Question 1 Notions de base [15 points]

a) [6 points] Le modèle de réflexion de la lumière en un point proposé par Phong est largement utilisé en infographie. Ce modèle contient trois types distincts de réflexion : *ambiante*, *diffuse* et *spéculaire*. Dans ce modèle, identifiez les types de réflexion de la lumière sur un objet qui sont influencés par chaque élément ci-dessous. Cochez un seul choix par ligne. *(-1.0 point par erreur afin de pénaliser les choix aléatoires !)*

	aucune (-)	ambi. seulement a	diff. seulement d	spéc. seulement s	ambi. et diff. a+d	ambi. et spéc. a+s	diff. et spéc. d+s	ambi., diff., spéc. a+d+s)	
i) La position de la source lumineuse influence la ou les réflexions ...	☐	☐	☐	☐	☐	☐	☒	☐	+1
ii) L'intensité de la source lumineuse influence la ou les réflexions ...	☐	☐	☐	☐	☐	☐	☐	☒	+1
iii) La position de l'observateur influence la ou les réflexions ...	☐	☐	☐	☒	☐	☐	☐	☐	+1
iv) Les propriétés de matériau influencent la ou les réflexions ...	☐	☐	☐	☐	☐	☐	☐	☒	+1
v) Les positions des autres objets de la scène influencent la ou les réflexions ...	☒	☐	☐	☐	☐	☐	☐	☐	+1
vi) Le coefficient de brillance influence la ou les réflexions ...	☐	☐	☐	☒	☐	☐	☐	☐	+1
vii) Le vecteur normal à la surface influence la ou les réflexions ...	☐	☐	☐	☐	☐	☐	☒	☐	+1

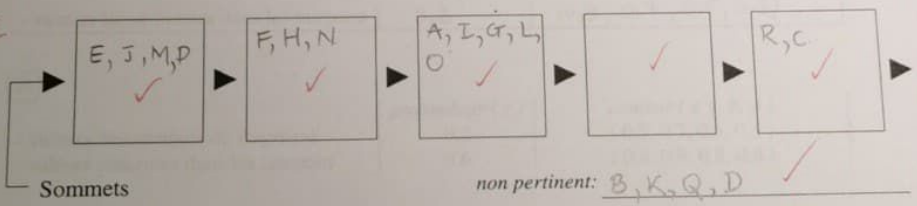
2

b) [2 points] Une mise à l'échelle est fondamentalement une opération de *multiplication*, tandis qu'une translation est, au contraire, une opération d'*addition*. Pourtant, les librairies graphiques (comme OpenGL) cumulent toutes ces opérations dans une seule matrice de transformation courante qui n'est utilisée, elle, que pour *multiplier* des coordonnées. Comment est-ce possible ? Nommez et expliquez succinctement l'artifice mathématique utilisé.

Cela est possible avec l'utilisation des coordonnées homogènes. Elles consistent à ajouter un vecteur (un axe) supplémentaire dans la matrice. Ainsi, d'une matrice 3x3 on passe à une matrice 4x4. Dans le cas des transformations multiplicatives le vecteur ajouté est (0,0,0,1) permettant de ne pas modifier le résultat obtenu (le vecteur a aussi une coordonnée supplémentaire = 1). Pour une translation $(\bar{T}x, \bar{T}y, \bar{T}z)$ on a la matrice $\begin{bmatrix} 1 & 0 & 0 & \bar{T}x \\ 0 & 1 & 0 & \bar{T}y \\ 0 & 0 & 1 & \bar{T}z \\ 0 & 0 & 0 & 1 \end{bmatrix}$ permettant l'addition lors de la multiplication de matrice.

c) [7 points] La figure suivante illustre le pipeline de transformations des sommets tel que vu au cours.

7



non pertinent: B, K, Q, D

Pour chacun des groupes de mots ou énoncés suivants, inscrivez la lettre correspondante ci-dessus, soit dans le carré approprié (selon ce à quoi il est associé ou ce qu'il peut contrôler), soit sur la ligne intitulée « *non pertinent* » (s'il n'est pas associé à ce pipeline). *(-0.5 point par erreur.)*

- A : *angle d'ouverture*
- B : *anticrénelage*
- C : *clôture*
- D : *fusion de couleurs*
- E : *matrice de modélisation*
- F : *caméra synthétique*
- G : *point de fuite*
- H : *positionner l'observateur*
- I : *projection orthographique*
- J : *réflexion par rapport à une droite*
- K : *test du stencil*
- L : *volume de visualisation*
- M : *transformations de modélisation*
- N : `MatricePipeline::LookAt()`
- O : `MatricePipeline::Perspective()`
- P : `MatricePipeline::Translate()`
- Q : `glColor()`
- R : `glViewport()`

8

Question 2 Fragments [8 points]

Toutes les sous-questions qui suivent présentent différentes situations pour lesquelles on souhaite déterminer le résultat d'un « test unitaire » pour tester l'effet de certains énoncés OpenGL. Chaque sous-question est indépendante des autres.

