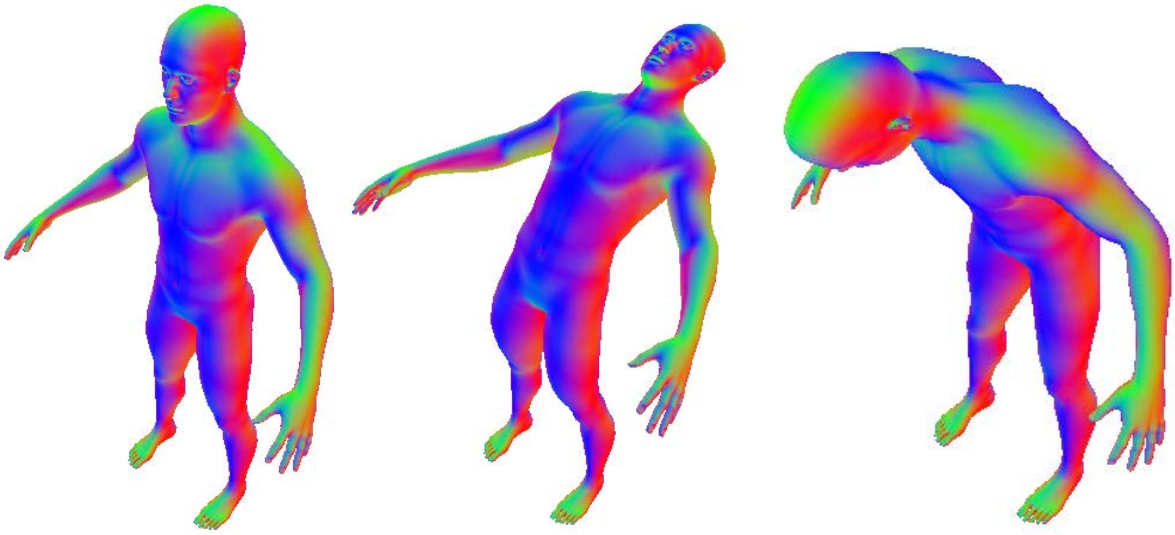
Contortion (contortion.*)

Escriu **VS+FS** que dibuixin el model humà (**man.obj**) amb una rotació que variï en el temps:



El **VS** haurà d'aplicar a cada vèrtex (en *object space*) una rotació respecte un eix paral·lel a l'eix X. L'angle de rotació *A* vindrà determinat per **(y-0.5)·sin(t)**, on y és la coordenada Y del vèrtex en *object space*. Només volem rotar els vèrtexs de la part superior del cos, per tant **l'angle haurà de ser 0 pels vèrtexs amb y per sota de 0.5** (l'alçada aproximada del genoll).

La rotació la volem fer respecte un eix paral·lel a l'eix X que passa pel punt **(0, 1, 0)**, que està a prop del centre del cos. Per tant, **abans i després** d'aplicar la rotació al vèrtex, haureu d'aplicar la translació corresponent. Recordeu que la matriu de rotació respecte l'eix X és:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos A & -\sin A \\ 0 & \sin A & \cos A \end{pmatrix}$$

El VS haurà d'escriure gl_Position com habitualment. El color calculat pel VS serà directament el color d'entrada, sense il·luminació.

El **FS** simplement escriurà el color que li arriba del VS.

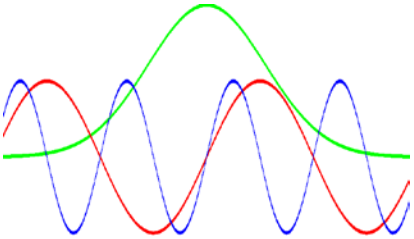
Recordeu no fer cap càlcul innecessari.

Fitxers i identificadors (ús obligatori):

`contortion.vert, contortion.frag`

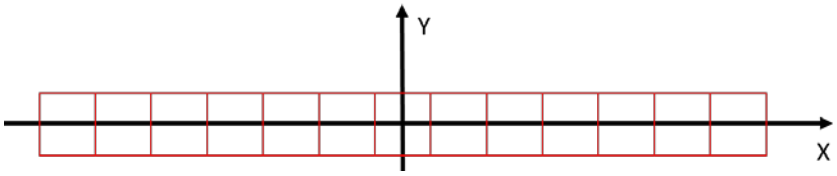
Plotter (plotter.*)

Escriu un **VS+FS** per dibuixar les gràfiques d'algunes funcions:



Les corbes R, G, i B són en realitat tires de triangles que el VS ha d'encarregar-se de deformar segons la funció (assumirem contínua) a dibuixar.

