

GG1003

Windows编程

胡学萱

李俊琴

实验6-1 封装并使用精灵类

余庆祥

2220631136

BDP

2022/11/11

**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6-1 封装并使用精灵类 |
| **2、实验日期** | **2022/11/11** |
| **3、实验目的** | |
| 1.理解精灵类设计中各属性、方法的作用与含义。  2.会在主函数中使用精灵类创建对象，会调用相应方法完成精灵渲染、平移、碰撞检测、旋转等操作。 | |
| **4、实验环境** | |
| 1、vs2019 | |
| **5、实验的步骤和方法** | |
| 1. 根据对精灵类设计的分析，实现自己的精灵类CMySprite，仅包含位置、大小、缩放比例、是否透明、纹理hdc等基本属性，包含构造析构、渲染函数，各属性的Get/Set接口函数等基本方法。之后在while循环中使用精灵对象渲染自身，结合键盘消息，按下“I”“D”键，可以放大、缩小精灵；按下方向键，可以使精灵向相应方向行走一步；按下T键可以透明渲染，O键不透明渲染。 2. 对Framework中的CSprite，在主函数中使用它，把精灵渲染在窗口中。在1题功能的基础上，R可控制旋转，鼠标点击可选中精灵（可以先判断是否选中再决定是否移动）。也可以通过GetKeyState来获取键盘按键信息。 3. 在2的基础上增加一个精灵（一个monster，一个walker），分别使用W、X、A、D和上下左右来控制两个精灵行走，当发生碰撞时，比较小的那一个消失。  思考题 在游戏中很多精灵不是静止不动的，它们总是在重复着一些动作，例如行走、奔跑、攻击等等，那么该如何表示这些运动着的精灵呢？预习动画精灵类的设计实现内容。 | |
| **6、数据记录和计算** | |
| **（代码编写思路及关键代码）**  第一题代码：  CMySprite.h：  #pragma once  #include <windows.h>  class CMySprite  {  private:  int x; // 位图输出位置x  int y; // 位图输出位置y  int w; // 位图输出宽度  int h; // 位图输出高度  HDC hdcSprite; // 纹理HDC  HDC otherHDC; // 存放其它的HDC  float scale = 1; // 缩放倍数  bool isTransparent = false; // 是否透明  public:  CMySprite(int x, int y, int w, int h, HDC hdc) // 构造函数  {  this->x = x;  this->y = y;  this->w = w;  this->h = h;  hdcSprite = hdc;  }  ~CMySprite(){}; // 析构函数    int GetXPos(); // 获得x轴位置  void SetXPos(int x\_pos); // 设置x轴位置  int GetYPos(); // 获得y轴位置  void SetYPos(int y\_pos); // 设置y轴位置  // 获取/设置Scale  float GetScale();  void SetScale(float scale\_num);  // 获取/设置isTransparent  bool GetIsTransparent();  void SetIsTransparent(bool);  void Render(HDC hdc); // 渲染函数  };  CMySprite.cpp：  #include "CMySprite.h"  #pragma comment(lib,"msimg32.lib")  // 渲染  void CMySprite::Render(HDC hdc)  {  // TODO: 输出位置需要随着方向键的改变而改变  if (GetIsTransparent())  {  // 透明渲染  TransparentBlt(hdc, x, y, w \* GetScale(), h \* GetScale(),  hdcSprite, 0, 0, w, h, GetPixel(hdcSprite, 0, 0));  return;  }  // 不透明渲染  StretchBlt(hdc, x, y, w \* GetScale(), h \* GetScale(), hdcSprite, 0, 0, w, h, SRCCOPY);  }  // 获取/设置XY轴  int CMySprite::GetXPos()  {  return x;  }  void CMySprite::SetXPos(int x\_pos)  {  x = x\_pos;  }  int CMySprite::GetYPos()  {  return y;  }  void CMySprite::SetYPos(int y\_pos)  {  y = y\_pos;  }  // 获取/设置Scale  float CMySprite::GetScale()  {  return scale;  }  void CMySprite::SetScale(float scale\_num)  {  scale = scale\_num;  }  // 获取/设置isTransparent  bool CMySprite::GetIsTransparent()  {  return isTransparent;  }  void CMySprite::SetIsTransparent(bool b)  {  isTransparent = b;  }  main.cpp关键代码：  LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)  {  switch (message)  {  case WM\_CREATE:  break;  case WM\_KEYDOWN:  switch (wParam)  {  case 'I': // 放大  mySprite->SetScale(mySprite->GetScale() + 0.3);  break;  case 'D': // 缩小  if (mySprite->GetScale() <= 0) break;  mySprite->SetScale(mySprite->GetScale() - 0.3);  break;  case 'T': // 透明  mySprite->SetIsTransparent(true);  break;  case 'O': // 不透明  mySprite->SetIsTransparent(false);  break;  // 左上右下移动  case VK\_LEFT:  mySprite->SetXPos(mySprite->GetXPos() - 1);  break;  case VK\_UP:  mySprite->SetYPos(mySprite->GetYPos() - 1);  break;  case VK\_RIGHT:  