《异世界窗户桌面系统·完整需求规格与技术实现研究报告（v1.0）》

1. 执行摘要与系统架构综述

本研究报告旨在详细阐述基于 Windows 11（特别针对 24H2 架构演进）的“异世界窗户”（Isekai Window）实时 3D 桌面系统的设计规范、技术架构与实现路径。该系统不仅仅是一个动态壁纸应用，而是一个集成了底层操作系统交互、计算机视觉（CV）追踪、神经符号人工智能（Neuro-Symbolic AI）与高保真 3D 渲染的深度人机交互（HCI）平台。其核心愿景是将传统的二维显示器转化为一个具有真实物理视差的“魔法窗口”，窗口内驻留一位具备自主意识、能够协助用户管理系统任务的二次元 AI 伙伴。

本报告的撰写基于对现有 Windows Shell 架构、Unity 渲染管线（URP）、大型语言模型（LLM）接口及生物识别追踪技术的详尽调研。分析指出，Windows 11 24H2 版本对桌面窗口管理器（DWM）与 Shell 层级的重构，使得传统的 WorkerW 注入技术失效，必须采用基于 Progman 的子窗口注入新范式 [1]。同时，为了实现“打破第四面墙”的视觉体验，系统必须摒弃传统的 3D 相机旋转逻辑，转而采用罗伯特·库伊马（Robert Kooima）提出的“广义透视投影”（Generalized Perspective Projection）算法，即离轴投影（Off-Axis Projection），以根据用户头部位置实时重构视锥体 [2]。

在智能体构建方面，系统采用模块化“大脑-身体”分离架构：以大语言模型（Gemini Pro/GPT-4）为认知核心，通过函数调用（Function Calling）机制与本地受控沙箱（Sandbox）进行交互，从而在确保系统安全的前提下实现对文件、日程及番茄钟等工具的操控 [3]。

---

2. Windows 11 24H2 Shell 深度集成与渲染层架构

实现“异世界窗户”的首要技术挑战，在于如何将高负载的 3D 渲染上下文稳定地嵌入到 Windows 桌面图标层（Icon Layer）之下、系统背景层之上。这需要对 Windows 窗口句柄（HWND）层级进行极低级别的操作。

2.1 桌面合成器架构演进：从 WorkerW 到 Progman

在 Windows 10 及 Windows 11 的早期版本（23H2 及之前）中，动态壁纸应用通常利用 User32.dll 中的未公开机制。开发者通过向 Progman（Program Manager）窗口发送 0x052C 消息，会触发系统生成一个名为 WorkerW 的顶层窗口，将桌面图标所在的 SHELLDLL_DefView 与背景分离。开发者随后可以将自己的渲染窗口挂载到 WorkerW 之后 [5]。

然而，根据最新的技术情报，Windows 11 24H2 引入了重大的架构变更。微软为了优化性能并重构任务视图体验，将所有桌面合成逻辑重新收归至 Progman 窗口层级之下，取消了旧式的顶层 WorkerW 分离机制 [1]。这意味着，任何试图将窗口简单置于 HWND_BOTTOM 或寻找旧版 WorkerW 的代码，在 24H2 上运行时会出现渲染层级混乱——窗口要么遮挡图标，要么被系统壁纸覆盖，甚至导致 Explorer 进程崩溃 [1]。

表 1：**Windows** 桌面窗口层级架构对比分析

特性维度	**Windows 10 / 11 (Pre-24H2)**	**Windows 11 (24H2 及后续版本)**	技术影响与应对策略
根父节点	桌面窗口 (Desktop Window)	Progman (程序管理器) 句柄	必须重新定位根锚点，放弃对 Desktop Window 的直接依赖。
图标容器	SHELLDLL_DefView (通常为 WorkerW 子节点)	SHELLDLL_DefView (Progman 的直接子节点)	图标层现在是 Progman 的直接子窗口，注入位置必须在其 Z 序之后。
壁纸容器	WorkerW (顶层兄弟窗口)	WorkerW (Progman 的直接子节点)	壁纸层级下沉，自定义窗口必须插入到 SHELLDLL_DefView 与 WorkerW 之间。
注入技术	创建 WorkerW 并在其上绘制	向 Progman 注入子窗口 (Child Window	需使用 SetParent API 将 Unity 窗口变

