UPBGE 从入门到精通：游戏开发全指南

前言

UPBGE（Upgraded Blender Game Engine）作为 Blender 内置游戏引擎的增强版本，凭借其与 Blender 无缝集成的优势，成为独立游戏开发者、学生及爱好者的理想选择。它不仅继承了 Blender 强大的 3D 建模、动画制作功能，还优化了物理引擎、逻辑系统和渲染性能，支持跨平台发布，让开发者能够一站式完成从资源创建到游戏上线的全流程。

本书专为零基础读者和有 Blender 基础但缺乏游戏开发经验的用户编写，遵循 “理论 + 实操” 的教学模式，从基础概念入手，逐步深入核心功能，最终通过实战案例帮助读者掌握独立开发游戏的能力。无论你是想制作 2D 小游戏、3D 冒险游戏，还是交互式体验项目，本书都能为你提供清晰的指导。

第一部分：UPBGE 基础入门

第 1 章：认识 UPBGE

1.1 UPBGE 的起源与优势

UPBGE 源于 Blender 自带的游戏引擎，由社区开发者持续优化升级，解决了原版引擎性能不足、功能有限等问题。其核心优势包括：

- 无缝集成 Blender：无需切换软件，3D 模型、动画、材质可直接用于游戏开发，大幅提升开发效率；
- 强大的物理系统：支持刚体、柔体、流体物理模拟，还原真实的物体交互效果；
- 灵活的逻辑编辑器：提供可视化逻辑砖块和 Python 脚本两种开发方式，兼顾新手友好性和高级扩展性；
- 跨平台兼容性：支持 Windows、macOS、Linux 系统，可发布为独立游戏程序或网页游戏；
- 开源免费：无版权费用，适合个人开发者和小型团队降低开发成本。

1.2 安装与配置 UPBGE

1.2.1 下载 UPBGE

访问 UPBGE 官方网站（https://upbge.org/），根据操作系统选择对应的版本下载。建议选择稳定版（Stable Release），避免开发过程中出现兼容性问题。

1.2.2 安装步骤

- Windows 系统：运行下载的.exe 文件，按照向导选择安装路径，完成后即可启动；
- macOS 系统：将.dmg 文件中的 UPBGE 拖入应用程序文件夹，双击启动；
- Linux 系统：解压.tar.gz 文件，进入解压目录，运行./upbge 即可启动。

1.2.3 界面初识

启动 UPBGE 后，默认界面与 Blender 一致，主要包括以下区域：

- 3D 视图：用于创建、编辑 3D 场景和物体；
- 大纲视图：显示场景中所有物体的层级关系；
- 属性编辑器：用于设置物体、材质、逻辑等属性；
- 逻辑编辑器：编写游戏逻辑的核心区域，分为逻辑砖块和 Python 脚本两种模式；
- 时间线 / 动画编辑器：用于制作动画效果。

第 2 章：基础操作与场景搭建

2.1 核心操作技巧

2.1.1 物体的创建与选择

- 创建物体：在 3D 视图中按 Shift+A，选择要创建的物体类型（如网格、灯光、相机等）；
- 选择物体：左键单击物体即可选择，Shift + 左键可多选；按 A 可全选或取消全选；
- 框选物体：按住左键拖动，框选范围内的物体将被选中。

2.1.2 物体的移动、旋转与缩放

- 移动：选中物体后，按 G 键进入移动模式，拖动鼠标即可移动；按 X/Y/Z 键可限制在对应轴移动；
- 旋转：按 R 键进入旋转模式，拖动鼠标旋转；按 X/Y/Z 键限制旋转轴；
- 缩放：按 S 键进入缩放模式，拖动鼠标缩放；按 X/Y/Z 键限制缩放轴；
- 精确操作：在移动、旋转、缩放时输入数值，可实现精确调整（如 G X 5 表示沿 X 轴移动 5 个单位）。

2.1.3 视图控制

- 旋转视图：按住鼠标中键拖动；
- 平移视图：按住 Shift + 鼠标中键拖动；

- 缩放视图：滚动鼠标中键；
- 切换视角：按数字键 0-9（0 为相机视角，1-9 为不同正交视角）。

2.2 场景搭建流程

2.2.1 场景设置

- 打开属性编辑器，切换到 “场景” 选项卡；
- 设置场景名称、分辨率（渲染属性中的 “分辨率 X/Y”）、帧率（默认 60fps）；
- 启用游戏模式：在属性编辑器 “渲染” 选项卡中，勾选 “游戏引擎” 为 UPBGE。

