**成绩**



**现代人机交互**

**项目作业**

**题　　目： 机器人游戏中键盘和游戏**

**手柄的效率差异分析**

**学 院： 计算机学院**

**专业名称： 计算机科学与技术**

**学 号： 1120172988**

**姓 名： 刘若轩**

**任课教师： 梁玮**

**评 阅 人：**

# 机器人游戏中键盘和游戏手柄的效率差异分析实验报告

**一、实验目的**

分析机器人游戏在不同交互方式（键盘、游戏手柄）中用户控制人物移动的效率差异

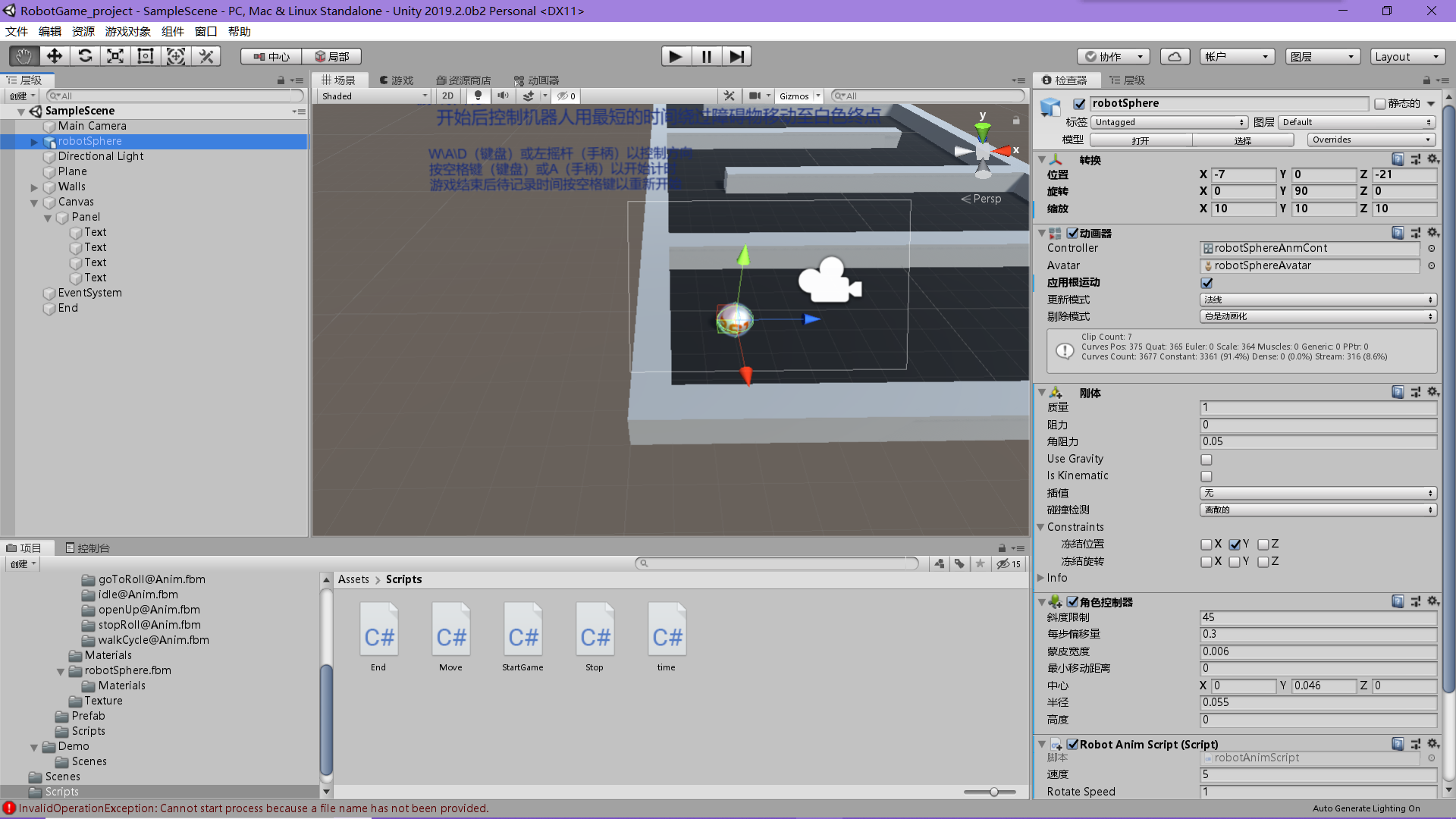
**二、实验背景**

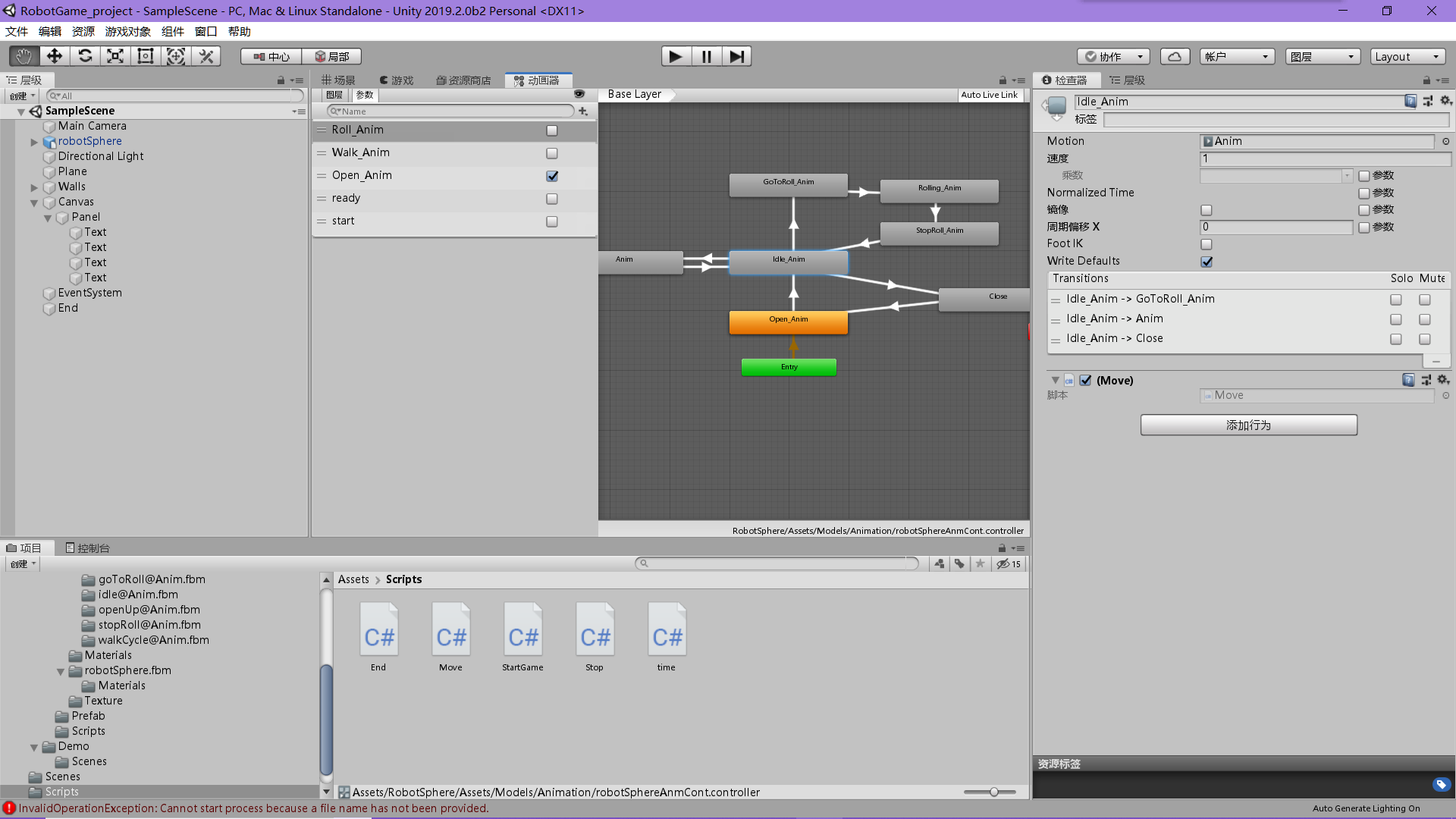
在当前的游戏市场中，常见的交互方式包括主机端的游戏手柄和VR设备，PC端的键盘、鼠标、游戏手柄和移动设备端的屏幕触摸以及各类语音交互方式。而在大型游戏中，控制人物移动、触发事件，最常见的交互方式当属键盘和游戏手柄。在控制人物移动的交互方式选择中，有全向移动和前进-转向移动，手柄在控制载具移动（如马匹、车辆）会使用前进-转向移动。然而，手柄和键盘的前进-转向移动存在差异，如前进时键盘一旦触发，矢量为一，而手柄垂直适量取决于手柄偏移角度的余弦值乘以矢量一，这导致了，两种交互方式速度有差异。在本实验中，选择了同等条件（一旦手柄有垂直或水平分量，以矢量一的速度前进、转向）的前进-转向移动，来比较两种交互方式的效率。通过比较两种交互方式在基于Unity平台开发的机器人小游戏中控制人物移动的效率差异，即比较两种交互方式在控制人物前进、转向以达到终点的时间差异，来分析键盘和游戏手柄各自的优势和劣势以及各自偏好人群的特性。

