

VISUALIZACION Y EXPORTACIONES

de reconstrucciones COLMAP

SEBASTIAN MUÑOZ → JUMUNOZLE@UNAL.EDU.CO
CARLOS CAMACHO → CACAMACHO@UNAL.EDU.CO
JUAN DANIEL RAMÍREZ → JUARAMRIEZMO@UNAL.EDU.CO
SERGIO DAVID LOPEZ → SLOPEZPA@UNAL.EDU.CO





ÍNDICE

INTRO A COLMAP

EXPO DE DATOS DESDE COLMAP

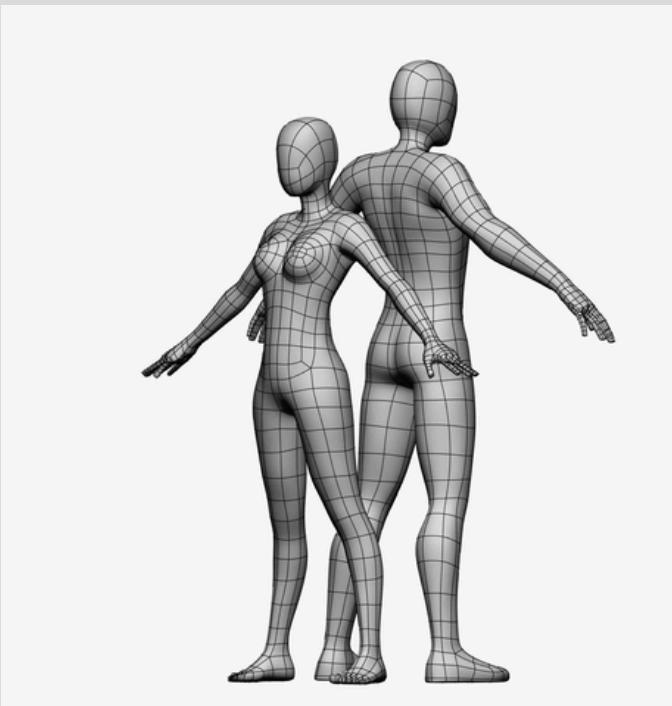
INTEGRACIÓN CON THREE.JS

INTEGRACION EN APPS 3D

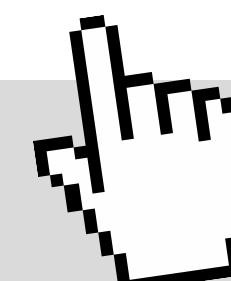
ARCHIVOS EXPORTABLE S DESDE COLMAP

- COLMAP genera reconstrucciones 3D a partir de imágenes.
- Exporta nubes de puntos y mallas en varios formatos:

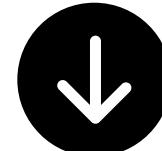
.PLY: NUBE DE PUNTOS O MALLA DENSA.
.OBJ: MALLA CON TEXTURA.
.FBX: USADO EN ENTORNOS COMO UNITY.



QUÉ RESULTADOS GENERA COLMAP?



PARÁMETROS DE RECONSTRUCCIÓN EN COLMAP



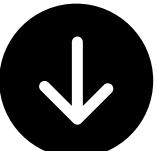
Match features:

- Extracción: SIFT (por defecto).
- Emparejamiento: exhaustive, sequential, spatial, etc.



Structure-from-Motion:

- Estimación de cámaras: Pinhole, Simple Radial, etc.
- Bundle Adjustment para optimizar cámaras y puntos 3D.
- Multi



Multi-view Stereo (MVS):

- Genera mapas de profundidad.
- Fusión produce fused.ply (nube densa).

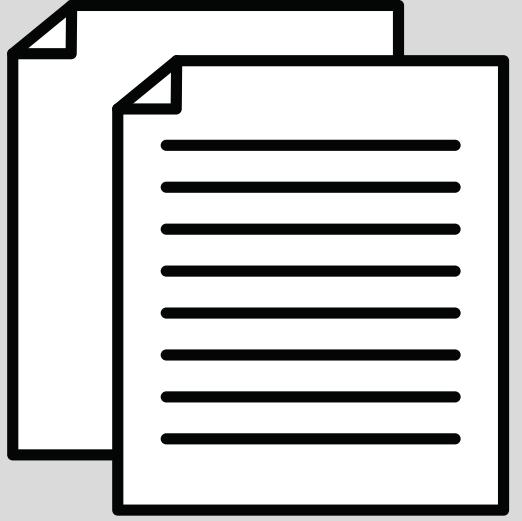
¿CÓMO EXPORTAR?

1. Completa la reconstrucción.
2. En el GUI: File > Export model
 - Elegir: Formato: .ply, .obj, .fbx
 - Tipo: sparse, dense, mesh

