Scuola (scuola.*)

Escriu VS+FS per re-il·luminar (usant **plane.obj**) una imatge basada en la famosa obra del pintor Rafael Sanzio, *La scuola di Atene*. Us proporcionem tres textures (scuola.jpg, scuola-depth.jpg i scuola-normal.jpg) que representen el color, la profunditat (estimada) i la normal (estimada):


scuola.jpg


scuola-depth.jpg


scuola-normal.jpg

El VS farà les tasques imprescindibles, passant-li al FS la informació que necessiti.

El FS calcularà el color en funció d'un uniform int mode = 0.

Si **mode és 0** [2 punts], el color del fragment serà simplement el color original (el del colorMap).

Si **mode és 1** [4 punts], el color serà el color original, multiplicat per un factor f1 que dependrà de la profunditat representada a la textura depthMap1. Concretament, el factor serà (1-d)*AO on d és la profunditat obtinguda del depth map, i AO és el uniform float AO = 1.5. Com que només volem enfosquir les parts llunyanes, feu que el factor que apliqueu no sigui més gran que 1.

Si **mode és 2** [3 punts], el color serà el color original, multiplicat per un factor f2 que dependrà de la il·luminació. Concretament, el factor serà N·L, on N és la normal unitària calculada a partir del normal map (normalMap2), i L és el light vector.

Pel que fa a la normal, com que el vector que us retorna el normal map tindrà components en [0, 1], caldrà que les transformeu a valors en [-1, 1] abans de normalitzar. Atesa l'orientació del pla a la càmera per defecte, i <u>només per a aquest exercici</u>, heu d'assumir que la normal ja està en eye space.

Pel que fa al light vector, l'heu de calcular com feu habitualment, però amb una llum que descriu (en eye space) un cercle de radi R situat sobre el pla Z=1 (de la càmera) i centrat al punt (0,0,1). La llum donarà una volta completa cada segon, començant al punt (1,0,1) i girant al voltant de l'eix Z de la càmera en sentit antihorari.

Si **mode és 3** [1 punt], el color del fragment incorporarà els dos factors f1 i f2 dels modes anteriors.

Gràfics
Curs 2024/25 Q2

Aquí teniu un exemple del resultat esperat:


mode 0


mode 1


mode 2, time 0


mode 3, time 0

Fitxers i identificadors	
Fitxers a lliurar (respecteu minúscules; individualment o en un zip sense carpetes)	`scuola.vert` `scuola.frag`

Fitxers a ~/assig	`plane.obj`
	`scuola.jpg`
	`scuola-depth.jpg`
	`scuola-normal.jpg`

Fitxers que acompanyen l'enunciat	-

Constants	`const float PI = 3.141592;`
	`const vec3 golden = vec3(1.0, 0.84, 0.0);`
	`const float shininess = 5.0;`

Uniforms (a banda dels de l'enunciat)	`uniform sampler2D colorMap, depthMap1, normalMap2;`
	`uniform float R = 2;`
	`uniform float time;`

Coin (coin.*)

En aquest exercici volem implementar la moneda d'un joc *arcade* i necessitem calcular la seva il·luminació i animació. Escriu un **VS+FS** per tal d'animar i calcular la il·luminació de l'objecte **coin.obj**. El vídeo **coin.mp4** conté el resultat final esperat. L'exercici es puntua depenent de la variable **uniform int mode;** les puntuacions màximes que teniu per cada mode són orientatives.

mode = 0

mode = 1 (time = 0.4)

mode = 1 (time = 0.85)

<u>Si mode és 0</u> [5 punts]

El **VS** farà les tasques imprescindibles.

