Documentation

SciVisu
Interface de visualisation de données scientifique en WebGL

BLAGNAC
iut Midi-Pyrénées
UNIVERSITÉ TOULOUSE Jean Jaurès

Table des matières

1Template Geometry....2
1.1Création d'objets....3
1.2Changement de couleur....6
1.3Changement de position....7
1.4Changement de taille....8
1.5Changement multiple....9
1.6Sélection d'un objet....10
1.7Suppression d'un objet....12
1.8Changement de la scène....13
2Template JSON....14
3Template OBJ....16

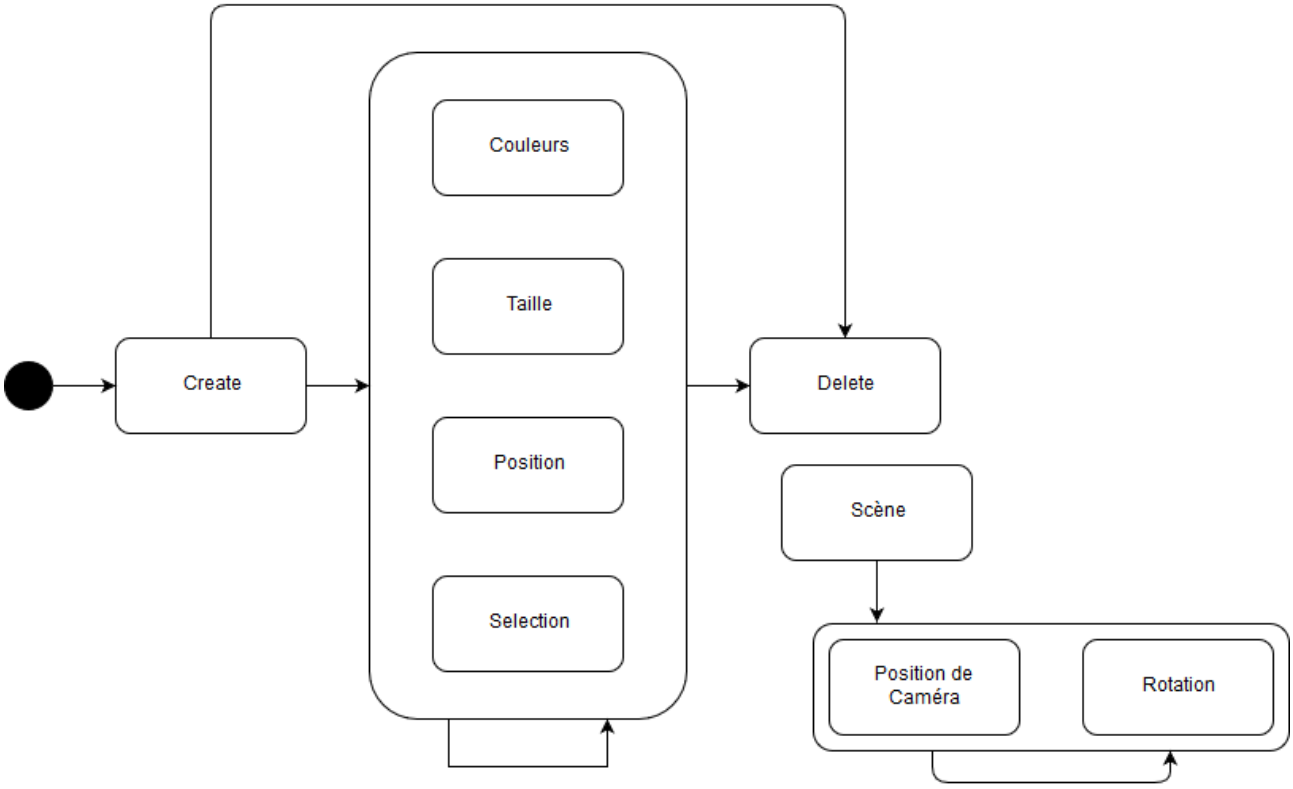
1 Template Geometry

Avec le template Geometry, nous pouvons créer des formes en envoyant des messages MQTT sur **archibald.snootlab.info:2532**

Une authentification est nécessaire (« ptut »/ « ptut »).

Les sources sont disponibles sur le GitHub du projet **,** dans le dossier « **templateGeometry** ».

Plusieurs actions sont alors possibles.



