

GG1003

Windows编程

胡学萱

胡学萱

实验6-4 精灵管理类的实现及使用

余庆祥

2220631136

BDP

2022/11/30

**内页写作格式**

**一、实验名称：**要用最简练的语言反映实验的内容。

**二、实验日期：**写明做实验的具体年、月、日及组別。

**三、实验目的：**使用简洁的文字或关键字来敘述，是以怎样的目的作此实验的。

**四、实验环境：**实验的操作系统和软件等。

**五、实验的步骤和方法：**这是实验报告极其重要的内容。这部分要写明经过哪几个步骤。

**六、数据记录和计算：**指从实验中测到的数据以及计算结果。

**七、实验结果或结论：**即根据实验过程中所见到的现象和测得的数据，得出结论。

**八、备注或说明**：可写上实验成功或失败的原因，实验后的心得体会、建议等。

|  |  |
| --- | --- |
| **1、实验名称** | 实验6-4 精灵管理类的实现及使用 |
| **2、实验日期** | **2022/11/30** |
| **3、实验目的** | |
| 1.理解CSpriteManager对所有精灵更新、渲染的实现，对层叠显示的精灵前后遮挡关系的管理实现。  2.会使用CSpriteManager实现对多个CSprite对象、CAnimationSprite对象的管理。 | |
| **4、实验环境** | |
| 1、vs2019 | |
| **5、实验的步骤和方法** | |
| 1、在上次代码的基础上，在窗口中创建一个Monster、一个Walker，他们是由精灵管理类管理的。也就是说，两个精灵的创建、更新渲染、清理工作是由精灵管理类负责，而不是main。注意资源管理类和精灵管理类的关系。  2、在题1基础上，修改代码，将多个Monster、Walker动态创建。例如窗口中一开始没有任何精灵，按下M键则多创建一个精灵对象Monster（最多10个），按下W键则多创建一个动画精灵对象Walker（最多10个）。各个精灵位置随机，鼠标点击可选中一个Monster，之后它改变为最上层，方向键可控制其移动，若walker被撞到则消失。参考效果如ManagerDemo。 思考题 1.假设现在有另外一种精灵CMySprite，它是CSprite的子类，目前CSpriteManager可以对这种类进行管理吗？为什么？如何才能一起管理它？  2.使用数组实现管理，在插入、删除等操作时都很不方便，试一试将CSpriteManager改写，用链表代替数组来管理精灵。 | |
| **6、数据记录和计算** | |
| **（代码编写思路及关键代码）**  第一题代码：  // 准备环境、后备缓冲  RECT rt;  GetClientRect(hwnd, &rt);  HDC hdc = GetDC(hwnd);  HDC hdcMem = CreateCompatibleDC(hdc);  HBITMAP hBmMem = CreateCompatibleBitmap(hdc, rt.right - rt.left, rt.bottom - rt.top);  SelectObject(hdcMem, hBmMem);  // 创建资源管理类对象，加载需要的所有资源  pRes = new CResourceManager(hwnd);  pRes->LoadPictureFromFile(L"Monster", L"monster1.bmp");  pRes->LoadPictureFromFile(L"Walker", L"walker1.bmp");  // 创建精灵管理类对象，创建Monster，设置初始状态  pSpriteManager = new CSpriteManager(pRes);  pMonster = pSpriteManager->CreateSprite(L"Monster", 10, 10);  pMonster->SetTransparent(true);  pWalkers[0] = pSpriteManager->CreateAnimationSprite(L"Walker", 4, 5, 50, 200, 32, 54);  pWalkers[0]->SetTransparent(true);  pWalkers[0]->Play();  // 取得系统时间  DWORD t0 = GetTickCount(), t1;  While里：  // 更新客户区尺寸  GetClientRect(hwnd, &rt);  // 取得系统时间  t1 = GetTickCount64();  // 设置图形模式，设置允许世界转换的高级图形模式  SetGraphicsMode(hdcMem, GM\_ADVANCED);  // 清除背景  FillRect(hdcMem, &rt, (HBRUSH)GetStockObject(WHITE\_BRUSH));  // 去掉多余的像素  SetStretchBltMode(hdcMem, COLORONCOLOR);  // 精灵管理类更新、渲染  pSpriteManager->Update((t1 - t0) / 1000.0f);  pSpriteManager->Render(hdcMem);  t0 = t1;  // 恢复图形模式  SetGraphicsMode(hdcMem, GM\_COMPATIBLE);  // 输出精灵  BitBlt(hdc, 0, 0, rt.right, rt.bottom, hdcMem, 0, 0, SRCCOPY);  