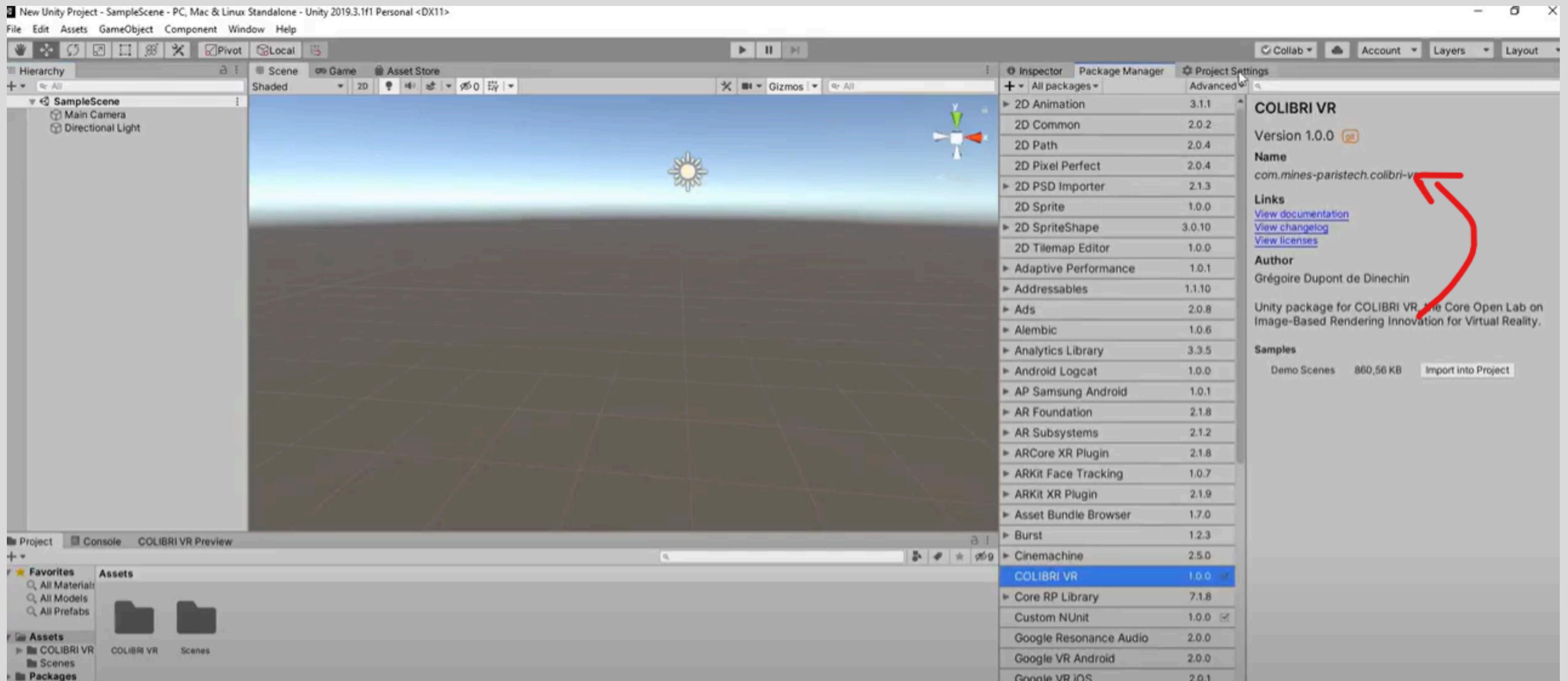
```
colmap model_converter \
--input_path ./sparse/0 \
--output_path ./output \
--output_type PLY
```

1. A veces se requiere convertir el archivo .ply a .obj o .fbx:

- ```
import open3d as o3d
pcd = o3d.io.read_point_cloud("modelo.ply")
o3d.io.write_point_cloud("modelo.obj", pcd)
```

