BEIJING 2022

Alpine Skiing VR

目录

CONTENTS

01

项目背景

项目背景

01

01 冬奥会“皇冠上的明珠”

02 场地费用限制无法普及

03 安全性与趣味性

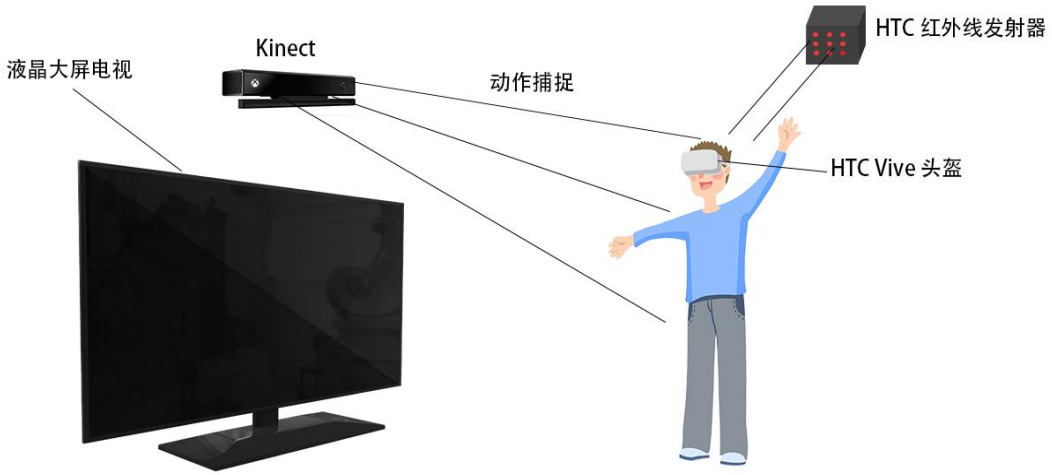
04 体验感与沉浸感

02

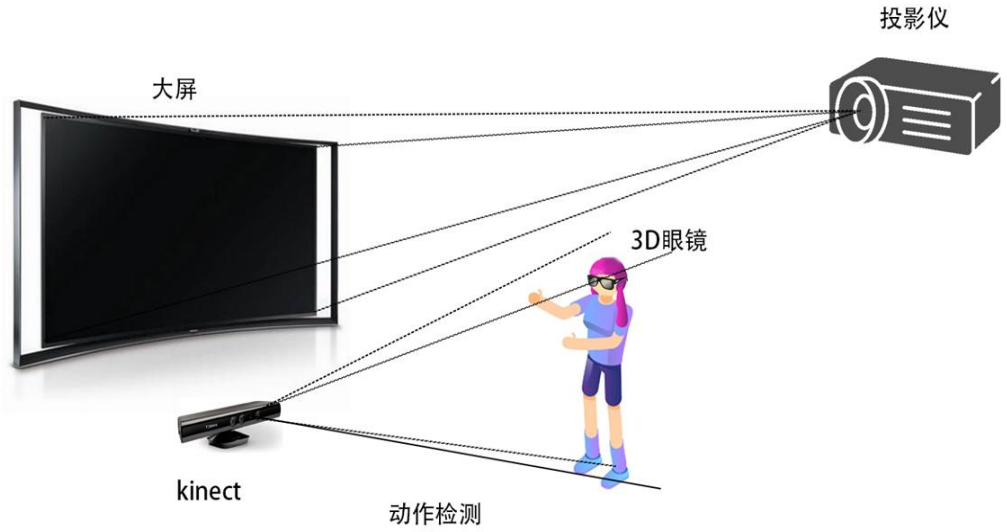
运行环境与设备

运行环境与设备

02



VR眼镜版



立体大屏版

03

模块设计与开发

设计理念

03

01 接近真实的运动体验

02 高度仿真的赛场设计

03 超越现实的赛事挑战

04 多种沉浸的呈现方式

接近真实的运动体验

物理空间

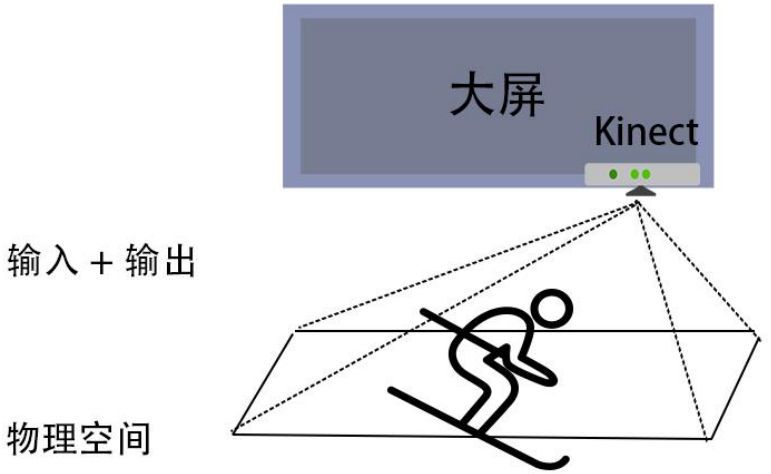
数字空间

沉浸感差

无真实运动体验

借助外设

传统的游戏交互方式



物理空间

数字空间

沉浸感强

真实的运动体验

无佩戴

本项目设计的交互方式

接近真实的运动体验

03

新手教程

01 入门指南

02 动作演示

03 生动形象

高度仿真的赛场设计

03

普通级高山滑雪场景总体图

01 回转

02 体感交互

03 旗门

04 奥运元素

05 计分规则

高度仿真的赛场设计

03

吉祥物“冰墩墩”与“雪容融”