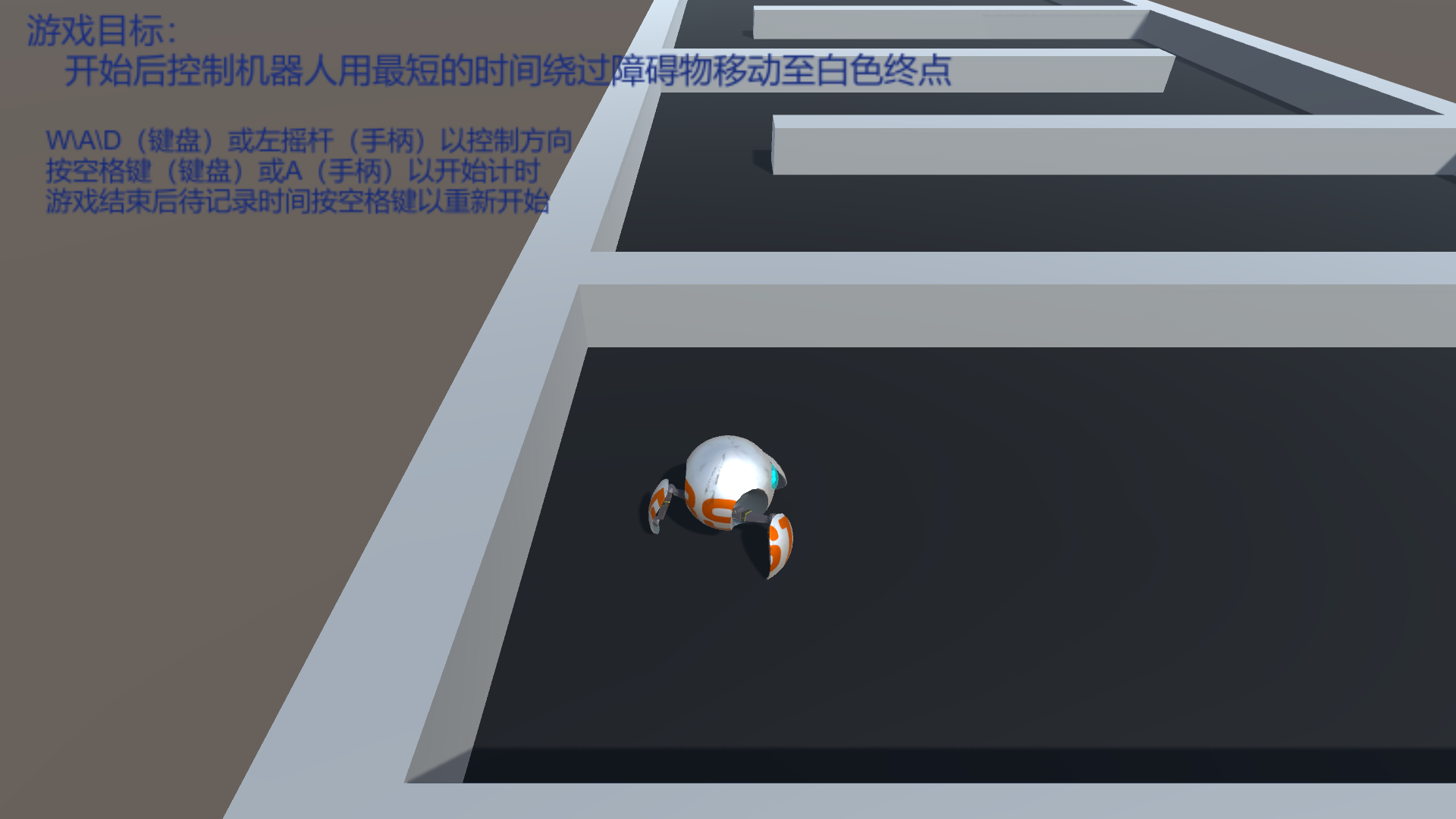
**三、实验内容**

* **研究对象**

基于unity平台开发的机器人游戏







代码实现：

**using** **UnityEngine**;

**using** **System.Collections**;

**public** **class** **robotAnimScript** : MonoBehaviour

{

**public** float speed = 6.0F;

**public** float rotateSpeed = 1.0F;

**private** Transform tran;

**private** Animator anim;

**public** bool stop = **false**;

CharacterController controller;

**private** float TimeCos;

**private** float StartTime;

**private** int TimeNum;

*// Use this for initialization*

**void** Start()

{

tran = gameObject.GetComponent<Transform>();

anim = gameObject.GetComponent<Animator>();

controller = GetComponent<CharacterController>();

}

*// Update is called once per frame*

**void** Update()

{

**if** (anim.GetBool("start"))

{

StartTime = Time.time;

float h = Input.GetAxis("Horizontal");*//返回-1到1的实数值，可以来构造向量*

float v = Input.GetAxis("Vertical");

**if** (h > 0.2)

transform.Rotate(0, rotateSpeed \* 1 \* 2, 0);*//水平旋转*

**if** (h < -0.2)

transform.Rotate(0, rotateSpeed \* -1 \* 2, 0);*//水平旋转*

Vector3 forward = transform.TransformDirection(Vector3.forward);

float curSpeed = speed \* 1 \* 3;

*//按下按键，人物才播放行走动画*

**if** (anim.GetCurrentAnimatorStateInfo(0).IsName("Base Layer.Rolling\_Anim"))

{

controller.SimpleMove(forward \* speed \* 6);

