

Documentation

SciVisu

Table des matières

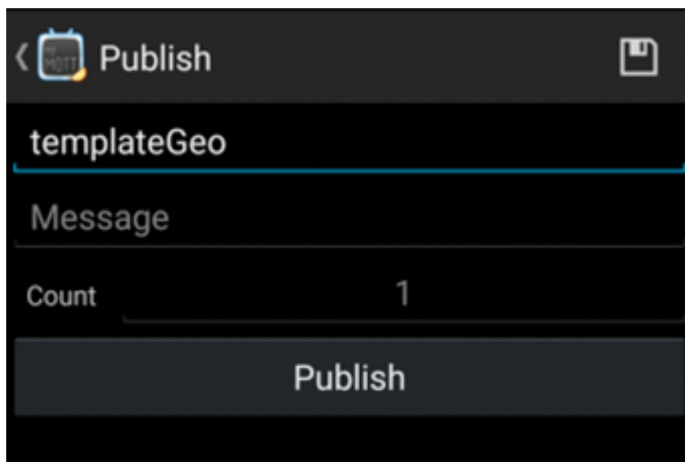
1 Template Geometry.....	3
1.1 Création d'une sphère :.....	4
1.2 Changement de taille d'un objet.....	7
1.3 Changement de couleur d'un Objet.....	9
1.4 Supprimer un Objet.....	11
1.5 Selectionner un Objet.....	13
2 Template Objet – JSON :.....	20
3 Template Objet – OBJ :.....	23

1 Template Geometry

Avec le template Geometry, nous pouvons créer des formes en envoyant des messages MQTT.

Nous utiliserons ici l'application mobile 'MyMQTT™' pour nos exemples.

La connexion étant déjà établie, il ne nous reste qu'à publier sur le broker, et envoyer le message que nous souhaitons.

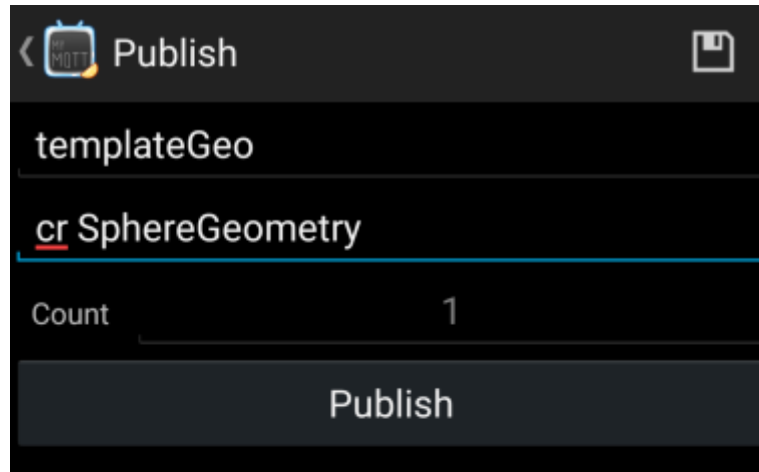


Les commandes sont disponibles sont :

- Creation objet : commande → cr [BoxGeometry|SphereGeometry|PlaneGeometry|PointGeometry]
- Entrer position : commande → pos id x.y.z
- Entrer l'échelle : commande → scl id scale
- Changer couleur : commande → col id hexa
- Sélectionner un objet : commande → select id
- Détruire un objet : commande → delete id

1.1 Création d'une sphère :

Sur l'application :



Sur la console :

```
Connecte  
Message arrive: cr SphereGeometry  
Topic:      templateGeo  
1
```

Ce qui donne :