1.1 Création d'objets

Pour créer un objet, il faut utiliser le topic :

templateGeo/obj/create/

On envoie sur ce topic un message en format JSON, respectant la structure suivante :

On peut donc initialiser plusieurs éléments à la création d'un objet : la couleur, la position et la taille de l'objet. On peut également choisir de sélectionner l'objet en mettant le champ « select » à la valeur « 1 ».

Ces paramètres peuvent également être modifié ultérieurement.

Si l'utilisateur ne souhaite pas saisir les paramètres de couleur, forme ou taille, le champ doit être remplis avec « null » et « 0 » pour le champ select.

Le champ « shape » peut prendre les valeurs :

- Cube
- Sphere
- Plan
- Point
- Line

Création d'un objet		
Champs	Assigné	Non assigné
shape	Liste ci-contre	
color	#xxxxxx	null
position	x,y,z	null
scale	x	null
select	1	0

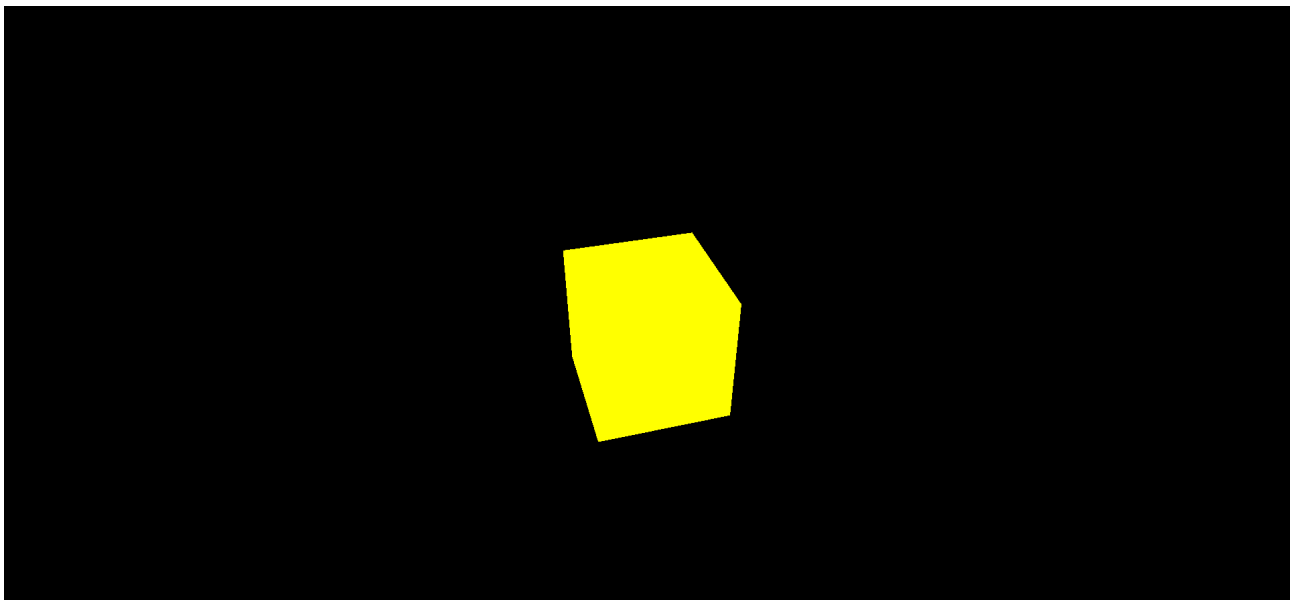
Pour l'exemple présenté, on obtient ce résultat.

Après la création d'un objet, un message est envoyé sur le topic **templateGeo/retourID/** qui permet de récupérer l'ID de l'objet, pour pouvoir le modifier avec les autres actions possibles.

Attention : si l'ID utilisé pour les modifications n'existe pas alors vous perdrez la connexion avec le site web.

La numérotation des IDs ne commence automatique à 0. Votre scène est composé de caméra et d'autres composants dont l'identifiant sera attribué avant celui de votre objet.

A la création de l'objet, le serveur va s'abonner aux topics correspondant à toutes les actions possibles pour l'objet créé.