tran.Translate(Vector3.forward \* 6f \* Time.deltaTime);

**if** (Input.GetKeyDown(KeyCode.Space))

anim.SetBool("Roll\_Anim", **false**);

}

**else**

{

**if** (v > 0.1) *//判断是否按下键w*

{

**if** (anim.GetBool("ready"))

{

anim.SetBool("Walk\_Anim", **true**);

controller.SimpleMove(forward \* curSpeed);

tran.Translate(Vector3.forward \* 3f \* Time.deltaTime \* 4);

}

}

**else**

{

anim.SetBool("Walk\_Anim", **false**);

}

**if** (Input.GetKeyDown(KeyCode.Space))

{

anim.SetBool("Roll\_Anim", **true**);

}

*// Close*

**if** (Input.GetKeyDown(KeyCode.LeftControl))

{

**if** (!anim.GetBool("Open\_Anim"))

{

anim.SetBool("Open\_Anim", **true**);

anim.SetBool("ready", **false**);

}

**else**

{

anim.SetBool("Open\_Anim", **false**);

anim.SetBool("ready", **false**);

}

}

}

}

}

**private** **void** OnTriggerEnter(Collider other)

{

**if** (other.gameObject.tag == "Finish")

{

stop = **true**;

}

}

}

**using** **System.Collections**;

**using** **System.Collections.Generic**;

**using** **UnityEngine**;

**public** **class** **StartGame** : MonoBehaviour

{

**private** Animator anim;

**private** float TimeCos;

**private** float StartTime;

**private** int TimeNum;

*// Start is called before the first frame update*

**void** Start()

{

StartTime = Time.time;

anim = gameObject.GetComponent<Animator>();

