

GG1003

Windows编程

胡学萱

李俊琴

实验6-2 封装并使用动画精灵类

余庆祥

2220631136

BDP

2022/11/18

**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6-2 封装并使用动画精灵类 |
| **2、实验日期** | **2022/11/18** |
| **3、实验目的** | |
| 1.理解动画精灵类设计中各属性、方法的作用与含义。  2.会在主函数中使用动画精灵类创建对象，会调用相应方法完成动画精灵动画播放功能，比如停止、播放、按方向动作等。 | |
| **4、实验环境** | |
| 1、vs2019 | |
| **5、实验的步骤和方法** | |
| 1. 在mian中创建CSprite类对象pMonster，在wihle循环中将它渲染出来。键盘控制实现四方向移动。使用双缓冲解决闪烁问题。（上次实验已完成大部分工作） 2. 在题1基础上，加入动画精灵类对象pWalker到main中，while循环中将它渲染出来。W、X、A、D四个键负责它的上下左右移动，注意改变移动方向时，动画精灵纹理起始截取位置要正确设置，从而pWalker可以改变方向行走；S、P键控制它的停止、播放；当pWalker和pMonster碰撞时， pWalker消失（死亡）。 3. 动画精灵类拥有精灵类的所有功能，旋转、缩放等。在2题的基础上，按下I、D键可以使pWalker放大、缩小。此时，若两者碰撞，则高度较小的消失（死亡）。提示：应该判断包围矩形的高，而不是对象的height  思考题  1. CAnimationSprite类中，txStart、tyStart两个属性表示什么含义？在什么方法中使用，起到了什么作用？ 2. Update函数是如何实现动画精灵帧的控制的？进入新的一帧时，SetFrame函数做了什么事情？   3.精灵的种类、数量越来越多，而创建一个精灵都需要准备它的纹理，这样代码比较乱。如果设计一个资源管理类，由它来管理资源，需要创建精灵时从中查找资源来使用即可，那么，该如何设计？试着做一下。 | |
| **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：  RECT clientRT;  GetClientRect(hwnd, &clientRT);  HDC hdc = GetDC(hwnd);  // 精灵  HDC hdcSprite = CreateCompatibleDC(hdc);  HBITMAP hBmSprite = (HBITMAP)LoadImage(NULL, L"Monster.bmp", IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);  SelectObject(hdcSprite, hBmSprite);  BITMAP bm;  GetObject(hBmSprite, sizeof(BITMAP), &bm);  pMonster = new CSprite(hdcSprite, 100, 100, bm.bmWidth, bm.bmHeight);  // 双缓冲  HDC hdcMem = CreateCompatibleDC(hdc);  HBITMAP hBmMem = CreateCompatibleBitmap(hdc, clientRT.right - clientRT.left, clientRT.bottom - clientRT.top);  SelectObject(hdcMem, hBmMem);  // 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);  // 渲染  pMonster->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);  }  DeleteObject(hBmSprite);  DeleteDC(hdcSprite);  ReleaseDC(hwnd, hdc);  if (pMonster) delete pMonster;  case WM\_KEYDOWN:  switch (wParam)  {  // 左上右下移动  case VK\_LEFT:  pMonster->SetPos(pMonster->GetXPos() - 10, pMonster->GetYPos());  break;  case VK\_UP:  pMonster->SetPos(pMonster->GetXPos(), pMonster->GetYPos() - 10);  break;  case VK\_RIGHT:  pMonster->SetPos(pMonster->GetXPos() + 10, pMonster->GetYPos());  break;  case VK\_DOWN:  pMonster->SetPos(pMonster->GetXPos(), pMonster->GetYPos() + 10);  break;  }  第二题关键代码：  RECT clientRT;  GetClientRect(hwnd, &clientRT);  HDC hdc = GetDC(hwnd);  // monster  HDC hdcMonster = CreateCompatibleDC(hdc);  HBITMAP hBmMonster = (HBITMAP)LoadImage(NULL, L"Monster.bmp", IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);  SelectObject(hdcMonster, hBmMonster);  BITMAP bmMonster;  GetObject(hBmMonster, sizeof(BITMAP), &bmMonster);  pMonster = new CSprite(hdcMonster, 100, 100, bmMonster.bmWidth, bmMonster.bmHeight);  // walker  HDC hdcWalker = CreateCompatibleDC(hdc);  HBITMAP hBmWalker = (HBITMAP)LoadImage(NULL, L"walker.bmp", IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);  SelectObject(hdcWalker, hBmWalker);  BITMAP bmWalker;  GetObject(hBmWalker, sizeof(BITMAP), &bmWalker);  pWalker = new CAnimationSprite(hdcWalker, 