El **FS** serà l'encarregat d'il·luminar la moneda amb un aspecte metàl·lic, tal com es detalla a continuació:

Declareu les constants:
const vec3 golden = vec3(1.0, 0.84, 0.0);
const float shininess = 5.0;

La il·luminació estarà basada en el model de Phong amb components ambient, difosa i especular. Per a la llum ambient utilitza el color **(0.2, 0.2, 0.2)** i per a la llum difusa el color **(0.7, 0.7, 0.7)**. Tant el material ambient com difós seran de color **golden**. Per a la component especular fes servir la constant **shininess** amb una llum i material blancs.

La il·luminació serà direccional, és a dir, ha de ser equivalent a una font de llum infinitament lluny que il·lumina tots els punts de l'objecte amb la mateixa direcció. La direcció dels raigs de llum cap a l'objecte serà, en object space, **(-1, -1, -1).**

Finalment, volem ressaltar la il·luminació al canto de la moneda. Per aconseguir aquest efecte hem utilitzat un valor de shininess baix; però, quan mirem de cara el model, la taca especular és massa gran. Les coordenades de textura **st** del model **coin.obj** estan dissenyades de forma que les parts més especulars del model tinguin una **s** més petita que **0.5**, i les parts menys especulars tenen una **s** més gran que **0.5**. Per tant, multiplica el terme especular per un factor de **0.5** quan **s >= 0.5**.

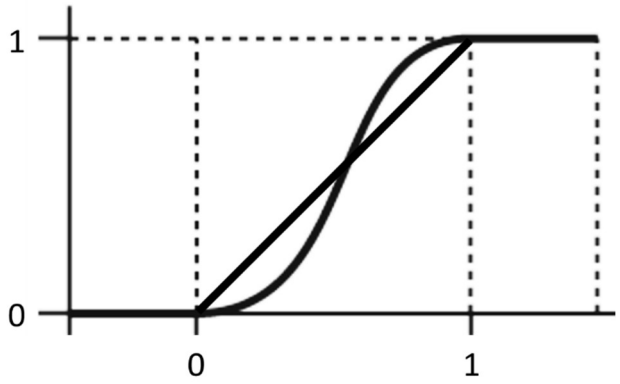
Gràfics
Curs 2024/25 Q2

<u>Si mode és 1</u> [5 punts]

El **FS** utilitzarà la il·luminació del mode 0 i farà les tasques imprescindibles.

El **VS** animarà la moneda perquè faci un salt i dues rotacions. L'animació comença cada **3 segons** (cíclicament) però el salt i les dues rotacions només s'executen durant el **primer segon**.

En comptes d'utilitzar directament la variable time per controlar el salt i les rotacions, l'animació estarà controlada per una variable $\boldsymbol{t} \in [\mathbf{0}, \mathbf{1}]$ calculada amb la funció **smoothstep** a partir del temps de l'animació. Així, l'animació del salt i les rotacions serà més suau que si usem directament time:



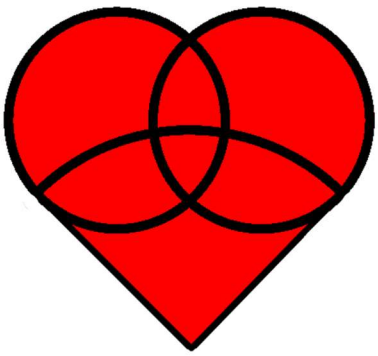
Per al desplaçament vertical durant el salt, utilitza la funció **cos** amb una amplitud de **0.5**. A més, durant el salt, la moneda ha de fer dues rotacions completes. Modifica el que creguis necessari per a mantenir una il·luminació correcta durant tot el salt. Recorda que la matriu de rotació en 3D, per l'eix Y i un angle α és:

$$\begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

Fitxers i identificadors	
Fitxers a lliurar (respecteu minúscules; individualment o en un zip sense carpetes)	`coin.vert` `coin.frag`
Fitxers a ~/assig	`coin.obj`
Fitxers que acompanyen l'enunciat	`coin.mp4`
Constants (a banda de les de l'enunciat)	`const float PI = 3.141592;`
Uniforms (a banda dels de l'enunciat)	`uniform int mode = 0;` `uniform float time;`

heart (heart.*)

Escriu un VS+FS que, de manera procedural, generi un cor. Podeu suposar que aquest shader el provarem només amb l'objecte Plane, que té coordenades de textura del (0,0) al (1,1). Els punts, radis i distàncies que mencionem en aquest exercici fan referencia a coordenades **en espai de textura**.



