Amueblado de propiedades en 3D

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

FLOORFY

Autor: Joan Manuel Ramos Refusta

Día de presentación: 30/06/2021

Director: Sergi Jiménez

Ponente: Antonio Chica

1. Floorfy

1. Floorfy

- Tour virtual

2. Proyecto

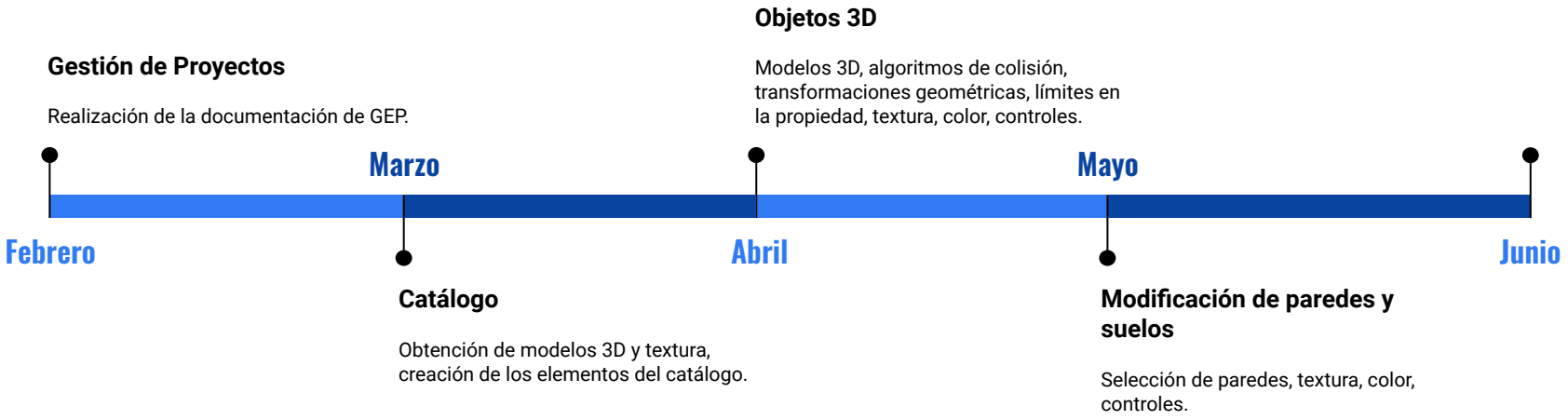
2. Proyecto

- ## Objetivos del proyecto

    - Amueblar inmuebles
    - Manipulación de objetos
    - Cambiar textura y color de objetos
    - Cambiar textura y color de paredes y suelos del inmueble.

