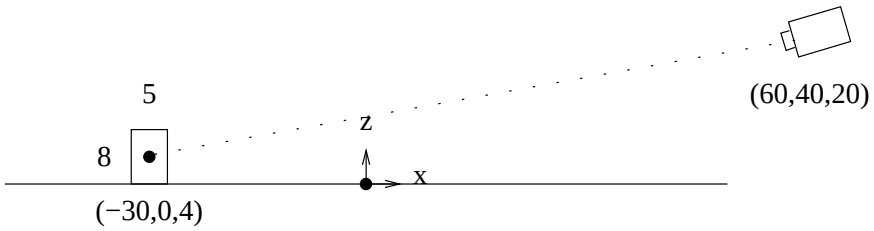
c) Donnez le diagramme du pipeline de transformations OpenGL et indiquez quelle partie du pipeline est modifiée par la matrice de visualisation généralement obtenue grâce à gluLookAt ? [2 points]

Question 78 Visualisation 3D, transformations et projections [13 points]

Soit un monde virtuel simple qui se compose d'un aréna rectangulaire, d'un but fixe et d'une très grande quantité de rondelles en mouvement. Les deux vues orthogonales de la figure ci-dessous indiquent les positions et dimensions pertinentes : la surface de l'aréna est de 100 x 60 unités, le but mesure 5 x 10 x 8 et son centre est à $(-30, 0, 4)$, la caméra est positionnée à $(60, 40, 20)$ et est dirigée vers le centre du but. On considère que les rondelles occupent un faible volume de l'espace comparé à celui du but.



Vue de plan



Vue de côté

Projection perspective

Nous désirons définir une projection perspective représentant un champ de vision de 160 degrés et permettant de voir tout ce qui se trouve devant la caméra jusqu'à une distance maximale de 1000 unités.

a) Si la fenêtre graphique sur l'écran a une taille de 600 x 300 pixels (largeur x hauteur), donnez les valeurs des paramètres des fonctions gluPerspective(), gluLookAt() et glViewport(). Pour cette sous-question, nous ne voulons que les appels de fonction avec la valeur de leurs paramètres. Il n'est pas nécessaire de placer ces paramètres dans le bon ordre, mais vous devez donner la signification de chacun d'eux. [3 points]

b) Voici un extrait de programme devant permettre l'affichage de notre scène en animation. La fonction PaintGL() est appelée en continu par le programme mais, malheureusement, 6 erreurs dans l'utilisation des fonctions OpenGL s'y sont glissées et empêchent le programme de bien se comporter : nous n'obtenons pas le résultat voulu ! On vous demande donc de corriger ce programme. Indiquez **et** expliquez chaque erreur identifiée. (Il n'y a aucun autre énoncé OpenGL ailleurs qui affecte les matrices du pipeline graphique ; l'affichage des balles et les listes d'affichage ne contiennent aucune erreur ; la compilation ne cause pas de problème ; seul le résultat affiché est incorrect.) [6 points]

```
Graphique::PaintGL()
```

```
{
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glViewport(...);     // paramètres de la question a)
   glPushMatrix();
   gluPerspective(...); // paramètres de la question a)
   glPopMatrix();

   glMatrixMode(GL_MODELVIEW);
   gluLookAt(...);      // paramètres de la question a)
   glViewport(...);     // paramètres de la question a)
   but->afficher();
   arena->afficher();
   balle->afficher();   // ne contient pas d'erreur
   glXSwapBuffers();
}

But::afficher()
{
   glLoadIdentity();
   glTranslated(-30,0,0);
   glCallList(But)      // la liste But affiche le but centré
                        // à (0,0,0) selon les dimensions prévues
                        // et ne contient pas d'erreur
}

Arena::afficher()
{
   glBegin(GL_QUADS);
      glVertex3f(-50,-30,0);
      glVertex2f(50,-30);
      glVertex3f(50,30,0);
      glVertex3f(-50,0,30);
   glEnd();
}
```

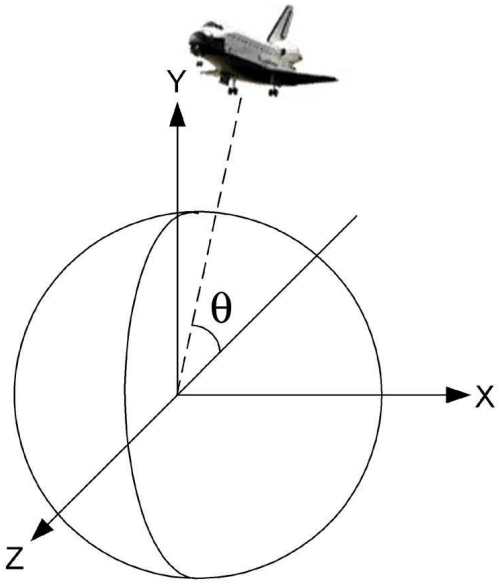
Projection orthographique

Le mode sélection d'OpenGL peut être utilisé pour obtenir des effets variés et dans un but autre que la simple sélection avec la souris. Supposons ainsi qu'en mode sélection, nous redéfinissons le volume de visualisation (i) en utilisant une projection orthographique dont le plan de projection est parallèle au devant du but ; (ii) en regardant dans la direction donnée par le vecteur $(1, 0, 0)$; et (iii) en s'assurant que le volume de visualisation corresponde exactement au volume occupé par le but dans l'espace virtuel.

c) Donnez les valeurs des paramètres à mettre dans les fonctions gluLookAt() et glOrtho() afin de satisfaire ces exigences. Pour cette sous-question, nous ne voulons que les appels de fonction avec la valeur de leurs paramètres. Il n'est pas nécessaire de placer ces paramètres dans le bon ordre, mais vous devez donner la signification de chacun d'eux. [2 points]

d) Supposons qu'à chaque étape d'affichage nous faisons une sélection OpenGL en utilisant le volume de visualisation décrit ci-dessus. Dites à quoi correspond le nombre de "hits" (la valeur retournée par glRenderMode()) et à quoi cela aurait pu servir dans votre projet. [2 points]