}

*// Update is called once per frame*

**void** Update()

{

TimeCos = Time.time - StartTime;

**if** ((Input.GetKeyDown(KeyCode.Space)|| Input.GetButtonDown("Submit"))&& TimeCos>3.5)

anim.SetBool("start", **true**);

}

}

**using** **System.Collections**;

**using** **System.Collections.Generic**;

**using** **UnityEngine**;

**public** **class** **ThirdMove**: MonoBehaviour

{

*//创建一个公共的游戏物体，在Unity编辑器中可以将一个物体拖至脚本处的 player 的框中，脚本中就可以直接获取此物体，就不需要查找的步骤*

*//此处的功能就是将我们需要摄像机跟随的物体添加进去，也就是 Player*

**public** GameObject player;

*//创建一个变量，用来存储摄像机与物体 Player 未知的默认的差值*

**private** Vector3 offSet;

**private** **void** Start()

{

*//我们给 offSet 赋值，值的的大小为摄像机的 Position 三维向量的值减去 物体 Position 三维向量的值，也就是相机与物体之间的距离*

offSet = transform.position - player.transform.position;

}

**private** **void** LateUpdate()

{

*//使我们的相机的位置永远位于物体的位置+ offSet 这个差值*

transform.position = **new** Vector3(player.transform.position.x,0, player.transform.position.z) + offSet;

}

}

**using** **System.Collections**;

**using** **System.Collections.Generic**;

**using** **UnityEngine**;

**using** **UnityEngine.UI**;

**using** **UnityEngine.SceneManagement**;

**public** **class** **time** : MonoBehaviour

{

**private** Text TimeTip;

**private** int start = 0;

**public** Text winText;

**private** float TimeCos;

**private** float StartTime;

**public** GameObject player;

**private** Animator anim;

**private** **void** Start()

{

TimeTip = transform.GetComponent<Text>();

anim = player.GetComponent<Animator>();

}

*// Update is called once per frame*

**void** Update()

{

**if** (anim.GetBool("start"))

{

**if** (start == 0)

{

StartTime = Time.time;

start = 1;

}

**if** (player.GetComponent<robotAnimScript>().stop == **false**)

{

TimeCos = Time.time - StartTime;

}

**else**

{

winText.text = "游戏结束";

**if**(Input.GetKeyDown(KeyCode.Space)|| Input.GetButtonDown("Submit"))

SceneManager.LoadScene(0);

}

TimeTip.text = "操作用时：" + System.Math.Round(TimeCos, 3) + "秒";

}

}

}

**using** **System.Collections**;

**using** **System.Collections.Generic**;

**using** **UnityEngine**;

**public** **class** **Stop** : StateMachineBehaviour

{

*// OnStateEnter is called when a transition starts and the state machine starts to evaluate this state*

*//override public void OnStateEnter(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)*

*//{*

*//*

*//}*

*// OnStateUpdate is called on each Update frame between OnStateEnter and OnStateExit callbacks*

*//override public void OnStateUpdate(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)*

*//{*

*//*

*//}*

*// OnStateExit is called when a transition ends and the state machine finishes evaluating this state*

*//override public void OnStateExit(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)*

*//{*

*//*

*//}*

*// OnStateMove is called right after Animator.OnAnimatorMove()*

**override** **public** **void** OnStateMove(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)

{

animator.SetBool("ready", **false**);

*// // Implement code that processes and affects root motion*

}

*// OnStateIK is called right after Animator.OnAnimatorIK()*

*//override public void OnStateIK(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)*

*//{*

*// // Implement code that sets up animation IK (inverse kinematics)*

*//}*

}

**using** **System.Collections**;

**using** **System.Collections.Generic**;

**using** **UnityEngine**;

**public** **class** **Move** : StateMachineBehaviour

{

*// OnStateEnter is called when a transition starts and the state machine starts to evaluate this state*

*//override public void OnStateEnter(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)*

*//{*

*//*

*//}*

*// OnStateUpdate is called on each Update frame between OnStateEnter and OnStateExit callbacks*

*//override public void OnStateUpdate(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)*

*//{*

*//*

*//}*

*//OnStateExit is called when a transition ends and the state machine finishes evaluating this state*

*//override public void OnStateExit(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)*

