MIGU 咪咕

基于Web的数智人开发与实践

赵磊

MIGU 咪咕

CONTENTS
目录


01
数智人介绍

数智人介绍

虚拟数字人是指具有数字化外形的虚拟人物。与具备实体的机器人不同，虚拟数字人依赖显示设备存在，通过手机、电脑或者智慧大屏等设备才能显示。数字人具有“人形”、“人神”、“人设”才能成为“数智人”。

现实世界自然人

数智孪生平台

外在形象

形象（人形）

卡通 → 写实 → 超写实

个性化+像本人：依据用户画像提供个性化默配形象，结合PTA等技术生成拟合本人的数智人形象

数字人

交互（人神）

感知 → 决策 → 表达

感知+决策+表达：依托ASR、NLP、TTS等AI技术逐步打造智能驱动、感知决策、智能交互的数智人

数智人

内在属性

人设

数智人 + 画像 = 人设库

通用+共性：真实世界和虚拟世界用户画像、行为等数据汇总分析，建设通用人设库，人设信息与形象、交互素材绑定

孪生/虚拟数智人

虚拟世界数智人

数智人产品Web SDK

轻编H5 SDK

- **适用场景**：基于H5或者小程序，无个性化装扮和捏脸需求，但期望快速集成并展现3D数智人的简易场景

- **功能亮点**：
  - ◆轻量，随时动态更新
  - ◆跨平台（H5 + 小程序）
  - ◆模板丰富，支持运营管理平台按需配置
  - ◆ 提供多模式图片访问接口

- **应用项目**：动感地带、咪咕3D数智人名片、互联网认证SDK、江苏移动掌厅

重编H5 SDK

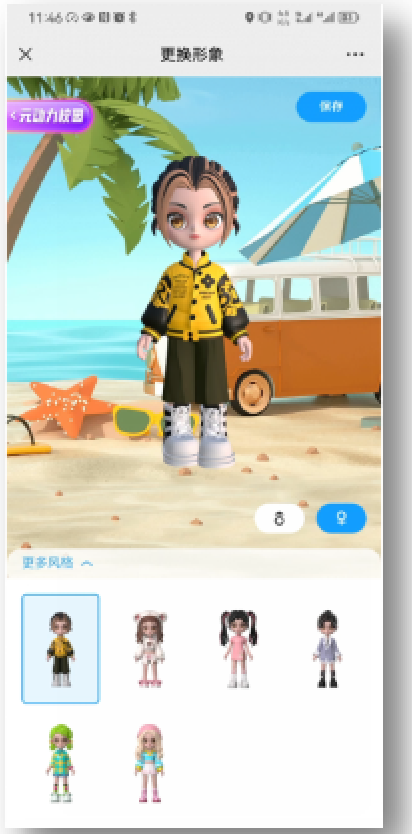
- **适用场景**：在H5或者小程序中对数智人进行更为个性化的形象生成、装扮、捏脸等操作的复杂场景

- **功能亮点**：
  - ◆基于 WebGL的高效渲染引擎
  - ◆支持PTA捏脸换装
  - ◆支持背景切换
  - ◆支持动画状态
  - ◆支持表情包生成