Le code source :

```
function AjoutObjectGeometry(valEnvoi){
  geometry = TrouverGeometrieCorres(valEnvoi);
  if(valEnvoi=='PointGeometry'){
    geometry.vertices.push(new THREE.Vector3( 0, 0, 0));
    var material = new THREE.PointsMaterial({color : 0xffff00,size: 100});
    mesh = new THREE.Points( geometry, material );
    sac3DObject.add( mesh );
  }else{
    var material = new THREE.MeshBasicMaterial({color : 0xffff00, wireframe: false});
    mesh=new THREE.Mesh(geometry, material);
    sac3DObject.add(mesh);
  }
  console.log(sac3DObject.children.length);
}
```

qui utilise

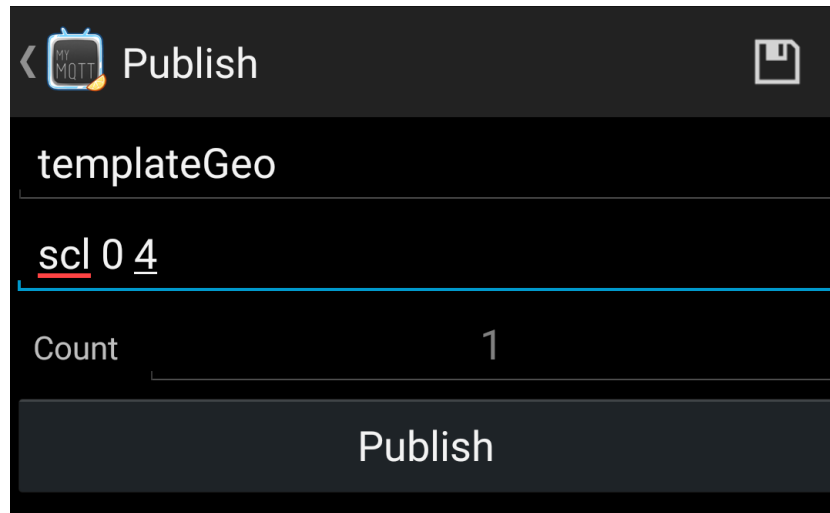
```
function TrouverGeometrieCorres(vEnvoi){
  var tabGeometries = [
    { type: 'BoxGeometry', geometry: new THREE.BoxGeometry( 200, 200, 200, 2, 2, 2 )},
    { type: 'SphereGeometry', geometry: new THREE.SphereGeometry( 100, 12, 12 ) },
    { type: 'PlaneGeometry', geometry: new THREE.PlaneGeometry( 200, 200, 200 ) },
    { type: 'PointGeometry', geometry: new THREE.Geometry()}
  ];
  var vgeometry = tabGeometries[0].geometry;
  for(var i=0; i<tabGeometries.length;i++){
    if(tabGeometries[i].type==vEnvoi){
      vgeometry=tabGeometries[i].geometry;
    }
  }
  return vgeometry;
}
```

La fonction détecte ce qui a été envoyé par le MQTT et crée l'objet correspondant en conséquence.

On peut également créer des cubes et des plans avec BoxGeometry et PlaneGeometry.

1.2 Changement de taille d'un objet

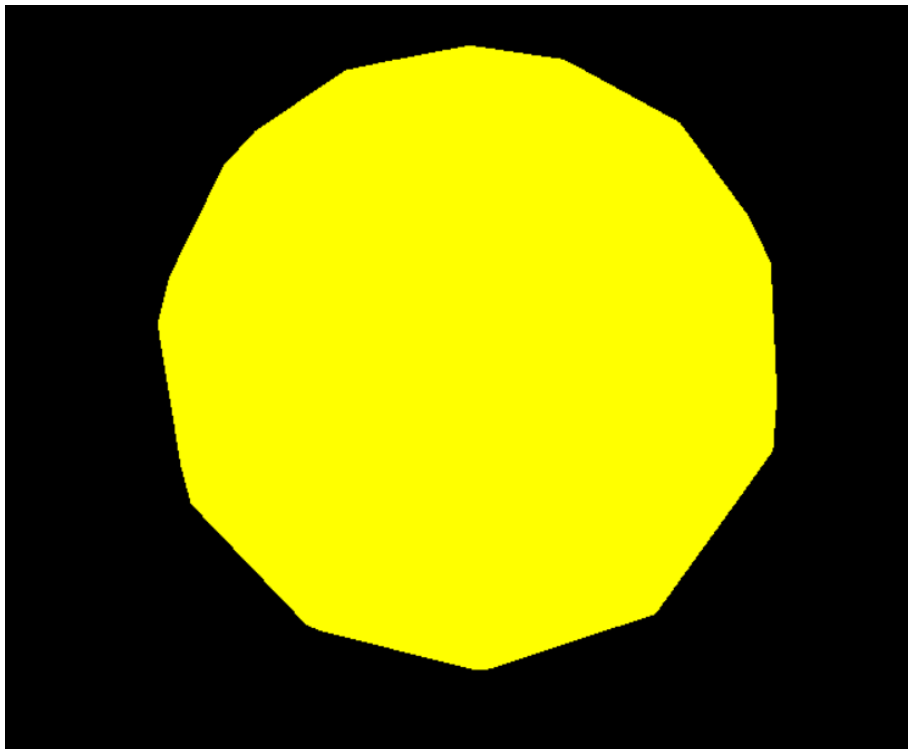
Sur l'application :



