|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 物理量 | 测量数据 | | | 平均值 |
| 原地跳跃高度 | 2.35cm | | | 2.35cm |
| 原地跳跃用时 | 0.92s | 0.94s | 0.97s | 0.94s |

牛顿运动定律在《蛋仔派对》中得到了较好的运用。通过测量原地跳跃相关的的物理量，由可计算出蛋仔岛上的重力加速度g为7.09m/s2；虽然这与现实生活中的重力加速度数值有一定差异，但也足够解释蛋仔的一般行为，如跳跃、移动等；另外，向前跳跃与原地起跳的高度的差异也与现实中起跳现象相符，使得游戏的真实感大大提升。

其次，我们分析弹簧绳的使用中涉及到的物理模型。

弹簧绳是《蛋仔派对》中一种经典的道具，通过向前扔出勾爪，弹簧绳伸长产生弹力，进而对蛋仔产生向前的拉力，若蛋仔在勾爪落地的瞬间起跳，则能获得一个较大的向前的速度，到弹簧恢复原长时再次跳跃，利用惯性仍然能达到比一般起跳更远的距离；弹簧绳的一般使用已经能达到较好的效果，但根据胡克定律，在弹簧绳的弹性范围内，伸长量越大，弹力越大。试想若能通过一些方法增大弹簧绳的伸长量，就能把弹簧绳的使用发挥到极致。在此提供一下方法：第一，在扔出弹簧绳后向后回拉行走一段距离，可直接增大其伸长量；第二，扔出勾爪前起跳，由于勾爪在空中飞行的时间更长，使得弹簧的伸长量也变长。

|  |  |  |  |
| --- | --- | --- | --- |
| 物理量 | 移动速度 | 移动一段距离用时 | 滚动相同距离用时 |
| 测量数据 | 4.57cm/s2 | 8.05s | 4.30s |

结论

通过分析游戏中牛顿力学、动量、刚体力学、冲击波、光学和电学方面的一些物理模型，发现《蛋仔派对》的建模基本符合现实世界的物理规律，体系较为完备，但在很多方面也有进步的空间，如喷泉的高度、悬空向前扑等。物理的、科学的建模在游戏中占据十分重要的地位，能使玩家获得更逼真的游戏体验，因此分析游戏中的物理模型，并对其做出的客观评价是有意义的。

光学

当光照射到物体表面时，会发生反射、吸收、衍射、折射、干涉等现象。部分光被物体吸收转化为其他形式的能量，部分光经过反射、透射、进入人的视觉系统，使我们能够看到物体，在游戏的建模中，为了模拟这种现象，制作者会建立一些简化的物理模型来创建更真实的游戏体验。例如，折射和反射是一种用于在游戏中增强透明物体效果的技术。它通过计算光线在透明物体表面的反射和折射来模拟透明物体的真实外观；环境光遮蔽是一种用于在游戏场景中增强阴影效果的技术，通过在物体表面计算光线的遮蔽程度，来模拟光线在环境中的传播。

波动光学中的偏振、干涉、衍射现象要比粒子光学中的反射、折射复杂得多，并且是比较微观的现象，宏观维度下比较难观察，所以游戏中一般不考虑波动光学的现象。但柔和的阴影（衍射原理）是游戏中需要考虑的问题。

（插图 一些光影）

import pygame

# 初始化 Pygame

pygame.init()

# 设置屏幕大小

screen\_width = 800

screen\_height = 600

screen = pygame.display.set\_mode((screen\_width, screen\_height))

# 设置人物的位置和大小

player\_width = 50

player\_height = 50

player\_x = screen\_width // 2 - player\_width // 2

player\_y = screen\_height - player\_height

# 设置人物的移动速度

player\_speed = 5

# 游戏循环

running = True

while running:

# 处理事件

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

# 处理键盘输入

keys = pygame.key.get\_pressed()

if keys[pygame.K\_LEFT]:

player\_x -= player\_speed

if keys[pygame.K\_RIGHT]:

player\_x += player\_speed

if keys[pygame.K\_UP]:

player\_y -= player\_speed

if keys[pygame.K\_DOWN]:

player\_y += player\_speed

# 绘制人物和背景

screen.fill((255, 255, 255))

pygame.draw.rect(screen, (0, 0, 255), (player\_x, player\_y, player\_width, player\_height))

pygame.display.update()

# 退出 Pygame

pygame.quit()