2.2.2 基础物体添加

- 地面：创建一个平面（Shift+A > 网格 > 平面），按 S 键缩放至合适大小（如 S 10），作为游戏地面；
- 玩家物体：创建一个立方体（Shift+A > 网格 > 立方体），命名为 “Player”，调整位置至地面上方；
- 灯光：创建一盏太阳光（Shift+A > 灯光 > 太阳光），调整角度和强度，确保场景明亮；
- 相机：默认相机已存在，调整相机位置和角度，确保能拍摄到主要游戏区域（按 0 键切换到相机视角预览）。

2.2.3 材质与纹理基础

- 创建材质：选中物体，在属性编辑器切换到 “材质” 选项卡，点击 “新建”，命名材质（如 “GroundMaterial”）；
- 设置颜色：在材质节点编辑器中，添加 “diffuse BSDF” 节点，调整颜色属性；
- 应用材质：点击材质面板中的 “Assign”，将材质应用到选中物体；
- 添加纹理：在材质节点编辑器中添加 “图像纹理” 节点，加载图片文件，连接到 diffuse BSDF 的颜色输入，实现纹理贴图效果。

第二部分：核心功能详解

第 3 章：逻辑系统入门 —— 逻辑砖块

3.1 逻辑砖块的核心概念

逻辑砖块是 UPBGE 中可视化的逻辑编辑工具，无需编写代码即可实现简单游戏逻辑。其核心组成包括：

- 传感器（Sensors）：触发逻辑的条件（如键盘输入、碰撞检测、时间触发等）；
- 控制器（Controllers）：处理传感器信号的逻辑（如与、或、非逻辑，脚本调用等）；
- 执行器（Actuators）：逻辑触发后执行的动作（如物体移动、播放动画、切换场景等）。

逻辑砖块的工作流程：传感器检测到触发条件 → 向控制器发送信号 → 控制器处理信号后向执行器发送指令 → 执行器执行对应动作。

3.2 常用传感器详解

3.2.1 键盘传感器（Keyboard）

- 功能：检测键盘按键的按下、释放或保持状态；
- 配置：在逻辑编辑器中添加键盘传感器，在“按键”选项中选择要检测的按键（如 W、A、S、D、空格等）；
- 触发模式：“按下”（按键按下时触发一次）、“释放”（按键释放时触发一次）、“保持”（按键按住期间持续触发）。

3.2.2 碰撞传感器（Collision）

- 功能：检测当前物体与其他物体的碰撞；
- 配置：添加碰撞传感器，在“属性”中选择要检测的碰撞对象（可选择特定物体或所有物体）；
- 触发模式：“开始碰撞”（首次碰撞时触发）、“持续碰撞”（碰撞期间持续触发）、“结束碰撞”（碰撞结束时触发）。

3.2.3 时间传感器（Timer）

- 功能：按设定时间间隔触发信号；
- 配置：添加时间传感器，设置“间隔”（触发时间间隔，单位秒）、“次数”（触发次数，0 为无限次）；
- 应用场景：定时生成敌人、自动切换画面等。

3.3 常用执行器详解

3.3.1 运动执行器（Motion）

- 功能：控制物体的移动、旋转或速度；
- 配置：添加运动执行器，选择“运动类型”（如位置、旋转、速度），设置对应轴的数值；
- 应用：实现物体的前后左右移动、跳跃等效果。

3.3.2 动画执行器（Animation）

- 功能：播放物体的动画；
- 配置：添加动画执行器，选择要播放的动画片段、播放速度、循环模式；
- 应用：角色走路动画、门开关动画等。

3.3.3 场景执行器（Scene）

- 功能：切换场景、重启场景或结束游戏；
- 配置：添加场景执行器，选择 “类型”（如切换场景、重启场景、结束游戏），切换场景时选择目标场景；
- 应用：游戏开始界面切换到游戏场景、游戏结束时切换到结束界面。

