

Stereopsis 软件报告

浙江大学



课程名称: 智能终端程序设计

电子邮件地址: 3130000696@zju.edu.cn

实验日期: 2015 年 9 月 ----- 2015 年 12 月

小组成员: 黄昭阳 3130000696

张函祎 3130102345

粘杰明 3130102183

学院: 计算机科学与技术

1. 引言

1.1 概述

随着现代计算机计算能力的不断增强，计算机图形学与计算机视觉的不断发展，增强现实成为了一个越来越热门的话题，谷歌眼镜、oculus 头盔等典型的增强现实穿戴式设备面世，虽然由于其实用性不够强，产业链不成熟等等诸多原因还未普及大众，但是随着行业技术发展，用于增强现实的穿戴式设备必将如同手机一般普及大众。

当今众多的文物由于容易受到各种环境因素的干扰和破坏而不得不被保护在特制的收藏室内，而即使是在各大博物馆的展馆中，游客也必须与展品保持一定距离，一面发生意外，以及其他种种因素都使得艺术藏品不能为大众所近距离观赏。受到米开朗琪罗著名雕塑 Pietà（圣殇）被加州奥克兰 Fathom 制作室使用 3D 扫描建模技术复制原作品的启发，我们决定设计一款以手机作为载体的增强现实技术实现。使用手机摄像头拍摄环境时，在特定的电子展览平台上能够观察的立体的 3D 展品。这样使得人们能够欣赏近乎原型的艺术品，而且能够通过手机的移动自由调整观察位置，甚至能够十分近距离的观察细节及雕刻的花纹。

由于专利原因，部分代码及库不予以上传，请老师见谅。

1.2 开发环境

Macbook Pro 10.10.5

Xcode 7.1.1

iOS 8.3

1.3 分工

程序：

算法讨论与设计：小组成员

摄像头功能界面：黄昭阳

模型选取界面：张函祎

App Logo 设计：粘杰明

书面文稿：

文档代码原理分析：黄昭阳

程序流程图：张函祎

演示文稿：

1.4 背景

a. 开发系统的名称

Stereopsis

b. 任务提出者：黄昭阳、张函祎

开发者：小组全体

用户：iOS 用户

1.5 使用说明

开启程序后进入模型选择界面，屏幕中央为选择展示的模型，通过滑动来替换选择，点击左上方按钮为进入模型展示界面，点击右上方按钮查看软件声明。

进入模型展示界面后程序会调用手机摄像头来获取环境画面。程序内置两张图片，并计算和筛选好了识别度较强的特征点。通过对摄像头获取的每一帧画面使用 **SIFT** 算法进行特征点识别和匹配，获取程序内置图片在 3D 空间中的姿态和位置。通过摄像头获取的每一帧图片的特征点识别和匹配，以及手机陀螺仪加速度传感器的数据计算，对手机进行摄像机标定，获取摄像机的位置参数。这样我们以图片中心为世界坐标系的原点进行虚拟世界的 3D 建模，还对图片周围场景的物体做了十分粗略的深度还原，以增强虚拟世界位置与现实世界位置匹配的稳定性。

当特征点数量检测到足够多（这个时间其实十分短暂），虚拟与现实的匹配稳定后，我们将以现实环境中的图片作为展览台，导入手机存储的 3D 模型，与现实结合使用 **OpenGL ES** 在图片的正上方 3D 成像。模型（即之前在模型选择界面）将在展览台（即既定图片）正上方轻微浮动，用户可以使用手机近距离观察，甚至可以深入到模型内部观察，实现文物的全方位展示效果。

屏幕左上方设置返回按钮，通过该按钮返回到第一个界面再次进行模型的选取。

2 系统的结构

重要的类与函数表：

模型选择界面：

ViewController 类

开始界面，呈现模型图片供选择；进入立体模型选择或版权页面

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
```

跳转到模型展示界面前的准备，传递选择模型的参数

```
- (void)pickerView:(UIPickerView *)pickerView didSelectRow:(NSInteger)row inComponent:(NSInteger)component
```

Pickerview选中一行的委托

```
- (UIView *)pickerView:(UIPickerView *)pickerView viewForRow:(NSInteger)row forComponent:(NSInteger)component reusingView:(UIView *)view
```

Pickerview委托，提供每一行展示的内容