2. Proyecto

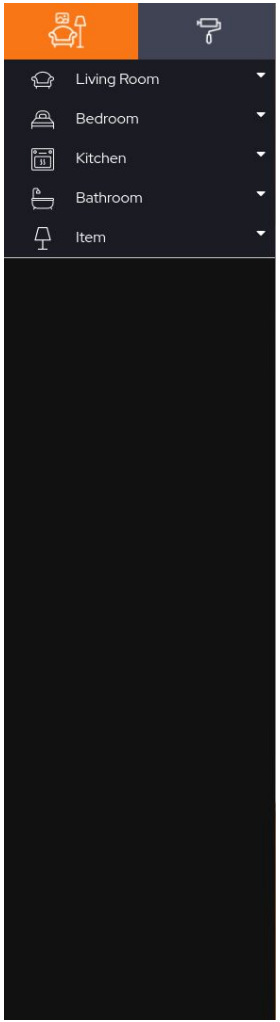
- ## Metodología empleada

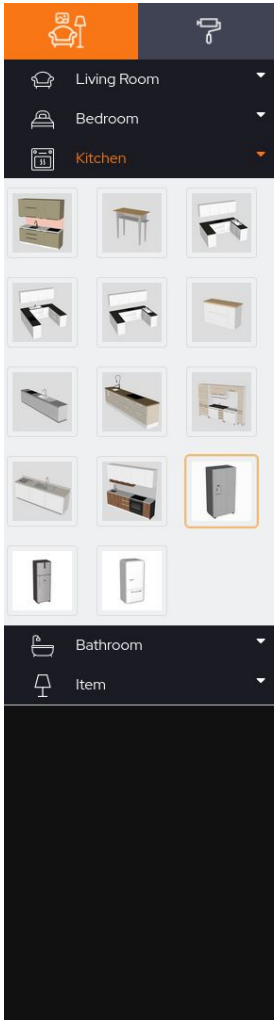


3. Catálogo

3. Catálogo

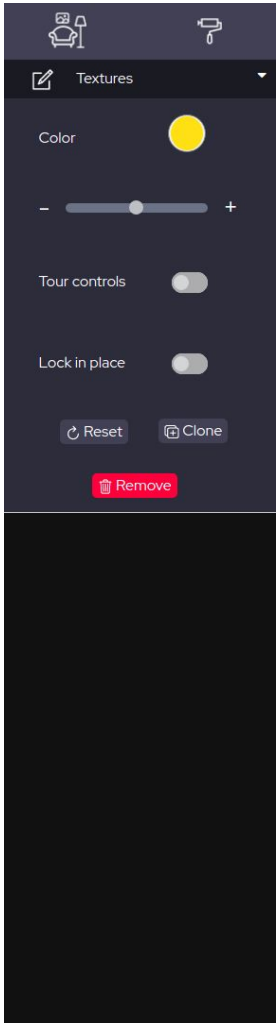
- Catálogo de modelos





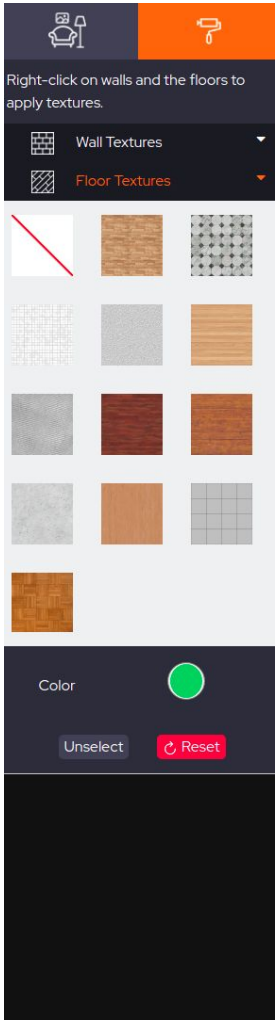
3. Catálogo

- Catálogo de modelos
- Object Options



3. Catálogo

- Catálogo de modelos
- Object Options
- Paint Options



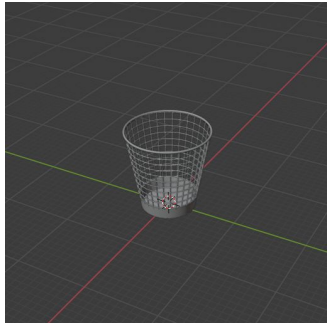
4. Añadir objetos

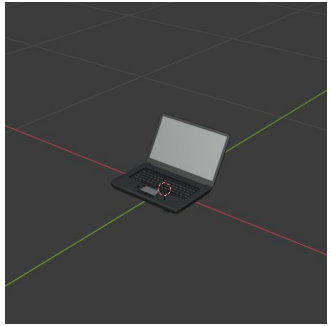
4. Añadir objetos

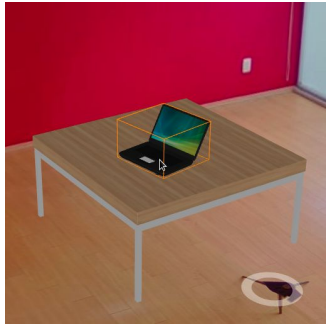
- Bounding Box

4. Añadir objetos

- Bounding Box
- Pivot Point

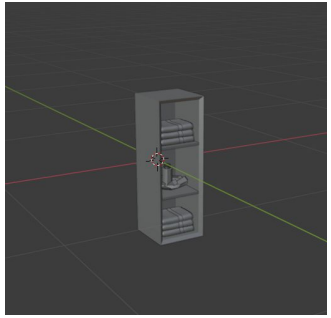


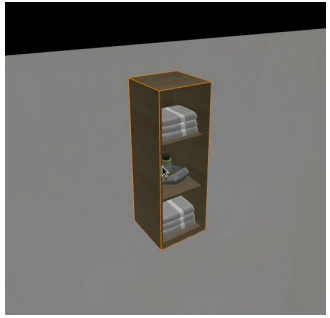


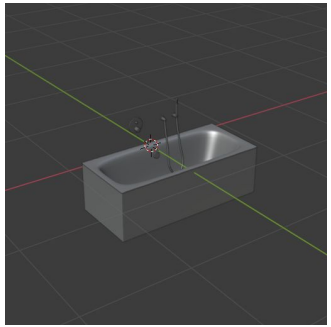


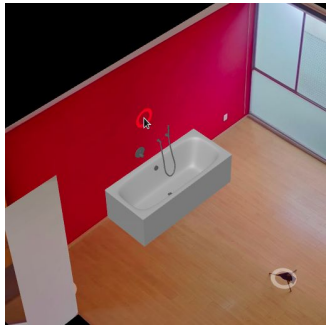