4, 5, 500, 100, bmWalker.bmWidth / 4, bmWalker.bmHeight / 4, 0, 0, bmWalker.bmWidth, bmWalker.bmHeight);  pWalker->SetTransparent(true);  pWalker->Play();  // Get systemTime  DWORD t0 = GetTickCount(), t1;  // 双缓冲  HDC hdcMem = CreateCompatibleDC(hdc);  HBITMAP hBmMem = CreateCompatibleBitmap(hdc, clientRT.right - clientRT.left, clientRT.bottom - clientRT.top);  SelectObject(hdcMem, hBmMem);  // 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);  }  t1 = GetTickCount();  pWalker->Update((t1 - t0) / 1000.0f);  t0 = t1;  // 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);  // 渲染  if (pMonster->CollideWith(\*pWalker))  {  pWalker->SetVisible(false); // 设置不可见  }  pMonster->Render(hdcMem);  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);  }  DeleteObject(hBmMonster);  DeleteObject(hBmWalker);  DeleteDC(hdcMonster);  DeleteDC(hdcWalker);  ReleaseDC(hwnd, hdc);  if (pMonster) delete pMonster;  case WM\_KEYDOWN:  switch (wParam)  {  // monster左上右下移动  case VK\_LEFT:  pMonster->SetPos(pMonster->GetXPos() - 10, pMonster->GetYPos());  break;  case VK\_UP:  pMonster->SetPos(pMonster->GetXPos(), pMonster->GetYPos() - 10);  break;  case VK\_RIGHT:  pMonster->SetPos(pMonster->GetXPos() + 10, pMonster->GetYPos());  break;  case VK\_DOWN:  pMonster->SetPos(pMonster->GetXPos(), pMonster->GetYPos() + 10);  break;  // walker左上右下移动  case 'A':  case 'a':  pWalker->SetPos(pWalker->GetXPos() - 10, pWalker->GetYPos());  pWalker->SetTextureStartPos(0, pWalker->GetHeight() \* 1);  break;  case 'W':  case 'w':  pWalker->SetPos(pWalker->GetXPos(), pWalker->GetYPos() - 10);  pWalker->SetTextureStartPos(0, pWalker->GetHeight() \* 3);  break;  case 'D':  case 'd':  pWalker->SetPos(pWalker->GetXPos() + 10, pWalker->GetYPos());  pWalker->SetTextureStartPos(0, pWalker->GetHeight() \* 2);  break;  case 'X':  case 'x':  pWalker->SetPos(pWalker->GetXPos(), pWalker->GetYPos() + 10);  pWalker->SetTextureStartPos(0, pWalker->GetHeight() \* 0);  break;  case 'S':  case 's':  pWalker->Stop();  break;  case 'P':  case 'p':  pWalker->Resume();  break;  }  第三题关键代码：  RECT clientRT;  GetClientRect(hwnd, &clientRT);  HDC hdc = GetDC(hwnd);  // monster  HDC hdcMonster = CreateCompatibleDC(hdc);  HBITMAP hBmMonster = (HBITMAP)LoadImage(NULL, L"Monster.bmp", IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);  SelectObject(hdcMonster, hBmMonster);  BITMAP bmMonster;  GetObject(hBmMonster, sizeof(BITMAP), &bmMonster);  pMonster = new CSprite(hdcMonster, 100, 100, bmMonster.bmWidth, bmMonster.bmHeight);  // walker  HDC hdcWalker = CreateCompatibleDC(hdc);  HBITMAP hBmWalker = (HBITMAP)LoadImage(NULL, L"walker.bmp", IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);  SelectObject(hdcWalker, hBmWalker);  BITMAP bmWalker;  GetObject(hBmWalker, sizeof(BITMAP), &bmWalker);  pWalker = new CAnimationSprite(hdcWalker, 4, 5, 500, 100, bmWalker.bmWidth / 4, bmWalker.bmHeight / 4, 0, 0, bmWalker.bmWidth, bmWalker.bmHeight);  pWalker->SetTransparent(true);  