*//{*

*//*

*//}*

*// OnStateMove is called right after Animator.OnAnimatorMove()*

**override** **public** **void** OnStateMove(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)

{

animator.SetBool("ready", **true**);

*// // Implement code that processes and affects root motion*

}

*// OnStateIK is called right after Animator.OnAnimatorIK()*

*//override public void OnStateIK(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)*

*//{*

*// // Implement code that sets up animation IK (inverse kinematics)*

*//}*

}

* **交互任务**

用尽可能短的时间完成游戏

* **独立变量**

交互方式：键盘cherryMX8.0-RGB黑色侧刻红轴、手柄xbox ones（红色保护套）

* **依赖变量**

游戏完成时间



* **测试人群**

十四个人，共十一男三女，随机分为两组

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 第一组 | N | C | D | G | H | K | M |
| 第二组 | A | B | E | F | I | J | L |

* **实验过程**

1、实验说明及同意书

向测试人群发放《实验说明及同意书》，待测试人员阅读完毕后签署姓名，开始准备实验。（整个实验过程中若测试人员退出，未完成的数据不作为实验数据记录）

2、练习训练

运行游戏，让测试人员根据屏幕说明，以两种交互方式各游戏一分钟以熟悉，两分钟后结束练习训练。

3、开始实验

两组测试人员（各七人）以不同的顺序（先键盘后手柄和先手柄后键盘）各进行三次实验（被试内），以消除顺序影响。试验结束后记录各自游戏时长。（下两表为实验记录表格）

|  |  |  |
| --- | --- | --- |
| 组一编号（姓名） | 键盘时长 | 手柄时长 |
| N（项章楠） |  |  |
|  |  |
|  |  |
| C（周志豪） |  |  |
|  |  |
|  |  |
| D（田宇航） |  |  |
|  |  |
|  |  |
| G（褚峰） |  |  |
|  |  |
|  |  |
| H（王中辉） |  |  |
|  |  |
|  |  |
| K（郑子珍） |  |  |
|  |  |
|  |  |
| M（王晗） |  |  |
|  |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| 组二编号（姓名） | 手柄时长 | 键盘时长 |
| A（田希宇） |  |  |
|  |  |
|  |  |
| B（冯禹逍） |  |  |
|  |  |
|  |  |
| E（华润洲） |  |  |
|  |  |
|  |  |
| F（吴昊磊） |  |  |
|  |  |
|  |  |
| I（鲁楚薇） |  |  |
|  |  |
|  |  |
| J（赵丹毓） |  |  |
|  |  |
|  |  |
| L（郭子奇） |  |  |
|  |  |
|  |  |

本实验约时长十分钟

4、调查问卷

想测试人群发放调查问卷以收集信息和意见，待填写完毕后回收。

* **实验数据**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号（姓名） | 键盘时长 | 手柄时长 | 编号（姓名） | 手柄时长 | 键盘时长 |
| N（项章楠） | 12.961 | 31.991 | A（田希宇） | 14.817 | 12.171 |
| 10.472 | 15.434 | 18.563 | 10.304 |
| 12.269 | 17.066 | 25.306 | 11.545 |
| C（周志豪） | 13.728 | 23.124 | B（冯禹逍） | 15.283 | 22.434 |
| 9.091 | 37.545 | 64.363 | 13.81 |
| 9.515 | 16.332 | 29.111 | 16.733 |
| D（田宇航） | 20.503 | 70.168 | E（华润洲） | 57.081 | 14.236 |
| 43.795 | 68.409 | 29.727 | 12.753 |
| 27.637 | 44.861 | 18.514 | 12.62 |
| G（褚峰） | 18.637 | 13.678 | F（吴昊磊） | 13.819 | 10.066 |
| 17.29 | 11.888 | 15.025 | 11.322 |
| 12.188 | 11.788 | 21.443 | 11.455 |
| H（王中辉） | 21.244 | 17.098 | I（鲁楚薇） | 71.906 | 72.432 |
| 18.289 | 12.928 | 67.038 | 25.29 |
| 16.15 | 15.125 | 56.824 | 16.067 |
| K（郑子珍） | 28.305 | 17.714 | J（赵丹毓） | 71.166 | 71.557 |
| 18.62 | 86.025 | 85.684 | 22.9 |
| 11.292 | 37.61 | 205.731 | 20.112 |
| M（王晗） | 21.011 | 69.386 | L（郭子奇） | 20.886 | 9.257 |
| 12.071 | 73.431 | 16.303 | 10.413 |
| 14.201 | 110.449 | 12.179 | 10.706 |