4. Añadir objetos

- Bounding Box
- Pivot Point









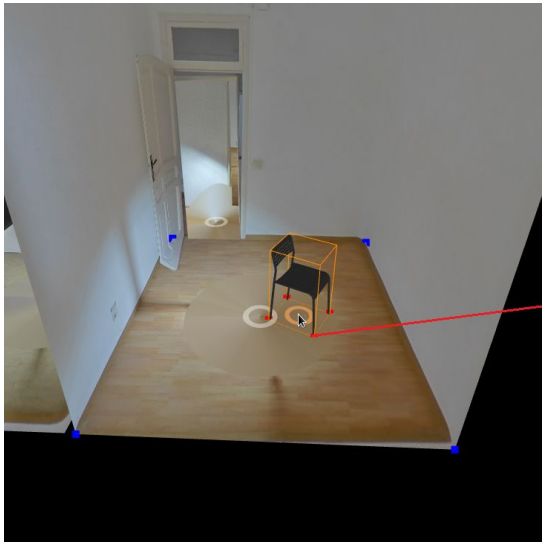
4. Añadir objetos

- Objetos tipo floor

4. Añadir objetos

- ## Objetos tipo floor

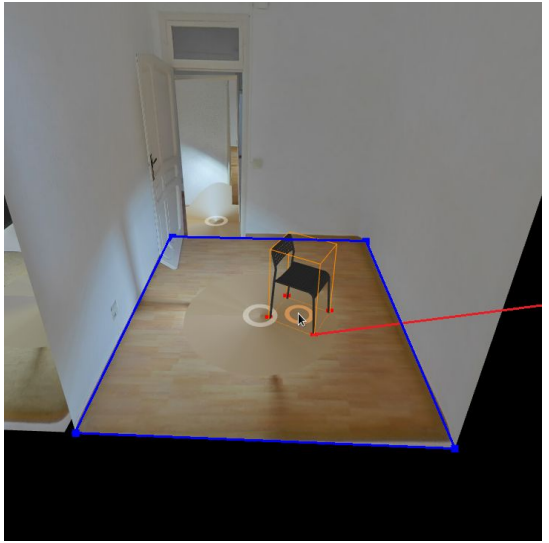
```
Por cada punto pointsObject[i] de pointsObject {
    Generar un punto aux igual que pointsObject[i] con valor en x infinito;
    Inicializar contador = 0;
```



4. Añadir objetos

- ## Objetos tipo floor

```
Por cada punto pointsObject[i] de pointsObject {
    Generar un punto aux igual que pointsObject[i] con valor en x infinito;
    Inicializar contador = 0;
    Por cada segmento edgesFloor[j] de edgesFloor {
        Comprobar intersección de aux-pointsObject[i] con edgesFloor[j];
        Si hay intersección, ++contador;
    }
```