3.4 简单逻辑案例：玩家移动

1. 选中玩家物体（Player），切换到逻辑编辑器；
2. 添加键盘传感器：点击 “添加传感器”→“键盘”，命名为 “MoveForward”，选择按键 “W”，触发模式设为 “保持”；
3. 添加运动执行器：点击 “添加执行器”→“运动”，命名为 “MoveForwardAct”，设置 Z 轴速度为 5（假设 Z 轴为前后方向，可根据场景调整）；
4. 连接逻辑：用鼠标拖动传感器的输出端到执行器的输入端，中间自动添加一个 “与控制器”；
5. 同理，添加 W、A、S、D 四个方向的键盘传感器和对应的运动执行器，实现玩家的前后左右移动；
6. 点击逻辑编辑器中的 “运行游戏” 按钮（ ），在 3D 视图中按 W、A、S、D 即可控制玩家移动。

第 4 章：Python 脚本编程基础

4.1 UPBGE 中的 Python 环境

UPBGE 内置 Python 解释器，支持 Python 3.x 语法，可通过 Python 脚本实现更复杂的游戏逻辑。相比逻辑砖块，Python 脚本更灵活，能实现自定义物理、AI 行为、数据存储等高级功能。

在逻辑编辑器中，切换到 “脚本” 模式，即可创建和编辑 Python 脚本。脚本文件默认保存为.py 格式，可在多个物体或场景中复用。

4.2 核心 API 介绍

4.2.1 bge 模块

bge 是 UPBGE 的核心模块，提供了访问游戏引擎功能的所有接口，常用子模块包括：

- bge.logic：获取游戏逻辑相关信息（如当前场景、控制器、传感器等）；
- bge.scene：场景相关操作（如获取场景中的物体、添加物体、切换场景等）；
- bge.objects：获取场景中的物体，通过物体名称或 ID 访问；
- bge.types：定义了游戏中的核心类型（如 KX_GameObject、KX_Sensor、KX_Controller 等）。

4.2.2 常用对象与方法

- 获取当前场景：`scene = bge.logic.getCurrentScene()`；
- 获取当前物体：`own = controller.owner`（controller 为当前控制器）；
- 获取传感器：`sensor = controller.sensors["传感器名称"]`；
- 控制物体移动：`own.applyMovement([x, y, z], True)`（True 表示局部坐标，False 表示世界坐标）；
- 控制物体旋转：`own.applyRotation([x, y, z], True)`；
- 检测按键：通过键盘传感器的 `positive` 属性判断按键是否被触发（如 `sensor.positive`）。

4.3 脚本案例：玩家移动与跳跃

1. 选中玩家物体，在逻辑编辑器中添加键盘传感器：
    - 名称 “MoveW”，按键 “W”，触发模式 “保持”；
    - 名称 “MoveA”，按键 “A”，触发模式 “保持”；
    - 名称 “MoveS”，按键 “S”，触发模式 “保持”；
    - 名称 “MoveD”，按键 “D”，触发模式 “保持”；
    - 名称 “Jump”，按键 “空格”，触发模式 “按下”。
1. 添加 Python 控制器：点击 “添加控制器”→“Python”，命名为 “PlayerControl”，在 “脚本” 选项中点击 “新建”，创建脚本文件 “player_control.py”；
2. 编写脚本代码：

```
import bge
def main():
```

```
# 获取控制器和当前物体
controller = bge.logic.getCurrentController()
own = controller.owner

# 获取键盘传感器
move_w = controller.sensors["MoveW"]
move_a = controller.sensors["MoveA"]
move_s = controller.sensors["MoveS"]
move_d = controller.sensors["MoveD"]
jump = controller.sensors["Jump"]

# 初始化移动速度和跳跃力
move_speed = 0.1
jump_force = 5.0

# 计算移动方向
move_dir = [0, 0, 0]
if move_w.positive:
    move_dir[2] -= move_speed  # 沿 Z 轴负方向移动（根据场景调整）
if move_s.positive:
    move_dir[2] += move_speed
if move_a.positive:
    move_dir[0] -= move_speed
if move_d.positive:
    move_dir[0] += move_speed

# 应用移动
own.applyMovement(move_dir, True)

# 跳跃逻辑（检测是否在地面上，避免二段跳）
if jump.positive and own.onGround:
```

```
        own.applyForce([0, 0, jump_force], True)
# 运行主函数
main()
```

