|  |
| --- |
| 北京理工大学 |
| **基于手机平台的室内空间虚拟漫游系统** |
|  |

|  |
| --- |
| 学生： 熊方瑜  学号：1120112056  指导老师：黄天羽  2015-5-31 |

摘 要

作为虚拟现实技术的重要分支，虚拟漫游技术在游戏、医学、建筑、旅游、航空航天等多种行业发展很快。由于其可贵的3I特性——沉浸感、交互性和构想性，使得沿用固定漫游路径等手段的其他漫游技术和系统无法与之相比。虚拟漫游系统是人们生产生活应中最普遍的例子，使得人们可以在屏幕上就能够访问具有视觉沉浸感的几何空间环境。

虚拟漫游的应用在虚拟校园、虚拟博物馆、虚拟样板间、虚拟公园已经有非常成熟的案例，本课题的主要工作是基于Unity3D移动端引擎，构建基于手机自拍功能的虚拟漫游系统，能通过用户自定义生成所需要的室内空间模型，避免了3D手工建模。系统支持一般手机触摸交互的漫游浏览操作，要求尽可能有友好的用户体验。

本文首先介绍了系统涉及的基本理论与技术，然后进行需求分析，并对系统的UI和功能进行详细设计，之后实现本系统，并测试其功能性、易用性和效率性能，最后对系统今后进一步的发展和完善进行展望。

**关键词：**虚拟漫游；用户体验；Unity3D；NGUI；Android

Abstract

Virtual roaming technology is developing rapidly in gaming, medical, construction, tourism, aerospace and other industries. Because of its commendable 3I features – immersion, interaction and imagination, making other roaming technology and systems which follow fixed roaming path beyond comparison. Virtual roaming system should be the most common examples of peoples’s living and production that people can have access to the geometry of the space visually immersive environment on the screen.

The applications of Virtual roaming have very mature cases in virtual campus, virtual museum, virtual prototype room and virtual park. The main work of this study is building a virtual roaming system with self-timer function of the phone based on Unity 3D. The system can generate the required interior model by user overrides and avoid manual 3D modeling. The system supports roaming browse operation of general mobile phone touch interaction which requires user-friendly experience as much as possible.

This paper first introduces the basic theory and techniques of the system, then does the needs analysis and the design of the system IU and functionality in detail. In the next part, the paper implement the system and test its functionality, usability, performance and efficiency. The last part of the paper prospects the further development and improvement of the system.

**Keyword:** Virtual roaming; User experience; Unity3D; NGUI; Android;

目录

[摘 要 I](#_Toc420936965)

[Abstract II](#_Toc420936966)

[第1章 绪 论 1](#_Toc420936967)

[1.1 选题背景 1](#_Toc420936968)

[1.2 研究现状及发展前景 1](#_Toc420936969)

[1.3 论文的主要内容及组织结构 3](#_Toc420936970)

[第2章 基本理论与技术 4](#_Toc420936971)

[2.1 系统开发环境介绍 4](#_Toc420936972)

[2.2 虚拟漫游技术简介 4](#_Toc420936973)

[2.3 Unity3D技术简介 5](#_Toc420936974)

[2.4 Unity3D插件——NGUI技术简介 7](#_Toc420936975)

[2.5 Android技术简介 11](#_Toc420936976)

[第3章 基于用户友好系统的需求分析和设计 12](#_Toc420936977)

[3.1 基于用户友好系统的需求分析 12](#_Toc420936978)

[3.1.1 任务概述 12](#_Toc420936979)

[3.1.2 功能需求 12](#_Toc420936980)

[3.2 系统界面设计 13](#_Toc420936981)

[3.2.1 编辑场景界面设计 13](#_Toc420936982)

[3.2.2 场景漫游中的界面设计 15](#_Toc420936983)

[3.2.3 Android端界面设计 17](#_Toc420936984)

[3.3 系统详细设计 17](#_Toc420936985)

[3.3.1 Android端详细设计 17](#_Toc420936986)

[3.3.2 Unity3D端详细设计 19](#_Toc420936987)

[3.4 系统的实现 29](#_Toc420936988)

[3.4.1 编辑场景的UI实现 29](#_Toc420936989)

[3.4.2 编辑场景的场景实现 30](#_Toc420936990)

[3.4.3 漫游场景的UI实现 31](#_Toc420936991)

[3.4.4 漫游场景的场景实现 31](#_Toc420936992)

[3.4.5 Unity3D端与Android通信模块的实现 32](#_Toc420936993)

[3.4.6 Android端的实现 32](#_Toc420936994)

[3.5 关键算法实现 34](#_Toc420936995)

[3.5.1 基于特定边的缩放算法 34](#_Toc420936996)

[3.5.2 基准线位置算法 34](#_Toc420936997)

[3.6 本章小结 36](#_Toc420936998)

[第4章 系统运行及性能分析 36](#_Toc420936999)

[4.1 运行环境 36](#_Toc420937000)

[4.2 运行效果 36](#_Toc420937001)

[4.3 本章小结 44](#_Toc420937002)

[第5章 总结与展望 45](#_Toc420937003)

[5.1 总结 45](#_Toc420937004)

[5.2 展望 45](#_Toc420937005)

[致 谢 47](#_Toc420937006)

[参考文献 48](#_Toc420937007)

# 绪 论

## 选题背景

虚拟现实的英文全称为Virtual Reality（VR技术），是通过电脑模拟从而产生三维空间的虚拟世界，让用户得以体验关于视觉、听觉、触觉等感官的模拟，如同身历其境一般，并能够无限制并及时地观察三维空间内的事物。用户控制虚拟人物或摄像机进行位移时，电脑能够在同一时间进行复杂的运算，将精确的三维世界视频传入电脑屏幕进行处理从而使得用户产生身临其境之感。该技术集成了计算机图形、计算机仿真、人工智能、感应、显示及网络并行处理等技术的最新发展成果，是一种由计算机技术辅助生成的高技术模拟系统。

作为虚拟现实技术的重要分支，虚拟漫游技术在游戏、医学、建筑、旅游、航空航天等多种行业发展很快。由于其可贵的3I特性——沉浸感、交互性和构想性，使得沿用固定漫游路径等手段的其他漫游技术和系统无法与之相比。

## 研究现状及发展前景

计算机作为一种计算工具和分析工具的出现，产生了很多解决问题的新方法，其中之一就是虚拟现实技术的产生与发展。国内外虚拟漫游技术主要涉及到两个研究领域：真实场景虚拟漫游；虚拟场景漫游。

真实场景这种虚拟漫游的最大特点是：被漫游的对象在现实生活中是客观存在的，所以会基于对象的真实数据来制作漫游对象因此其漫游形式是异地虚拟的，但并不影响任何体验。它在计算机中创建出得基于现实的虚拟环境，能够使用户体验到身处其中的沉浸感使之具有与环境优良的交互作用能力，并有助于启发构思。VR已不仅仅被关注于计算机图象领域，它已涉及更广的领域，如电视会议、网络技术和分布计算技术，并向分布式虚拟现实发展。虚拟现实技术已成为新产品设计开发的重要手段。

这种虚拟漫游在名胜景观上和军事上很有用处。在2003年完成的故宫VR《紫禁城·天子的宫殿》项目中，参观者借助操控器能够在虚拟的紫禁城中自由漫游，能游览康乾盛世的紫禁城，近距离地观看太和殿的全景。

虚拟场景指现实生活中并不存在，虚构的场景。虚拟场景漫游作为一种前景大好的技术领域，其应用越来越广泛。在城乡规划、建筑设计、室内装潢等建筑行业，或在虚拟战争演练场和作战指挥模拟训练方面，抑或在游戏设计与娱乐行业，乃至在促进未来新艺术形式诞生等方面，均有它的发挥空间，而且虚拟场景漫游的应用也能代表着这些行业的新技术和新水平。

随着虚拟漫游技术在城市规划、军事等方面应用的不断深入，在建模与绘制方法、交互方式和系统构建方法等方面，对虚拟现实技术都提出了更高的需求。为了满足这些新的需求，近年来，虚拟现实相关技术快速发展基于 “低成本、高性能”的原则，表现出一些新的特点和发展趋势。主要表现在以下方面：

1. 动态环境建模技术。

  虚拟漫游技术的核心内容是建立虚拟环境，获取实际环境的三维数据是构建动态环境建模技术的首要因素，后根据需要建立相应的虚拟环境模型。

1. 生成实时三维图形并显示技术。

如今三维图形的生成技术已比较成熟，然而在保证图形的质量和复杂程度的前提下，如何提高刷新频率将是今后重要的研究内容，而 “实时生成”是其中的关键点。此外，立体显示和传感器技术的发展关系到虚拟漫游，现有的虚拟设备还不能满足需要，研究开发新的三维图形生成和显示技术成为虚拟漫游发展的关键因素。

1. 适人化、智能化人机交互设备的研制。

虽然智能头盔和数据手套等通过设备进行交互的方式能够增强用户沉浸感，但是他们在实际应用中的效果并不尽人意，无法达到沉浸交互的目的。为了提高虚拟漫游中的交互性效果，最好采用人类的视觉、听觉、触觉和自然语言等作为交互的方式。

1. 大型网络分布式虚拟现实的研究与应用。

网络虚拟现实能够通过新型的人机交互设备介入计算机，使得在一个基于网络的计算机集合中的多个用户能够体验到其产生的多维的、适用于用户应用的（适人化）相关虚拟情景环境。分布式虚拟环境系统除了满足复杂虚拟环境计算的需求外，还应满足分布式仿真与协同工作等应用对共享虚拟环境的自然需求。分布式虚拟现实系统必须支持系统中多个用户、信息对象(实体)之间通过消息传递实现的交互。分布式虚拟现实可以看作是基于网络的虚拟现实系统，是可供多用户同时异地参与的分布式虚拟环境，处于不同地理位置的用户如同进入到同一个真实环境中。目前，分布式虚拟现实系统已成为国际上的研究热点，相继推出了相关标准，在国家“八六三”计划的支持下，由北京航空航天大学、杭州大学、中国科学院计算所、中国科学院软件所和装甲兵工程学院等单位共同开发了一个分布虚拟环境基础信息平台，为我国开展分布式虚拟现实的研究提供了必要的网络平台和软硬件基础环境。

## 论文的主要内容及组织结构

论文的研究目的是利用Unity3D游戏引擎建立一个基于手机平台的室内空间虚拟漫游系统，为用户提供一个场景漫游的平台。论文的主要阐述以下几个方面：

1. 介绍总结虚拟漫游的发展和现状，阐述其在生活中的重要性和实用性。
2. 介绍本系统中有关的基本理论与技术，主要介绍Unity3D及其插件NGUI的基本功能，和Android的基本技术。
3. 对系统进行需求分析、界面设计、详细设计，最终开发出一个基于手机平台的室内空间虚拟漫游系统，实现了在编辑场景中有很好的易用性，在漫游场景中能流畅漫游。
4. 运行本系统，从功能性，易用性和性能三方面测试本系统。
5. 提出本系统可发展和改进的方面。

论文共有三个部分来组织本文结构：

第一部分是第一章绪论，简要介绍课题背景、发展与现状，说明课题的主要内容和论文的组织结构。

第二部分包含三章，是论文和工作的核心：

1. 第二章对本课题所使用的基本理论与技术做简要介绍，主要包括Unity3D及其插件技术，和Android技术。
2. 第三章对本课题的需求做了简要分析，并对系统的UI和功能进行概要和详细设计，最后实现设计。
3. 第四章对本系统的运行情况进行了总结。

第三部分是本课题的总结，说明工作成果，评价做出的系统，指出存在的不足之处，并对系统今后进一步的发展和完善提出需求和建议。

# 基本理论与技术

## 系统开发环境介绍

硬件环境：

CPU：Inter Core i7-3630QM四核处理器

显卡：NVIDIA GeForce GT650M 2G DDR3 独显

内存：8G DDR3内存

硬盘：128G

软件环境：

开发平台：Windows 7操作系统，其友好的用户界面使得它成为目前PC机上很流行的个人操作系统。

开发语言：Microsoft Visual C#、Java

C#作为Unity3D使用的脚本语言之一有严谨的语法和强大的功能。

Java是Android开发中必不可少的语言。

开发软件：Unity3D 4.3.4、Eclipse LUNA、Microsoft Visual 2012

## 虚拟漫游技术简介

虚拟现实技术(VR技术)是当代信息科技发展的代表，它涉及多方面的技术，如机器视觉、三维仿真技术等多个信息技术分支，是对这些技术更高层次的集成、渗透与综合应用，是当前科学技术研究领域的先锋。虚拟现实技术为让用户体验身临其境的沉浸感，利用计算机的相关技术模拟各种三维虚拟场景并能够让用户实时全面地感知三维虚拟场景中的物体。

虚拟漫游系统是虚拟现实在人们生产生活应用中最普遍的例子，使得人们可以在屏幕上就能够访问具有视觉沉浸感的几何空间环境。虚拟漫游的应用在虚拟校园、虚拟博物馆、虚拟样板间、虚拟公园已经有非常成熟的案例，本课题的主要工作是建立一个基于手机平台的室内空间虚拟漫游系统，能通过用户自定义生成所需要的室内空间模型，避免了3D手工建模。

本课题基于Unity3D移动端引擎，构建基于手机自拍功能的虚拟漫游系统。

## Unity3D技术简介

Unity是由Unity Technologies开发的一个游戏引擎，能让玩家轻松创建三维视频游戏、建筑可视化、实时三维动画等类型互动内容，并支持多平台。其编辑器运行在Windows或Mac OS X下，可发布游戏至Windows、Mac、Wii、iPhone、Windows phone 8和Android平台。