		Injection)	为 Progman 的子窗口。
Z 序稳定性	高 (利用系统 Bug 长期稳定)	低 (需处理重绘与焦点抢占)	需持续监听 WM_WINDOWPOSCHANGING 消息以强制维持层级 [8]。

2.2 Progman 子窗口注入技术实施方案

针对 24H2 架构，本系统必须采用“嵌入式子窗口”策略。Unity 应用程序不应作为独立的顶层窗口运行，而必须“寄生”于 Windows Shell 进程中。

核心实现步骤详述：

1. 桌面状态初始化（The Summoning）：
   系统启动时，首先通过 P/Invoke 调用 SendMessageTimeout 向 Progman 发送 0x052C 消息。这一步至关重要，它强制 Windows 初始化桌面背景绘制机制，确保 SHELLDLL_DefView 和背景 WorkerW 均已创建且处于活跃状态 1。
2. 句柄侦察与拓扑分析：
   利用 FindWindow 和 FindWindowEx 遍历窗口树，锁定以下关键句柄：
   - hProgman: FindWindow("Progman", null) —— 目标父节点。
   - hShellView: FindWindowEx(hProgman, 0, "SHELLDLL_DefView", null) —— 必须位于其下的图标层。
   - hWorkerW: FindWindowEx(hProgman, 0, "WorkerW", null) —— 必须位于其上的背景层。

     在此阶段，系统应进行健全性检查。如果未找到 hShellView，说明系统可能处于某种非标准 Shell 模式（如某些 Kiosk 模式配置），此时应尝试回退或报错 5。
3. 父级重定向（Reparenting）：
   获取 Unity 主窗口句柄 hUnity 后，调用 SetParent(hUnity, hProgman)。这一步将 Unity 窗口从顶层窗口转变为 Progman 的子窗口。此时，Unity 窗口将继承 Progman 的生命周期特性，并在“显示桌面”（Win+D）操作时表现出与桌面一致的行为 10。
4. Z 序外科手术（Z-Order Injection）：
   这是实现“图标悬浮于异世界之上”视觉效果的关键。调用 SetWindowPos 函数，精心构造 uFlags（如 SWP_NOACTIVATE | SWP_NOMOVE | SWP_NOSIZE）：
   - 将 hUnity 插入到 hShellView 之后 (HWND_BOTTOM 或具体句柄)。
   - 或者，更稳健的做法是将 hWorkerW（系统壁纸）推到 hUnity 之后。

代码逻辑需特别注意处理 Windows 的动态刷新。当用户更换主题或分辨率时，WorkerW 可能会被销毁重建，系统必须通过 SetWinEventHook 监听系统事件，以便在 Shell 重启时自动重新注入 [1]。

2.3 输入穿透与交互层设计

全屏运行的 Unity 窗口若不做处理，将拦截所有鼠标和键盘输入，导致用户无法点击桌面图标或使用右键菜单，这严重违背了“桌面系统”的基本可用性需求。

透明输入转发机制：

- 点击测试（**Hit Testing**）：Unity 窗口需开启 WS_EX_LAYERED 和 WS_EX_TRANSPARENT 扩展样式，但这会导致所有输入穿透，包括对 AI 角色的交互。因此，必须采用更精细的混合方案。
- 全局鼠标钩子（**Global Mouse Hook**）：利用 SetWindowsHookEx 安装 WH_MOUSE_LL 钩子。系统实时监控鼠标坐标：
  - 当光标位于 AI 角色（二次元美少女）的包围盒（Bounding Box）或系统UI（如番茄钟面板）范围内时，钩子拦截输入并转发给 Unity 处理，实现“摸头”、“拖拽文件给 AI”等交互。
  - 当光标位于 3D 场景的空白区域（即“空气”部分）时，钩子放行输入，使其自然穿透到后方的 SHELLDLL_DefView，允许用户框选图标或打开文件 [5]。

---

3. “魔法窗口”光学引擎：广义透视投影技术

为了将物理显示器转化为通往异世界的真实窗口，传统的 3D 渲染相机模型（针孔相机模型，透视点位于屏幕中心）已无法满足需求。当用户头部移动时，传统相机会导致画面内容跟随旋转，产生“鱼缸里的画”的虚假感。本系统必须实现广义透视投影（**Generalized Perspective Projection**），即离轴投影。

