计算机图形学

系统说明文档

Date：2017-06-22

成员 1　315010**** ******

成员 2　315010**** ******

成员 3　315010**** ******

一、第三方库

实验中我们组使用了第三方库 glew 和 glm。glm.h 和 glm.cpp 是经组员修改的 glm 库。

二、实验内容与特色

我们程序实现了以下的基本要求：

1. 具有基本体素（立方体、球、圆柱、圆锥、多面棱柱、多面棱台）的建模表达能力；
2. 具有基本三维网格导入导出功能（建议 OBJ 格式）；
3. 具有基本材质、纹理的显示和编辑能力；
4. 具有基本几何变换功能（旋转、平移、缩放等）；
5. 基本光照明模型要求，并实现基本的光源编辑（如调整光源的位置，光强等参数）；
6. 能对建模后场景进行漫游如 Zoom In/Out， Pan, Orbit, Zoom To Fit 等观察功能；
7. 能够提供动画播放功能（多帧数据连续绘制），能够提供屏幕截取/保存功能。

同时我们选做了以下额外要求：

1. 具有 NURBS 曲面建模能力；
2. 漫游时可实时碰撞检测；
3. 构建了基于此引擎的完整三维游戏，具有可玩性；
4. 具有一定的对象表达能力，能够表达门、窗、墙等。

在此基础之上，我们的项目具有以下特点：

1. 丰富的场景元素和精致的物件模型。整个场景中共有 77 组不同的模型，带有 77 个不同的纹理，其中包含 54 张纹理贴图。场景中出现的所有模型和贴图均为本组所有成员在两周内自己搭建与绘制的。
2. 全屏多采样抗锯齿的实现。这个功能的开启需要显卡设置的配合，在开启状态下可以避免绝大多数锯齿状边缘的出现，提高了场景美观度。
3. 视区对窗口大小的自适应。无论是截图功能还是第一人称漫游模式，观察与操作不会受窗口大小改变而导致意想不到的结果。
4. 实现了第一人称视角模拟漫游的功能。在场景中玩家可以进行旋转视角、上下台阶、自由移动等操作。
5. 透明材质的绘制。通过启用 GL_BLEND 并调整绘制顺序，场景中实现了大量透明物体，如玻璃、窗帘等，的绘制与表达，增强了场景的真实感。
6. 背景音乐与视频播放功能。默认为开启并且循环播放，可使用按键控制。
7. 简单动画的实现。门与窗帘可以做出简单的旋转与缩放动画，利用异步延时执行函数的功能来实现。
8. 引入拾取机制，使第一人称模式具有更多的可玩性。
9. 系统信息的直观显示。场景内绘制了多个关键物体（灯光、摄像机目标等）的定位器，在屏幕左上方显示实时帧速率，在屏幕右上方显示摄像机相关参数，在屏幕左下方显示系统消息，与玩家保持良好的交互。
10. 项目开发全程使用 Git 管理源码并统一了代码风格，降低了同伴之间合作与代码合并的难度。使用大量注释，确保了代码的可读性。使用分工进度表，确保了项目各环节的有效推进。

三、程序架构

整个程序分为以下几个模块

1. head.h - 包含所有宏和类的定义、enum 定义、全局变量和函数的声明
2. glum.cpp - 经组员修改的 glm 库

3. Main.cpp - 控制台应用程序入口，调用初始化函数，并进入 glut 主循环
4. Draw.cpp - 绘制模块，包含模型，NURBS 曲面等绘制
5. Light.cpp - 光照控制模块
6. Texture.cpp - 纹理绘制模块
7. System.cpp - 系统函数模块，包括初始化，键鼠事件等基本功能函数的实现
8. Util.cpp - 实用函数模块，包括视角操作和特殊功能的相关函数的实现

四、模块内容

1. head.h

包含必要的头文件；定义宏与 enum；声明变量；声明以下各个模块的主要功能函数。

2. glum.cpp

（1）GLMmodel* glmReadOBJ(char* filename)

这个函数用于从 obj 文件中读取模型，传入参数为 obj 文件的路径。obj 文件中包含主要四种类型的前缀：

① v - 表示本行指定一个顶点。此前缀后跟着 3 个单精度浮点数，分别表示该定点的 X、Y、Z 坐标值

② vt - 表示本行指定一个纹理坐标。此前缀后跟着两个单精度浮点数。分别表示此纹理坐标的 U、V 值

③ vn - 表示本行指定一个法线向量。此前缀后跟着 3 个单精度浮点数，分别表示该法向量的 X、Y、Z 坐标值

④ f - 表示本行指定一个表面(Face)。一个表面实际上就是一个三角形图元。

不同的前缀表明了存储数据的类型，文件开始先声明了所用到的材质文件（.mtl），接着每一行以前缀开头声明此行数据的类型。

因此读入的时候也分种类放入 model 类里，作为 model 参数的一部分。

（2）GLvoid glmDraw(GLMmodel* model, GLuint mode)