Unity3D的主要特性：

* 层级式的综合开发环境，可视化编辑，详细的属性编辑器和动态的游戏预览。Unity也被用来快速的制作游戏或者开发游戏原型。
* 可开发微软Microsoft Windows和Mac OS X 的可执行文件，在线内容（通过Unity Web Player插件支持Internet Explorer、Firefox、Safari、Mozilla、Netscape、Opera和 Camino），Mac OS X的Dashboard工具，Wii程序和iPhone应用程序。
* 自动资源导入：项目中的资源会被自动导入，并根据资源的改动自动更新。虽然很多主流的三维建模软件为Unity所支持，不过对于3ds Max、Maya、Blender、Cinema 4D和Cheetah3D的支持比较好，并支持一些其他的三维格式。
* 图形引擎使用的是Direct3D(Windows),OpenGL(Mac,Windows)和自有的APIs(Wii)
* 支持Bump mapping、Reflection mapping、Parallax mapping、Screen Space Ambient Occlusion、动态阴影使用的是Shadow Map技术，并支持Render-to-texture和全屏Post Processing效果。
* Shaders编写使用ShaderLab语言，同时支持自有工作流中的编程方式或Cg、GLSL语言编写的Shader。一个Shader可以包含众多变量及一个参数接口，允许Unity去判定参数是否为当前所支持并适配最适合参数，并自己选择相应的Shader类型以获得广大的兼容性。
* 内置对Nvidia的PhysX physics engine支持。
* 游戏脚本基于Mono，一个兼容于.NET Framework 2.0的跨平台开源包，因此程序员可用JavaScript、C#或Boo加以编写。
* The Unity Asset Server：一个支持各种游戏和脚本的版本控制方案，使用PostgreSQL作为后端。
* 音效系统基于OpenAL程序库，可以播放Ogg Vorbis的压缩音效。
* 视频播放采用Theora编码。
* 内置地形编辑器，支持树木与植被贴片。
* 内置Lightmapping以及Global illumination。
* 多人网络连接功能由第三方包提供，有Raknet、Photon、SmartFoxServer……等，多种选择。

Unity3D的基本界面介绍

Unity3D的基本界面非常简单，几个窗口就可以实现几乎全部的编辑功能。主界面如图2-1所示（图片已经过反色处理）：

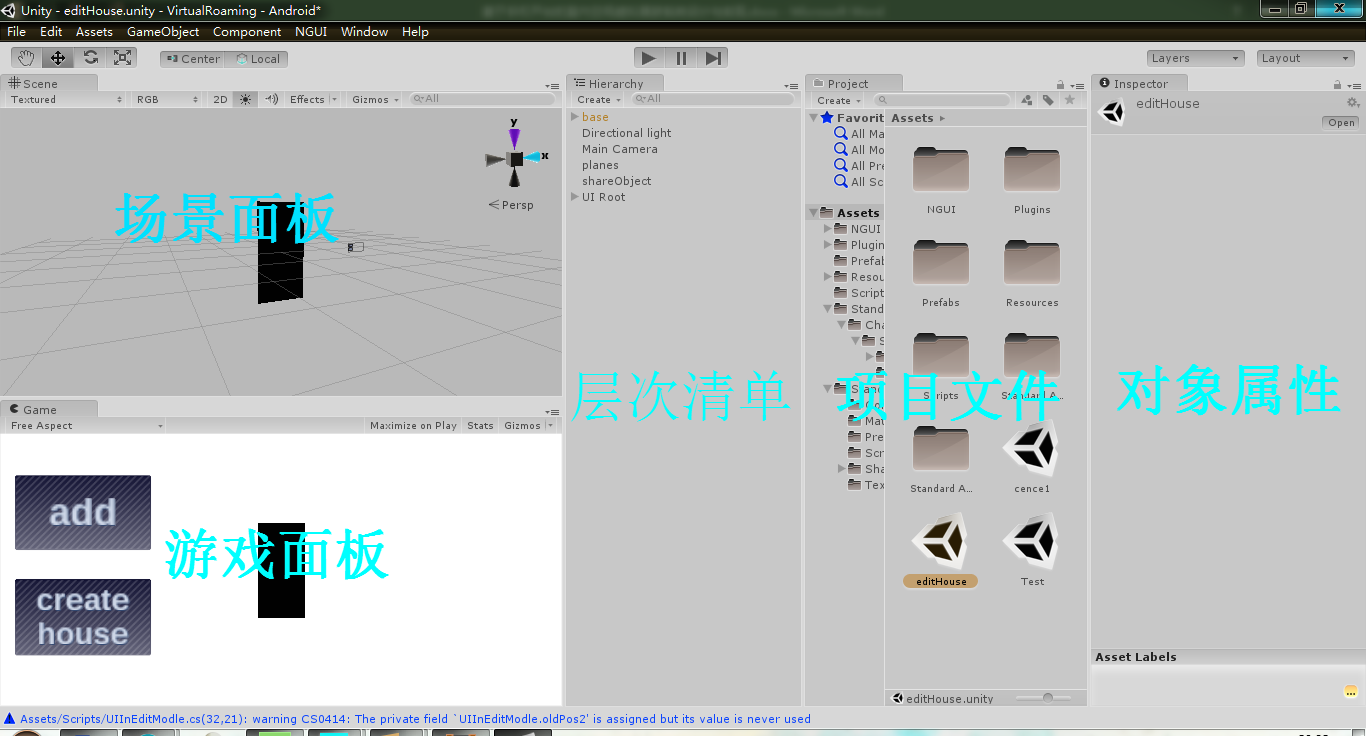


图 2‑1

场景面板：该面板为Unity3D的编辑面板；可以将所有的模型、灯光、以及其他材质对象拖放到当前场景中，构建游戏中所能呈现的景象。

动画面板：与场景面板不同，该面板是用来渲染场景面板中的景象的。该面板不能用作编辑，但却可以呈现完整的动画效果。显示的内容取决于场景摄像机的设置。

层次清单：该面板主要功能是显示放在场景面板中的所有的物体对象。

项目文件：该面板主要功能是显示该项目文件中的所有资源列表。除了模型、材质、字体等，还包括该项目的各个场景文件。

对象属性：该面板栏会呈现出任何对象的固有属性，包括三维坐标、旋转量、缩放大小、脚本的变量和对象等等。

## Unity3D插件——NGUI技术简介

NGUI是严格遵循KISS原则并用C#编写的Unity（适用于专业版和免费版）插件，提供强大的UI系统和事件通知框架。其代码简洁，一般少于200行代码。这使得程序员可以很容易地扩展NGUI的功能或调节已有功能。代码的简洁使得所有其它用户入门更加快且性能更高，学习过程更加有趣。

NGUI的主要特性：

* 完全集成到Inspector面板中。
* 不需点击Play按钮就能查看结果。
* 在场景视图中看到的就是在游戏视图中得到的（所见即所得）。
* 基于组件的、模块化的特性：要让你的界面控件做什么，只需为其附加相应的行为，而不需要编码。
* 全面支持IOS/Android和Flash。
* 灵活的事件系统。
* 可以让复杂的UIs只占用一个draw call。
* 可以直接在编辑器中创建、更新/修改纹理地图集，或从Texture Packer程序导入纹理地图集。
* 支持光照贴图、法线贴图、折射等特性。
* 支持硬边或柔性的面板裁剪。
* 支持灵活尺寸的表格，能够自动对控件进行排列。
* 通过IME输入法支持东方语言（有Web版本的演示程序）
* 内建本地化系统。
* 内建的键盘和摇杆支持。
* 提供大量有用的辅助脚本，从改变按钮颜色到拖拽对象。
* 简单的内建补间动画系统。
* 简洁和高度优化的C#代码。
* 没有DLL，也不依赖于其他外部资源NGUI的使用方式与Unity的使用方式一样。使用Widget Tool可快速创建模板化的控件，或者从基本组件创建你自己的控件。可以按照你的意愿拷贝/粘贴，把你的窗口保存为prefabs。所有一切只需简单地点
* 点击操作即可完成。当需要编写代码让控件移动时，可以选择简单的例子代码，可以把你的控件转变为按钮、输入框、基于事件改变颜色、播放声音、触发动画等等。

NGUI插件使用介绍：

1. 点击菜单栏上的NGUI选项，选择Create->Widget，将会生成如图2-2（图片已经过反色处理）所示模型；
2. 在项目文件栏中，点击All Prefabs，并在右边栏中选择需要的预设，如图2-3（图片已经过反色处理）所示，选择了Control – Colored Button，在对象属性面板中能看到其属性；
3. 将其拖入层次清单栏中刚创建的Container容器中，并设置其位置大小，如图2-4（图片已经过反色处理）所示；
4. 设置按钮的UISprite中的Anchors中的Type为Unified，如图2-5（图片已经过反色处理）所示，观察场景栏，白线为摄像机视线，黄线为按钮相对于摄像机视线的距离，此时便完成了适配绝大多数分辨率的UI。

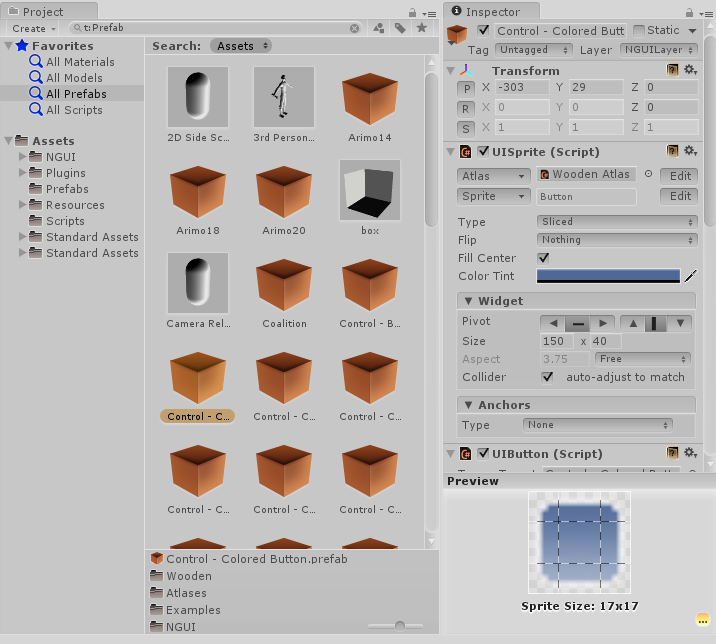


图 2‑3

图 2‑2

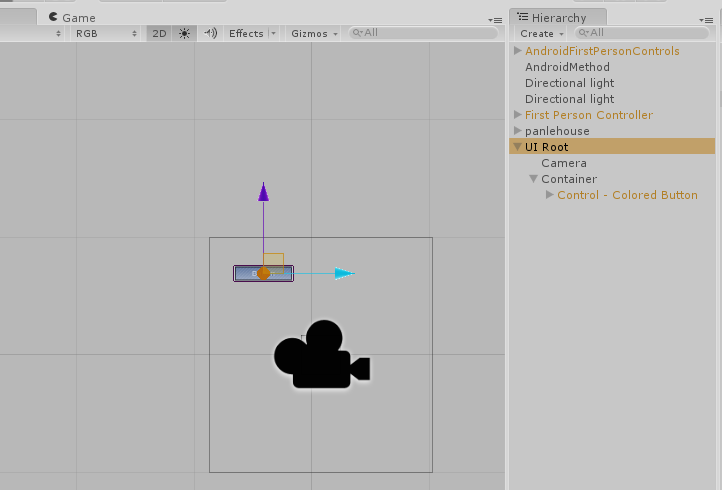
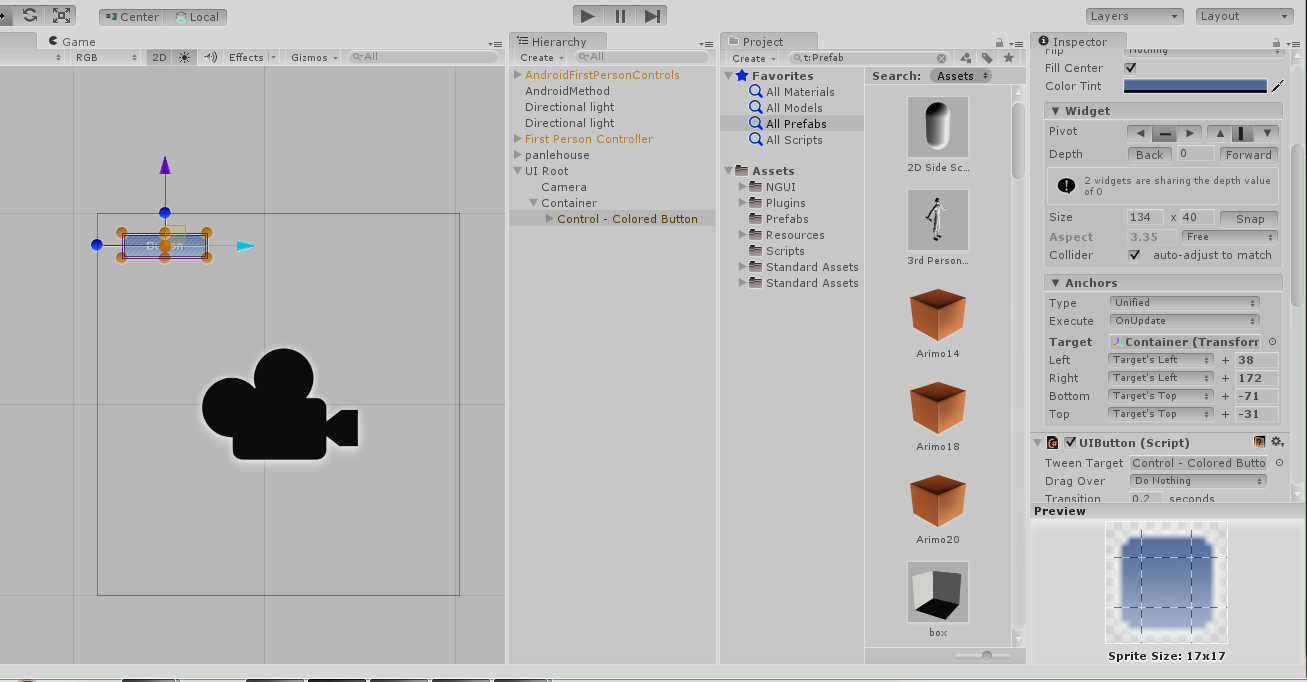


图 2‑4

图 2‑5

NGUI主要组件介绍：

1. 在UI Root (2D)对象上有个UIRoot脚本。这个脚本会重新调整游戏对象符合你的屏幕高度，有自动和手动选择高度。让你制定小部件在像素中的坐标，并且和游戏世界中的剩余对象相比依旧相对来说较小。
2. Camera对象包含Camera和UICamera脚本。UICamera脚本包含NGUI的事件系统（event system）。
3. Anchor包含UIAnchor脚本。虽然这个脚本可以附加给控件，但在这里可以避免Windows机器上半个像素偏移的问题。
4. Panel对象有UIPanel脚本，UIPanel是一个容器，它将包含所有UI小部件，并负责将所包含的部件组合优化，以减少绘制命令的调用。

