Win API

1. 마우스를 클릭하고 드래그하면 처음 클릭한 위치와 드래그한 마우스 위치 사이에 사각형 도형이 출력되고 마우스를 놓을시 화면에 그려지는 코드를 작성하시오

사각형을 그릴 때 DC에 투명 브러시를 선택한다

[도형 그리기, DC의 정보 수정, 그리기 모드]

|  |
| --- |
|  |
| 답안  LRESULT CALLBACK WndProc(HWND hWnd, UINT iMsg, WPARAM wParam, LPARAM lParam)  {  HDC hDC;  HBRUSH hMyBrush, hOldBrush;  static INT sx, sy, oldx, oldy;  int ex, ey;  static BOOL bNowDraw = FALSE;    switch (iMsg)  {  case WM\_LBUTTONDOWN:  sx = LOWORD(lParam);  sy = HIWORD(lParam);  oldx = sx;  oldy = sy;  bNowDraw = TRUE;  return 0;  case WM\_LBUTTONUP:  bNowDraw = FALSE;  return 0;  case WM\_MOUSEMOVE:  if (bNowDraw == TRUE) {  hDC = GetDC(hWnd);  SetROP2(hDC, R2\_NOT);  hMyBrush = reinterpret\_cast<HBRUSH>(GetStockObject(NULL\_BRUSH));  hOldBrush = reinterpret\_cast<HBRUSH>(SelectObject(hDC, hMyBrush));  Rectangle(hDC, sx, sy, oldx, oldy);  ex = LOWORD(lParam);  ey = HIWORD(lParam);  Rectangle(hDC, sx, sy, ex, ey);  oldx = ex;  oldy = ey;  ReleaseDC(hWnd, hDC);  SelectObject(hDC, hOldBrush);  }  return 0;  case WM\_DESTROY:  PostQuitMessage(0);  break;  }  return DefWindowProc(hWnd, iMsg, wParam, lParam);  } |
| 결과(화면 캡쳐) |

1. 다음과 같은 슈퍼 마리오의 화면을 주어진 타일맵과 비트맵 출력을 이용하여 재현하시오

예시와 완벽하게 동일할 필요는 없고 원하는데로 구성해도 된다

예시 밑의 타일맵을 활용하여 제작한다.

비트맵 복사함수의 인수 중 복사원의 좌표를 수정하면 하나의 비트맵 이미지로 화면에 여러가지 이미지를 출력할 수 있다

화면 크기를 조절하면 크기에 맞게 화면이 조절되게 해야한다

마리오 게임을 구현하는 것이 아닌 화면만 따라하는 문제이다

[비트맵 출력, 맵핑 모드]

|  |
| --- |
|  |
| 답안  LRESULT CALLBACK WndProc(HWND hWnd, UINT iMsg, WPARAM wParam, LPARAM lParam)  {  HDC hDC, hMem;  PAINTSTRUCT ps;  RECT rect;  HBITMAP hMyBitmap, hOldBitmap;  INT i, j;  switch (iMsg)  {  case WM\_PAINT:  hDC = BeginPaint(hWnd, &ps);  SetMapMode(hDC, MM\_ISOTROPIC);  SetWindowExtEx(hDC, 420, 200, NULL);  GetClientRect(hWnd, &rect);  SetViewportExtEx(hDC, rect.right, rect.bottom, NULL);  hMem = CreateCompatibleDC(hDC);  hMyBitmap = LoadBitmap(GetModuleHandle(NULL), MAKEINTRESOURCE(IDB\_BITMAP4));  hOldBitmap = reinterpret\_cast<HBITMAP>(SelectObject(hMem, hMyBitmap));    //배경 400, 176  StretchBlt(hDC, 0, 0, 400, 176, hMem, 128, 80, 16, 16, SRCCOPY);  StretchBlt(hDC, 128, 16, 152, 72, hMem, 416, 32, 112, 64, SRCCOPY);    //구름  StretchBlt(hDC, 16, 16, 16, 32, hMem, 0, 64, 16, 32, SRCCOPY);  for (i = 0; i < 3; i++) {  StretchBlt(hDC, 32 + i \* 16, 16, 16, 32, hMem, 16, 64, 16, 32, SRCCOPY);  }  StretchBlt(hDC, 80, 16, 16, 32, hMem, 32, 64, 16, 32, SRCCOPY);  StretchBlt(hDC, 352, 4, 48, 32, hMem, 0, 64, 48, 32, SRCCOPY);  StretchBlt(hDC, 48, 100, 48, 32, hMem, 0, 64, 48, 32, SRCCOPY);  //바닥  for (j = 0; j < 2; j++) {  for (i = 0; i < 25; i++) {  StretchBlt(hDC, 0 + i \* 16, 160 + j \* 16, 16, 16, hMem, 0, 0, 16, 16, SRCCOPY);  }  }  //물음표 벽돌  StretchBlt(hDC, 176, 100, 16, 16, hMem, 384, 0, 16, 16, SRCCOPY);  StretchBlt(hDC, 352, 100, 16, 16, hMem, 384, 0, 16, 16, SRCCOPY);  StretchBlt(hDC, 320, 100, 16, 16, hMem, 384, 0, 16, 16, SRCCOPY);  StretchBlt(hDC, 336, 40, 16, 16, hMem, 384, 0, 16, 16, SRCCOPY);  //부술 수 있는 벽돌  for (i = 0; i < 3; i++) {  StretchBlt(hDC, 304 + i \* 32, 100, 16, 16, hMem, 16, 0, 16, 16, SRCCOPY);  }  //캐릭터, 적  StretchBlt(hDC, 130, 144, 12, 16, hMem, 274, 64, 12, 16, SRCCOPY);  StretchBlt(hDC, 306, 144, 16, 16, hMem, 176, 80, 16, 16, SRCCOPY);  //덤불(?)  StretchBlt(hDC, 352, 144, 48, 16, hMem, 176, 48, 48, 16, SRCCOPY);  StretchBlt(hDC, 162, 144, 16, 16, hMem, 176, 48, 16, 16, SRCCOPY);  for (i = 0; i < 3; i++) {  StretchBlt(hDC, 178 + i \* 16, 144, 16, 16, hMem, 192, 48, 16, 16, SRCCOPY);  }  StretchBlt(hDC, 226, 144, 16, 16, hMem, 208, 48, 16, 16, SRCCOPY);  //작은 산(?)  StretchBlt(hDC, 242, 144, 16, 16, hMem, 128, 32, 16, 16, SRCCOPY); // / 모양  StretchBlt(hDC, 258, 144, 16, 16, hMem, 128, 48, 16, 16, SRCCOPY); // ㅁ" 모양  StretchBlt(hDC, 258, 128, 16, 16, hMem, 144, 32, 16, 16, SRCCOPY); // ^ 모양  StretchBlt(hDC, 274, 144, 16, 16, hMem, 160, 32, 16, 16, SRCCOPY); // \ 모양  //큰 산(?)  StretchBlt(hDC, 0, 144, 16, 16, hMem, 128, 48, 16, 16, SRCCOPY); // ㅁ" 모양  StretchBlt(hDC, 0, 128, 16, 16, hMem, 128, 32, 16, 16, SRCCOPY); // / 모양  StretchBlt(hDC, 16, 144, 16, 16, hMem, 144, 48, 16, 16, SRCCOPY); // ㅁ 모양  StretchBlt(hDC, 16, 128, 16, 16, hMem, 128, 48, 16, 16, SRCCOPY); // ㅁ" 모양  StretchBlt(hDC, 16, 112, 16, 16, hMem, 144, 32, 16, 16, SRCCOPY); // ^ 모양  StretchBlt(hDC, 32, 144, 16, 16, hMem, 160, 48, 16, 16, SRCCOPY); // "ㅁ 모양  StretchBlt(hDC, 32, 128, 16, 16, hMem, 160, 32, 16, 16, SRCCOPY); // \ 모양  StretchBlt(hDC, 48, 144, 16, 16, hMem, 160, 32, 16, 16, SRCCOPY); // \ 모양  SelectObject(hMem, hOldBitmap);  DeleteObject(hOldBitmap);  DeleteDC(hMem);  EndPaint(hWnd, &ps);;  break;  case WM\_SIZE:  InvalidateRect(hWnd, NULL, TRUE);  break;  case WM\_DESTROY:  PostQuitMessage(0);  break;  }  return DefWindowProc(hWnd, iMsg, wParam, lParam);  } |
| 결과(화면 캡쳐) |

1. (심화)다음과 같은 갤러가(갤러그)를 직접 만들어 본다

게임에 필요한 비트맵 리소스는 직접 만들어 본다

플레이어는 캐릭터를 좌우로 움직일 수 있고 스페이스를 누르면 위로 날아가는 탄환을 캐릭터에서 발사할 수 있다

탄환은 구조체를 16크기의 배열로 만들어두며 화면 밖으로 나가거나 적에 충돌하면 다시 재활용한다

16마리의 적은 상단의 고정된 위치에 배치되며 움직이지 않고 공격도 하지 않고 탄환에 맞으면 사라진다

적이 탄환에 맞아 사라지면 화면 상단 점수에 추가한다

모든 적이 사라지면 게임이 끝나며 프로그램을 종료시킨다

[비트맵 출력]

|  |
| --- |
|  |
| 답안 |
| 결과(화면 캡쳐) |