Objet créé sans caractéristiques de bases

1.2 Changement de couleur

Pour changer la couleur d'un objet, il faut utiliser le topic :

templateGeo/obj/color/{id de l'objet}/

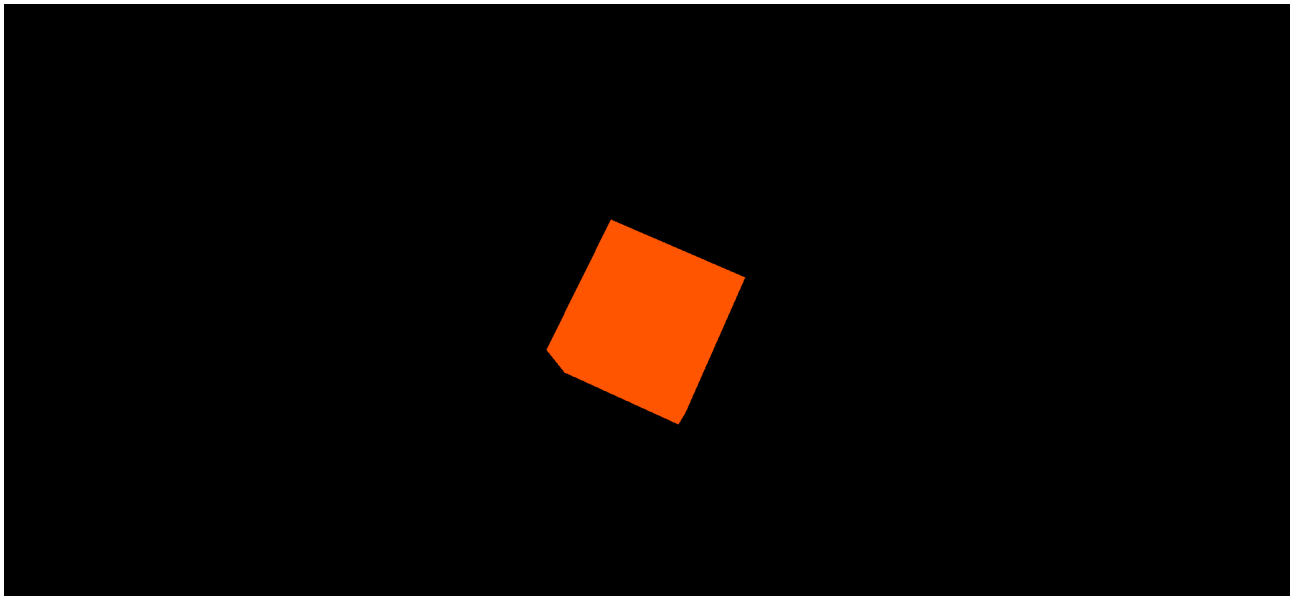
On envoie sur ce topic un message en format JSON, respectant la structure suivante :

```
{
  "color":"#FF5500"
}
```

On doit donc utiliser l'ID de l'objet retourné par la méthode de création.

La couleur doit être exprimé en Hexadécimal, précédé de **#** ou **0x** .

Pour le cube créé sans argument, cela donne :



1.3 Changement de position

Pour changer la position d'un objet, il faut utiliser le topic :

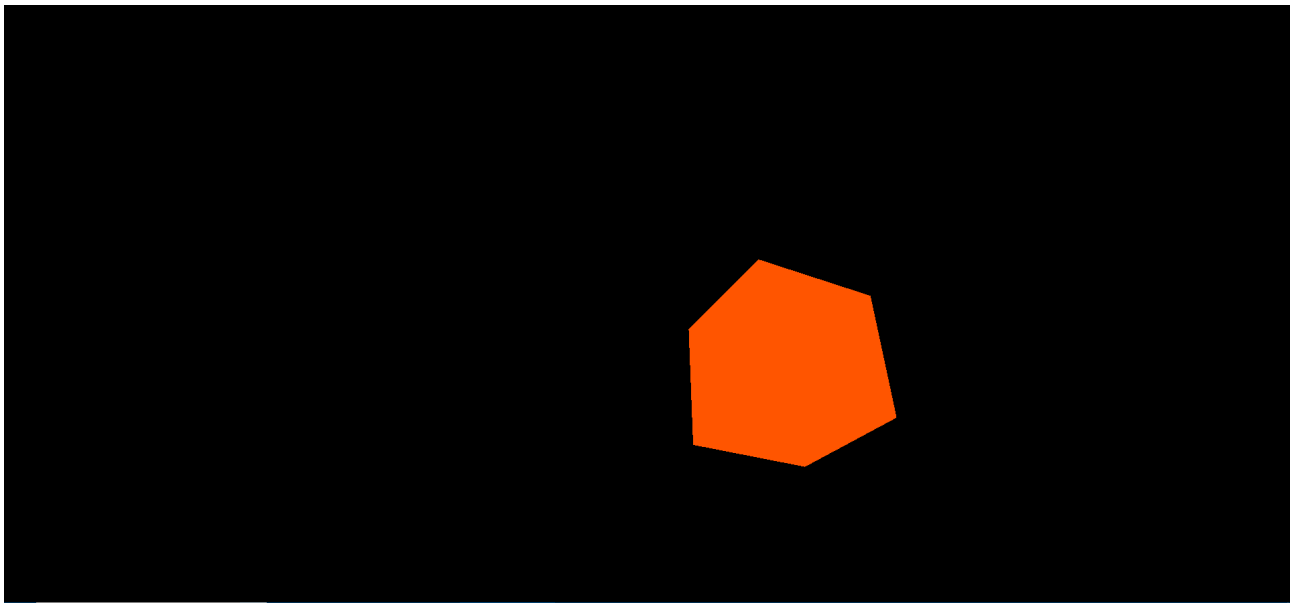
templateGeo/obj/position/{id de l'objet}/