1. 连接逻辑：将所有键盘传感器的输出端连接到 Python 控制器的输入端；
2. 运行游戏，即可通过 W、A、S、D 控制移动，空格跳跃。

第 5 章：物理系统与碰撞检测

5.1 UPBGE 物理系统概述

UPBGE 集成了 Bullet 物理引擎，支持刚体、柔体、流体、布料等物理模拟，让物体的运动和交互更符合现实规律。物理系统的核心是 “物理类型”，每个物体可设置不同的物理类型以实现不同的物理效果。

5.2 物理类型设置

选中物体，在属性编辑器的 “物理” 选项卡中设置 “物理类型”，常用类型包括：

- 静态（Static）：固定不动的物体（如地面、墙壁），不会被其他物体推动，适合作为场景障碍物；
- 动态（Dynamic）：受重力、力和碰撞影响的物体（如玩家、敌人、道具），可自由移动和交互；
- kinematic（运动学）：由脚本或动画控制运动的物体，不受重力影响，但会与动态物体发生碰撞（如电梯、移动平台）；
- 传感器（Sensor）：不参与物理碰撞，但能检测与其他物体的接触（如触发机关、检测玩家进入区域）。

5.3 碰撞检测与物理属性调整

5.3.1 碰撞形状设置

碰撞形状决定了物体的碰撞检测范围，默认情况下 UPBGE 会根据物体网格自动生成碰撞形状，也可手动设置：

- 在 “物理” 选项卡的 “碰撞形状” 中选择形状类型（如盒子、球体、胶囊体、凸包、网格）；
- 简单物体（如立方体、球体）推荐使用盒子、球体等简单碰撞形状，提升性能；
- 复杂物体（如角色模型）推荐使用胶囊体或凸包碰撞形状，平衡精度和性能。

5.3.2 物理属性调整

- 质量：动态物体的质量（单位 kg），质量越大，惯性越大，越难被推动；
- 摩擦：物体表面的摩擦系数，值越大，滑动时减速越快；
- 恢复系数：物体碰撞后的反弹程度，值越大，反弹越高（0 为不反弹，1 为完全反弹）；
- 重力：场景全局重力可在 “场景” 属性的 “物理” 选项中设置，单个物体可在 “物理” 选项卡中勾选 “自定义重力” 设置单独重力。

5.4 物理系统案例：弹跳平台

1. 创建地面（静态物理类型）和一个立方体（命名为 “BouncePlatform”，动态物理类型）；
2. 选中弹跳平台，设置物理属性：质量 5，恢复系数 0.8，摩擦 0.2；
3. 添加碰撞传感器：检测与玩家的碰撞，触发模式 “开始碰撞”；
4. 添加 Python 控制器，编写脚本实现碰撞后施加向上的力：

```
import bge
def main():
    controller = bge.logic.getCurrentController()
    own = controller.owner
    collision_sensor = controller.sensors["Collision"]

    if collision_sensor.positive:
        # 向碰撞对象施加向上的力
        for obj in collision_sensor.hitObjectList:
            if obj.name == "Player":
                obj.applyForce([0, 0, 10], True)
main()
```

1. 运行游戏，玩家碰撞弹跳平台后将被向上弹起。

第三部分：进阶功能与实战案例

第 6 章：动画系统与角色控制

6.1 动画制作基础

6.1.1 关键帧动画创建

- 选中物体，在时间线中设置起始帧（如 1），调整物体的位置、旋转或缩放；
- 按 I 键，选择要记录的动画属性（如位置、旋转、缩放），创建关键帧；
- 移动时间滑块到目标帧（如 30），调整物体属性，再次按 I 键创建关键帧；
- 播放动画：按 Alt+A 在 3D 视图中预览动画。

6.1.2 动画编辑器使用

在动画编辑器中可精确调整动画曲线：

- 选择物体和动画属性，编辑关键帧的位置和插值方式；
- 调整 “缓入”“缓出” 效果，让动画更流畅；
- 复制、粘贴关键帧，批量编辑动画。

6.2 角色动画控制

6.2.1 骨骼动画导入与设置

- 导入带有骨骼动画的角色模型（如 FBX、OBJ 格式）；
- 在属性编辑器的 “对象数据属性” 中，确保骨骼动画被正确识别；
- 在逻辑编辑器中添加 “动画执行器”，选择要播放的动画片段。

