Nom: [redacted] Prénom: [redacted]

École polytechnique de Montréal

Département de génie informatique et génie logiciel

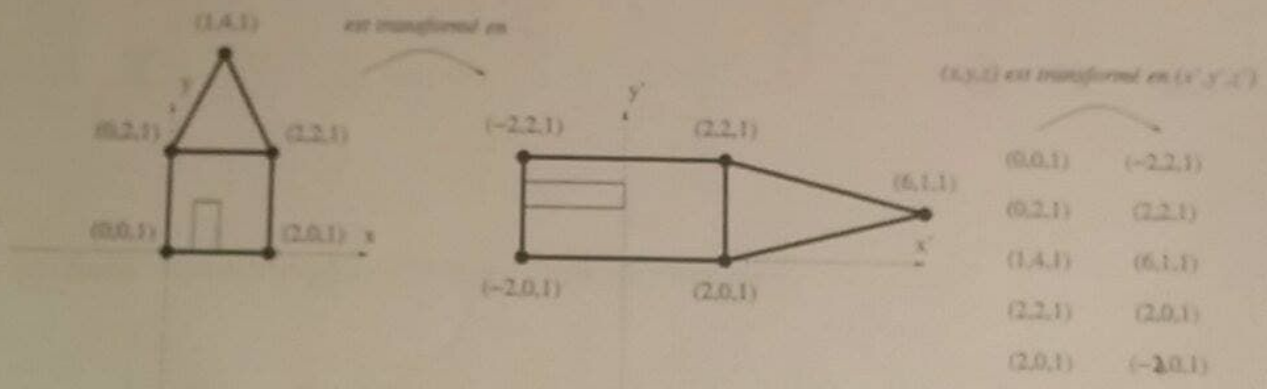
INF2705: Infographie (Hiver 2016)
Contrôle périodique

Notes:
- Toute documentation interdite, calculatrice programmable et ordinateur portatif interdits. Calculatrice non programmable permise.
- Cet examen comprend 4 questions sur 9 pages pour un total de 40 points.
- ⇒ *Répondez aux questions directement sur le questionnaire.*

3
3
9
9
24

Question 1 Transformations affines [6 points]

Tel que montré ci-dessous, une certaine transformation générale G permet de transformer la maison de gauche en celle de droite.



Sur cette figure, on voit que cette transformation G modifie les coordonnées x et y des sommets, mais ne change pas leur coordonnée z.

Si on utilise des coordonnées homogènes, on peut exprimer la transformation générale G comme le produit de trois transformations élémentaires, $G = T1 \cdot T2 \cdot T3$, et on peut alors appliquer l'opération $\vec{X'} = G \cdot \vec{X}$.

(suite à la page suivante)

a) 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.

T1 : translation (translate) T2 : *rotation* T3 : Scale (Mise à echelle)

b) Écrivez les deux matrices 4x4 qui correspondent à ces transformations $T1$ et $T3$.

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

c) Écrivez la matrice 4x4 qui correspond à la transformation G.

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

≠ le produit ci-dessus !

Question 2 Notions de base [5 points]

a) [2 points] Nous avons vu que les sommets des primitives sont transformés par le pipeline graphique avant d'être affichés à l'écran. Supposons que les seuls énoncés suivants sont utilisés pour définir le pipeline graphique :

```
MatricePipeline matrModel, matrVisu, matrProj;
matrModel.LoadIdentity();
matrVisu.LoadIdentity();
//       Ortho( gauche, droite,  bas, haut,  planAvant, planArrière );
matrProj.Ortho( x1, x2, y1, y2, -1, 1 );
// Les matrices sont utilisées de façon standard par les nuanceurs
progBase.assignerUniformMatrix4fv( "matrModel", matrModel );
progBase.assignerUniformMatrix4fv( "matrVisu", matrVisu );
progBase.assignerUniformMatrix4fv( "matrProj", matrProj );
glViewport( i0, j0, w, h );
```

Dans ce contexte, écrivez les formules qui permettent de convertir un point `(x,y)` en une position `(i,j)` à l'écran. Ces formules dépendront de `x`, `y`, `x1`, `x2`, `y1`, `y2`, `i0`, `j0`, `w` et `h`. (Vous pouvez supposer que `x1 <= x <= x2` et `y1 <= y <= y2`.)

2

$i = \left(\left(\frac{x - x_1}{x_2 - x_1}\right) \cdot w\right) + i_0$

$j = \left(\left(\frac{y - y_1}{y_2 - y_1}\right) \cdot h\right) + j_0$

b) [3 points] Trois types distincts de réflexion sont définis dans le modèle de réflexion de la lumière vu en classe : *ambiante*, *diffuse* et *spéculaire*. Dans ce modèle, quel(s) type(s) de réflexion dépend(ent) ...