## Android技术简介

Android是一个基于Linux的开放源代码移动设备操作系统，在智能手机和平板电脑中有广泛应用。由Google成立的Open Handset Alliance（OHA，开放手持设备联盟）持续领导与开发中。Android已发布的最新版本为Android 5.1.1(Lollipop)。

Android的主要特性：

* Android平台最大优势就是其开放性，使用Android系统开放的平台允许任何移动终端厂商使用Android系统，继而拥有更多的开发者。随着用户和应用的日益丰富，一个崭新的平台也走向成熟。苹果的IOS系统封闭，不向任何厂商提供；Windows Phone系统封闭，厂商需要花高价购买；Nokia的Symbian之前也一直是封闭，现在已经末落了。
* 由于Android的开放性，Android产品将会更加丰富多彩、各具特色，而数据同步和软件的兼容性等都不会因此受影响。用户若从诺基亚的Symbian风格手机改用装载Android系统的手机，将不会导致联系人等资料的丢失，同时还可将Symbian中优秀的软件带到新手机上使用。
* Android平台提供第三方开发商不会受到各种条框的限制阻挠，提供其一个十分宽泛、自由的环境，通过此种途径，产生了大量新颖别致的软件。
* Android平台手机能无缝结合Google的许多服务与应用，如地图、邮件等。

# 基于用户友好系统的需求分析和设计

## 基于用户友好系统的需求分析

### 任务概述

本课题的主要工作是建立一个基于手机平台的室内空间虚拟漫游系统。手机平台与桌面PC相比，其图形处理器性能较弱，因此基于图像渲染的虚拟漫游系统更具有技术的可实时性，且系统应用成本较低。

本课题基于Unity3D移动端引擎，构建基于手机自拍功能的虚拟漫游系统，避免3D手工建模。系统支持一般手机触摸交互的漫游浏览操作，要求尽可能有友好的用户体验。

### 功能需求

1. **主要功能见表3-1：**

表 3‑1

|  |  |  |
| --- | --- | --- |
| **名称** | **描述** | **备注** |
| 编辑场景 | 提供用户编辑场景，可在这里编辑房屋大小、位置信息。 | 需要友好的用户体验 |
| 添加房屋 | 用户可添加房屋数量。 |  |
| 生成模型 | 根据用户设置的房屋信息，生成室内场景。 |  |
| 场景漫游 | 用户可在生成的室内进行漫游。 |  |
| 编辑室内场景 | 用户可拍照，并将照片付在室内。 |  |

1. **系统框架图如图3-1**



图 3‑1

图 3‑2

1. **用户特征：**

任何拥有Android智能手机用户都可使用，没有技术要求，易上手。

1. **运行环境：**

硬件平台Android 2.3版本及以上，有1G及以上RAM，20M及以上ROM。

## 系统界面设计

### 编辑场景界面设计

编辑场景界面设计如图3-2（图片已经过反色处理）

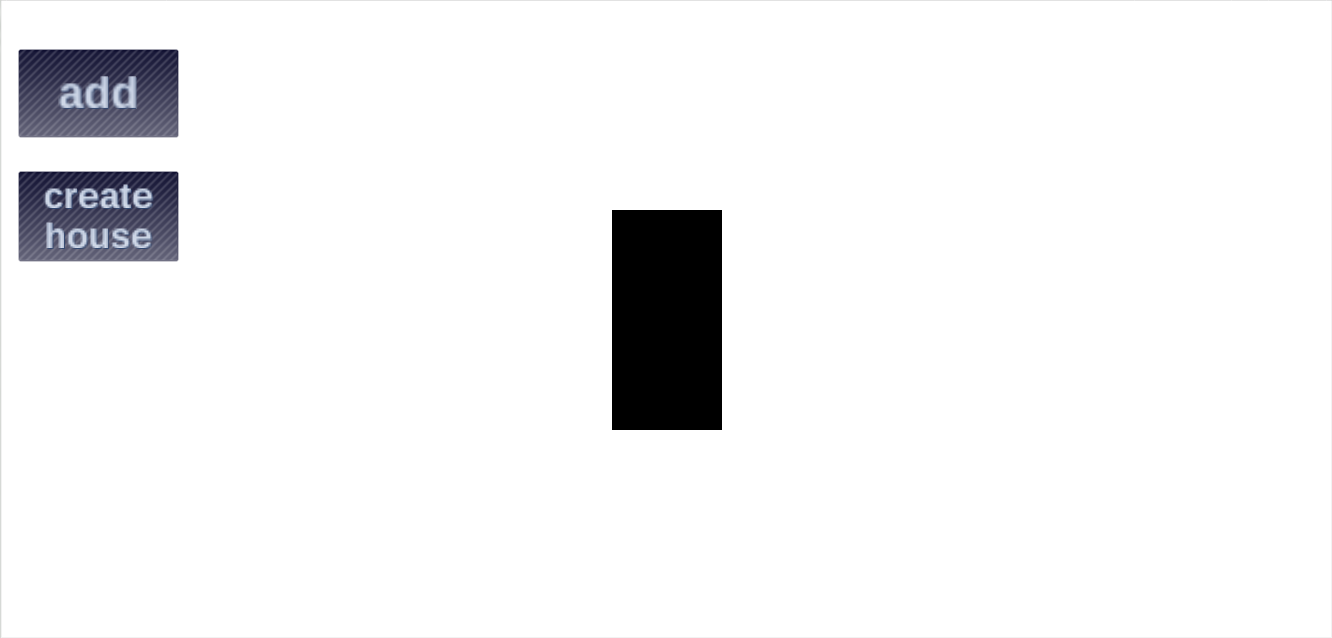


图 3‑3

点击“add”按钮后，添加房屋，并设置成不同颜色。

在编辑场景中，使用单手指在黑色区域滑动，可移动整体位置；

使用双手指滑动，可缩放整体大小；

在房屋区域（图3-3中白色区域）使用单手指滑动，可移动此房屋的位置，且房屋整体颜色变暗；在

房屋区域边缘使用单手指滑动，高亮边缘区域，房屋整体颜色变暗，并根据手势缩放此房屋的大小；

在对具体房屋进行编辑时，若选择的房屋某边（校对边）距离其他房屋某边（参考边）小于一个具体值时，在基准边位置显示基准线（半透明虚线，如图3-4，图片已经过反色处理）；