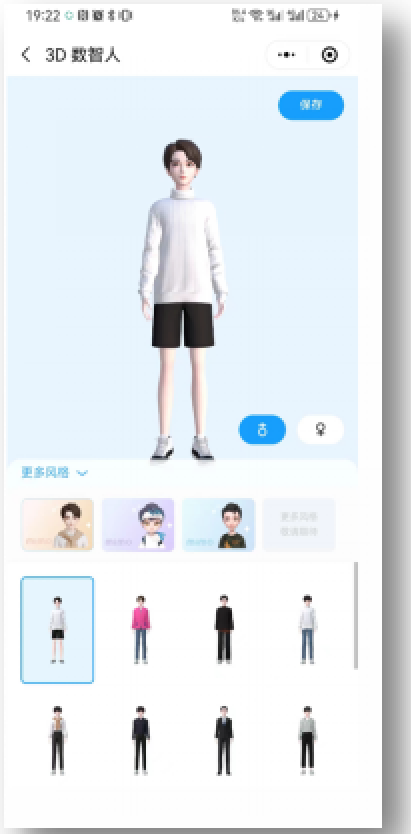
- **应用项目**：江苏移动点亮屏幕H5、数智人大拜年

数智人产品Web SDK

轻编H5 SDK

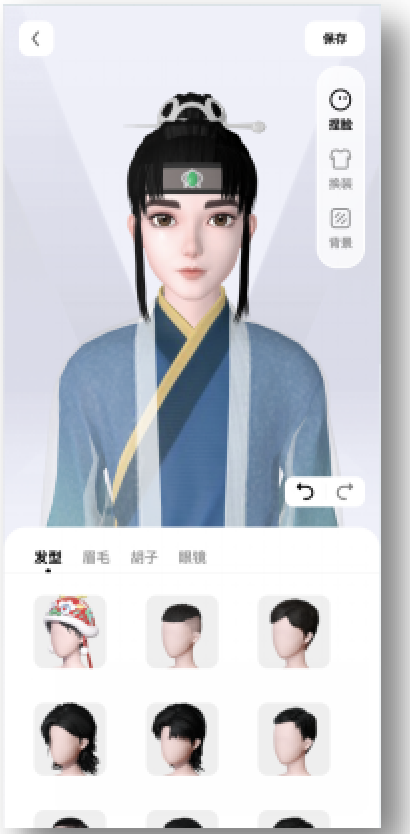


动感地带

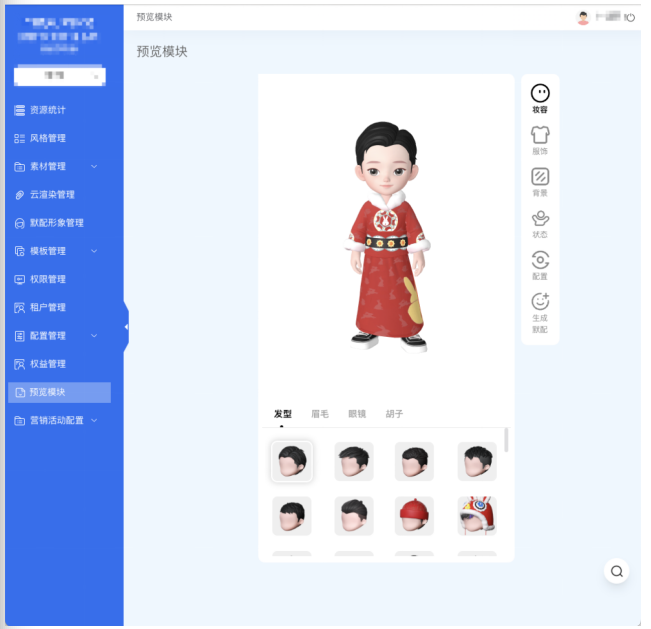


咪咕名片

重编H5 SDK



默认效果

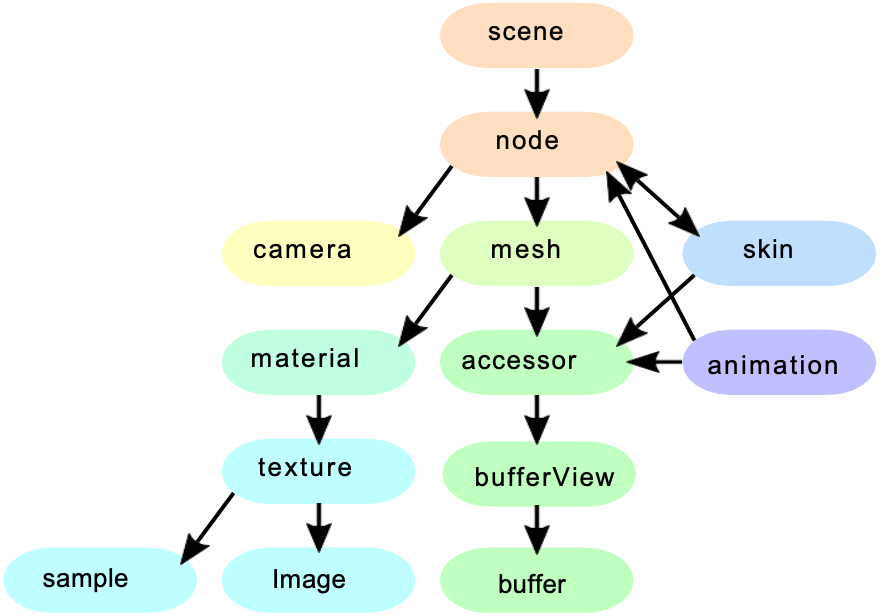


数字人运营配置


02
数智人开发与实践

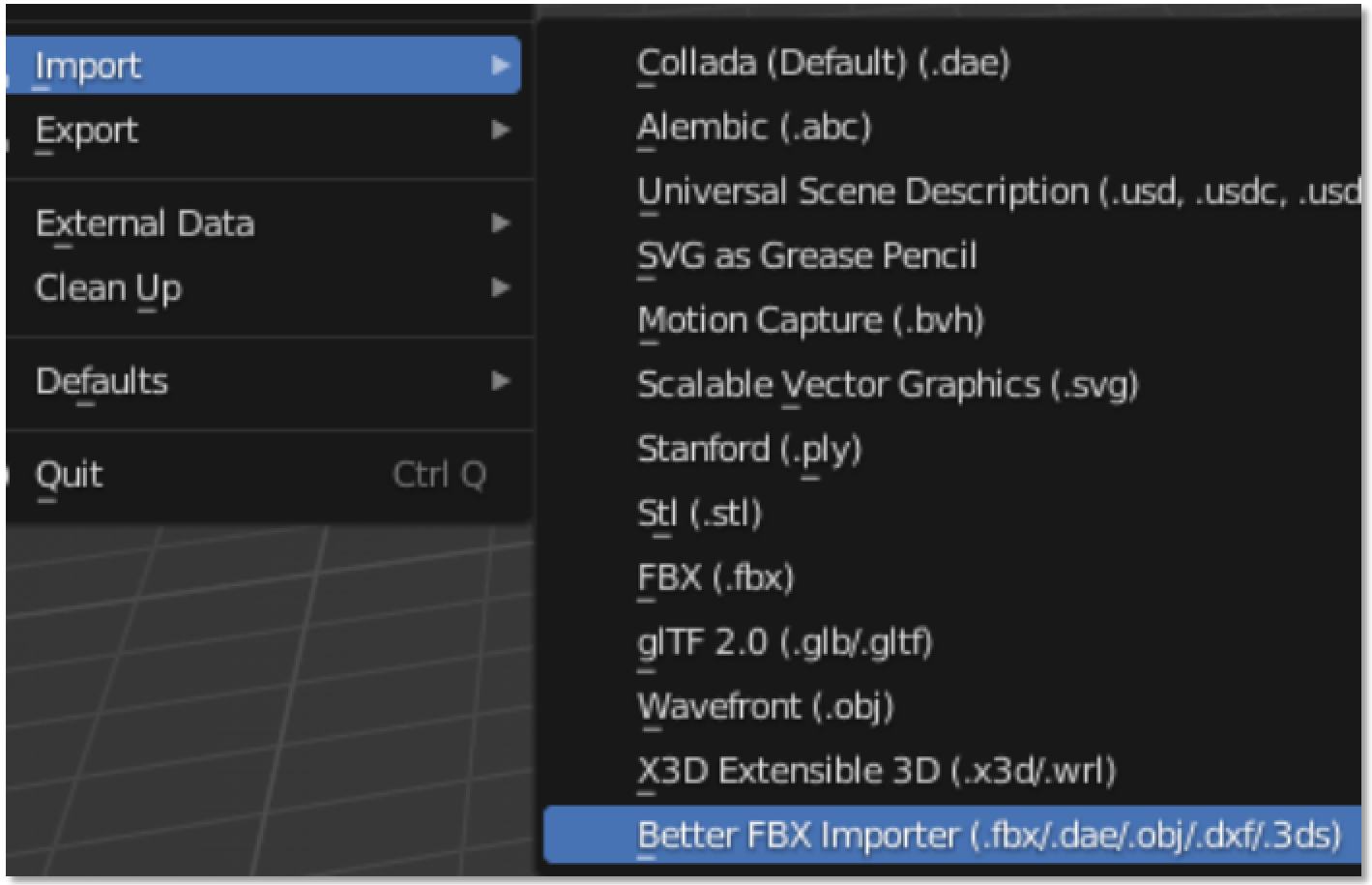
glTF 介绍

- glTF 是一种用于存储和加载3D 场景的标准化文件格式，由科纳斯组织（Khronos） 推出，其目的是为了统一用于应用程序渲染的 3D 格式
- glTF减少了 3D 格式中除了与渲染无关的冗余信息，是最小化 3D 文件资源
- glTF支持 3D 模型几何体、材质、动画及场景、摄影机等信息
- glTF 导出格式有两种后缀格式可供选择：.gltf 和 .glb

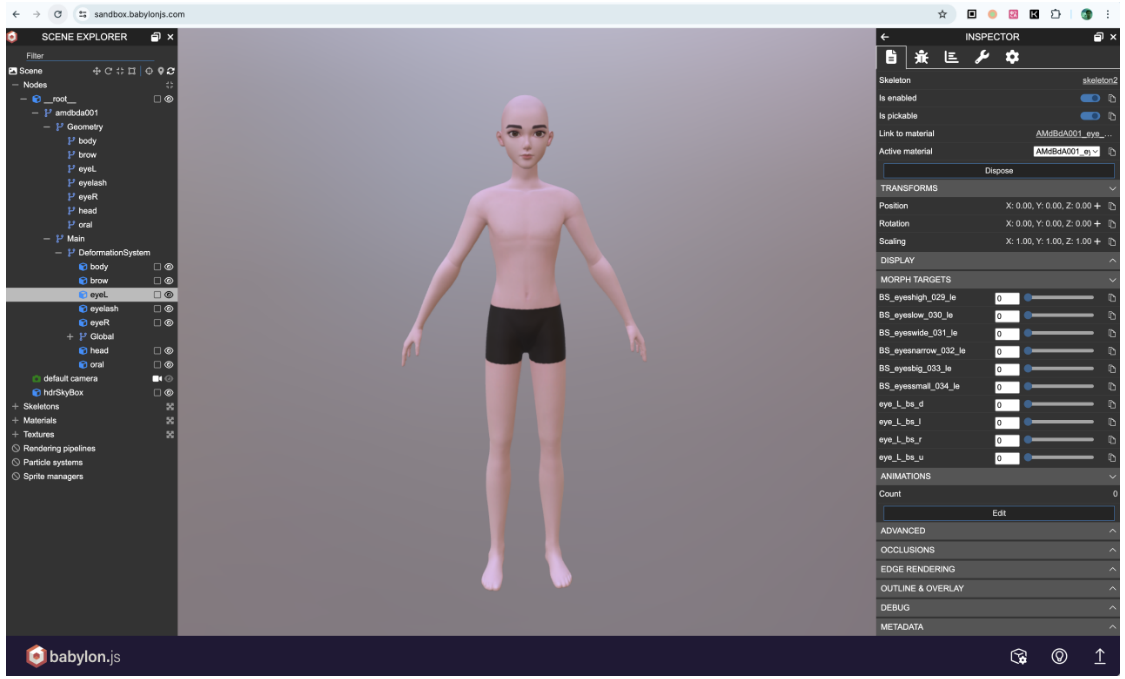


GLB 模型导出

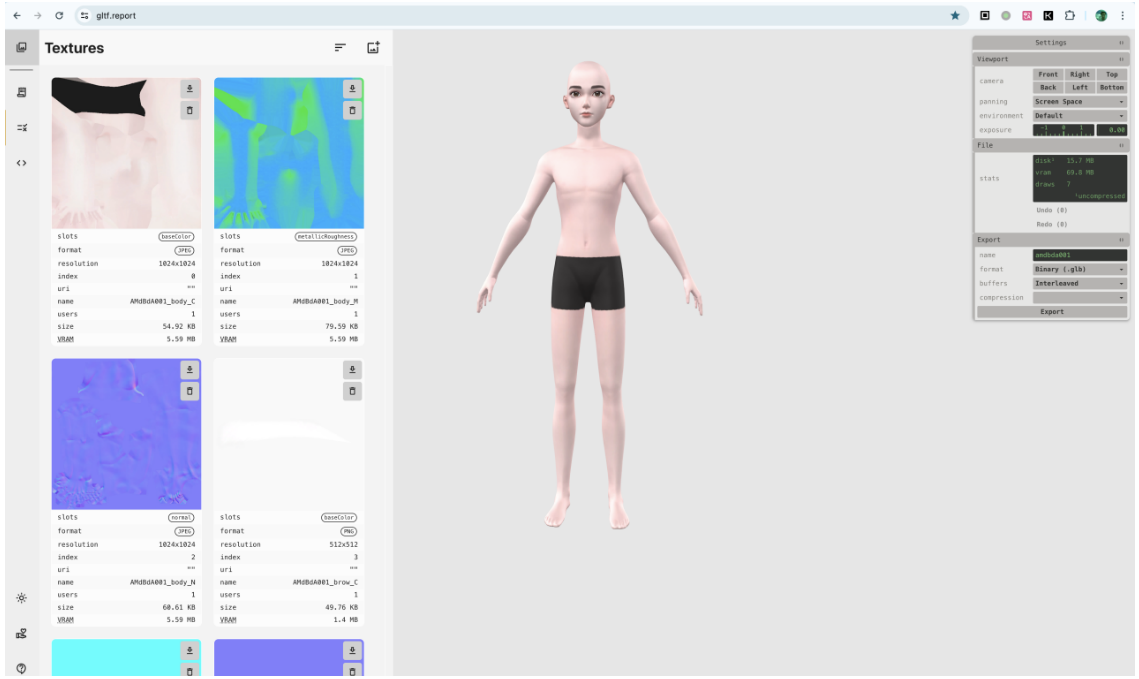
Maya导出的FBX直接导入blender会发生骨骼层级错位（兼容问题），可以通过使用Better FBX解决



GLB 查看工具



Babylon.js SandBox



gltf.report

FBX转glTF

fbx2gltf是一个可以将fbx转为gltf/glb的工具， 可能会存在如下坑：

- 在不同操作系统混用fbx2gltf（例如基膜在Linux上转换，动画在Mac上转换生成），渲染模型可能会出现一些异常
- 针对某些模型，例如鞋子、衣服等模型，转为gltf/glb时，可能会丢失其骨骼信息，需要重新绑定到到基膜上。

```
function skeletonRebind(model: Group) {
    model.traverse((child: SkinnedMesh) => {
        if (child.skeleton) {
            const childBones = child.skeleton.bones
            // 从基膜骨骼中查找对应对应的骨骼信息
            const bones = childBones.map(bone => baseModel.bonesMap.get(bone.name)).filter(bone => bone)
            if (bones.length === childBones.length) {
                const skeleton = new Skeleton(bones, child.skeleton.boneInverses)
                child.skeleton.dispose()
                // 将新增的模型骨骼信息重新绑定到基膜上
                child.bind(skeleton, child.matrixWorld)
            }
        }
    })
}
```

GLB压缩

美术同学导出的GLB文件一般比较大，可以通过@gltf-transform/cli等工具进行压缩

```
~ % gltf-transform resize player.glb  player-new.glb --width 512 --height
512
info: player.glb (11.24 MB) → player-new.glb (5.09 MB)

~ % gltf-transform draco player-new.glb player-new-draco.glb
info: prune: No unused properties found.
info: player-new.glb (5.09 MB) → player-new-draco.glb (4.66 MB)
```

GLB压缩

gltf-transform也提供了API，方便编写脚本对glb进行批量处理

```
const { NodeIO } = require('@gltf-transform/core')
const { weld, draco, DRACO_DEFAULTS, textureResize } = require('@gltf-transform/functions')
const { ALL_EXTENSIONS } = require('@gltf-transform/extensions')
const draco3d = require('draco3dgltf')

const Logger = createLogger()

export default async function compress(options: {
    input: string
    output: string
    textureSize: number
}): Promise<void> {
    const io = new NodeIO()
    io.registerExtensions(ALL_EXTENSIONS).registerDependencies({
        'draco3d.decoder': await draco3d.createDecoderModule(),
        'draco3d.encoder': await draco3d.createEncoderModule()
    })
    const document = await io.read(options.input)

    await document.transform(
        weld(),
        textureResize({
            size: [options.textureSize || 256, options.textureSize || 256]
        }),
        draco(DRACO_DEFAULTS)
    )
    await io.write(options.output, document)

    return Promise.resolve()
}
```

GLB动画提取

Maya等工具导出的动画如果带有 BlendShape 信息，动画文件会很大，包含了很多无用的 Mesh 信息。
一个15Mb的文件，可能动画信息才130kb。

- 方案一： 在THREE.js中，通过GLTFLoader解析后，将AnimationClip里面的动画信息保存为JSON文件，再次加载时，重新生成AnimationClip，然后利用调用AnimationMixer.clipAction进行播放处理。

缺点：生成的JSON文件不通用，只能用于特定渲染引擎，文件依然较大（例如比心动作：450Kb）。

- 方案二： 调用gltf-transform相关API，将原始glb文件的动画提取出来，保存然后保存为glb

优点：通用，紧凑的二进制文件尺寸较小（比心动作：130Kb）。

GLB动画提取

```
const io = new NodeIO()
const gltfDoc = await io.read(options.input)
const docRoot = gltfDoc.getRoot()

const position = gltfDoc
    .createAccessor('position')
    .setArray(new Float32Array([0, 0, 0]))
    .setType(Accessor.Type.VEC3)
```

创建空顶点

GLB动画提取

```
// 遍历处理Mesh
docRoot.listMeshes().forEach((mesh) => {
    const extras = mesh.getExtras()
    // 检测是Mesh是否存在权重
    if (mesh.getWeights().length) {
        mesh.listPrimitives().forEach((p) => {
            p.dispose()
            const np = gltfDoc.createPrimitive()
            np.setAttribute('POSITION', position)
            np.setMode(Primitive.Mode.POINTS)
            mesh.addPrimitive(np)

            const indices = gltfDoc
                .createAccessor('indices')
                .setArray(new Uint16Array([0]))
                .setType(Accessor.Type.SCALAR)
                .setBuffer(docRoot.listBuffers()[0])
            np.setIndices(indices)

            const targetNames = (extras.targetNames as string[]) || mesh.getWeights()

            // 保留BlendShape变形信息
            targetNames.forEach((v, idx) => {
                const target = gltfDoc.createPrimitiveTarget()
                target.setName(v || `${idx}`)
                target.setAttribute('POSITION', position)
                np.addTarget(target)
            })
        })
        mesh.setWeights(mesh.getWeights())
    } else {
        // 没有权重信息的Mesh可以直接删除
        mesh.listPrimitives().forEach((p) => {
            p.dispose()
        })
        mesh.dispose()
    }
})
```

处理Mesh

GLB动画提取

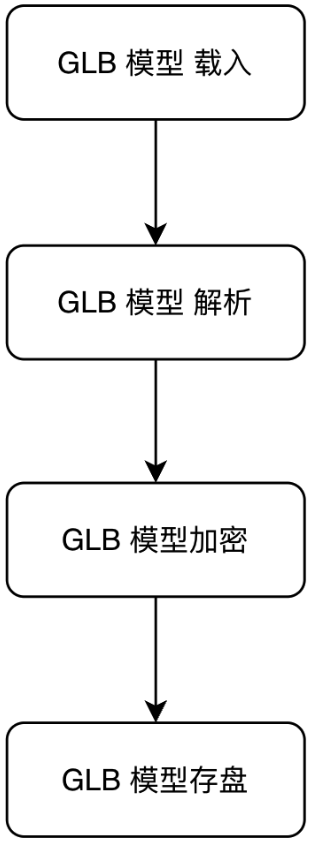
```
docRoot.listMaterials().forEach((mat) => mat.dispose())
docRoot.listTextures().forEach((tex) => tex.dispose())
docRoot.listSkins().forEach((skin) => skin.dispose())

await gltfDoc.transform(
    prune({
        keepLeaves: true
    })
)
await io.write(options.output, gltfDoc)
```

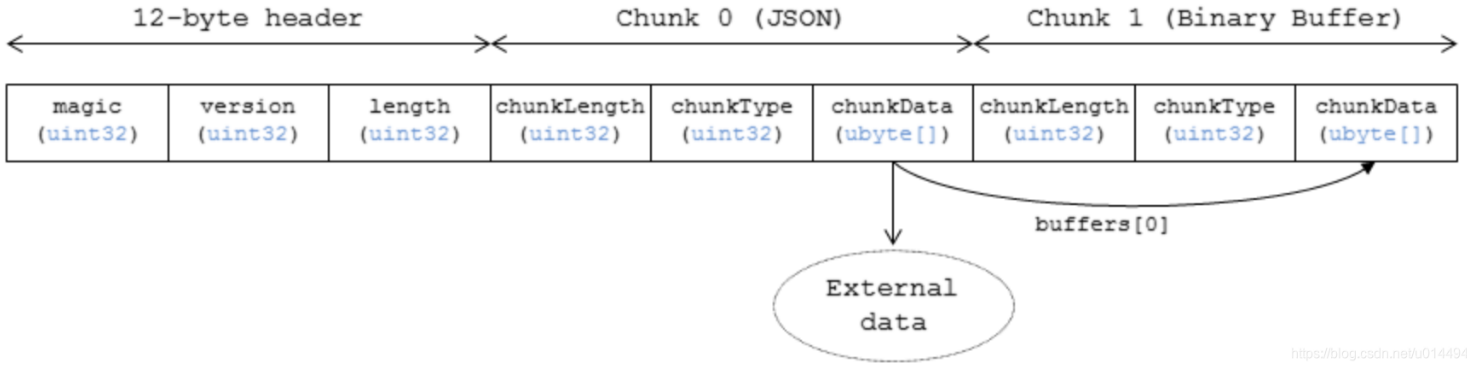
删除无用材质、纹理和蒙皮

GLB加密

目的： 保护3D 素材，加密并溯源。保证非授权用户无法使用加密后的3D 模型



GLB 文件加密流程



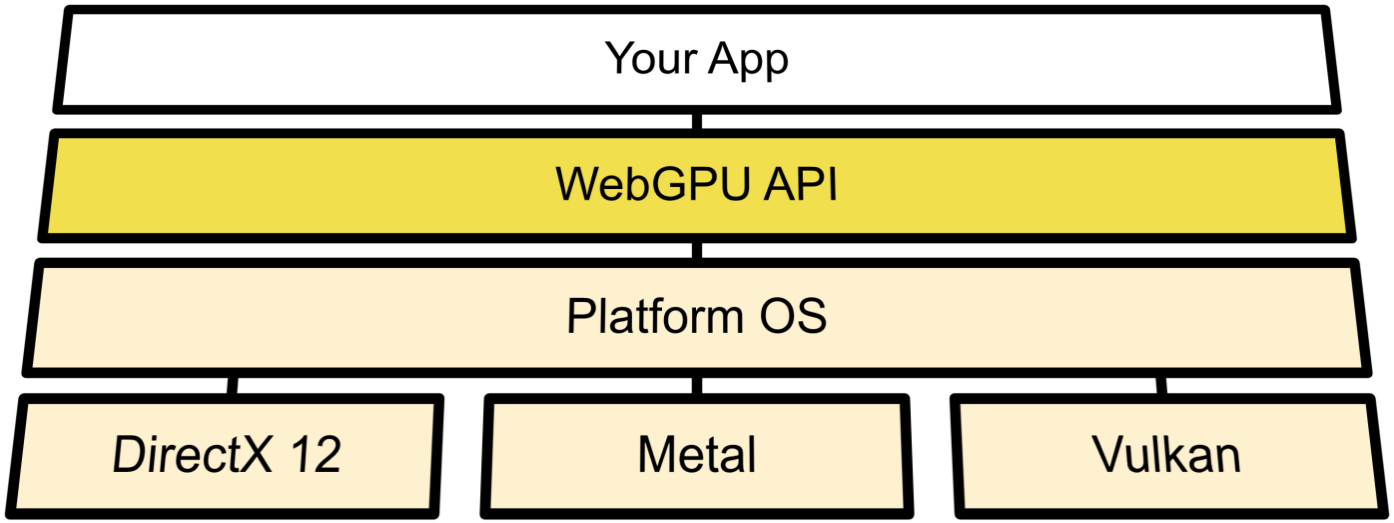
其他

- 3D 模型模型过大时容易导致手机浏览器闪退，可以通过 gltf-transform 来压缩模型，模型纹理贴图建议为512x512，可以根据需要使用 draco / meshopt / quantize等压缩
- 在销毁模型时， 需要手动删除其对应的 material 、texture 、 geometry，解决内存泄露问题。
- 3D 模型的动画需和3D 模型匹配。如果动画文件里面BS 动画包含的 BS系数数量和基模不一致时，需要进行进行填充操作，即保证动画文件里面BS的数量和基模的 BS 数量一致，否则会出现渲染异常。
- 3D 模型渲染时可以考虑开启GPU 高性能模式（powerPreference设置为high-performance），可以解决模型渲染变黑问题；同时可以关闭 shadow 阴影 ，这样提搞渲染帧率
- 当前3D模型文件比较大, 要最好缓存，比如使用 IndexDB 来缓存，这样可以减少网络请求，提升使用体验 。
- Nginx端建议开启对3D 模型进行 GZIP 压缩，这样可以进一步减少模型大小