Pour chaque test, on vous donne les valeurs des attributs du fragment courant et les valeurs présentes dans les différents tampons (*profondeur, couleur, stencil*) et, selon les énoncés OpenGL, on vous demande d'écrire les valeurs subséquentes qui seront présentes dans les différents tampons.

1

a)

	profondeur (z)	*couleur (r, g, b, a)*
- valeurs des attributs du fragment	0.2	(0.9, 0.7, 0.5, 0.3)
- valeurs présentes dans les tampons	0.6	(0.8, 0.8, 0.8, 0.8)

```
glDisable( GL_STENCIL_TEST );
glEnable( GL_DEPTH_TEST );
glDepthFunc( GL_LEQUAL ); // " <= " ✓
glDisable( GL_BLEND );
```

- valeurs subséquentes dans les tampons	0,2	(0,9, 0,7, 0,5, 0,3)

1

b)

	profondeur (z)	*couleur (r, g, b, a)*
- valeurs des attributs du fragment	0.2	(0.9, 0.7, 0.5, 0.3)
- valeurs présentes dans les tampons	0.6	(0.8, 0.8, 0.8, 0.8)

```
glDisable( GL_STENCIL_TEST );
glDisable( GL_DEPTH_TEST ); // Désactiver le test fait aussi en sorte que le
                            // tampon de profondeur ne sera pas modifié
glDepthFunc( GL_GEQUAL ); // " >= "
glDisable( GL_BLEND );
```

- valeurs subséquentes dans les tampons	0,6	(0,9, 0,7, 0,5, 0,3)

Note :
```
void glStencilFunc( GLenum func, GLint ref, GLuint mask );
void glStencilOp( GLenum sfail, GLenum zfail, GLenum pass );
```

c)

	profondeur (z)	*couleur (r, g, b, a)*	*stencil*
- valeurs des attributs du fragment	0.2	(0.9, 0.7, 0.5, 0.3)	-
- valeurs présentes dans les tampons	0.6	(0.8, 0.8, 0.8, 0.8)	1

```
glEnable( GL_STENCIL_TEST );
glStencilFunc( GL_GEQUAL, 2, 3 ); // " >= " ✓
glStencilOp( GL_REPLACE, GL_DECR, GL_KEEP );
glEnable( GL_DEPTH_TEST );
glDepthFunc( GL_LESS ); // " < ", la valeur de défaut ✓
glDisable( GL_BLEND );
```

- valeurs subséquentes dans les tampons	0.2	(0.9, 0.7, 0.5, 0.3)	1

d)

	profondeur (z)	*couleur (r, g, b, a)*	*stencil*
- valeurs des attributs du fragment	0.2	(0.9, 0.7, 0.5, 0.3)	-
- valeurs présentes dans les tampons	0.6	(0.8, 0.8, 0.8, 0.8)	1

```
glEnable( GL_STENCIL_TEST );
glStencilFunc( GL_LESS, 2, 3 ); // " < " ✗
glStencilOp( GL_REPLACE, GL_DECR, GL_KEEP );
glEnable( GL_DEPTH_TEST );
glDepthFunc( GL_GEQUAL ); // " >= ", l'inverse de la valeur de défaut
glDisable( GL_BLEND );
```

- valeurs subséquentes dans les tampons	0.6	(0.8, 0.8, 0.8, 0.8)	2

e)

	profondeur (z)	*couleur (r, g, b, a)*	*stencil*
- valeurs des attributs du fragment	0.2	(0.9, 0.7, 0.5, 0.3)	-
- valeurs présentes dans les tampons	0.6	(0.8, 0.8, 0.8, 0.8)	1

```
glEnable( GL_STENCIL_TEST );
glStencilFunc( GL_GREATER, 2, 3 ); // " > " ✓
glStencilOp( GL_REPLACE, GL_DECR, GL_KEEP );
glEnable( GL_DEPTH_TEST );
glDepthFunc( GL_EQUAL ); // " == " ✗
glDisable( GL_BLEND );
```

- valeurs subséquentes dans les tampons	0.6	(0.8, 0.8, 0.8, 0.8)	0

Note : `void glBlendFunc( GLenum sfactor, GLenum dfactor );`

f)

	profondeur (z)	*couleur (r, g, b, a)*
- valeurs des attributs du fragment	0.2	(0.9, 0.7, 0.5, 0.3)
- valeurs présentes dans les tampons	0.6	(0.8, 0.8, 0.8, 0.8)

```
glDisable( GL_STENCIL_TEST );
glEnable( GL_DEPTH_TEST );
glDepthFunc( GL_LESS ); // " < "
glEnable( GL_BLEND );
glBlendFunc( GL_ONE, GL_ONE_ZERO ); // " 1, 0 "
```