在基准线已经显示的情况下，若停止编辑房屋，则此房屋需吸附到基准线上。

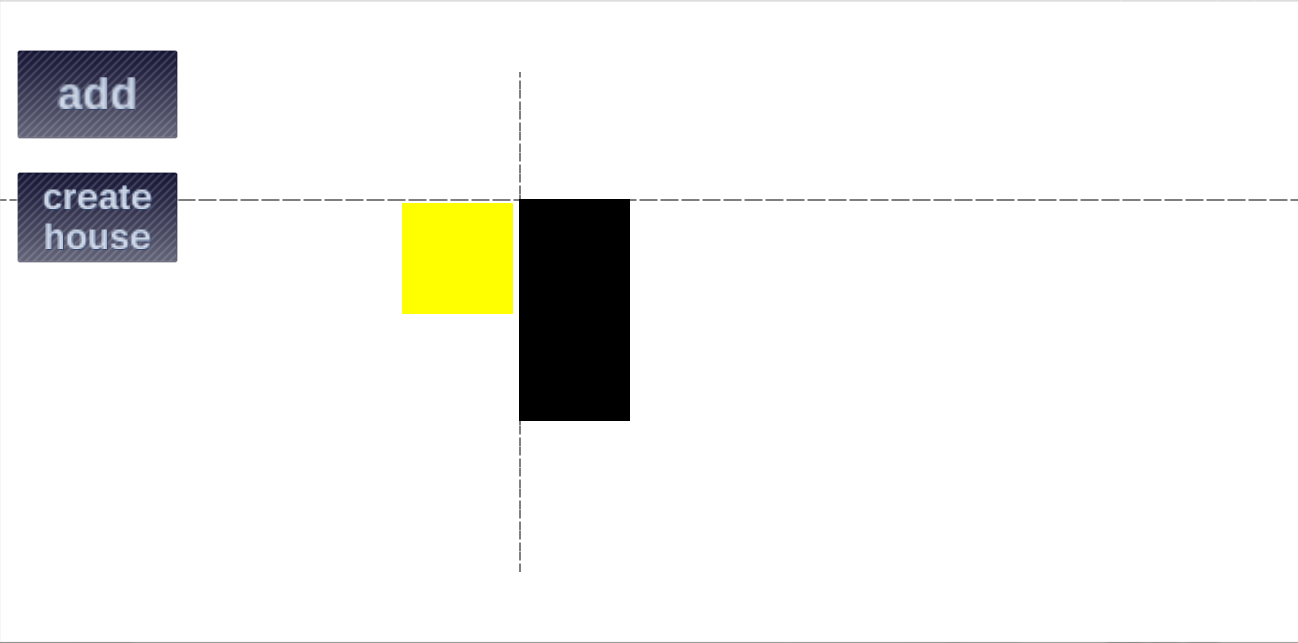


图 3‑4

### 场景漫游中的界面设计

1. **漫游模式界面设计如图3-5**

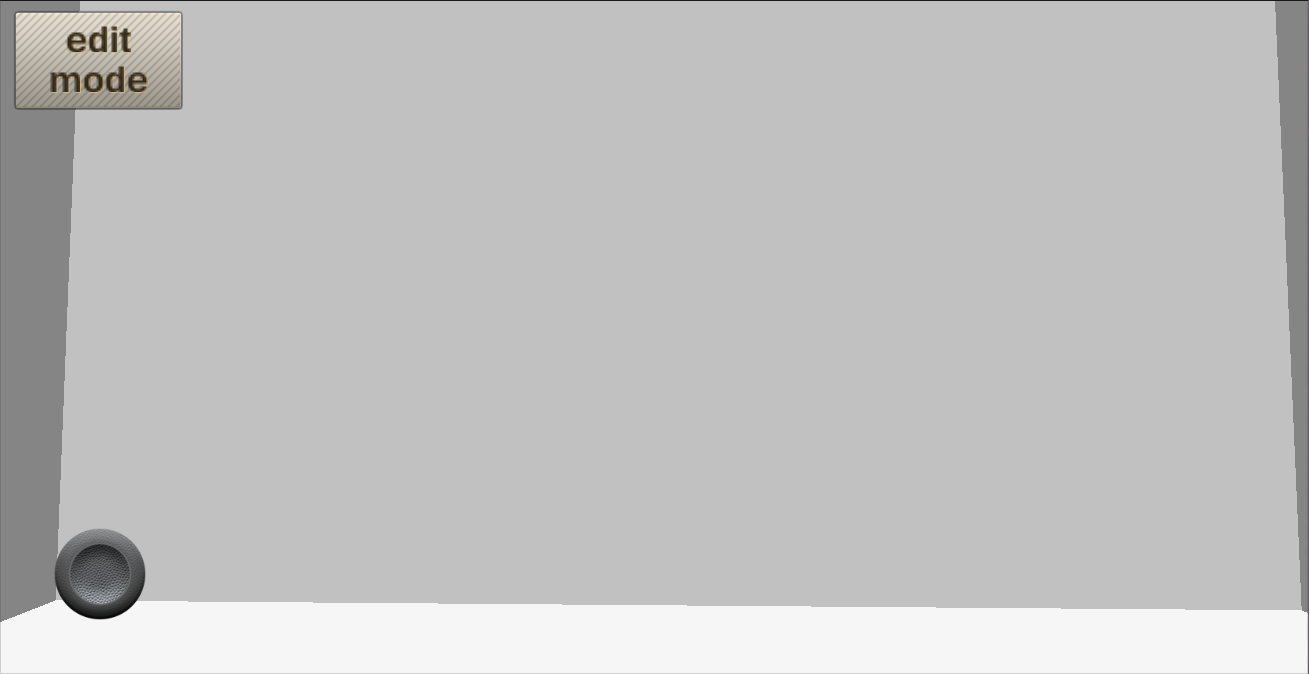


图 3‑5

在漫游模式中，点击按钮“edit mode”进入编辑模式；

左下角虚拟摇杆控制人物在场景中的走动；

单手指滑动屏幕，使人物改变方向。

1. **编辑模式界面设计如图3-6**

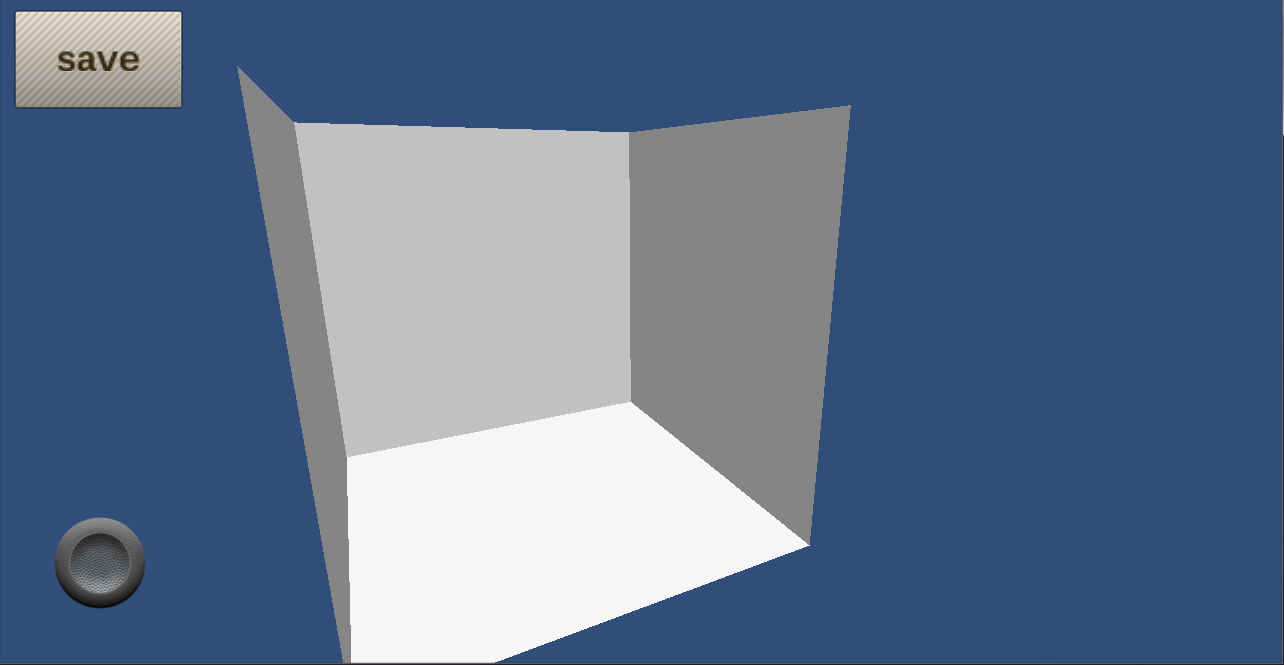


图 3‑6

编辑模式中，点击按钮“save”保存编辑；

左下角虚拟摇杆控制摄像机在场景中的位置；

单手指滑动屏幕，使摄像机改变方向；

单手指点击模型中的对应面，弹出对话框如图3-7（图片已经过反色处理）；

点击“pick picture”按钮，调出Android手机中的图片；

点击“take photo”按钮，调出Android手机中的摄像机；

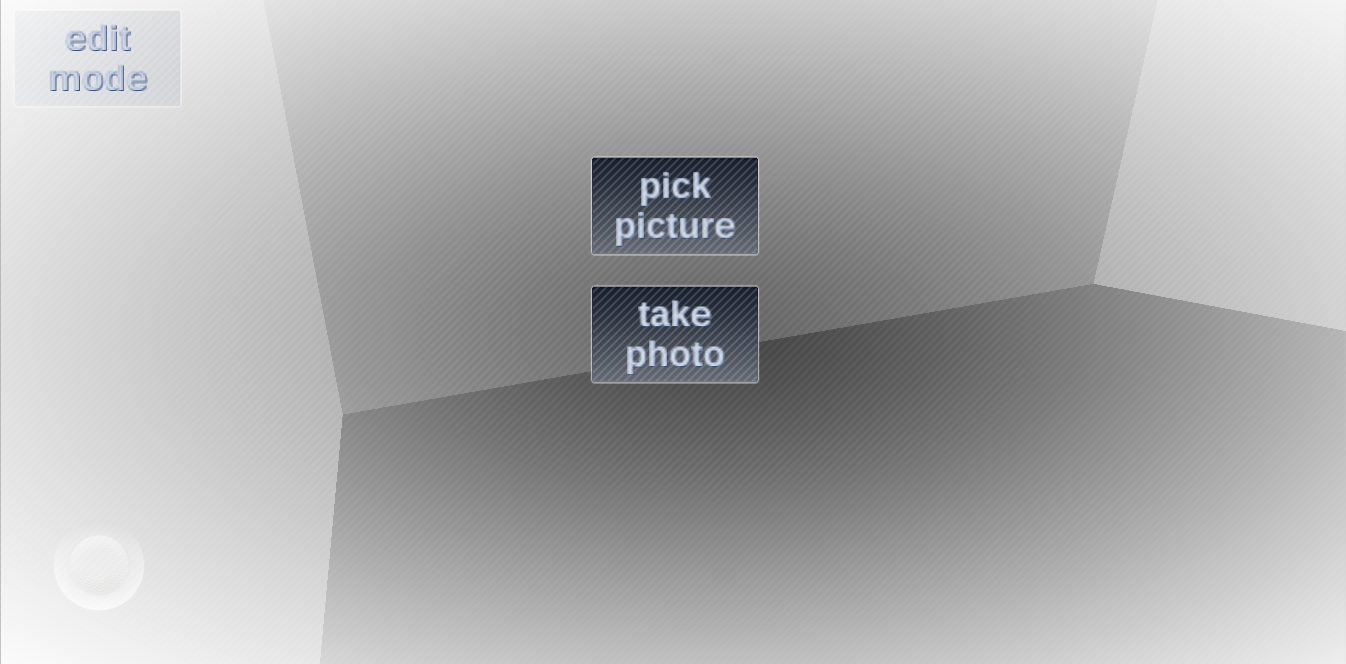


图 3‑7

### Android端界面设计

Android端界主要包含摄像机拍照界面、图库界面、编辑图片界面。这些界面都由Android系统组成，不需要额外设计。

## 系统详细设计

本系统分为Android端和Unity3D端，Unity3D端中分为编辑场景和漫游场景。

### Android端详细设计

1. **MUnityActivity模块：**

包含成员与接口：

* public static MUnityActivity mInstance；
* public void getPhoto(String request,String name)；

此模块需继承UnityPlayerActivity，来实现Android端与Unity3D端的互联；

mInstance成员，在onCreate方法中能够成功初始化，此为Unity端调用Android端方法的必备条件；

在getPhoto方法中开启新活动ViewActivity，将request参数以额外参数“type”传入到新活动中，name参数以额外参数”name“传入到新活动中。

1. **ViewActivity模块：**

包含成员与接口：

* public static final int NONE = 0;
* public static final int PHOTOHRAPH = 1;
* public static final int PHOTOZOOM = 2;
* public static final int PHOTORESOULT=3;
* public static final String TAKEPHOTO=”takePhoto”;
* public static final String PICKPHOTO=”pickPhoto”;

此模块需继承Activity，并重写其中的onCreate、onActivityResult和onKeyDown方法；

成员NONE：出错时，会使用此参数；

成员PHOTOHRAPH：调用系统相机拍照后返回的参数；

成员PHOTOZOOM：调用系统图库后返回的参数；

成员PHOTORESOULT：编辑完照片后返回的参数；

成员TAKEPHOTO：打开系统相机的参数；

成员PICKPHOTO：打开系统图库的参数；

在onCreate初始化中，根据mUnityPlayer传递的request参数，打开系统相机或图库；打开系统相机，传入PHOTOHRAPH参数；打开图库传入PHOTOZOOM参数；

在onActivityResult方法中，根据参数选择打开图像编辑器或存储图像；

编辑完成照片后，需要将照片信息，包括图片路径（以“filename“关键字保存）、图片大小（以”imagesize“关键字保存）以及mUnityPlayer传递的name参数（以” name“关键字保存）保存至SharedPreferences中，以便在Unity3D端读取；

保存信息后，向Unity3D端发送消息，读取图片资源，并结束此活动；

在onKeyDown方法中，添加当按下返回键，向Unity3D端发送消息，读取图片资源，并结束此活动。

### Unity3D端详细设计

1. **与Android通信模块—— NotifyCenter模块的详细设计：**

公共接口：

* public static void openCamera(string name);
* public static void openGallery(string name);

在此模块的静态初始化中，设置私有静态类参数\_plugin，如图3-8



图 3‑8

方法openCamera中，调用Android端getPhoto方法，传入参数“getPhoto “、” takePhoto “和name参数；

方法openGallery中，调用Android端getPhoto方法，传入参数“getPhoto”、” pickPhoto “和name参数；

1. **与Android通信模块——AndroidMethod模块的详细设计：**

公共接口：

* public void LoadImage();

方法LoadImage中，读取在Android端储存的图片信息，并将图片设置在相应的模型上。

1. **整体共享数据——shareData**

* public static Vector3 basePosition; base模型的位置信息；
* public static Vector3 baseScale; base模型的大小信息；
* public static List<Vector3> positions; 其他模型的位置信息；
* public static List<Vector3> scales; 其他模型的大小信息；
* public static List<float> topPos; 其他模型的上边位置信息；
* public static List<float> bottomPos; 其他模型的下边位置信息；
* public static List<float> leftPos; 其他模型的左边位置信息；
* public static List<float> rightPos; 其他模型的右边位置信息；
* public static float btPos; base模型的上边位置信息；
* public static float bbPos; base模型的下边位置信息；
* public static float blPos; base模型的左边位置信息；
* public static float brPos; base模型的右边位置信息；

1. **编辑场景——模型详细设计：**

* 光照：全局光照，白色光，强度为0.5；
* 主相机：透镜类型相机，渲染除UI以外的所有模型，位置(0,0,-50)，深度-1，添加脚本UIInEditModel，设置脚本中Parent Planes为模型中的planes，Base Plane为base，Prefab为prefabPlane，Camera为MainCamera，Dotted Line Controller为UIRoot，Main Camera为模型中的MainCamera，Min Offset Of Model为20，Zoom Speed为5，Move Speed为1；
* 编辑场景UI
* base模型
* planes容器
* shareObject 共享数据容器，添加脚本shareData。
* 编辑场景UI结构如图3-9（图片已经过反色处理）

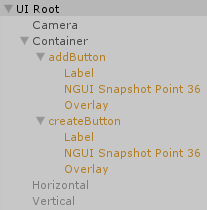


图 3‑9

**UI Root：**附加脚本DottedLineController，设置脚本中的Horizontal为模型中的Horizontal，Vertical为模型中的Vertical，UICamera为模型中的Camera；

**Camera：**深度1.68，事件类型为3D UI，只渲染NGUILayer；

**Container：**深度3，将其和UI Root联接，使之能随屏幕大小改变而改变；

**addButton：**相对位置(-63,68,0)，深度0，将其和Container联接，响应模型MainCamera，响应事件UIInEditModle中的addPlane；设置Label的Text为“add”，字体大小50，深度2

**createButton：**相对位置(-63,-55,0)，深度0，将其和Container联接，响应模型MainCamera，响应事件UIInEditModle中的createHouse；设置Label的Text为“create house”，字体大小50，深度2

**Horizontal：**初始设为隐藏，贴图设为虚线贴图，类型Tiled，半透明度50%，深度1，大小1000 \* 2，使用高级模式将其和UI Root联接，使之能在x轴上移动，但不能在y轴上移动，添加脚本MoveXu，设置脚本中的Orientation为HORIZONTAL，UICamera为模型中的Camera；

**Vertical：**初始设为隐藏，贴图设为虚线贴图，类型Tiled，半透明度50%，深度1，大小1000 \* 2，使用高级模式将其和UI Root联接，使之能在y轴上移动，但不能在x轴上移动，添加脚本MoveXu，设置脚本中的Orientation为VERTICAL，UICamera为模型中的Camera；

* base模型结构如图3-10（图片已经过反色处理）

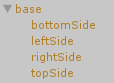


图 3‑10

**base：**旋转角度X=-90，缩放大小Z=2，添加脚本EditModel设置Speed为115；

**子模型：**分别在base模型的4边，缩放其宽度为0.1，长度与base模型保持一致，添加Zoom脚本，设置Speed为10；

将base模型保存为prefabPlane。

* planes容器

编辑场景中，每点一次“add”按钮，将prefabPlane实例化后并添加到planes容器中；

1. **编辑场景——UIInEditModle**

公共参数和接口：

* public Transform parentPlanes; planes容器
* public Transform basePlane; base模型
* public GameObject prefab; 模型预设
* public Transform camera; MainCamera的Transform
* public DottedLineController dottedLineController; 控制校准线
* public Camera mainCamera; MainCamera
* public float minOffsetOfModelPosition; 当模型间距离小于此值时，显示校准线
* public float zoomSpeed; 缩放整体的速度
* public float moveSpeed; 移动整体的速度
* public static UIInEditModle shareInstance; 共享实例
* public static bool canMove = true; 是否能移动整体
* public void addPlane();
* public void createHouse();

此模块需继承MonoBehaviour类，并重写Start和Update方法；

Start方法中初始化shareInstance等必要数据，并初始化共享数据；

Update方法中根据触摸点数移动或缩放整体；

方法addPlane为按钮“add”响应事件，点击一次按钮，在不同位置实例化一个prefabPlane，设置不同的颜色，并以不同数字作为其名称，完成后，将其位置信息保存；

方法createHouse为按钮“create house”响应事件，点击按钮，保存模型数据，并打开新场景；

1. **编辑场景——editModel**

公共参数和接口：

* public int speed; 模型移动速度；
* public string modelName; 模型名称；
* public bool isOnMouseDrag; 是否正在被移动；
* public bool overModel; 鼠标是否在模型上（电脑端使用）；
* public float minOffset; 当模型间距离小于此值时，显示校准线；
* public enum SIDE{} 枚举边类型；
* public enum MODE{} 枚举校准线类型；

此模块需继承MonoBehaviour类，并重写Start、Update、OnMouseDrag、OnMouseOver、OnMouseExit和OnMouseUp方法；

Start方法中初始化必要数据；

Update方法中根据触摸点数及位置，移动或显示校准线，并保存模型信息；

OnMouseDrag方法中移动模型到某位置；

OnMouseOver方法中设置overModel为true；

OnMouseExit方法中设置overModel为false；

OnMouseUp方法中设置当校准线显示时，将模型移动到校准线位置，并使校准线消失；

SIDE包含{LEFT,RIGHT,TOP,BOTTOM}；

MODE包含{VERTICAL,HORIZONTAL}；

1. **编辑场景——zoom**

公共参数和接口：

* public int speed; 缩放速度；
* public string parentName; 父模型名称；
* public bool onDrag; 是否在被拖动；
* public bool overModel; 鼠标是否在模型上（电脑端用）；

此模块需继承MonoBehaviour类，并重写Start、Update、OnMouseDrag、OnMouseOver、OnMouseExit和OnMouseUp方法；

Start方法中初始化必要数据；

Update方法中根据触摸点数及位置，高亮边缘或缩放模型，并保存模型信息；

OnMouseDrag方法中缩放模型到某大小；

OnMouseOver方法中设置overModel为true，设置父模型中的overModel为true；

OnMouseExit方法中设置overModel为false，设置父模型中的overModel为false；

OnMouseUp方法中设置onDrag为false，设置父模型中的isOnMouseDrag为false，设置UIInEditModle.canMove为true；

1. **编辑场景——DottedLineController**

公共参数和接口：

* public GameObject horizontal; 竖线模型；
* public GameObject vertical; 横线模型；
* public Camera UICamera; UI摄像机；
* public static DottedLineController shareInstance; 共享实例；
* public static bool verticalLineIsAppear; 横线是否显示；
* public static bool horizontalLineIsAppear; 竖线是否显示；
* public void setDottedLine (Vector3 horizontalPosition,Vector3 verticalPostion);
* public void appear(Vector3 position, ORIENTATION appear);
* public void disappear(ORIENTATION o);

此模块需继承MonoBehaviour类，并重写Start方法

Start方法中初始化必要参数；

setDottedLine方法将横线竖线分别移到对应参数位置；

appear方法根据appear参数设置对应校准线位置并显示校准线；

disappear方法根据参数o隐藏对应校准线；

1. **编辑场景——MoveXu**

公共参数和接口：

* public ORIENTATION orientation; 此模型的类型；
* public Camera UICamera； UICamera；
* public enum ORIENTATION{} 校准枚举类型；
* public void moveTo(Vector3 moveToPos)；