这个函数用于绘制已经读入的模型。两个参数分别是 model，绘制模式。向 GL 传入三角面的顶点，并调用 glMaterialfv()读取材质信息。

（3）GLvoid glmDrawTransparency(GLMmodel* model, GLuint mode, GLfloat transparency)

这是经我们修改的绘制函数，它的功能是在 glmDraw 的基础上将传入参数 transparency 作为不透明度，应用于绘制的物体。

3. Main.cpp

（1）int main(int argc, char *argv[])

控制台应用程序入口，调用初始化函数，并进入 glut 主循环。

4. Draw.cpp

（1）void initObj()

该函数用于初始化模型，我们组的模型并不是一个整体，因此需要使用一个数组来存放所有的模型，然后从文件中导入 obj 文件，批量调用 glmReadOBJ（）函数来导入。

（2）void initNurbsSurface()

该函数用于初始化 NURBS 曲面。

（3）void drawScene()

该函数批量调用 drawModel 函数，绘制主场景。

（4）void drawModel(int modelnum, GLfloat x, GLfloat y, GLfloat z,

int texturenum = -1, int mode = GL_MODULATE, GLfloat rotate = 0.0f);

该函数是绘制模型函数，集中了定位、贴图、旋转操作。在 drawScene()中调用。处理模型定位和旋转后，glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, mode)指定纹理光照混合模式，调用 glBindTexture()将纹理绑定到模型上，最后调用 glmDraw()对输入模型进行绘制。

（5）void drawVideo()

该函数以改变纹理的方式实现低帧率视频的循环播放。

（6）void drawNurbs()

调用 drawNurbsSurface()绘制 NURBS 曲面。

（7）void drawCrosshair()

绘制第一人称漫游模式下视点中心的十字，利用 GL 基本图元的绘制，指定线宽之后绘制十字。

（9）void drawLocator(GLfloat* center, GLfloat radius)

该函数用于绘制一个球状的定位器。实现方法是以指定点为中心，绘制一定半径大小的线框球体。

（10）GLint genDisplayList(int type)

采用显示列表的方式绘制，通过 type 判断所需要的绘制类型决定调用 drawScene()还是 drawNurbs()，返回生成的显示列表标识符。

5. Light.cpp

（1）void initLight()

在场景中创建三个灯光并启用。随后进行设置：

1. 镜面反射计算时以视点为准。
2. 镜面反射在纹理应用之后进行。
3. 纹理与光照的混合方式为 GL_MODULATE。

（2）三个灯光中的 0 号灯为可调整灯光，其位置、颜色分量、光强（常数衰减系数）均可被控制。1 号灯和 2 号灯均为固定光源，给场景提供必要的照明。其实现分布在随后的 initLight0、initLight1 和 initLight2 这三个过程函数中，不再赘述。

6. Texture.cpp

（1）void initTexture()

这个函数调用 readMTL 与 loadObjectTextures 这两个自定义函数，目的是从 mtl 文件中读取贴图路径并在内存中为其分配纹理标识符。

（2）void readMTL(char * fileName)

传入参数为 mtl 文件的路径，功能是将文件中的贴图路径存入数组 texturefilename 并使用变量 textureobjectcount 进行计数。

（3）void loadObjectTextures()

与 initVideo 函数类似，它按照 textureobjectcount 的计数，为数组 textureobjects 申请足够多的纹理标识符，随后调用 loadTexture 将它们加载为纹理贴图。

（4）void initVideo()

这个函数申请 32 个纹理标识符用于存储视频帧，随后生成递增的文件名并调用 loadTexture 将它们加载为纹理贴图。

（5）unsigned char *loadBitmapFile(char *filename, BITMAPINFOHEADER *bitmapInfoHeader)

这个函数将磁盘上的 BMP 文件读入到内存中，被 loadTexture 调用。

（6）void loadTexture(int i, char* filename, bool type)

这个函数判断传入的贴图类型是属于普通贴图还是视频帧，随后调用 loadBitmapFile 读取文件并将它绑定到 textureobjects 中的纹理标识符。

7. System.cpp

（1）void processMouseClick(int button, int state, int x, int y)

