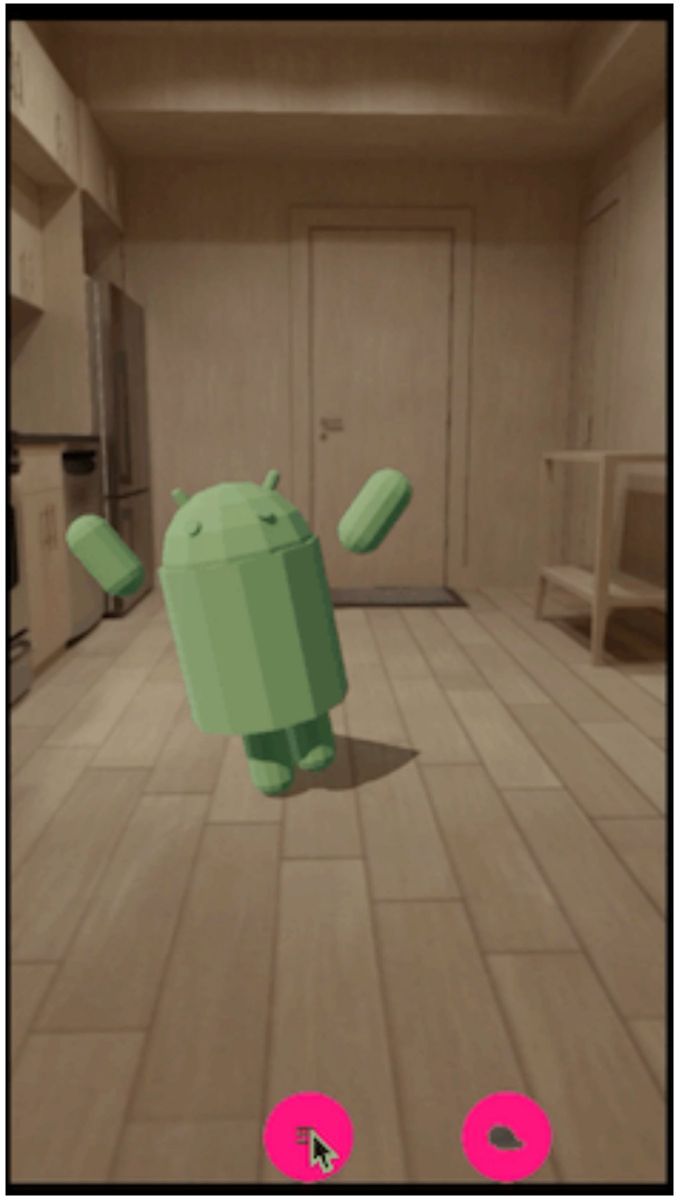
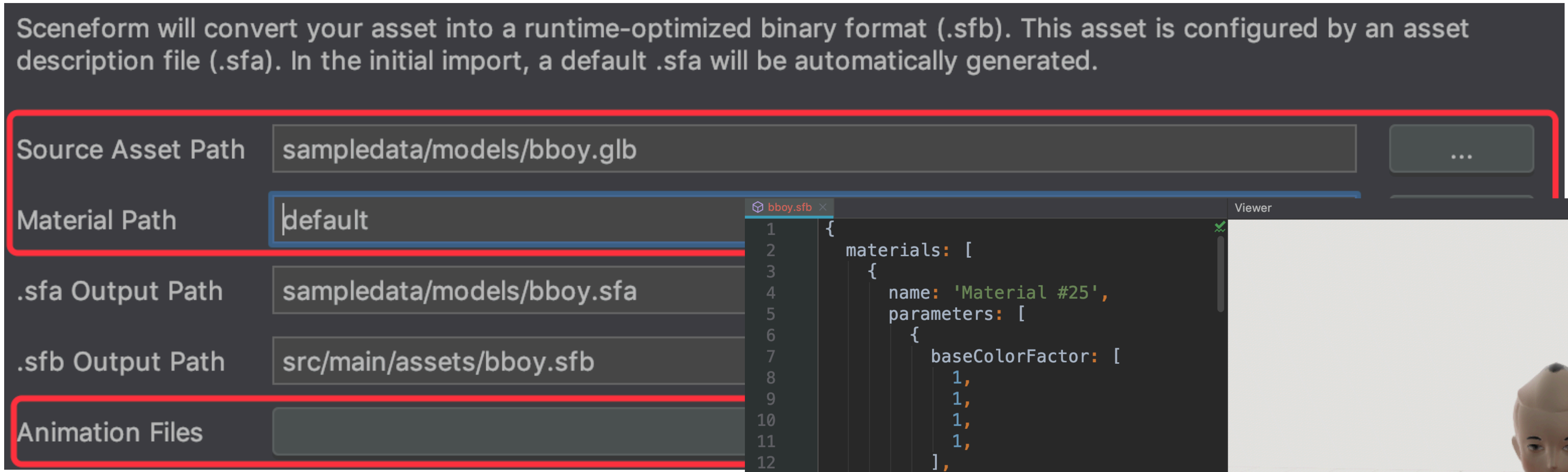