此模块需继承MonoBehaviour类；

ORIENTATION包含{VERTICAL,HORIZONTAL,ALL}；

moveTo方法将校准线移动到参数位置；

1. **漫游场景——模型详细设计**

* 光照：双全局光照，白色光，强度为0.5；
* 主相机：初始为关闭状态，透镜类型相机，渲染除UI以外的所有模型，位置(0,-1,-1)，深度0，添加脚本EditMode，设置脚本中Base Box为模型中的base，Parent为模型中的Parent，Mode Change Button为模型中的ModeChangeButton，Mode Button Label为模型中ModeChangeButton下的Label，First Person Controller为模型中的FirstPersonController，Edit设置为模型中的EDIT，Pick Picture Button为pickPicture，Take Photo Button为takePhoto，Main Camera设置为模型中的Camera，Zoom Speed为0，Move Speed为1，Joy为模型中的joy；
* First Person Controller：添加SetCence和EditMode脚本；

在SetCence中设置BaseBox为模型重的base，Parent为模型中的Parent，Prefab为预设box；

在EditMode中设置Base Box为模型中的base，Parent为模型中的Parent，Mode Change Button为模型中的ModeChangeButton，Mode Button Label为模型中ModeChangeButton下的Label，First Person Controller为模型中的FirstPersonController，Edit设置为模型中的EDIT，Pick Picture Button为pickPicture，Take Photo Button为takePhoto，Main Camera设置为模型中的Camera，Zoom Speed为0，Move Speed为1，Joy为模型中的joy；

* AndroidMethod：添加AndroidMethod脚本；
* 漫游场景UI
* base模型
* Parent容器
* 漫游场景UI结构如图3-11（图片已经过反色处理）



图 3‑11

**UI Root：**深度0；

**Camera：**深度1，事件类型为3D UI，只渲染NGUILayer；

**Container：**深度1，将其和UI Root联接，使之能随屏幕大小改变而改变；

**ModeChangeButton：**位置(-63,101,0)，深度2，大小170 \* 100，与Container联接，响应事件FirstPersonContrller，响应EditMode中的onModeButtonClick方法；设置Label中的Text为edit mode，字体大小50，深度3；

**EDIT：**深度为3，将其和UI Root联接；

**Control – Background：**深度2，大小1342 \* 668，将其和EDIT联接；

**pickPicture：**位置(-9,127.5,0)，深度3，大小170 \* 100，将其和EDIT联接，点击响应事件First Person Controller，响应EditMode中的onPickPictureClick方法；设置Label中的Text为pick picture，字体大小50，深度4；

**takePhoto：**位置(0,-1,0)，深度3，大小170 \* 100，将其和EDIT联接，点击响应事件First Person Controller，响应EditMode中的onTakePhotoClick方法；设置Label中的Text为take photo，字体大小50，深度4；

* base模型结构如图3-12（图片已经过反色处理）



图 3‑12

**base：**添加脚本Box，将子模型对应设置在脚本上；

**子模型：**分别是base模型的5边；

将base模型保存为box预设。

* Parent容器

从编辑场景进入漫游场景时，将box预设实例化后，根据共享数据设置其参数，并放入Parent容器中；

1. **漫游场景——SetCence**

公共参数和接口：

* public GameObject baseBox; base模型；
* public Transform parent; parent容器；
* public GameObject prefab; 预设内容；

此模块需继承MonoBehaviour，并重写Start方法；

Start方法中根据共享数据，设置房屋模型的参数（大小位置）；

1. **漫游场景——EditMode**

公共参数和接口：

* public GameObject baseBox; base模型；
* public GameObject parent; parent容器；
* public GameObject modeChangeButton; ModeChangeButton按钮；
* public UILabel modeButtonLabel;模型中ModeChangeButton下的Label；
* public GameObject firstPersonController; 模型中的FirstPersonController；
* public GameObject edit; 模型中的EDIT容器；
* public GameObject pickPictureButton;模型中的pickPicture按钮；
* public GameObject takePhotoButton; 模型中的takePhoto按钮；
* public GameObject mainCamera; 模型中的Camera；
* public float zoomSpeed; 缩放速度；
* public float moveSpeed; 移动速度；
* public Joystick joy; 虚拟摇杆；
* public static EditMode shareInstance ; 共享实例
* public static bool canMove = true; 能否移动；
* public static bool isInEditMode = false;是否在编辑模式中；
* public void setTexture(Texture t , string name , string side);
* public void setPictureSize ( int width , int height ) ;
* public void onModeButtonClick();
* public void onPickPictureClick();
* public void onTakePhotoClick();

此模块需继承MonoBehaviour，并重写Start和Update方法；

setPictureSize方法用来将贴图更好的适配模型

setTexture方法根据name调用对应box模型中的setTexture方法；

onModeButtonClick方法是按钮响应方法，点击此按钮将在漫游模式和场景编辑模式中切换；

onPickPictureClick方法是按钮响应方法，点击此按钮将调用Android端的图库程序；

onTakePhotoClick方法是按钮响应方法，点击此按钮将调用Android端的拍照程序；

Start方法中初始化一些必要参数；

Update方法中根据模式，和手势移动相机或人物；

1. **漫游场景——Box**

公共参数和接口：

* public GameObject left; 子模型中的left；
* public GameObject front; 子模型中的front；
* public GameObject right; 子模型中的right；
* public GameObject back; 子模型中的back；
* public GameObject bottom; 子模型中的bottom；
* public void setBox(Vector3 scale);
* public void setTexture (Texture t , string side);
* public void setBoxByAspect ( int x , int y ) ;

此模块需继承MonoBehaviour，并重写Start方法；

setBox方法根据所给大小参数设置房屋模型；

setTexture方法根据side参数将贴图付到对应side的模型上；

setBoxByAspect 方法根据参数x和参数y设置模型的高；

Start方法中初始化一些必要参数；

## 系统的实现

### 编辑场景的UI实现

1. 在Unity3D菜单栏中选择NGUI->Create->Widget，将会自动创建出UI Root及其子元素Camera和Container；
2. 设置Container大小，将其调整到和UI Root一样大，并设置其深度为3；
3. 设置Camera中的UICamera的Event Type为3D UI（由于需要使用按钮，及按钮的一些动画，因此选择用3D UI）；
4. 在项目文件栏中的All Prefabs中找到Control – Colored Button，将其拖入到Container下，成为Container的子元素；
5. 设置按钮的名称为“addButton”，设置其位置、大小、颜色和深度，设置UISprite下的Anchors的Type为Unified，设置其Target为Container，设置其Left，Right为Target’s Left，Bottom和Top为Target’s Top，在UIButton中的On Click中设置Notify为Main Camera，Method为UIInEditModle/addPlane；
6. 设置”addButton“子元素Label的Font Size为50，Text为”add“，深度为2；
7. 重复第4到第6步，添加createButton按钮，设置其On Click的Method为UIInEditModle/createHouse，设置其子元素Label的Text为“create“；
8. 在Unity3D菜单栏中选择NGUI->Create->Sprite，将会在UI Root下创建一个Sprite元素；
9. 设置Sprite元素名称为Horizontal，默认隐藏，旋转Z轴90度，在UISprite中将虚线的Sprite添加进去，设置大小为1000 \* 2，设置Anchors下的Type为Advanced，Bottom为UI Root的Center，Top为UI Root的Center，Left和Right置空；
10. 向Horizontal中添加Move Xu脚本，设置Orientation为HORIZONTAL，UICamera为UI Root下的Camera；
11. 重复第8到第10步，添加名称为Vertical的Sprite元素，无旋转，Anchors下的Left为UI Root的Left，Right为UI Root的Right，Top和Bottom置空，Move Xu中设置Orientation为VERTICAL；

### 编辑场景的场景实现

1. 根据编辑场景的模型详细设计，将灯光、摄像机、base模型、planes容器和shareObject共享数据制作完成后放入层次清单栏中；
2. 在UIInEditModle脚本中，使用预编译关键字“#if UNITY\_EDITOR”，“#else”，“#endif”，分别处理在Unity3D端的功能和在手机端的功能；

在Unity3D环境中，使用“Left Shift”来控制整体的放大，“Left Control”来控制整体的缩小，使用鼠标左键的拖动来实现整体的移动；

在手机环境中，若单指操作，则移动整体，若两指操作，则根据手势放大或缩小整体；

在手机环境中，每帧通过式（3-1）计算出两指间距，然后比较间距，若变大，表明正在做放大手势，若变小，表明正在做缩小手势

（3-1）

通过移动Main Camera的x和y轴来控制整体的移动，移动其y轴来控制整体的缩放。

在此脚本中还需要实现添加房屋的方法，首先通过代码：

GameObject prego = Instantiate ( prefab ) as GameObject ;

实例化prefabPlane预设，再设置其位置颜色等属性即可。

在createHouse方法中，保存数据后，通过代码：

Application . LoadLevel ( " cence1 " ) ; 加载漫游场景。

1. 在EditModel中，同样通过预编译来分别处理Unity3D和手机端功能；

在此脚本中，实现每帧根据单指或鼠标移动来移动模型，并通过关键算法“基准线位置算法”来判断是否显示以及在什么位置显示基准线。

1. 在zoom中，同样通过预编译来分别处理Unity3D和手机端功能；

在此脚本中，实现每帧根据单指或鼠标移动来缩放模型，并通过关键算法“基于特定边的缩放算法”来缩放模型。

1. DottedLineController和MoveXu脚本属于控制器，用来控制基准线的位置和显示；

### 漫游场景的UI实现

1. 方法同编辑场景的UI实现，不同的是，需要添加两个Widget，一个名为Container，一个名为EDIT，且EDIT深度比Container高；
2. 在Container中有一个按钮，名为 “ ModeChangeButton “ ，在屏幕左上角位置，通过设置Anchors来适配多分辨率；
3. 在EDIT中有两个按钮和一个背景，两按钮为 “ pickPicture “和 “ takePhoto” ，在屏幕中间位置；背景为半透明黑色背景，覆盖整个屏幕；通过设置Anchors来适配多分辨率；

### 漫游场景的场景实现

1. 根据漫游场景的模型详细设计，将灯光、摄像机、base模型、Parent容器、第一人称控制器和joy虚拟摇杆制作完成后放入层次清单栏中；
2. EditMode中，实现onModeButtonClick方法，若点击时，在漫游模式，则切换到编辑模式，使Camera相机工作，隐藏第一人称控制器，并使房屋模型中的墙面可点击；若不在漫游模式，则切换到漫游模式，使Camera隐藏，使第一人称控制器工作，并使房屋模型中的墙面不可点击；

在编辑模式中，若点击墙面，显示EDIT容器中所有模型；

在脚本中实现onPickPictureClick方法，调用通信模块的 NotifyCenter模块中的openGallery方法，并传入墙面父元素的名称和墙面名称；

实现onTakePhotoClick方法，调用信息模块的NotifyCenter模块中的openCamera方法，并传入墙面父元素的名称和墙面名称；

### Unity3D端与Android通信模块的实现

1. NotifyCenter脚本中，根据设计做对应初始化，并实现openGallery方法和openCamera方法，通过代码：

\_plugin . Call ( “getPhoto “ , “pickPhoto” , name ) ;

或 \_plugin . Call ( “getPhoto “ , “takePhoto” , name ) ;

调用Android端方法；

1. AndroidMethod脚本中，实现LoadImage方法，使用代码

PlayerPrefs . GetString ( “ imagesize ” ) ; 获取imagesize参数；

PlayerPrefs . GetString ( " filename “ ) ; 获取filename参数；

PlayerPrefs . GetString ( " name “ ) ; 获取name参数；

并通过协程：StartCoroutine; 加载图片形成贴图；同时调用SetImage方法；

实现LoadTexture ( stirng filename , string name )方法，加载图片形成贴图并调用EditMode中的setTexture方法；

实现SetImage ( string size ) 方法，获取图片分辨率，并调用EditMode中的setPictureSize方法；

### Android端的实现

1. MUnityActivity中实现getPhoto方法，通过代码：

Intent intent = new Intent ( this , ViewActivity . class ) ;

intent . putExtra ( “ type “ , request ) ;

intent . putExtra ( “ name “ , name ) ;

this . startActivity ( intent ) ;

来实现传递两个参数并打开新活动；

1. ViewActivity中通过代码：

Intent intent = new Intent ( MediaStore . ACTION\_IMAGE\_CAPTURE ) ;intent . putExtra ( MediaStore . EXTRA\_OUTPUT , Uri.fromFile ( new File ( Environment . getExternalStorageDirectory () , " temp.jpg " ) ) ) ;startActivityForResult ( intent , PHOTOHRAPH ) ;

打开摄像机；

通过代码：

Intent intent = new Intent ( Intent . ACTION\_PICK , null ) ;

intent . setDataAndType ( MediaStore . Images . Media . EXTERNAL\_ CONTENT\_URI , IMAGE\_UNSPECIFIED ) ;

startActivityForResult ( intent, PHOTOZOOM ) ;

打开图库；

通过代码：

SharedPreferences sp = this . getSharedPreferences ( getPackageName () , 0 );

实例化SharedPreferences；

并通过代码：

sp . edit () . putString ( “ imagesize “ , size ) . commit ();

sp . edit () . putString ( “ filename “ , filename ) . commit ();

sp . edit () . putString ( “ name “ , name ) . commit ();

将图片信息保存；

通过代码：

UnityPlayer . UnitySendMessage ( " AndroidMethod " , " LoadImage " ) ;

调用Unity3D端方法；

## 关键算法实现

### 基于特定边的缩放算法

为了让用户在缩放模型时感觉更加友好，在拖拽某条边时，除了根据手势缩放其对应方向的大小外，其他方向不能有缩放，并且模型的其他位置不能移动，所以设计此算法来解决这个问题。

首先，根据用户触摸的边，来判定只启用横向缩放还是纵向缩放，并使用代码：

Input . GetAxis ( “ Mouse X “ ) 或 Input . GetAxis ( “ Mouse Y “ )

来获取X轴或Y轴方向上每帧的偏移量，让模型整体比例减去在X轴或Y轴上的偏移量，就得到了模型在每一帧中的比例；

由于直接修改X轴（或Y轴）的比例，会使模型整体在X轴（或Y轴）两边都有缩放（这样让模型的中心位置没有变化），所以需要修改模型整体的中心位置。在每一帧中，将模型中心位置的X轴（或Y轴）减去第一步得出的偏移量乘以5（由于Unity3D单位的关系，这里需要乘以5），就使得在效果上达到初始目的。最后需要注意将第一步的偏移量减半，缩放比例才能和预计一致。