3.1 离轴投影数学模型与视锥体变形

在离轴投影模型中，显示器被视为 3D 空间中的一个固定矩形窗口（Portal）。摄像机的视锥体（Frustum）不再是对称的四棱锥，而是根据观察者眼睛位置（Pe）相对于屏幕四角（Pa, Pb, Pc）的位置动态生成的斜棱锥 [13]。

核心算法逻辑（基于 Robert Kooima 论文）：
系统需要在每一帧（Frame）计算以下向量基：

1. 屏幕右向量 **(vr)**：(Pb - Pa).normalized

2. 屏幕上向量 **(vu)**：(Pc - Pa).normalized
3. 屏幕法向量 **(vn)**：Cross(vr, vu).normalized

随后，计算眼睛位置 Pe 在这组基底上的投影，从而确定视锥体的近剪裁面（Near Clip Plane）的边界参数（Left, Right, Bottom, Top）：

$$l = \vec{vr} \cdot (P_a - P_e) \times \frac{n}{d}$$

$$r = \vec{vr} \cdot (P_b - P_e) \times \frac{n}{d}$$

$$b = \vec{vu} \cdot (P_a - P_e) \times \frac{n}{d}$$

$$t = \vec{vu} \cdot (P_c - P_e) \times \frac{n}{d}$$
其中 n 为近剪裁面距离，d 为眼睛到屏幕平面的垂直距离（$-\vec{vn} \cdot (P_a - P_e)$）[2]。

3.2 Unity URP 渲染管线实现

在 Unity 的通用渲染管线（URP）中实现此逻辑需要绕过相机系统的默认行为。

- 矩阵覆写（**Matrix Override**）：不同于传统管线，URP 对相机投影矩阵的控制更为严格。我们需要在 LateUpdate 生命周期中，利用 Camera.projectionMatrix 属性手动赋值。
- **Matrix4x4.Frustum** 调用：利用 Unity 内置的 Matrix4x4.Frustum(l, r, b, t, n, f) 函数构建基础投影矩阵。需要注意的是，直接计算出的矩阵可能需要进行坐标系转换（Unity 为左手坐标系），且需根据 Camera.worldToCameraMatrix 进行相应的视口平移，确保虚拟摄像机的位置与物理世界的眼睛位置严格对应 [16]。
- 阴影剔除修正（**Culling Fix**）：离轴投影会导致视锥体极度变形，Unity 默认的视锥体剔除（Frustum Culling）算法可能会错误地剔除掉位于视线边缘但实际可见的物体（尤其是阴影投射体）。解决方案是手动扩大相机的 Culling Matrix，或者在 URP 设置中增加“保守剔除”（Conservative Culling）的余量 [13]。
- 物理尺度校准：为了达到“真实视差”，系统必须具备物理尺度感知能力。需提供配置界面让用户输入显示器的物理尺寸（如 27英寸），从而计算出像素与物理毫米的比例关系，确保虚拟世界中的 10cm 移动在视觉上等同于物理世界的 10cm 移动 [20]。

---

4. 6DOF 传感器融合与生物识别追踪

离轴投影的真实感完全依赖于对用户头部位置（X, Y, Z）的高频、低延迟追踪。系统需构建一个多

模态传感器融合层，兼顾普通用户的网络摄像头方案与极客用户的专业硬件方案。

4.1 视觉追踪主方案：MediaPipe (CPU 瓶颈与优化)

Google 的 MediaPipe 框架提供了高精度的 Face Mesh 解决方案，能够提取 468 个面部特征点，从而解算出高精度的头部 6DOF 姿态 [21]。

Windows 平台的性能挑战：
现有研究表明，MediaPipe 的 Unity 插件（如 Homuler 移植版）在 Windows 桌面平台上不支持 GPU 加速，只能依赖 CPU 推理 22。这在桌面级应用中是一个巨大的性能风险，因为 CPU 还需要处理物理模拟和复杂的系统逻辑。
应对策略：异步并行管线
为了避免追踪计算阻塞主渲染线程（导致掉帧），系统必须设计异步架构：

1. 独立线程/进程：将 MediaPipe 的推理过程剥离到独立的 Task 或子进程中运行。
2. 数据平滑（**Smoothing**）：由于网络摄像头存在噪点，且 CPU 推理可能产生抖动，必须在原始数据输出后接入 **1€ Filter (One Euro Filter)** 或 卡尔曼滤波（**Kalman Filter**），在保持低延迟的同时消除微小抖动，否则用户在静止时会看到画面轻微震颤，严重破坏沉浸感 [23]。