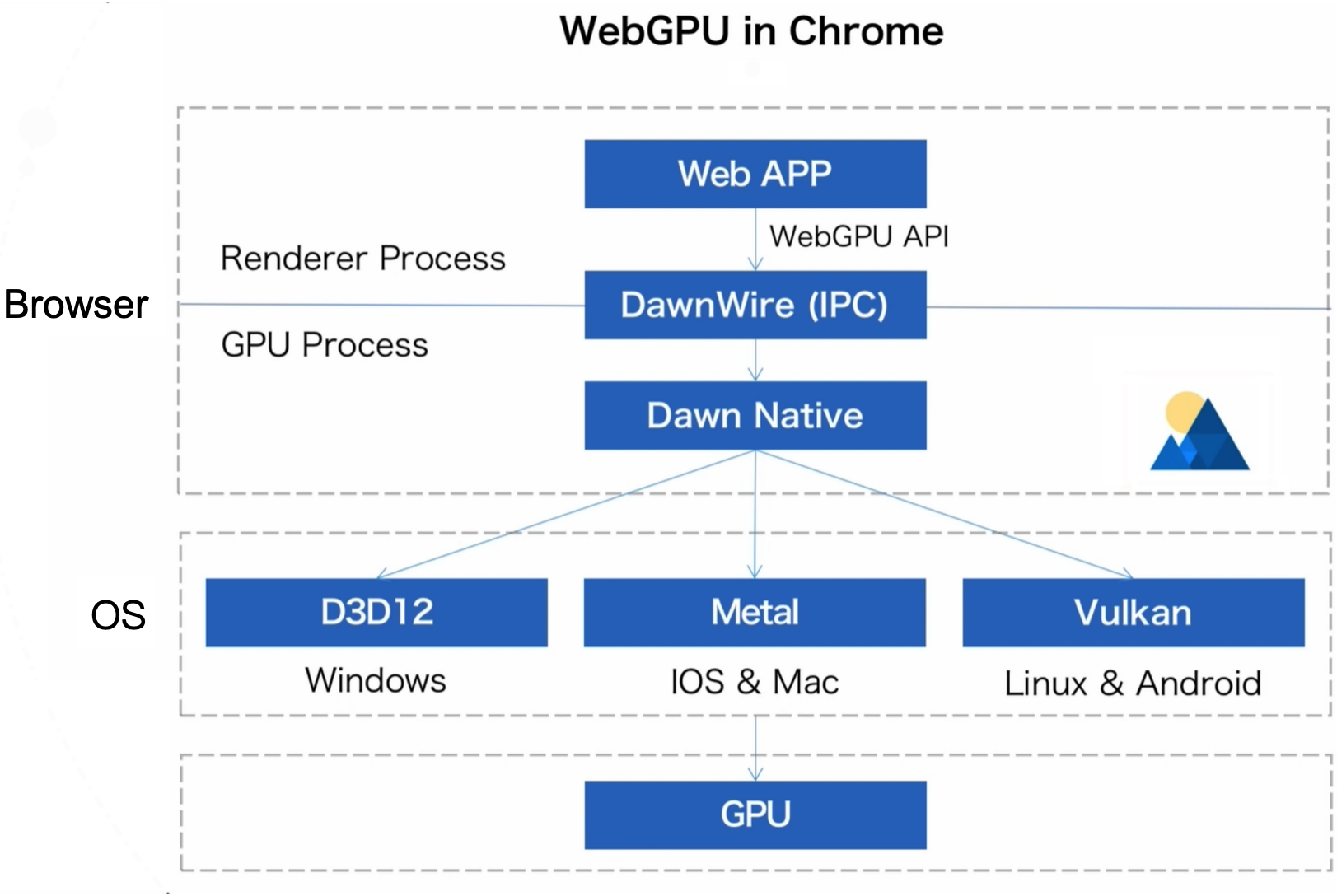

03
关于数智人与W3C的思考

WebGPU

- WebGPU 是新一代 Graphics API for Web JavaScript，用于 GPU 计算和绘制
- W3C GPU 工作组统一标准，全平台支持
- 封装了现代图形API（DX12、Vulkan、Metal），为 Web释放了更多的GPU 硬件的功能
- 更强、更快、更灵活、更现代



WebGPU


WebGPU in Chrome
Browser
Renderer Process
GPU Process
Web APP
WebGPU API
DawnWire (IPC)
Dawn Native
OS
D3D12
Metal
Vulkan
Windows
IOS & Mac
Linux & Android
GPU

WebGPU

MIGU 咪咕

WebGPU - WD

An API for complex rendering and compute, using hardware acceleration. Use cases include demanding 3D games and acceleration of scientific calculations. Meant to supersede WebGL.

Usage % of all users ?

Global 70.51%

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE
4-79	12-79				
3 80-93	3 80-93	3.1-11		10-72	
3 4 94-112	3 4 94-112	2 11.1-14.1	2-62	3 73-98	
5 113-124	113-124	15-17.4	1 63-125	5 99-108	6-10
5 125	125	17.5	1 126	5 109	11
5 126-128		17.6	1 127-129		
		6 TP			

Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
	3.2-17.4	4-23		12-12.1		2.1-4.4.4				2.5
124	17.5	24	all	5 80	15.5	124	1 125	14.9	13.52	1 3.1
	17.6									

Notes | Test on a real browser | Known issues (0) | Resources (5) | Feedback

All major browser engines are working on implementing this spec.

1 Can be enabled in Firefox with the `dom.webgpu.enabled` flag.

2 Can be enabled in Safari on MacOS with the `WebGPU` experimental feature.

3 Can be enabled in some Chromium browsers (on Windows, MacOS, Linux) with the `enable-unsafe-webgpu` flag.

4 Part of an origin trial.

5 Not enabled on Linux by default.

6 Can be enabled via the Develop menu.

未来

- WebGPU 是3D渲染的未来，可以大幅度提高渲染性能。一些AI框架（例如ONNX Runtime）后端已开始使用WebGPU来加速

- 当前WebGPU在移动设备上兼容性还不足，需要社区一起共同推动

- 生成式AI和渲染的结合？（更智能的GPU调度，调度性能核，效率核等）

Thanks

MIGU 咪咕

北京市石景山区石景山路68号金安科幻广场9号楼