最后还需要做一步处理。由以上两步能使整体模型的放大缩小达到初始目的，但是在缩放整体模型时，其子模型，也就是其四边，都会随着整体模型的缩放而缩放，因此最后需要修正其子模型的大小。由于改变X轴方向的大小，会改变整体模型X轴方向的大小，改变Y轴方向大小，会改变整体模型Z轴方向大小，切控制子模型的粗细都是子模型的X轴，所以当调整左右两边大小时，将左右两边子模型的X轴比例设为0.1 （原始大小）除以当前模型比例的X轴数据；当调整上下两边大小时，将上下两边子模型的Y轴比例设为0.1（原始大小）除以当前模型比例的Z轴数据。

通过这三步，缩放模型时，将有非常友好的用户体验。

### 基准线位置算法

当用户编辑模型时，如果用户自己慢慢调整模型位置，将不能很好的贴住其他模型的边缘，从而在生成模型时，会有模型重叠等问题。因此设计此算法，当用户移动模型时，若挨近别的模型边缘，将会出现位置基准线，此时若停止触摸模型，则将模型贴在基准线上。

当用户移动模型时，则计算离其最近的其他模型的边，若此值小于一个预设值时，则在适当位置显示基准线。但是若在每帧中去场景里的每个模型里寻找最近边，将有很大计算开销。因此在shareData共享数据中，设计了4个浮点型参数来记录base模型4条边的X轴或Y轴参数（上下边记录Y轴参数，左右边记录X轴参数），还设计了4个浮点型列表来记录非base模型4条边的X轴或Y 轴参数。又因为4条边的父元素位置不同，所以不能简单的以4条边的坐标做对比，必须将4条边的世界坐标转换为屏幕坐标后再做对比。所以shareData共享数据中设计的8个参数都用来记录4条边对应的屏幕坐标参数。

首先，用户移动整体场景、缩放整体场景、移动模型位置和缩放模型位置时，都记录场景中模型4条边的屏幕坐标参数。

然后，在移动模型时，将模型的4个边坐标换为屏幕坐标，并对比shareData中的各个参数。将模型的上下两边和其他模型的上下两边进行两两对比，取出竖直方向最小距离差，将模型的左右两边和其他模型的左右两边进行两两对比，取出水平方向最小距离差。

将上一步得出的两个距离差分别比较预设值（设置其为20在手机端有很好的体验），若小于预设值，判断匹配边类型，若是竖直方向，则将基准线屏幕坐标的Y轴设置为匹配边的屏幕坐标（shareData保存的对应数据），并显示竖直方向基准线，若是水平方向，则将基准线屏幕坐标的X轴设置为匹配边的屏幕坐标，并显示水平方向基准线。

最后，若在显示基准线的情况下停止触摸模型，将竖直校准边屏幕坐标的Y轴数据减去竖直基准屏幕线屏幕坐标的Y轴数据，得到竖直方向的距离，用水平校准边屏幕坐标的X轴数据减去水平校准边屏幕坐标的X轴数据，得到水平方向的距离。将此模型的屏幕坐标X轴和Y轴分别减去水平方向距离和竖直方向距离，并将屏幕坐标转换为世界坐标就能达到吸附的效果。

## 本章小结

本章是本论文中的核心章节，介绍了基于用户友好系统的需求，并进行了详细的设计工作，然后将系统各模块的具体实现工作完整展示出来，最后详细讲解了系统中涉及到的关键算法。