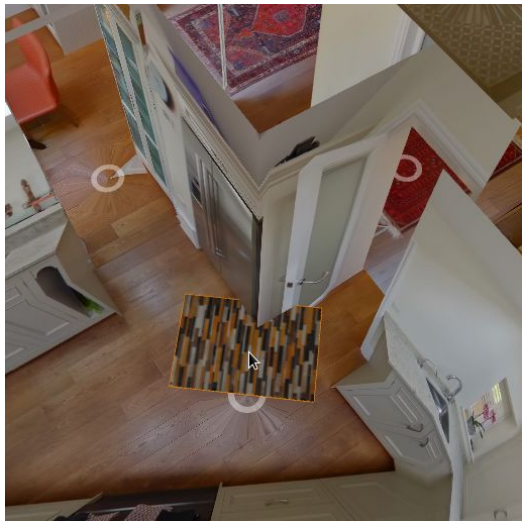
4. Añadir objetos

- ## Objetos tipo floor

```
Por cada punto pointsObject[i] de pointsObject {
    Generar un punto aux igual que pointsObject[i] con valor en x infinito;
    Inicializar contador = 0;
    Por cada segmento edgesFloor[j] de edgesFloor {
        Comprobar intersección de aux-pointsObject[i] con edgesFloor[j];
        Si hay intersección, ++contador;
    }
    Si contador es par, return false;
}
Si finaliza el algoritmo, return true;
```

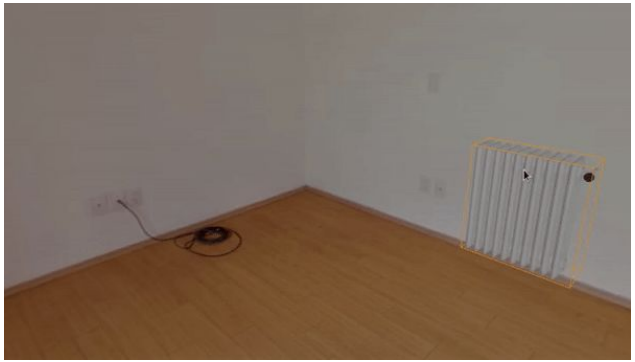
4. Añadir objetos

- Objetos tipo floor



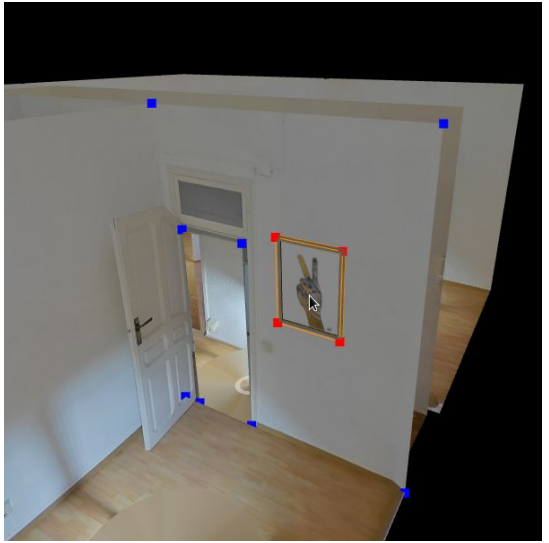
4. Añadir objetos

- Objetos tipo wall y wallFloor



4. Añadir objetos

- Objetos tipo wall y wallFloor



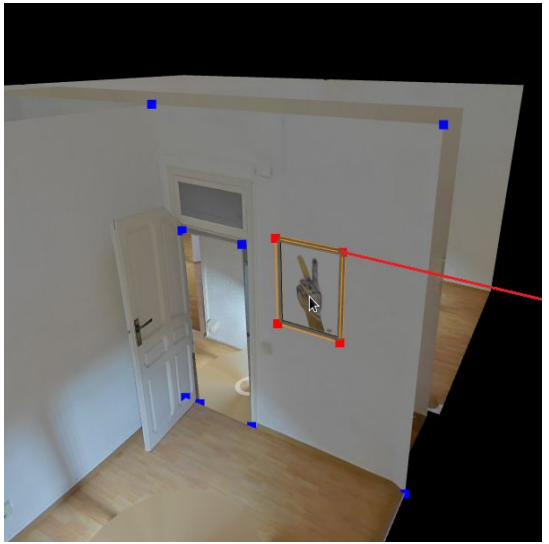
4. Añadir objetos

- ## Objetos tipo wall y wallFloor

Por cada punto **pointsObject[i]** de **pointsObject** {

Generar un punto **aux** que va desde **pointsObject[i]** en dirección perpendicular a la normal de la pared interseccionada;