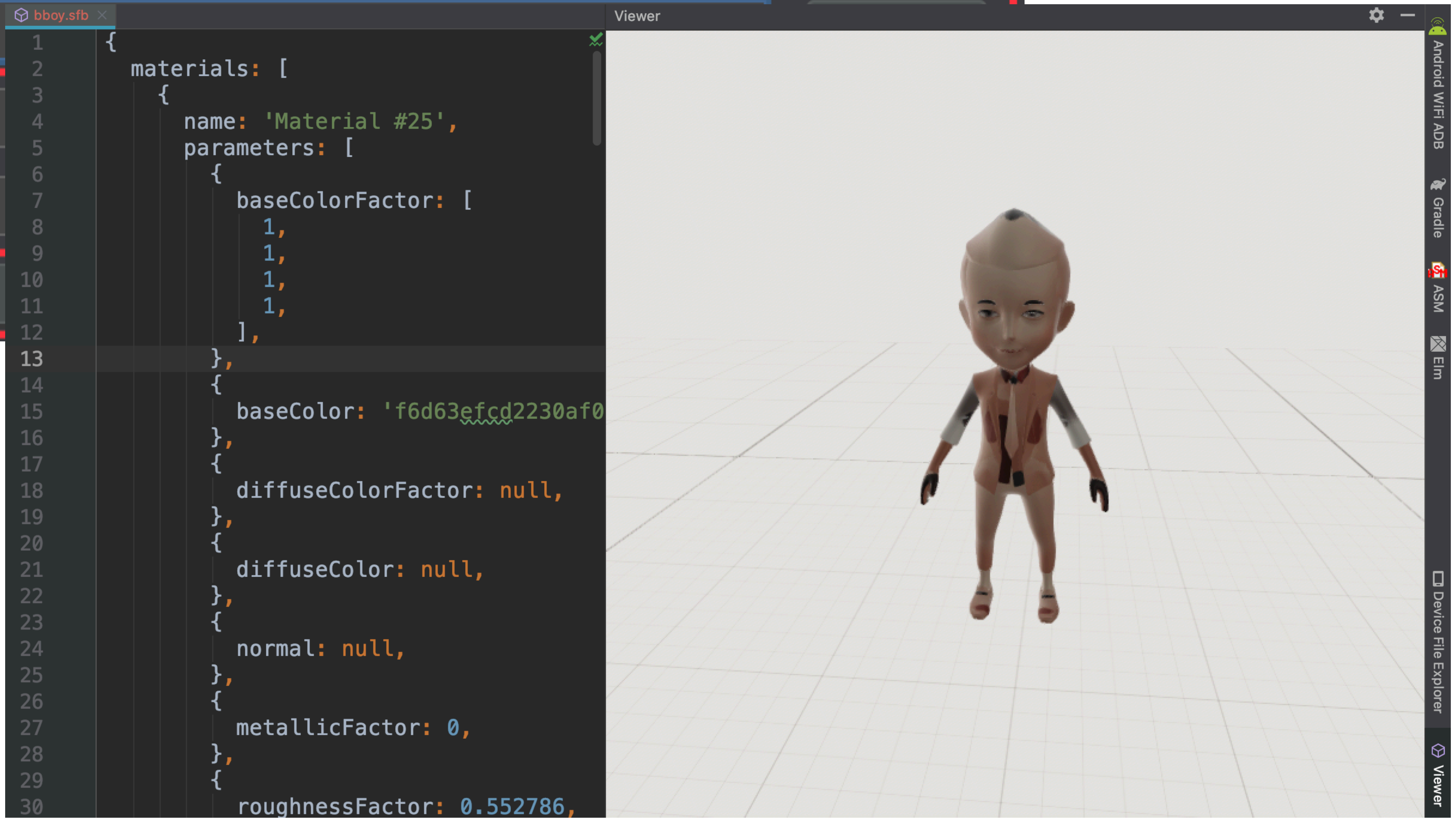
插件工具 Google Sceneform Tools (Beta)

将输入的各格式文件转换成 Sceneform 格式，并支持预览。Sceneform 支持 OBJ、FBX 和 glTF 格式的 3D asset，最终都转成sfb格式。



Sceneform will convert your asset into a runtime-optimized binary format (.sfb). This asset is configured by an asset description file (.sfa). In the initial import, a default .sfa will be automatically generated.

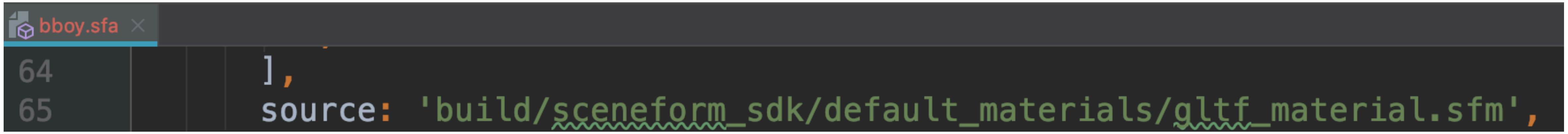
Source Asset Path	sampledata/models/bboy.glb
Material Path	default
.sfa Output Path	sampledata/models/bboy.sfa
.sfb Output Path	src/main/assets/bboy.sfb
Animation Files	



```
{
  materials: [
    {
      name: 'Material #25',
      parameters: [
        {
          baseColorFactor: [
            1,
            1,
            1,
            1,
          ],
        },
        {
          baseColor: 'f6d63efcd2230af0
        },
        {
          diffuseColorFactor: null,
        },
        {
          diffuseColor: null,
        },
        {
          normal: null,
        },
        {
          metallicFactor: 0,
        },
        {
          roughnessFactor: 0.552786,
```