On envoie sur ce topic un message en format JSON, respectant la structure suivante :

```
{
  "position_x":"200",
  "position_y":"200",
  "position_z":"200"
}
```

On doit utiliser l'ID de l'objet retourné par la méthode de création.

Pour le cube créé sans argument, cela donne :



1.4 Changement de taille

Pour changer la taille d'un objet, il faut utiliser le topic :

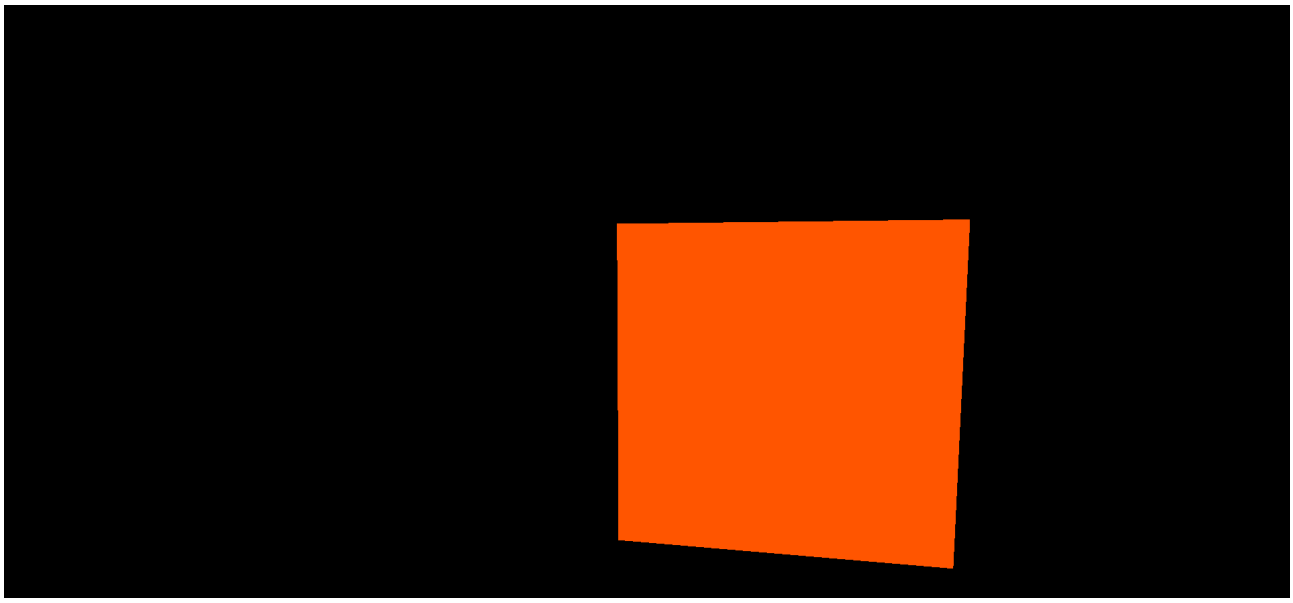
templateGeo/obj/scale/{id de l'objet}/

On envoie sur ce topic un message en format JSON, respectant la structure suivante :

```
{
  "scale":"2"
}
```

On doit donc utiliser l'ID de l'objet retourné par la méthode de création. Dans cet exemple, le scale va doubler la taille de notre objet.