Sur la console :

```
Message arrive: scl 0 4  
Topic:      templateGeo
```

Ce qui donne :



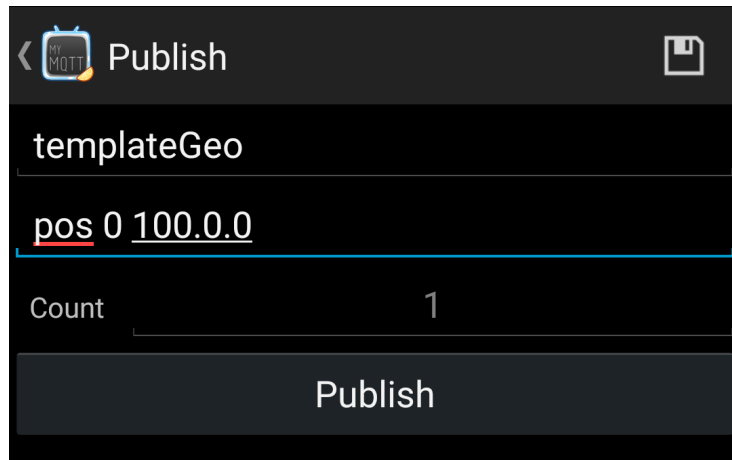
Le code source :

```
function setScale(id2,val){  
    var object = sac3DObject.getObjectById(sac3DObject.children[id2].id);  
    object.scale.set( val, val, val );  
}
```

On va redéfinir la taille de l'objet en fonction du nombre passé par le MQTT.

1.3 Changement de couleur d'un Objet

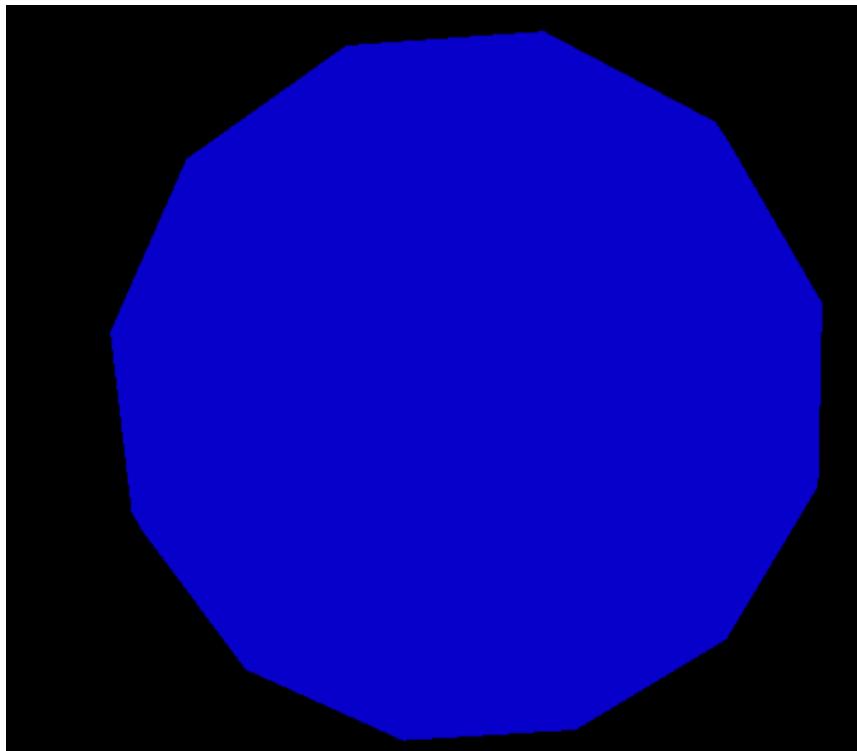
Sur l'application :