6.2.2 动画切换逻辑

通过 Python 脚本根据玩家行为切换动画（如走路、跑步、跳跃）：

```
import bge
def main():
    controller = bge.logic.getCurrentController()
    own = controller.owner

    # 获取键盘传感器
    move_w = controller.sensors["MoveW"]
    jump = controller.sensors["Jump"]

    # 获取动画执行器
```

```
    walk_anim = controller.actuators["WalkAnim"]
    idle_anim = controller.actuators["IdleAnim"]
    jump_anim = controller.actuators["JumpAnim"]

    # 动画切换逻辑
    if jump.positive:
        controller.activate(jump_anim)
        controller.deactivate(walk_anim)
        controller.deactivate(idle_anim)
    elif move_w.positive:
        controller.activate(walk_anim)
        controller.deactivate(idle_anim)
        controller.deactivate(jump_anim)
    else:
        controller.activate(idle_anim)
        controller.deactivate(walk_anim)
        controller.deactivate(jump_anim)
main()
```

第 7 章：UI 界面设计与交互

7.1 UI 元素创建

UPBGE 的 UI 系统基于 Blender 的 “纹理” 和 “字体” 功能，可通过以下方式创建 UI：

- 使用 2D 纹理作为 UI 背景、按钮图标；
- 在 “属性编辑器” 的 “世界” 或 “物体” 属性中添加 “UI 纹理”；
- 使用 “文本物体” 创建 UI 文字（Shift+A > 文本），设置字体、大小和颜色。

7.2 UI 交互逻辑

通过 “鼠标传感器” 检测鼠标点击、悬停等操作，实现 UI 交互：

1. 创建一个平面物体作为按钮，添加纹理作为按钮图标；
2. 添加鼠标传感器：检测 “鼠标点击” 和 “鼠标悬停” 事件；
3. 添加 Python 控制器，编写脚本实现点击按钮切换场景：

```
import bge
def main():
    controller = bge.logic.getCurrentController()
    own = controller.owner
    mouse_click = controller.sensors["MouseClick"]

    if mouse_click.positive:
        # 切换到游戏场景
        bge.logic.getCurrentScene().replace("GameScene")
main()
```

第 8 章：实战案例 ——2D 平台跳跃游戏

8.1 游戏规划

- 游戏类型：2D 横版平台跳跃游戏；
- 核心玩法：玩家控制角色在平台间跳跃，收集道具，躲避障碍物，到达终点；
- 技术要点：2D 视角设置、角色移动与跳跃、碰撞检测、道具收集、UI 计分、场景切换。

8.2 开发步骤

8.2.1 场景搭建

1. 设置 2D 视角：在 3D 视图中按 5 切换到正交视角，按 1 切换到前视图，调整相机位置，确保场景呈 2D 效果；
2. 创建地面和平台：使用平面物体创建多个平台（静态物理类型），分布在场景中；
3. 添加玩家：创建一个胶囊体（动态物理类型），命名为 “Player”，设置碰撞形状为胶囊体；
4. 添加道具：创建球体物体（动态物理类型），命名为 “Coin”，添加金色材质；
5. 添加障碍物：创建红色立方体（静态物理类型），命名为 “Obstacle”；
6. 添加终点：创建一个绿色平面（传感器物理类型），命名为 “FinishLine”。

8.2.2 角色控制逻辑

参考第 4 章 Python 脚本案例，实现 W（跳跃）、A（左移）、D（右移）控制，优化跳跃逻辑（限制二段跳）：

```
import bge
def main():
    controller = bge.logic.getCurrentController()
    own = controller.owner

    # 传感器
    left = controller.sensors["Left"]
    right = controller.sensors["Right"]
    jump = controller.sensors["Jump"]

    # 移动参数
    move_speed = 0.15
    jump_force = 6.0
    move_dir = [0, 0, 0]

    # 左右移动
    if left.positive:
        move_dir[0] -= move_speed
    if right.positive:
        move_dir[0] += move_speed

    own.applyMovement(move_dir, True)

    # 跳跃（限制二段跳）
    if jump.positive and own.onGround:
        own.applyForce([0, 0, jump_force], True)
main()
```

8.2.3 道具收集与计分

1. 给玩家添加碰撞传感器，检测与道具（Coin）的碰撞；
2. 在 Python 脚本中添加计分逻辑，使用全局变量存储分数：

```
import bge
# 初始化分数
if "score" not in bge.logic.globalDict:
    bge.logic.globalDict["score"] = 0
def main():
    controller = bge.logic.getCurrentController()
    own = controller.owner
    coin_collision = controller.sensors["CoinCollision"]

    if coin_collision.positive:
        # 收集道具，分数+1，删除道具
        bge.logic.globalDict["score"] += 1
        coin_collision.hitObject.endObject()
        print("Score:", bge.logic.globalDict["score"])
main()
```