● Material材料素材

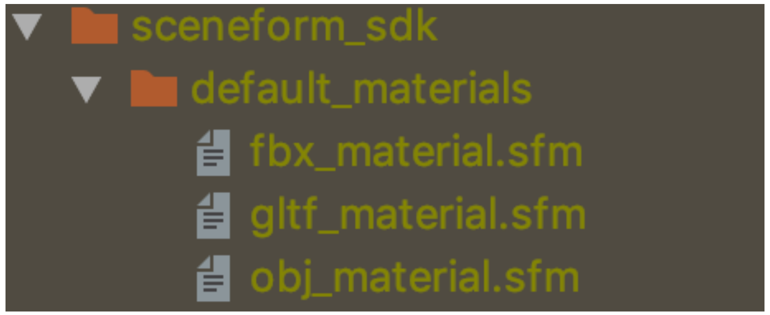
使用缺省的mat



```
bboy.sfa
        ],
        source: 'build/sceneform_sdk/default_materials/gltf_material.sfm',
```

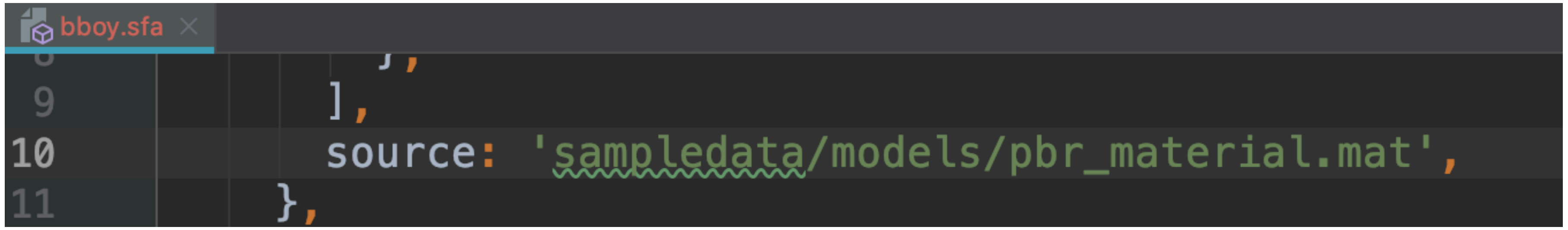
对于**默认材料**（`*.sfm`），请参阅受支持的参数的列表：

- OBJ asset：`obj_material.sfm`
- FBX asset：`fbx_material.sfm`
- glTF asset：`gltf_material.sfm`



```
▼ sceneform_sdk
  ▼ default_materials
      fbx_material.sfm
      gltf_material.sfm
      obj_material.sfm
```

使用自定义的mat（插件工具会导致AS crash）

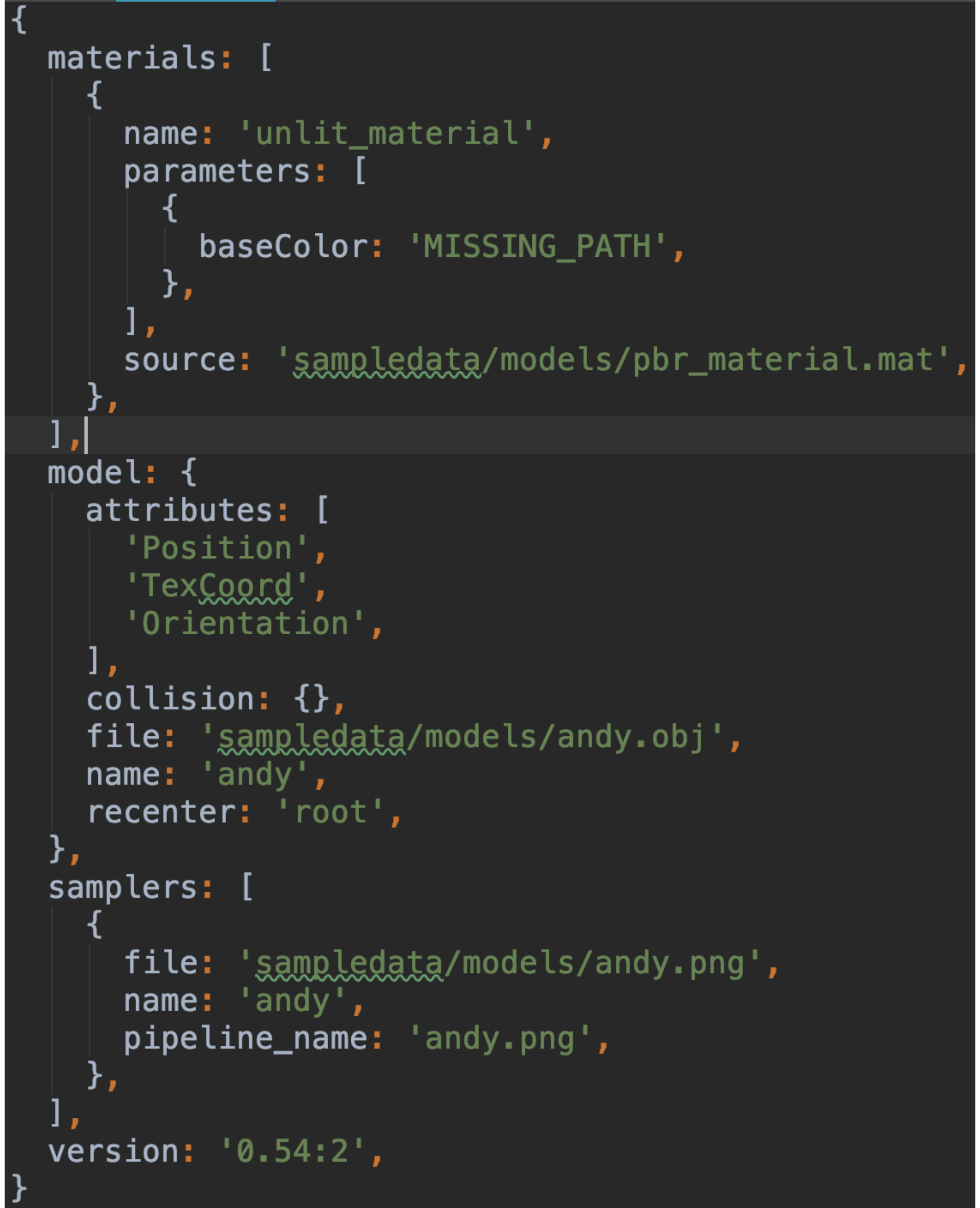


```
bboy.sfa
      ],
      source: 'sampledata/models/pbr_material.mat',
    },
```