下一章将会对实现的系统做功能及性能测试，并分析结果。

# 系统运行及性能分析

## 运行环境

系统运行在2011年上市的HTC 霹雳2（Rezound）Android手机下，具体参数如下：

操作系统：Android OS 4.0.3

CPU：高通 骁龙Snapdragon MSM8660 双核 1.5HZ

GPU：高通 Adreno220

RAM：1GB

主屏分辨率：1280x720 像素

后置摄像头：800万像素

## 运行效果

开启程序后，如图4-1（图片已经过反色处理）

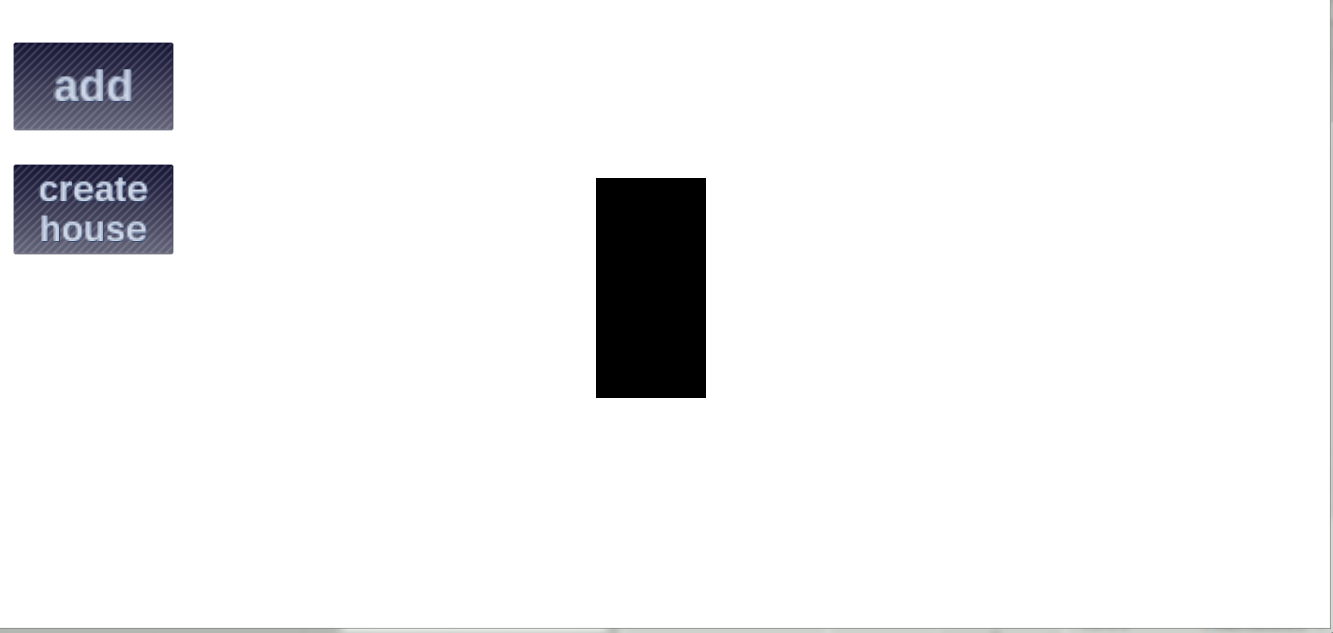


图 4‑1

点击两次“add”按钮，生成2个新模型，如图4-2（图片已经过反色处理）

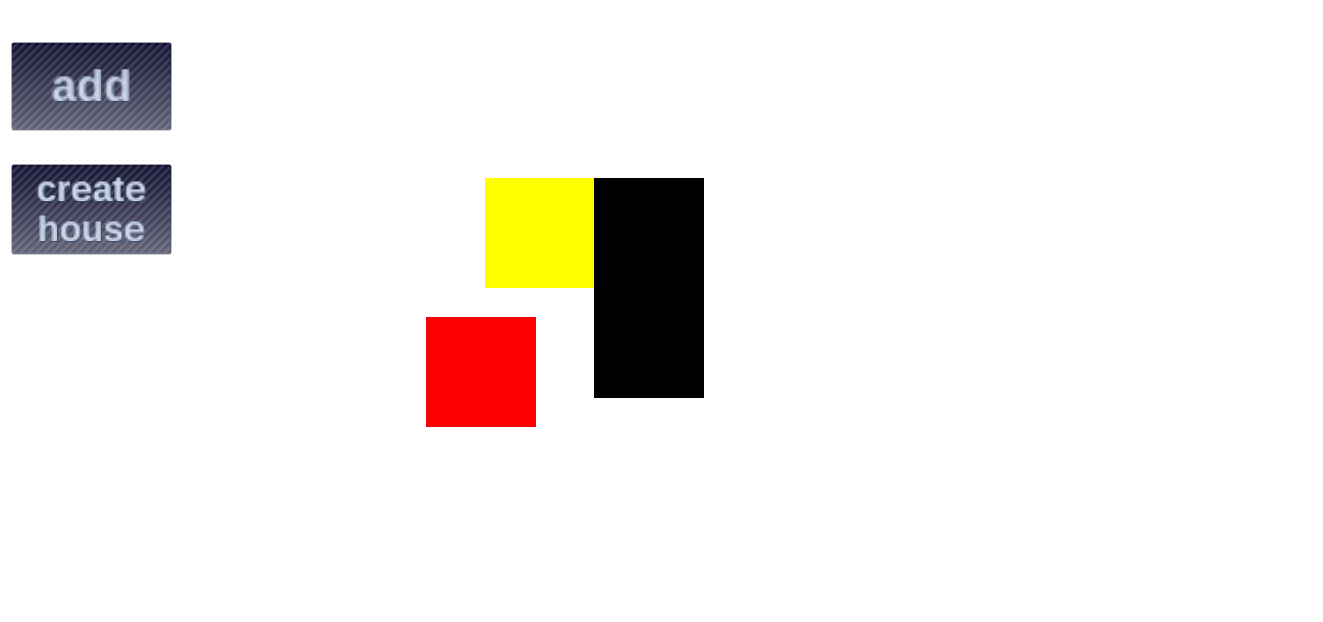


图 4‑2

当手指触摸浅蓝色模型时，模型变色，如图4-3（图片已经过反色处理）

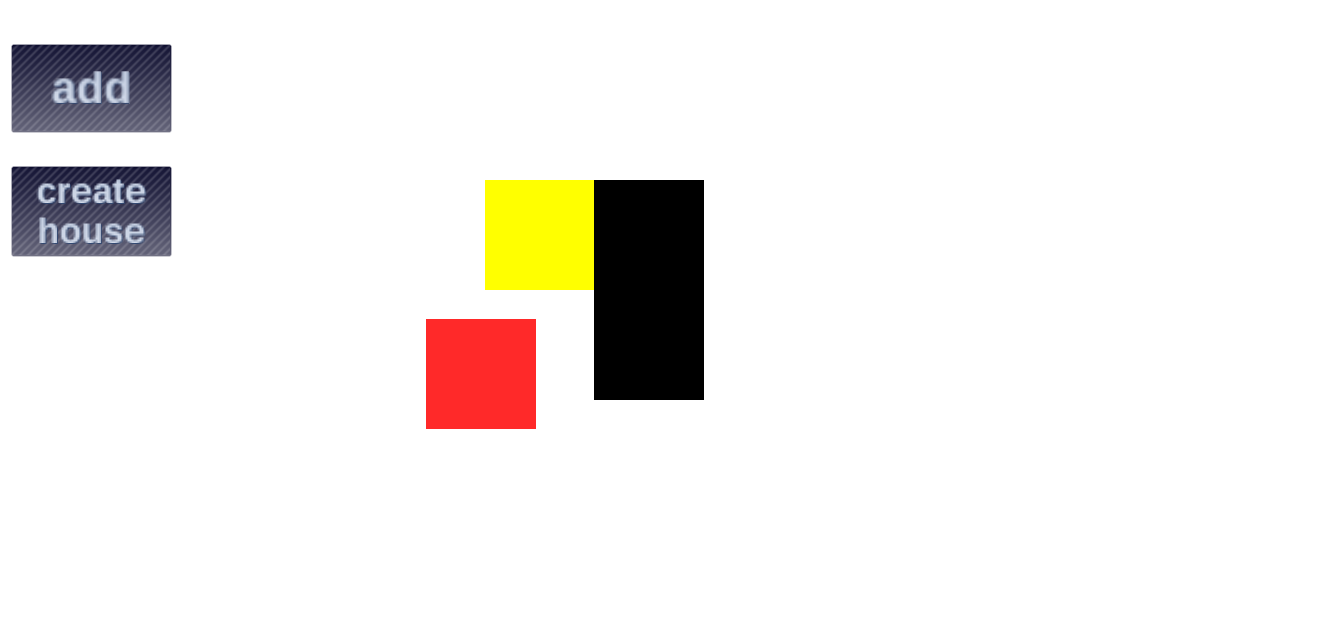


图 4‑3

手指放在浅蓝色模型下边上，下边出现白色，如图4-4（图片已经过反色处理）

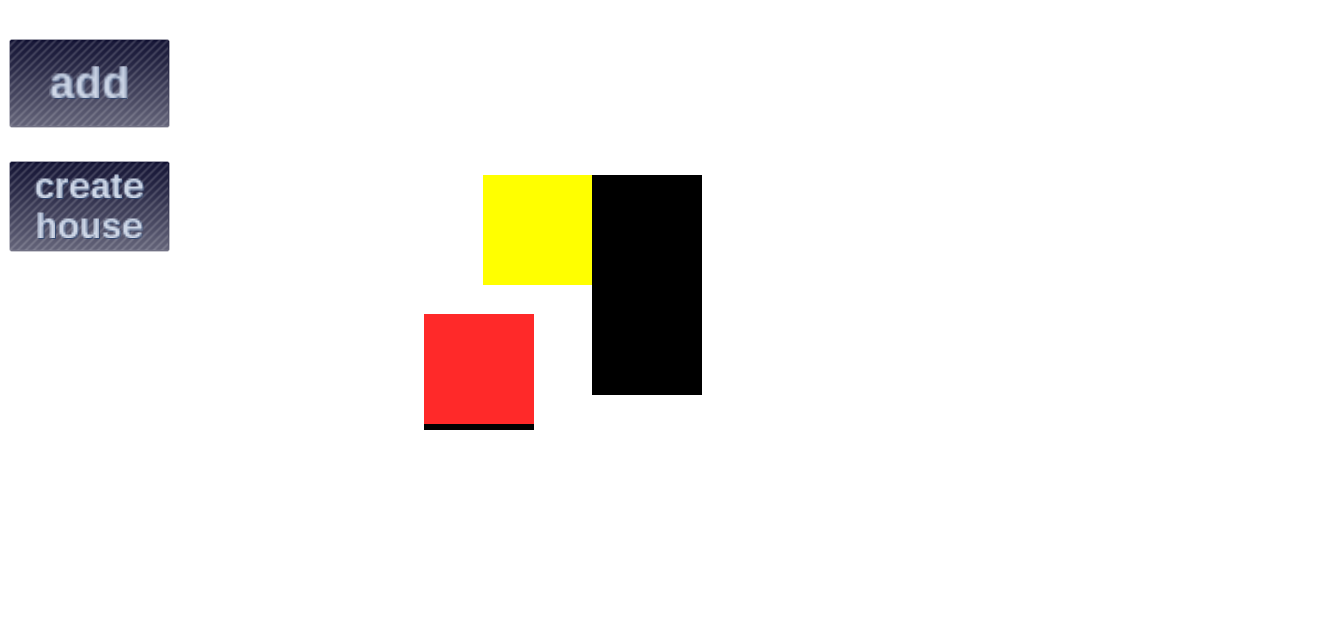


图 4‑4

移动浅蓝色模型到靠近另两个模型处，出现虚线基准线，如图4-5（图片已经过反色处理）

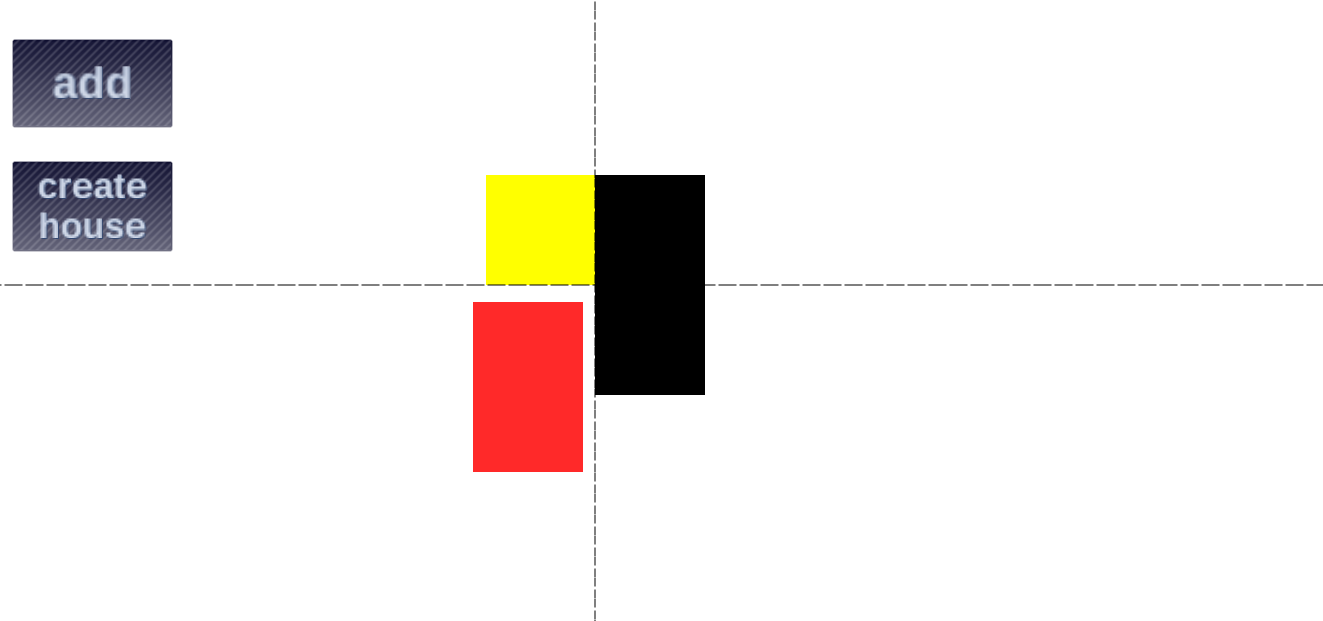


图 4‑5

放开手指，浅蓝色模型自动吸附，如图4-6（图片已经过反色处理）

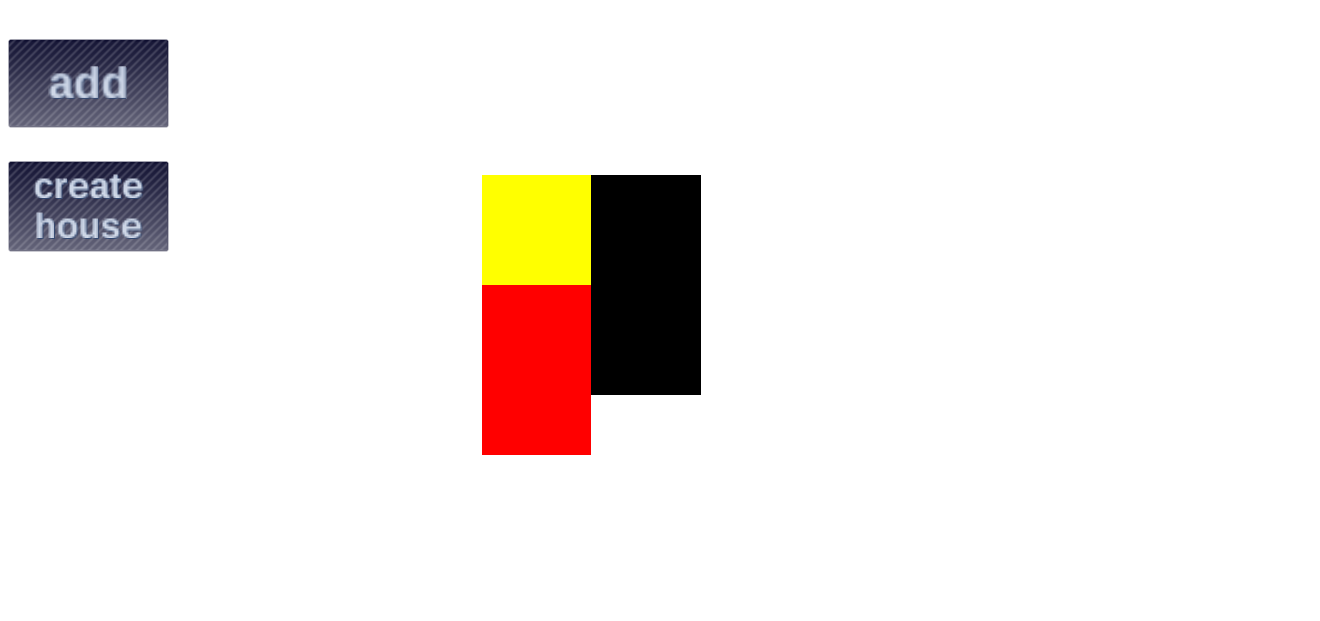


图 4‑6

点击“create”按钮，进入漫游模型，如图4-7

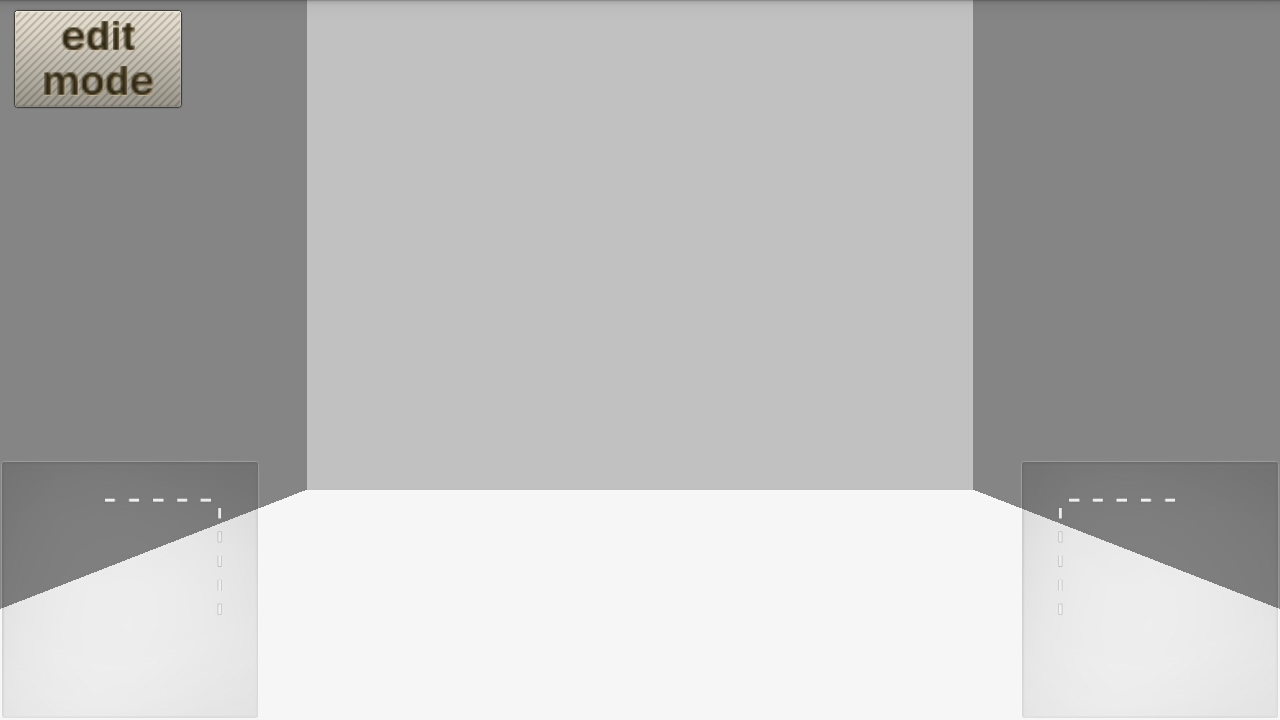


图 4‑7

点击“edit mode”按钮，并移动摄像机到一定位置，如图4-7，观察到模型和编辑场景中模型位置和相对大小都相同，达到预期效果。

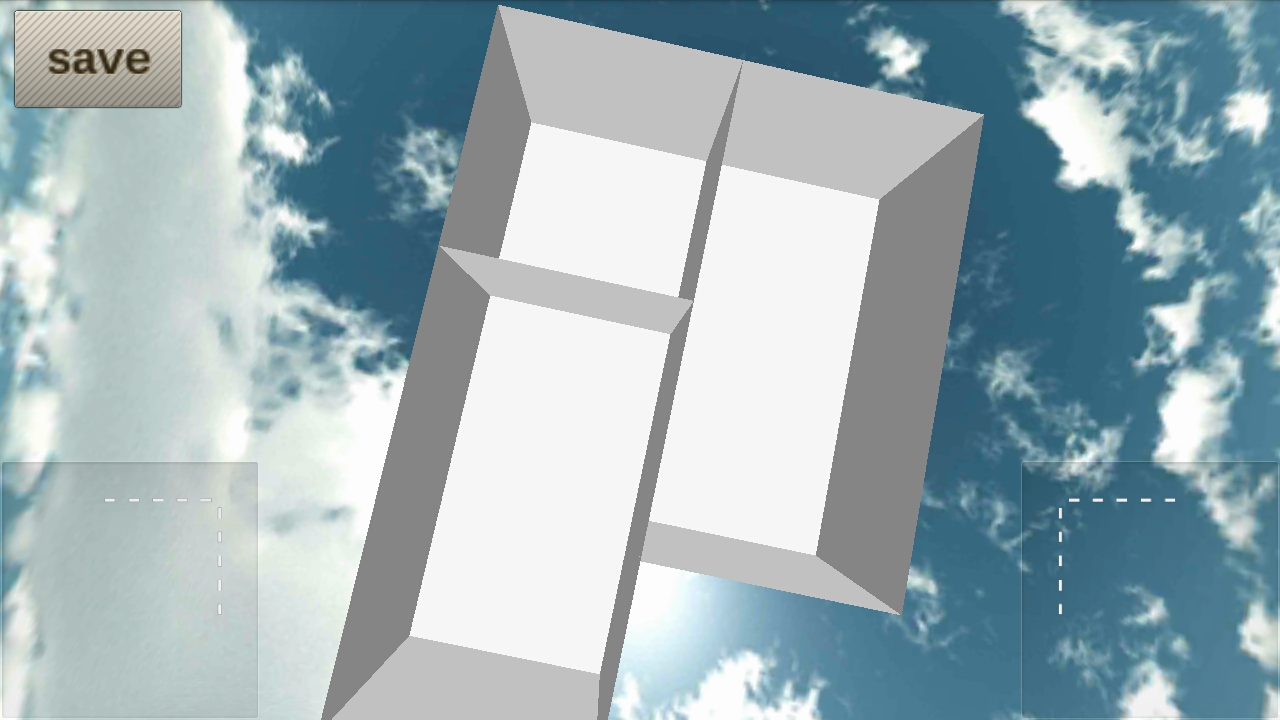


图 4‑8

点击其中一面墙，出现两个按钮，如图4-9（图片已经过反色处理）



图 4‑9

点击“take photo”按钮，调用Android摄像机，拍照后，点击保存，进入图片编辑程序，如图4-10（图片已经过反色处理）

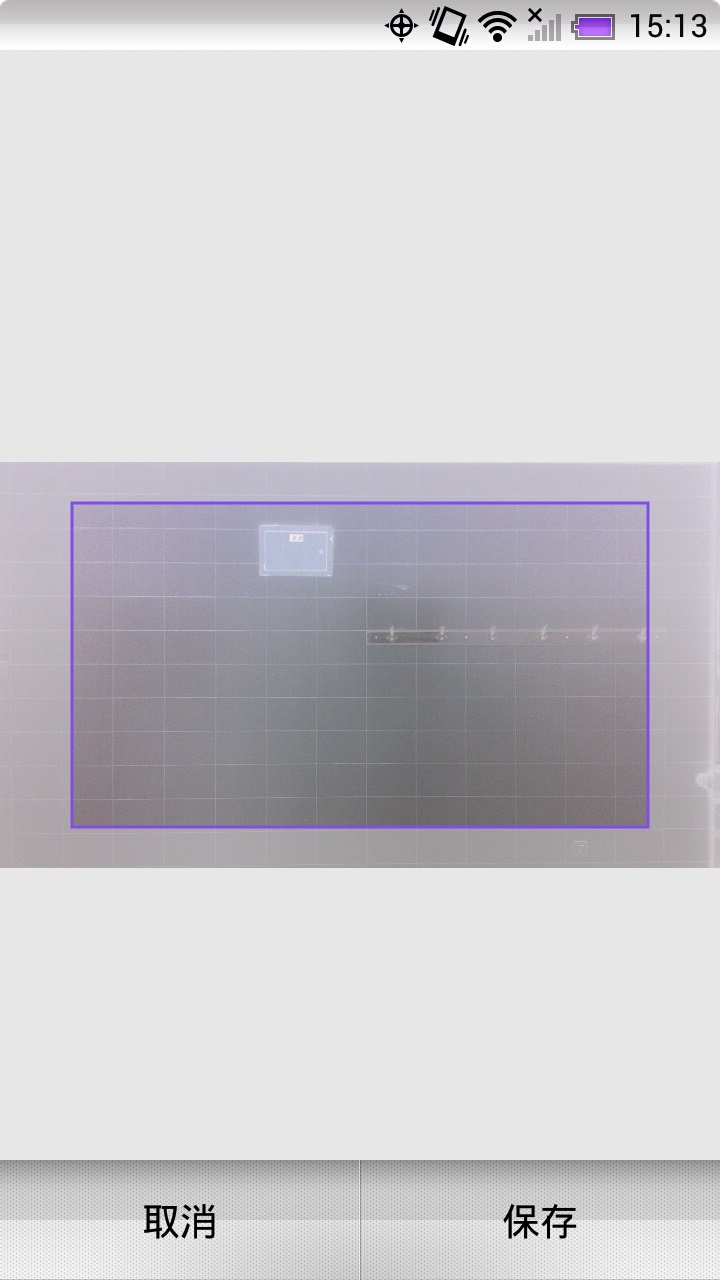


图 4‑10

点击保存后，自动进入漫游场景，如图4-11（图片已经过反色处理），可观察到墙上贴图为拍照图片

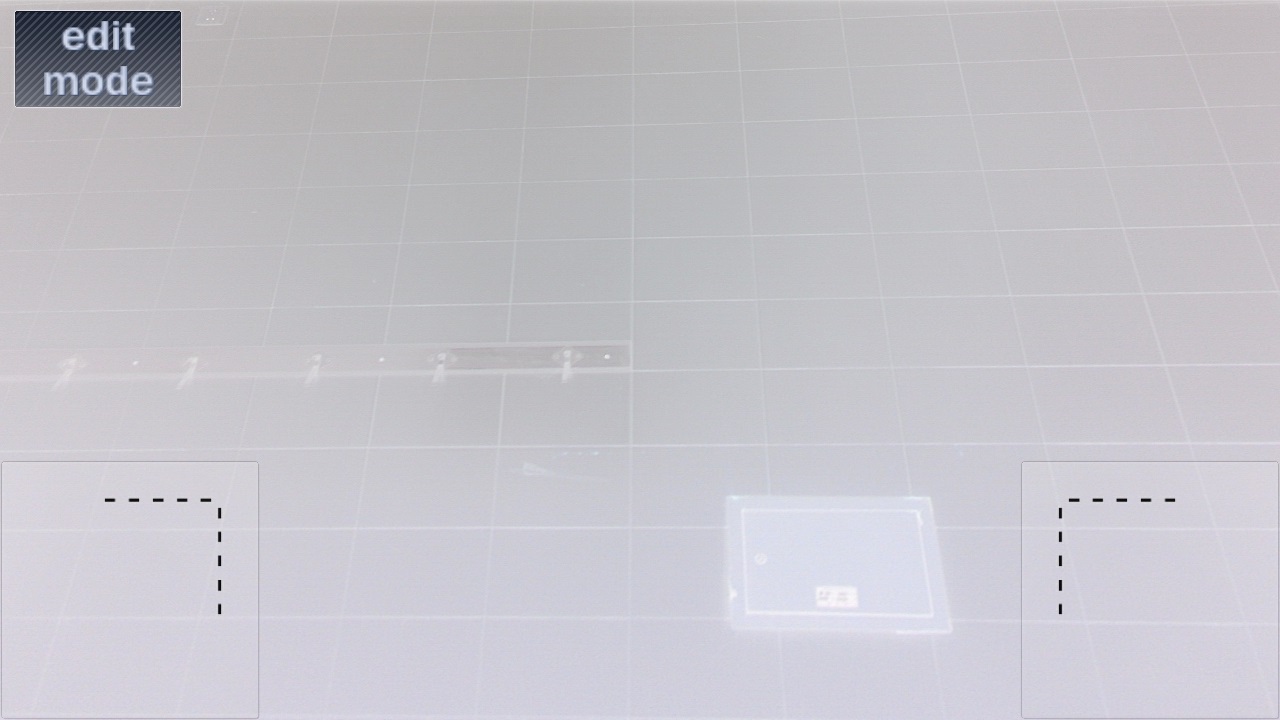


图 4‑11

点击另一面，并点击“pick picture”按钮，调用Android图库，选择一张图片后，进入图片编辑程序，点击保存后自动进入漫游场景，如图4-12（图片已经过反色处理）

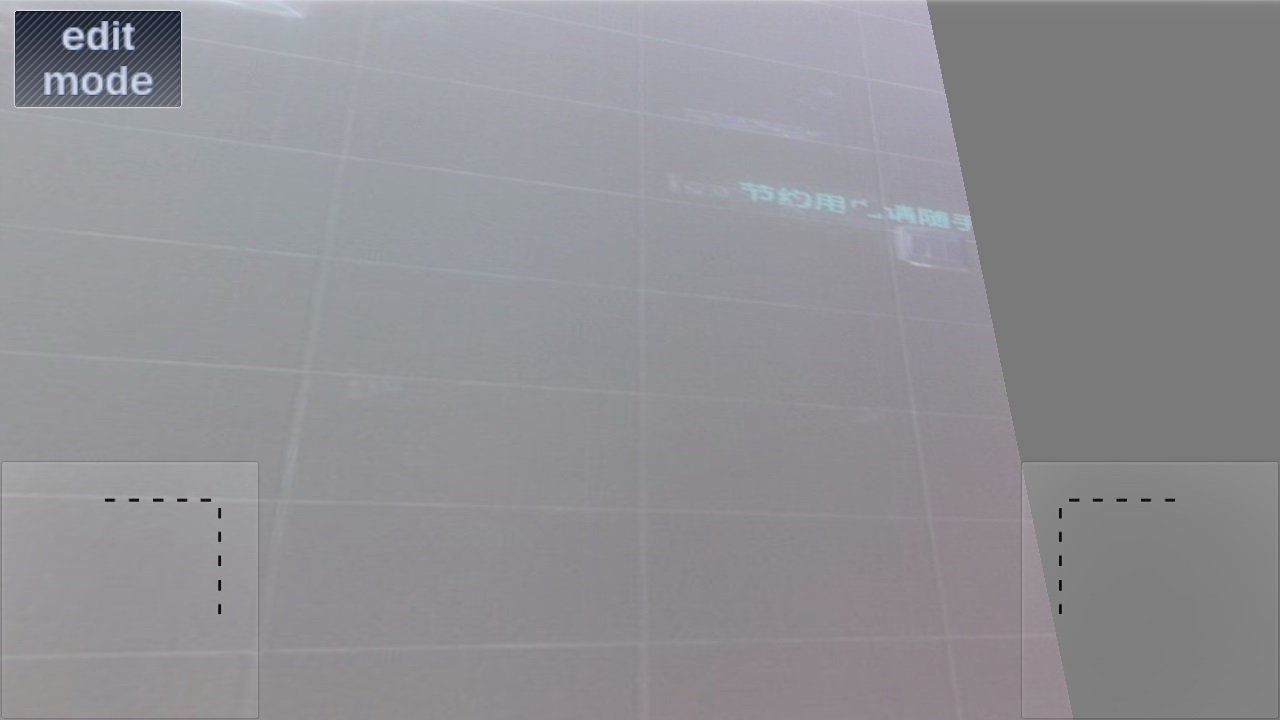


图 4‑12

在编辑场景中，程序运行流畅，在缩放和移动场景或模型时，并无卡顿；在漫游场景中，角色或摄像机移动或旋转并无卡顿，在点击“take photo”按钮或“pick picture”按钮后，有小段延迟，在保存图片时，根据图片大小会有不同长度的延时。

实测在添加了20个模型时，在编辑场景中无卡顿，在漫游场景中，也很流畅；在为模型增加贴图后，在漫游场景的编辑模式中，将摄像机拉远到一定距离，显示所有的模型后，帧数有所降低，但能够运行；而漫游模式中，使用第一人称控制器将不会受到太多性能影响。

## 本章小结

在功能上，完全符合此课题的需求。

在交互方面，能让用户感觉友好，符合此课题的需求。

在2011年上市的中等配置的手机上，能非常流畅的运行此程序，说明程序的效率很高，符合此课题的需求。

# 总结与展望

## 总结

本文通过基于手机平台的室内空间虚拟漫游系统的实现，记述了有关基本理论和程序设计实现过程。在系统开发过程中，采用Unity3D游戏引擎，利用Unity3D和Android的特性进行设计，完成了以下研究工作：

1. 深入研究利用Unity3D游戏引擎制作手机游戏。熟练使用场景、摄像机等游戏中必不可少的组件。熟练使用C#编写游戏脚本，并使用了MVC架构，将模型控制器视图分开控制，使程序扩展性更强。
2. 深入研究利用Unity3D游戏引擎中的NGUI插件制作UI。熟练使用NGUI中的各种控件，并研究了其适配多分辨率的方法。研究了多摄像机之间的协调与配合。
3. 深入研究Android中的活动定义、消息传递和通过参数打开系统活动。并研究Android与Unity3D的消息传递机制。
