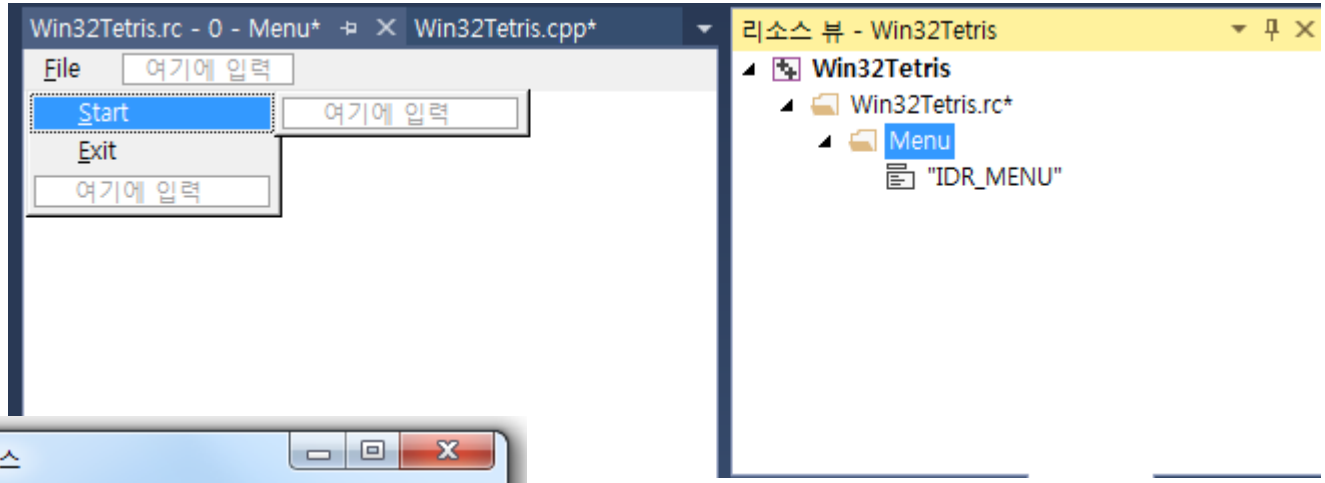
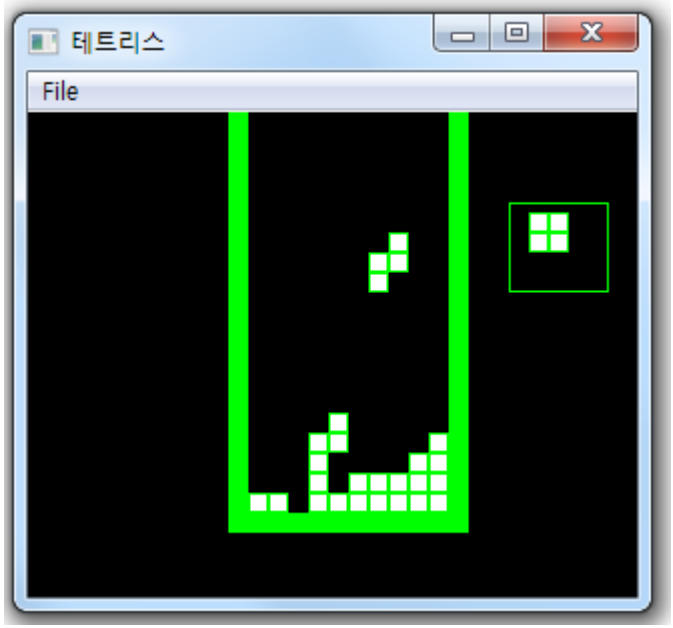
API Programming

Tetris Game

Project: Win32Tetris

Menu Structure





Project: Win32Tetris

Declare

Win32Tetris

```
#include "resource.h"
#include <windows.h>
#include <stdio.h>

int Block[7][4][4][4] = { 0,1,0,0,
0,1,0,0,
0,1,0,0,
0,0,0,0,

0,0,0,0,
1,1,1,1,
0,0,0,0,
0,0,0,0,

0,1,0,0,
0,1,0,0,
0,1,0,0,
0,1,0,0,

0,0,0,0,
1,1,1,1,
0,0,0,0,
0,0,0,0,

0,0,1,0,
0,0,1,0,
0,1,1,0,
0,0,0,0,

1,1,1,0,
0,0,1,0,
0,0,0,0,
0,0,0,0,

1,1,0,0,
1,0,0,0,
1,0,0,0,
0,0,0,0,

0,0,0,0,
1,0,0,0,
1,1,1,0,
0,0,0,0,

0,0,0,0,
0,0,1,0,
1,1,1,0,
0,0,0,0,

0,1,1,0,
0,0,1,0,
0,0,1,0,
0,0,0,0,

1,1,1,0,
1,0,0,0,
0,0,0,0,
0,0,0,0,

0,0,0,0,
0,1,0,0,
1,1,1,0,
0,0,0,0,

0,0,1,0,
0,1,1,0,
0,0,1,0,
0,0,0,0,

1,1,1,0,
0,1,0,0,
0,0,0,0,
0,0,0,0,

1,1,0,0,
0,1,1,0,
0,0,1,0,
0,0,0,0,

0,1,1,0,
1,1,0,0,
0,0,0,0,
0,0,0,0,

0,1,0,0,
0,1,1,0,
0,0,1,0,
0,0,0,0,

0,1,1,0,
1,1,0,0,
0,0,0,0,
0,0,0,0,

0,1,0,0,
1,1,0,0,
1,0,0,0,
0,0,0,0,

1,1,0,0,
0,1,1,0,
0,0,0,0,
0,0,0,0,

1,1,0,0,
1,1,0,0,
0,0,0,0,
0,0,0,0,

1,1,0,0,
1,1,0,0,
0,0,0,0,
0,0,0,0,

1,1,0,0,
1,1,0,0,
0,0,0,0,
0,0,0,0,

1,1,0,0,
1,1,0,0,
0,0,0,0,
0,0,0,0 };

int BackGround[21][12]; // 배경의 배열

#define WM_NewBlock WM_USER + 1 // 새로운 블록을 출력
#define YES 1
#define NO 0
#define SUCCESS 1
#define FAIL 0
#define ON 1
#define OFF 0
#define ALIVE 1
#define DEAD 0
```

Declare

```
160  BOOL bTime = OFF; // 타임
161
162  int BlockNum; //블록의 수
163  int RotateNum; // 블록의 회전
164  int NowX, NowY; // 현재의 블록 좌표
165  int NextBlockNum; // 다음 블럭의 수
166  int FullLineNum; //블록의 한 줄이 다 채어졌을 경우
167  int PlayerState; // 게임을 진행할지 결정
168
169  HWND hWnd;
170
171  void InitBackGround(void); // 배경의 배열 초기화 하는 함수
172  void DrawBackGround(void); // 배경을 그리는 함수
173  void DrawBlock(void); // 블록을 그리는 함수
174  void EraseBlock(void); // 블록을 지우는 함수
175  void DrawNextBlock(void); // 다음 블록을 그리는 함수
176  BOOL BlockCanMove(int x, int y); // 블록이 움직일 수 있는지를 체크하는 함수
177  void LeftMove(void); //블록을 왼쪽으로 움직이는 함수
178  void RightMove(void); //블록을 오른쪽으로 움직이는 함수
179  void Rotate(void); //블록을 회전하는 함수
180  BOOL DownMove(void); // 블록을 아래로 움직이는 함수
181  void UpdateBackGround(void); // 배경의 배열을 업데이트하는 함수
182  void CheckFullLine(void);  // 블록의 한 줄이 꽉 찼는지를 체크
183  void EraseFullLine(int); //블록의 한줄이 꽉 차면 지우는 함수
184
185  LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
186
187
188
189
```

WinMain

```
190 int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,LPSTR lpCmdLine,int nShowCmd)
191 {
192     MSG mSg;
193     char szTitle[] = "테트리스";
194     char szClass[] = "Class";
195     WNDCLASSEX WndEx;
196
197     WndEx.cbSize = sizeof(WndEx);
198     WndEx.style = NULL;
199     WndEx.lpfnWndProc = WndProc;
200     WndEx.cbClsExtra = 0;
201     WndEx.cbWndExtra = 0;
202     WndEx.hInstance = hInstance;
203     WndEx.hIcon = LoadIcon(NULL, "");
204     WndEx.hCursor = LoadCursor(NULL, IDC_ARROW);
205     WndEx.hbrBackground = (HBRUSH)GetStockObject(BLACK_BRUSH);
206     WndEx.lpszMenuName = "IDR_MENU";
207     WndEx.lpszClassName = szClass;
208     WndEx.hIconSm = LoadIcon(hInstance, "");
209
210     RegisterClassEx(&WndEx);
211
212     hWnd = CreateWindowEx(NULL,szClass,szTitle, WS_OVERLAPPEDWINDOW,
213         0,0,320,300,NULL,NULL,hInstance,NULL);
214     ShowWindow(hWnd, nShowCmd);
215     UpdateWindow(hWnd);
216
217     while (TRUE)
218     {
219         if (PeekMessage(&mSg, NULL, 0, 0, PM_NOREMOVE))
220         {
221             if (!GetMessage(&mSg, NULL, 0, 0))
222                 break;
223             TranslateMessage(&mSg);
224             DispatchMessage(&mSg);
225         }
226     }
227     return mSg.wParam;
228 }
229
```