4.2 专业级回退方案：UDP / OpenTrack 桥接

为了满足高端用户（拥有 TrackIR、Tobii 眼动仪或 VR 设备）的需求，系统必须开放标准的 UDP 数据接口。

- 协议设计：监听本地特定端口（如 4242），接收标准化的 6 双精度浮点数包（X, Y, Z, Yaw, Pitch, Roll）。
- **OpenTrack** 集成：OpenTrack 是开源的头部追踪中间件，支持数百种输入设备。本系统通过编写 OpenTrack 的 UDP 输出插件配置，可以直接利用其强大的生态。例如，用户可以使用 iPhone 的 FaceID 摄像头通过网络发送高精度数据给 OpenTrack，再转发给本系统，从而完全卸载 PC 的 CPU 视觉计算压力 [25]。
- 自动切换逻辑：传感器融合模块会实时监测 UDP 端口。一旦检测到有效的数据流（Non-zero stream），系统自动挂起内部的 MediaPipe 线程，切换至 UDP 输入源，实现性能与精度的双重优化。

---

5. 认知核心：神经符号 AI 代理架构

“异世界”的 AI 伙伴不仅是聊天机器人，更是操作系统层面的智能代理（Agent）。这要求系统采用**神经符号（Neuro-Symbolic）**架构：利用大语言模型（LLM）处理模糊意图与情感，利用确定性的代码逻辑处理系统操作。

5.1 大脑-身体分离架构

为了保证扩展性与稳定性，智能体被拆分为三个逻辑层：

1. 大脑（**Cognition Layer**）：基于云端 API（Gemini Pro / GPT-4）或本地量化模型（Llama 3 8B，需高性能 GPU）。负责意图识别、对话生成与情感分析。
2. 身体（**Expression Layer**）：位于 Unity 内部。接收“大脑”输出的情感标签（如 [Joy], ``）并驱动 VRM 模型的 BlendShape（表情基形态）；接收文本并通过 TTS（如 Azure Neural TTS 或 VITS）生成语音与口型同步（Lip Sync）[27]。
3. 手（**Execution Layer**）：位于.NET Core 后台进程。负责实际执行文件操作、API 调用等高风险任务。

5.2 基于 Function Calling 的任务管理

为了实现“番茄钟、日历、邮件管理”，不能依赖 LLM 生成不可控的代码或文本，必须严格使用 **Function Calling (Tool Use)** 协议 [3]。

实现流程案例：

1. 用户指令：“明天下午三点有个会议，帮我记一下。”
2. **Schema** 定义：系统向 LLM 预置工具定义：

```JSON
{
  "name": "add_calendar_event",
  "description": "Add an event to the system calendar",
  "parameters": {
    "type": "object",
    "properties": {
      "title": {"type": "string"},
      "start_time": {"type": "string", "format": "iso8601"}
    },
    "required": ["title", "start_time"]
```

```
  }
}
```

3. **LLM** 推理：LLM 分析语境，不直接回复文本，而是返回结构化 JSON：{"tool": "add_calendar_event", "args": {"title": "会议", "start_time": "2023-10-27T15:00:00"}}。
4. 安全执行：C# 后端捕获此 JSON，进行参数校验（检查时间格式、冲突），然后调用 Microsoft Graph API 执行写入。
5. 闭环反馈：执行结果（"Event ID: 123 created"）被回传给 LLM，LLM 随后生成最终回复："好的，会议已添加，别迟到哦！"

此机制确保了 AI 对系统的操作是确定性的、受控的，并且完全解耦了自然语言的模糊性。

---

6. 系统集成与安全沙箱机制

允许 AI 访问文件与邮件系统存在巨大的安全隐患（如 Prompt Injection 攻击导致误删文件）。系统必须构建严格的"人在回路"（Human-in-the-Loop）安全沙箱。

6.1 Microsoft Graph API 与现代化鉴权

对于日历和邮件功能，直接通过 IMAP/SMTP 或屏幕抓取 Outlook 窗口是不稳定且不安全的（尤其是面对 MFA 多因素认证时）。系统必须集成 **Microsoft Graph SDK** [29]。