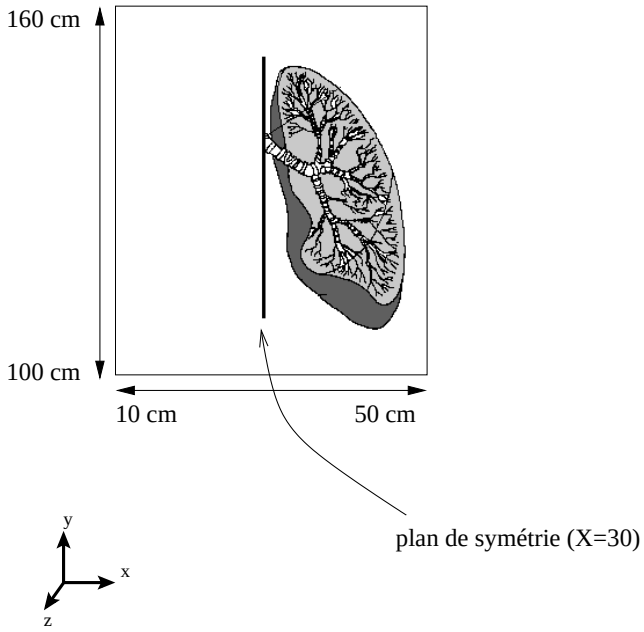
Question 79 Visualisation 3D et caméra synthétique [8 points]



a) On vous demande de réaliser un simulateur pour une navette spatiale en orbite autour de la terre. L'orbite passe par les deux pôles. Dans votre modèle, le centre de la terre est placé à l'origine et le plan (y,z) traverse les deux pôles (le plan (x,z) est l'équateur). La navette se trouve à une distance de 100 unités du centre de la terre et, afin de pouvoir atterrir facilement, son plancher est toujours orienté vers le sol comme illustré sur la figure ci-dessus. Écrivez les énoncés OpenGL et la valeur de leurs paramètres qu'il faut mettre dans la fonction "PlaceCamera" afin d'initialiser la matrice de visualisation qui donne le point de vue d'un astronaute assis aux commandes de la navette. Cette fonction recevra en paramètre l'angle θ de l'orbite. [8 points]

```
void PlaceCamera( const GLdouble theta )
{
   ... (ajoutez ici les énoncés appropriés)...
   glMatrixMode( GL_MODELVIEW );
   ... (ajoutez ici les énoncés appropriés)...
}
```

Question 80 Listes d'affichage et visualisation 3D



a) Le système de reconstruction radiographique 3D produit des modèles de poumons dont les coordonnées sont exprimées dans un repère global. Le repère gobal est (X, Y, Z) où X est l'axe gauche-droite du patient, Y l'axe caudal-cranial (des pieds vers la tête), Z l'axe postéro-antérieur (du dos vers l'avant) et l'origine se trouve au sol. Ainsi, pour un patient donné, en position debout, nous avons obtenu un modèle du poumon droit et nous avons déjà construit une liste d'affichage `PoumonDroit` qui dessine le modèle filaire de ce poumon droit. (Les coordonnées de la liste d'affichage sont exprimées dans le repère global.) On souhaite maintenant dessiner un poumon gauche symétrique au poumon droit par rapport au plan X=30. Donnez les énoncés OpenGL qui permettent d'afficher le poumon gauche après avoir affiché le poumon droit, en faisant appel à la liste d'affichage déjà définie. (Quels énoncés doivent être mis entre les glPushMatrix() et glPopMatrix() ?) **[6 points]**

```
glPushMatrix( );
glCallList( PoumonDroit );
.
.
.
glPopMatrix( );
```

b) L'ensemble des deux poumons est centré dans une boîte englobante dont le plan de symétrie est à X = 30 et dont les dimensions sont fixées comme suit (voir figure ci-dessus) :

$$10 \text{ cm} \leq X \leq 50 \text{ cm}, \quad 100 \text{ cm} \leq Y \leq 160 \text{ cm}, \quad -10 \text{ cm} \leq Z \leq 10 \text{ cm}$$

On désire effectuer une projection orthogonale selon l'axe Z afin de visualiser sur un moniteur l'ensemble des deux poumons du patient en position debout (comme en (a)), centré dans une fenêtre rectangulaire. Le rapport d'aspect sera, bien sûr, respecté. Donnez les énoncés OpenGL définissant les transformations du pipeline graphique afin d'afficher les deux poumons. Précisez la valeur des paramètres de chacune des fonctions utilisées, donnez la signification de chaque paramètre et dites quelle partie du pipeline graphique

chaque fonction affecte. (Il n'est pas absolument nécessaire de placer les paramètres dans le bon ordre.) **[7 points]**

c) Dans une seconde phase de l'examen radiographique, le patient est maintenant en position inclinée. On obtient alors de nouvelles coordonnées pour le poumon droit de ce patient ; le plan de symétrie est maintenant décrit par l'équation Y = X + 100. Quelles sont les transformations élémentaires (rotation, translation, mise à l'échelle) et les transformations inverses qui doivent être appliquées aux coordonnées du poumon droit afin de calculer les coordonnées du poumon gauche ? Donnez les matrices 4x4 associées à chaque transformation en spécifiant les valeurs numériques des éléments de la matrice. On ne veut pas les énoncés OpenGL, mais on demande plutôt les transformations mathématiques ainsi que l'ordre dans lequel elles doivent être appliquées (leur composition). **[5 points]**

Question 81 Visualisation 3D

Une application est affichée dans une fenêtre OpenGL de dimensions 300x200 pixels. Cette application utilise glFrustum pour générer une matrice de projection (qui ne respecte pas le rapport d'aspect) en utilisant les paramètres suivants :

Côté gauche :	-15	Côté droit :	15
Bas :	-10	Haut :	10
Plan avant :	5	Plan arrière :	100

a) Quelles modifications doit-on apporter à Côté gauche, Côté droit, Bas et Haut de manière à cadrer le plus précisément possible un disque de rayon 20 situé au centre du volume de visualisation, à 50 unités de l'observateur et lui faisant face. Vous devez respecter le rapport d'aspect. **[2 points]**

b) Aurait-il été possible de cadrer précisément le disque : **[2 points]**

i) en ne modifiant que Plan avant ? (Si oui, donnez la valeur à utiliser.)

