Computer Science and Application 计算机科学与应用, 2025, 15(9), 139-151
Published Online September 2025 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2025.159232

Hans汉斯

基于D3D12的低轨卫星星座可视化建模仿真研究

闫雪飞，王　良，李　孟

中国人民解放军32039部队，北京

收稿日期：2025年8月3日；录用日期：2025年9月5日；发布日期：2025年9月15日

摘　要

目前，低轨卫星星座在国际社会呈现蓬勃发展态势，并成为热点研究话题。低轨卫星星座的可视化建模仿真是开展低轨卫星星座的轨道构型、星间链路、路由规划、资源调度等研究的基础，现有的仿真软件还存在扩展性差、版权受限等不足。鉴于D3D12 (Direct3D12)可视化技术相比OpenGL具有更优的渲染性能，本文采用D3D12技术对低轨卫星星座进行了可视化建模仿真研究，设计了系统架构，实现了二体问题以及卫星终端可见性求解，并给出了地球、卫星、轨道、波束、星间链路以及经纬度的二三维渲染方法，最后基于UWP (Windows通用应用平台)框架进行了编程实现，通过对典型卫星星座仿真并与STK (Satellite Tool Kit)仿真结果对比验证了本文方法的可行性和有效性。

关键词

低轨卫星星座，可视化，仿真，D3D

Visualization and Simulation Study of the LEO Constellation Based on D3D12

Xuefei Yan, Liang Wang, Meng Li

Unit 32039, PLA, Beijing

Received: Aug. 3rd, 2025; accepted: Sep. 5th, 2025; published: Sep. 15th, 2025

Abstract

The development of LEO Constellation is taking a flourishing state at present, and related study becomes the hot topic. Visualization and simulation study of the LEO Constellation is the foundation to study its configuration, crosslink, route-programming, resource-dispatch, and so on, yet the existing software falls short of the low expansibility and limit copyright etc. considering the drawing

文章引用: 闫雪飞, 王良, 李孟. 基于 D3D12 的低轨卫星星座可视化建模仿真研究[J]. 计算机科学与应用, 2025, 15(9): 139-151. DOI: 10.12677/csa.2025.159232

performance of the D3D12 visualization technique is nice than OpenGL, this paper studied the visualization and simulation of the LEO Constellation with the D3D12 technique, designed the architecture and solved the orbit equation and terminal visibility in two-body form, at the same time provided the 2D/3D drawing methods of the earth, satellite, orbit, beam, crosslink, curve of longitude and Latitude, at last realized these methods based on UWP(universal windows platform) and validated the practicability and effectiveness of these methods through the simulation of the typical LEO constellations and the comparison with STK.

Keywords

LEO Constellation, Visualization, Simulation, D3D

Copyright © 2025 by author(s) and Hans Publishers Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

1. 引言

低轨互联网星座由于具有卫星容量大、覆盖面广、传输时延低、弹性抗毁能力强等优势成为各国争相发展的对象，典型的例如美国的“星链”、英国的“一网”。这其中以“星链”发展最为迅速，其发射数量年均增长约 40%，在轨已超 5800 颗、20 重覆盖，在 70 余个国家投入商业应用，已有超 230 万用户，展现了巨大的发展潜力。尤其是鉴于“星链”星座在俄乌冲突中的军事化应用和出色表现，各国都在积极发展自己的低轨星座系统，进一步加剧了低轨星座发展热潮。受国际太空领域发展影响，近几年我国低轨星座也呈火热发展态势，并涌现了国家卫星互联网、银河航天、上海垣信等众多低轨卫星星座，并且都已经发射多颗在轨试验星，未来随着行业生态的健全和发展，我国低轨卫星星座前景广阔。

由于低轨卫星高速运动，星座拓扑动态变化，需要借助有效的可视化软件辅助观察和研究卫星相对于地面站的运动过程，对于分析低轨卫星星座的运动特性、覆盖特性进而支撑轨道布局和跟踪建链等应用均具有重要的意义。为满足航天领域研究人员的可视化仿真需求，美国 AGI 公司专门推出了一款卫星仿真工具包 STK (Satellite Tool Kit)，可用于卫星星座轨道建模及可视化仿真，并已在领域内得到了广泛应用[1]。文献[2]基于 STK 以及 Matlab 工具对地面站、中继星、用户星、干扰星的星间、星地可见性进行了仿真研究。文献[3]基于 STK 对北斗三号系统的空域覆盖性能和定位精度分别进行仿真分析。文献[4]基于 STK 对双星火箭发射任务进行了可视化研究，并对星箭分离、卫星状态进行了可视化呈现。

STK 虽然功能强大，但是在许多问题研究中也存在一定局限性，例如可扩展性差、价格昂贵、版权受限等[5]，为此文献[5]基于 Opengl 和 WPF 开发了一款新的低轨卫星建模仿真软件，能够对卫星轨道、波束、星下点轨迹进行可视化仿真，运行于 Windows 操作系统。Opengl 是一种跨平台开放式三维图形应用和程序接口，在 Unix、Windows、Mac 等多个平台上均能够提供较为优秀的渲染能力[5]。D3D12 (DirectD 12)是由微软公司推出的适用于 Windows 平台的最新版本的 3D 渲染引擎[6]，相比 Opengl，由于 D3D12 可绕过图形显示接口直接进行各种硬件的底层操作，其渲染性能要更加优异，并常用于大型 3D 游戏的实时渲染。为此，本文尝试采用 D3D12 渲染引擎针对低轨卫星星座进行可视化建模仿真开发，采用 UWP (Windows 通用应用平台)界面框架进行系统设计，基于面向对象编程语言 C++进行实现，以期为低轨卫星星座的可视化建模仿真研究提供借鉴。

2. 系统组成

低轨卫星星座建模仿真系统架构由人机交互界面、轨道运动模型、渲染引擎和图形资源组成，其相

互关系如下图 1 所示。

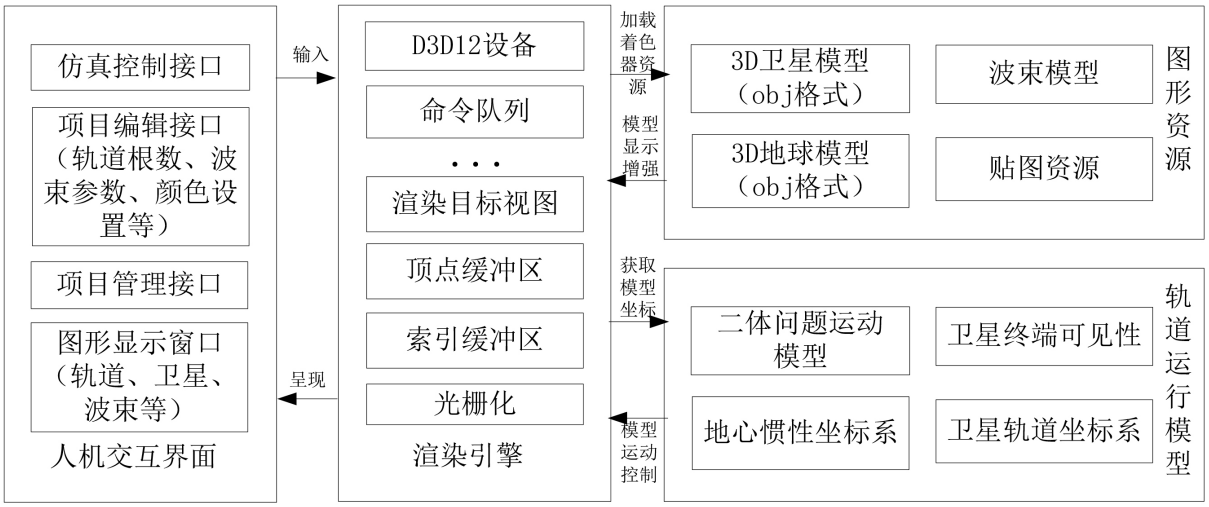

Figure 1. The architecture of the LEO satellite constellation simulation system based on UWP and D3D12
图 1. 基于 UWP 和 D3D12 的低轨卫星星座建模仿真系统架构

2.1. 人机交互界面

人机交互界面由仿真控制接口、项目编辑接口、项目管理接口、图形显示窗口几部分组成。其中，仿真控制接口主要负责控制仿真的开始、暂停和停止，根据需要改变仿真的步长。项目编辑接口主要负责对卫星轨道的参数进行设置，包括轨道六根数以及轨道颜色、波束颜色等，同时支持按照 Walker 构型进行星座的编辑，批量添加卫星。项目管理接口主要负责项目的新建、加载以及存储管理等。图形显示接口主要负责对 3D 场景进行实时呈现。

2.2. 渲染引擎

渲染引擎主要根据系统需要对 3D 模型进行绘制和呈现。针对 D3D12 渲染引擎，相关模块主要包括 D3D12 设备、命令队列、命令列表、渲染目标视图、各类缓冲区资源等。D3D12 渲染流程主要包括 D3D 初始化、创建命令队列和命令列表、创建围栏、创建交换链、创建描述符堆、更新缓冲区资源、渲染、页面翻转、同步等几个流程，如图 2 所示。其中，D3D 初始化主要是创建 D3D 设备，获取提供显卡控制接口的对象 ID3D12Device。命令队列 ID3D12CommandQueue 和命令列表 ID3D12GraphicsCommandList 是 CPU 与 GPU 之间交互的媒介，通过 ID3D12CommandQueue，CPU 可以将绘制命令交给 GPU 进行绘制。围栏 ID3D12Fence 主要是用于保持 CPU 和 GPU 之间的同步，避免资源读写冲突。交换链 IDXGISwapChain 用于前后台缓冲区的交换呈现，当后台缓冲区完成渲染后，通过 IDXGISwapChain 将后台缓冲区翻转为前台缓冲区进行呈现(Present)，而前台缓冲区变为后台缓冲区进行渲染，通过交换链提高渲染效率。描述符堆 ID3D12DescriptorHeap 是存储顶点、索引、纹理等一系列资源描述符 descriptor 的内存区域，GPU 通过 descriptor 调取资源进行绘制。在每一帧绘制前，都需要对缓冲区资源进行更新，例如物体的世界矩阵、观察矩阵等。当完成所有初始化和资源绑定工作后，将 ID3D12GraphicsCommandList 提交给 GPU 执行渲染命令，实际上是对当前的后台缓冲区进行写入，主要通过顶点着色器、像素着色器等各类着色器程序进行渲染。当完成绘制后，通过页面翻转进行 Present。而后等待 GPU 处理完成后再进行下一帧的绘制工作。

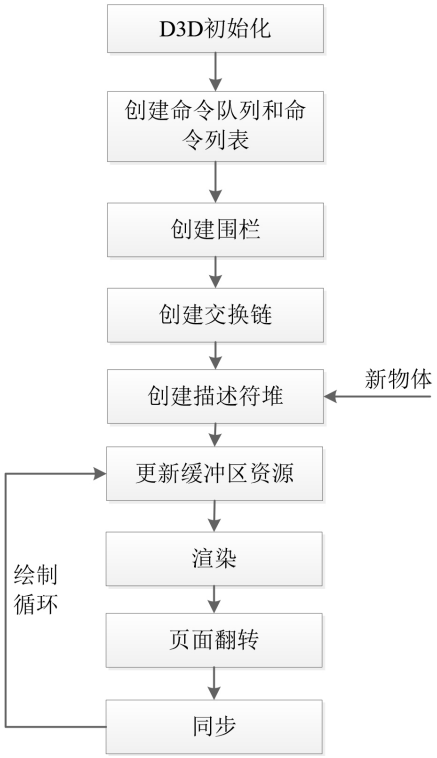

Figure 2. The rendering process of the D3D12
图 2. D3D12 渲染流程

2.3. 图形资源

图形资源主要是指用于 D3D 渲染所需的贴图、几何模型、材质等，是生成一个 3D 模型所需的外部资源。本文所需的贴图资源主要包括地球的贴图材质、卫星的贴图材质、星空的贴图材质以及地球、卫星、波束等各类几何图形结构等。

2.4. 轨道运行模型

为了对卫星运行轨道进行建模仿真，需要轨道运行方程作为支撑。首先要建立卫星运行的全局坐标系和本体坐标系，然后对卫星的轨道动力学方程进行计算，进而得到卫星的实时位置，继而通过 3D 引擎进行呈现。常用二体运动方程对卫星的运行轨道进行推算，它主要根据万有引力定律，将卫星和地球视为两个质点进行求解。为了提高轨道的精度，还可以考虑各种摄动力的影响，例如 J2 摄动力。

3. 系统实现过程

3.1. 轨道运动模型建立及求解

采用地心赤道惯性坐标系来描述卫星的运动方程，地心赤道惯性坐标系的坐标原点 O 位于地球中心，X 轴沿地球赤道平面指向春分点，Z 轴沿着地球自转轴指向协议地极，Y 轴与 X 轴和 Z 轴垂直并构成右手坐标系。

3.1.1. 二体运动方程及其求解

二体问题是天体物理学中的一个经典模型，主要研究的对象是空间中通过万有引力相互作用的两个物体。通过万有引力定律可以解得卫星在地心惯性坐标系(如图 3 所示)下的运动方程为：

$$\vec{r}''+u\frac{\vec{r}}{r^3}=0 \tag{1}$$

其中 $u=3.986\times10^{14}\,m^3/s^2$ 为地心引力常数，$\vec{r}$ 为卫星的位置向量，$|r|=\sqrt{r_x^2+r_y^2+r_z^2}$ 为卫星距离地心的距离，r_x 为卫星沿 X 轴的坐标分量，Y、Z 轴类似。上式属于三元二阶联立微分方程，在已知卫星初始位置 $\vec{r}_0$ 以及初始速度 $\vec{r}_0'$ 的条件下，可以使用数值法进行求解[7]。本文采用四阶龙格－库塔法进行求解，首先将上式改写为状态方程形式：

$$\begin{cases}\vec{r}''=f(t,r)=-u\dfrac{\vec{r}}{r^3}\\ \vec{r}'=\vec{v}\end{cases} \tag{2}$$

