OOP 2

On va faire des BlendShapes.

Les fichiers fournis contiennent :

- une simple classe `Vector3` servant à représenter un point dans l'espace, avec des propriétés `x`, `y` et `z` ainsi qu'une méthode `str`
- un main (le point d'entrée du programme)

Echauffement

- Exécuter le programme pour s'assurer qu'il dit bien bonjour
- Dans la fonction main, déclarer une variable `zero` représentant un point aux coordonnées (0,0,0)
- Déclarer une variable `ten` représentant un point aux coordonnées (10,10,10)

BlendShape

1 - Class BlendShape

- Déclarer une class BlendShape
- Ajouter un constructeur qui prend trois paramètres:
  - `name`, son nom
  - `origin`, un point de base sans déformation
  - `target`, un point après déformation maximale
- Ajouter trois propriétés, `name`, `origin` et `target` renvoyant respectivement le nom, le point sans déformation et le point après déformation maximale

2 - Interpolation linéaire

a) Clamp01

Dans la class BlendShape :

- Déclarer une méthode `clamp01`, prenant un paramètre `value`
- Implémenter la méthode `clamp01` pour faire en sorte qu'elle renvoie `value` clampé entre 0 et 1 (c'est-à-dire que ce qui est renvoyé doit toujours être entre 0 et 1, même si `value` est négatif ou plus grand que 1)

> Astuce: Python possède une fonction min(a, b) renvoyant la plus petite valeur entre a et b.
> min(1, 2.5) ou min(1, 4.75) renverront toujours 1.
> min(1, 0.5) renverra 0.5.
> min(1, -2.5) renverra -2.5.
> Python dispose également de la fonction max(a, b), renvoyant la plus grande des deux valeurs.
> max(0, b) renverra toujours un nombre positif.

b) LerpValue

Nous avons désormais une méthode `clamp01` sous la main, qu'on peut appeler avec `self.clamp01(value)`. Je dis ça au hasard.

Toujours dans la class BlendShape :

- Déclarer une méthode `lerpValue` prenant trois paramètres, `a`, `b` et `factor`
- Implémenter `lerpValue` :
  - Clamper factor entre 0 et 1
  - Calculer l'interpolation linéaire entre `a` et `b` par `factor` (voir formule plus bas)
  - Renvoyer la valeur calculée

> L'interpolation linéaire entre deux nombres A et B se calcule comme suit :
> `value = (F * B) + ((1-F) * A)`
> où F est compris entre 0 et 1

c) Lerp

Nous avons désormais `lerpValue` qui permet de calculer l'interpolation linéaire entre deux nombres, ce qui peut être fort utile. Peut-être. On sait pas.

- Déclarer une méthode `lerp` prenant un paramètre `factor`
- Implémenter `lerp` :
  - Calculer une valeur `lerp_x` étant l'interpolation linéaire entre les propriétés `x` de origin et de target par `factor`
  - Faire de même pour calculer `lerp_y` et `lerp_z` sur les composantes `y` et `z` respectivement
  - Construire un nouveau Vector3 avec les valeurs interpolées
  - Renvoyer ce Vector3

3 - Tests

Dans la fonction main :

- Déclarer une variable blend_shape appelée "Super BlendShape" avec pour origine `zero` et pour cible `ten`
- interpoler `blend_shape` avec une valeur de 0.5 et vérifier qu'on obtient un point en (5,5,5)
- interpoler `blend_shape` avec une valeur de 0 et vérifier qu'on obtient un point en (0,0,0)
- interpoler `blend_shape` avec une valeur de 1 et vérifier qu'on obtient un point en (10,10,10)
- interpoler `blend_shape` avec une valeur de 1200 et vérifier qu'on obtient un point en (10,10,10)
- Bravo