Pour le cube créé sans argument, cela donne :



1.5 Changement multiple

Pour changer plusieurs caractéristiques d'un objet, il faut utiliser le topic :

templateGeo/obj/json/{id de l'objet}/

On envoie sur ce topic un message en format JSON, respectant la structure suivante :

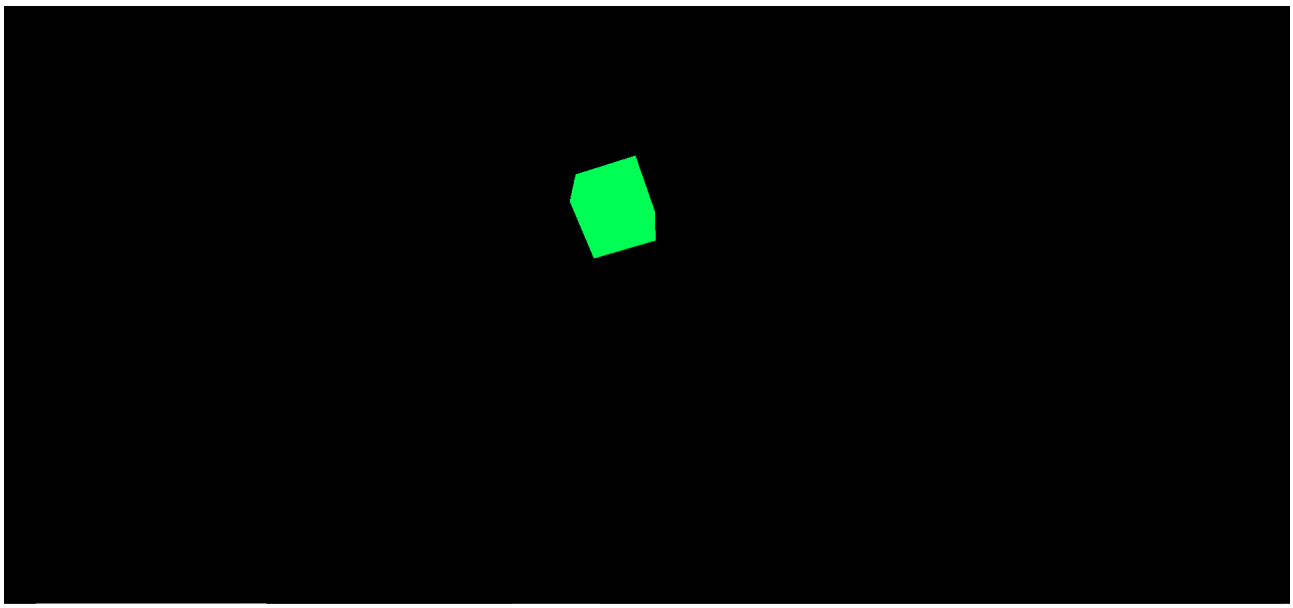
On doit donc utiliser l'ID de l'objet retourné par la méthode de création.

Les caractéristiques pouvant être changés sont les mêmes que celles à la création d'un objet : couleur, position, taille et la sélection ou non d'un objet.

On rappel que si l'utilisateur ne souhaite pas saisir les paramètres de couleur, forme ou taille, le champ doit être remplis avec « null » et « 0 » pour le champ select.

Champs	Assigné	Non assigné
color	#xxxxxx	null
position	x,y,z	null
scale	x	null
select	1	0

Pour le cube créé sans argument, cela donne :



1.6 Sélection d’un objet

Pour sélectionner un objet, il faut utiliser le topic :

templateGeo/obj/select/{id de l’objet}/

Aucun message n’est nécessaire, le simple envoie d’un message vide sur le topic permet de sélectionner l’objet.

La sélection d’un objet peut aussi se faire avec un clic souris sur l’objet.

La sélection d’un objet permet d’avoir accès a ce panneau de modification.

Avec ce panneau de modification, on peut modifier manuellement plusieurs caractéristiques d’un objet tel que la taille et la position, grâce a des barres de contrôle.

1.7 Suppression d’un objet

Pour supprimer un objet, il faut utiliser le topic :

templateGeo/obj/delete/{id de l’objet}/

Aucun message n’est nécessaire, le simple envoie d’un message vide sur le topic permet de sélectionner l’objet.

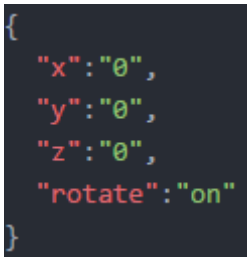
La suppression d’un objet permet de le retirer de la scène. Les IDs des autres objets ne changent pas.

1.8 Changement de la scène

Pour changer la scène, il faut utiliser le topic :

templateGeo/scene/

On envoie sur ce topic un message en format JSON, respectant la structure suivante :



Les positions pouvant être changées correspondent à la place de la caméra sur la scène. Elle est de base aux positions (0,0,1000).