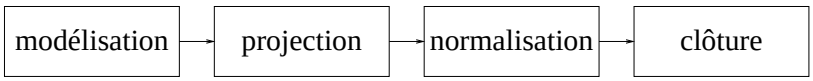
ii) en ne modifiant que Plan arrière ? (Si oui, donnez la valeur à utiliser.)

c) Aurait-on obtenu les mêmes résultats en (a) si on avait remplacé le disque par une sphère de rayon 20 ? Expliquez. **[1 point]**

d) Supposez que le centre du disque soit situé au point (5, 10, -5) en coordonnées universelles et qu'un vecteur normal à ce disque soit (3, 4, 0). Le point (-11, 22, -5) est situé sur la circonférence du disque et doit se retrouver dans le haut de l'image projetée. Écrivez l'appel à gluLookAt requis pour obtenir la vue décrite à la sous-question (a). **[3 points]**

Question 82 Visualisation 3D

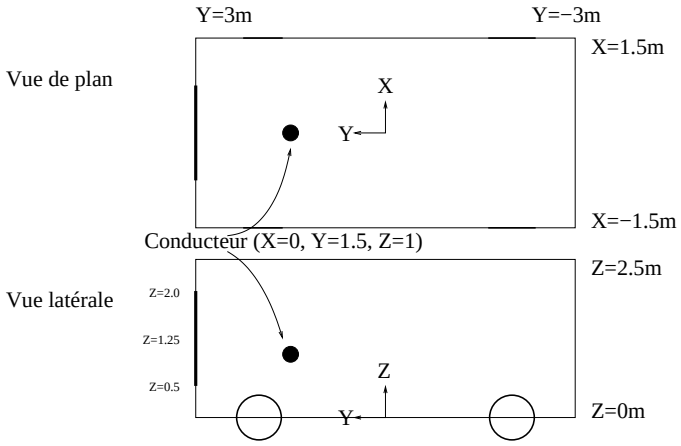
a) Comme nous l'avons vu dans le cours, le pipeline graphique d'OpenGL est illustré ainsi :



La librairie OpenGL offre certaines fonctions pour modifier le comportement de ce pipeline graphique : `glFrustum()`, `glLoadIdentity()`, `glOrtho()`, `glViewport()`, `gluLookAt()`, `gluOrtho2D()`, `gluPerspective()` et `gluPickMatrix()`. Indiquez la ou les parties du pipeline graphique à laquelle s'adresse normalement chaque fonction. **[3 points]**

b) Avant d'utiliser certaines des fonctions précédentes, il faut faire un appel à `glMatrixMode()`. Expliquez l'utilité de cette fonction. **[1 point]**

Pour les sous-questions suivantes, supposons qu'une première fonction `Dessine`() permet d'afficher l'intérieur d'un autobus similaire à une des boîtes à savon du projet. La fonction trace tout ce qui est à l'intérieur en supposant que l'origine est au plancher au centre du véhicule et que les largeur, longueur et hauteur correspondent respectivement à des variations en X de -1.5m à 1.5m (de gauche à droite), en Y de -3m à 3m (d'arrière à avant) et en Z de 0m à 2.5m (du plancher au plafond). Afin de voir où on s'en va, une fenêtre de 1.5m x 1.5m est située au centre à l'avant du véhicule (Y=3m). Pour les sous-questions suivantes, utilisez les énoncés OpenGL qui initialisent les paramètres appropriés en utilisant seulement les fonctions identifiées dans les questions a et b. (Nous ne voulons que les appels de fonction avec la valeur de leurs paramètres. Il n'est pas nécessaire de placer les paramètres dans le bon ordre, mais vous devez donner la signification de chaque paramètre.)



c) Écrivez tous les appels aux fonctions pour obtenir, en projection orthogonale, la vue de plan illustrée de l'intérieur du véhicule et rien de plus, avec une clôture de 400 x 200 pixels. **[4 points]**

d) Certains diront peut-être que c'est un véhicule un peu bizarre, mais le conducteur est assis bien au centre en X=0, (ni trop à gauche, ni trop à droite), sa tête est à Z=1 mètre du sol et il est à Y=1.5 mètre de l'avant du véhicule. On désire afficher à l'écran ce qui est à l'extérieur de l'autobus seulement, jusqu'à une distance de 50m de l'avant du l'autobus. Écrivez tous les appels aux fonctions pour obtenir, en projection perspective, la vue d'exactement ce que voit le conducteur au travers de la fenêtre dans une clôture de 200 x 200 pixels. **[4 points]**

Question 83 Visualisation 3D

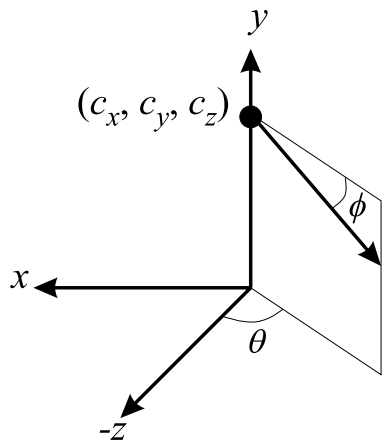
a) Soit l'appel OpenGL suivant :

```
gluPerspective( 90.0, 2.0, 1.0, 100.0 );
```