● **sfa材料素材**

Sceneform Asset 定义 (*.sfa) 文件是 Sceneform 二进制 asset (*.sfb) 的配置文件。 它指向您的源 asset 中的模型、材料定义和纹理。

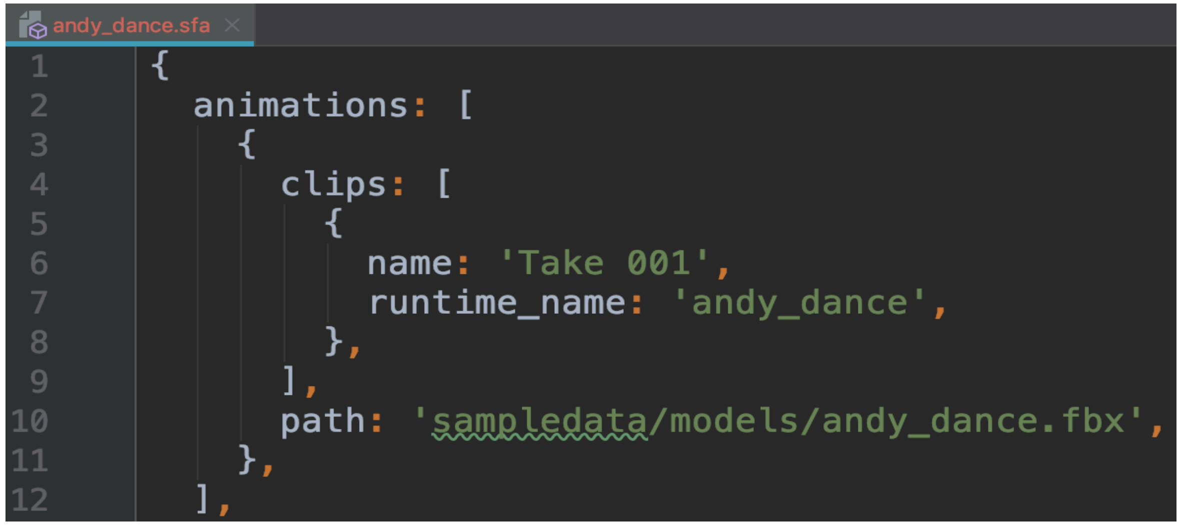
此文件会在首次导入时由 Sceneform Android Studio 插件自动生成，但可以调整属性。



```
{
  materials: [
    {
      name: 'unlit_material',
      parameters: [
        {
          baseColor: 'MISSING_PATH',
        },
      ],
      source: 'sampledata/models/pbr_material.mat',
    },
  ],
  model: {
    attributes: [
      'Position',
      'TexCoord',
      'Orientation',
    ],
    collision: {},
    file: 'sampledata/models/andy.obj',
    name: 'andy',
    recenter: 'root',
  },
  samplers: [
    {
      file: 'sampledata/models/andy.png',
      name: 'andy',
      pipeline_name: 'andy.png',
    },
  ],
  version: '0.54:2',
}
```