4. 对系统进行了优化。为使编辑模型时画面流畅运行，并达到对用户友好的目的，设计了基准线位置算法，提高了编辑模型时的性能。

## 展望

本文对基于手机平台的室内空间虚拟漫游系统建立取得了成功，在虚拟漫游方向取得了一些积极的成果，实现了预先确定的目标。由于虚拟漫游技术涉及面广，在本论文的研究过程中，无法触及所有的内容，所以本论文还有着很多可以继续完善和研究的地方：

1. 本系统在设计模型时，没有考虑门的作用，只涉及到封闭的室内环境，离虚拟现实还差的很远。加以更多的时间和精力，将可以使模型不在拘泥于封闭的室内环境，覆盖面更广，使系统的实用性更高。
2. 本系统在拍照贴图时，没有对图像进行过多的处理，所以照片照的不好的话，会严重影响用户体验。学习更多的图像处理技术，来对照片进行处理，使得在漫游时，能更加接近现实。
3. 本系统在拍照或选择图库时，是调用Android系统中的系统相机活动进行拍照或调用系统图库活动来完成的，不支持其他主流移动系统如IOS系统。研究IOS中的拍照API、IOS和Unity3D消息传递机制，在Unity3D中，使用预编译来控制相机调用方式，使系统至少能够跨IOS和Android两大主流移动平台。

致 谢

在一个学期的工作之后，本论文终于接近尾声，即将完成。回顾过去的几个月时间，在自己努力的同时，我的老师、学长和女朋友给予了我很大的帮助，为本论文的顺利完成添加很多力量，在此特向他们表示感谢。

首先我要感谢的是我的毕业设计的指导教师，北京理工大学软件学院黄天羽老师。黄老师做事干练、细致认真、待人友善，处处为学生着想。早在学期开始前的寒假阶段就主动与我联系，帮助我提前制定计划。由于我在本学期还在实习中，空余时间并不多，黄老师特意为我安排时间，让我在这期间，能和黄老师交流，并给与我督促与指导，对于我有疑惑的问题，黄老师也全部给予耐心的解答。黄老师的帮助使我在毕业设计的每一个阶段的工作都安排的井井有条并留有余地，最终成功的按时完成整个论文。

其次我要感谢的是芦尊学长。芦尊学长在我毕业设计时给了我很多关于项目时间安排的指导，使我能够科学的安排时间，避免了最后时刻刷夜赶进度。

最后我要感谢我的女朋友，在我的毕业设计遇到瓶颈时，给我加油打气；在写论文时，给我提供了很大的帮助。

参考文献

1. 陈彦菲. 校园内场景的虚拟漫游技术研究[D]. 大连:大连理工大学, 2008.
2. 吴宇飞. 多用户虚拟北邮校园漫游系统的研究与开发[D]. 北京:北京邮电大学, 2006.
3. 杨更生. 虚拟现实技术在旅游教学中的应用[J]. 新疆教育学院学报, 2010, (2): 111-114
4. 刘科. 中国虚拟博物馆的实现模式及发展前景[D]. 北京:北京工业大学, 2010.
5. 王子健. 基于虚拟现实技术的机器人遥操作系统设计[D]. 河北:河北工业大学, 2010.
6. 许微. 虚拟现实技术的国内外研究现状与发展[J]. 现代商贸工业, 2009, (2): 279-280
7. 初东晖. 浅谈虚拟旅游的关键技术[J]. 科海故事博览·科技探索, 2013, (9): 266-266
8. 陈夏艳. 基于图像建模的虚拟漫游技术研究[D]. 合肥:中国科学技术大学, 2010.
9. 林深华, 范志尚, 蒋建兵, 朱亚超. 基于Android平台Unity3D游戏设计与实现[J]. 企业科技与发展, 2013, (10): 40-42
10. 任国栋, 陈林华, 陶学锋, 方先行. 基于Unity3D的虚拟博物馆信息可视化系统①[J]. 计算机系统应用, 2013, (9): 86-90
11. 孙敏. 虚拟交互设计在中医藏象中的运用研究[J]. 科技风, 2013, (14): 79-88
12. 黄志艳. 3G-Android手机平台应用开发研究[J]. 科技广场, 2013, (10): 54-56
13. 万思. 基于Android系统用于智能手机的运用[J]. 金山, 2012, (3): 1-2
14. 秘锡辰. Android应用软件安全加固技术研究[D]. 北京:北京交通大学, 2013.
15. 黄涛.基于Unity3D的虚拟校园漫游系统的研究和实现[D]. 广西师范大学 2014
16. 夏志海.基于Cocos2d-x的手机游戏坦克大战的设计与实现[D]. 哈尔滨工业大学 2013
17. 曾勇.基于Unity3d的挖掘机模拟训练系统研究[D]. 长安大学 2013
18. 蔡翊.基于Unity3d的金丝峡地质公园虚拟旅游系统的研究与实现[D]. 西安科技大学 2012
19. 高金生.基于3D技术的城市漫游系统[D]. 长安大学 2007
20. 李岩.建筑三维漫游系统的设计与实现[D]. 中南大学 2010