- valeurs subséquentes dans les tampons	0.2	(0.9, 0.7, 0.5, 0.3)

g)

	profondeur (z)	*couleur (r, g, b, a)*
- valeurs des attributs du fragment	0.2	(0.9, 0.7, 0.5, 0.3)
- valeurs présentes dans les tampons	0.6	(0.8, 0.8, 0.8, 0.8)

```
glDisable( GL_STENCIL_TEST );
glEnable( GL_DEPTH_TEST );
glDepthFunc( GL_GREATER ); // " > "
glEnable( GL_BLEND );
glBlendFunc( GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA ); // " SrcA, 1-SrcA "
```

- valeurs subséquentes dans les tampons	0.6	(0.8, 0.8, 0.8, 0.8)

h)

	profondeur (z)	*couleur (r, g, b, a)*
- valeurs des attributs du fragment	0.2	(0.9, 0.7, 0.5, 0.3)
- valeurs présentes dans les tampons	0.6	(0.8, 0.8, 0.8, 0.8)

```
glDisable( GL_STENCIL_TEST );
glEnable( GL_DEPTH_TEST );
glDepthFunc( GL_ALWAYS ); // " toujours "
glEnable( GL_BLEND );
glBlendFunc( GL_SRC_ALPHA, GL_ONE ); // " SrcA, 1 "
```

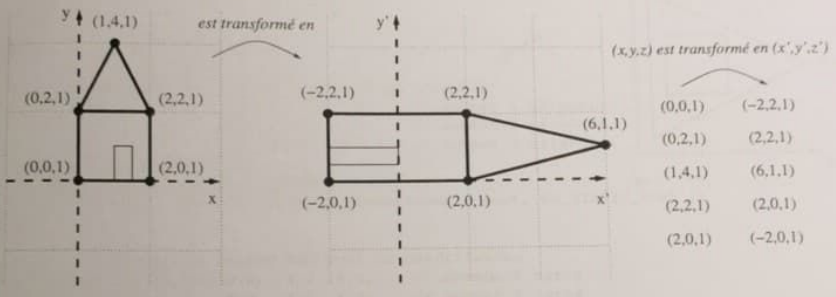
0.3 × 0.8 + 0.9

- valeurs subséquentes dans les tampons	0.2	(1, 1, 0.95, 0.89)

Question 3 Transformations dans le pipeline graphique [5 points]

5

a) [5 points] Tel que montré ci-dessous, une certaine transformation générale G permet de transformer la figure de gauche en celle de droite.



(x,y,z) est transformé en (x',y',z')

(0,0,1)	(−2,2,1)
(0,2,1)	(2,2,1)
(1,4,1)	(6,1,1)
(2,2,1)	(2,0,1)
(2,0,1)	(−2,0,1)

Sur cette figure, on voit que cette transformation G modifie les coordonnées x et y des sommets sans changer leur coordonnée z. En utilisant des coordonnées homogènes, la transformation générale G peut être exprimée comme le produit de trois transformations élémentaires, $G = T1 \cdot T2 \cdot T3$ et on peut alors appliquer l'opération $\vec{X'} = G \cdot \vec{X}$.

i) Si $T2$ est une transformation élémentaire de rotation, écrivez les noms des deux autres transformations élémentaires $T1$ et $T3$ (sans donner les paramètres) que vous utiliseriez pour construire G.

2

T1 : Translation T2 : *rotation* T3 : Mise à l'échelle

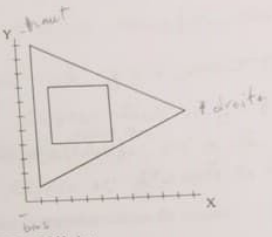
ii) Écrivez les deux matrices 4x4 qui correspondent aux transformations $T1$ et $T3$ identifiées ci-dessus.

