ICS 03.80.30
CCS A 12

T/GDEIIA

团　体　标　准

T/GDEIIA XXXX—2023

元宇宙 3D 图形渲染框架技术规范

Metaverse specification for meta-cosmic 3D graphics rendering framework

（征求意见稿）

2023－XX－XX 发布　　2023－XX－XX 实施

广东省电子信息行业协会　　发布

目　　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

本文件的内容不涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由圣名科技(广州)有限责任公司提出。

本文件由广东省电子信息行业协会归口。

本文件起草单位：圣名科技(广州)有限责任公司、汉威广园(广州)机械设备有限公司、中科云宇宙(北京)科技有限公司、广东省科学院智能制造研究所、湖南瑞菱科技有限公司、华南理工大学、日照广亚机电设备有限公司、航天云网数据研究院(广东)有限公司、日照钢铁控股集团有限公司、安徽智寰科技有限公司、烟台大学。

本文件主要起草人：贺圣茗、陈磊、成西平、夏时谦、袁君奇、李伟光、李平、邓志文、崔舒强、梁耀、车爽、陈启愉、陈浩、王宇、南博文、陈金伟、迟万军、牟健慧、司占强、翟中平、张海滨、徐晖、李龙云、陈惠朗、穆允全、党立国、潘春雷、孟安、丁朋。

本文件为首次发布。

元宇宙 3D 图形渲染框架技术规范

1 范围

本文件规定了元宇宙3D图形渲染的系统架构、逻辑流程、数据管理、技术语言、数据格式、接口管理等。

本文件适用于指导进行元宇宙3D图形渲染的构建。

2 规范性引用文件

本文件没有规范性引用文件。

3 术语和定义

下列术语和定义适用于本文件。

3.1 元宇宙 Metaverse

元宇宙是人类运用数字技术构建的，由现实世界映射或超越现实世界，可与现实世界交互的虚拟世界，具备新型社会体系的数字生活空间。

3.2 几何模型 Geometric model

用几何概念描述物理或者数学物体形状。

3.3 3D 图形 Three Dimensional

3d 是 three-dimensional 的缩写，也就是三维图形。在计算机里显示 3d 图形，指在平面里显示三维图形。

3.4 渲染 Render

渲染在电脑绘图中是指用软件从模型生成图像的过程。模型是用严格定义的语言或者数据结构对于三维物体的描述，它包括几何、视点、纹理以及照明信息。

3.5 3D 渲染 3d rendering

3D 渲染是计算机从 3D 场景(多边形，材质和光照)中获取原始信息并计算最终结果的过程。输出通常是单个图像或一起渲染和编译的一系列图像。渲染通常是 3D 创建过程的最后阶段，但例外是是否将渲染带入 Photoshop 进行后处理。如果要渲染动画，它将被导出为视频文件或一系列图像，以后可以将它们缝合在一起。

3.6 业务模型 Business model

根据现场实际业务过程而建立的模型。

3.7 数字孪生 Digital Twins

由物理资产、虚拟镜像和用户界面组成的混合模型。
[来源：ISO/TR24464:2020,3.1.4]

3.8 业务逻辑 Service Logic

根据元宇宙交互内容，通过代码或脚本实现的交互功能。

3.9 网络通信 Communications Network

通信网络是指将各个孤立的设备进行物理连接，实现人与人，人与计算机，计算机与计算机之间进行信息交换的链路，从而达到资源共享和通信的目的。它是网络数据交互和获取的途径之一。

3.10 场景 Scene

场景是每个项目里面放置内容的容器，可以将已创建的 3D 图形加入到场景中，支持在场景内放置模型，灯光和照相机。也可以通过调整场景的位置，让场景内的所有内容都一起跟着调整位置。

3.11 照相机 Camera

根据投影方式的不同，照相机又分为正交投影照相机与透视投影照相机。使用透视投影照相机获得的结果是类似人眼看到的有“近大远小”的效果；而使用正交投影照相机获得的结果就像平面画3D的效果，在三维空间内平行的线，投影到二维空间中也一定是平行的。

