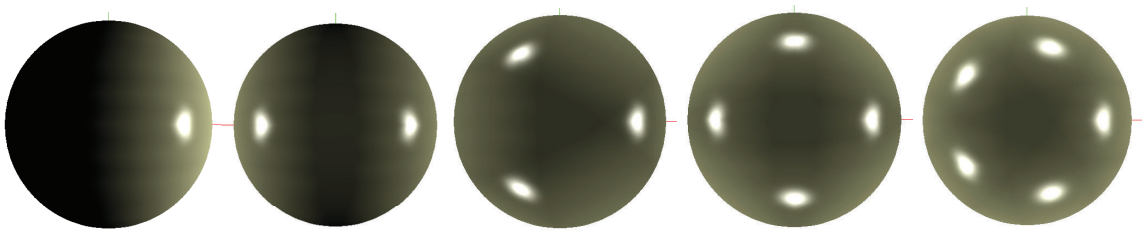
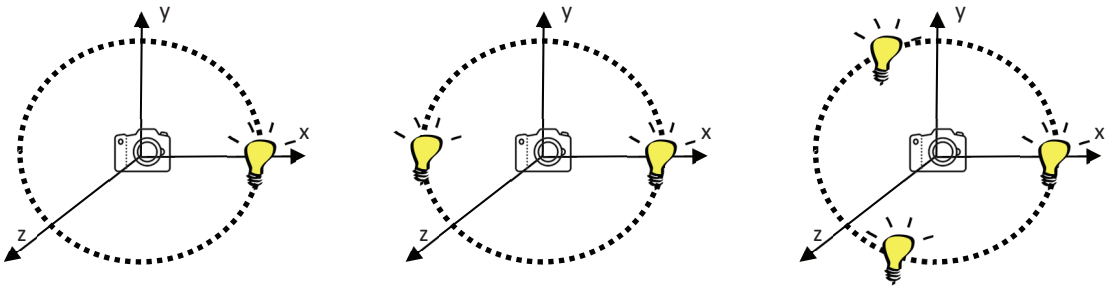
Nlights (nlights.*)

Escriu **VS+FS** per aplicar il·luminació de Phong **per fragment**, amb n llums fixos respecte la càmera, on n és un **uniform int n=4**. Aquí tens l'esfera amb n=1-5 llums:



Els llums estaran situats al voltant d'un cercle de **radi 10** situat al **pla Z=0 de la càmera** i centrat a la càmera. El primer llum estarà situat al punt de coordenades eye space (10, 0, 0), i la resta estaran equidistribuïts seguint el cercle, com es mostra a la figura per n=1-3 llums:



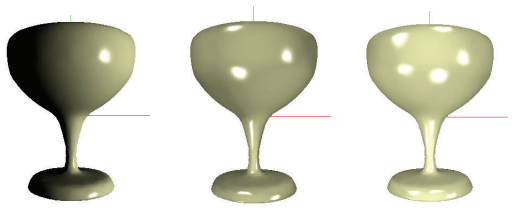
El VS farà les tasques habituals i passarà al FS les dades necessàries (vèrtex i normal) pel càlcul d'il·luminació.

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

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

la qual **ignora la contribució ambient** i **divideix la contribució difosa per $\sqrt{n}$.**

Aquí teniu la copa amb n=1,3,5 llums:



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

Identificadors obligatoris:

```
uniform int n = 4;
const float pi = 3.141592;
```

Digits (digits.*)

Escriu **VS+FS** per **texturar** l'**objecte plane** de forma que es mostri la part entera del uniform float time:

Usarem la textura **digits.png**:

El VS, a banda de les tasques habituals, multiplicarà per 3 la coordenada de textura **s** que li passa al FS. D'aquesta manera, els valors de **s** arribaran al FS dins l'interval [0,3].

El FS **accedirà a la textura dígits** amb coordenades de textura adients (similar a com vas fer a l'exercici explosion). Pots assumir que time<=999, de forma que la seva part entera es pot representar amb només 3 dígits. Podeu extreure els dígits de int(time) **usant els operadors / i %** (divisió entera i mòdul de la divisió entera).

Observa que l'offset a aplicar a la coordenada de textura *s* depèn del valor que es rep del VS:

- 0 <= s < 1 → cal mostrar dígit de les centenes;
- 1 <= s < 2 → cal mostrar dígit de les desenes;
- 2 <= s <= 3 → cal mostrar dígit de les unitats.

Observa també que l'offset serà de la forma d/10.0, on d és el dígit a mostrar.

La coordenada de textura *t* no requereix cap modificació.

Un cop tenim el color C de la textura, **descartarem** el fragment si C.a és < 0.5. Altrament, el color del fragment serà **vermell**.

Identificadors obligatoris:

uniform sampler2D colorMap;
uniform float time;

Cubify (cubify.*)

De forma a anàloga a com vas fer a spherize, en aquest exercici has de programar un VS que, abans de transformar cada vèrtex a coordenades de clipping, el projecti des de l'origen de coordenades del model, sobre un **cub centrat a l'origen d'aresta 2**, és a dir el cub amb vèrtexs a (±1, ±1, ±1).

Igual que en spherize, la projecció és al llarg de la recta que uneix el vèrtex amb l'origen (i serà el punt en què aquesta intersecta al cub). El VS copiarà el color de cada vèrtex al FS sense modificacions, i aquest es limitarà a fer servir el color rebut.

Per a fer-ho, pot ajudar-te observar que la coordenada de **major valor absolut del punt projectat serà 1**.

Alguns exemples de punts amb la seva projecció en el cub:

- (**10**, 0, 0) → (**1**,0,0)
- (0, **-5**, 0) → (0, **-1**, 0)
- (1, 2, **4**) → (0.25, 0.5, **1**)

Aquí tens el resultats esperats amb els models glass.obj, cesna.obj i cow.obj: