PORTAL : REMOTE LAB

概要设计文档

2018 年 Java 程序设计项目

本游戏参考《Portal》系列游戏的基本设定，为同人衍生作品，不做任何商业用途

赵元培 马纳波 王勇程

小组成员

目录

游戏设计文档

游戏梗概

欢迎来到光圈科技！本游戏中，您将扮演光圈科技公司的一名测试人员，通过您的电脑连接至光圈传送门远程测试系统。光圈科技公司着力于可以打出能瞬间移动的传送门的传送枪以及可以产生传送门的特殊墙体材料，您需要对传送门的性能进行测试，不仅如此，您还需要利用这些传送门的特点在模拟场景中进行地形解谜，最终达到测试终点。那么，请开始测试吧！

游戏流大纲

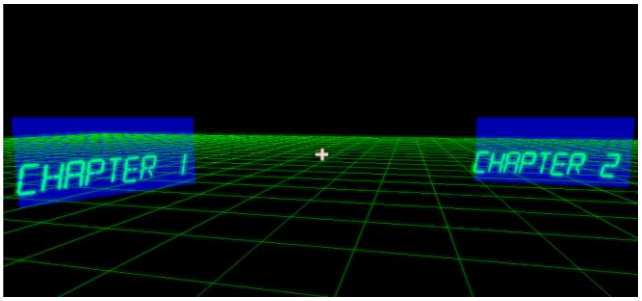
《Portal : Remote Lab》是一款 3D 第一人称解谜游戏，主角是一名光圈科技测试人员，在模拟系统中对传送门系列设备进行测试。玩家可以通过手中的传送枪在特定的墙面上打出两个相连的传送门，以此在两个传送门之间进行瞬间移动，到达本来达不到的地方；玩家在穿越传送门时的动量得以保留，玩家有时也需要巧妙利用这一点穿越地形；同时地形中还存在需要特定箱子的按压式机关、在其上行走可以加速的加速地形和可以获得更高弹跳的弹跳地形，玩家需要充分利用和结合这些元素进行测试通关。

操作

使用 WASD 键进行前后左右移动；使用鼠标进行视角旋转；使用鼠标左右键打出 A/B 传送门；使用 E 键拾取物体；使用 space 键进行跳跃；在选择关卡画面中使用 space 键进行选择；在游戏关卡中使用 F1 键退回选关画面，当前游戏进度不保存。

游戏体验

《Portal : Remote Lab》的开始菜单（即选关界面）是 3D 的，玩家通过旋转视角让视线对准相应关卡，选关界面设计具有浓厚的科技感和电子感，3D 的界面结合电子背景音乐，给玩家不一样的界面交互体验；同时还有机器人声对玩家进行背景介绍，让玩家充分进入游戏设定中。



选关界面

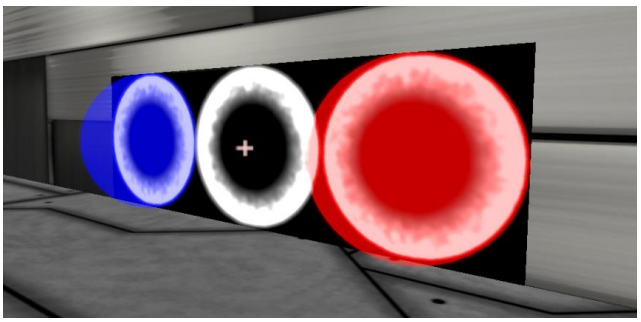
进入关卡后，随着机器人声对玩家进行操作说明和关卡说明，玩家可以看到该关卡的界面基本要素。本游戏的整体画面风格偏向简约抽象和未来金属质感，配合电子感十足的音乐，玩家可以在解谜的同时充分感受该艺术风格。在玩家成功达到本关卡终点时，玩家脚下会升起一个半透明台面，托举玩家上升，画面随之渐渐变暗，随后玩家成功过关，到达下一关卡。当玩家通过所有关卡后，游戏会滚出鸣谢名单，伴随着《Portal 2》游戏原声音乐结束游戏。

