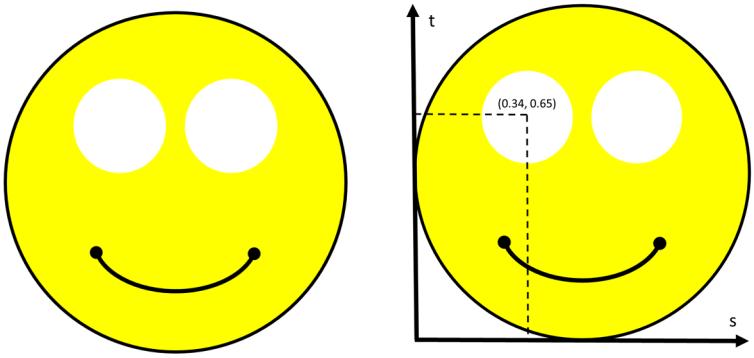
Smile (smile.*) 2.5 punts

*Useu **/assig/grau-g/viewer** en tots els exercicis.*

Escriu **VS+FS** per texturar l'objecte **plane.obj** amb un *smile* amb ulls animats.

La textura que fareu servir (smile.png) té els ulls en blanc:

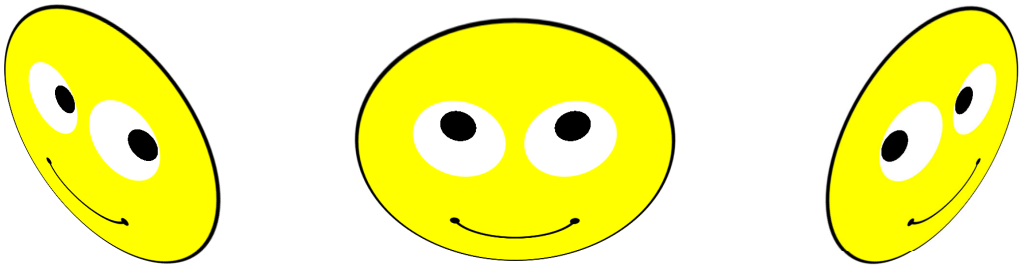


Els centres dels ulls tenen coordenades de textura C1= **(0.34, 0.65)** i C2=**(0.66, 0.65).**

El VS farà les tasques per defecte, però no ha de calcular cap color.

El FS bàsicament escriurà directament el color de la textura, amb l'excepció de l'iris de cada ull, que serà negre. Siguin (s,t) les coordenades de textura del fragment (vtexCoord). Si assumim que el centre de cada iris coincideix amb el centre de cada ull, el fragment haurà de ser negre si (s,t) és dins el cercle de radi 0.05 centrat en C1 o C2.

Volem però que l'iris es mogui com si mirés a la càmera:



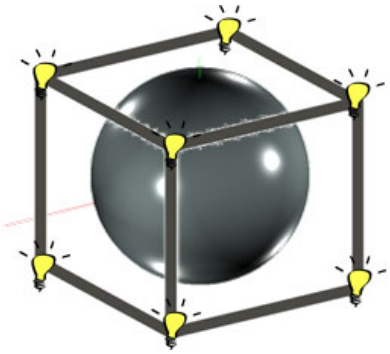
Per aconseguir aquest efecte, considerarem que el centre de l'iris es desplaça respecte el centre de l'ull una distància que depèn de la normal N en *eye space*. Concretament, el desplaçament de C1 i C2 serà, per tots dos ulls, de **-0.1*N.xy**. Aquests nous centres són els que determinaran la posició dels iris.

Identificadors obligatoris:

smile.vert, smile.frag *(minúscules!)*
uniform sampler2D colormap;

8lights (8lights.*) 2.5 punts

Escriu **VS+FS** per aplicar il·luminació de Phong **per fragment**, amb **8 llums fixos respecte l'escena**. Concretament, les posicions dels llums en *world space* coincidiran amb els 8 vèrtexs de la capsa contenidora de l'escena (useu boundingBoxMin i boundingBoxMax per obtenir la posició d'aquests llums).



El VS farà les tasques habituals i passarà al FS les dades necessàries (vèrtex i normal) pel càlcul d'il·luminació.

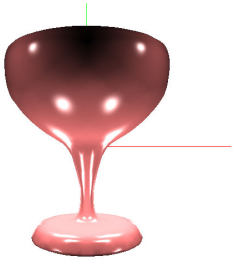
El FS calcularà el color del fragment acumulant la contribució dels *8* llums. Per evitar imatges massa saturades, useu per cada llum l'expressió

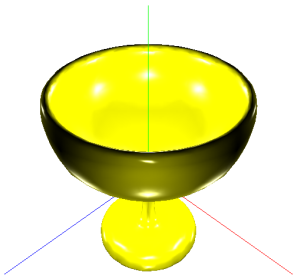
$$\sum K_d\, I_d\, (N \cdot L_i)/\mathbf{2} \;+\; K_s\, I_s\, (R_i \cdot V)^S$$

la qual **ignora la contribució ambient** i **divideix la contribució difosa per 2.**

Pels llums i material usa les propietats habituals (matDiffuse, matSpecular, lightDiffuse, lightSpecular...).

Vigila amb l'eficiència, per exemple, mira de no fer crides innecessàries a normalize().