```
- (CGFloat)pickerView:(UIPickerView *)pickerView
```

```
rowHeightForComponent:(NSInteger)component
```

 Pickerview委托，设置行高

```
- (CGFloat)pickerView:(UIPickerView
*)pickerView
```

```
widthForComponent:(NSInteger)component
```

 Pickerview委托，设置行宽

```
- (NSInteger)numberOfComponentsInPickerView:
(UIPickerView *)pickerView
```

 Pickerview委托，返回列数

```
- (NSInteger)pickerView:(UIPickerView
*)pickerView
```

```
numberOfRowsInComponent:(NSInteger)componen
t
```

 Pickerview委托，返回行数

```
- (UIImage *)scaleImage:(UIImage *)image
withScale:(CGFloat)scaleSize
```

 缩放图片

```
- (UIImage *)clipImage:(UIImage *)image
frame:(CGSize)frame
```

 裁剪图片

```
- (void)setupImageArray
```

 读取pickerview要展示的内容

```
- (void) viewDidLoad
```

初始化视图内容

模型展示界面：

PointCloudApplication 类

与 **pointcloud** 算法包进行交互，根据给定的图片进行图片识别和简易 **SLAM** 深度还原

```
- load_camera_texture;
```

获取摄像机镜头的画面，放到**OpenGL ES**中作为纹理处理

```
- process_camera_frame;
```

根据所给时间戳，与各种数据进行处理，还原出相机外参、内参矩阵，作为之后**OpenGL**绘制虚拟3D图像的相机位置矩阵和投影矩阵