Sur la console :

```
Message arrive: pos 0 100.0.0  
Topic:      templateGeo  
0
```

Ce qui donne :



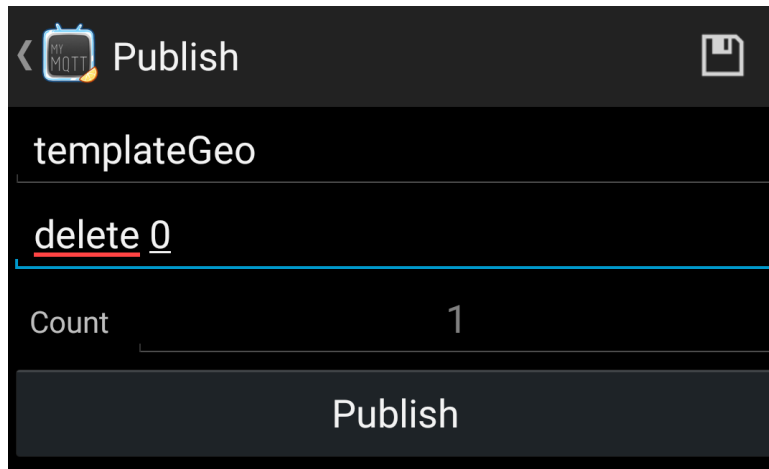
Le code source :

```
function setColor(id3,val){  
    var object = sac3DObject.getObjectById(sac3DObject.children[id3].id);  
    object.material.color.setHex( val.replace("#","0x") );  
}
```

Il s'agira ici, comme dans le changement de position, de changer la couleur de l'objet selon ce qui arrive depuis le MQTT (on remplace le '#' par '0x' pour que le webGL l'interprète correctement)

1.4 Supprimer un Objet

Sur l'application :