这是处理鼠标点击事件的函数，在左键点击并按下的时候，会打开或关闭抗锯齿模式，并在窗口中显示 switch on/off。右键单击时激活拾取检测并修改相关状态变量。在 main 函数中被 glutMouseFunc(processMouseClick)注册。

（2）void processMouseMove(int x, int y)

这是处理鼠标移动事件的函数，当模式切换为第一人称控制的时候，就会激活这个函数。我们用 fpsmode 来定义该状态，0 为关闭状态，1 为开启状态。它将鼠标的上下左右移动转换为摄像机目标在以摄像机为球心的球面上的运动，实现视角的变化。同时，函数维护鼠标坐标不允许其超出窗口范围。在 main 函数中被 glutPassiveMotionFunc(processMouseMove)注册。

（3）void processNormalKey(unsigned char k, int x, int y)

该函数负责处理键盘事件。区分按键值并调用相关函数。它在 main 函数中被 glutKeyboardFunc(processNormalKey)注册。按键功能将在后面的操作说明中详细介绍。

（4）void showSysStatus()

该函数在屏幕上显示系统状态，FPS,相机位置，目标位置，摄像机的球面坐标。

利用 glutGet(GLUT_ELAPSED_TIME)获取当前时间，并与上一次保存的时间进行比较，目的是每隔约 1000ms 计算间隔时间内绘制的帧数。在 redraw()重绘制函数中被调用。

（5）void init()

该函数调用各基础功能的初始化函数：

```
initMap();              // 初始化地图
initTexture();          // 初始化纹理
initVideo();            // 初始化视频
initObj();              // 初始化模型
initNurbsSurface();     // 初始化NURBS曲面
initLight();            // 初始化灯光
```

随后生成三个显示列表用于分别绘制主场景、NURBS曲面和具有动画的物件。

（6）void redraw()

重绘制函数。每次被调用时会清除颜色缓存并重新绘制场景中的所有元素。我们使用了Display List即显示列表来加速渲染。

8. Util.cpp

（1）void callList(GLint listcode)和 void updateList(GLint listcode, int type)

调用显示列表，更新显示列表。

（2）void initMap()

初始化真值表，从文件里读入一个二维的真值表，用于碰撞检测。

（3）void cameraMakeZero(GLfloat* camera, GLfloat* target, GLfloat* polar)

使相机的位置归零。

（4）void updateCamera(GLfloat* camera, GLfloat* target, GLfloat* polar)

更新相机位置。

（5）void updatePolar(GLfloat* camera, GLfloat* target, GLfloat* polar)

更新相机的球面坐标。

（6）void updateTarget(GLfloat* camera, GLfloat* target, GLfloat* polar)

更新目标的坐标。

（7）void saveCamera(GLfloat* camera, GLfloat* target, GLfloat* polar)

void loadCamera (GLfloat* camera, GLfloat* target, GLfloat* polar)

保存和读取摄像机位置相关坐标，用 cameramatrix[3][3]矩阵保存摄像机位置、摄像机目标位置及其球面坐标参数。

（8）bool detectCollision(GLfloat* camera)

通过真值表来进行碰撞检测，我们模型的碰撞检测仅限于二维的，一整个平面制作成真值表，有物体占用的地方为 false，没有物体占用的地方为 true，根据当前的相机位置，通过计算获得需要判断的点的坐标，然后在真值表上寻找这个坐标里的值是 0 还是 1，以此来确定是否碰撞。若碰撞则会调用 loadCamera 重置摄像机和目标位置。

（9）void updateWindowcenter(int* window, int* windowcenter)

更新当前窗口中心在屏幕的绝对坐标。

（10）bool screenshot(int width, int height)

截图模块。从当前颜色缓存中读取颜色值，开辟一块与视区大小相同的图像内存空间并写入颜色值，随 BMP 格式必要信息写入物理文件。

（11）void processMusic(int value)

背景音乐控制函数。根据 value 值控制音乐的播放与暂停。

（12）void animationTimer(int value)

这是一个计时器，通过这个函数用于绘制动画。我们的程序里有两个动画，分别是门的开关和窗帘的开关。以门为例子，首先判断门的状态是开或者关，决定门的旋转角度的正负，也就是接下来的动画是关或者开门，然后通过 updateList(&listcode_door, DOOR);更新显示列表，完成下一个状态的绘制，接着判断门的状态是否达到终止（到达 0°或者 90°），若是没有到达终止状态，则调用 glutTimerFunc(33, timer, DOOROPENING)，这个函数能够再次调用 timer 函数，达到连续绘制，形成动画效果。窗帘动画同理。