关卡设置

本游戏共有 6 个关卡，难度从简到难逐步上升，每个关卡均包含不同的游戏元素，列举如下。

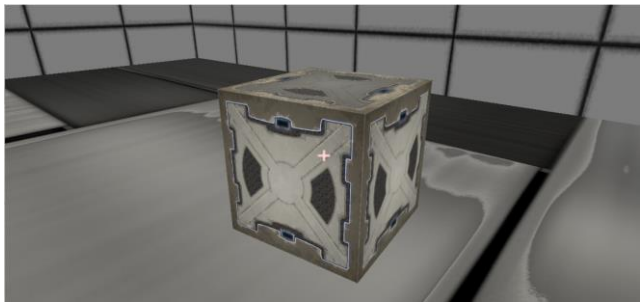
传送门

玩家鼠标左右键可以打两个传送门（颜色不同以示区分），两个传送门之间可以相互传送，然而传送门只能打在特定的墙面（图中有白色光环的地方）



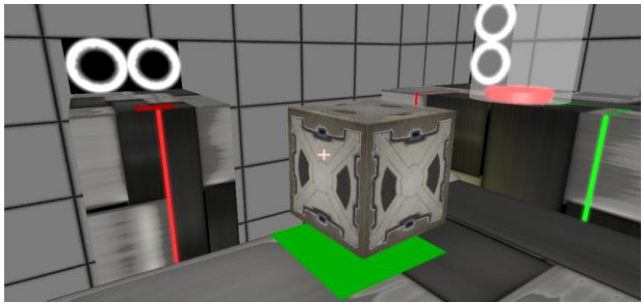
重力箱

箱子具有金属质感，用于按压机关以启动出口，玩家可以拾取放下箱子，值得注意的是，箱子也可以通过传送门，并且也保留动量。



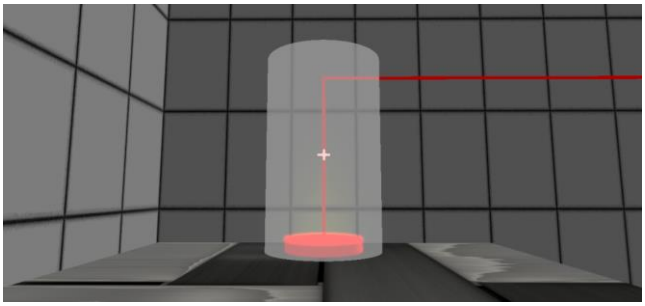
按压式机关

机关和导线使用发光材质，在没有激活时为红色，激活时为绿色，如下图所示，可以存在多个机关共同控制一个出口，当所有机关全部激活时，出口打开。



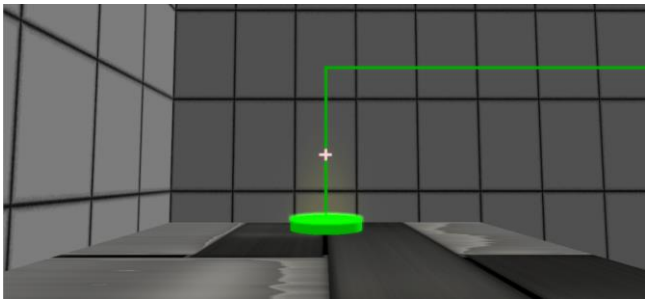
出口（关闭）

关闭时的出口底座为红色，四周有一圈半透明墙壁格挡玩家进入，玩家需要通过激活开关来打开出口。激活所有开关后防护罩会缓慢下降，机关被关闭后防护罩又会立刻上升。



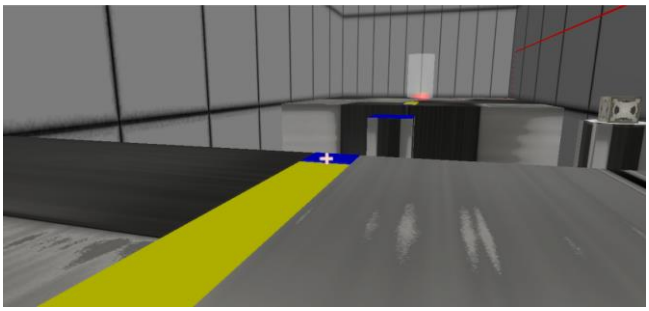