myMQTT Publish

templateGeo

delete 0

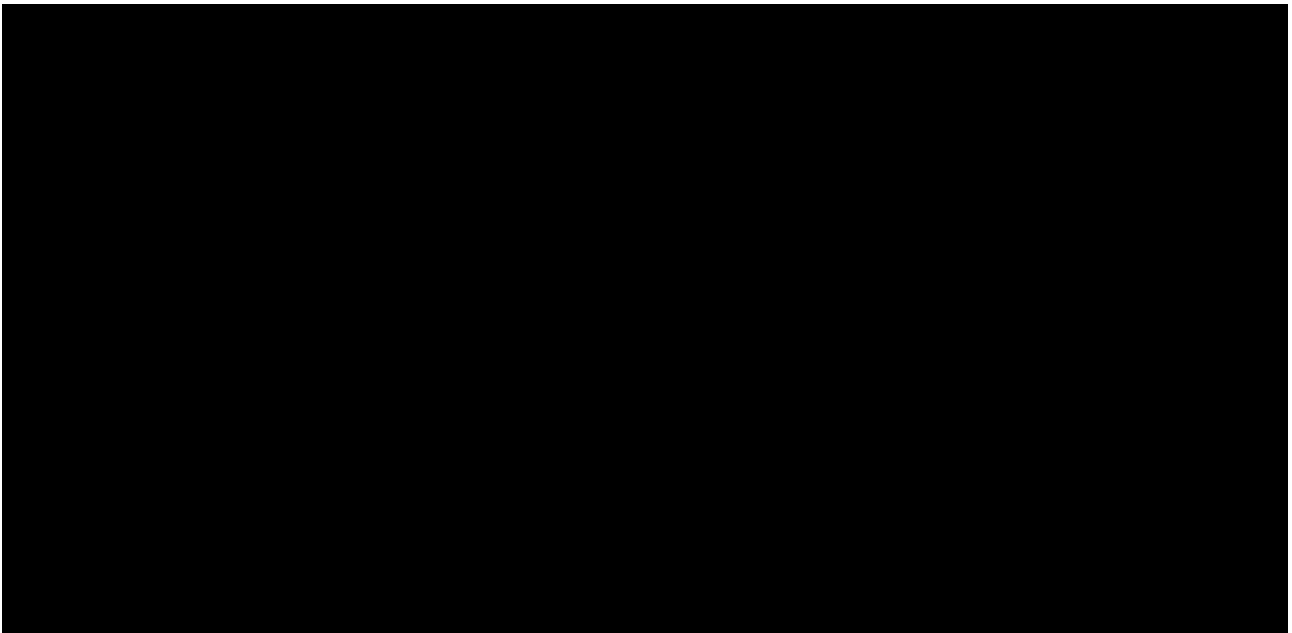
Count 1

Publish

Dans la console :

```
Message arrive: delete 0  
Topic:      templateGeo
```

Ce qui donne :



On a supprimé l'objet

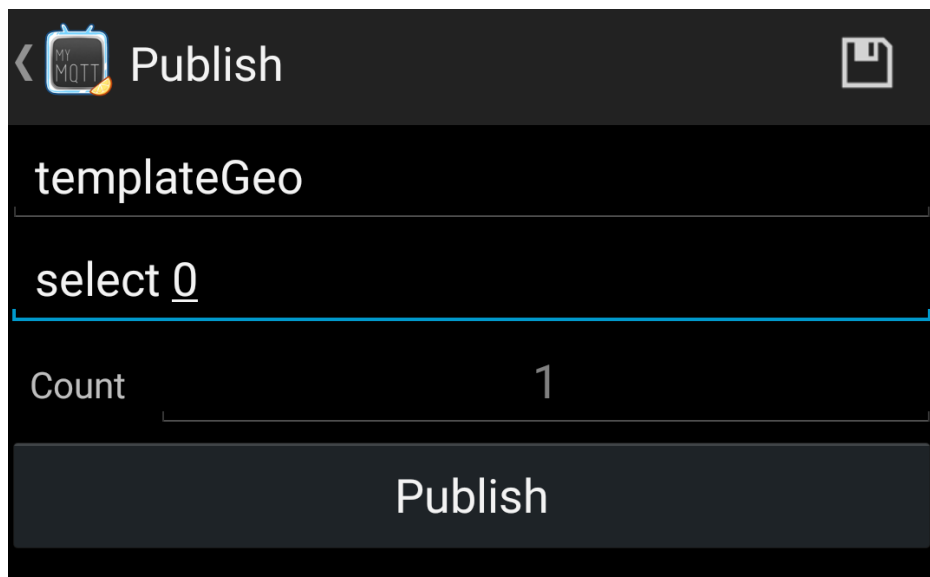
Le code source :

```
function deleteElement(id3){  
  var object = sac3DObject.getObjectById(sac3DObject.children[id3].id);  
  object.geometry.dispose();  
  sac3DObject.remove(object);  
  console.log(sac3DObject.children.length);  
  if((id3==0)&&(sac3DObject.children.length!=0)){menuUpdate(id3);}else if(id3>0){menuUpdate(id3-1);}else{menuP.close();}  
}
```

On passe l'ID de l'objet, qui sera supprimé ensuite.

1.5 Sélectionner un Objet

Sur l'application :



Sur la console :

```
Message arrive: select 0
Topic:      templateGeo
```

Ce qui donne :



On obtient le panel de gestion de l'objet manuel qui s'ouvre sur le template, une fois l'objet sélectionné.

Le code source :

```
function selectObject(id){
    var object = sac3DObject.getObjectById(sac3DObject.children[id].id);

    if(SELECTED !== object){
        if(SELECTED) SELECTED.material.color.set(0xff0000);
        SELECTED = object;
        idObjectSelection = id;
        options.ids=idObjectSelection;
        SELECTED.material.color.set(0xffffffff);
    }

    menuOnSelectGeometry(idObjectSelection);
}
```