运动员过旗门示意图

超越现实的赛事挑战

03

大师级趣味滑雪场景总体图

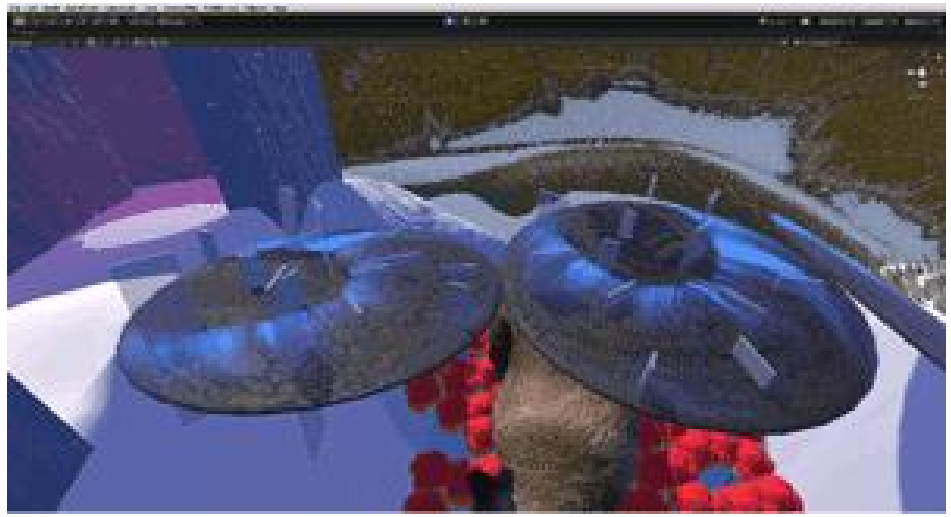
01 冒险挑战

02 魔幻色彩

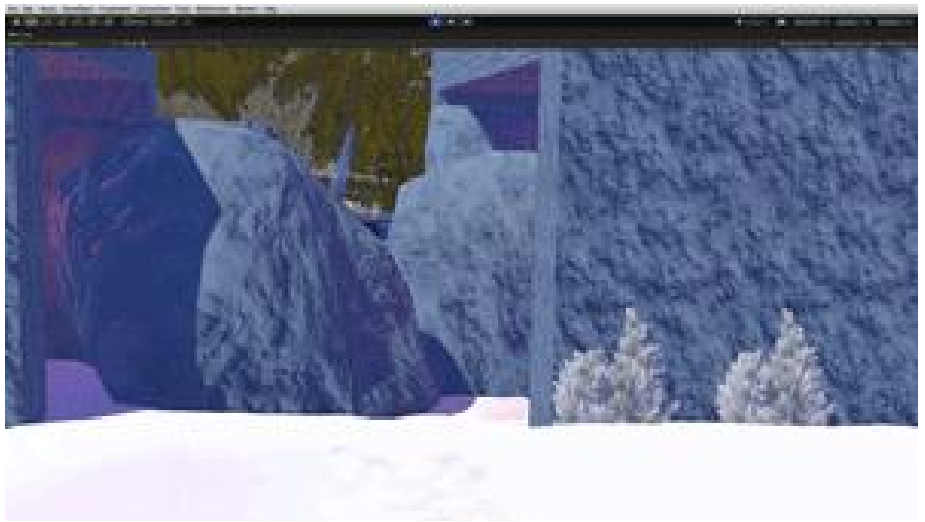
03 虚实融合

超越现实的赛事挑战

03



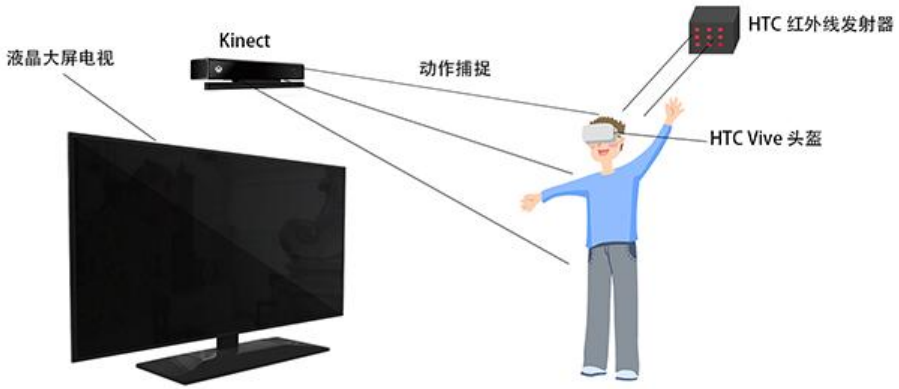
翻滚的大转盘



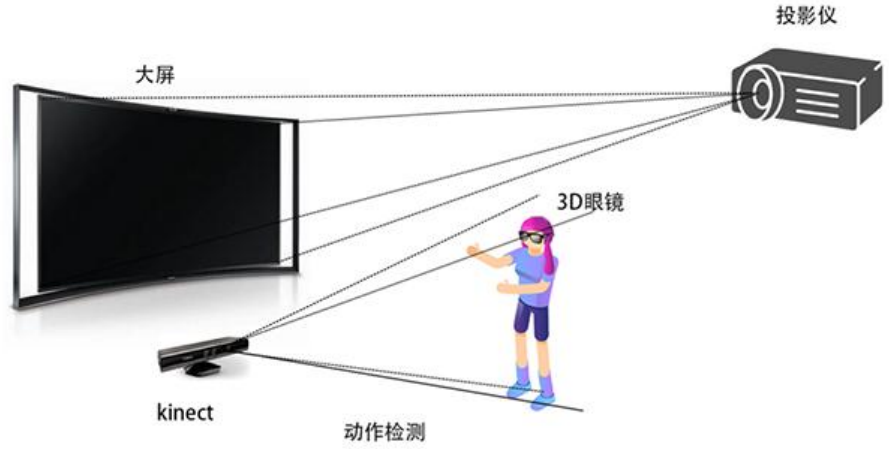
移动的巨石阵

多种沉浸的呈现方式 03

运行环境示意图（兼容两种硬件环境）



VR眼镜版



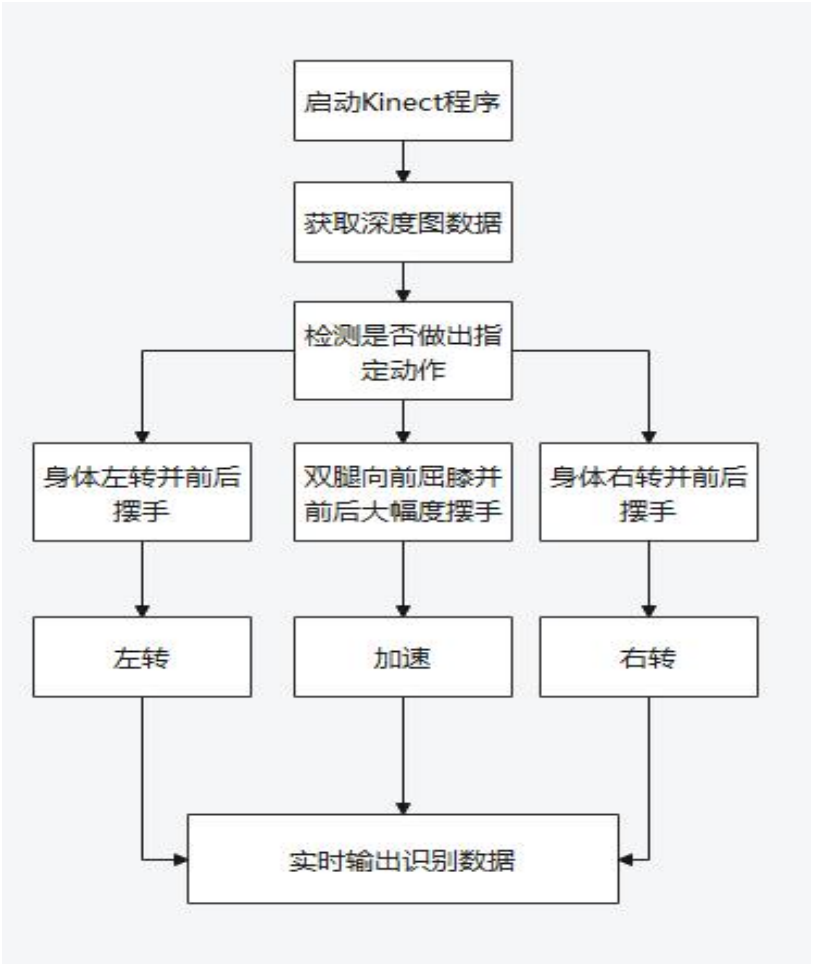
立体大屏版

04

技术难点

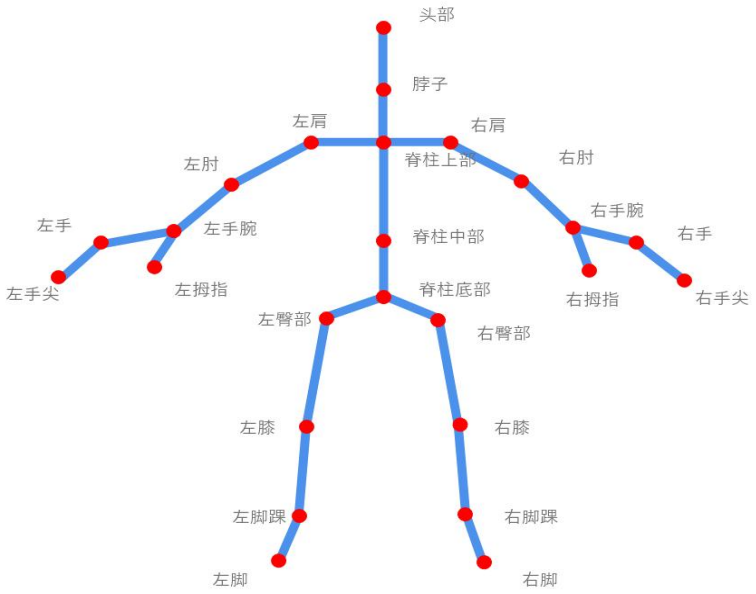
Kinect人体检测算法设计与实现 04

人体骨骼框架检测



动作检测流程

人体骨骼框架数据分析与平滑处理 04



人体关节点示意图

```
Joint joints[JointType_Count]; // 定义骨骼信息

n_body->GetJoints(JointType::JointType_Count, joints);  // 获取骨骼信息节点

int elbow = JointType_ElbowRight;

int hand = JointType_HandRight;


int coutss = 0;
float curpos = joints[hand].Position.X;
float center = joints[elbow].Position.X;  //  得到人手部和肘部的X坐标的位置  都是右手

float shoulderL = joints[JointType_ShoulderLeft].Position.Y;
float head = joints[JointType_Head].Position.Y;
float handRY = joints[JointType_HandRight].Position.Y;
float handRL = joints[JointType_HandLeft].Position.Y;//得到人肩膀和手的高度位置;

float handR = joints[JointType_HandRight].Position.X;
float handL = joints[JointType_HandLeft].Position.X;

float spine = joints[JointType_SpineMid].Position.X;
```