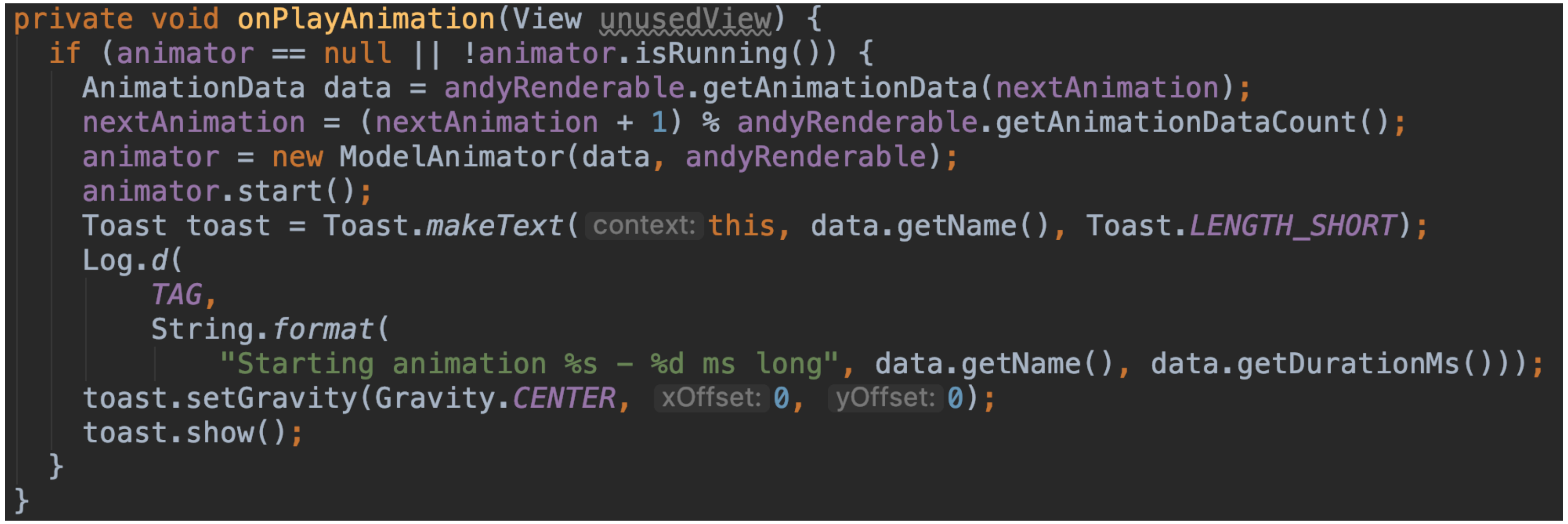
● model animation

- ✓ 动画必须制作成fbx文件
- ✓ 为了确保兼容ARCore，fbx文件保存的时候必须开启一些设置
- ✓ 可以将多组动画打到一个sfb文件中



andy_dance.sfa

```
{
  animations: [
    {
      clips: [
        {
          name: 'Take 001',
          runtime_name: 'andy_dance',
        },
      ],
      path: 'sampledata/models/andy_dance.fbx',
    },
  ],
```



```
private void onPlayAnimation(View unusedView) {
  if (animator == null || !animator.isRunning()) {
    AnimationData data = andyRenderable.getAnimationData(nextAnimation);
    nextAnimation = (nextAnimation + 1) % andyRenderable.getAnimationDataCount();
    animator = new ModelAnimator(data, andyRenderable);
    animator.start();
    Toast toast = Toast.makeText( context: this, data.getName(), Toast.LENGTH_SHORT);
    Log.d(
        TAG,
        String.format(
            "Starting animation %s - %d ms long", data.getName(), data.getDurationMs()));
    toast.setGravity(Gravity.CENTER,  xOffset: 0,  yOffset: 0);
    toast.show();
  }
}
```

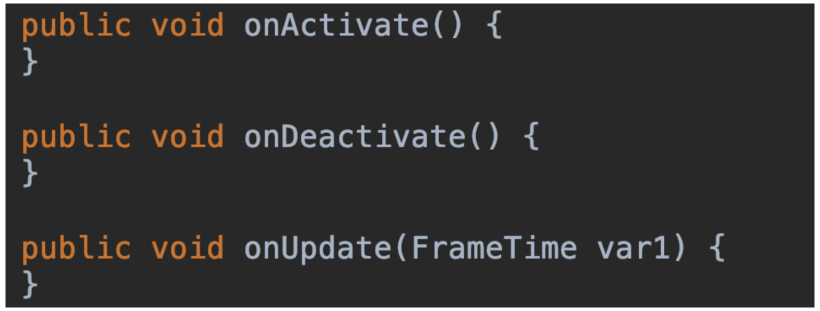
```
// Support for animated model renderables.
implementation "com.google.ar.sceneform:animation:1.15.0"
```