SPSS数据表格



实验过程中有测试人员受到干扰，该测试人员数据未记录，无测试人员主动退出，所有数据真实有效。

**四、实验结果（双因素方差分析）**

**描述统计结果**



**多元方差分析结果**

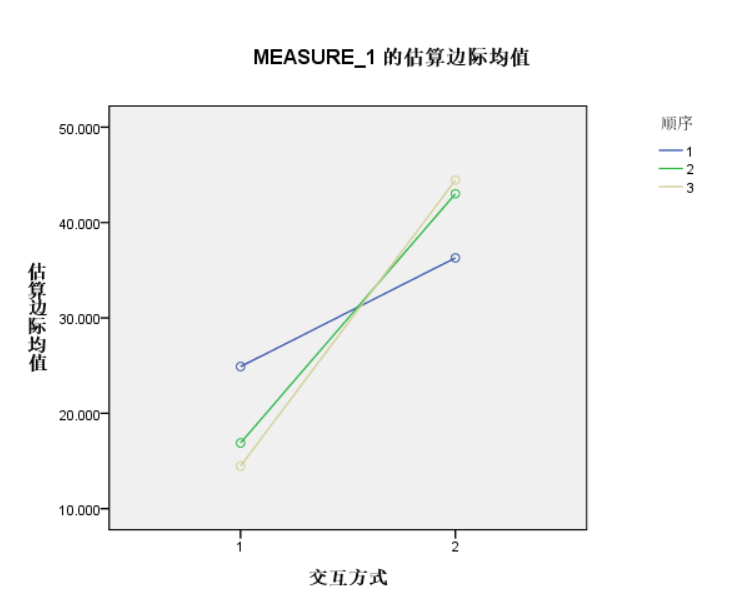


多因变量方差分析结果表明：交互方式的主效应极其显著(p<.01)

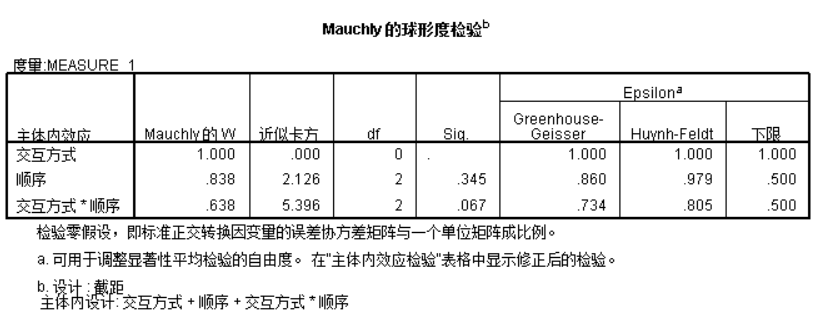
顺序的主效应不显著(=.035,ns)

交互方式与顺序的交互作用不显著(p>.05)

