

POLYTECHNIQUE
MONTRÉAL

Questionnaire examen final

INF2705

Sigle du cours

Identification de l'étudiant(e)

Nom : _____ Prénom : _____

Sigle et titre du cours

INF2705 - Infographie

Professeur	Groupe	Trimestre	
Benoît Ozell	Tous	20203	
Jour	Date	Durée	Heures
Vendredi	11 déc. 2020	2h30	13h30
Documentation	Calculatrice	Outils électroniques	
<input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Toute <input type="checkbox"/> Voir directives particulières	<input checked="" type="checkbox"/> Aucune <input type="checkbox"/> Toutes	Les appareils électroniques personnels sont interdits.	

Directives particulières

- Le professeur ne répondra à aucune question durant cet examen. Si vous estimez que vous ne pouvez pas répondre à une question pour diverses raisons, veuillez le justifier puis passer à la question suivante.
- Il est strictement interdit de débrocher l'examen.
- IMPORTANT : inscrire votre matricule sur toutes les pages numérotées.

Cet examen contient **5** questions sur un total de **14** pages (incluant cette page).

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

Question 1 Notions de base [11 points]

- a) [2 points] Quelle est la différence principale entre une projection orthographique et une projection perspective ?

La différence globale est l'angle d'ouverture

Pour la perspective, il est variable

Alors que pour la projection ortho, il n'existe pas !

- b) [2 points] En infographie, qu'est-ce qu'un *lutin* et quel est l'intérêt d'en utiliser dans un logiciel d'infographie ?

Un lutin est simplement une partie d'une grande texture qui simule des frames (frames) de mouvement d'un objet.

Les lutins peuvent être utilisés pour simuler le mouvement de comportants, particules dans un logiciel. *texture sur plusieurs
mème sans mouvement*

- c) [3 points] Dans le pipeline graphique programmable, nous avons utilisé le produit des matrices de modélisation, de visualisation et de projection par le sommet courant afin d'obtenir une position à l'écran. Donnez deux exemples distincts qui illustrent l'utilité d'avoir accès, dans les nuageurs, aux trois matrices séparément plutôt qu'à une seule matrice qui contiendrait le produit des trois. Justifiez vos réponses.

1) Calcul de l'illumination : les vecteurs \vec{I} \vec{O} \vec{N} sont dans le repère de la caméra. Il nous faut donc les coordonnées du sommet dans celui-ci.

2) Translation : Pour trouver les coordonnées de translation, dans le nouveau Tex Eval, il faut que les coordonnées de la patch soient dans le repère de la caméra. On multiplie par la matrice de projection après !

d) [4 points] Nous avons vu que les nuanceurs de tessellation permettent de subdiviser une primitive en y générant des triangles intermédiaires. Les sous-triangles ainsi générés ajoutent toutefois au temps de traitement nécessaire pour l'affichage de la scène.

i) Donnez *deux* exemples, autres que celui du TP3, où il est utile et pertinent de raffiner ainsi l'affichage d'une primitive. Justifiez vos réponses.

1)

Pour raffiner les contours d'un objet, il faut beaucoup de triangles. On ne veut pas garder tous ces triangles sur le CPU et les envoier au GPU. Une façon efficace est donc d'utiliser le nuancier de tessellation.

2)

Si on veut que l'objet soit beaucoup raffiné quand il est proche de la caméra vs moins raffiné quand il est loin. Utiliser le raffinement est utile car on peut contrôler le niveau de tessellation selon la distance à la caméra.

C

ii) À l'inverse, donnez *deux* exemples où il est inutile et contreproductif de raffiner ainsi l'affichage d'une primitive. Justifiez vos réponses.

1)

Illumination de Phong : Comme les fragments et leurs normales vont être tout le temps interpolées, il est contreproductif de raffiner les primitives.

2)

Application de textures dans le cas d'un panneau : Comme les coordonnées de texture vont être interpolées sur toute le quadrilatère, il n'y a pas besoin de raffiner l'affichage.

