3D 版“I Wanna Maker”程序思路构想（使用 unity）

1. 概念概述与灵感来源

本程序希望是一款基于 Unity3D 图形库的 3D 精密平台跳跃游戏。其核心理念来源于 2D 游戏“I Wanna Maker”，目标是在三维空间中重现很有特色的令人血压暴涨的游戏体验。与原版游戏类似，本 3D 版本也将专注于精确的玩家操控和精心设计的关卡，玩家需要通过精准的跳跃和操作来克服各种致命的障碍，最终达成目标 [1]。本游戏希望使用“I Wanna Maker”的核心玩法，即高难度的平台跳跃挑战，玩家需要具备极高的操作技巧和对关卡布局的熟悉程度。关卡中将布满各种设计巧妙（？）的陷阱和敌人，需要玩家在毫厘之间做出正确的判断和操作。同时，我们也会考虑加入一些与原版相似的元素，例如隐藏的收集品（如宝石）和需要特定技巧才能触发的机关，以增加游戏的可玩性和探索性[1]。但与“I Wanna Maker”的 2 维表现形式不同，本程序将采用 3D 视角。这一根本性的改变将带来全新的游戏体验，以及比原版更变态的难度。三维空间将允许更复杂的关卡设计，例如多层结构、环绕式场景以及需要玩家在三维空间中进行精确移动和定位的挑战。虽然游戏的核心玩法将与“I Wanna Maker”保持一致，但 3D 视角可以为我们提供新的游戏机制和视觉风格。

2. 核心游戏玩法机制

2.1 3D 玩家控制系统

玩家在游戏中的基本移动将通过标准的 3D 控制方式实现，使用 WASD 控制角色在水平面（X 轴，Y 轴）上的移动。为了确保游戏的操作精度，角色的移动速度和加速度将被尽我所能的仔细调整，以实现比较跟手的操控感。这对于一款强调精确性的平台跳跃游戏至关重要，任何操作上的延迟或不确定性都可能导致玩家的挫败感（虽然本身可能也会导致挫败感）。除了基本的行走和奔跑，我们还会考虑加入额外的移动能力（比如二段跳），以适应 3D 环境下的挑战。这些额外的移动方式将需要与核心的跳跃机制良好地结合，共同构成玩家在 3D 世界中探索和挑战的基础。

2.2 跳跃机制与其他玩家能力

跳跃是本游戏的核心机制之一。玩家将能够通过按下空格来控制角色的跳跃动作。Unity3D 的物理引擎应该可以负责模拟跳跃的轨迹和重力效果，确保其符合物理规律，并为玩家提供稳定和可预测的跳跃体验。与环境的互动是重要的部分，例如玩家可能需要推动特定的物体来搭建新的道路，或者利用场景中的机关来改变关卡的布局（按钮）。

2.3 与 3D 游戏世界的互动

玩家在游戏中将面临各种危险的障碍，例如尖刺、飞行的弹幕和移动的平台。碰撞检测将判断玩家角色是否与这些元素发生接触。一旦检测到碰撞，游戏将根据预设的规则做出相应的反应，导致玩家角色死亡并需要重新开始关卡。除了障碍，关卡中还将包含各种可互动的元素。例如，玩家可能需要触发开关来打开前进的道路，或者拉动拉杆来改变平台的运动轨迹。收集品，如原版中的宝石，也可能会以 3D 的形式出现在关卡中，鼓励玩家探索隐藏的区域并完成额外的挑战。

3. 3D 视觉风格考量

3.1 潜在的视觉风格

在 3D 视觉风格的选择上，我将考虑简约风格。采用简洁的几何形状和纯粹的色彩构成游戏世界，将视觉干扰降到最低，从而更加突出游戏的挑战性和可玩性 。这种风格可以有效地降低游戏的性能需求，并确保玩家的注意力完全集中在游戏的核心机制上。例如，平台可能只是简单的立方体或长方体，而障碍物则用鲜艳的颜色进行标记，以便于识别。

3.2 选择美术方向的考量

在选择最终的视觉风格时，清晰度将是首要考虑的因素。对于一款高难度的平台跳跃游戏而言，玩家必须能够清晰地辨认出所有的平台、障碍物和互动元素。因此，我们将确保关键的

游戏元素在视觉上是清晰且易于区分的。性能也是一个重要的考量因素，尤其是在追求更复杂的视觉效果时。过高的性能需求可能会限制游戏的受众，并可能导致在某些设备上运行不流畅（虽然我们写的不太可能这么吃配置）。简约风格可能更偏向于纯粹的挑战。

4. 基于 Unity3D 和 C++ 的技术实现

4.1 Unity3D 在 3D 开发中的关键特性

Unity3D 引擎为 3D 游戏开发提供了功能支持[2]。通过光照和阴影系统，我们可以增强 3D 环境的沉浸感。如果需要，我们还可以利用多摄像机系统来实现更高级的渲染效果或用户界面显示。在物理方面，Unity 内置了成熟的 3D 物理引擎 NVIDIA PhysX，可以处理玩家的移动、碰撞检测以及与环境的互动[3]。

4.2 使用 C++ 的理由与方法

本程序计划使用 C++ 语言来开发部分关键的游戏逻辑，其主要目的是为了追求更高的性能（因为是 C++程序设计课），尤其是在处理复杂的计算的场合[4]。采用 C++ 和 Unity3D 混合开发模式的主要优势在于可以充分利用优点。C++ 在性能方面具有显著的优势，尤其适合处理计算密集型的任务，例如复杂的物理模拟、自定义的渲染逻辑或人工智能算法。通过将这些关键部分用 C++ 实现，可以有效地提升游戏的整体性能。此外，使用 C++ 还可以方便地集成现有的 C/C++ 中间件或库，从而节省开发时间和成本。在某些情况下，C++ 还可以提供对底层操作系统功能的更直接访问。调试 C++ 代码可能很困难，因为涉及到更底层的内存管理和平台相关的细节。

5.总结

本程序希望通过 Unity3D 图形库和 C++ 语言的结合，打造一款具有挑战性和趣味性的 3D 精密平台跳跃游戏，其核心理念和难度将与经典的“I Wanna Maker”系列基本保持一致。通过 3D 视角的引入，我们期望能够为玩家带来全新的游戏体验，并探索更多样化的关卡设计和游戏机制。未来的开发工作将包括原型核心玩法的实现、关卡场景设置[5]与描绘以及 bug 修正。长远来看，我希望能加入 3D 关卡编辑器和在线关卡分享功能，借鉴“I Wanna Maker 在用户生成内容方面的成功经验。在整个开发过程中，我将重视迭代开发和测试，不断优化游戏的玩法和体验。

引用出处：

1. I Wanna Maker, https://www.iwannamakergame.com/
2. A Comprehensive Guide to Unity 3D's Features, And Uses - 300Mind, https://300mind.studio/blog/unity-3d-features-guide/
3. Unity3D Physics - Rigidbodies, Colliders, Triggers - YouTube, https://www.youtube.com/watch?v=dLYTwDQmjdo
4. C# vs C++: Complete Comparison Between Unity and Unreal Programming Language, https://www.circuitstream.com/blog/c-vs-c-complete-comparison-between-unity-and-unreal-programming-language
5. Design - Level design..? | GameMaker Community, https://forum.gamemaker.io/index.php?threads/level-design.3038/