其中 $\vec{v}$ 为速度矢量，其求解公式为：

$$\begin{cases}\vec{L}_1=f\left(t_n,\vec{r}_n\right)\\ \vec{L}_2=f\left(t_{n+1/2},\vec{r}_n+\dfrac{\Delta t}{2}\vec{v}_n\right)\\ \vec{L}_3=f\left(t_{n+1/2},\vec{r}_n+\dfrac{\Delta t}{2}\vec{v}_n+\dfrac{\Delta t^2}{4}\vec{L}_1\right)\\ \vec{L}_4=f\left(t_{n+1},\vec{r}_n+\Delta t\vec{v}_n+\dfrac{\Delta t^2}{2}\vec{L}_2\right)\\ \vec{r}_{n+1}=\vec{r}_n+\Delta t\vec{v}_n+\dfrac{\Delta t^2}{6}\left(\vec{L}_1+\vec{L}_2+\vec{L}_3\right)\\ \vec{v}_{n+1}=\vec{v}_n+\dfrac{\Delta t}{6}\left(\vec{L}_1+2\vec{L}_2+2\vec{L}_3+\vec{L}_4\right)\\ t_{n+1}=t_n+\Delta t\\ t_{n+1/2}=t_n+(1/2)\Delta t\end{cases} \tag{3}$$

卫星的初始位置和速度可以根据给定的轨道根数进行求解，共有六个参数，其中，a 为轨道半长轴，代表椭圆轨道长轴的二分之一，e 为偏心率，代表轨道两个焦点之间的距离与长轴的比值，i 为轨道倾角，代表轨道所在平面与地球赤道平面的夹角，Ω 为升交点赤经，代表 X 轴与卫星轨道升交点之间的夹角，ω 为近地点幅角，代表升交点与近地点在同一轨道平面中所形成的夹角，t_p 为近地点时刻，代表卫星通过近地点的时刻。

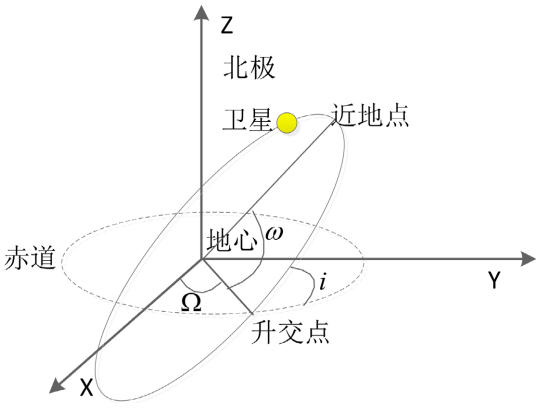

Figure 3. The location of the satellite in the geocentric inertial coordinate system
图 3. 卫星在地心惯性坐标系中的位置

通过以上六个参数就可以确定卫星轨道的形状，以及任意时刻卫星在轨道平面中的位置。在给出卫星轨道六参数的条件下，假定初始时刻为t_p，则可以通过轨道根数求出卫星在t_p时刻的位置$\vec{r}_0$以及速度$\vec{v}_0$，进而利用四阶龙格－库塔公式进行迭代计算出其它时刻的位置$\vec{r}_t$以及速度$\vec{v}_t$。轨道根数与卫星位置和速度的转换公式如下[7]：

$$\begin{cases} \vec{r}_0 = \dfrac{a\left(1-e^2\right)}{1+e\cos f}\vec{P} \\ \vec{v}_0 = \sqrt{u\left(\dfrac{2}{|r|}-\dfrac{1}{a}\right)}\vec{Q} \end{cases} \tag{4}$$

$$\vec{P} = \begin{bmatrix} \cos\Omega\cos(\omega+f)-\sin\Omega\sin(\omega+f)\cos i \\ \sin\Omega\cos(\omega+f)-\cos\Omega\sin(\omega+f)\cos i \\ \sin(\omega+f)\sin i \end{bmatrix} \tag{5}$$

$$\vec{Q} = \begin{bmatrix} -\cos\Omega\sin(\omega+f)-\sin\Omega\cos i\cos(\omega+f) \\ -\sin\Omega\sin(\omega+f)+\cos\Omega\cos i\cos(\omega+f) \\ \sin i\cos(\omega+f) \end{bmatrix} \tag{6}$$

式中f为真近点角，$t=t_p$时，$f=0$。

3.1.2. 经纬度计算

还可以通过计算卫星的经纬度，实现对卫星二维运动轨迹绘制的目的。在计算出卫星的三维坐标后，可以通过以下公式计算卫星的纬度[8]：

$$\begin{cases} \beta = \arctan\left(r_z/\sqrt{r_x^2+r_y^2}\right), r_x \neq 0, r_y \neq 0 \\ \beta = pi/2, r_z > 0, r_x = 0, r_y = 0 \\ \beta = -pi/2, r_z < 0, r_x = 0, r_y = 0 \end{cases} \tag{7}$$

通过下式计算卫星的经度：

$$\begin{cases} \phi = \arctan\left(r_y/r_x\right), r_x > 0 \\ \phi = pi/2 + \arctan\left(r_y/r_x\right), r_x < 0, r_y \geq 0 \\ \phi = -pi/2 + \arctan\left(r_y/r_x\right), r_x < 0, r_y \leq 0 \\ \phi = pi/2, r_x = 0, r_y > 0 \\ \phi = -pi/2, r_x = 0, r_y < 0 \end{cases} \tag{8}$$