Question 2 Illumination [14 points]

a) [8 points] Dans le modèle de réflexion locale de la lumière communément utilisé en infographie, trois sortes de réflexion sont définies : *ambiante*, *diffuse* et *spéculaire*. Pour chaque élément ci-dessous, identifiez toutes les réflexions (parmi ces trois) sur laquelle ou lesquelles cet élément a une influence ; écrivez « *aucune* » si l'élément n'a aucune influence.

- Le vecteur normal à la surface. diffuse spéculaire

- Le coefficient de brillance. Spéculaire

- La position de la source lumineuse. diffuse spéculaire

- La position de l'observateur. spéculaire

Q

- La position des autres objets de la scène. Aucune

- Le nombre total d'objets dans la scène. Aucune

- Les propriétés de matériau de l'objet. ambiante diffuse spéculaire

- Un autre objet situé entre l'objet éclairé et la source lumineuse. Aucune

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

Tous les directions.

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

Tous les directions

d) [4 points] Dans la dernière partie du TP2 avec les poissons-théières, les vecteurs vers la source de lumière et vers l'observateur étaient constants :

lumiDir = vec3(0, 0, 1); obsVec = vec3(0, 0, 1);

L'énoncé mentionnait rapidement que « *ces simplifications pour la position de la lumière et pour la position de l'observateur sont souvent utilisées pour que les calculs d'illumination soient plus simples* ».

Expliquez pourquoi les calculs d'illumination sont alors plus simples.

1.) Tout d'abord \vec{L} et \vec{o} sont normalisés

2.) Réflexion diffuse $\vec{N}(x, y, z) \cdot \vec{L}(0, 0, 1) = 3^!$

3.) Réflexion spéculaire Blinn $\rightarrow B = \frac{\vec{L} + \vec{o}}{\|\vec{L} + \vec{o}\|} = (0, 0, 1) \quad \vec{N}(x, y, z)$
d'où $\vec{B} \cdot \vec{N} = 3^!$

Réflexion spéculaire Phong $\rightarrow \text{reflect}(-\vec{L}, \vec{N})$ où $\vec{N}(x, y, z)$

$$\vec{R} = 2\vec{N}(\vec{L} \cdot \vec{N}) - \vec{L} = 2\vec{N} - \vec{L}$$

$$\vec{R} \cdot \vec{N} = \vec{N} \cdot (2\vec{N} - \vec{L}) = 2\vec{N} \cdot \vec{N} - \cancel{\vec{L} \cdot \vec{N}}^3 \quad \text{car } \vec{N} \text{ normal} \\ = 2\vec{N} \cdot \vec{N} = 2 = 3^! \quad \underline{\underline{=}}$$

On a donc $\vec{N} \cdot \vec{L} = \vec{B} \cdot \vec{N} = \vec{R} \cdot \vec{N} = 3$ qui est la coordonnée normalisée de $\vec{N}^!$

Question 3 Nuanceurs en GLSL [12 points]

a) [5 points] Dans le cadre d'un projet en infographie, on vous donne un programme avec les nuanceurs listés ci-dessous (cette page et la suivante). Vous constatez que le code GLSL est assez mal documenté et que les noms de variables ne sont pas toujours explicites. :(

Écrivez les dix (10) commentaires manquants dans les `main()` des nuanceurs de sommets et de fragments. Afin de montrer que vous comprenez ce que fait l'énoncé, débutez avec un verbe à l'infinitif et décrivez bien le but de l'énoncé dans votre commentaire. (Par exemple, en référence au dernier TP avec les lutins, préférez «*// appliquer la gravité à cette particule*» plutôt que simplement «*// calculer la vitesse*».)