```
- render_point_cloud
```

根据摄像机图片筛选了一系列特征点，将特征点使用纹理渲染上去。

```
- void draw_logo;
```

绘制屏幕上显示的**logo**图片，如返回按钮和**pointcloud**的**logo**

```
- void setup_video_texture;
```

将摄像机图片参数归一化，制成纹理以便算法处理

- **setup_graphics**

图形绘制的预处理

- **render_frame**

接受来自摄像头的纹理数据，开始整体的屏幕界面绘制渲染流程

GLView 类

由于要将摄像头画面与 **OpenGL ES** 绘制的画面进行合并，这里进行基础的调整和环境处理

- **createFrameBuffer:**

初始化并绑定各种缓冲区，以便之后的绘制

- **DestroyFrameBuffer:**

程序退出时释放所获得的缓冲区

- **drawView:**

开始绘制画面

HardwareController 类

程序需要获取摄像头每一帧的画面，获取传感器数据，以及屏幕的点击事件响应，这里用来获取所有的硬件数据，并对全局进行整体的调控。

- **touchesBegan**

- **touchesEnded**

-
- **touchesMoved**
 - **touchesCancelled**

对屏幕点击事件的响应，如调整相机焦距等等，来自**GLView**的委托

- **startGraphics;**
- **stopGraphics;**

图形绘制的流程控制

- **startCamera;**
- **restartCamera;**

相机的流程绘制

- **captureOutput: didOutputSampleBuffer:
fromConnection:;**

摄像机链接的建立

- **(void)drawView:(GLView*)view;**
- **(void)setupView:(GLView*)view;**

绘制到用户的屏幕，来自**GLView**的委托

ObjLibrary 类

由于没有实现用户导入模型功能，这里给开发者提供模型的增删

PaintPanel 类

实现虚拟模型的绘制功能

- `setup_objs`

准备需要绘制的模型

- `render_content`

使用 OpenGL ES 绘制模型

OBJ 类

一个 `OBJ` 对象对应一个模型，负责模型的导入和绘制

