Portal : Remote Lab 项目报告

Java 程序设计课程项目

组长：赵元培

组员：马纳波、王勇程

创意来源

Steam 上的一款经典解谜游戏《PORTAL》。玩家以第一视角在三维空间内进行探索，从起点开始游戏，通过使用各种道具、利用地形等手段地图内的破解机关，最终到达出口位置完成挑战。游戏由易到难，不仅需要玩家大开脑洞，还需要挑战玩家的游戏操控技巧。

基本逻辑

玩家手持特制的射线枪，可在地图内的一些墙壁上打出至多 2 个传送门。玩家走入一个传送门后可被传送至另一个传送门，同时保留对入口传送门的相对速度。

本游戏主要道具有加速带、弹跳块、机关块等。同时重力与速度的利用作为两个重要因素需要玩家在操作和通关的时候全盘考虑。

1. 物理引擎简介

1.1 物理引擎名称：Jmonkey Engine

JMonkeyEngine(简称 jME)是一款免费、开源的游戏引擎，专门为那些希望用最新技术创作 3D 游戏的 Java 程序猿而制作。该引擎采用纯 Java 开发，目的是让更多人能够使用，并且软件部署时间也更短。同时，Jmonkey 本身也提供了一个 IDE，方便了在 JME 引擎下游戏的制作。

1.2 优点

兼容性、跨平台、跨设备的运行能力；

jME 游戏可以发布在 Windows，Mac 和 Linux 系统上，也可以发布在 Android 和 iOS 系统上。

1.3 设计方式

JMonkey 与可视化的 RPG Maker 或 FPS modder 不同，jME3 虽然提供了丰富的组件，但是组件的调用和实现细节需要通过自己编程实现。无法通过简单的可视化操作完成。

1.4 游戏组件

丰富的 3D 组件及详细的文档支持。3D 组件包含了材质、刚体设定、速度、位置、自定义控件等多种物理建模必须的属性，而且组件的可扩展性极强，开发者可以通过继承、重写等方式开发自己的游戏道具和其特殊的属性。

2. Jmonkey 基本模块介绍

2.1 Geometry

Geometry 是一个基本类，是场景中唯一可见的对象，每个对象的 Geometry 都唯一存储着它的几何属性。

2.2 Rigid&Spatial

Geomatry 继承了 Rigid 类。Rigid 对象可通过 add 方法添加到 Geometry 对象中。拥有 Rigid 属性的 Geometry 类将在空间中成为刚体，从而可以获得碰撞检测等方法。Rigid 的几何属性需要自己设定。而 Spatial 则存储着物体的空间属性，譬如 location, rotation, scale 等属性。

2.3 Space（Local / World)

我们所有的几何体都被添加到一个 Space 空间内，每个 Space 内的几何体都有自己的空间和相对位置。如果游戏中有多个 Space 空间，则可以通过 World Space 空间，来获得几何体在整个游戏内的绝对位置。

