Moteur -> Ajouter un objet (x,y, weight, height, tex)
	Modifier l'emplacement de l'objet et la texture associée

class Engine:
	Array<ObjectGl> objects

	add_objectGl(objects)
	remove_objectGL(objects)
	render()

class ObjectGl:
	VertexArray vertexArray
	Shader shader
	Matrix4f transform

	translate(vec3)
	scale(vec3)
	rotateX(float)
	rotateY(float)
	rotateZ(float)
	textureSwap()
	textureSwapWrap(x, y) //coord en pixels

	render()

class ObjectGlColor(ObjectGl)
	float[] color
	swapColor(vec4)

class ObjectGlTex(ObjectGl):
	Array<Texture> texture
	Array<float[]> texture_wrap

class ObjectGlTexColor(ObjectGlTex):
	float[] color
	swapColor(vec4)

TODO:
	Ajouter une caméra orthographique
	Implémenter les classes
	Faire une scène de test/ cycle sur les animations d'un atlas
	Trouver un moyen d'afficher à la suite des textures de taille différente sans stretch
	Faire un système de conversion des coordonnées écran vers coordonnées openGL//LEO
	Faire le système d'input // LEO
	Limiter les rendus à 60 fps
	Primitive changer pour faire des formes quelconques (x, y, h, w)

Voir pour le son
Voir comment ajouter du texte sur la fenêtre

Interface:

```
ObjectGLTexColor player1 = new ObjectGlTex(coord, tex, color)
ObjectGLTexColor player2 = new ObjectGlTex(coord, tex, color)
ObjectGLColor hp = new ObjectGlColor(coord, color)
//Le jeu commence
//Un personnage se déplace
player1.translate(vec3deplacement)
// le personnage stop son déplacement
player1.translate(vec3deplacement) // le vec est à 0
// un personnage qui attaque
player1.textureSwapWrap(coord) //coord de la frame à afficher dans l'atlas
// l'attaque du personnage touche le second clignote en blanc
player2.swapColor(1.0f,1.0f,1.0f,0.8f)
// il perd des points de vie
hp.scale(0.6f, 1.0f, 1.0f)
```