Écrivez un appel à `glFrustum()` réalisant exactement la même projection perspective. **[2 points]**

b) Bien qu'on puisse convertir n'importe quel appel à `gluPerspective()` en un appel à `glFrustum()`, le contraire n'est pas possible. Expliquez pourquoi et donnez un exemple d'appel à `glFrustum()` qui ne peut être converti en `gluPerspective()`. **[1 point]**

c) On souhaite utiliser `gluLookAt` pour modéliser une caméra de surveillance installée au plafond d'un magasin. La caméra se situe à la position (c_x, c_y, c_z) et le plafond correspond au plan $y = c_y$. L'angle θ indique la rotation de la caméra par rapport à l'axe y, un angle $\theta = 0$ correspond à une caméra pointant vers les z négatifs. L'angle ϕ indique l'angle entre la caméra et le plafond. On sait que $0 \leq \theta \leq 2\pi$ et $0 \leq \phi < \pi/2$ (notez bien l'inégalité stricte). La figure suivante illustre cette caméra de surveillance :



Écrivez la fonction `camera(`c_x`,`c_y`,`c_z`,`θ`,`ϕ`)` qui utilise `gluLookAt` pour configurer la matrice de visualisation modélisant la caméra de surveillance. Vous n'avez pas à configurer la projection. Assurez-vous que vous modifiez la bonne matrice ! **[4 points]**

d) Réécrivez la fonction précédente en remplaçant `gluLookAt` par des appels à `glRotatef` et `glTranslatef`. (Question difficile !) **[3 points]**

Question 84 Visualisation 3D et caméra synthétique

Pour mieux visiter la planète Mars, on pourrait vouloir mettre en orbite une navette autour de la planète. Pour simuler cette possibilité, on souhaite écrire un programme en OpenGL qui montrera la vue que les astronautes auront à partir de la navette.

a) L'orbite de la navette passe d'un pôle à l'autre à une hauteur de $2*10^6m$ au-dessus du sol. On convient que l'origine de notre repère orthogonal est placée au centre de Mars, que l'axe des Z passe par les pôles et que l'axe des X passe par un point bien connu des astronautes (le cratère Airy-0 qui définit le méridien 0). Considérant que le rayon moyen de Mars est d'environ $3.4 * 10^6m$, le rayon total de l'orbite est donc de $5.4 * 10^6m$ ($5\ 400\ 000\ m$). L'angle θ (en degrés) donne la position courante de la navette sur son orbite (dans le plan YZ). Notez que le plancher de la navette demeure toujours parallèle à la surface du sol (figure 1). Écrivez la fonction `DefinirCamera` avec les énoncés OpenGL nécessaires en indiquant bien la valeur des paramètres qu'il faut mettre afin d'initialiser la matrice de visualisation qui donne le point de vue d'un astronaute assis aux commandes de la navette et qui regarde droit devant lui (sans voir la planète). (N'oubliez pas de faire appel à `glMatrixMode` !) **[5 points]**

```
void DefinirCamera( const GLdouble theta ) { ... }
```

b) Pour bien fonctionner, notre logiciel de visualisation doit aussi définir le contenu de la matrice de projection. L'astronaute est assis aux commandes de la navette et regarde droit devant lui, vers le centre d'une fenêtre rectangulaire qui est perpendiculaire à la direction de son regard (figure 2). Il est intéressé par ce qui est visible à l'extérieur de la navette, dans l'espace autour de la planète. Son siège est positionné à $3\ m$ du centre de la fenêtre et celle-ci mesure $4\ m$ en largeur et $1.6\ m$ en hauteur. Écrivez la fonction `DefinirProjection` avec les énoncés OpenGL nécessaires en indiquant bien la valeur des paramètres qu'il faut mettre. (N'oubliez pas `glMatrixMode` !) **[4 points]**

```
void DefinirProjection( void ) { ... }
```