1

i) ... du vecteur normal à la surface ? diffuse et spéc ✗

ii) ... de la position de l'observateur ? Spéculaire

iii) ... de la position de la source lumineuse ? ~~ambiante~~ d.ff et spéc ✗ ✗

Question 3 Pipeline graphique [19 points]

Un petit drone se promène au-dessus d'un terrain plat, sans jamais dépasser l'altitude réglementaire. L'opérateur est situé au centre du terrain et les positions successives du drone (par rapport à l'opérateur) sont enregistrées dans un tableau contenant $(x, y, hauteur)$ où x et y varient entre -50 m et 50 m, tandis que $hauteur$ varie entre 0 m et 100 m.

On souhaite afficher le vol du drone dans un logiciel graphique, similaire aux travaux pratiques du cours. On définit d'abord l'origine à la position de l'opérateur au centre du terrain. On pose la caméra synthétique fixe au niveau du sol à l'extérieur du terrain, à $(0, -60, 0)$, avec son objectif bien orienté vers l'opérateur et le haut de la caméra pointant vers le ciel. On utilise une projection orthogonale qui définit un volume de visualisation correspondant exactement à l'espace de vol du drone.

Le programme principal de l'application utilise ces énoncés :

```
MatricePipeline matrModel, matrVisu, matrProj;
matrModel.LoadIdentity();
matrVisu.LookAt( ... );
matrProj.Ortho( ... );
// Les matrices sont utilisées de façon standard par les nuanceurs
progBase.assignerUniformMatrix4fv( "matrModel", matrModel );
progBase.assignerUniformMatrix4fv( "matrVisu", matrVisu );
progBase.assignerUniformMatrix4fv( "matrProj", matrProj );
glViewport( 0, 0, 500, 500 ); // le rapport d'aspect est fixe et respecté
```

Par ailleurs, dans le nuanceur de sommets, on trouve ces lignes (parmi quelques autres) :

```
vec4 pos = matrVisu * matrModel * Vertex;
float zPos = abs( pos.z );
gl_Position = matrProj * pos;
```

et celles-ci dans le nuanceur de fragments :

```
vec2 leFrag = gl_FragCoord.xy;
float zFrag = gl_FragCoord.z / gl_FragCoord.w;
```

Considérant ces informations, remplissez les espaces soulignés ci-dessous et sur la page suivante.

a) [6 points] Dans ces deux énoncés du programme principal, les paramètres appropriés sont :

```
matrVisu.LookAt( 0 , -60, 0 , 0 , 0 , 0 , 0.0, 1.0, 0.0 );
//       LookAt( obsx , obsy , obsz , pViséx,pViséy,pViséz,  upx , upy , upz );

matrProj.Ortho( -2 , 2 , 0 , 100, -10 , -110 );
//       Ortho( gauche,droite,  bas, haut,  avant, arrière);
```

b) [10 points] Dans le nuanceur de sommets :

- Les unités de la variable `Vertex` sont des mètres gl float / gl double
- La valeur de `Vertex.x` varie entre -50 et 50.
- La valeur de `Vertex.y` varie entre -50 et 50.
- La valeur de `Vertex.z` varie entre 0 et 100
- Les unités de la variable `pos` sont des coordonné (Pixel) le vec n'est pas défini quel type il prend.
- La valeur de `pos.x` varie entre -50 et 50.
- La valeur de `pos.y` varie entre -50 (0) et 50 (100).
- La variable `zPos` mesure la distance entre la caméra et le drone.
- Puisque le drone est toujours visible, on peut affirmer que « `pos.z` >> 0 » est toujours vrai.
(Choisir parmi $<, \leq, =, \neq, \geq, >$ ou écrire ? si on ne peut rien affirmer.)
- La valeur de `pos.z` varie entre 0 (-10) et 100m (-110).

c) [3 points] Dans le nuanceur de fragments :

- Les unités de la variable `leFrag` sont des Pixel.
- La valeur de `leFrag.y` varie entre ? et ?.
- Puisque le drone est toujours visible, on peut affirmer alors que « `zFrag` = 0 » est toujours vrai.
(Choisir parmi $<, \leq, =, \neq, \geq, >$ ou écrire ? si on ne peut rien affirmer.)

?0

Question 4 Fragments [10 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 on vous demande de donner, selon les énoncés OpenGL, les valeurs subséquentes qui seront présentes dans les différents tampons.

a)

	profondeur (z)	*couleur (r, g, b, a)*
- valeurs des attributs du fragment	0.3	(1.0, 0.9, 0.8, 0.7)
- valeurs présentes dans les tampons	0.7	(0.4, 0.4, 0.4, 0.4)

```
glDepthFunc( GL_GEQUAL ); // " >= ", l'inverse de la valeur par défaut
glEnable( GL_DEPTH_TEST );
```

- valeurs subséquentes dans les tampons	0,7	0,4 0,4 0,4 0,4

b)

	profondeur (z)	*couleur (r, g, b, a)*
- valeurs des attributs du fragment	0.3	(1.0, 0.9, 0.8, 0.7)
- valeurs présentes dans les tampons	0.7	(0.4, 0.4, 0.4, 0.4)

```
glDepthFunc( GL_GREATER ); // " > "
glDisable( GL_DEPTH_TEST ); // Désactiver ce test fait en sorte que le
                            // tampon de profondeur ne sera pas modifié
```