Observeu que el fons és de color blanc, i el cor és de color vermell. La forma del cor la podem aproximar amb dos cercles a la part d'amunt, i un tercer de cercle a la part d'avall. Els dos cercles d'amunt tenen **radi 0.225 i centres C1 = (0.35, 0.6) i C2 = (0.65, 0.6)**. El tercer cercle d'avall té **radi 0.45 i centre C3 = (0.5, 0.13)**. D'aquest tercer cercle només aprofitem un sector.

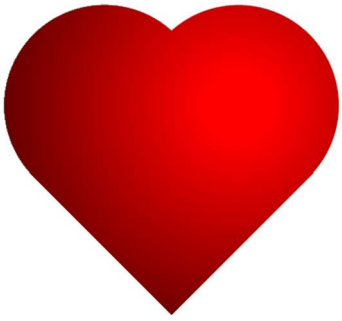
El color final del fragment dependrà d'un `uniform int mode = 0`.

Si `mode` és `0` **[4 punts]**, el color serà blanc o vermell, segons si és fons o si forma part del cor.

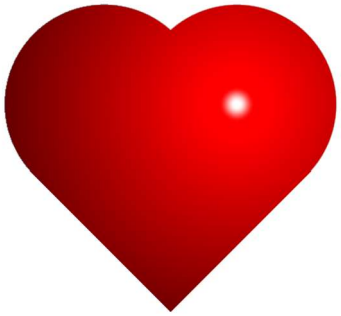
Si `mode > 0` **[2 punts]** haureu d'aplicar un antialiàsing a la frontera fent servir la funció `aastep`:

```
float aastep(float threshold, float x)
{
	float width = 0.7*length(vec2(dFdx(x), dFdy(x)));
	return smoothstep(threshold-width, threshold+width, x);
}
```

Si `mode > 1` **[2 punts]**, simularem un ombrejat del cor fent que el color dels fragments estigui modulat en funció de la distancia `d` del fragment al centre del cercle de la dreta (**C2**): per això multiplicarem aquests fragments per `max(1.1 - d / 0.75, 0)`.



Si **`mode > 2` [1 punt]**, simularem una taca especular blanca, circular, al mateix centre **C2**. Feu servir la funció **`smoothstep`** per a que la taca tingui intensitat màxima quan **`d < 0.005`**, i que aquesta intensitat decreixi suaument a partir d'aquí fins a ser `0` quan **`d > 0.04`**.



Finalment, continuant amb mode > 2, volem que el cor tingui una **animació** com si estigués batejant **[1 punt]**. Per això afegiu un escalat als vèrtexs en espai objecte amb un factor d'escala **`S`** depenent del temps **`time`** en segons:

```
S = 0.5 + abs(sin(time*2))*0.2
```

Nota: en aquest exercici, la sortida de cada `mode` és incremental i ha d'incloure la feina dels modes anteriors. Així, per exemple, si `mode == 3` s'espera que hi hagi antialiàsing, ombrejat i taca especular a la vegada, i si `mode == 2` només s'espera antialiàsing i ombrejat.

Fitxers i identificadors	
Fitxers a lliurar (respecteu minúscules; individualment o en un zip sense carpetes)	`heart.vert` `heart.frag`
Fitxers a ~/assig	`plane.obj`
Fitxers que acompanyen l'enunciat	`-`
Constants (a banda de les de l'enunciat)	`-`
Uniforms (a banda dels de l'enunciat)	`uniform int mode = 0;` `uniform float time;`