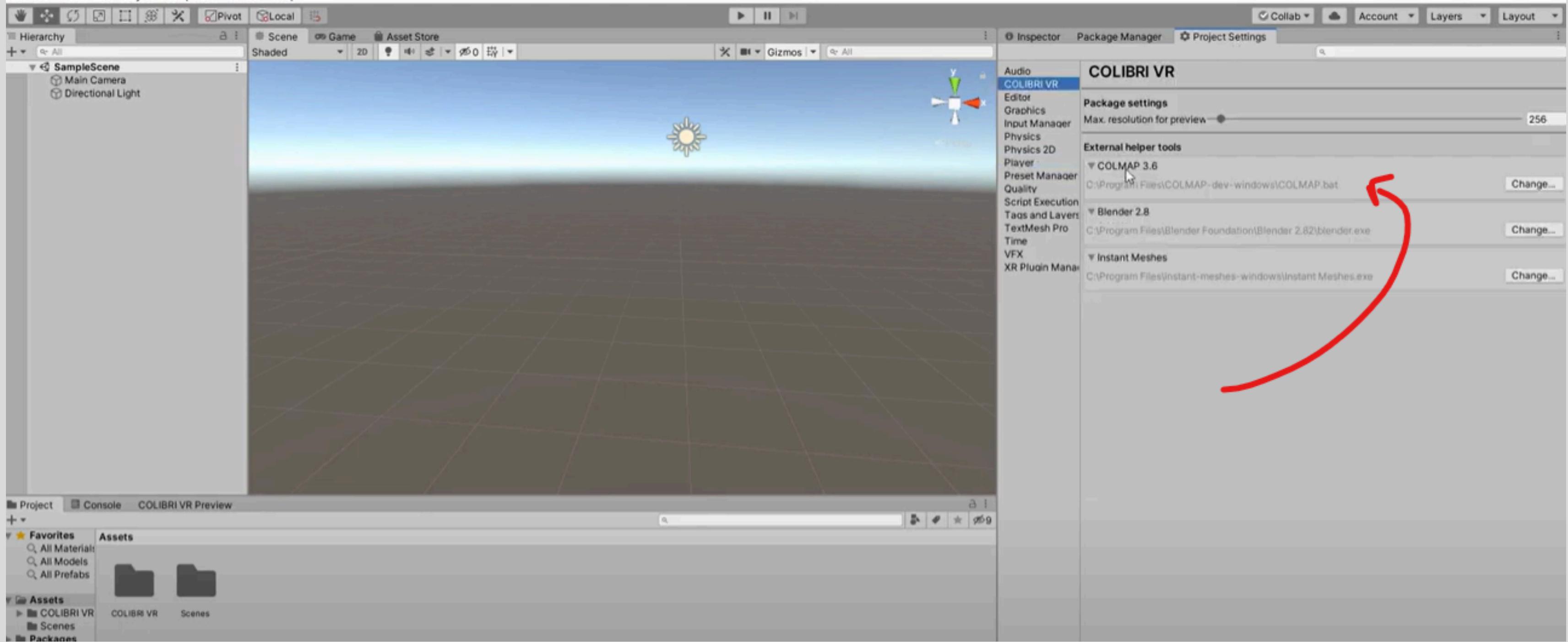


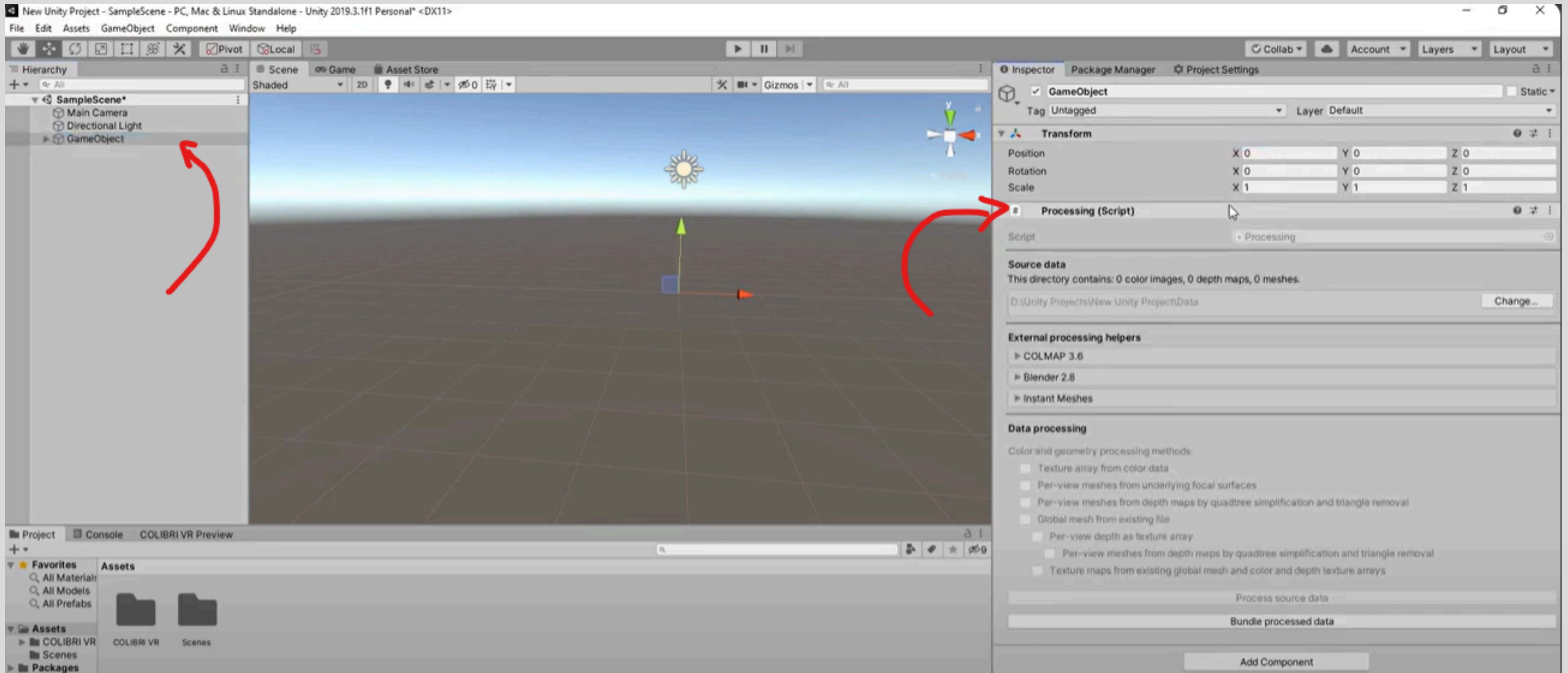
# **EJEMPLOS PRÁCTICOS *CON UNITY***



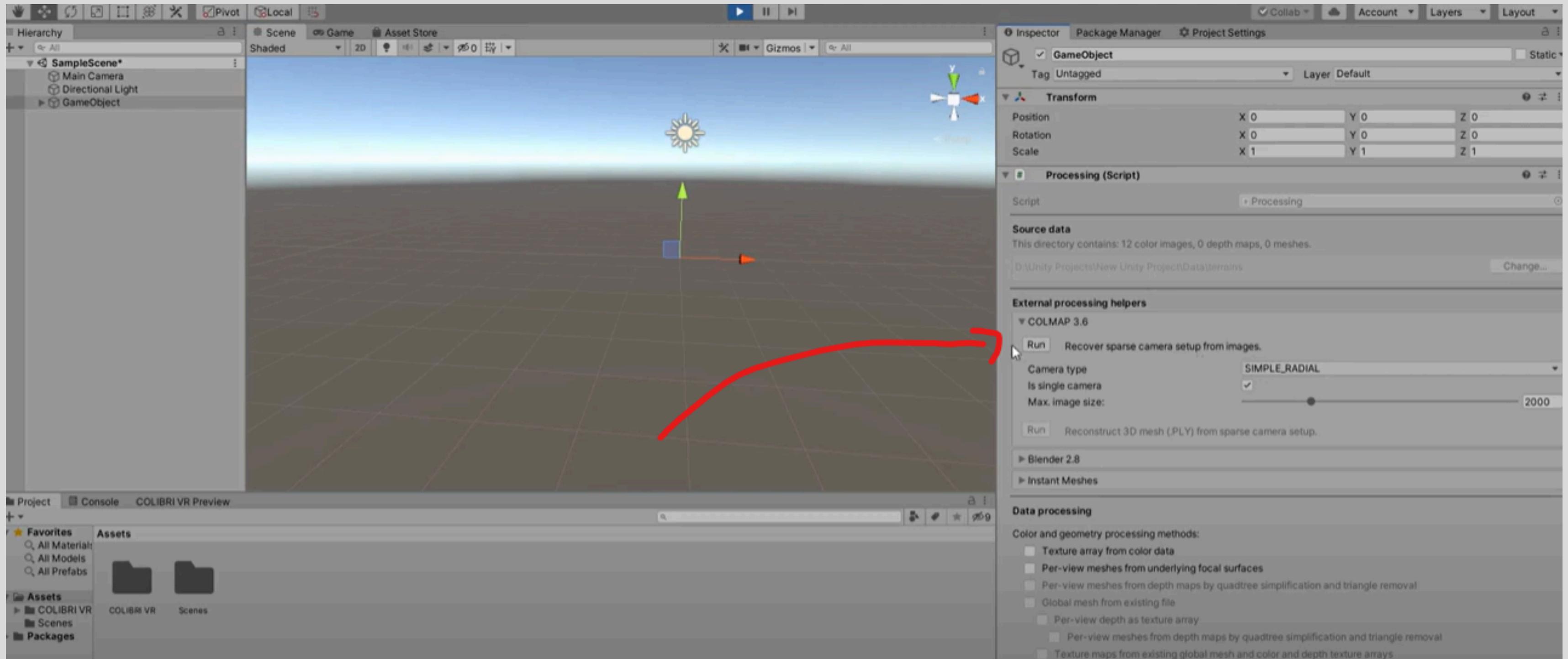
# New Unity Project - SampleScene - PC, Mac & Linux Standalone - Unity 2019.3.1f1 Personal <DX11>