式中ϕ为赤经，实际的卫星经度α还需要考虑地球自转的影响，设$\lambda = 0.00007\ rad/s$为地球自转角速度，G_0为初始时刻的格林尼治赤经，假定初始时刻为$t_p=0$，即卫星过近地点的时刻，则t时刻的卫星经度α：

$$\alpha = \phi - \left(G_0 + \lambda t\right) \tag{9}$$

3.1.3. 终端可见性计算

在已知卫星海拔高度$|r|$、卫星经纬度、终端经纬度、最低可见仰角 E 的条件下，可以计算终端对卫星的可见性，首先计算终端与卫星的地心角：

$$\gamma = \arccos\left(\cos(\beta_{land}) * \cos\left(\beta_{sat}\right) * \cos\left(\left|\alpha_{sat} - \alpha_{land}\right|\right) + \sin\left(\beta_{land}\right) * \sin\left(\beta_{sat}\right)\right) \tag{10}$$

其中 α_{land} 为终端经度，β_{land} 为终端纬度，α_{sat} 为卫星经度，β_{sat} 为卫星纬度，然后计算卫星终端与卫星之间的仰角：

$$\xi = \arctan\left(\left(\cos(\gamma) - \frac{R_e}{|r|}\right) \Big/ \sin(\gamma)\right) \tag{11}$$

则当 $\xi > E$ 的情况下，卫星可见。

3.2. 3D 模型渲染

3.2.1. 地球模型渲染

地球模型可以采用球形作为地球的几何模型，主要使用顶点着色器以及像素着色器进行渲染，其着色器资源包括顶点、索引、贴图和材质。需要在每一帧更新的资源主要是地球的世界矩阵，根据地球的自转速率在每一帧进行更新。用于地球渲染的顶点着色器伪代码如下：

代码名称：地球模型顶点着色器

输入：顶点 vertex_input，缓冲区 ID：Instance
输出：顶点 vertex_output
1. 获取常用缓冲区数据：ConstantData = GConstantbuffer[Instance]
2. 将顶点变换到世界空间：position = vertex_input.position × ConstantData.world
3. 将顶点变换到齐次裁剪空间：vertex_output.world = position × ViewProjectionMatrix
4. 将法线坐标变换到世界空间：vertex_output.normal = vertex_input.Normal × ConstantData.world
5. 对贴图坐标进行变换：vertex_output.texC = vertex_input.texC × ConstantData.texTransform
6. 获取材质的索引：vertex_output.matindex = ConstantData.matindex
7. 输出：return vertex_output

用于地球渲染的像素着色器伪代码如下：

代码名称：地球模型像素着色器

输入：顶点 vertex_output
输出：颜色 color
1. 获取材质数据：MaterialData = gMaterialData [vertex_output. matindex]
2. 进行贴图采样：color = gDiffuseMap.Sample(vertex_output. texC)
3. 进行漫反射光照计算：color = gAmbientLightComput(gAmbientLight, MaterialData ,color)
4. 进行方向光照计算：color = directLightComput(directLight, MaterialData ,color)
5. 输出：return color

3.2.2. 卫星模型渲染

卫星模型可以通过建模软件进行模型的建模，然后导入 D3D 渲染引擎进行渲染。主要使用顶点着色器以及像素着色器进行渲染，其着色器资源包括顶点、索引、贴图和材质。需要在每一帧更新的资源主要是卫星的世界矩阵，根据卫星的运行轨道在每一帧进行更新。其着色器伪代码同地球模型。

3.2.3. 轨道模型渲染

在进行轨道渲染前，需要首先生成指定时间段的卫星坐标数据，然后通过绘制直线列表的方式进行逐段直线的绘制，最终连起来就形成了轨道。本文分别基于二体运动方程以及 J2 摄动方程进行了卫星轨

道计算，满足不同建模精度需求。用于轨道模型渲染的着色器类型包括顶点着色器和像素着色器，着色器资源主要包括顶点缓冲区和索引缓冲区，不需要材质和贴图。其顶点着色器外代码如下：

代码名称：轨道模型顶点着色器

输入：顶点 vertex_input，缓冲区 ID：Instance
输出：顶点 vertex_output
1. 获取常用缓冲区数据：ConstantData = GConstantbuffer [Instance]
2. 将顶点变换到世界空间：position = vertex_input.position × ConstantData.world
3. 将顶点变换到齐次裁剪空间：vertex_output.world = position × ViewProjectionMatrix
4. 复制颜色数据：vertex_output.Color = vertex_input.Color
5. 输出：return vertex_output

用于轨道渲染的像素着色器伪代码如下：

代码名称：轨道模型像素着色器

输入：顶点 vertex_output
输出：颜色 color
1. 获取颜色数据：color = vertex_output.Color
2. 输出：return color