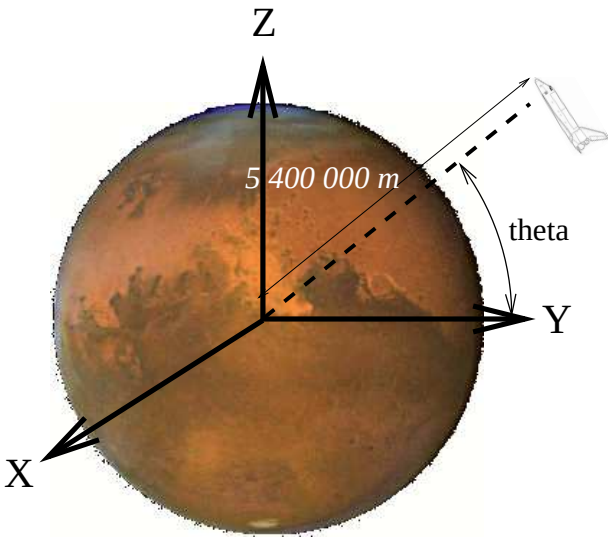


figure 1

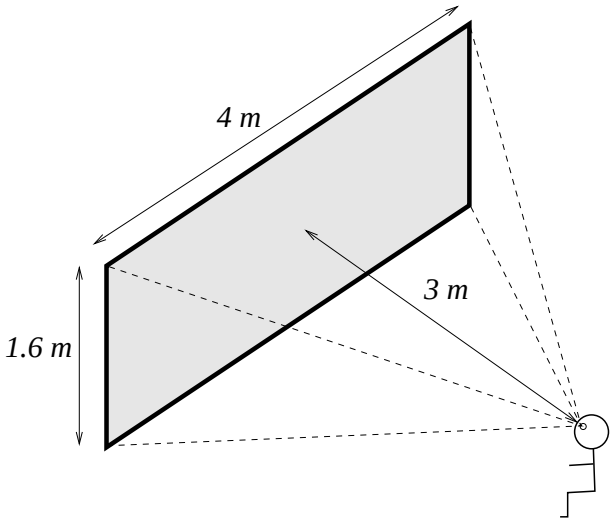


figure 2

Question 85 Pipeline graphique

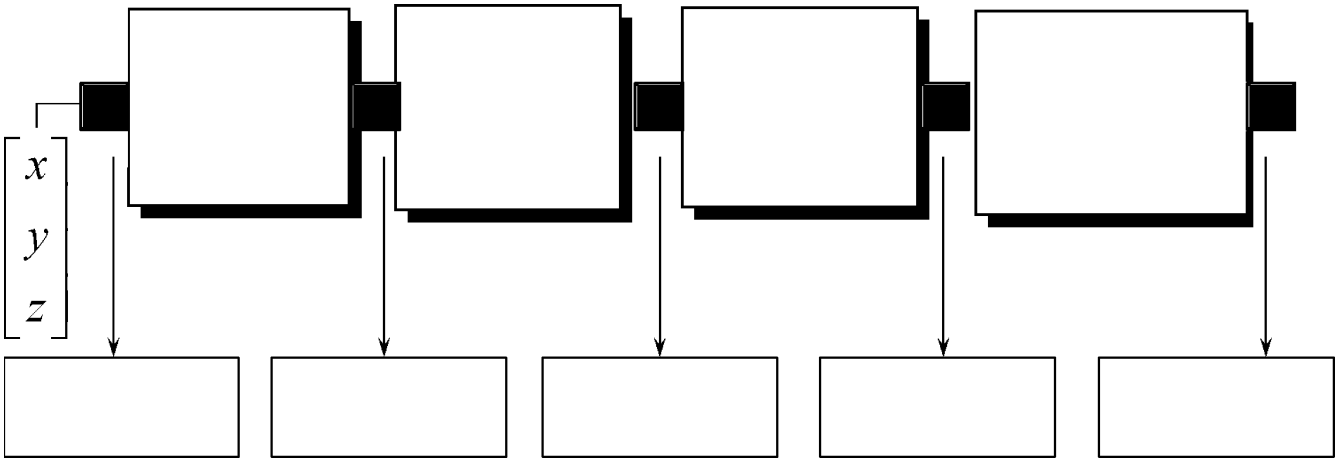
a) Trois sphères de rayon $r = 1$ sont positionnées à ces coordonnées 2D : $(3, 0)$, $(4, -5)$ et $(-2, 2)$. Donnez le coin inférieur gauche (xmin, ymin) et le coin supérieur droit (xmax, ymax) du plus petit rectangle qui englobe ces trois sphères. **[1 point]**

(xmin, ymin) = (_____ , _____) (xmax, ymax) = (_____ , _____)

b) Quel problème peut se produire si on utilise directement la boîte englobante trouvée ci-dessus comme fenêtre virtuelle sans se soucier des valeurs courantes de la clôture ? **[1 point]**

c) Si la clôture a les coordonnées (imin,jmin) = $(100, 0)$ et (imax,jmax) = $(400, 600)$, calculez alors les coordonnées (xmin, ymin) et (xmax, ymax) de la plus petite fenêtre virtuelle permettant d'englober les trois sphères sans produire le problème identifié ci-dessus. **[4 points]**

d) La figure suivante illustre le pipeline OpenGL de la transformation des sommets.



i) Pour chacune des neuf cases, choisissez l'expression la plus appropriée dans la liste suivante et inscrivez cette expression (ou la lettre correspondante) dans la case. **[4 points]**
(Il n'est évidemment pas nécessaire d'utiliser toutes les expressions !)

- `a` : « coordonnées d'affichage »
- `b` : « coordonnées d'appareil normalisées »
- `c` : « coordonnées d'appareil »
- `d` : « coordonnées d'objet »
- `e` : « coordonnées de couleur »
- `f` : « coordonnées de découpage »
- `g` : « coordonnées de stencil »
- `h` : « coordonnées de visualisation »
- `i` : « coordonnées normalisées »
- `j` : « coordonnées universelles »
- `k` : « matrice de modélisation »
- `l` : « matrice de profondeur »
- `m` : « matrice de projection »
- `n` : « matrice de stencil »
- `o` : « mémoire de trame »
- `p` : « tampon de profondeur »
- `q` : « tampon de stencil »

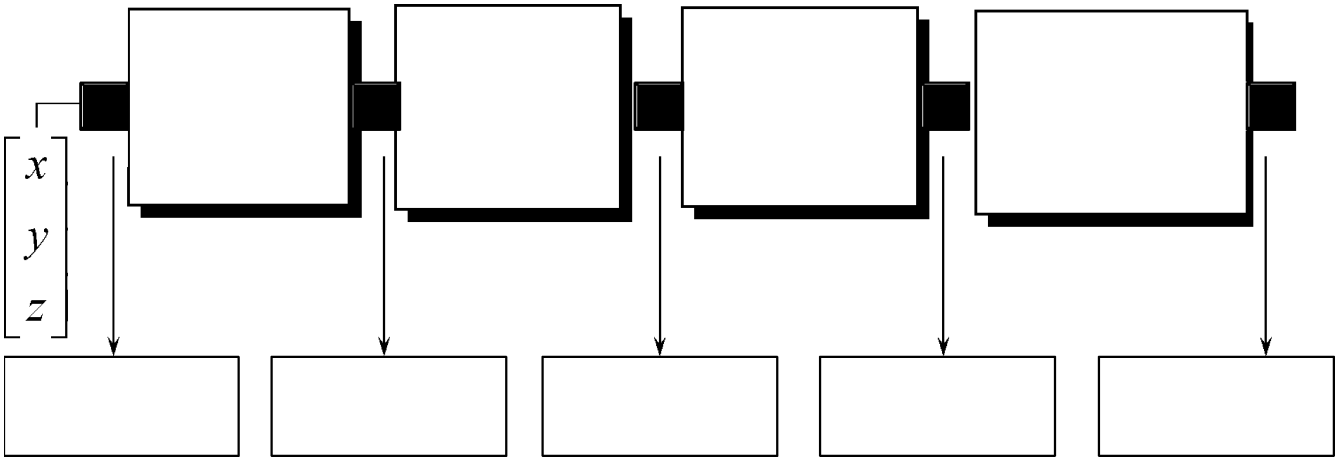
- `r` : « transformation de clôture »
- `s` : « transformation de conversion »
- `t` : « transformation de normalisation »

ii) Inscrivez aussi les quatre expressions suivantes (ou les lettres) dans les cases appropriées. **[4 points]**
(Ici, chaque expression doit trouver sa place dans une case !)