（13）void processPick(GLint* window)

这是通过拾取机制进行绘制，通过调用startPicking(window)在窗体内开启拾取模式，然后调用绘制函数绘制场景，最后调用stopPicking()停止拾取。采用拾取机制绘制能够让鼠标在场景内识别一定范围内的物体，就可以实现鼠标点击选取物体的功能。该过程参考了课本相关内容。

（14）void startPicking(GLint * window)

这是开启拾取模式的函数：

```
// 设定用于点击后返回数据的数组
glSelectBuffer(BUFSIZE, selectBuf);
// 开启拾取模式
glRenderMode(GL_SELECT);
// 设定拾取的选框。
gluPickMatrix(window[X] / 2.0, viewport[3] - window[Y] / 2.0, 5, 5, viewport);
// 初始化名字堆栈
glInitNames();
```

（13）void stopPicking()

这是关闭拾取模式的函数，hits = glRenderMode(GL_RENDER);将其切换回普通绘制模

式，然后保存hits（点击的返回的数值），最后调用processHits(hits, selectBuf)到数组中寻找被点击选取的物体。

（14）void processHits(GLint hits, GLuint buffer[])

这是用于确定点击后选取的物体的函数。根据返回的深度信息寻找具有最小深度的物体作为用户点击的对象，然后确定对象的名字，根据物体的名字输出拾取的物体。

五、游戏内容

这是一个密室逃脱类型的 FPS 寻宝游戏。玩家扮演一个困在屋子里的勇士，需要马上离开这间上锁的屋子踏上拯救公主的旅程，在场景里隐藏了一把钥匙，玩家需要通过第一人称视角漫游场景，寻找此物件来拯救公主。当玩家找到钥匙的时候，需要瞄准并用鼠标右键点击它，然后房门就会自动打开，玩家就能获得自由以及头盔。如果不尽快走出屋子的话，玩家将获得绿色的大草原。

六、操作说明

（1）视角变换操作

默认状态下为摄像机围绕摄像机目标的运动，W、S 为升高/降低摄像机，A、D 为顺时针/逆时针环绕，Q、E 为接近/远离目标。

按 Z 键切换到摄像机目标的控制模式，使用 A、D 控制摄像机目标在世界坐标系的 X 坐标，W、S 控制 Y 坐标，Q、E 为 Z 坐标。

按 C 键切换到第一人称漫游模式，使用 WSAD 控制前后左右移动，遇到障碍物会触发碰撞而受阻。需要注意的是如果按下 C 时鼠标不在窗口区域，请将鼠标移至窗口内，方能进行控制。

（2）灯光操作

按 R/G/B 可以分别循环增加灯光 0 的红绿蓝分量，按+/-以增加/降低灯光 0 的光强，按↑/↓来调整灯光位置（同时观察场景中心表示灯光的定位器）。

（3）曲面物体操作

按 N 键显示/隐藏 NURBS 物体，它的位置在屋顶上方。非第一人称漫游模式时需要将摄像机和目标分别向上移动。

（4）截图操作

按 X 键截取当前窗口内容，图片保存在工程目录 images 文件夹下。

（5）背景音乐

默认为在主程序运行 2 秒后开始播放，可按 V 键中止或继续播放。

（6）动画

窗帘和门的动画本应在达成游戏目标后自动播放，但是为了观察效果，可以按 O（门）和 P（窗帘）来控制对应动画的播放，再按一次可回退。但这不会允许玩家脱离游戏条件控制而走出房间，仅为观察所用。为了不影响游戏功能请保持门和窗帘在闭合状态。

（7）鼠标操作

在任意状态下在窗口中单击鼠标左键可以启用/禁用多采样抗锯齿。需要注意的是这个功能受显卡设置（如：NVIDIA 控制面板-管理 3D 设置-平滑处理）限制。

在第一人称模式下单击鼠标右键，系统将检测当前准心所瞄准的物体并在后台记录。

（8）缩放以适应

在第一人称模式下对准特定物体并单击鼠标右键后，按下 F 键，摄像机将移动到最适合观察该物体的位置，可以将混乱的视角快速调整到适合观察的视角，此为系统中的缩放以适应功能。目前开放此功能的物体有电视机、地毯和壁画。