其渲染结果如图 4 所示：

Figure 4. The rendering result of the satellite orbit
图 4. 卫星轨道渲染结果

3.2.4. 波束模型渲染

卫星波束可以用圆锥体进行表示，同样使用顶点着色器和像素着色器进行渲染，着色器资源包括顶点、索引，不需要材质和贴图。其着色器伪代码同轨道模型伪代码。需要说明的是，在绘制波束时，需要在每一帧都更新波束的位置以及姿态，使得卫星波束一直指向地面。其中位置可以通过直接获取卫星的位置坐标即可，主要是姿态的变换较复杂，本文通过 3 个步骤进行转换，如图 5 所示：第一步将波束绕 X 轴旋转 α，$\alpha = i$；第二步将波束绕 r_2 轴旋转 β；第三步将波束绕 Z 轴旋转 γ，$\gamma = \Omega$。其渲染结果如图 6 所示。其中 β 按下式求解：

$$\begin{cases} \beta' = arccos\left(\dfrac{r_1 \bullet r_4}{|r_1||r_4|}\right), if\ x_4 > 0, \beta = -\beta', else\ \beta = \beta' \\ r_1 = (0, \cos\alpha, \sin\alpha) \\ r_2 = r_x \times r_1 \\ r_x = (1,0,0) \\ r = r_4 \times R_z \end{cases} \tag{12}$$

$$R_z = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{13}$$

式中 x_4 为 r_4 的 X 轴分量， R_z 为将波束绕 Z 轴旋转 γ 的旋转矩阵，则 $r_4 = r \times R_z'$ ， $r = (x_t, y_t, z_t)$ 为波束的当前姿态向量， x_t, y_t, z_t 为波束的 X 轴 Y 轴 Z 轴分量，它实际上就是卫星的位置坐标。

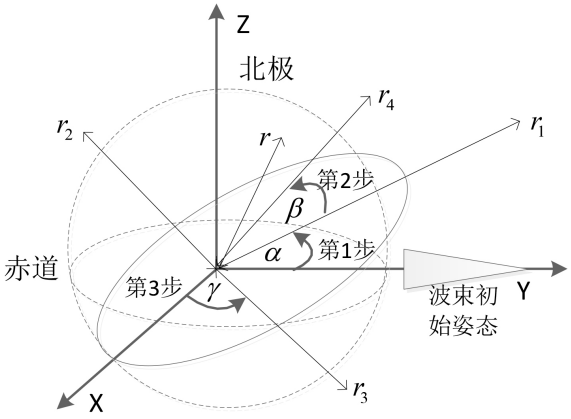

Figure 5. The conversion process of the satellite beam
图 5. 波束变换示意图

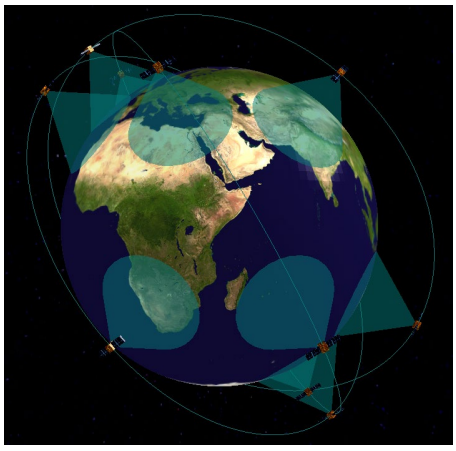

Figure 6. The rendering result of the satellite beam
图 6. 卫星波束渲染结果

3.2.5. 星间链路模型渲染

星间链路可以用一段线段进行表示，同样使用顶点着色器和像素着色器进行渲染，着色器资源包括顶

点、索引，不需要材质和贴图。其着色器伪代码同轨道模型伪代码。需要说明的是，由于卫星的位置一直在变化，因此每个顶点的位置都要实时变化，可使用动态顶点缓冲区进行顶点数据的更新，如图 7 所示。

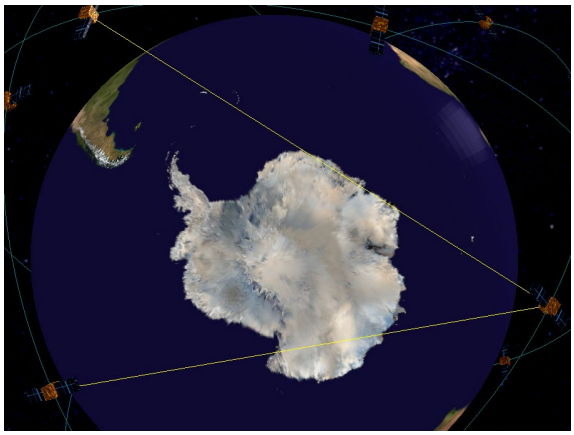

Figure 7. The rendering result of the inter-satellite link
图 7. 星间链路渲染结果

3.3. 二维运动轨迹渲染

本文采用 direct2D 技术进行 2D 渲染，为了支持与 3D 的交互，本文先通过在 D3D12 的设备上创建 D3D11 设备，然后使用 D3D11 设备获取 2D 渲染接口，进而实现在 3D 画面上进行 2D 图形的渲染。主要包括以下步骤：

代码名称：2D 渲染

```
1. D3D11On12CreateDevice(*D3D12device,&D3D11device) //利用 D3D12 设备创建 D3D11 设备
2. D3D11device.As(&d3d11On12Device) //使用 D3D11 设备检索 D3D11on12 设备接口
3. d3d11On12Device.As(&dxgiDevice)  //使用 D3D11on12 获取生成图像数据的 DXGI 接口
4. d2dFactory->CreateDevice(dxgiDevice.Get(),&d2dDevice)) //使用 D2D 工厂将 DXGI 接口转换为 D2D 设备
5. d2dDevice->CreateDeviceContext(deviceOptions,&d2dDeviceContext)) //使用 D2D 设备获取 D2D 设备上下文接口
6. …..//在完成 3D 绘制后，通过一系列命令获取二维图面
7. d2dDeviceContext->DrawText(string,format,position…) //在二维图面上绘制文字
8. d2dDeviceContext->DrawBitmap(bitmap,format,position…)  //在二维图面上绘制图片
9. d2dDeviceContext->DrawGeometry(Gemetry,format,position…) //在二维图面上绘制线段
```

在进行卫星二维运动界面的渲染过程中，使用 DrawBitmap 函数绘制二维地图，使用 DrawGeometry 函数绘制卫星的运动轨迹，DrawText 绘制卫星的名字以及其他必要的文字。

3.4. 人机交互界面设计

UWP 采用 XAML (可扩展标记语言)进行系统界面设计，通过 XAML，可以使得 Windows 界面设计更加高效灵活并兼具美观性。本文采用 XAML 设计的人机交互界面如下图所示，包括 3D 模型呈现窗口、菜单栏、工具栏、项目编辑界面。其中，3D 模型呈现窗口采用 SwapChainPanel 控件进行设计，SwapChainPanel 专门用于 D3D 渲染图形的实时显示，通过利用 SwapChainPanel，可以将 D3D 渲染结果嵌入到 UWP 界面中，实现 Windows 通用软件界面与 D3D 引擎技术的相结合，如图 8 所示。基于 XAML 的界面伪代码如下所示：

代码名称：系统界面设计

```
<Page>
<Grid Row=3 Column=1> //整个界面分为 3 行 1 列
<Panel Grid.row=0>    //第一行放置菜单栏
<Panel Grid.row=1>    //第二行放置工具栏
<SplitPane Grid.row=2> //第三行放置 3D 界面
<SplitPane.Pane>
<TreeView>  //浮出界面放置项目列表，以树形结构展现
</ SplitPane.Pane>
<SplitPane.Content>
<SwapChainPanel>  //主界面放置 3D 渲染视图
</SplitPane.Content>
</Grid>
</Page>
```

通过 XAML 设计的系统界面如下图所示。

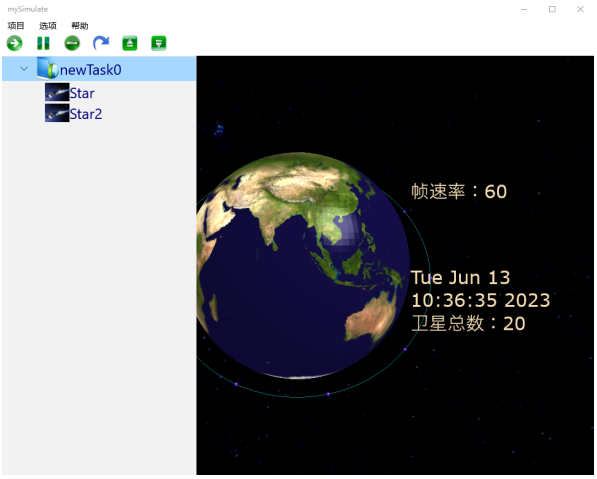

Figure 8. The UI of the system
图 8. 系统界面

4. 试验验证

本文通过对模拟星链卫星、模拟一网卫星星座进行建模仿真验证本文方法的可行性和有效性。

4.1. 卫星参数

模拟星链卫星、模拟一网卫星星座的星座参数如表 1 所示，假设各个星座的构型均为 Walker 构型。

Table 1. Parameter of the simulation Constellation
表 1. 仿真星座参数

星座名称	每轨卫星数量	轨道高度(Km)	轨道倾角(度)
模拟星链星座	22	550	53
模拟一网星座	49	1200	87.9

采用了普通家用计算机进行了仿真运行，操作系统为 win10，CPU 主频 3.6 GHz，内存 8 GB，显卡核心频率 1.2 GHz，显卡内存 2 GB。

4.2. 仿真结果

为了验证本文算法的准确性，将本文计算结果与 STK 仿真结果进行了对比，轨道高度为 1200 km，轨道倾角为 87.9˚，升交点赤经、近地点幅角、近地点时刻均为 0，STK 计算模型选择二体模型，如表 2 所示为仿真结果，结果表明本文计算的轨道数据与 STK 仿真结果基本一致，表明了方法的准确性。