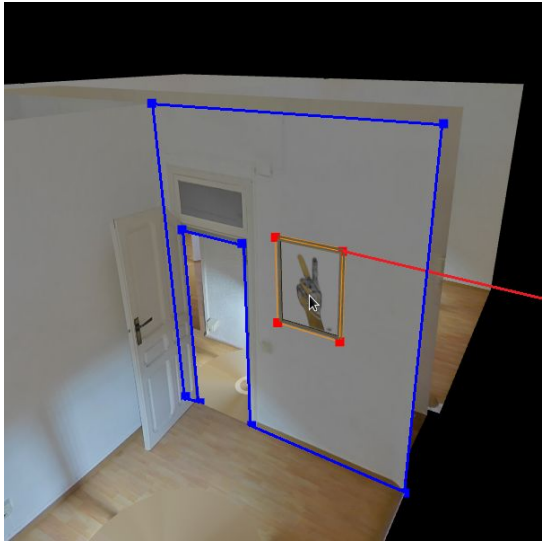
Inicializar **contador** = 0;



4. Añadir objetos

- ## Objetos tipo wall y wallFloor

```
Por cada punto pointsObject[i] de pointsObject {

    Generar un punto aux que va desde pointsObject[i] en dirección
    perpendicular a la normal de la pared interseccionada;

    Inicializar contador = 0;

    Por cada segmento edgesWall[j] de edgesWall {

        Comprobar intersección de aux-pointsObject[i] con edgesWall[j];

        Si hay intersección, ++contador;

    }
```



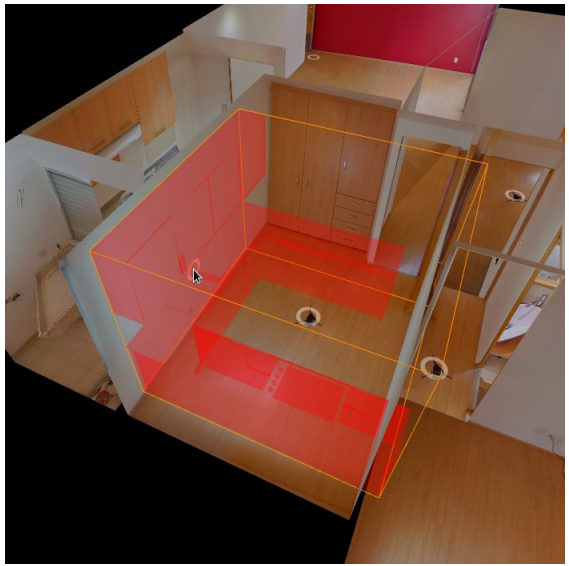
4. Añadir objetos

- ## Objetos tipo wall y wallFloor

```
Por cada punto pointsObject[i] de pointsObject {

    Generar un punto aux que va desde pointsObject[i] en dirección
    perpendicular a la normal de la pared interseccionada;

    Inicializar contador = 0;

    Por cada segmento edgesWall[j] de edgesWall {

        Comprobar intersección de aux-pointsObject[i] con edgesWall[j];

        Si hay intersección, ++contador;

    }

    Si contador es par, return false;

}

Si finaliza el algoritmo, return true;
```