i) Le nuanceur de sommets "nuanceurSommets.glsl" :

```
#version 410
layout (std140) uniform LightSourceParameters
{
    vec4 ambient, diffuse, specular, position;
} LightSource;

uniform mat4 matrModel, matrVisu, matrProj;
uniform mat3 matrNormale;

layout(location=0) in vec4 Vertex;
layout(location=2) in vec3 Normal;
layout(location=3) in vec4 Color;

out Attribs { vec3 l, n, o; } AttribsOut;

1 void main(void)
2 {
3
4     // ? Calculer la position du sommet à l'écran
5     gl_Position = matrProj * matrVisu * matrModel * Vertex;
6
7
8     // ? Calculer la position du sommet dans le réfère de la caméra
9     vec3 p = (matrVisu * matrModel * Vertex).xyz;
10
11    // ? Recalculer la normale au vu de la transformation de matrModel le faire
12    AttribsOut.n = matrNormale * Normal;
13
14    // ? Calculer le vecteur entre le point et la source de lumière (l' de l'illumination )
15    AttribsOut.l = (matrVisu * LightSource.position).xyz - p;
16
17
18    // ? Calculer le vecteur d'observation de l'illumination
19    AttribsOut.o = normalize(-p);
20
21 }
```

ii) Le nuanceur de fragments "nuanceurFragments.glsl":

```
#version 410
layout (std140) uniform LightSourceParameters
{
    vec4 ambient, diffuse, specular, position;
} LightSource;

layout (std140) uniform MaterialParameters
{
    vec4 emission, ambient, diffuse, specular;
    float shininess;
} FrontMaterial;

in Attribs { vec3 l, n, o; } AttribsIn;

out vec4 FragColor;

28 void main(void)
29 {
30     vec3 L = normalize( AttribsIn.l );
31     vec3 N = normalize( AttribsIn.n );
32     vec3 O = normalize( AttribsIn.o );
33
34
35
36     // ? Calculer la composante ambiaante
37     vec4 c = FrontMaterial.ambient * LightSource.ambient;
38
39
40
41
42     // ? Calculer le coefficient d'atténuation
43     float d = length( AttribsIn.l );
44     float a = min( 1.0, 1.0 / ( 0.01 + 0.05 * d + 0.1 * d * d ) );
45
46
47
48
49     // ? Calculer le produit scalaire  $\vec{N} \cdot \vec{L}$ . S'il est négatif, mettre 0.
50     float f = max( 0.0, dot( N, L ) );
51
52
53
54
55     // ? Calculer la composante diffuse
56     c += a * FrontMaterial.diffuse * LightSource.diffuse * f;
57
58
59
60
61     // ? Remettre la couleur dans l'intervalle [0, 1]
62     FragColor = clamp( c, 0.0, 1.0 );
63
64 }
```

- b) [1 point] Comment nomme-t-on le *modèle d'illumination* implémenté dans les nuanceurs de la sous-question précédente ?

Modèle Phong.

- c) [3 points] Quel rendu visuel produit habituellement le *modèle d'illumination* implémenté dans ces nuanceurs, le rendu A ou le rendu B ? Justifiez succinctement votre raisonnement.



rendu A :



rendu B :

3

Rendu A , car le rendu B a un effet d'interpolation de couleurs .
On sait que Phong ne fait pas ça . Le rendu B est brouillant

- d) [3 points] Dans les nuanceurs fournis, aucune réflexion spéculaire n'est calculée. Écrivez les énoncés manquants dans ces nuanceurs afin d'ajouter le calcul de la réflexion spéculaire selon Blinn, ce qui donnera les rendus C ou D ci-dessous. Indiquez clairement entre quelles lignes s'insèrent vos ajouts.



rendu C :



rendu D :

Entre les lignes 57 - 63 :

$$f = \max(0.0, \text{dot}(N, \text{normalize}(L + O))) ;$$

$$C += ax \text{FrontMaterial.specular} \times \text{LightSource.Specular} \times \text{pow}(f, \text{FrontMaterial.shininess}) ;$$

Question 4 Utilisation des textures [7 points]