4 系统架构与基本要求

4.1 系统架构

元宇宙 3D 图形渲染框架，包括创建几何模型、加载几何模型、读取几何模型、应用到场景四部分内容。

a）创建几何模型，准备模型数据以及它们的纹理和材质信息，模型的创建是由依次点——线——面——体最终形成几何模型。

b）加载几何模型，可以加载无材质模型，也可以加载有材质模型，通过一个个顶点加载进来生成三角面。

c）读取几何模型，又叫解析几何模型，获取到顶点及三角面，通过一系列算法解析，最终成为可被读取的数据。

d）应用到场景，将可读取的几何模型数据，加载渲染到可查看的画布上面，最终应用到场景，成为我们想要的效果。

4.2 逻辑流程

元宇宙 3D 图形渲染框架技术逻辑流程参考图 1。

通过创建场景到配置场景的灯光、相机，可以使用 web 和 PC 端两种方式渲染模型，分别是加载已经创建好的几何模型文件，或者自己使用代码创建几何模型，可以加载模型的材质数据，也可以更改材质数据，从而加入到刚才创建的场景里去，最后创建渲染器，把元宇宙 3D 模型渲染出来。

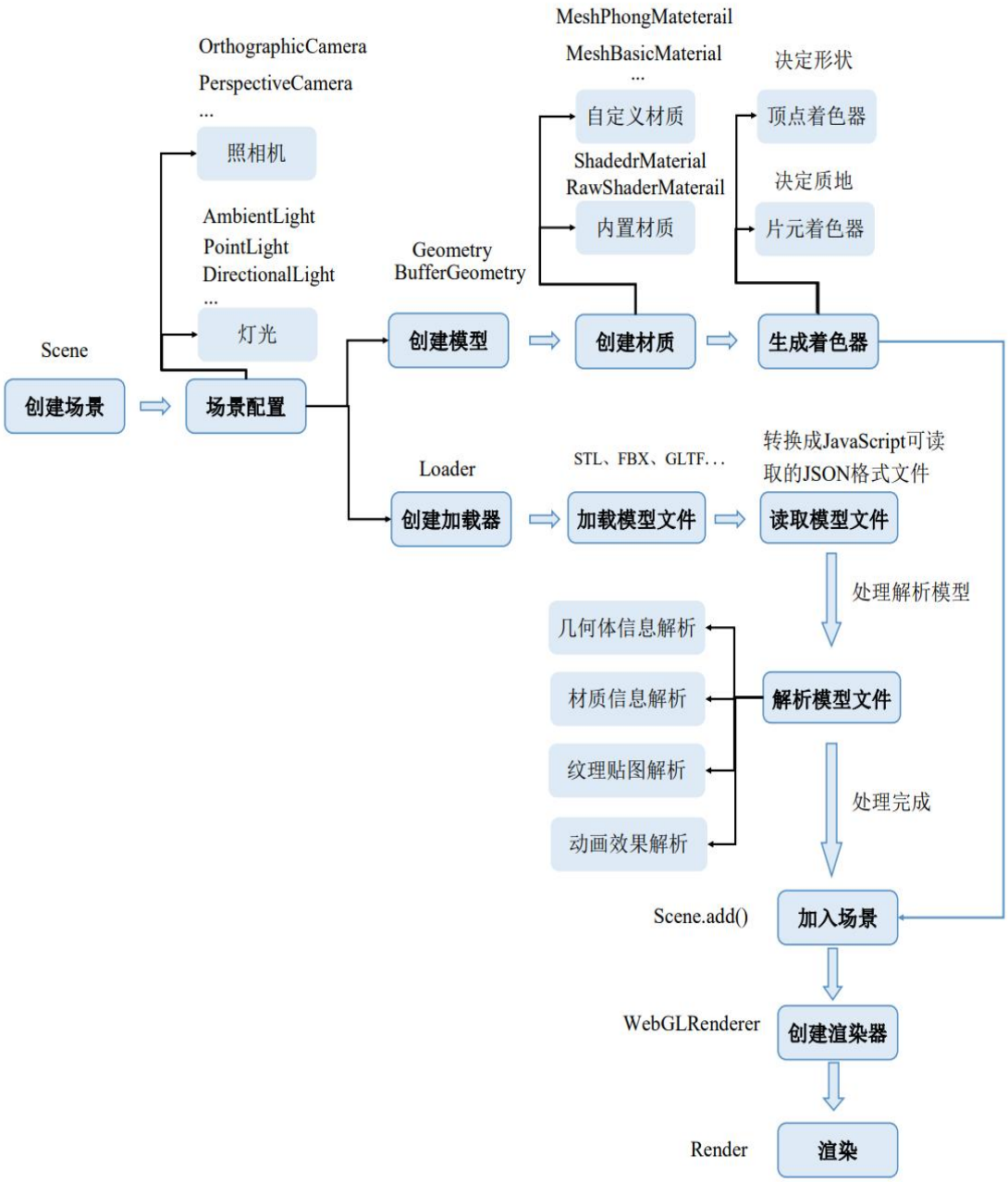

图 1 元宇宙3D图形渲染逻辑架构图（web）

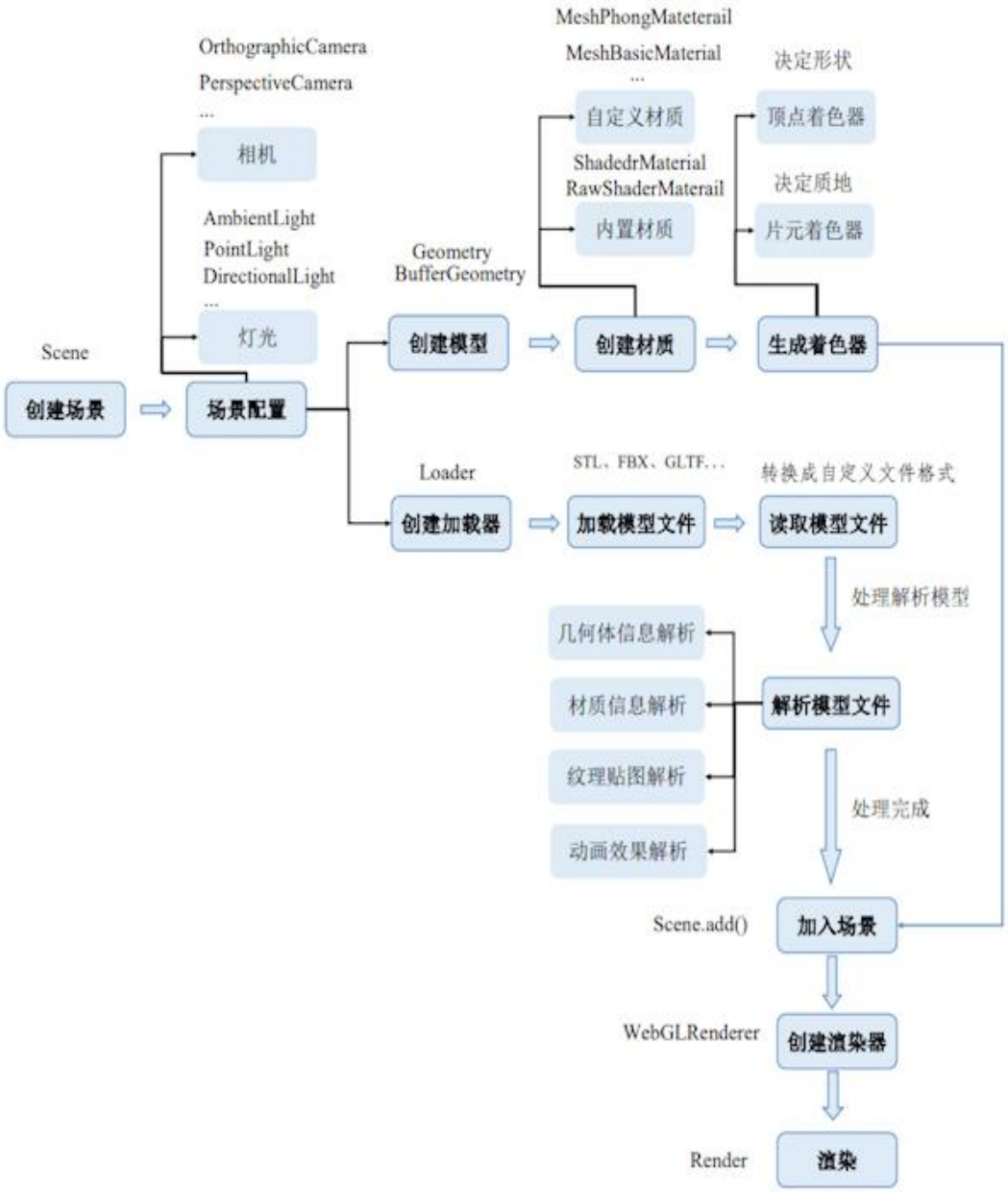

图 2 元宇宙 3D 图形渲染逻辑架构图（PC）

4.3 技术语言

4.3.1 WEB 端使用语言

元宇宙 3D 图形渲染框架技术语言有 JavaScript 语言,使用 3dengin.js 库实现渲染。

4.3.2 PC 端使用语言