Une fois le menu sélectionné, on utilise cette méthode (ci-dessous) pour modifier l'objet :

```
function menuOnSelectGeometry(idObjectSelection){
    if(menuP._controllers.length==0){
        id=menuP.add( options, 'ids').listen().onChange( function( value ) { options.ids=idObjectSelection} );
        sc =menuP.add( options, 'scale').min(0.1).max(4).listen().onChange( function( value ) { console.log(idObjectSelection);setScale(idObjectSelection,options.scale); } );
        x= menuP.add( options, 'x').min(-900).max(900).listen().onChange( function( value ) { setPosition(idObjectSelection,options.x,options.y,options.z); } );
        y=menuP.add( options, 'y').min(-900).max(900).listen().onChange( function( value ) { setPosition(idObjectSelection,options.x,options.y,options.z); } );
        z=menuP.add( options, 'z').min(-900).max(900).listen().onChange( function( value ) { setPosition(idObjectSelection,options.x,options.y,options.z); } );
        col = menuP.addColor( options, 'color').name('color').listen().onChange( function( value ) { setColor(idObjectSelection,options.color); } );
        rst = menuP.add(options,'reset').name("Reset Element").listen().onChange( function( value ) { resetElement(idObjectSelection); } );
        detruire=menuP.add(options,'detruire').name("detruire").listen().onChange( function( value ) { deleteElement(idObjectSelection); } );
        menuP.open();
    }else{
        menuUpdate(idObjectSelection);
    }
}
```

On peut également sélectionner un objet avec la souris, en utilisant cette méthode (ci-dessous) :

```
function onMouseDown( event ){
    mouse.x = ( event.clientX / window.innerWidth ) * 2 - 1;
    mouse.y = - ( event.clientY / window.innerHeight ) * 2 + 1;
    raycaster.setFromCamera(mouse, camera)//position du raycaster avec la camera et la position de la souris
    var intersects = raycaster.intersectObjects( sac3DObject.children );
    if( intersects.length>0){
        var intersect = intersects[ 0 ];
        var id = intersect.object.id;
        var idSac=0;
        for(var i=0;i<sac3DObject.children.length;i++){
            if(sac3DObject.children[i].id==id){
                idSac=i;
            }
        }
        selectObject(idSac);
    }
}
```

La partie MQTT du template :

```
client.onMessageArrived = function (message) {
    console.log("Message arrive: " + message.payloadString);
    console.log("Topic:      " + message.destinationName);
    var msg=message.payloadString;

    var tab=msg.split(" ");

    if(tab[0]=="cr"){
        var vgeo = tab[1];
        AjoutObjectGeometry(vgeo);
        cpt = cpt + 1;
    }else if(tab[0]=="pos"){
        var vdonnee = tab[1].split(" ");
        var vid = vdonnee[0];
        console.log(vdonnee[0]);
        var vpos = tab[2].split(".");
        var vx = vpos[0];
        var vy = vpos[1];
        var vz = vpos[2];
        setPosition(vid,vx,vy,vz);
    }else if(tab[0]=="select" ){
        var vid = tab[1];
        selectObject(vid,lastSelect);

    }else if(tab[0]=="delete"){
        var vid = tab[1];
        deleteElement(vid);
    }else if(tab[0]=="scl"){
        var vid = tab[1];
        var vscale = tab[2];
        setScale(vid,vscale);
    }else if(tab[0]=="col"){
        var id = tab[1];
        var color = tab[2];
        setColor(id,color);
    }
}
```

Cette partie va permettre de, selon le message qui arriver, 'gérer' la message pour lancer la fonction correspondante.

2 Template Objet – JSON :

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

On utilisera le code Python ci-dessous pour envoyer les fichiers json.

```
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 PublishJSON.py parrot.js
```