（9）文件导出

按下“.”键，系统会输出 out.obj 到工程目录的 output 文件夹下。

七、成员分工

场景部分（道具设定、场景建模、纹理贴图）由卢婷芳负责，代码部分（系统功能的实现）由杨皓天负责，协调和进度监督由张韫哲负责。各负责人拥有负责范围内工作内容的分配权力，同时对自己负责内容的最终效果负责任。

八、程序运行截图

场景总览 1　室内

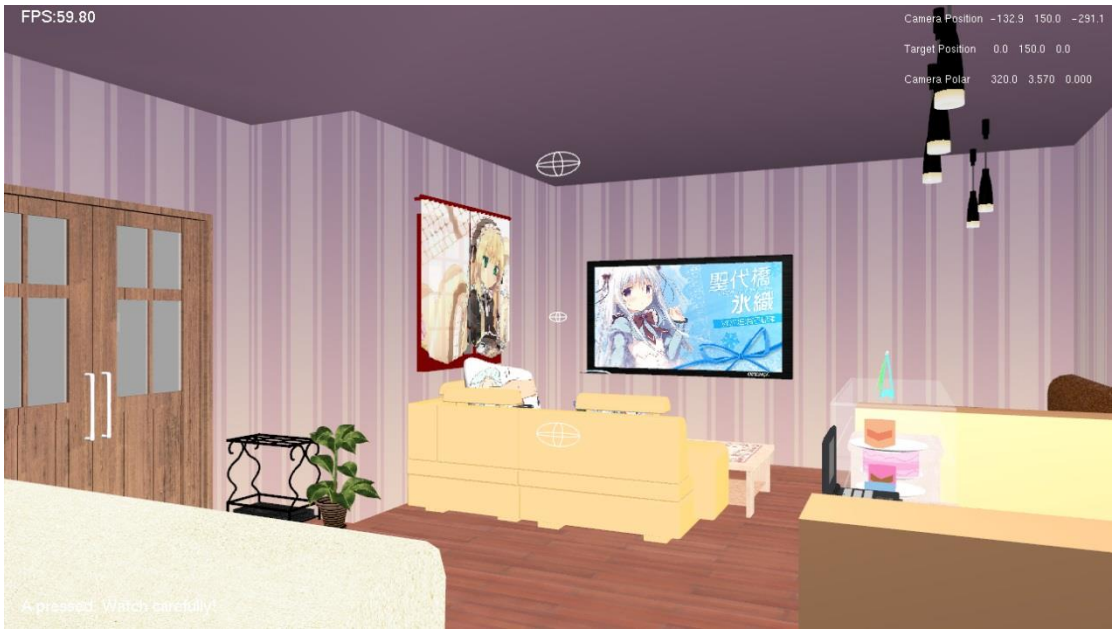
场景总览 2　鸟瞰 1

场景总览 3　鸟瞰 2

视角 1　前台