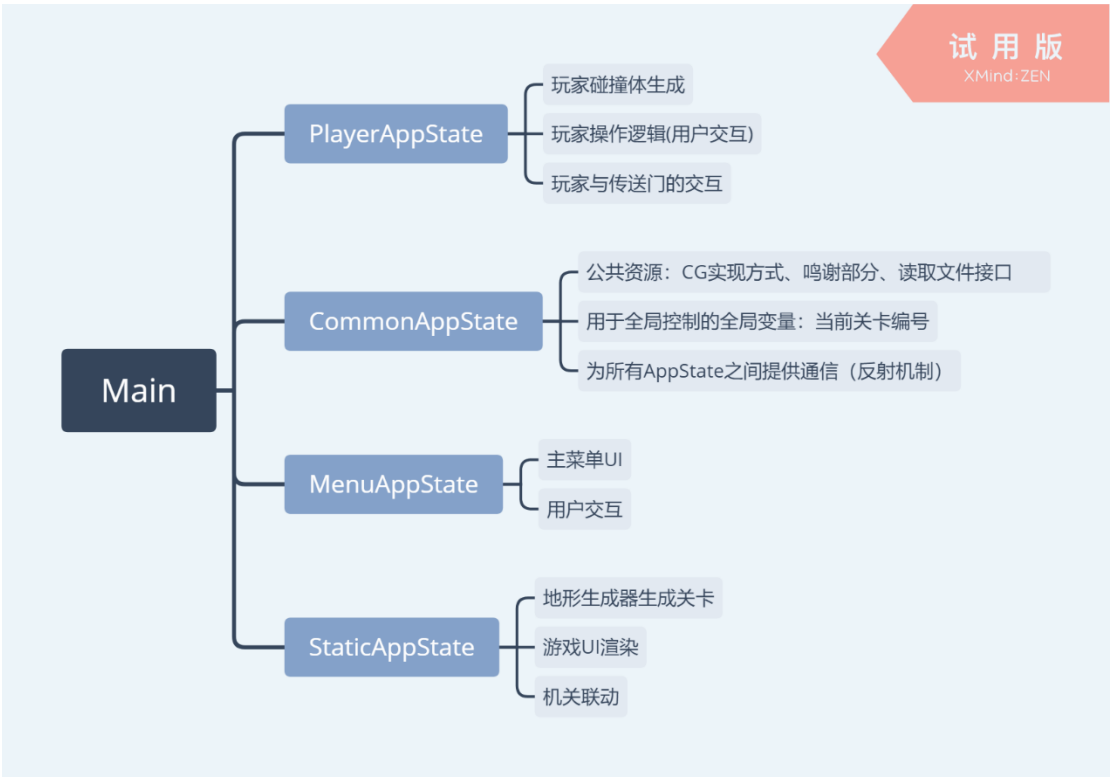
2.4. Node

游戏空间内的所有 Geometry 类最终需要被添加至 rootNode/Node 中进行管理，并显示到游戏中。Node 是一个不可见的对象，但处理着场景中各种逻辑关系。

2.5 Camera

游戏玩家的视角，默认响应键盘“WASD”4 键的控制及鼠标的控制，从而实现视角的移动。

游戏框架


试 用 版
XMind:ZEN
Main
PlayerAppState
玩家碰撞体生成
玩家操作逻辑(用户交互)
玩家与传送门的交互
CommonAppState
公共资源：CG实现方式、鸣谢部分、读取文件接口
用于全局控制的全局变量：当前关卡编号
为所有AppState之间提供通信（反射机制）
MenuAppState
主菜单UI
用户交互
StaticAppState
地形生成器生成关卡
游戏UI渲染
机关联动

游戏核心逻辑


试 用 版
XMind:ZEN
游戏核心逻辑
移动与视角逻辑
视野变换逻辑
Player移动逻辑
跳跃逻辑
传送门逻辑
传送门设置
Player传送
道具传送
道具设计逻辑
加速带
弹跳块
机关块

1. 移动与视角逻辑

1.1 视角变换逻辑

视野变换逻辑：Jmonkey 为我们提供了一个 Camera 类，Camara 是游戏控制者的主视角。Camera 可响应键盘“WASD”4 键的控制及鼠标的控制，从而实现视角的转换。