4. Añadir objetos

- Objetos tipo wall y wallFloor



4. Añadir objetos

- ## Colisión entre objetos

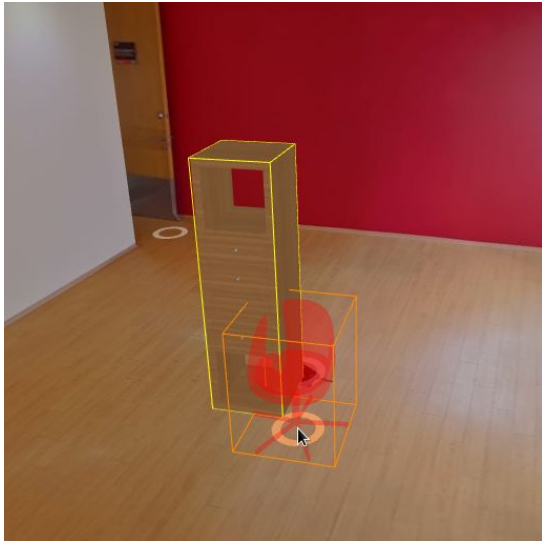
Por cada **cara[i]** del **BBOX** de **A** {

    Por cada **segmento[j]** del **BBOX** de **B** {

        Calcular el punto de **intersección** entre el plano en el que está **cara[i]** y la línea que forma el **segmento[j]**;

        Si el punto de **intersección** encontrado pertenece al **segmento[j]** y a la **cara[i]**, **return false**;

Si finaliza el algoritmo, **return true**;