El model de partida serà **stripsx3.obj** el qual conté 3 bandes (una per cada funció a dibuixar) anàlogues a la figura, però amb primitives més petites:



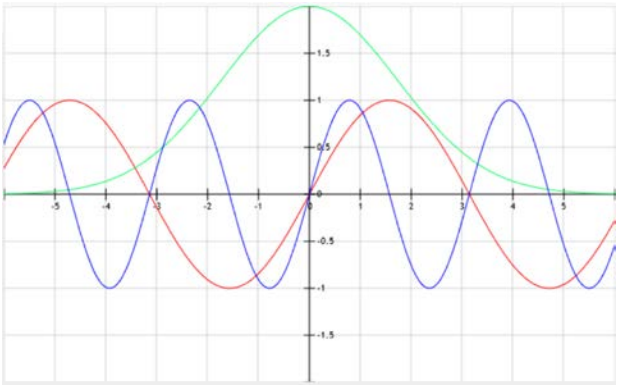
Les tres bandes estan sobre plans diferents (Z=-2, Z=0 i Z=2) per a que el VS les pugui distingir. La coordenada X varia en [-1, 1]. La coordenada Y està en un interval molt petit al voltant del 0 per donar gruix a cada banda.

Volem dibuixar les funcions **$\gamma=\sin(\varkappa)$, $\gamma=2\exp(-\varkappa^2/6)$, $\gamma=\sin(2\varkappa)$**. El domini estarà definit per:

uniform float xmin = -6;
uniform float xmax = 6;

I el recorregut visible de la funció per:

uniform float ymin = -2;
uniform float ymax = 2;



El VS haurà de fer les següents tasques:

1. Convertir linealment vertex.x (originàriament entre -1 i 1) a un valor $\varkappa$ entre xmin i xmax
2. Avaluar la funció **$\gamma=\sin(\varkappa)$, $\gamma=2\exp(-\varkappa^2/6)$, o $\gamma=\sin(2\varkappa)$,** depenent de si vertex.z és -2, 0 ó 2.
3. Convertir linealment γ (entre ymin i ymax) a un valor Y entre -1 i 1.
4. El vèrtex final en clip space serà **(vertex.x, vertex.y+Y, 0, 1)**, independentment de la càmera.
5. El color final serà R, G, o B depenent de si vertex.z és -2, 0 ó 2, resp.

El FS simplement ha de copiar el color que li arriba.

Fitxers i identificadors (ús obligatori):

`plotter.vert, plotter.frag; uniform float xmin, xmax, ymin, ymax;`

Skeletons (skeletons.*)

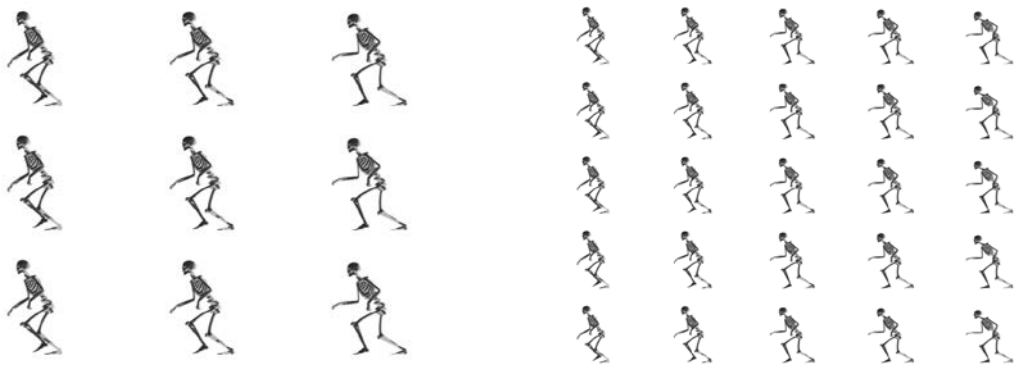
Escriu **VS+FS** per dibuixar l'objecte **plane.obj** amb una textura (**skeletons.png**) que conté els **44 fotogrames** de l'animació d'un esquelet, en una única fila. Aquí teniu la textura:

I aquí teniu un exemple del resultat esperat:

El VS haurà de fer aquestes tasques:

1. Modificar les coordenades de textura originals (texCoord) com si volguéssim repetir la textura *tiles* cops en horitzontal i vertical (**uniform int tiles=1**).
2. Calcular el frame de l'animació a mostrar, tenint en compte que l'animació s'ha de reproduir a **30 fps**, començant pel primer frame.
3. Calcular les coordenades de textura adients (de manera similar a l'exercici explosion), que tindran la forma **vec2(s/44 + offset, t)**, on (s,t) fa referència a les coordenades de textura ja multiplicades per tiles. Nota: si tiles = 1, la s estarà en [0,1] i només es mostrarà el frame seleccionat; altrament, s estarà en [0, tiles] i per tant cada columna d'esquelets començarà en un frame diferent, que és el que volem.
4. Escriure a gl_Position les coordenades originals del vèrtex, sense cap transformació, per tal que ocupi tot el viewport (no useu les matrius de la càmera).

El FS calcularà el color simplement accedint a textura a les coordenades que rep del VS. Per obtenir el color final del fragment, el FS haurà **d'invertir les components RGB del color** de la textura (per exemple, 0 passarà a ser 1 i viceversa). Aquí teniu més exemples, amb tiles = 3 i tiles = 5:



Identificadors (ús obligatori):

```
skeletons.vert, skeletons.frag, uniform int tiles = 1;
```