8.2.4 UI 计分与场景切换

1. 创建 UI 文本物体，显示分数；
2. 添加 Python 控制器，实时更新分数显示：

```
import bge
def main():
    controller = bge.logic.getCurrentController()
    own = controller.owner

    # 更新分数显示
    own.text = "Score: " + str(bge.logic.globalDict.get("score", 0))
main()
```

1. 给终点（FinishLine）添加碰撞传感器，检测玩家碰撞后切换到结束场景：

```
import bge
def main():
    controller = bge.logic.getCurrentController()
    own = controller.owner
    finish_collision = controller.sensors["FinishCollision"]

    if finish_collision.positive and finish_collision.hitObject.name == "Player":
        # 切换到结束场景
        bge.logic.getCurrentScene().replace("EndScene")
main()
```

8.2.5 测试与优化

1. 运行游戏，测试角色移动、跳跃、道具收集、碰撞检测等功能；
2. 优化物理属性（如调整玩家质量、重力、平台摩擦），使操作更流畅；
3. 优化场景性能，删除多余物体，简化复杂物体的碰撞形状；
4. 添加音效（使用 “声音执行器” 播放音效文件），提升游戏体验。

8.3 发布游戏

1. 在属性编辑器的 “渲染” 选项卡中，设置游戏分辨率、图标等；
2. 点击 “游戏” 菜单 > “发布游戏”，选择发布平台（Windows、macOS、Linux）；
3. 设置发布路径和文件名，点击 “发布”，UPBGE 将生成独立游戏程序。

第四部分：高级技巧与资源拓展

第 9 章：性能优化技巧

9.1 场景优化

- 简化模型：减少场景中物体的多边形数量，删除不必要的细节；
- 合并物体：将多个静态物体合并为一个（选中物体按 Ctrl+J），减少碰撞检测和渲染开销；

- 隐藏远处物体：使用 “距离传感器” 或 Python 脚本，隐藏距离相机过远的物体。

9.2 物理优化

- 使用简单碰撞形状：优先选择盒子、球体等简单碰撞形状，避免使用网格碰撞形状；
- 减少动态物体数量：尽量将静止物体设为静态物理类型，动态物体数量控制在合理范围；
- 调整物理精度：在 “场景” 属性的 “物理” 选项中，降低 “物理精度”（默认 60，可设为 30-40），提升性能。

9.3 渲染优化

- 降低纹理分辨率：将大尺寸纹理压缩为合适分辨率（如 1024x1024）；
- 关闭不必要的渲染效果：如阴影、反射、折射等，仅在关键物体上启用；
- 使用 LOD（细节层次）：为复杂模型创建多个细节版本，远处显示低细节模型。

第 10 章：资源拓展与社区支持

10.1 常用资源网站

- UPBGE 官方文档：https://upbge.org/docs/（权威教程和 API 参考）；
- Blender 资源库：https://www.blenderkit.com/（免费 3D 模型、材质、纹理）；
- 音效资源：https://freesound.org/（免费音效和音乐）；
- 字体资源：https://www.dafont.com/（免费游戏字体）。

10.2 社区支持

- UPBGE 论坛：https://forum.upbge.org/（提问、分享作品和经验）；
- Blender 中文社区：https://www.blendercn.org/（中文教程和交流）；
- GitHub 仓库：https://github.com/UPBGE/upbge（提交 bug、参与开发）。

后记

UPBGE 作为一款开源免费的游戏引擎，凭借与 Blender 的深度集成，为开发者提供了高效便捷的游戏开发解决方案。本书从基础入门到进阶实战，系统讲解了 UPBGE 的核心功能和开发技巧，希望能帮助你快速掌握游戏开发能力。

游戏开发是一个不断学习和实践的过程，建议你在学习本书后，尝试开发自己的小游戏，不断探索和拓展 UPBGE 的功能。如果在开发过程中遇到问题，可通过社区论坛寻求帮助，也可关注 UPBGE 的官方更新，了解最新功能和优化。

祝你在游戏开发的道路上取得成功!

> （注：文档部分内容可能由 AI 生成）