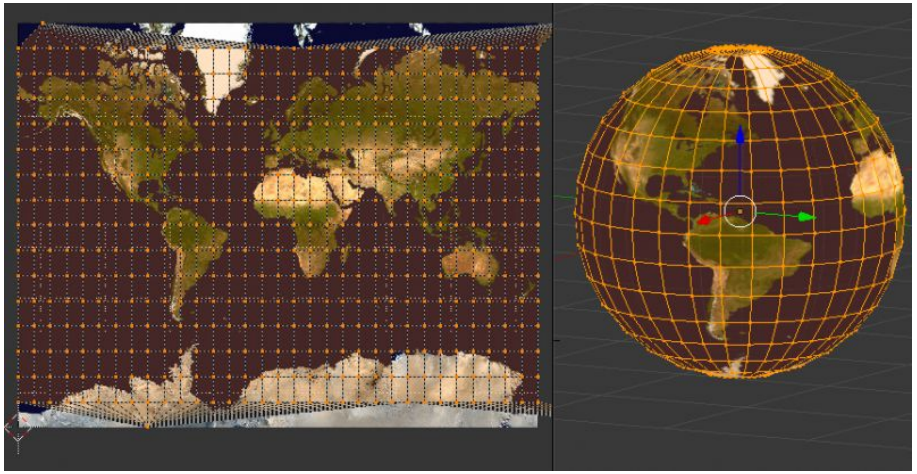
5. Color y textura

5. Color y textura

- Layers

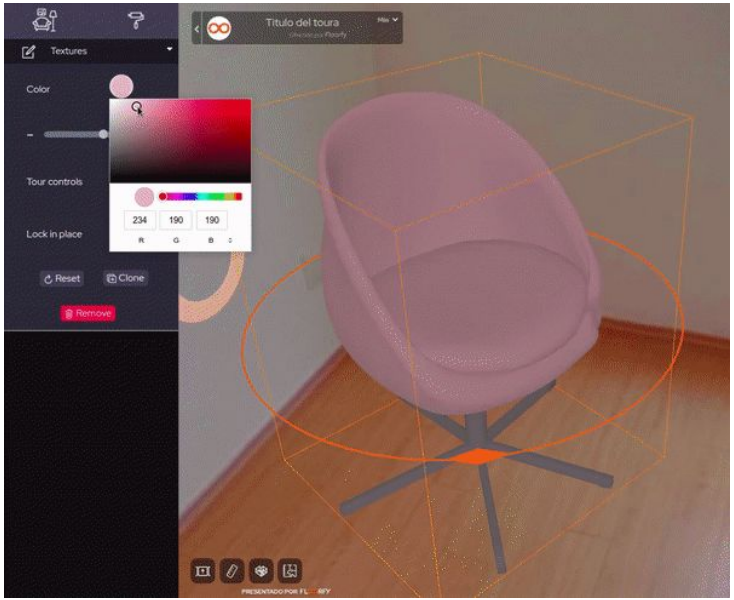
5. Color y textura

- Layers
- UV mapping



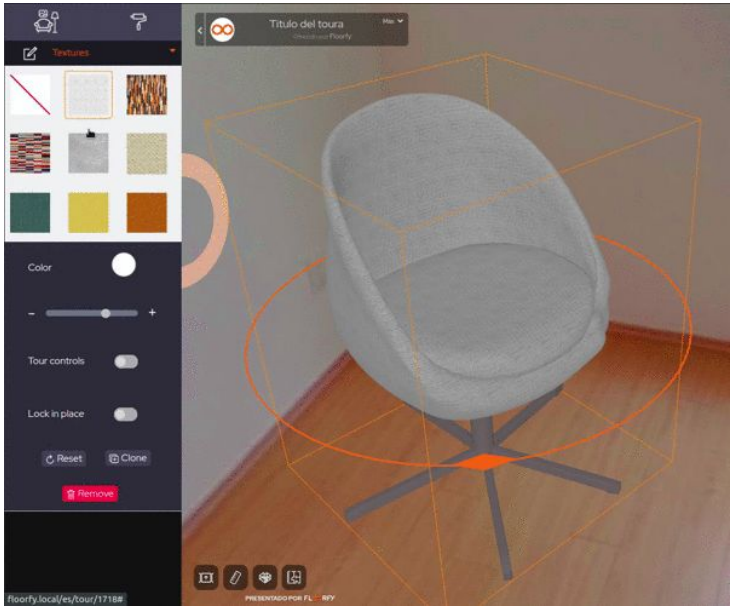
5. Color y textura

- Color



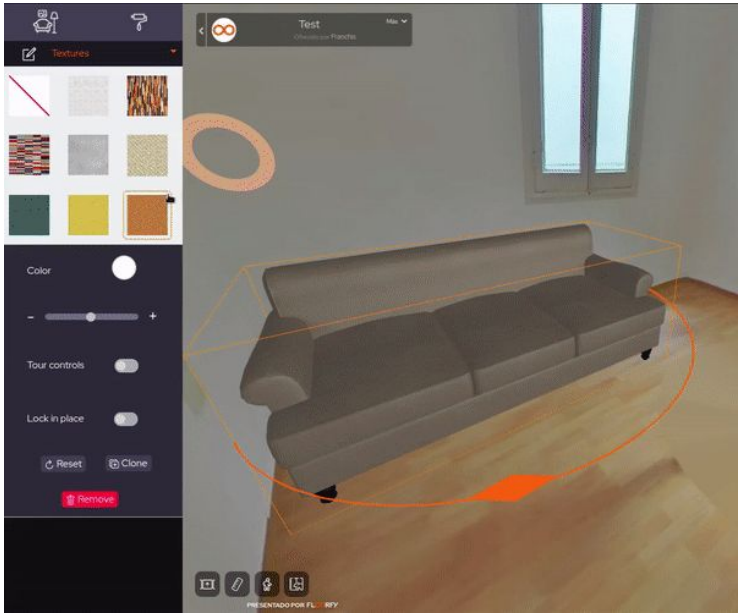
5. Color y textura

- Textura



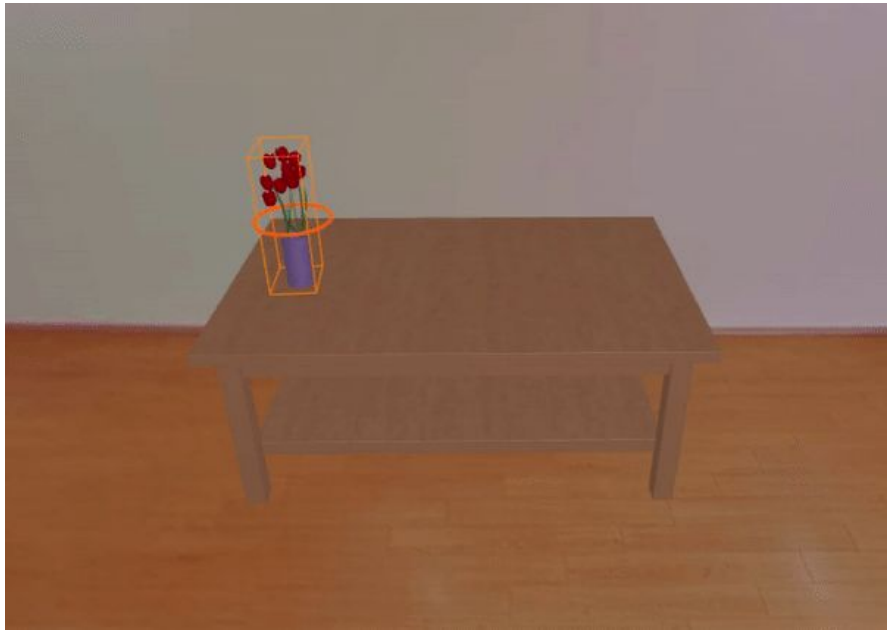
5. Color y textura

- Reset



5. Color y textura

- Clone



6. Controles

6. Controles

- Tour controls

	Tour controls	Controles de manipulación de objetos
Rueda del ratón	Zoom	Escalar objeto
Click derecho	Mover cámara	Rotar 45º el objeto
Click izquierdo	Navegación hotspots	-