视角 2　休息区

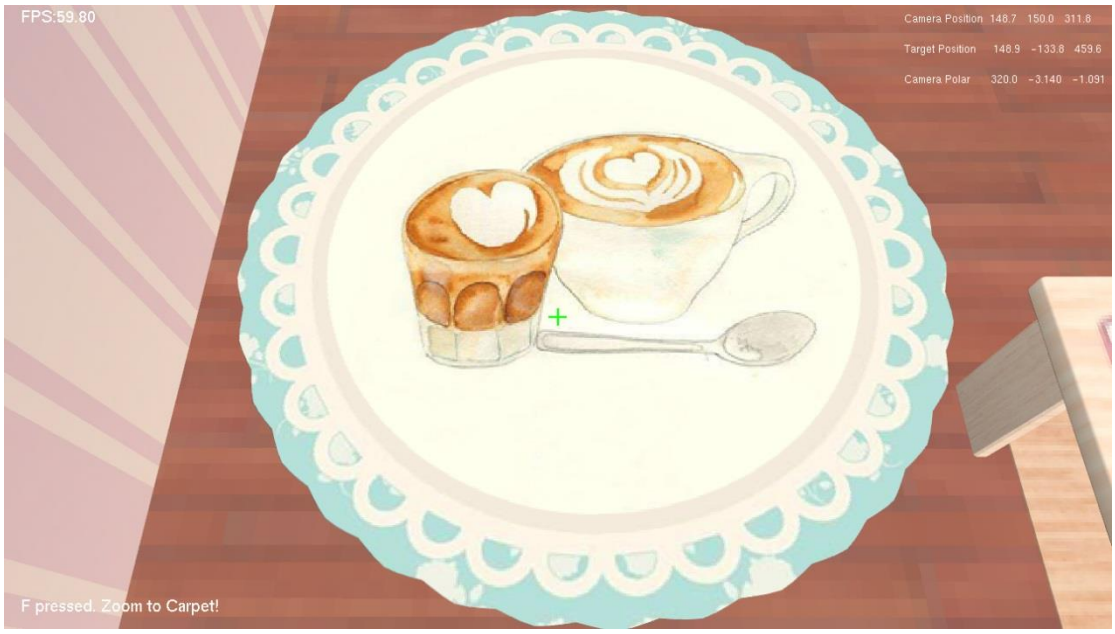
视角 3　门外小景



视角 4　室内布置

视角 5　沙发

视角 6　用餐区

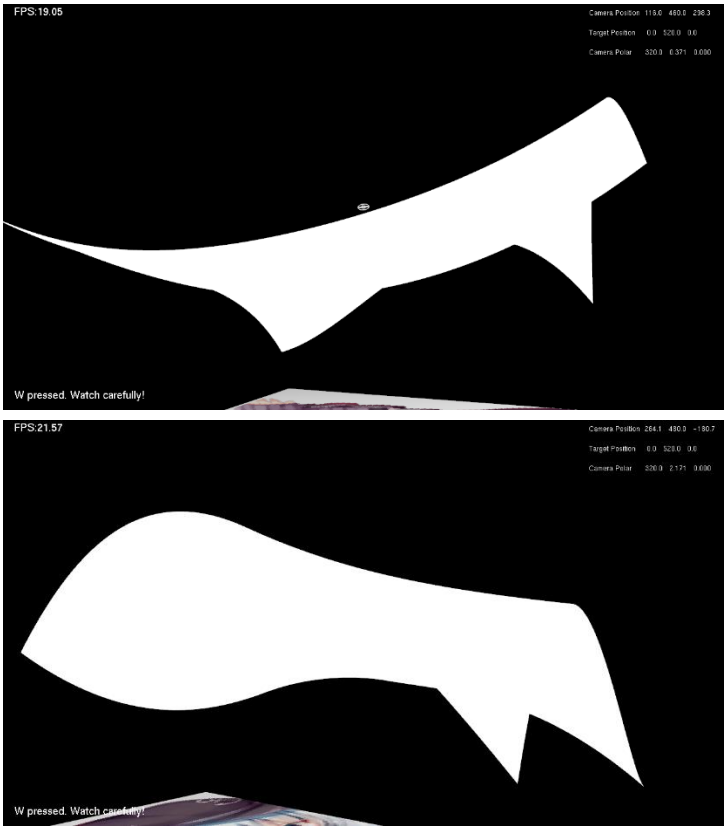


特写 1　地毯

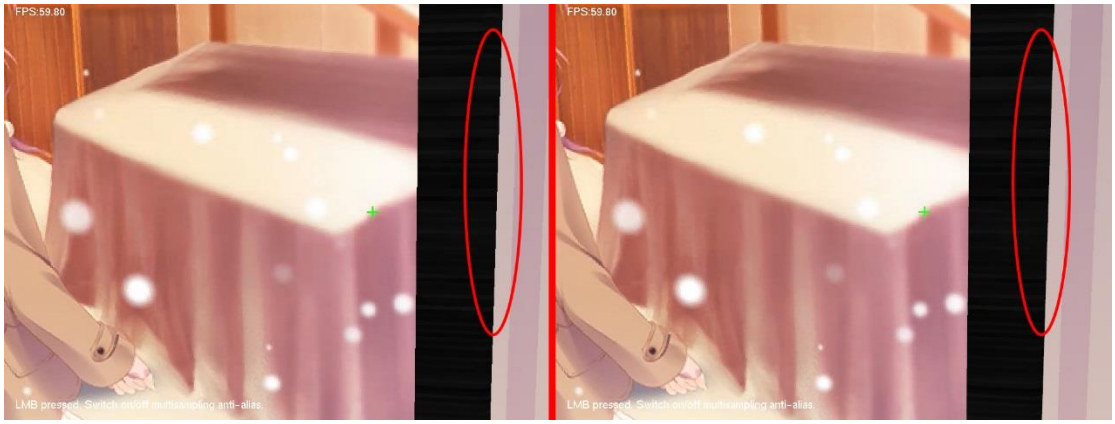
特写 2　电视机

特写 3　台历

特写 4　书架



NURBS 曲面



抗锯齿对比

发现关键道具

达成游戏胜利条件