骨骼框架检测代码

Unity与Kinect间Socket 通信

04

```cpp
server_addr.sin_port = htons(9999);
while (true) {
    //创建套接字
    s_server = socket(AF_INET, SOCK_STREAM, 0);
    if (bind(s_server, (SOCKADDR*)&server_addr, sizeof(SOCKADDR)) == SOCKET_ERROR) {
        cout << "套接字绑定失败！" << endl;
        WSACleanup();
    }
    else {
        cout << "套接字绑定成功！" << endl;
    }
    //设置套接字为监听状态
    if (listen(s_server, SOMAXCONN) < 0) {
        cout << "设置监听状态失败！" << endl;
        WSACleanup();
    }
    else {
        cout << "设置监听状态成功！" << endl;
        break;
    }

}

cout << "服务端正在监听连接，请稍候...." << endl;
//接受连接请求
len = sizeof(SOCKADDR);
s_accept = accept(s_server, (SOCKADDR*)&accept_addr, &len);
if (s_accept == SOCKET_ERROR) {
    cout << "连接失败！" << endl;
    WSACleanup();
    return 0;
}
```

服务器端

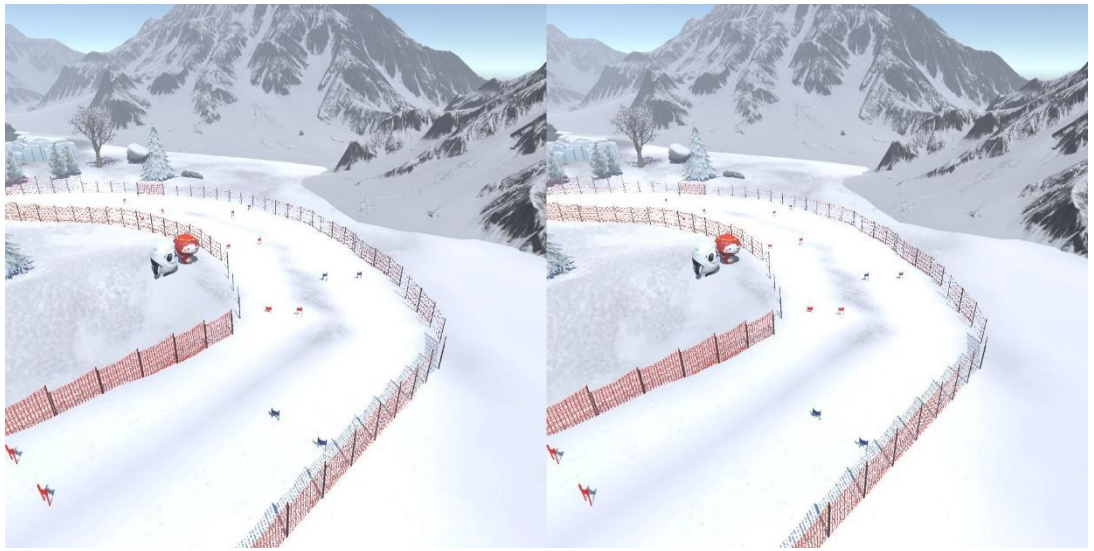
```cpp
//建立连接
private void connectServer()
{
    try
    {
        socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        socket.Connect(IPAddress.Parse("127.0.0.1"), 9999);
        Debug.Log("服务器连接");
        receiveSit = new Thread(ReceiveSit);
        receiveSit.Start();
    }
    catch (Exception ex)
    {
        Debug.Log("服务器连接失败");
        Debug.Log(ex.Message);
    }

}
private void ReceiveSit()
{
    while (true)
    {
        if (socket.Connected == false)
        {
            Debug.Log(("断开连接"));
            break;
        }
```