Un petit logiciel graphique charge les deux images "scene.bmp" et "neige.bmp" (voir ci-dessous) dans deux unités de texture distinctes et trace deux triangles formant un carré. Les deux textures sont utilisées sur ce carré pour produire l'image *Scène complètement enneigée* (à droite ci-dessous).



scene.bmp



neige.bmp



Scène complètement enneigée

- a) [3 points] Le programme principal `main.cpp` ainsi que le nuanceur de sommets ont fait tout ce qui est nécessaire pour le bon fonctionnement de l'application. Complétez le nuanceur de fragments en GLSL afin de produire le résultat désiré. (Notez bien que la scène n'est pas plus sombre lorsqu'enneigée.)

```
#version 410
uniform sampler2D laTextureScene, laTextureNeige;
in Attribs { vec2 texCoord; } AttribsIn;
out vec4 FragColor;
void main( void )
{
    // vec4 texture( sampler2D sampler, vec2 coo );
    Vec2 texCoord = AttribsIn.texCoord;
    vec4 coul = texture( laTextureScene, texCoord ) + texture( laTextureNeige, texCoord );
    FragColor = clamp( coul, 0.0, 1.0 );
}

```

3

- b) [2 points]** On souhaite donner l'illusion que les flocons de neige tombent, c'est-à-dire que la position des flocons varie en fonction du temps, comme illustré dans les images *Carrés texturés de façon variable selon le temps* ci-dessous.



Carrés texturés de façon variable selon le temps (à t=0.0, t=0.25, t=0.5, t=0.85, t=1.0 seconde).

L'animation fait en sorte que les flocons de neige descendent régulièrement, disparaissent en bas de l'image pour réapparaître en haut, et continuent à descendre régulièrement. Dans ces images, observez le déplacement du flocon entouré d'un rectangle rouge : il descend régulièrement puis revient en haut pour continuer sa descente. Un flocon de neige repasse ainsi à sa position initiale chaque seconde.

En vue de produire l'animation avec les nuanceurs, le programme principal transmet aux nuanceurs la variable uniform float temps (en secondes), initialisée à 0 et incrémentée de façon régulière par le programme principal.

Écrivez les modifications à faire au nuanceur de fragments de la sous-question précédente (ou réécrivez-le) afin de produire le résultat désiré.

```
#version 410
uniform float temps;
uniform sampler2D laTextureScene, laTextureNeige;
in Attribs { vec2 texCoord; } AttribsIn;
out vec4 FragColor;
void main( void )
{
```

2

vec2 texCoord = AttribsIn.texCoord;

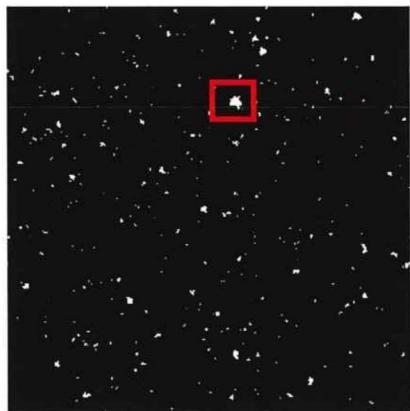
vec2 depTexCoord = vec2(texCoord.s, texCoord.t + temps - int(temps));
partie entière !

vec4 coul = texture(laTextureScene, texCoord) + texture(laTextureNeige, depTexCoord);

FragColor = clamp(coul, 0.0, 1.0);

}

- c) [2 points] En utilisant toujours les mêmes textures et le même programme principal et nuancier de sommets, on souhaite afficher des flocons de neige qui tombent (comme précédemment), mais avec une neige qui est deux fois plus dense, comme illustré dans l'image *Scène et neige plus dense* ci-dessous. (Notez que les flocons sont plus nombreux, mais ils n'ont pas changé de taille.)



neige.bmp



Scène et neige (version originale)