3

$$G = \begin{pmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Question 4 Visualisation 3D [12 points]

Considérez les énoncés OpenGL ci-dessous utilisés pour afficher, tel qu'illustré dans le schéma ci-contre, un quadrilatère plein et un triangle plein.




```
void FenetreTP::initialiser()
{
   // initialiser le premier VAO pour le triangle
   GLfloat coot[] = ( 1.0,  1.0, 6.0,     // sommet 1 triangle
                     11.0,  6.0, 6.0,     // sommet 2 triangle
                      1.0, 11.0, 6.0 );  // sommet 3 triangle
   glBindVertexArray( vao[0] );
   glBindBuffer( GL_ARRAY_BUFFER, vbo[0] );
   glBufferData( GL_ARRAY_BUFFER, sizeof(coot), coot, GL_STATIC_DRAW );
   ...

   // initialiser le second VAO pour le quadrilatère
   GLfloat cooq[] = ( 2.0,  4.0, 4.0,     // sommet 1 carré
                      6.0,  4.0, 4.0,     // sommet 2 carré
                      6.0,  8.0, 4.0,     // sommet 3 carré
                      2.0,  8.0, 4.0 );  // sommet 4 carré
   glBindVertexArray( vao[1] );
   glBindBuffer( GL_ARRAY_BUFFER, vbo[1] );
   glBufferData( GL_ARRAY_BUFFER, sizeof(cooq), cooq, GL_STATIC_DRAW );
   ...
}


void FenetreTP::afficherScene()
{
   glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
   glEnable( GL_DEPTH_TEST );
   glDepthFunc( GL_LESS ); // la valeur de défaut

   matrModel.LoadIdentity();

   // void LookAt( obsX,obsY,obsZ, versX,versY,versZ, upX,upY,upZ );
   matrVisu.LookAt( 0.0, 0.0, 8.0,  0.0, 0.0, 0.0,  0.0, 1.0, 0.0 );

   // void Ortho( gauche,droite, bas,haut, planAvant,planArriere );
   matrProj.Ortho( -6.0, 6.0, -6.0, 6.0, 0.0, 8.0 );

   // tracer la scène
   glBindVertexArray( vao[0] );
   glDrawArrays( ... );  // tracer le triangle
   glBindVertexArray( vao[1] );
   glDrawArrays( ... );  // tracer le quadrilatère
}
```

(suite page suivante)

a) [2 points] Quelle est la valeur de Z des fragments du triangle après la transformation de *visualisation* ? Expliquez succinctement vos calculs.

La valeur de z des fragments du triangle est après transformation:
8 - 6 = 2

Explication : • Tous les sommets du triangle sont à z = 6 avant transf
• la transformation de visualisation met l'observateur à z = 8.
La direction positive de l'axe z étant dans la page (valeur vers 0,0,0), on a le triangle qui une position positive dans ce repère et à 2 unité de l'origi

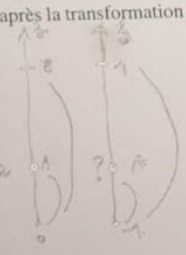
b) [2 points] Quelle est la valeur de Z des fragments du quadrilatère après la transformation de *visualisation* ? Expliquez succinctement vos calculs.

On a ici z = 8 - 4 = 4

Explication : Similaire à la question précédente.

c) [2 points] Quelle est la valeur de Z (entre -1.0 et 1.0) des fragments du triangle après la transformation de *projection* ? Expliquez succinctement vos calculs.

$$\frac{2-0}{8-0} = \frac{z+1}{1+1} \Rightarrow \frac{1}{4} = \frac{z+1}{2} \Rightarrow z+1 = \frac{1}{2} \Rightarrow \boxed{z = -0,5}$$



Explication : On aura un ratio conservé en passant d'un repère à l'autre on l'exploite donc pour trouver ce résultat.

2

d) [2 points] Quelle est la valeur de Z (entre -1.0 et 1.0) des fragments du quadrilatère après la transformation de *projection* ? Expliquez succinctement vos calculs.

$$\frac{4-0}{8-0} = \frac{z+1}{1+1} \Rightarrow \frac{1}{2} = \frac{z+1}{2} \Rightarrow \boxed{z = 0}$$

2 Explication : Similaire à c.

e) [2 points] i) Écrivez la comparaison faite par le test de profondeur qui est exécuté lors de l'affichage du quadrilatère. (Écrivez la valeur de profondeur, la comparaison et l'autre valeur de profondeur.) ii) Est-ce que le quadrilatère sera visible dans le rendu final ?

i) Comparaison : 0.0 < 0.5

ii) Non, le quadrilatère ne sera pas visible car tous ses fragments échouent le test de profondeur.

2

f) [2 points] Dans les énoncés de la page précédente, on affiche le triangle suivi ensuite du quadrilatère. Si on traçait d'abord le quadrilatère et ensuite le triangle, est-ce que la réponse à la sous-question ci-dessus serait différente ? Pourquoi ?

Non, la réponse ne serait pas différente car peu importe l'ordre dans lequel on les trace, la profondeur des fragments n'est pas modifié. Ainsi on ferait le test (pour le triangle) $-0.5 < 0.0$ Oui donc les pixels du quadrilatère seront remplacés pour ceux du triangle.

2