- `A` : « caméra »
- `B` : « glFrustum »
- `C` : « Cohen-Sutherland »
- `D` : « point de fuite »

Question 86 Pipeline graphique

a) La figure suivante illustre le pipeline OpenGL de la transformation des sommets.



i) Pour chacune des neuf cases ci-dessus, choisissez l'expression la plus appropriée dans la liste suivante et inscrivez la lettre correspondante dans la case. **[4 points]**
(Une seule expression par case. Il n'est évidemment pas nécessaire d'utiliser toutes les expressions !)

- **`a`** : système de coordonnées
- **`b`** : coordonnées d'affichage
- **`c`** : coordonnées d'appareil normalisées
- **`d`** : coordonnées d'appareil
- **`e`** : coordonnées d'objet
- **`f`** : coordonnées de primitive
- **`g`** : coordonnées de couleur
- **`h`** : coordonnées de découpage
- **`i`** : coordonnées polaires
- **`j`** : coordonnées de stencil
- **`k`** : coordonnées de visualisation
- **`l`** : coordonnées normalisées
- **`m`** : coordonnées universelles
- **`n`** : matrice de modélisation
- **`o`** : matrice de profondeur
- **`p`** : matrice de projection
- **`q`** : matrice de forme
- **`r`** : matrice de stencil
- **`s`** : mémoire de trame
- **`t`** : transformation de clôture
- **`u`** : transformation de conversion
- **`v`** : transformation de normalisation
- **`w`** : transformation de couleur
- **`x`** : tampon de couleur
- **`y`** : tampon de stencil
- **`z`** : tampon de profondeur

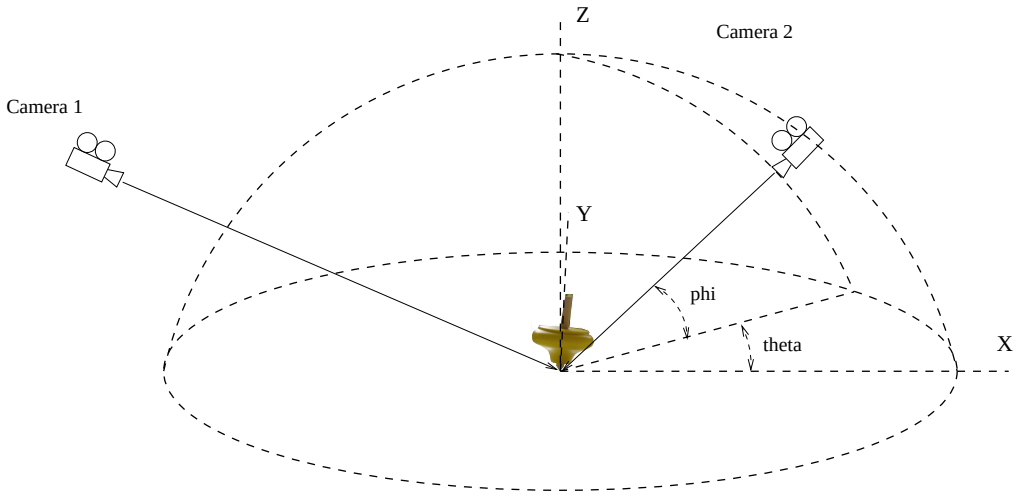
ii) Pour chacune des quatre expressions suivantes, choisissez la case la plus appropriée et inscrivez-y la lettre correspondante. **[4 points]**

- **A** : volume de visualisation
- **B** : caméra synthétique
- **C** : point de fuite
- **D** : Cohen-Sutherland

iii) Pour chacun des sept groupes de fonctions suivants, choisissez la ou les cases les plus appropriées (selon ce que chacun de ces groupes peut contrôler) et inscrivez-y le chiffre correspondant. **[4 points]**

- **1** : `glFrustum()`, `gluPerspective()`
- **2** : `glLoadIdentity()`
- **3** : `glLoadMatrix()`, `glMultMatrix()`
- **4** : `glOrtho()`, `gluOrtho2D()`
- **5** : `glTranslate()`, `glScale()`, `glRotate()`
- **6** : `glViewport()`
- **7** : `gluLookAt()`

Question 87 Visualisation 3D et caméra synthétique



Vous concevez un nouveau jeu de toupies et vous voulez suivre les mouvements d'une toupie avec des caméras en proposant deux vues possibles :

i) La vue « 3^e personne » où la caméra 1 suit la toupie et imite ses mouvements, tout en restant éloignée d'une certaine distance et toujours positionnée le long du vecteur (-1,-1,+1) par rapport à la toupie.

ii) La vue sphérique où la caméra 2 peut tourner librement autour de la position de la toupie (sur la surface marquée en pointillés dans le diagramme ci-dessus). Les mouvements de cette caméra sont toutefois limités : la caméra ne peut se déplacer que dans l'hémisphère situé au-dessus du plan X-Y et la caméra ne peut être directement au-dessus de la toupie.

La distance de la caméra par rapport à la toupie est contrôlable pour les deux points de vue.

La signature de la fonction `PositionnerCamera` est :

```
void PositionnerCamera( const int numero, const float distance,
                        const float theta, const float phi,
                        const float toupieX, const float toupieY )
{ ... }
```

où :

- `numero` : est le numéro de la caméra (1 ou 2) ;
- `distance` : est la distance de la caméra à la toupie, valide pour les deux caméras ;
- `theta` et `phi` : sont les angles (en degrés) pour positionner la caméra 2 (inutiles pour la caméra 1) ;
- `toupieX` et `toupieY` : sont la position de la toupie dans le plan $Z = 0$.