Scène et neige plus dense

Écrivez les modifications à faire au nuancier de fragments de la sous-question précédente (ou réécrivez-le) afin de produire le résultat désiré. (Indice : le même flocon encadré de rouge est présent à quatre endroits.)

```
#version 410
uniform float temps;
uniform sampler2D laTextureScene, laTextureNeige;
in Attribs { vec2 texCoord; } AttribsIn;
out vec4 FragColor;
void main( void )
{
    vec2 texCoord // question précédente
    -> vec2 depthCoord // question précédente

    Z vec4 coul = texture ( laTextureNeige , depthCoord );
    coul += texture ( laTextureNeige , vec2( depthCoord.s , depthCoord.t + 0.5 ) );
    coul += texture ( laTextureNeige , vec2( depthCoord.s + 0.5 , depthCoord.t ) );
    coul += texture ( laTextureNeige , depthCoord + vec2( 0.5 ) );
    coul += texture ( laTextureScene , texCoord );

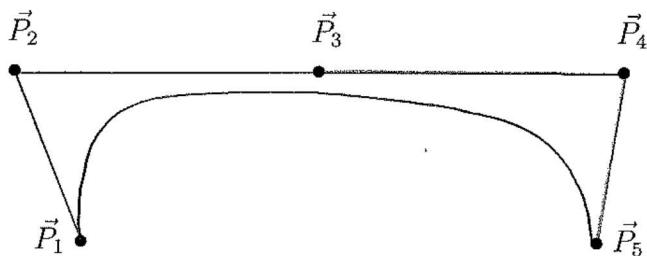
    FragColor = clamp ( coul , 0.0 , 1.0 );
}
```

}

Question 5 Courbes paramétriques polynomiales [6 points]

a) [3 points] Considérez cinq points de contrôle $\{\vec{P}_1, \vec{P}_2, \vec{P}_3, \vec{P}_4, \vec{P}_5\}$ utilisés pour construire une courbe de Bézier.

i) Dessinez la courbe de Bézier utilisant ces cinq points de contrôle.



3

ii) Par lequel ou lesquels des cinq points de contrôle la courbe de Bézier passe-t-elle certainement ? Pourquoi ?

par \vec{P}_1 et \vec{P}_5 . Le premier et dernier !

$$\text{On a } J_0(t) = \binom{4}{0} t^0 (1-t)^{4-0} = (1-t)^4 \text{ on a donc } J_0(0) = 1 \cdot P_1$$

$$J_4(t) = \binom{4}{4} t^4 (1-t)^{4-4} = t^4 \text{ on a donc } J_4(1) = 1 \cdot P_5$$

pour $t=0$, on passe par P_1 et pour $t=1$ on passe par P_5 !

iii) Quelle(s) contrainte(s) devrait-on imposer à ces points de contrôle pour que la courbe de Bézier passe par les cinq points ?

Ces points devront être alignés !

b) [2 points] Identifiez deux différences fondamentales entre les courbes de Bézier et les splines cubiques.

1)

Splines cubiques \rightarrow courbe d'interpolation (passe par tous les points contrôles)
 Bézier \rightarrow courbe d'approximation (ne passe pas forcément par tous les points contrôles)

2

2)

Les fonctions de base d'une Bézier sont des polynômes de même ordre

Selon : $\binom{n}{i} t^i (1-t)^{n-i}$ ordre variable

Alors que les fonctions de la spline cubique sont simplement les polynômes d'Hermitte. Ordre polynomial FIXE



c) [1 point] Les fonctions de base d'une B-Spline satisfont toujours la relation $\sum_{i=1}^n N_{i,k}(t) = 1$ pour toute valeur de k ou de t . Expliquez en quoi cette propriété est utile.

Cette propriété assure que la courbe résultante va toujours être incluse dans l'enveloppe convexe formée par les points de contrôle !
 On pourra savoir si des courbes de Bézier ne s'intersectent facilement !



Bon hiver !

Cet examen comprend 5 questions sur 14 pages pour un total de 50 points.

Benoît Ozell

(Si vous utilisez cette page pour répondre à une question, inscrivez clairement le numéro de la question.)