第二题关键代码：  LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)  {  // 鼠标点击相关  static float current\_x, current\_y;  static bool isChecked;  // 渲染精灵  PResDesc resDes;  BITMAP bm;  static int monsterCount = 0;  static int walkerCount = 0;  static int index = 0;  RECT rt;  GetClientRect(hwnd, &rt);  CSprite\* a;  switch (message)  {  case WM\_CREATE:  break;  case WM\_LBUTTONDOWN:  current\_x = LOWORD(lParam);  current\_y = HIWORD(lParam);  for (int i = monsterCount - 1; i > 0; i--)  {  if (pMonster[i]->IsSelected(current\_x, current\_y))  {  index = i;  pMonster[i]->SetZOrder(monsterCount + walkerCount - 1);  pSpriteManager->ReZOrder(pMonster[i]);  break;  }  }  break;  case WM\_KEYDOWN:  switch (wParam)  {  // 创建精灵  case 'W':  case 'w':  if (monsterCount > 9) break;  pMonster[monsterCount] = pSpriteManager->CreateSprite(L"Monster", rand() % (rt.right - rt.left - 128), rand() % ((rt.bottom - rt.top) / 2 - 144));  pMonster[monsterCount]->SetTransparent(true);  monsterCount++;  break;  case 'M':  case 'm':  if (walkerCount > 9) break;  pWalkers[walkerCount] = pSpriteManager->CreateAnimationSprite(L"Walker", 4, 5, rand() % (rt.right - rt.left - 32), (rt.bottom - rt.top) / 2 + rand() % ((rt.bottom - rt.top) / 2 - 54), 32, 54);  pWalkers[walkerCount]->SetTransparent(true);  pWalkers[walkerCount]->Play();  walkerCount++;  break;  // monster左上右下移动  case VK\_LEFT:  pMonster[index]->SetPos(pMonster[index]->GetXPos() - STEP, pMonster[index]->GetYPos());  break;  case VK\_UP:  pMonster[index]->SetPos(pMonster[index]->GetXPos(), pMonster[index]->GetYPos() - STEP);  break;  case VK\_RIGHT:  pMonster[index]->SetPos(pMonster[index]->GetXPos() + STEP, pMonster[index]->GetYPos());  break;  case VK\_DOWN:  pMonster[index]->SetPos(pMonster[index]->GetXPos(), pMonster[index]->GetYPos() + STEP);  break;  }  // 响应按下方向键后来判断是否碰撞  for (int i = 0; i < 10; i++)  {  if (!pWalkers[i]) break;  if (!pMonster[i]) break;  RECT rtSprite = \*(pWalkers[i]->GetBoundingBox());  if (pMonster[index]->CollideWith(rtSprite))  {  pSpriteManager->DeleteSprite(pWalkers[i], true);  walkerCount--;  }  }  break;  case WM\_PAINT:  break;  case WM\_DESTROY:  PostQuitMessage(0);  return 0;  }  return DefWindowProc(hwnd, message, wParam, lParam);  } | |
| **7、实验结果或结论（**实验结果怎么样？你从这个实验你学会了什么？得出了什么结论？） | |
| **（实验结果截图+文字说明）**  第一题效果视频如 01.mp4 所示。  第一题效果视频如 02.mp4 所示。 | |
| **8、备注或说明**  **）** | |
| **精灵管理类可以帮助开发者快速创建精灵，节省时间。** | |