Implémentez la fonction `PositionnerCamera` qui permet de positionner la caméra par rapport à la toupie à chaque rendu, en tenant compte des contraintes indiquées. Votre fonction doit mettre à jour toutes les matrices nécessaires à l'affichage, incluant la matrice de projection, en appelant les fonctions OpenGL pertinentes. (Vous pouvez choisir la perspective qui vous semble la plus appropriée, mais une ouverture de 90 degrés serait correcte.) Suite à l'appel de la fonction `PositionnerCamera`, il ne reste plus qu'à effectuer l'affichage de la toupie (que vous n'avez pas à faire ici mais plutôt dans votre TP2 !). **[9 points]**

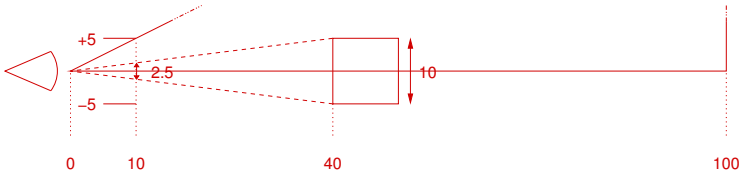
Question 88 Visualisation 3D

Une application ouvre une fenêtre OpenGL de 400 x 200 pixels. Cette application utilise `glFrustum` pour générer une matrice de projection en utilisant les paramètres suivants :

```
// glFrustum( GLdouble Gauche, GLdouble Droit,
//            GLdouble Bas, GLdouble Haut,
//            GLdouble PlanAvant, GLdouble PlanArrière );

   glMatrixMode( GL_PROJECTION );
   glLoadIdentity( );
   glFrustum(  -10.0, 10.0,  -5.0, 5.0,  10.0, 100.0  );
```

Un cube de 10 x 10 x 10 unités est positionné à 40 unités de l'observateur, directement dans l'axe de visée, au centre du volume de visualisation. La normale de la face avant du cube pointe directement vers l'observateur et celui-ci voit ainsi une forme bien carrée.

a) Quelles sont les dimensions de la face avant du cube une fois projetée sur le plan avant du volume de visualisation ? Illustrez votre raisonnement et vos calculs avec un diagramme. **[3 points]**



La face du cube projetée sur le plan avant est obtenue par une relation de proportion :
10 / 40 = Dim / 10 ⟹ Dim = 100 / 40 = ***2.5*** *[1pt] (ou* ***100 x 100*** *pixels).*

b) Quelles modifications doit-on apporter à Gauche, Droit, Bas et Haut (sans modifier PlanAvant ou PlanArrière) de manière à cadrer ce cube, c'est-à-dire de manière à ce que la face avant du cube occupe le plus d'espace possible dans la fenêtre ? Expliquez vos calculs. (N'oubliez pas de respecter le rapport d'aspect !) **[2 points]**

On pourrait avoir bêtement : Gauche = -1.25, Droite = 1.25, Bas = -1.25, Haut = 1.25 ;
pour respecter le rapport d'aspect, on prend ***Gauche = -2.5, Droit = 2.5, Bas = -1.25, Haut = 1.25***

c) Est-il possible de cadrer le cube en ne modifiant que PlanAvant ? Si oui, donnez la valeur à utiliser. Sinon, expliquez pourquoi. **[2 points]**

Oui. On doit amener le PlanAvant exactement à la position de la surface avant du cube : ***PlanAvant = 40****.*

d) Est-il possible de cadrer le cube en ne modifiant que PlanArrière ? Si oui, donnez la valeur à utiliser. Sinon, expliquez pourquoi. **[2 points]**

Non. Le plan arrière ne sert qu'au découpage du volume et n'affecte pas l'ouverture de la pyramide de visualisation.

Annexe A - pour la question 32 (page 25)

(Vous pouvez détacher cette annexe et ne pas la remettre.)

```
static void dessine( )
{
   // Initialisation du pipeline graphique
   glMatrixMode( GL_PROJECTION );
   glLoadIdentity( );
   glOrtho( 0, 10,  0, 10,  -10, 10 );

   glMatrixMode( GL_MODELVIEW );
   glLoadIdentity( );

   // fixer la couleur de l'arrière-plan à "blanc"
   glClearColor( 1.0, 1.0, 1.0, 1.0 );

   // initialiser les différents tampons
   glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT );

   glEnable( GL_DEPTH_TEST );

   glBegin( GL_QUADS ); {
      glColor4f( 0.5, 0.5, 0.5, 1.0 );
      glVertex3i( 1, 5, -5 );
      glVertex3i( 1, 3, -5 );
      glVertex3i( 9, 3, -5 );
      glVertex3i( 9, 5, -5 );
   } glEnd();

// EXEMPLE DE DESSIN DU CONTENU DE LA MÉMOIRE DE TRAME

   glBegin( GL_QUADS ); {
      glColor4f( 0.5, 0.5, 0.5, 0.5 );
      glVertex3i( 7, 6, -7 );
      glVertex3i( 7, 9, -7 );
      glVertex3i( 4, 9, -7 );
      glVertex3i( 4, 6, -7 );
   } glEnd();
```

```
// (a) DESSINEZ LE CONTENU DE LA MÉMOIRE DE TRAME

   glDisable( GL_DEPTH_TEST );

   glBegin( GL_QUADS ); {
      glColor3f( 0.0, 0.0, 0.0 );
      glVertex3i( 5, 0, -6 );
      glVertex3i( 6, 0, -6 );
      glVertex3i( 6, 9, -6 );
      glVertex3i( 5, 9, -6 );
   } glEnd();

// (b) DESSINEZ LE CONTENU DE LA MÉMOIRE DE TRAME
```

```
   // void glStencilOp( GLenum fail, GLenum zfail, GLenum zpass )
   // void glStencilFunc( GLenum func, GLint ref, GLuint mask )

   glEnable( GL_STENCIL_TEST );
   glStencilOp( GL_REPLACE, GL_REPLACE, GL_REPLACE );
   glStencilFunc( GL_NEVER, 1, 1 );

   glBegin( GL_QUADS ); {
      glColor3f( 0.5, 0.5, 0.5 );
      glVertex3i( 0, 0, -2 );
      glVertex3i( 3, 0, -2 );
      glVertex3i( 3, 4, -2 );
      glVertex3i( 0, 4, -2 );
   } glEnd();

// (c) DESSINEZ LE CONTENU DE LA MÉMOIRE DE TRAME ET DU STENCIL

   glStencilFunc( GL_NOTEQUAL, 1, 1 ) ;
   glStencilOp( GL_KEEP, GL_KEEP, GL_KEEP );

   glBegin( GL_QUADS ); {
      glColor3f( 0.0, 0.0, 0.0 );
      glVertex3i( 0, 0, -3 );
      glVertex3i( 4, 0, -3 );
      glVertex3i( 4, 5, -3 );
      glVertex3i( 0, 5, -3 );
   } glEnd();

   glDisable( GL_STENCIL_TEST );

// (d) DESSINEZ LE CONTENU DE LA MÉMOIRE DE TRAME ET DU STENCIL

   // void glBlendFunc( GLenum sourcefactor, GLenum destfactor )

   glEnable( GL_DEPTH_TEST );
   glEnable( GL_BLEND );
   glBlendFunc( GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA );

   glBegin( GL_QUADS ); {
      glColor4f( 0.0, 0.0, 0.0, 0.0 );
      glVertex3i( 7, 2, 0 );
      glVertex3i( 9, 2, 0 );
      glVertex3i( 9, 6, 0 );
      glVertex3i( 7, 6, 0 );
   } glEnd();

   glDisable( GL_BLEND );

// (e) DESSINEZ LE CONTENU DE LA MÉMOIRE DE TRAME

   glutSwapBuffers();
   return;
}
```