- `loadObj`

导入 `Obj` 文件数据，并绑定对应纹理

- `drawObj`

绘制这个模型

Texture 类

一个 `Texture` 对象对应一张纹理图，负责纹理的导入和绘制

- `LoadTextureFileName`

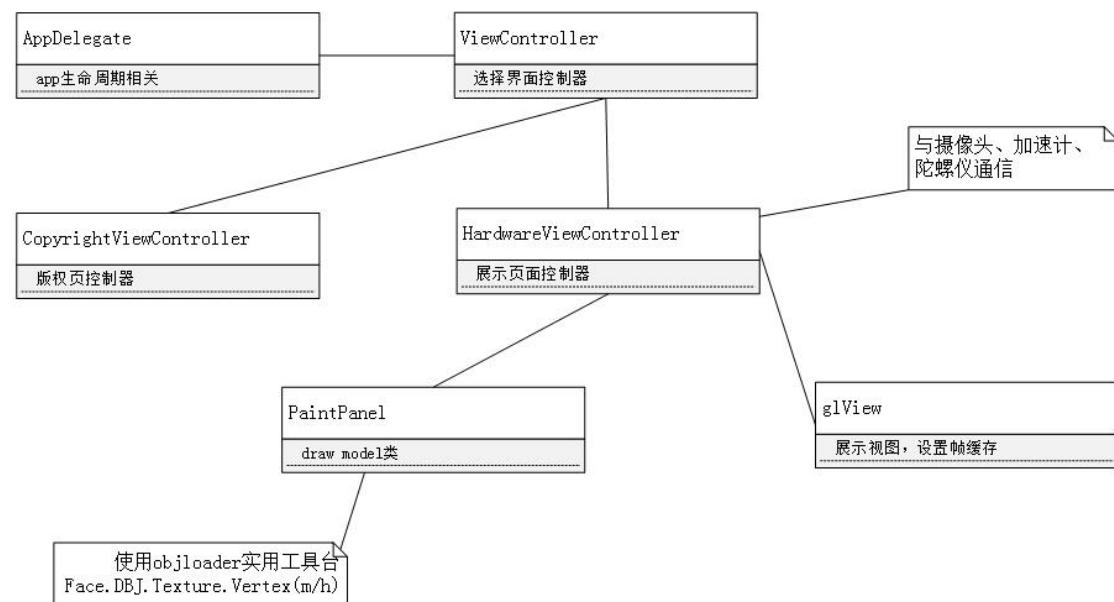
导入纹理文件数据

- `UseTexture`

使用该纹理进行绘制

3. 界面设计

应用的界面逻辑比较简单，主要由选择页、版权页、功能页组成，跳转关系如下：

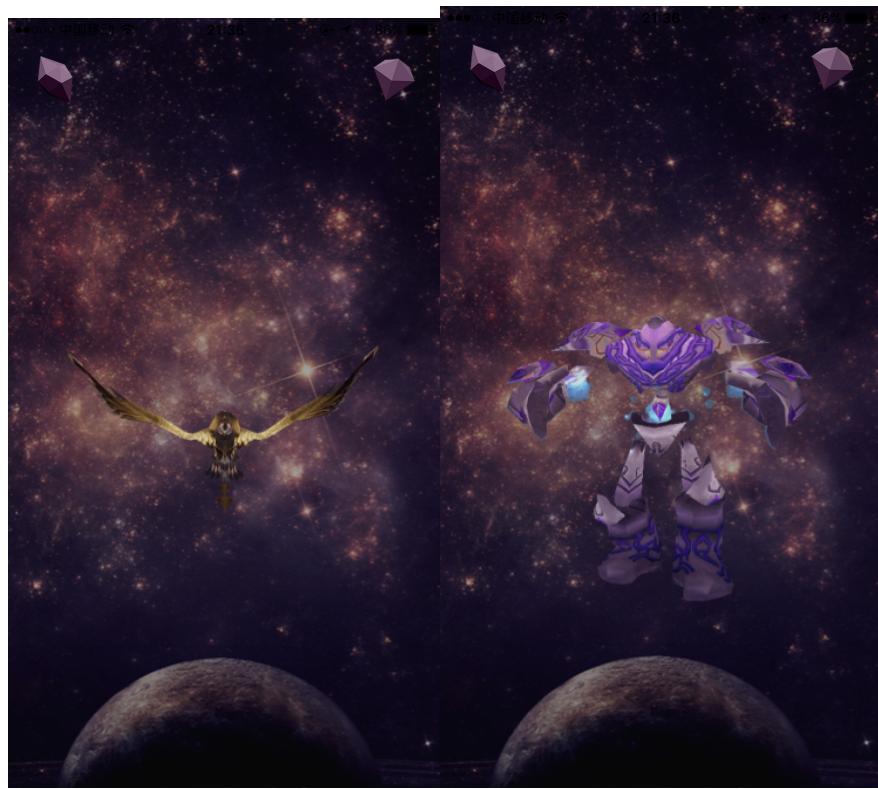


选择页的最上方有两个陨石，点击左侧陨石进入功能页，点击右侧陨石进入版权页。考虑到点击陨石并不会产生不可逆的后果，可供点击的陨石也只有两颗，用户很容易尝试出功能，而且配上说明文字与太空背景不是很协调，我们对此的设计方案是不配按钮文字。

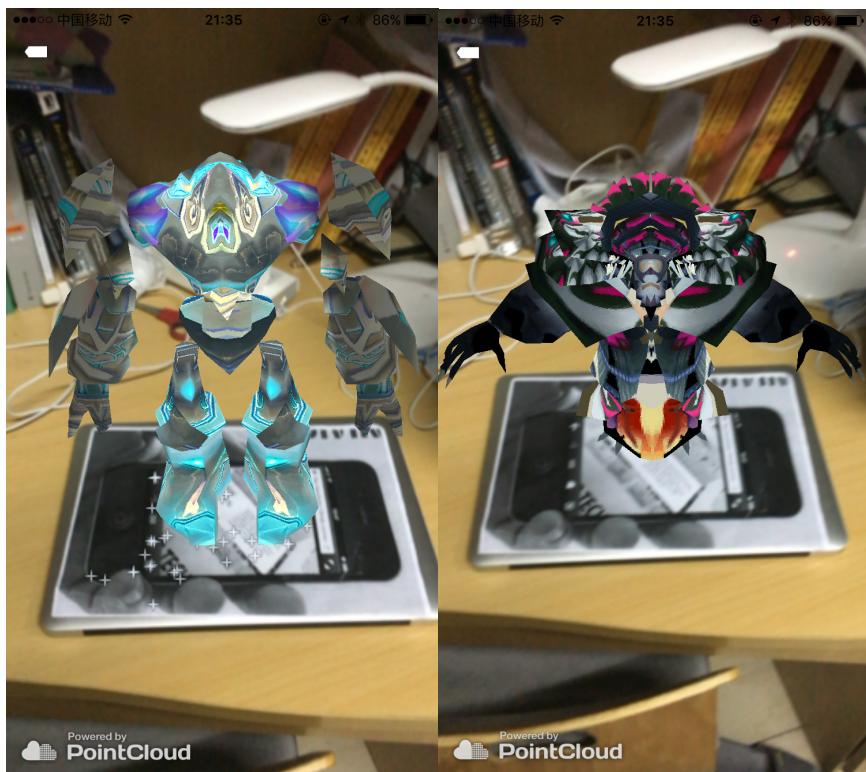
4.程序关键截图

选择页：





功能页：





版权页：



5. 实验感想

实验从立项一开始的目的就十分明确，利用现有的计算机视觉和图形学知识实现博物馆珍贵文物的展示，但是由于现在手头没有比较好的博物馆模型，于是使用在网上找到的魔兽人物原型替代展示，虽然展示的不是文物，但是实际上只需要将对应模型替换掉即可。

由于此次实验涉及到的计算机视觉和计算机图形学算法知识较多而且相对较复杂，曾多次向鲍虎军老师请教相关算法，并从老师处获得一部分算法库资源，这里十分感谢鲍虎军老师不厌其烦的详细指导。

在这次实验的过程中，我们的整体软件架构能力、软件设计能力、

团队协作能力都得到了很大幅度的提高，在这个过程中，我们对 iOS 系统 App 开发流程有了进一步的认识，并且成功将 OpenGL ES 与其他的功能模块进行结合，并且对于计算机视觉与计算机图形学有了深入的了解，收获十分之大。

观看我们的源代码可以发现有一部分接口我们传递了但是还并没有投入使用，因为我们在实现这个软件的过程中有了更深一步的对这个软件进行更新和改造的想法，既然我们能够成功以图片识别为基础相对稳定的将虚拟世界坐标与现实世界坐标相结合，那么我们完全可以通过以图片为平台（在此次实验中即模型的展览台），增加更多的用户互动元素，比如相机位置的改变、屏幕的点击等等，来构建一个增强现实的小游戏。比如一个小型的第一人称射击游戏等等，我们不会因为这个实验的完成而停下我们软件开发的脚步，而是希望能够成功的开发出一款有一定可玩度的增强现实小游戏。