- 鉴权流程：采用 **OAuth 2.0 Device Code Flow**。用户在首次使用时，通过浏览器登录微软账号并授权应用特定范围（Scope）：Calendars.ReadWrite, Mail.Read。系统仅保存 Refresh Token，不接触用户密码。
- **API** 封装：后台服务封装 Graph API 调用。例如，AI 请求"读邮件"时，后台仅拉取最近 5 封邮件的 Subject 和 Sender 摘要喂给 LLM，绝不将所有邮件正文直接暴露给上下文窗口，以保护隐私并节省 Token [31]。

6.2 本地文件系统沙箱与权限控制

针对"文件管理"需求，系统实施严格的路径限制：

- 白名单机制：AI 默认只能读写特定的 User/IsekaiWorkspace/ 目录。

- 意图拦截（**Intent Interception**）：当 LLM 试图调用 delete_file 或 send_email 等敏感工具时，系统会触发二级确认机制。在 3D 窗口中，AI 角色会举起一块虚拟“剪贴板”或弹出一个全息窗口，显示：“我要删除这个文件，可以吗？”，只有当用户点击“确认”或通过头部姿态（点头）授权后，操作才会真正执行。这利用了离轴投影的交互优势，将安全确认变成了沉浸式体验的一部分 [33]。

6.3 进程隔离与 IPC 通讯架构

为了实现“商用级稳定性”和“完整退出机制”，渲染进程（Unity）与业务逻辑进程（Core Service）必须物理隔离 [35]。

- 架构设计：
  - IsekaiDisplay.exe (Unity): 纯粹的“瘦客户端”，负责渲染、动画、输入转发。如果显卡驱动崩溃，它不会拖垮后台任务。
  - IsekaiCore.exe (.NET 8): 负责 LLM 通讯、Graph API、文件 I/O、系统监控。
- 通讯管道：两者通过 **Named Pipes (命名管道)** 进行极低延迟的 IPC 通讯。采用 JSON-RPC 格式封装数据包。例如，Core 进程监控到新邮件，通过 Pipe 发送 {"method": "ShowNotification", "params": {"msg": "New Mail!"}}，Display 进程收到后即刻驱动角色做出反应 [24]。
- 看门狗（**Watchdog**）机制：引入第三个轻量级进程 IsekaiGuardian.exe。它实时监控 Display 和 Core 的心跳。如果 Unity 进程崩溃，Guardian 会立即介入，向 Progman 发送重置信号恢复默认壁纸，防止桌面变黑，随后尝试静默重启渲染器，实现“故障自愈” [1]。

---

7. 角色渲染与个性化扩展 (UniVRM)

为了满足“可自定义二次元美少女”的需求，系统全面支持 **VRM 1.0** 开放标准。

- 运行时加载：利用 UniVRM 库，系统支持在运行时动态加载用户放入 Characters/ 文件夹的 .vrm 模型。这要求系统在加载时动态重定向 Shader（将 VRM 的 MToon Shader 转换为 URP 兼容的 Lit Shader 或自定义卡通 Shader），以适配当前的光照环境 [28]。
- 表情重映射：系统维护一个情感映射表，将 LLM 输出的 7 种基本情感（Happy, Sad, Angry, Neutral 等）映射到 VRM 规范定义的 BlendShapeClips。对于不支持某些表情的旧版模型，系统通过算法自动回退到相近表情或仅使用肢体语言 [27]。

---

8. 性能优化与资源调度

- 睡眠模式（**Suspend Mode**）：当检测到全屏应用（如 3A 游戏或 IDE 全屏模式）运行时，系统必须将 Unity 渲染帧率降至 0 或 1fps，并释放部分显存，以确保不影响用户的主力工作。
- **GPU** 优先级：由于是桌面背景应用，不应抢占高优先级 GPU 时间片。但在用户交互（如鼠标滑过）瞬间，需动态提升线程优先级以保证响应流畅。

---

9. 结论

“异世界窗户桌面系统”通过整合 Windows 24H2 的底层 Progman 注入技术、罗伯特·库伊马的离轴投影算法以及基于 Function Calling 的神经符号 AI 架构，成功在技术层面论证了将操作系统桌面转化为沉浸式 3D 空间的可行性。该设计不仅解决了“WorkerW 失效”和“GPU 追踪不支持”等关键工程障碍，还通过沙箱机制与进程隔离架构，确保了系统在商业应用场景下的安全性与稳定性。这标志着桌面美化软件从“静态装饰”向“空间计算代理”的范式转移。