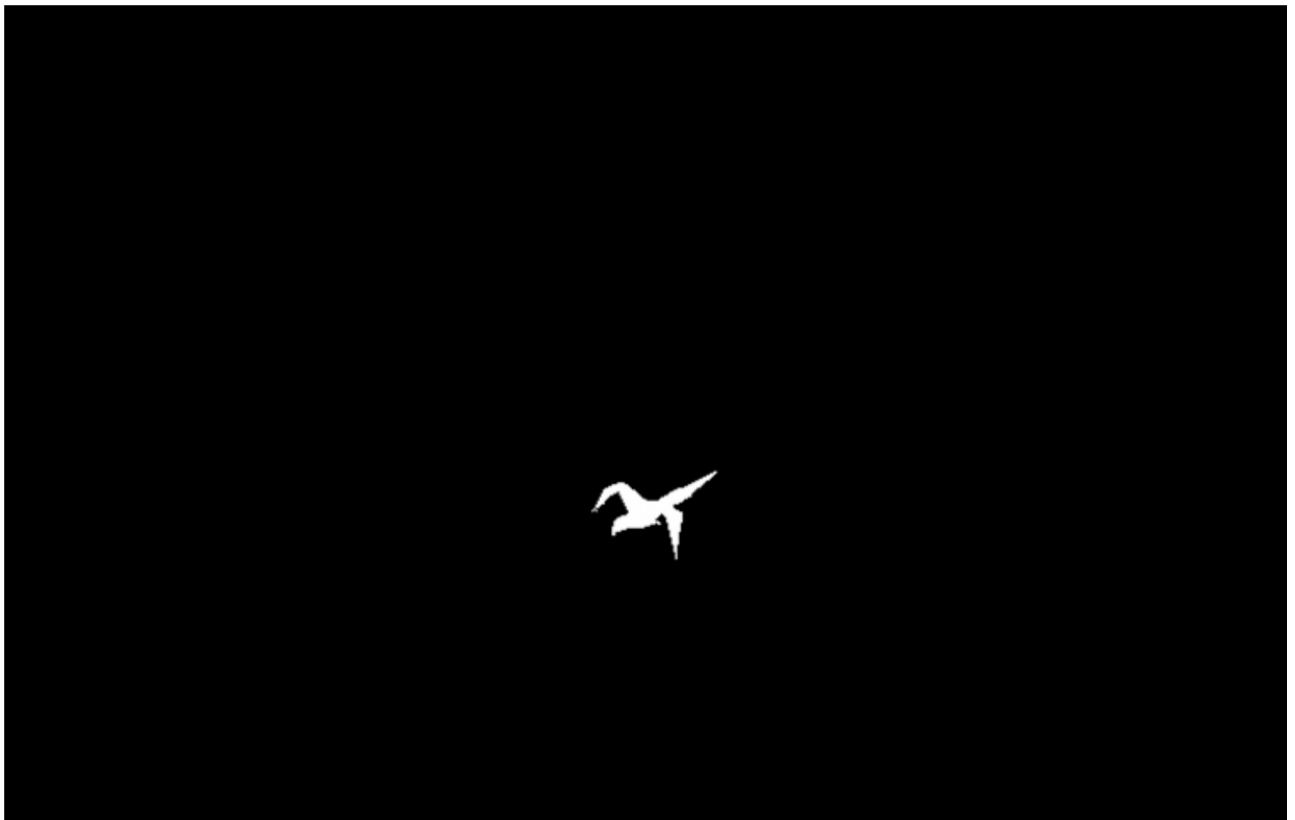
Ce qui donne sur le template, dans la console web :

```
THREE.WebGLRenderer 89
Connecte
taille : 45
compteur 1
compteur 2
```

compteur 45

```
⚠ THREE.JSONLoader: "morphColors" no longer supported. Using them as face colors.  
⚠ THREE.DirectGeometry.fromGeometry(): Undefined vertexUv 624  
⚠ THREE.DirectGeometry.fromGeometry(): Undefined vertexUv 625  
⚠ THREE.WebGLProgram: gl.getProgramInfoLog()
```

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



Le code MQTT qui permet de créer l'objet est celui-ci :

```
function createJsonObject(){  
  //On crée l'objet a partir du string  
  obj=JSON.parse(str);  
  //On utilise un Loader pour modeliser l'objet  
  var loader = new THREE.JSONLoader();  
  var model = loader.parse( obj );  
  var material = model.materials[ 0 ];  
  material.morphTargets = true;  
  //On créer l'objet Mesh  
  mesh = new THREE.Mesh( model.geometry, material );  
  mesh.scale.set( 0.01, 0.01, 0.01 );  
  mesh.matrixAutoUpdate = false;  
  mesh.updateMatrix();  
  //On l'ajoute a la scène  
  scene.add( mesh );  
}
```

3 Template Objet – OBJ :

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

Le MQTT n'est actuellement pas utilisable sur ce template, pour pouvoir visualiser l'objet, il faut le changer dans le code , a 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 a l'objet .OBJ.