**均值显示图**

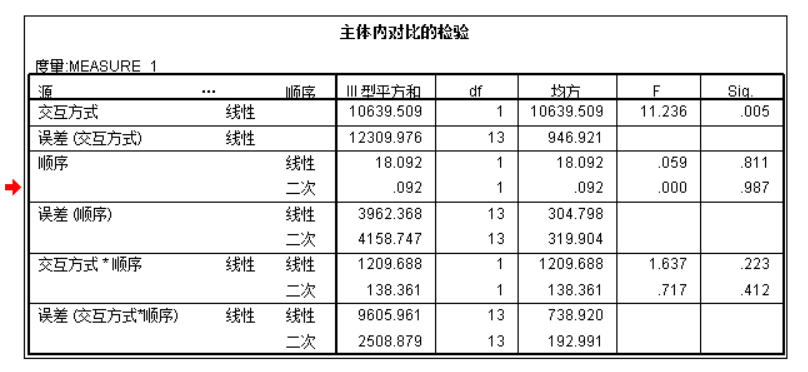


**其他**

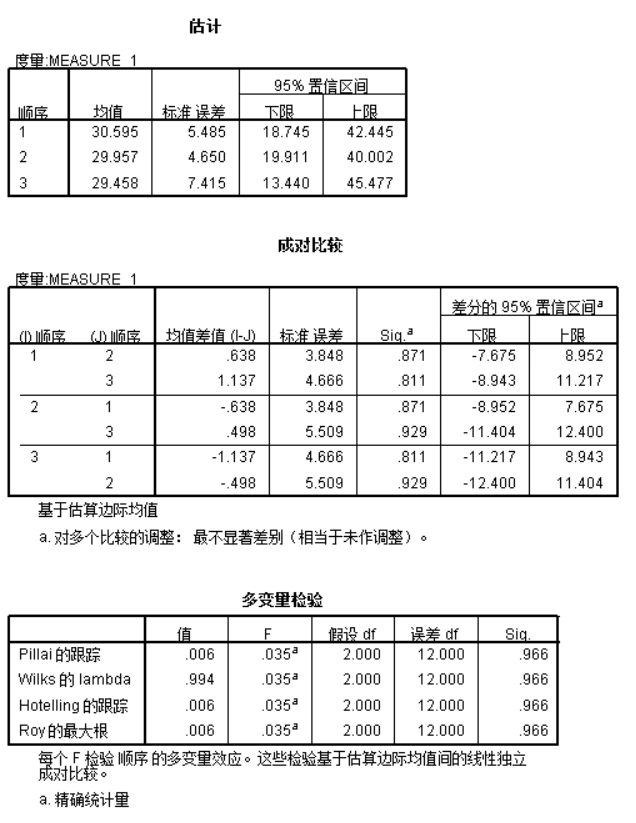


**一元方差分析结果（被试内变量效应检验）**





**多重比较结果**



**结论：**

完成的平均时间为30.004。键盘完成时间远短于手柄，平均值为18.749，手柄的平均时间是41.258。交互方式对完成时间的主效应是极其显著的(=11.236,p<.01)。顺序的主效应不可能是显著的。三次时间的平均值相差不大，且差异没有显著性(=.035,ns)。交互方式与顺序的交互作用不显著(p>.05)。

**五、讨论**

从分析结果我们可以发现，键盘是远快于手柄的，且差异是显著性的，但是从经验所得，第三人称控制人物移动手柄效率更高。这种反差我猜测可能是由于对设计的新的移动方式不熟悉（见实验背景），而且从调查问卷的反馈所得，部分人不熟悉手柄的交互方式，主观上认为键盘的熟悉度（平均值5.714）远高于手柄的熟悉度（平均值3.857），且有6人没有手柄操作经验，6人反映手柄操作难。除了不熟悉，另一方面我猜测这是由于主观上对于按键式和摇杆式的交互方式感受不同，按键式感受更真实，按就动，不按就停，而摇杆式感受更灵活。因此在相同的高速设置下，人们对于手柄的灵敏度接受度更差。未来可以关注键盘和手柄相同的移动方式下，精确度的差异以及对更舒适的手柄方式的普及。

**六、局限性**

首先，实验环境设置不理想，光线、噪音等因素众多，而且有测试人群可以从观战中提前熟悉操作，还在测试过程中会受到环境、人群的影响，影响心态从而影响实验结果。

二是实验测试人群的分布不广，年龄单一（分布于19、20），且性别比例失衡，不足以概括普适性的结论和猜测。

三是研究对象不够成熟，游戏操作方式欠妥，没有考虑长时间游戏的舒适度和可玩性，降低游戏体验。

四是对练习试验的设置欠缺考虑，试验是以什么样的顺序进行交互练习，每个交互方式确定时间练习还是自行感受，没有能把握好测试人群对新交互方式的学习度。

**七、附件**

视频

单个完整实验过程.MOV

其余实验过程.MOV

表格

实验数据.XLSX

文件

说明文档.DOCX

同意书签字

分析文件.SAV

分析结果.DOCX

调查问卷.DOCX

调查问卷

Unity项目

RobotGame\_project

Unity可执行文件

RobotGame\_executable