报告编制：高级系统架构师（*HCI/OS Kernel* 专项）

引用的著作

1. Win11 preview version 26XX dynamic wallpaper invalid · Issue ..., 访问时间为 十一月 22, 2025，https://github.com/rocksdanister/lively/issues/2074
2. Off-axis projection in Unity. An implementation of off-axis... | by Michel de Brisis | TRY Creative Tech | Medium, 访问时间为 十一月 22, 2025, https://medium.com/try-creative-tech/off-axis-projection-in-unity-1572d826541e
3. Function calling using the Gemini API | Firebase AI Logic - Google, 访问时间为 十一月 22, 2025，https://firebase.google.com/docs/ai-logic/function-calling
4. Function calling with LLMs(C#) - Microsoft Learn, 访问时间为 十一月 22, 2025, https://learn.microsoft.com/en-us/microsoftteams/platform/teams-ai-library/csharp/in-depth-guides/ai/function-calling
5. Draw on the windows 11 wallpaper in C++/Qt - Stack Overflow, 访问时间为 十一月 22, 2025, https://stackoverflow.com/questions/78880999/draw-on-the-windows-11-wallpaper-in-c-qt
6. Draw on Windows 10 wallpaper in C++ - Stack Overflow, 访问时间为 十一月 22, 2025, https://stackoverflow.com/questions/56132584/draw-on-windows-10-wallpaper-in-c
7. Simple Window Switcher is broken on Windows 11 24H2 · Issue #3765 · valinet/ExplorerPatcher - GitHub, 访问时间为 十一月 22, 2025, https://github.com/valinet/ExplorerPatcher/issues/3765
8. c++ - Pin window to desktop / Glue window to desktop / "Always-on-bottom"

window - Stack Overflow, 访问时间为 十一月 22, 2025, https://stackoverflow.com/questions/65028303/pin-window-to-desktop-glue-window-to-desktop-always-on-bottom-window
9. Make WPF Window Ignore Show Desktop (Win+D) - Stack Overflow, 访问时间为 十一月 22, 2025, https://stackoverflow.com/questions/71903355/make-wpf-window-ignore-show-desktop-wind
10. Prevent drawing over a child window under WorkerW when wallpaper is changed, 访问时间为 十一月 22, 2025, https://stackoverflow.com/questions/78169263/prevent-drawing-over-a-child-window-under-workerw-when-wallpaper-is-changed
11. c# - Keeping window visible through "Show Desktop"/Win+D - Stack Overflow, 访问时间为 十一月 22, 2025, https://stackoverflow.com/questions/10009623/keeping-window-visible-through-show-desktop-wind
12. What is the correct way to set a Delphi (FMX) form as a desktop wallpaper with a video playing in it (to make it like "live wallpaper")? - Stack Overflow, 访问时间为 十一月 22, 2025, https://stackoverflow.com/questions/79727730/what-is-the-correct-way-to-set-a-delphi-fmx-form-as-a-desktop-wallpaper-with-a
13. Vision & Visuals - DIS Lab, 访问时间为 十一月 22, 2025, https://dis.dankook.ac.kr/lectures/vr23/wp-content/uploads/sites/91/2023/09/VR-lecture3-Vision-Visuals.pdf
14. Implementation of Headtracking and 3D Stereo with Unity and VRPN for Computer Simulations - NASA Technical Reports Server (NTRS), 访问时间为 十一月 22, 2025, https://ntrs.nasa.gov/api/citations/20130014602/downloads/20130014602.pdf
15. Implementation of Generalized Perspective Projection on the Unity | by Hiruma Kazuya, 访问时间为 十一月 22, 2025, https://edom18.medium.com/implementation-of-generalized-perspective-projection-on-the-unity-c9472a94f083
16. Scripting API: Matrix4x4.Frustum - Unity - Manual, 访问时间为 十一月 22, 2025, https://docs.unity3d.com/6000.2/Documentation/ScriptReference/Matrix4x4.Frustum.html
17. Job System Tutorial - JacksonDunstan.com, 访问时间为 十一月 22, 2025, https://www.jacksondunstan.com/articles/4796
18. Cg Programming/Unity/Projection for Virtual Reality - Wikibooks, 访问时间为 十一月 22, 2025, https://en.wikibooks.org/wiki/Cg_Programming/Unity/Projection_for_Virtual_Reality
19. Create a global fog effect | High Definition RP | 16.0.6 - Unity - Manual, 访问时间为 十一月 22, 2025, https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@16.0/manual/create-a-global-fog-effect.html
20. ndaneil/interactive-3d-screen: Our monitors and laptop screens are 2D surfaces.