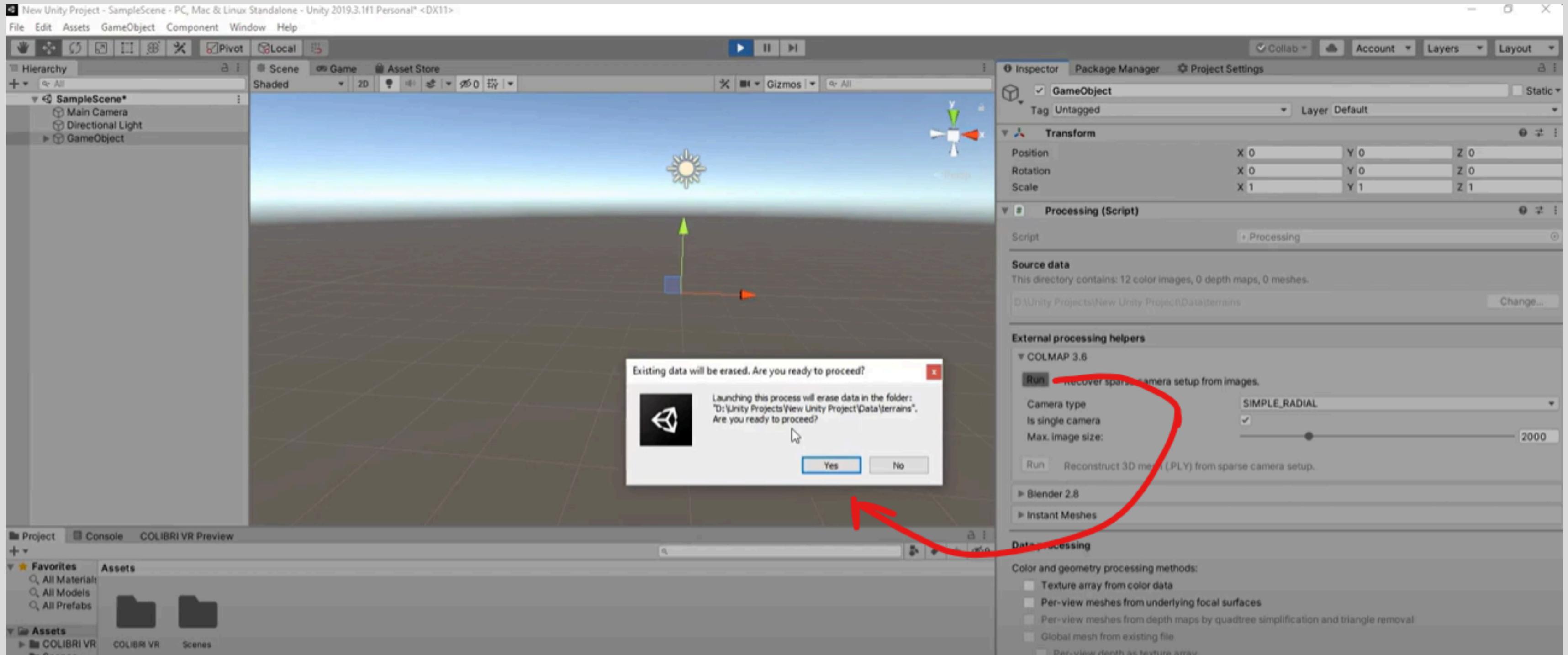
File Edit Assets GameObject Component Window Help