libsceneform_animation.so 172.4 KB

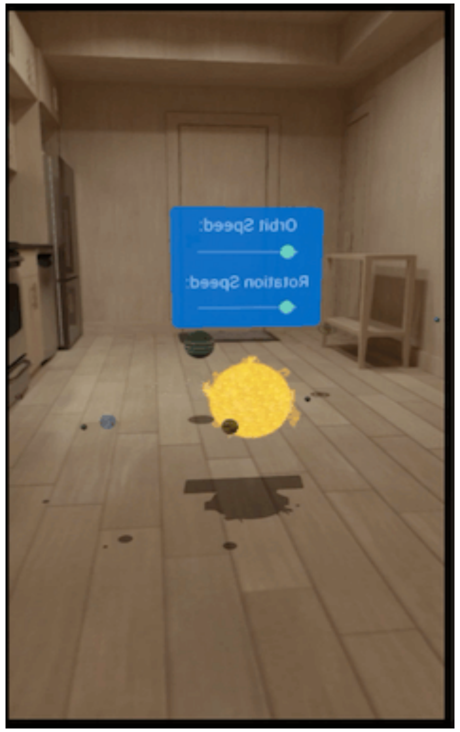
- **property animation**

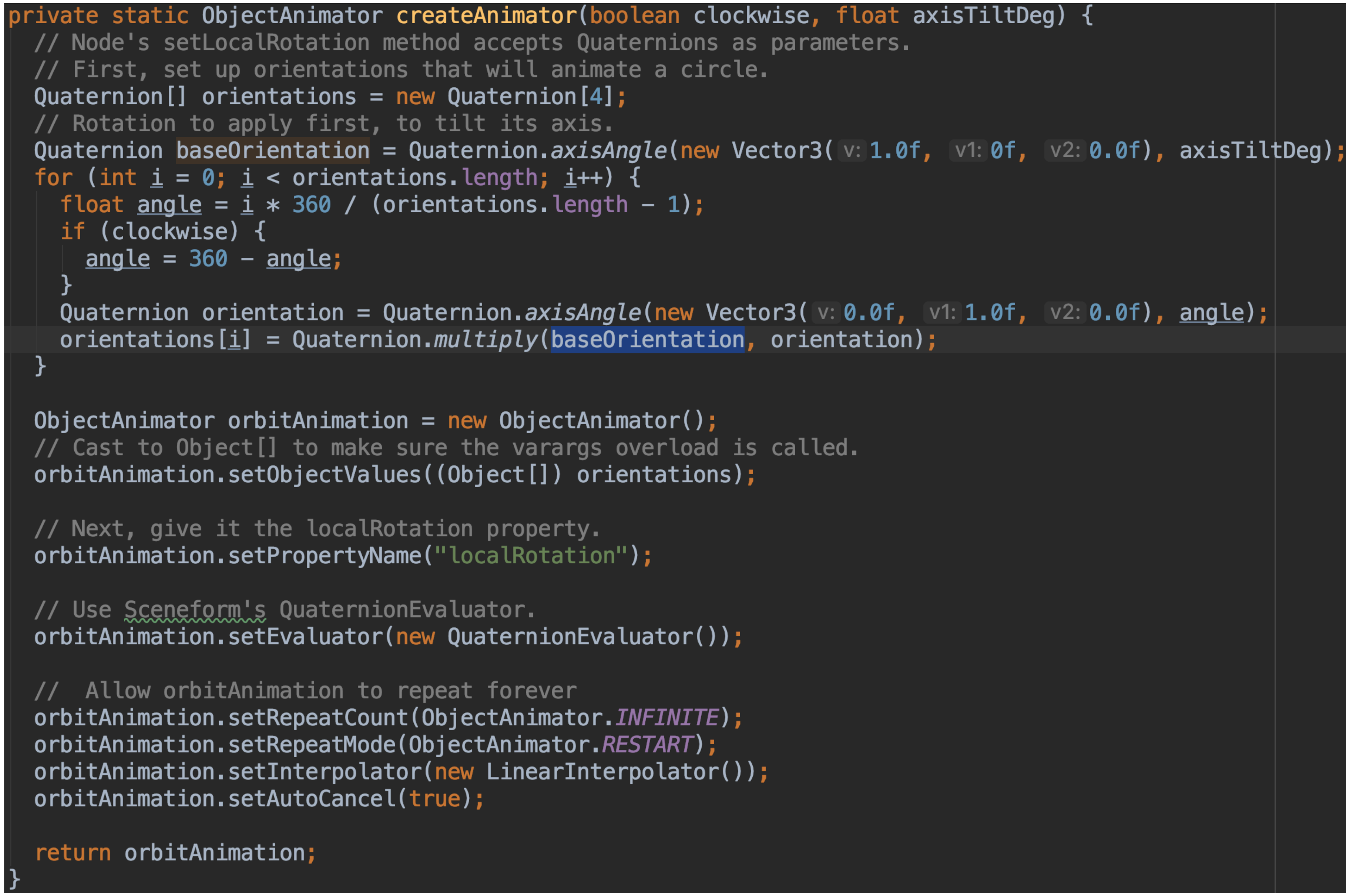
✓ Android里自身的属性动画概念

✓ 不用引入额外的库文件



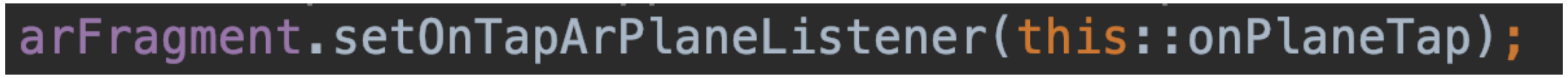
```java
public void onActivate() {
}

public void onDeactivate() {
}

public void onUpdate(FrameTime var1) {
}
```