Identificadors obligatoris:

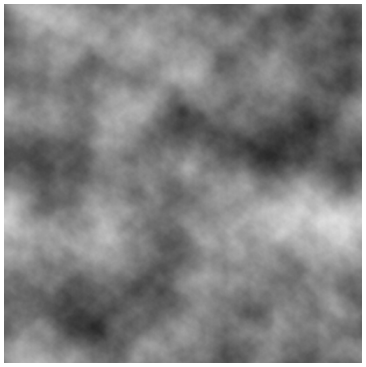
8lights.vert, 8lights.frag *(minúscules!)*

Plasma (plasma.*) 2.5 punts

Escriu VS+FS que implementi l'efecte conegut com a plasma.

El VS farà les tasques imprescindibles. No cal cap tipus d'il·luminació.

El FS accedirà a la textura **fbm.png** que us proporcionem, fent servir les coordenades de textura interpolades que arriben des del VS, i es quedarà amb el component vermell que anomenarem **r**.

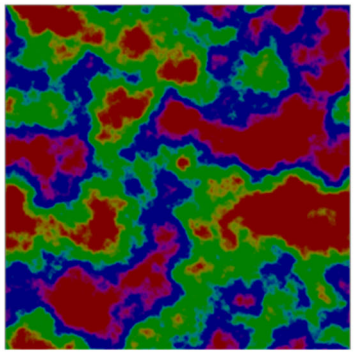


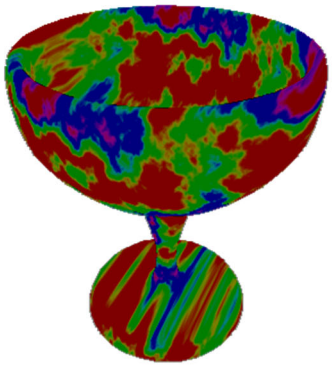
Aquest valor el farem servir per calcular una funció sinusoïdal amb els paràmetres següents:

$A = 1$	Amplitud
$f = 0.1$	Freqüència (en Hz)
$\varphi = 2\pi\mathbf{r}$	Fase (que depèn del component vermell **r** de la textura).

El valor resultant d'aquesta sinusoïdal v el transformarem en un color que copiarem a fragColor. Donat que v és el resultat d'una sinusoïdal d'amplitud 1, els seus valors es trobaran al rang [-1, 1].

Haureu d'assignar linealment aquest rang al gradient de colors format per la interpolació de: **vermell, groc, verd, cian, blau, magenta i vermell**, en aquest ordre. Per a fer la interpolació entre cada parella de colors podeu fer servir la funció mix. Un valor v de -1 es correspondrà al principi del gradient i es pintarà vermell, un valor v de 1 terminarà al final de gradient i també serà vermell, en canvi un valor v de 0 es pintarà de color cian.





Identificadors obligatoris:

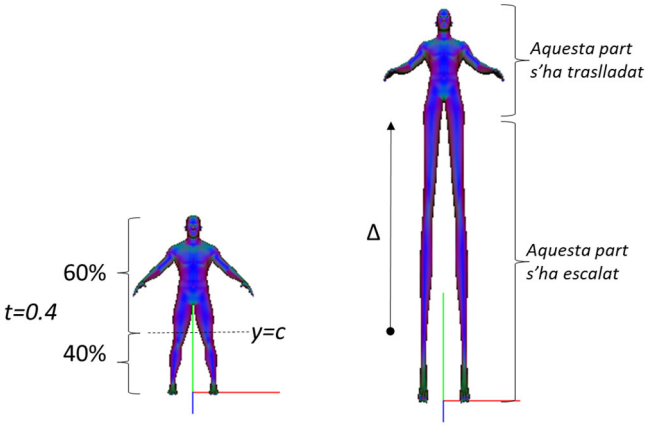
plasma.vert, plasma.frag *(minúscules!)*
uniform sampler2D fbm;
uniform float time;
const float pi = 3.14159;

Dalify (dalify.*) 2.5 punts

Escriu **VS+FS** per deformar el model en direcció vertical (eix Y en *model space*), per obtenir una aparença similar a la d'alguns animals en quadres de Salvador Dalí:

Les temptacions de Sant Antoni (Salvador Dalí, 1946)

El VS deformarà el model modificant únicament la coordenada Y en *model space*:

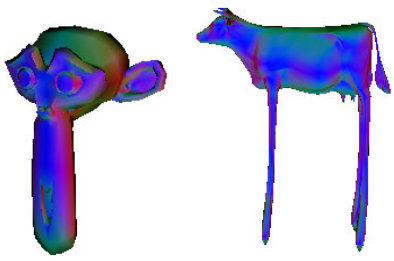


Sigui c el resultat d'interpolar linealment boundingBoxMin.y i boundingBoxMax.y, segons un paràmetre d'interpolació t, **uniform float t = 0.4**.

Si la coordenada Y és inferior a c, el VS li aplicarà l'escalat donat per **uniform float scale = 4.0** per tal d'allargar les potes del model. Altrament, no li aplicarà cap escalat, però sí una translació Δ en Y. Per calcular Δ, observeu que per tenir continuïtat a y=c, llavors **c*scale = c + Δ** (aïlleu Δ).

Degut a que no estem recalculant els plans de *clipping*, és possible que el model surti retallat.

El FS farà les tasques habituals.



Identificadors obligatoris:

dalify.vert, dalify.frag *(minúscules!)*
uniform float t = 4.0;
uniform float scale = 4.0;