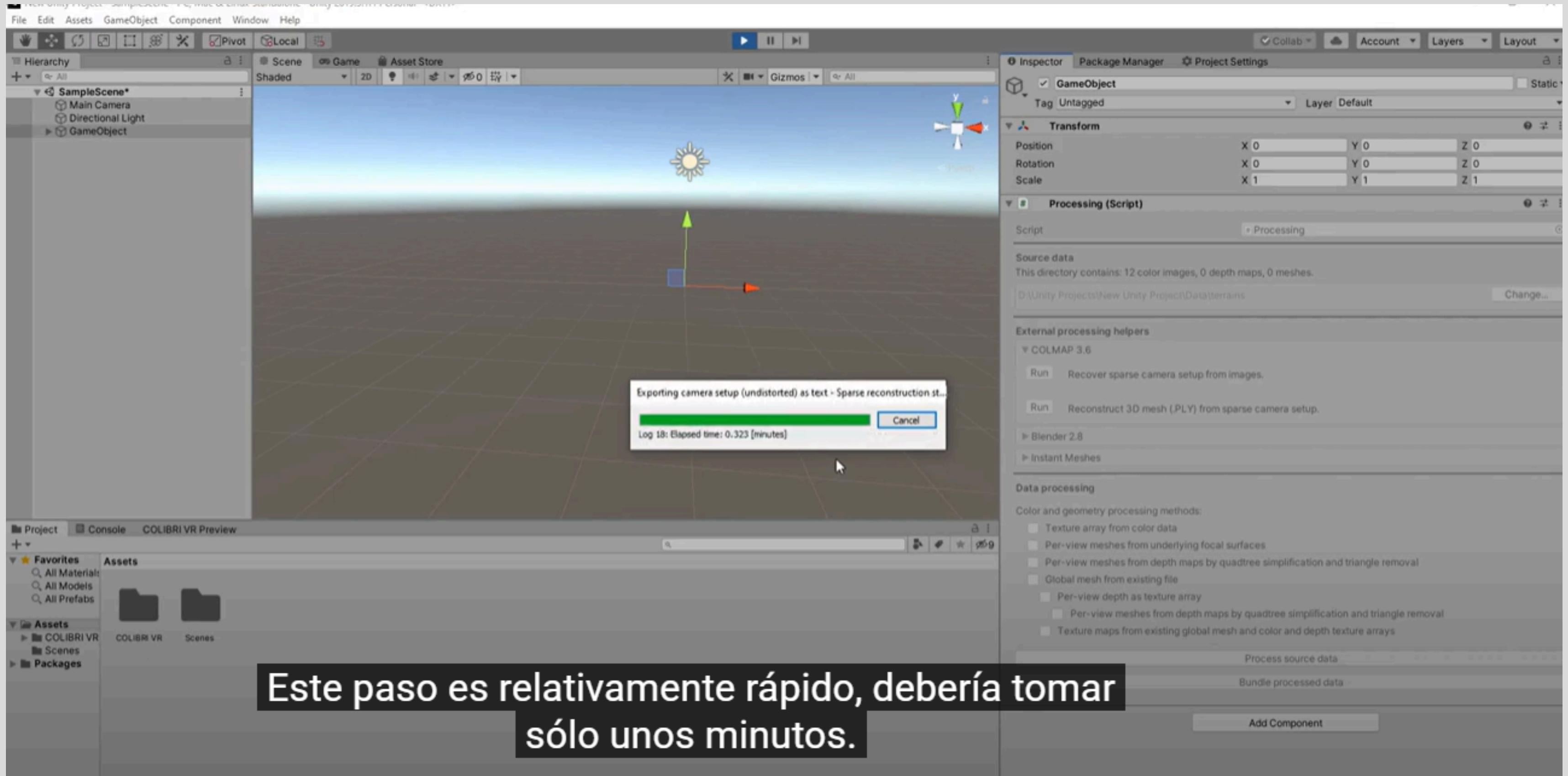


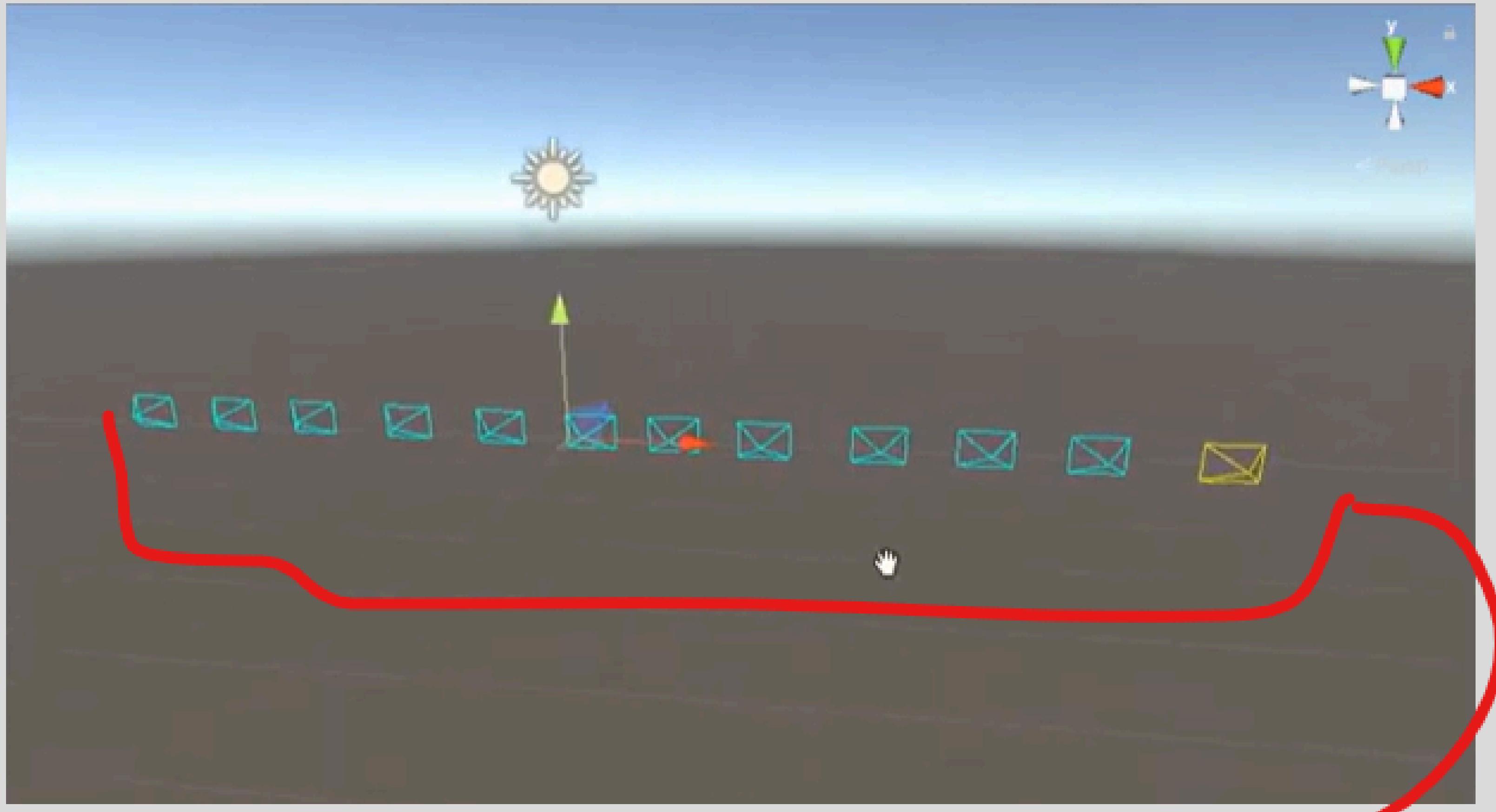


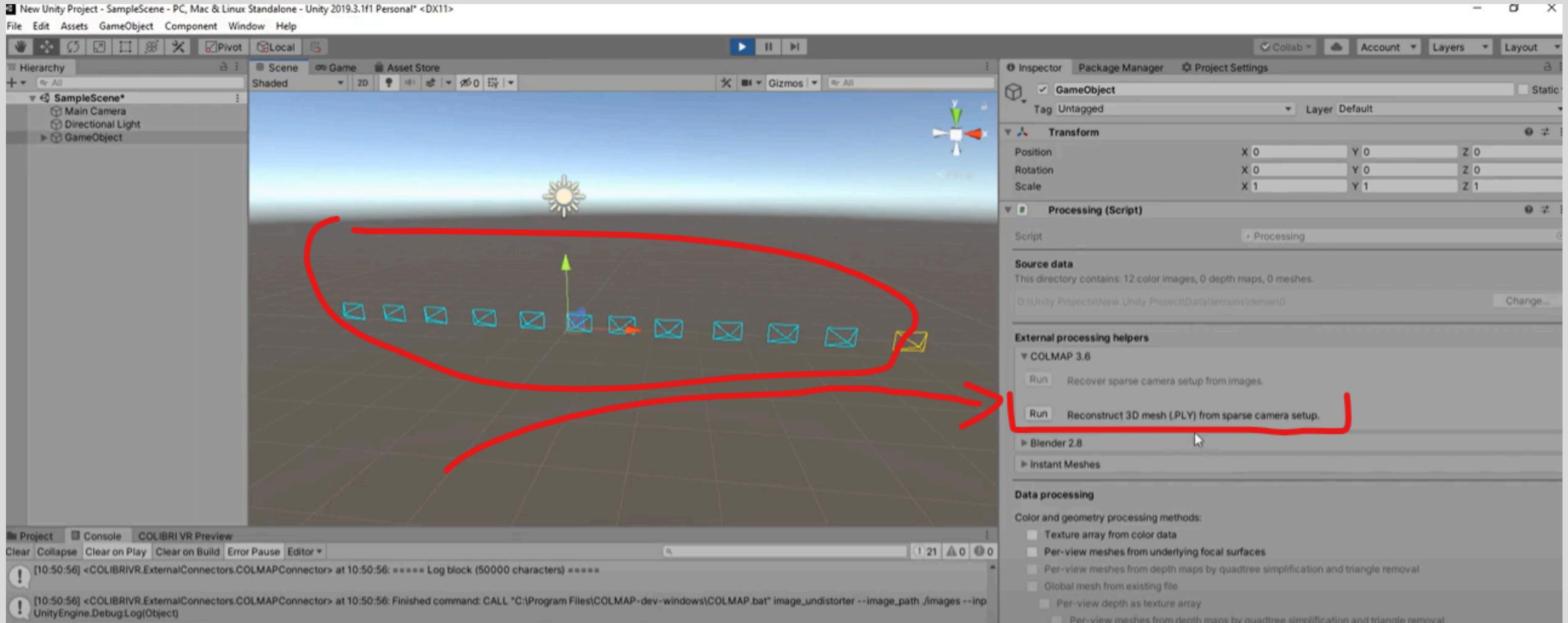


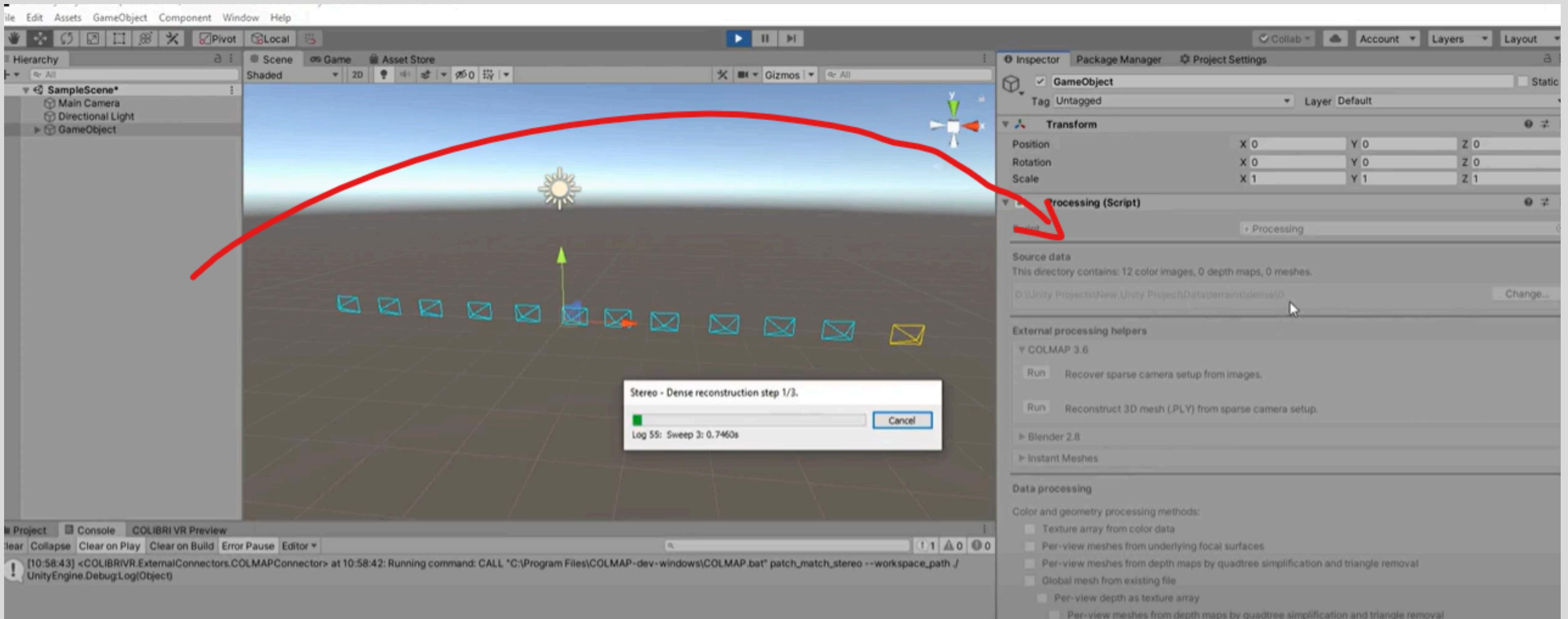


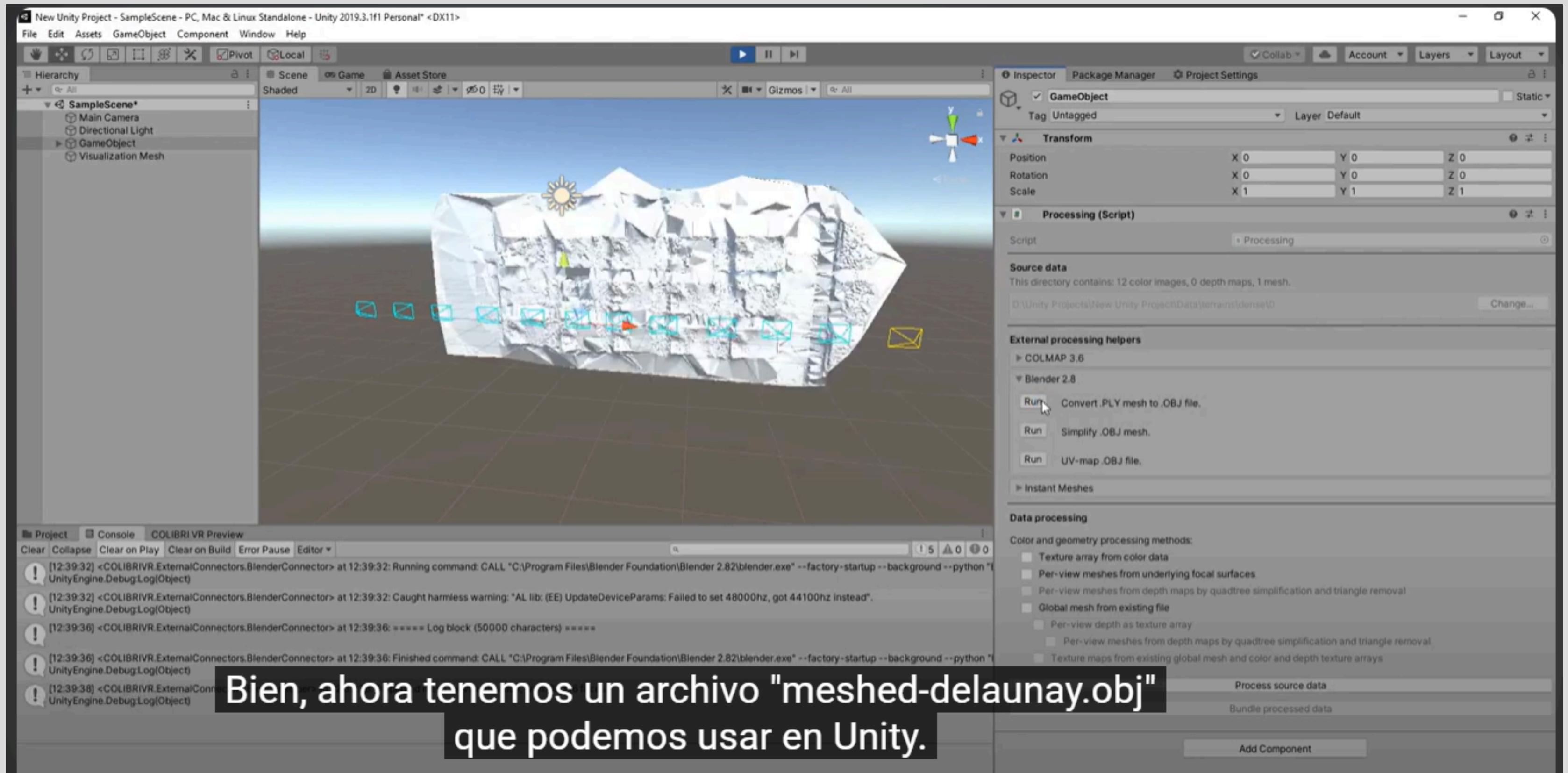


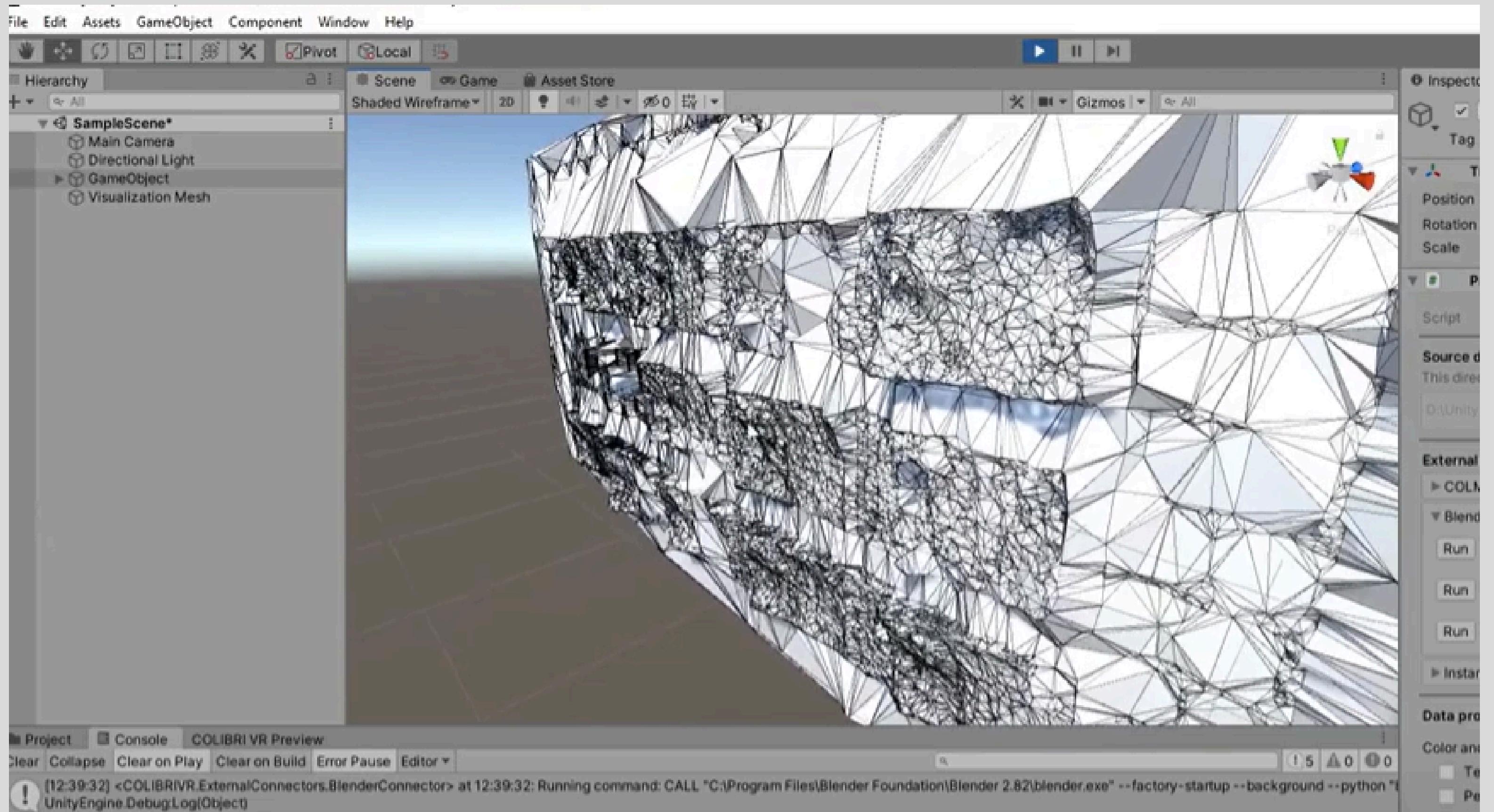












UNA VEZ QUE COLMAP HA GENERADO UNA RECONSTRUCCIÓN 3D, YA SEA UNA NUBE DE PUNTOS O UNA MALLA DENSA, EL SIGUIENTE PASO ES PREPARAR ESOS DATOS PARA SU USO EN ENTORNOS DE VISUALIZACIÓN COMO UNITY, THREE.JS O BLENDER. PARA ELLO, ES NECESARIO EXPORTAR, AJUSTAR Y LIMPIAR ADECUADAMENTE LOS MODELOS.

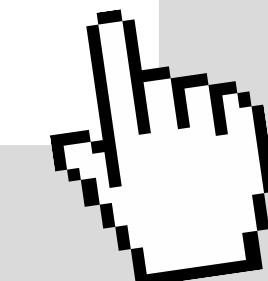


## Exportación desde COLMAP

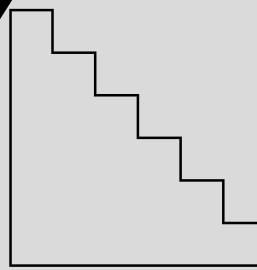


COLMAP permite exportar los resultados en varios formatos estándar o mediante herramientas externas en .fbx. Estos formatos son ampliamente compatibles con aplicaciones 3D modernas. Al exportar, es común utilizar:

- .ply para nubes de puntos crudas o filtradas.
- .obj para mallas texturizadas generadas con Delaunay o Poisson.
- .fbx si se requiere compatibilidad directa con motores como Unity.



# AJUSTE DE ESCALA, ORIENTACIÓN Y LIMPIEZA (THREE.JS)



## 1. Cargar la nube de puntos o malla



```
import { GLTFLoader } from 'three/examples/jsm/loaders/GLTFLoader';
import { PointLight, Scene, PerspectiveCamera, WebGLRenderer } from 'three';

const loader = new GLTFLoader();
const scene = new Scene();

loader.load('/models/colmap_output.glb', (gltf) => {
 const model = gltf.scene;

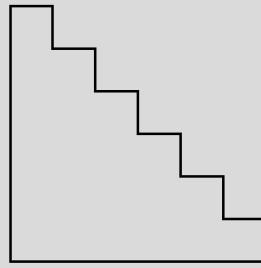
 // Ajustes clave:
 model.scale.set(0.01, 0.01, 0.01); // ← Ajusta la escala
 model.rotation.set(-Math.PI / 2, 0, 0); // ← Orientación (si Z apunta arriba)
 model.position.set(0, 0, 0); // ← Posición inicial

 scene.add(model);
});
```

Cuando se exporta desde COLMAP (.ply, .obj), se puede obtener ruido, duplicados o geometría sobrante. Para integrar esto correctamente en una app 3D, es útil filtrar o limpiar antes o durante la carga.

¿Qué es "limpieza"?

1. Eliminar puntos o geometría fuera de un rango espacial (por ejemplo, demasiado alejados).
2. Filtrar por bounding box: sólo conservar una región 3D.
3. Ocultar mallas específicas si son subcomponentes innecesarios.



Opcion 1: Limpieza con bounding box en Three.js (si ya está cargado)

```
model.traverse((child) => {
 if (child.isMesh) {
 child.geometry.computeBoundingBox();
 const box = child.geometry.boundingBox;

 const min = box.min;
 const max = box.max;

 // Si el objeto está fuera de un rango aceptable, lo ocultamos
 const isOutside = max.z > 5 || min.z < -2 || max.x > 5 || min.x < -5;
 if (isOutside) {
 child.visible = false;
 }
 }
});
```

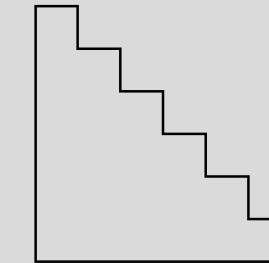
Opción 2: Limpieza previa con herramientas gráficas

En CloudCompare:  
Cargar .ply → Usar "Crop Tool" para seleccionar una  
región 3D.  
Aplicar filtro por densidad o distancia.  
Exportar la versión reducida.

En Blender:  
Importar .ply o .obj.  
Usar modo Edit → Box select para borrar puntos  
aislados.  
Exportar como .glb o .gltf limpio.

Esto permite en tiempo real desactivar parte de la geometría cargada

Opción 3: Limpieza automatizada por densidad (en Python)  
Si se trabaja por fuera de Three.js, se puede hacer la limpieza  
con Open3D:



```
import open3d as o3d

pcd = o3d.io.read_point_cloud("dense.ply")
cl, ind = pcd.remove_statistical_outlier(nb_neighbors=20, std_ratio=2.0)

pcd_clean = pcd.select_by_index(ind)
o3d.io.write_point_cloud("cleaned_dense.ply", pcd_clean)
```

Esto elimina puntos dispersos basados en la densidad local.



**GRACIAS POR LA  
ATENCIÓN**