PC 端使用的渲染引擎可以使用 UnrealEngine4/5(虚幻引擎)或者 unity 引擎进行渲染，代码语言可以使用 C++或者使用 C#，蓝图等脚本语言。

4.4 数据格式

目前主流的模型格式有 FBX,OBJ,glb/gltf。

Fbx：是目前 3D 工具主流的模型格式之一。fbx 里面包含了动画，材质，贴图，骨骼，灯光，摄像机等数据。但是它的格式是不公开的，所以需要使用 Autodesk 提供的 SDK 进行格式的读写修改等。

Obj：是主流的模型格式之一，它的特点是直接存储文本形式的顶点，面片，法向量等信息，纹理信息是直接存储在外部链接中的。

Glb：是一种开源的模型格式文件，它是开源的，存储格式是二进制文件，其中 glt 是 glb 的文本存储形式。其解析方法可以使用 gltf 封装的 SDK 进行解析读取。

4.5 接口管理

4.5.1 web 端接口管理作为集成平台即服务工具，可帮助用户获取 3D 图形渲染库数据。

(1) 场景

场景是放置物体、灯光和摄像机的地方

new dengin.Scene()

(2) 相机

相机就类似我们的人眼，可以观察到环境中的物体，以及光线效果等

new dengin.OrthographicCamera()

new dengin.PerspectiveCamera()

(3) 光源

场景中默认都是没有光的，就像一个没有窗户的空间，需要增加光源才能看到物体

new dengin.AmbientLight()

new dengin.DirectionalLight()

new dengin.HemisphereLight()

new dengin.PointLight()

new dengin.RectAreaLight()

new dengin.SpotLight()

(4) 渲染器

用于渲染已经配置好的场景，呈现到屏幕上观看

new dengin.WebGLRenderer()

(5) 几何体

场景中的物体

new dengin.BufferGeometry()

(6) 材质

场景中的物体的皮肤

new dengin.Material()

(7) 加载器

导入外部模型的一种方法库

new dengin.TextureLoader()

new dengin.CubeTextureLoader()

new dengin.ImageLoader()

new dengin.ObjectLoader()

new dengin.MaterialLoader()

new dengin.GLTFLoader()

new dengin.FBXLoader()

4.5.2 作为一个跨平台渲染框架，我们可以使用宏定义去区分不同的平台，但是保持同样的对外接口,在C++中接入 Web 端渲染功能。

```
#define WIN64    64 位 PC(系统宏)
#define WIN32    32 位 PC(系统宏)
#define PLATFORM_WIN64 自定义 Windows 64 位宏
#define PLATFORM_WIN32 自定义 Windows 32 位宏
#define PLATFORM_WEB    自定义 Web 平台端
```