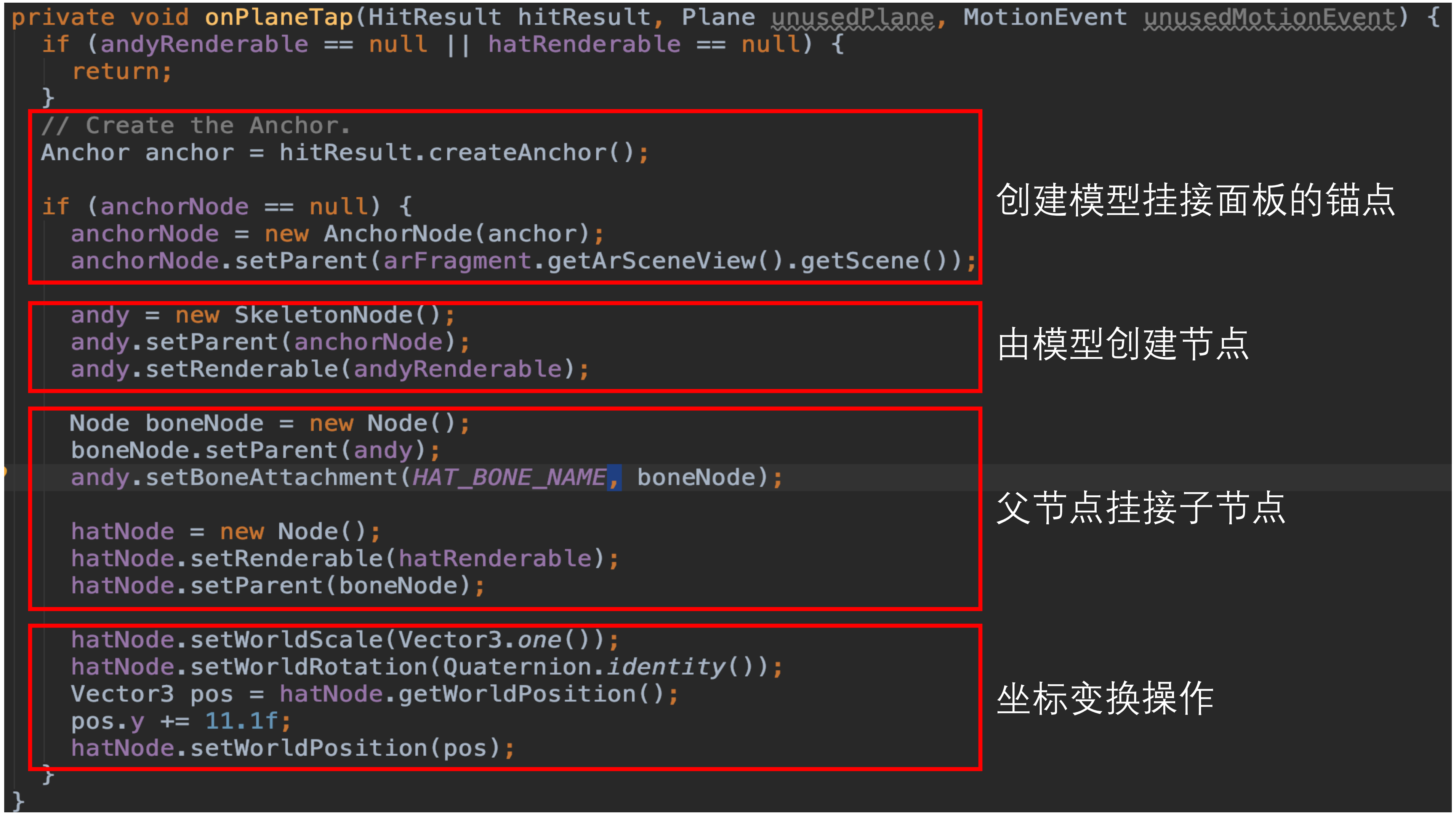


```java
private static ObjectAnimator createAnimator(boolean clockwise, float axisTiltDeg) {
  // Node's setLocalRotation method accepts Quaternions as parameters.
  // First, set up orientations that will animate a circle.
  Quaternion[] orientations = new Quaternion[4];
  // Rotation to apply first, to tilt its axis.
  Quaternion baseOrientation = Quaternion.axisAngle(new Vector3( v: 1.0f,  v1: 0f,  v2: 0.0f), axisTiltDeg);
  for (int i = 0; i < orientations.length; i++) {
    float angle = i * 360 / (orientations.length - 1);
    if (clockwise) {
      angle = 360 - angle;
    }
    Quaternion orientation = Quaternion.axisAngle(new Vector3( v: 0.0f,  v1: 1.0f,  v2: 0.0f), angle);
    orientations[i] = Quaternion.multiply(baseOrientation, orientation);
  }

  ObjectAnimator orbitAnimation = new ObjectAnimator();
  // Cast to Object[] to make sure the varargs overload is called.
  orbitAnimation.setObjectValues((Object[]) orientations);

  // Next, give it the localRotation property.
  orbitAnimation.setPropertyName("localRotation");

  // Use Sceneform's QuaternionEvaluator.
  orbitAnimation.setEvaluator(new QuaternionEvaluator());

  //  Allow orbitAnimation to repeat forever
  orbitAnimation.setRepeatCount(ObjectAnimator.INFINITE);
  orbitAnimation.setRepeatMode(ObjectAnimator.RESTART);
  orbitAnimation.setInterpolator(new LinearInterpolator());
  orbitAnimation.setAutoCancel(true);

  return orbitAnimation;
}
```