客户端

曲面立体大屏设置立体视觉 04

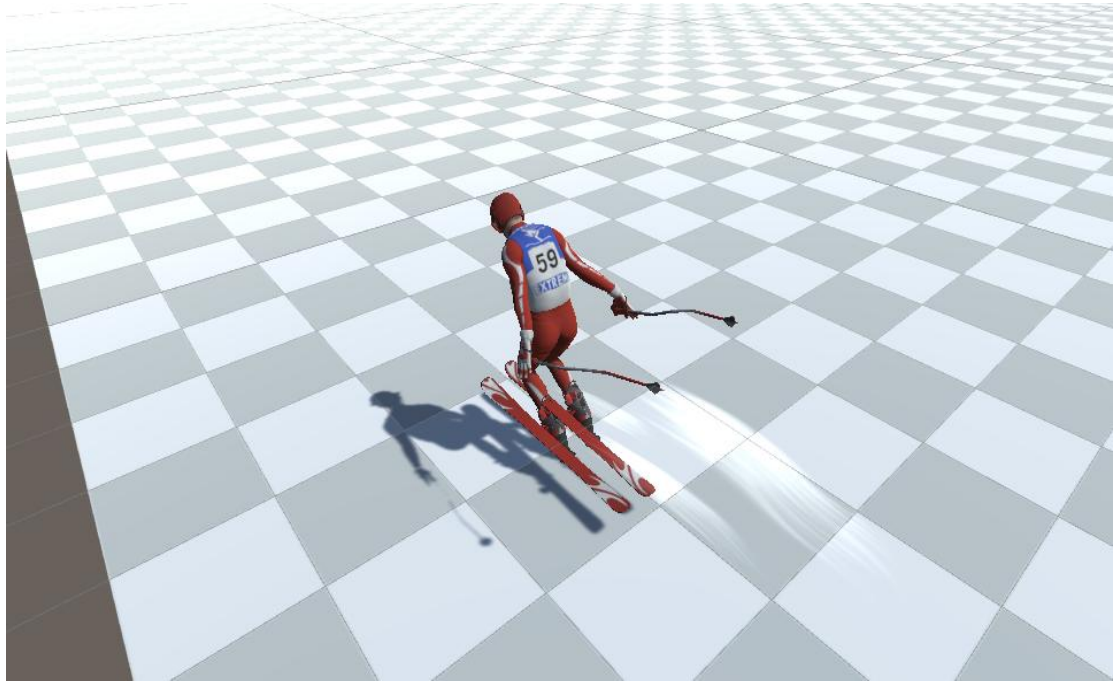
立体大屏游玩效果



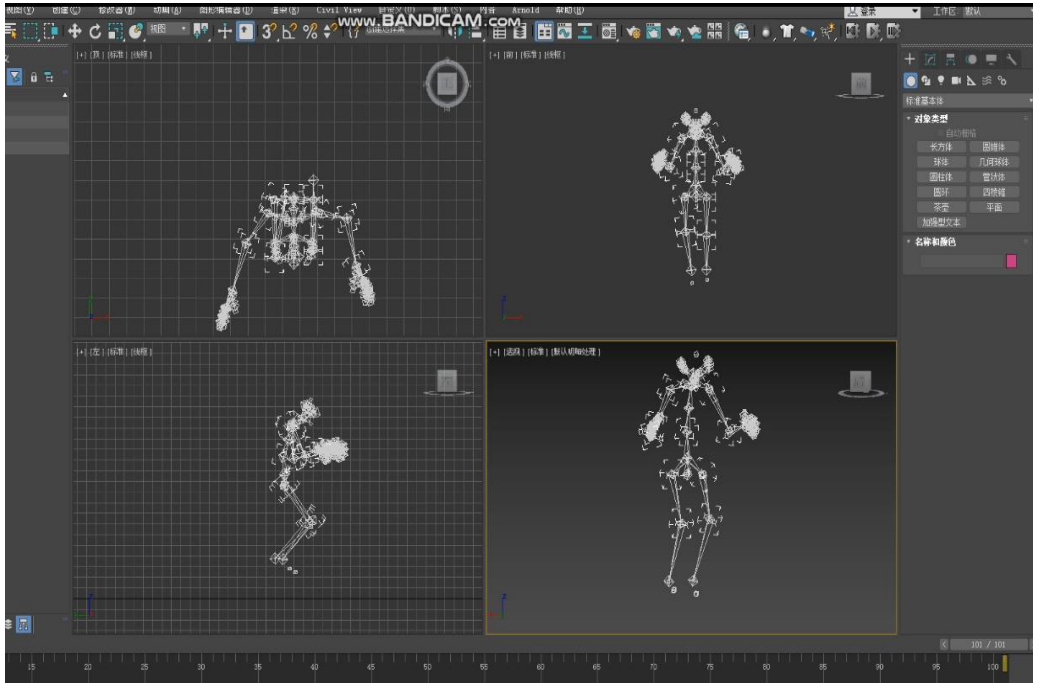
采用双相机模拟立体效果

(由于是偏振立体双画面，普通相机拍摄时会有重影)

滑雪运动员骨骼动画制作 04



人物动画与风速粒子特效



人物骨骼模型

粒子特效制作

04

领奖台粒子效果预览图

01 粒子系统

02 生命周期

03 粒子爆炸

雪地凹陷效果制作

04

雪地凹陷效果示意图

01 正交相机

02 深度图

03 tessellation技术

BEIJING 2022

感谢观看