(1) 场景

场景是放置物体、灯光和摄像机的地方

new dengin.Scene()

(2) 相机

相机就类似我们的人眼，可以观察到环境中的物体，以及光线效果等

new dengin.OrthographicCamera()

new dengin.PerspectiveCamera()

(3) 光源

场景中默认都是没有光的，就像一个没有窗户的空间，需要增加光源才能看到物体

new dengin.AmbientLight()

new dengin.DirectionalLight()

new dengin.HemisphereLight()

new dengin.PointLight()

new dengin.RectAreaLight()

new dengin.SpotLight()

(4) 渲染器

用于渲染已经配置好的场景，呈现到屏幕上观看

new dengin.WebGLRenderer()

(5) 几何体

场景中的物体

new dengin.BufferGeometry()

(6) 材质

场景中的物体的皮肤

new dengin.Material()

(7) 加载器

导入外部模型的一种方法库

new dengin.TextureLoader()

```
new dengin.CubeTextureLoader()
new dengin.ImageLoader()
new dengin.ObjectLoader()
new dengin.MaterialLoader()
new dengin.GLTFLoader()
new dengin.FBXLoader()
```

5 测试与分级

单元测试：单元测试是在程序开发的最初阶段进行的测试，主要针对程序的基本单元（如函数、方法等）进行测试，以保证程序的基本功能正确。

集成测试：集成测试是针对整个程序进行的测试，测试不同功能模块之间的集成和协作，以确保整个程序的功能和性能符合要求。

程序测试：程序测试是在程序开发的后期进行的测试，主要测试程序的功能、性能、安全等方面，以确保程序达到预期的要求和标准。

验收测试：验收测试是在程序开发完成后进行的测试，由用户或客户进行，主要测试程序是否符合用户需求，以及是否能够在用户环境中正常运行。

安全测试：安全测试是针对程序的安全性进行的测试，包括漏洞扫描、渗透测试、代码审查等多种测试方法，以确保程序的安全性能满足等级要求。

性能测试：性能测试是针对程序的性能进行的测试，包括负载测试、压力测试、并发测试等多种测试方法，以确保程序的性能满足等级要求。

附 录 A
（规范性）
三维场景渲染技术（又称 3D 渲染技术）方法

A.1 三维场景渲染技术（又称 3D 渲染技术）方法

三维场景渲染是计算机从三维场景内获取模型、材质和光照等基本信息并通过复杂计算输出真实感高图像的过程。三维场景渲染是三维图形平台内最重要的模块之一，主要负责对图形的组织、管理与显示，实际上是三维真实感图形的再现过程，具体表现在光线处理、纹理处理和图形显示处理。3D 渲染技术方法，主要包括以下几种：

A.1.1 扫描线渲染技术

扫描线技术可有效减少渲染时间，它在 3D 建模中基于逐个多边形提供实时渲染，而不是逐个像素进行渲染。通过这种技术，与预计算的照明一起使用时，可以达到每秒 60 帧（fps）的速度。

A.1.2 光线追踪渲染技术

光线追踪是一种渲染 3D 建模的方法。它通过 3D 场景中材料的反射或折射使用精确的“反射”来跟踪自然光。它可通过算法计算出每种颜色，并实现比扫描线更高的真实感。

A.1.3 射线投射渲染技术

当处理不需要显示许多细节的项目，则射线投射技术是最佳选择。它使用对象的几何图形逐行和逐像素渲染项目。它的主要用途涉及 3D 建模的实时仿真。要在计算阶段获得更好的性能以获得最佳结果，需要付出很多努力。

在实践中，通常基于每种技术的优点来组合光线跟踪和光线投射技术，以实现尽可能高的照片写实度。掌握这些技术并知道何时使用它们将有助于获得最佳结果，并有助于避免花费不必要的长时间来完成渲染。