

GG1003

Windows编程

胡学萱

胡学萱

实验6-3 资源管理类的实现及使用

余庆祥

2220631136

BDP

2022/11/25

**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6-3 资源管理类的实现及使用 |
| **2、实验日期** | **2022/11/25** |
| **3、实验目的** | |
| 1.理解资源管理类中资源结点的设计，理解查找资源、删除资源的实现方法。  2.会在主函数中使用资源管理类来管理众多资源，会调用相应方法查找资源，之后创建精灵来使用。 | |
| **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，使用资源管理类来管理Monster、Walker的纹理，在main中根据资源管理类中的资源创建Monster和Walker精灵。对比，体会资源管理类的好处。  **分析：**  **资源管理类对象来管理资源，那么创建对象之前的加载位图、准备设备环境等工作可以省略，交给资源管理类来创建资源即可。**  **需要创建对象时，根据资源名字查找到相应资源使用即可。**  **在程序的最后，清理资源工作交给资源管理类的析构即可。**  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）。各个精灵位置不同，因此可以看到多个精灵正常更新、渲染的效果。  **分析：**  **按下键时才要创建精灵，因此在按键时才通过资源管理类查找并使用资源，从而创建精灵。**  **多个精灵可使用数组保存；两种精灵的产生按键不同，可使用两个不同数组（也可使用一个数组吗？）；位置不同，可根据数组下标计算合适新位置。**  **退出程序前，对曾经创建了的精灵全部释放。** 思考题 精灵数量越来越多，在main中每次都需要调用它们依次更新、渲染，结束时还要一个一个释放，这是很不方便的，也容易漏操作。参考资源管理类的设计，想一想如何设计一个精灵管理类，来完成这些工作。 | |
| **6、数据记录和计算** | |
| **（代码编写思路及关键代码）**  第一题关键代码：  // 环境  RECT clientRT;  GetClientRect(hwnd, &clientRT);  HDC hdc = GetDC(hwnd);  // 资源管理类  pRes = new CResourceManager(hwnd);  pRes->LoadPictureFromFile(L"Monster", L"monster1.bmp");  pRes->LoadPictureFromFile(L"Walker", L"walker1.bmp");  // 资源变量  PResDesc resDes;  BITMAP bm;  // monster  resDes = pRes->GetResource(RES\_BMP, L"Monster");  GetObject((HBITMAP)resDes->dwHandle2, sizeof(BITMAP), &bm);  pMonster = new CSprite((HDC)resDes->dwHandle1, 10, 10, bm.bmWidth, bm.bmHeight);  pMonster->SetTransparent(true);  // walker  resDes = pRes->GetResource(RES\_BMP, L"Walker");  GetObject((HBITMAP)resDes->dwHandle2, sizeof(BITMAP), &bm);  pWalker = new CAnimationSprite((HDC)resDes->dwHandle1, 4, 5, 200, 100, bm.bmWidth / 4, bm.bmHeight / 4, 0, 0, bm.bmWidth, bm.bmHeight);  pWalker->SetTransparent(true);  pWalker->Play();  // 双缓冲  HDC hdcMem = CreateCompatibleDC(hdc);  HBITMAP hBmMem = CreateCompatibleBitmap(hdc, clientRT.right - clientRT.left, clientRT.bottom - clientRT.top);  SelectObject(hdcMem, hBmMem);  // Get systemTime  DWORD t0 = GetTickCount64(), t1;  // 循环  BOOL bMessage;  PeekMessage(&Msg, NULL, 0, 0, PM\_NOREMOVE);  while (Msg.message != WM\_QUIT)  {  bMessage = PeekMessage(&Msg, NULL, 0, 0, PM\_REMOVE);  if (bMessage)  {  TranslateMessage(&Msg);  DispatchMessage(&Msg);  }  // 设置允许世界转换的高级图形模式  SetGraphicsMode(hdcMem, GM\_ADVANCED);  // 写进双缓冲  FillRect(hdcMem, &clientRT, (HBRUSH)GetStockObject(WHITE\_BRUSH));  // 去掉多余的像素  SetStretchBltMode(hdcMem, COLORONCOLOR);  pMonster->Render(hdcMem);  t1 = GetTickCount64();  pWalker->Update((t1 - t0) / 1000.0f);  t0 = t1;  pWalker->Render(hdcMem);  // 恢复图形模式  SetGraphicsMode(hdcMem, GM\_COMPATIBLE);  // 输出精灵  BitBlt(hdc, 0, 0, clientRT.right, clientRT.bottom, hdcMem, 0, 0, SRCCOPY);  }  if (pMonster) delete pMonster;  if (pWalker) delete pWalker;  if (pRes) delete pRes;  ReleaseDC(hwnd, hdc);  第二题关键代码：  // 环境  RECT clientRT;  GetClientRect(hwnd, &clientRT);  HDC hdc = GetDC(hwnd);  // 资源管理类  pRes = new CResourceManager(hwnd);  pRes->LoadPictureFromFile(L"Monster", L"monster1.bmp");  pRes->LoadPictureFromFile(L"Walker", L"walker1.bmp");  // 双缓冲  HDC hdcMem = CreateCompatibleDC(hdc);  HBITMAP hBmMem = CreateCompatibleBitmap(hdc, clientRT.right - clientRT.left, clientRT.bottom - clientRT.top);  SelectObject(hdcMem, hBmMem);  // Get systemTime  DWORD t0 = GetTickCount64(), t1;  // 循环  BOOL bMessage;  PeekMessage(&Msg, NULL, 0, 0, PM\_NOREMOVE);  while (Msg.message != WM\_QUIT)  {  bMessage = PeekMessage(&Msg, NULL, 0, 0, PM\_REMOVE);  if (bMessage)  {  TranslateMessage(&Msg);  DispatchMessage(&Msg);  }  // 设置允许世界转换的高级图形模式  SetGraphicsMode(hdcMem, GM\_ADVANCED);  // 写进双缓冲  FillRect(hdcMem, &clientRT, (HBRUSH)GetStockObject(WHITE\_BRUSH));  // 去掉多余的像素  SetStretchBltMode(hdcMem, COLORONCOLOR);  for (int i = 0; i < monsterCount; i++)  {  pMonster[i]->Render(hdcMem);  }  t1 = GetTickCount64();  for (int i = 0; i < walkerCount; i++)  {  pWalker[i]->Update((t1 - t0) / 1000.0f);  pWalker[i]->Render(hdcMem);  }  t0 = t1;  // 恢复图形模式  SetGraphicsMode(hdcMem, GM\_COMPATIBLE);  // 输出精灵  BitBlt(hdc, 0, 0, clientRT.right, clientRT.bottom, hdcMem, 0, 0, SRCCOPY);  }  for (int i = 0; i < monsterCount; i++)  {  if (pMonster[i]) delete pMonster[i];  }  for (int i = 0; i < walkerCount; i++)  {  if (pWalker[i]) delete pWalker[i];  }    if (pRes) delete pRes;  ReleaseDC(hwnd, hdc);  return Msg.wParam;  }  LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)  {  switch (message)  {  case WM\_CREATE:  break;  case WM\_KEYDOWN:  switch (wParam)  {  case 'W':  case 'w':  if (walkerCount > 9) break;  resDes = pRes->GetResource(RES\_BMP, L"Walker");  GetObject((HBITMAP)resDes->dwHandle2, sizeof(BITMAP), &bm);  pWalker[walkerCount] = new CAnimationSprite((HDC)resDes->dwHandle1, 4, 5, 120 \* walkerCount, 200, bm.bmWidth / 4, bm.bmHeight / 4, 0, 0, bm.bmWidth, bm.bmHeight);  pWalker[walkerCount]->SetTransparent(true);  pWalker[walkerCount]->Play();  walkerCount++;  break;  case 'M':  case 'm':  if (monsterCount > 9) break;  resDes = pRes->GetResource(RES\_BMP, L"Monster");  GetObject((HBITMAP)resDes->dwHandle2, sizeof(BITMAP), &bm);  pMonster[monsterCount] = new CSprite((HDC)resDes->dwHandle1, 120 \* monsterCount, 10, bm.bmWidth, bm.bmHeight);  pMonster[monsterCount]->SetTransparent(true);  monsterCount++;    break;  }  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）**  第一题效果视频由01.mp4所示。  第二题效果视频由01.mp4所示。 | |
| **8、备注或说明**  **）** | |
| **资源管理类能够方便开发者从一个类里简单地获取纹理，不需要每次从头写代码。** | |