mySprite->SetXPos(mySprite->GetXPos() + 1);  break;  case VK\_DOWN:  mySprite->SetYPos(mySprite->GetYPos() + 1);  break;  }  break;  case WM\_PAINT:  break;  case WM\_DESTROY:  PostQuitMessage(0);  return 0;  }  return DefWindowProc(hwnd, message, wParam, lParam);  }  第二题关键代码：  CSprite\* mySprite;  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);  mySprite = 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);  mySprite->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 (mySprite) delete mySprite;  LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)  {  static float scale\_x = 1.0, scale\_y = 1.0;  static float angle = 0;  static int current\_x, current\_y;  static bool isChecked = false;  switch (message)  {  case WM\_CREATE:  break;  case WM\_LBUTTONDOWN:  current\_x = LOWORD(lParam);  current\_y = HIWORD(lParam);  if (mySprite->IsSelected(current\_x, current\_y))  {  isChecked = true;  }  break;  case WM\_KEYDOWN:  switch (wParam)  {  case 'I': // 放大  mySprite->SetScaleFactor(scale\_x += 0.3, scale\_x += 0.3);  break;  case 'D': // 缩小  if (scale\_x <= 0.5 || scale\_y <= 0.5) break;  mySprite->SetScaleFactor(scale\_x -= 0.3, scale\_x -= 0.3);  break;  case 'T': // 透明  mySprite->SetTransparent(true);  break;  case 'O': // 不透明  mySprite->SetTransparent(false);  break;  case 'R': // 旋转  angle += 30;  mySprite->SetRotationAngle(angle \* PI / 180);  break;  // 左上右下移动  case VK\_LEFT:  if (!isChecked) break;  mySprite->SetPos(mySprite->GetXPos() - 10, mySprite->GetYPos());  break;  case VK\_UP:  if (!isChecked) break;  mySprite->SetPos(mySprite->GetXPos(), mySprite->GetYPos() - 10);  break;  case VK\_RIGHT:  if (!isChecked) break;  mySprite->SetPos(mySprite->GetXPos() + 10, mySprite->GetYPos());  break;  case VK\_DOWN:  if (!isChecked) break;  mySprite->SetPos(mySprite->GetXPos(), mySprite->GetYPos() + 10);  break;  }  break;  case WM\_PAINT:  break;  case WM\_DESTROY:  PostQuitMessage(0);  return 0;  }  return DefWindowProc(hwnd, message, wParam, lParam);  }  第三题关键代码：  CSprite\* Monster;  CSprite\* Walker;  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);  Monster = new CSprite(hdcMonster, 100, 100, bmMonster.bmWidth, bmMonster.bmHeight);  // walker  HDC hdcWalker = CreateCompatibleDC(hdc);  HBITMAP hBmWalker = (HBITMAP)LoadImage(NULL, L"Box.bmp", IMAGE\_BITMAP, 0, 0, LR\_LOADFROMFILE);  SelectObject(hdcWalker, hBmWalker);  BITMAP bmWalker;  GetObject(hBmWalker, sizeof(BITMAP), &bmWalker);  Walker = new CSprite(hdcWalker, 500, 100, bmWalker.bmWidth, bmWalker.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);  if (Monster->CollideWith(\*Walker)) {  if(Monster->GetWidth() >= Walker->GetWidth() || Monster->GetHeight() >= Walker->GetHeight())  Monster->Render(hdcMem);  else  Walker->Render(hdcMem);  }  else {  Monster->Render(hdcMem);  Walker->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 (Monster) delete Monster;  if (Walker) delete Walker; | |
| **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：  <01.mp4>  第二题效果视频：  <02.mp4>  第三题效果视频：  <03.mp4> | |
| **8、备注或说明**  **）** | |
| **学会了利用类来快速创建对象，减少代码复用的工作量。**  **学会了利用矩阵来完成世界的坐标切换、旋转、平移、缩放等操作** | |