pWalker->Play();  // Get systemTime  DWORD t0 = GetTickCount(), t1;  // 双缓冲  HDC hdcMem = CreateCompatibleDC(hdc);  HBITMAP hBmMem = CreateCompatibleBitmap(hdc, clientRT.right - clientRT.left, clientRT.bottom - clientRT.top);  SelectObject(hdcMem, hBmMem);  // 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);  }  t1 = GetTickCount();  pWalker->Update((t1 - t0) / 1000.0f);  t0 = t1;  // 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);  // 渲染  if (pMonster->CollideWith(\*pWalker))  {  if (pMonster->GetBoundingBox()->top <= pWalker->GetBoundingBox()->top)  {  pMonster->SetVisible(false); // 设置不可见  }  else {  pWalker->SetVisible(false); // 设置不可见  }  }  pMonster->Render(hdcMem);  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);  }  DeleteObject(hBmMonster);  DeleteObject(hBmWalker);  DeleteDC(hdcMonster);  DeleteDC(hdcWalker);  ReleaseDC(hwnd, hdc);  if (pMonster) delete pMonster;  case WM\_KEYDOWN:  switch (wParam)  {  // monster左上右下移动  case VK\_LEFT:  pMonster->SetPos(pMonster->GetXPos() - 10, pMonster->GetYPos());  break;  case VK\_UP:  pMonster->SetPos(pMonster->GetXPos(), pMonster->GetYPos() - 10);  break;  case VK\_RIGHT:  pMonster->SetPos(pMonster->GetXPos() + 10, pMonster->GetYPos());  break;  case VK\_DOWN:  pMonster->SetPos(pMonster->GetXPos(), pMonster->GetYPos() + 10);  break;  // walker左上右下移动  case 'A':  case 'a':  pWalker->SetPos(pWalker->GetXPos() - 10, pWalker->GetYPos());  pWalker->SetTextureStartPos(0, pWalker->GetHeight() \* 1);  break;  case 'W':  case 'w':  pWalker->SetPos(pWalker->GetXPos(), pWalker->GetYPos() - 10);  pWalker->SetTextureStartPos(0, pWalker->GetHeight() \* 3);  break;  case 'D':  case 'd':  pWalker->SetPos(pWalker->GetXPos() + 10, pWalker->GetYPos());  pWalker->SetTextureStartPos(0, pWalker->GetHeight() \* 2);  break;  case 'X':  case 'x':  pWalker->SetPos(pWalker->GetXPos(), pWalker->GetYPos() + 10);  pWalker->SetTextureStartPos(0, pWalker->GetHeight() \* 0);  break;  case 'S': // 停止  case 's':  pWalker->Stop();  break;  case 'P': // 播放  case 'p':  pWalker->Resume();  break;  case 'R': // 旋转  case 'r':  angle += 30;  pWalker->SetRotationAngle(angle \* PI / 180);  break;  case 'I': // 放大  case 'i':  pWalker->SetScaleFactor(scale\_x += 0.3, scale\_x += 0.3);  break;  case 'F': // 缩小  case 'f':  if (scale\_x <= 0.5 || scale\_y <= 0.5) break;  pWalker->SetScaleFactor(scale\_x -= 0.3, scale\_x -= 0.3);  break;  case 'Z': // monster放大  case 'z':  pMonster->SetScaleFactor(monster\_sx += 0.3, monster\_sx += 0.3);  break;  case 'C': // monster缩小  case 'c':  if (monster\_sx <= 0.5 || monster\_sy <= 0.5) break;  pMonster->SetScaleFactor(monster\_sx -= 0.3, monster\_sx -= 0.3);  break;  } | |
| **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）**  实验结果如录屏文件所示。  思考题结论：  1：txStart、tyStart两个属性表示当前纹理坐标的基准；在SetFrame和SetTextureStartPos这两个方法里用到；起到了改变当前纹理被截取的起始坐标。  2：计算上一帧到此时刻流逝的时间，如果大于设定帧频表示的时间间隔，则调用SetFrame函数设置下一帧的纹理坐标。 | |
| **8、备注或说明**  **）** | |
| 动画类可以帮助开发者快速设计动作，例如根据不同的按键可以表现出不同的行为动画，大大减少代码的重复率。 | |