On mettra une valeur « null » pour ne pas modifier la valeur actuelle.

Un champ « rotate » est également disponible pour bloquer la rotation des objets sur la scène.

Changement de la scène		
Champs	Assigné	Non assigné
x	x	null
y	y	null
z	z	null
Champs	Activé	Désactivé
rotate	on	off

2 Template JSON

Avec le template JSON, nous pouvons visualiser en 3D un objet contenu dans un fichier json (extension .json ou .js) sur la scène.

Les sources sont disponibles sur le GitHub du projet , dans le dossier « **templateObject** »

On utilisera le code Python ci-dessous pour envoyer les fichiers json (disponible dans la branche « **master** » du GitHub du Projet).

```
import paho.mqtt.client as mqtt
import json, sys
import time

arg = sys.argv[1]

MQTT_PORT = 2532
MQTT_TOPIC = "JSONtemplate"
MQTT_IP = "91.224.148.106"

file = open(arg,"r")
client = mqtt.Client("Kilian")
client.connect(MQTT_IP,MQTT_PORT)
tab=file.readlines()
print(tab)
client.publish(MQTT_TOPIC,len(tab))
for i in range(len(tab)):
    time.sleep(0.1)
    client.publish(MQTT_TOPIC,tab[i])
file.close()
```

Le code va prendre le fichier json en argument et le transmettre sur le broker.

Exemple :

```
desportes@desportes-VirtualBox:~/Bureau/mqtt_webgl_template/PythonMQTT$ python P
ublishJSON.py parrot.js
```

Une fois le compteur au niveau de la taille du fichier (écrite au départ dans la console Web du navigateur), on obtient le rendu 3D de l'objet JSON envoyé à la base, ci-dessous, un perroquet.

3 Template OBJ

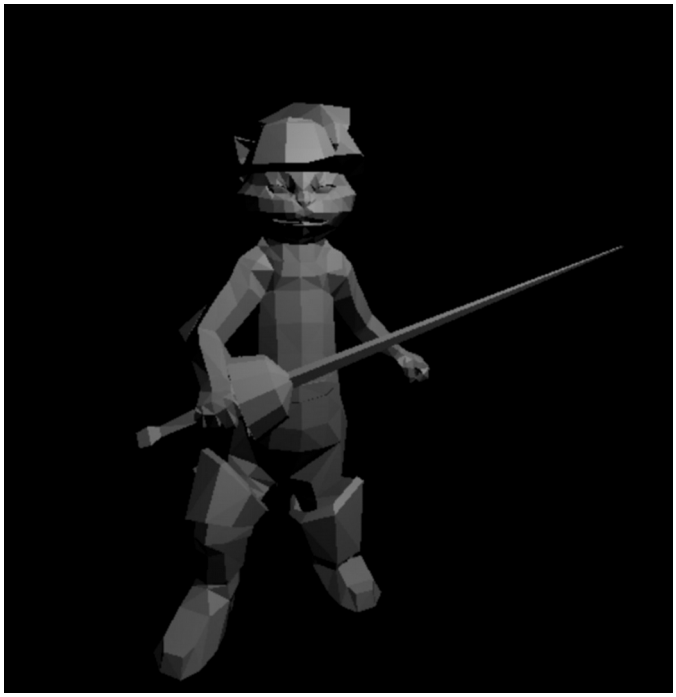
Avec le template OBJ, nous pouvons visualiser en 3D un objet contenu dans un fichier OBJ (extension .json ou .js) sur la scène.

Les sources sont disponibles sur le GitHub du projet , dans le dossier « **templateObject** »

Le MQTT n'est pas utilisable sur ce template, pour pouvoir visualiser l'objet, il faut le changer dans le code , à l'endroit indiqué ci-dessous :

```
objLoader.load("Objets/OBJ/Puss_in_Boots.obj", function(mesh){
```

Ici, l'objet « Puss_in_Boots.obj » , ce qui donne :



On peut également changer la texture de l'objet, sous forme d'un fichier .mtl, ici nous utilisons « Tent_Poles_01.mtl »

```
mtlLoader.load("Objets/MTL/Tent_Poles_01.mtl", function(materials){
```

Cependant, les fichiers MTL contenant des liens avec des fichiers photos .jpg ou .png ne sont pas pris en charge. Le fichier MTL ici donne juste un aspect gris à l'objet .OBJ.