Doble click	-	Añadir objeto a la escena

6. Controles

- Transformaciones geométricas
  - Traslación


```
object.position.set(x, y, z);
```

6. Controles

- Transformaciones geométricas
  - Traslación
  - Rotación


```
object.rotationY(angle);
```

6. Controles

- Transformaciones geométricas
  - Traslación
  - Rotación
  - Escalado


```
object.scale.set(x, y, z);
```

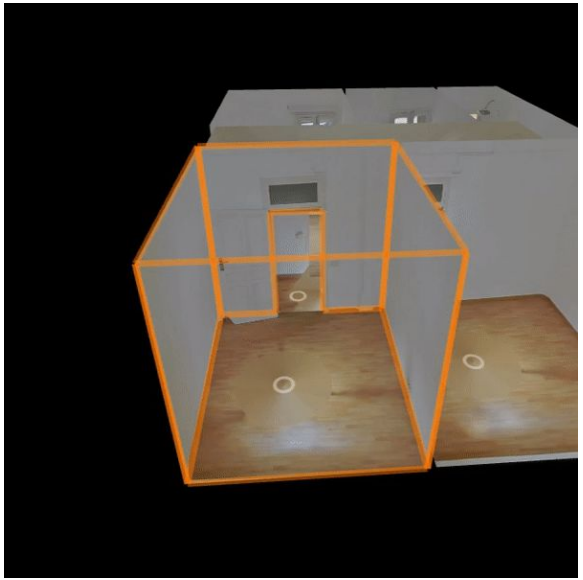
7. Paredes y suelos

7. Paredes y suelos

- Seleccionar paredes/suelos

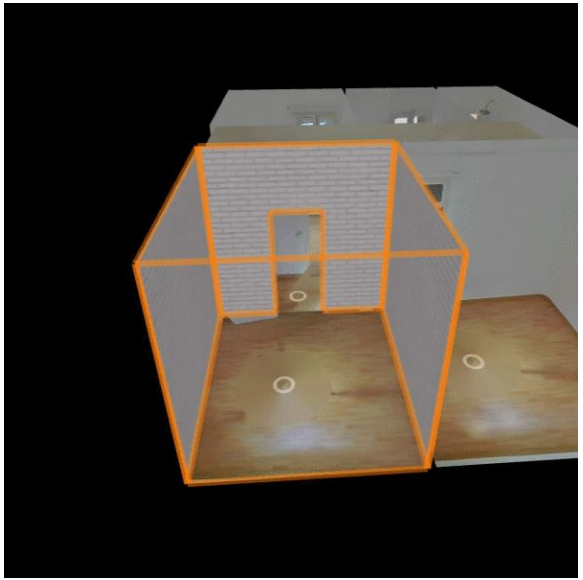
7. Paredes y suelos

- Textura



7. Paredes y suelos

- Color



8. Demo

9. Conclusiones