1.2 Player 移动逻辑

Player 移动逻辑：我们设计了一个控制单元实现了键盘对玩家的控制。绑定“WASD“四键，分别对我上、左、下、右四个移动方向。

当程序监听到对应键盘事件时，先判断玩家按着哪个键，确定移动方向。玩家位置 = 原位置 + 移动方向*玩家速度。

1.3 跳跃逻辑

绑定空间键作为跳跃控制。当程序监听到空格键时，赋予玩家一个向上的初速度。同时，为防止连续跳跃操作，在玩家有竖直速度的时候，对监听到的空格事件不作处理。

当玩家在空中时，玩家仍然可以前后左右移动，但是我们弱化了玩家的移动速度，使其水平位移只有地面效应的一半。即玩家位置 = 原位置 + 移动方向*玩家速度*0.5。

2. 传送门逻辑

2.1 传送门设置

首先，我们的所有墙壁都是由边长为 2 的正方形铺成的。墙壁有一个属性来判断它是普通的墙壁还是可以打传送门的墙壁。

我们从 Camera 主视角出发，发出了一条不可见的射线。这条射线从 Player 的几何中心发射直至足够远处。检查这条射线一路上碰撞到的各个几何体，用一个 private Geometry 变量保存下离射线起点最近的可以打传送门的那块正方形墙壁。通过几何关系计算得到正方形中心的坐标之后，再把传送门（球体）的集合中心设置在该正方形的几何中心处。由于传送门半径=1，恰好与边长为 2 的正方形墙壁相切。

2.2 Player 传送逻辑

传送时的空间位置关系：我们的传送门是一个半径为 1.5 的几何球体。我们实时判断玩家的几何中心位置与两个传送门中心的几何距离。当这个几何距离小于一个阈值(1.7 个单位长度)的时候，就将玩家传送至另一个传送门前。为了防止出现传送的死循环，我们把 Player 在第二个传送门前往外“推”了一点，使其与传送门距离等于 1.8 个单位长度。

传送之后速度的变化：在前一个传送门进行传送前，保存玩家在 x,y,z 轴三个方向的速率。在玩家从第二个传送门出来以后，按照相对位置恢复玩家的速度。

传送之后视角的变化：对于竖直的墙壁，玩家被传送出来以后第一视角与墙壁的法向量方向一致。对于地面或者天花板墙壁，为了方便操作，玩家被传送出来以后视角是水平的，而不是竖直着朝天或者朝地。

2.3 道具传送逻辑

与 Player 传送逻辑类似。有区别的是道具没有速度和视角的变化。

3. 道具设计逻辑

3.1 加速带设计逻辑

当 Player 在加速带上时，赋予其运动方向上更大的水平速度。

3.2 弹跳块设计逻辑

使玩家或道具在竖直方向上获得一个向上的速度增量

3.3 机关块设计逻辑

机关块控制着终点出口的开启与关闭。游戏开始时机关块为红色，玩家需要抓取方块置于机关块上方，此时机关块变绿，并解除终点出口的屏障。这里用到的核心逻辑也是方块与机关地板的距离检测。

4. 遇到的问题

4.1 Player 高度 Bug

我们设置的 player 是一个半径为 1.5 的球体。但是玩家实际上在地面上的距离并非 1.5，而是一个在 1.4 波动的浮点数。注意到这个问题之后，我们对各个逻辑进行了修正，解决了这个问题。

4.2 视角 Bug

当玩家以水平视角走出传送门时，如果鼠标不移动，会看到脚底下地图的另一面（因为视角与地板完全平行） 为了解决这个问题，我们用了一个 trick，让玩家微微抬头 0.01，解决了这个问题。

4.3 物块传送 Bug

将物块推出传送门时，物块有时会在没有支撑物的情况下悬在半空且不下落。为了解决这个问题，我们用了一个 trick，给予了物块一个极小的速度，解决了这个问题。

4.4 传送前后速度单位不一致

玩家在水平面的运动速度（0.1f）和竖直跳起初始速度（9.9f）的单位速度不统一，会造成通过传送门后速度过大从而导致飞出场景。在传送石，单独对竖直方向的速度进行修正。

5. 待实现功能

5.1 玩家自定义地图编辑

我们的程序已经实现了一个地图读取器功能，所有的场景布局都是通过读取地图文件实现的。玩家今后可按照我们地图文本文件的格式自定义地图文件，从而创建自己的地图。

5.2 更多的道具设计

亟待开发~