Annexe A - pour la question 33 (page 27)

(Vous pouvez détacher cette annexe et ne pas la remettre.)

```
void Scene::dessine( )
{
   // initialiser le pipeline graphique
   glMatrixMode( GL_PROJECTION );
   glLoadIdentity( );
   glOrtho( 0, 10,  0, 10,  0, 10 );
   glMatrixMode( GL_MODELVIEW );
   glLoadIdentity( );
   gluLookAt( 0, 0, 10,  0, 0, 0,  0, 1, 0 );

   // initialiser certaines variables d'état
   // void glBlendFunc( GLenum sourcefactor, GLenum destfactor )
   glBlendFunc( GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA );
   glDisable( GL_BLEND );
   glEnable( GL_DEPTH_TEST );
   glClearColor( 1.0, 1.0, 1.0, 1.0 );     // blanc

   // initialiser les différents tampons et tracer un premier rectangle
   glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT );
   glBegin( GL_QUADS ); {
      glColor4f( 0.5, 0.5, 0.5, 1.0 );
      glVertex3i( 3, 1, 6 );
      glVertex3i( 2, 1, 6 );
      glVertex3i( 2, 7, 6 );
      glVertex3i( 3, 7, 6 );
   } glEnd();

   // EXEMPLE DU DESSIN DU CONTENU LA MÉMOIRE DE TRAME

   glEnable( GL_BLEND );
   glBegin( GL_QUADS ); {
      glColor4f( 0.0, 0.0, 0.0, 0.5 );
      glVertex3i( 6, 1, 5 );
      glVertex3i( 6, 3, 5 );
      glVertex3i( 4, 3, 5 );
      glVertex3i( 4, 1, 5 );
   } glEnd();

   // (a) DESSINEZ LE CONTENU DE LA MÉMOIRE DE TRAME

   glDisable( GL_DEPTH_TEST );
   glBegin( GL_QUADS ); {
      glColor3f( 1.0, 1.0, 1.0 );
      glVertex3i( 0, 1, 4 );
      glVertex3i( 0, 2, 4 );
      glVertex3i( 9, 2, 4 );
      glVertex3i( 9, 1, 4 );
   } glEnd();
   glEnable( GL_DEPTH_TEST );

   // (b) DESSINEZ LE CONTENU DE LA MÉMOIRE DE TRAME
```

```
   // void glStencilFunc( GLenum func, GLint ref, GLuint mask )
   // void glStencilOp( GLenum fail, GLenum zfail, GLenum zpass )

   glEnable( GL_STENCIL_TEST );
   glStencilFunc( GL_ALWAYS, 2, 2 );
   glStencilOp( GL_REPLACE, GL_REPLACE, GL_REPLACE );

   glBegin( GL_QUADS ); {
      glColor3f( 1.0, 1.0, 1.0 );
      glVertex3i( 0, 0, 3 );
      glVertex3i( 4, 0, 3 );
      glVertex3i( 4, 4, 3 );
      glVertex3i( 0, 4, 3 );
   } glEnd();

   // (c) DESSINEZ LE CONTENU DE LA MÉMOIRE DE TRAME ET DU STENCIL

   glStencilFunc( GL_NOTEQUAL, 2, 2 ) ;
   glStencilOp( GL_KEEP, GL_KEEP, GL_KEEP );

   glBegin( GL_QUADS ); {
      glColor3f( 0.0, 0.0, 0.0 );
      glVertex3i( 0, 0, 7 );
      glVertex3i( 4, 0, 7 );
      glVertex3i( 4, 5, 7 );
      glVertex3i( 0, 5, 7 );
   } glEnd();

   glDisable( GL_STENCIL_TEST );

   // (d) DESSINEZ LE CONTENU DE LA MÉMOIRE DE TRAME ET DU STENCIL

   glDisable( GL_DEPTH_TEST );

   glBegin( GL_QUADS ); {
      glColor4f( 0.5, 0.5, 0.5, 0.0 );
      glVertex3i( 0, 3, 9 );
      glVertex3i( 3, 3, 9 );
      glVertex3i( 3, 5, 9 );
      glVertex3i( 0, 5, 9 );
   } glEnd();

   glDisable( GL_BLEND );

   // (e) DESSINEZ LE CONTENU DE LA MÉMOIRE DE TRAME

   glutSwapBuffers();
   return;
}
```