Table 2. Orbit simulation result comparison of the STK and this paper
表 2. 本文与 STK 的轨道计算结果对比

时间(s)	本文计算结果(/km)			STK 计算结果(/km)		
	X 坐标	Y 坐标	Z 坐标	X 坐标	Y 坐标	Z 坐标
0	7571	0	0	7571	0	0
60	7558.49	15.944	434.823	7558.486	15.944	434.8229
120	7520.993	31.836	868.208	7520.987	31.835	868.208
180	7458.632	47.622	1298.723	7458.625	47.622	1298.724
240	7371.613	63.251	1724.945	7371.608	63.25	1724.946
300	7260.224	78.671	2145.465	7260.222	78.67	2145.466
360	7124.833	93.83	2558.894	7124.837	93.83	2558.894
420	6965.885	108.68	2963.865	6965.899	108.6799	2963.863
480	6783.925	123.17	3359.035	6783.934	123.17	3359.035
540	6579.545	137.253	3743.1	6579.543	137.2533	3743.102
600	6353.406	150.883	4114.795	6353.403	150.8827	4114.796

图 9 为卫星星座的三维仿真结果，图 10 为卫星星座的二维仿真结果。通过仿真分析了不同星座构型在每天内的覆盖性，其中，卫星终端经度为−120 度，纬度为 40 度，最低仰角设置为 20 度，可以看出在只有一轨时，一网卫星的覆盖性要优于星链星座，如表 3 所示。

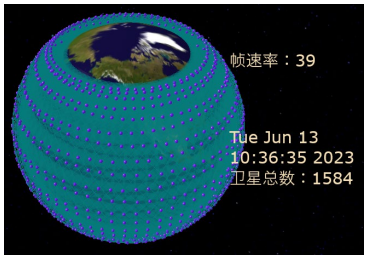

Figure 9. The simulation result of the simulated Starlink of the first stage
图 9. 模拟星链星座一阶段仿真结果

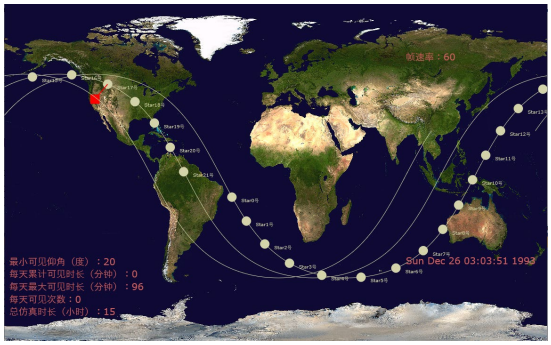

Figure 10. The trajectory of motion of the two-dimension simulation result of the simulated Starlink
图 10. 模拟星链卫星星座二维运动轨迹仿真结果

Table 3. Coverage performance of the Constellation with different configuration in one day
表 3. 一天内不同星座构型覆盖性能

星座名称	轨道数量	累计可见时长(分钟)	最大可见时长(分钟)	可见次数
模拟星链星座	1	223	96	24
模拟一网星座	1	342	224	2

5. 结束语

随着人类航天事业的蓬勃发展，针对航天领域的研究成为当前热点，而卫星轨道构型的可视化建模仿真是其中的重点方向，对于研究和分析卫星星座的布局合理性具有重要的意义。本文采用 D3D12 渲染技术对卫星星座模型进行了可视化渲染，同时基于 UWP 系统框架以及 C++语言进行系统开发，仿真结果表明了本文方法的有效性，为后续进一步开展高中低混合轨道布局研究、星座路由规划研究、卫星可见性研究等奠定了基础。

参考文献

[1] 李刚. 基于 STK 的综合态势显示与控制系统的研究[D]: [硕士学位论文]. 济南: 山东大学, 2019.

[2] 黄娟. 基于 MATLAB/STK 的卫星通信场景仿真设计与实现[D]: [硕士学位论文]. 合肥: 安徽大学, 2016.

[3] 代建中, 冯旭哲, 李文屏, 等. 基于 STK 的北斗三号卫星导航系统 AC 仿真分析[J]. 计算机仿真, 2022, 39(5): 22-25.

[4] 胡杰, 崔健, 王英杰. 基于 STK 的双星发射任务可视化仿真方案的研究[J]. 计算机工程与应用, 2017(53): 253-257.

[5] 翟小珂. 卫星运行可视化仿真平台的设计与实现[D]: [硕士学位论文]. 呼和浩特: 内蒙古大学, 2016.

[6] 陈卡. Directx93D 图形程序设计[M]. 上海: 上海科学技术出版社, 2003.

[7] 白文娟. 基于 OpenGL 的 GPS 卫星轨道仿真与可视化实现[D]: [硕士学位论文]. 哈尔滨: 哈尔滨工程大学, 2009.

[8] 杨平利, 黄少华, 江凌, 等. 卫星运行三维场景及星下点轨迹可视化研究[J]. 计算机工程与科学, 2012, 34(5): 101-106.