With the help of AI, it is possible to turn such flat surfaces into 3D interactive windows. - GitHub, 访问时间为 十一月 22, 2025, https://github.com/ndaneil/interactive-3d-screen
21. Face landmark detection guide | Google AI Edge, 访问时间为 十一月 22, 2025, https://ai.google.dev/edge/mediapipe/solutions/vision/face_landmarker
22. homuler/MediaPipeUnityPlugin: Unity plugin to run MediaPipe - GitHub, 访问时间为 十一月 22, 2025, https://github.com/homuler/MediaPipeUnityPlugin
23. How to improve performance holistic graph on windows10? · Issue #244 · homuler/MediaPipeUnityPlugin - GitHub, 访问时间为 十一月 22, 2025, https://github.com/homuler/MediaPipeUnityPlugin/issues/244
24. Named pipes in .NET (C#) - Medium, 访问时间为 十一月 22, 2025, https://medium.com/codenx/named-pipes-in-net-c-c0459e165371
25. Use OpenTrack + AITrack to get headtracking with just a normal webcam - Reddit, 访问时间为 十一月 22, 2025, https://www.reddit.com/r/StarWarsSquadrons/comments/kbpq2s/use_opentrack_aitrack_to_get_headtracking_with/
26. 1523-9926 Technology Interface International Journal - TIIJ, 访问时间为 十一月 22, 2025, https://tiij.org/issues/issues/fall2019/X__TIIJ%20fall%202019%20v20%20n1.pdf
27. sangho0n/Realtime_SMPLX_Unity - GitHub, 访问时间为 十一月 22, 2025, https://github.com/sangho0n/Realtime_SMPLX_Unity
28. [0.x][springbone][center] Imported VRM hair and clothes goes to the left no matter the position · Issue #2440 · vrm-c/UniVRM - GitHub, 访问时间为 十一月 22, 2025, https://github.com/vrm-c/UniVRM/issues/2440
29. Working with calendars and events using the Microsoft Graph API, 访问时间为 十一月 22, 2025, https://learn.microsoft.com/en-us/graph/api/resources/calendar-overview?view=graph-rest-1.0
30. Using Microsoft Graph to retrieve emails and calendar events for your solution - YouTube, 访问时间为 十一月 22, 2025, https://www.youtube.com/watch?v=G4Ej5A3k7FM
31. Get Started with the Outlook REST APIs - Microsoft Learn, 访问时间为 十一月 22, 2025, https://learn.microsoft.com/en-us/outlook/rest/get-started
32. Creating Events in Microsoft Calendar using Microsoft Graph - Stack Overflow, 访问时间为 十一月 22, 2025, https://stackoverflow.com/questions/56537974/creating-events-in-microsoft-calendar-using-microsoft-graph
33. Secure Boundaries: Understanding LLM Sandbox Environments - Sandgarden, 访问时间为 十一月 22, 2025, https://www.sandgarden.com/learn/llm-sandbox
34. Secure code execution - Hugging Face, 访问时间为 十一月 22, 2025, https://huggingface.co/docs/smolagents/en/tutorials/secure_code_execution
35. AppDomain Class (System) - Microsoft Learn, 访问时间为 十一月 22, 2025, https://learn.microsoft.com/en-us/dotnet/api/system.appdomain?view=net-9.0
36. How to sandbox code running another app (started using Process.Start)? - Stack Overflow, 访问时间为 十一月 22, 2025,

https://stackoverflow.com/questions/55228931/how-to-sandbox-code-running-another-app-started-using-process-start
37. Local inter-process communication over named pipes with ASP.NET Core or StreamJsonRpc in .NET - Anthony Simmon, 访问时间为 十一月 22, 2025, https://anthonysimmon.com/local-ipc-over-named-pipes-aspnet-core-streamjsonrpc-dotnet/
38. Suddenly can't get UniVRM to work with Unity? : r/vtubertech - Reddit, 访问时间为 十一月 22, 2025, https://www.reddit.com/r/vtubertech/comments/16ba3bk/suddenly_cant_get_univrm_to_work_with_unity/