- valeurs subséquentes dans les tampons	0,7	1,0 0,9 0,8 0,7

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

c)

	profondeur (z)	*couleur (r, g, b, a)*	
- valeurs des attributs du fragment	0.3	(1.0, 0.9, 0.8, 0.7)	0,7
- valeurs présentes dans les tampons	0.7	(0.4, 0.4, 0.4, 0.4)	0,3

```
glDepthFunc( GL_ALWAYS ); // " toujours "
glEnable( GL_DEPTH_TEST );
glBlendFunc( GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA );
glEnable( GL_BLEND );
```

- valeurs subséquentes dans les tampons	0,3	0,82, 0,75, 0,68, 0,61

d)

	profondeur (z)	*couleur (r, g, b, a)*	
- valeurs des attributs du fragment	0.3	(1.0, 0.9, 0.8, 0.7)	1
- valeurs présentes dans les tampons	0.7	(0.4, 0.4, 0.4, 0.4)	0,3

```
glDepthFunc( GL_LESS ); // " < "
glEnable( GL_DEPTH_TEST );
glBlendFunc( GL_ONE, GL_ONE_MINUS_SRC_ALPHA );
glEnable( GL_BLEND );
```

- valeurs subséquentes dans les tampons	0,3	1.0 1.0 0,92 0,82

e)

	profondeur (z)	*couleur (r, g, b, a)*	
- valeurs des attributs du fragment	0.3	(1.0, 0.9, 0.8, 0.7)	1
- valeurs présentes dans les tampons	0.7	(0.4, 0.4, 0.4, 0.4)	0

```
glDepthFunc( GL_NEVER ); // " jamais "
glEnable( GL_DEPTH_TEST );
glBlendFunc( GL_ONE, GL_ZERO );
glEnable( GL_BLEND );
```

- valeurs subséquentes dans les tampons	0,7	0,4 . 0,4 0,4 0,4

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

f)

	profondeur (z)	*couleur (r, g, b, a)*	*stencil*
- valeurs des attributs du fragment	0.3	(1.0, 0.9, 0.8, 0.7)	-
- valeurs présentes dans les tampons	0.7	(0.4, 0.4, 0.4, 0.4)	5

```
glDepthFunc( GL_GREATER ); // " > "
glEnable( GL_DEPTH_TEST );
glDisable( GL_BLEND );
glStencilFunc( GL_GEQUAL, 3, 7 ); // " >= "
glStencilOp( GL_REPLACE, GL_DECR, GL_INCR );
glEnable( GL_STENCIL_TEST );
```

- valeurs subséquentes dans les tampons	0,7	0,4 0,4 0,4 0,4	3

g)

	profondeur (z)	*couleur (r, g, b, a)*	*stencil*
- valeurs des attributs du fragment	0.3	(1.0, 0.9, 0.8, 0.7)	-
- valeurs présentes dans les tampons	0.7	(0.4, 0.4, 0.4, 0.4)	5

```
glDepthFunc( GL_LESS ); // " < "
glEnable( GL_DEPTH_TEST );
glDisable( GL_BLEND );
glStencilFunc( GL_NEVER, 3, 7 ); // " jamais "
glStencilOp( GL_REPLACE, GL_DECR, GL_INCR );
glEnable( GL_STENCIL_TEST );
```

- valeurs subséquentes dans les tampons	0,7	0,4 0,4 0,4 0,4.	3

h)

	profondeur (z)	*couleur (r, g, b, a)*	*stencil*
- valeurs des attributs du fragment	0.3	(1.0, 0.9, 0.8, 0.7)	-
- valeurs présentes dans les tampons	0.7	(0.4, 0.4, 0.4, 0.4)	5

```
glDepthFunc( GL_LESS ); // " < "
glEnable( GL_DEPTH_TEST );
glDisable( GL_BLEND );
glStencilFunc( GL_LESS, 3, 7 ); // " <= "
glStencilOp( GL_REPLACE, GL_DECR, GL_INCR );
glEnable( GL_STENCIL_TEST );
```

- valeurs subséquentes dans les tampons	0,3	1,0 0,9 0,8 0,7	4

i)

	couleur (sur 4 bits)
- valeurs des attributs du fragment	0011
- valeurs présentes dans le tampon	0101

0011
⊕ 0101
0110

```
glLogicOp( GL_XOR );
glEnable( GL_COLOR_LOGIC_OP );
```

- valeurs subséquentes dans le tampon | 0110

j)

	couleur (sur 4 bits)
- valeurs des attributs du fragment	0011
- valeurs présentes dans le tampon	0101

0011
+ 0101
0111

```
glLogicOp( GL_OR );
glEnable( GL_COLOR_LOGIC_OP );
```

- valeurs subséquentes dans le tampon | 0111

k)

	couleur (sur 4 bits)
- valeurs des attributs du fragment	0011
- valeurs présentes dans le tampon	0101

```
glLogicOp( GL_COPY ); // la valeur par défaut
glEnable( GL_COLOR_LOGIC_OP );
```

- valeurs subséquentes dans le tampon | 0011

Cet examen comprend 4 questions sur 9 pages pour un total de 40 points

Benoît Oze