出口（开启）

开启时的出口是绿色的，同时有发出光源，高亮显示。



加速地形和弹跳地形

黄色地面为加速地形，玩家在其上行走是速度会加速；蓝色地面为弹跳地形，玩家在其上弹跳能力会加强，同时其他物体掉落其上也会有弹跳效果。



技术设计文档

运行环境

操作系统	Mac OS X, Windows, Linux, Solaris
内存 (JVM heap size)	> 200 MB
CPU	> 1 GHz
显卡	AMD/ATI Radeon 9500, NVIDIA GeForce 5 FX, Intel GMA 4500 或更高性能的显卡
Java 运行环境	Java 5 及以上 需要 JVM 来运行 jME 游戏.

开发环境

操作系统	**Mac OS X, Windows, Linux, Solaris**
内存 (JVM heap size)	> 200 MB
CPU	> 1 GHz
显卡	AMD/ATI Radeon 9500, NVIDIA GeForce 5 FX, Intel GMA 4500 或更高性能的显卡
JDK	JDK 7 或更高

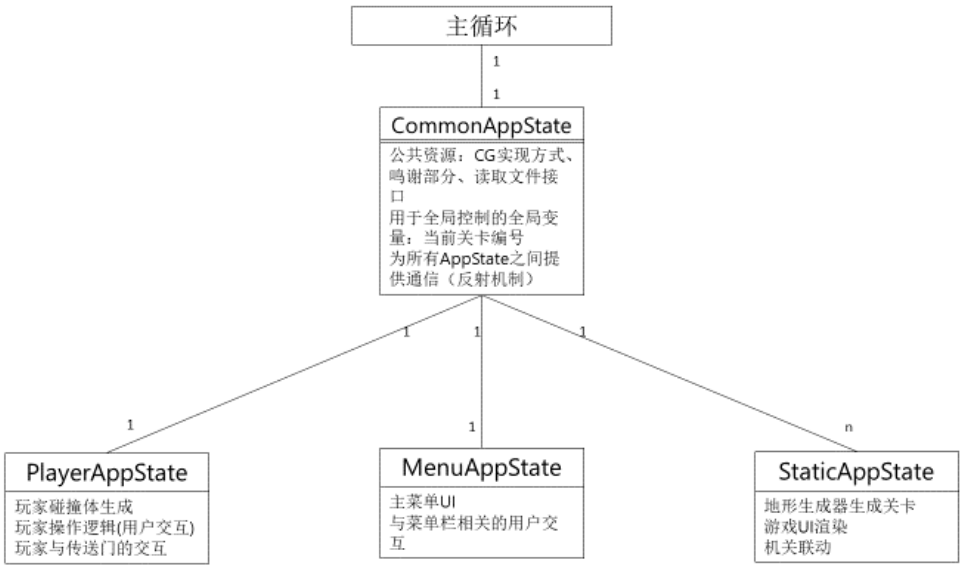
基本设计概念

游戏框架

基于组件的设计架构，我们将游戏分为 4 个组件:

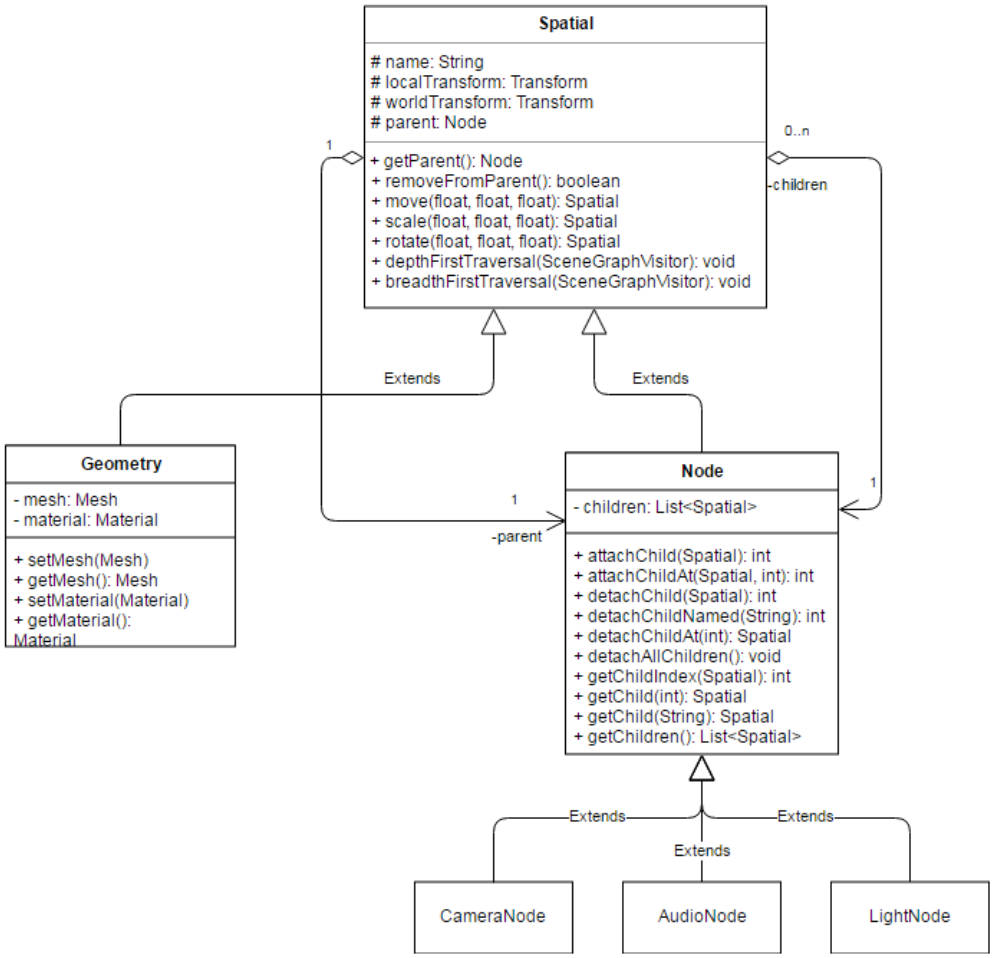
CommonAppState、PlayerAppState、StaticAppState、menuAppState

其中 CommonAppState 主要为 Main 和各个组件之间提供用于控制流程的全局变量，Main 负责调度资源，切换 AppState 场景。每个 AppState 独自管理自己的资源诸如模型、音频，监听处理自己的事件。



JME3 场景图

jMonkeyEngine3 的游戏场景图通过 Spatial、Geometry、Node 这 3 个类来实现，它们之间的关系如下图:



场景图(Scene Graph)是一种数据结构，用于管理游戏场景中的物体，场景中的每个物体都被称为 Spatial。

Spatial 表示 3D 空间中的一个物体，它在空间中有三种线性变换：位移(Translation)、旋转(Rotation)、缩放(Scale)。Spatial 是 Geometry 和 Node 的父类。

Geometry 存储了物体的网格和材质，代表游戏中的可视物体。

Node 是一个空间中的节点，每个节点(Node)中可以包含多个 Spatial，每个 Spatial 只能有一个父节点(Node)。Node 之间通过父子关系形成树形结构。

接口设计

外部接口

物理引擎采用开源的物理模拟计算引擎 bullet，详见 JME 文档

内部接口

CommonAppState

用于保存全局变量，提供公共方法

函数签名	用途
String[] readFileByLines(String fileName)	读取指定的文件
void setTrustValue(float v)	设置电梯高度
void PlayGratituded()	唤出鸣谢

BaseAppState

BaseAppState 是 jME3 的一个重要接口，主要用于处理全局的游戏机制、场景管理

文档部分接口：

函数签名	用途
initialize(asm,app)	When this AppState is added to the game, the RenderThread initializes the AppState and then calls this method.
cleanup()	子场景实例被移除后，即结束生命周期，清理子场景里的各种资源如监听器，GUI
update(float tpf)	子场景主循环，用于实现子场景自己的游戏逻辑
setActive(true) setActive(false)	设置场景状态，设置为 true 时将当前场景设置为可见
isActive()	查看当前场景的状态