● **模型绑定到面板和节点结构**



```
arFragment.setOnTapArPlaneListener(this::onPlaneTap);
```



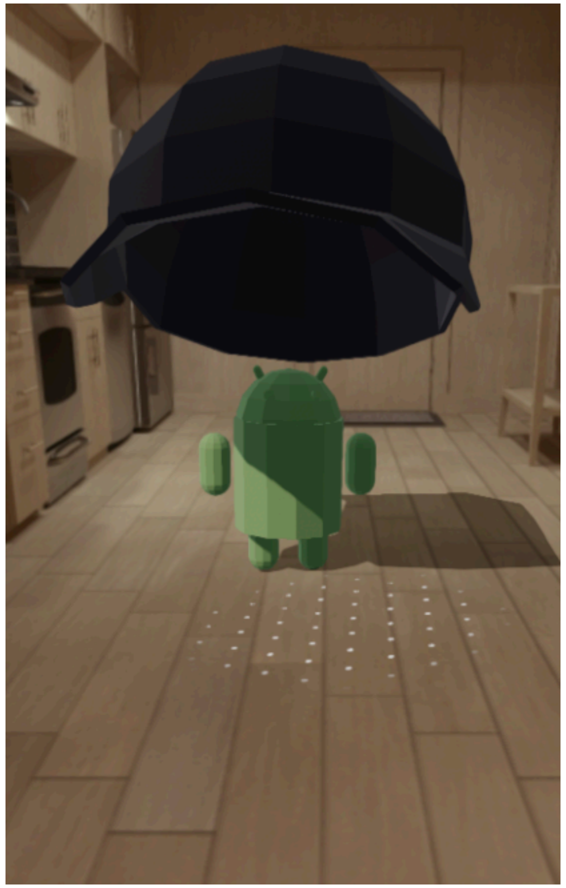
```
private void onPlaneTap(HitResult hitResult, Plane unusedPlane, MotionEvent unusedMotionEvent) {
  if (andyRenderable == null || hatRenderable == null) {
    return;
  }
  // Create the Anchor.
  Anchor anchor = hitResult.createAnchor();

  if (anchorNode == null) {
    anchorNode = new AnchorNode(anchor);
    anchorNode.setParent(arFragment.getArSceneView().getScene());

    andy = new SkeletonNode();
    andy.setParent(anchorNode);
    andy.setRenderable(andyRenderable);

    Node boneNode = new Node();
    boneNode.setParent(andy);
    andy.setBoneAttachment(HAT_BONE_NAME, boneNode);

    hatNode = new Node();
    hatNode.setRenderable(hatRenderable);
    hatNode.setParent(boneNode);

    hatNode.setWorldScale(Vector3.one());
    hatNode.setWorldRotation(Quaternion.identity());
    Vector3 pos = hatNode.getWorldPosition();
    pos.y += 11.1f;
    hatNode.setWorldPosition(pos);
  }
}
```

创建模型挂接面板的锚点

由模型创建节点

父节点挂接子节点

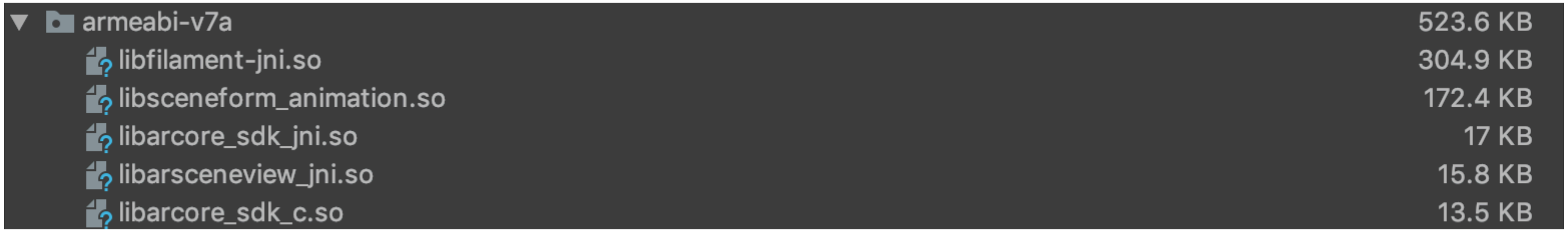
坐标变换操作



支持的设备:

https://developers.google.com/ar/discover/supported-devices

引入的资源大小：



▼ armeabi-v7a	523.6 KB
libfilament-jni.so	304.9 KB
libsceneform_animation.so	172.4 KB
libarcore_sdk_jni.so	17 KB
libarsceneview_jni.so	15.8 KB
libarcore_sdk_c.so	13.5 KB