Callback

```
230 LRESULT CALLBACK WndProc(HWND hWnd,UINT uMsg,WPARAM wParam,LPARAM lParam)
231 {
232     switch (uMsg)
233     {
234     case WM_COMMAND:
235         switch (wParam)
236         {
237         case FILE_START:
238             InitBackGround( );
239             DrawBackGround( );
240             PlayerState = ALIVE;
241             NextBlockNum = rand( ) % 7;
242             FullLineNum = 0;
243             SendMessage(hWnd, WM_NewBlock, 0, 0);
244             if (bTime == ON)
245                 KillTimer(hWnd, 3);
246             SetTimer(hWnd, 3, 1000, NULL);
247             bTime = ON;
248             break;
249         case FILE_EXIT:
250             DestroyWindow(hWnd);
251             break;
252         }
253         return FALSE;
254
```

Callback

```
255     case WM_NewBlock:
256         /* 이 메시는 새로운 블록이 들어올때 발상한다.
257         여기서는 새로운 블록을 입구에 그려주고
258         다음 블록 또한 윈도우의 오른쪽에 그려준다.*/
259         NowX = 3; // 블럭의 현재 x좌표
260         NowY = 0; // 블럭의 현재 y좌표
261         RotateNum = 0;
262         BlockNum = NextBlockNum; // 이번에 나올 블록
263         NextBlockNum = rand( ) % 7; // 다음에 나올 블록
264
265         DrawBlock( ); // 새 블록을 입그에 그린다.
266         DrawNextBlock( ); // 다음에 나올 블록을 그린다.
267
268         if (!BlockCanMove(NowX, NowY))
269             // 새 블록이 나올수 없으면
270             // 즉! 블록이 입구까지 가득차 있으면...
271             PlayerState = DEAD; // 게임을 종료한다.
272         return FALSE;
273
274     case WM_KEYDOWN:
275         switch (LOWORD(wParam))
276         {
277         case VK_LEFT:
278             LeftMove( ); break;
279         case VK_RIGHT:
280             RightMove( );break;
281         case VK_RETURN:
282             Rotate( );break;
283         case VK_DOWN:
284             DownMove( ); break;
285         case VK_SPACE:
286             while (DownMove( )); break;
287         }
288         return FALSE;
289
```

Callback

```
290         case WM_TIMER:
291             if (PlayerState == ALIVE)
292                 DownMove( );
293             else
294             {
295                 if (bTime == ON)
296                     KillTimer(hWnd, 3);
297             }
298             return FALSE;
299         case WM_DESTROY:
300             if (bTime == ON)
301                 KillTimer(hWnd, 3);
302             PostQuitMessage(0);
303             return FALSE;
304 
305         }
306         return DefWindowProc(hWnd, uMsg, wParam, lParam);
307 }
308 
309 
```

InitBackGround

```
310  void InitBackGround()
311  {
312      for (int row = 0; row<21; row++)
313          for (int col = 0; col<12; col++)
314          {
315              if (row == 20)
316                  BackGround[row][col] = 1;
317              else if (col == 0)
318                  BackGround[row][col] = 1;
319              else if (col == 11)
320                  BackGround[row][col] = 1;
321              else
322                  BackGround[row][col] = 0;
323          }
324  }
```

DrawBackGround

```
325
326 void DrawBackGround( )
327 {
328     HDC hDC = GetDC(hWnd);
329     HPEN hPen, hOldPen; // 펜은 사각형을 그릴때 사용된다.
330     HBRUSH hBrush, hOldBrush; // 브러시는 사각형을 채울때 사용된다.
331     int x, y;
332
333     hPen = CreatePen(PS_SOLID, 1, RGB(0, 255, 0)); // 초록색 펜
334     hBrush = CreateSolidBrush(RGB(0, 255, 0)); // 초록색 브러시
335
336     hOldPen = (HPEN)SelectObject(hDC, hPen);
337     hOldBrush = (HBRUSH)SelectObject(hDC, hBrush);
338
339     for (int row = 0; row<21; row++)
340         for (int col = 0; col<12; col++)
341             if (BackGround[row][col] == 1)
342             {
343                 x = 100 + col * 10;
344                 y = row * 10;
345                 Rectangle(hDC, x, y, x + 10, y + 10);
346             }
347     SelectObject(hDC, hOldPen);
348     SelectObject(hDC, hOldBrush);
349     DeleteObject(hPen);
350     DeleteObject(hBrush);
351     ReleaseDC(hWnd, hDC);
352 }
353
354
```

DrawBlock

```
355  void DrawBlock()
356  {
357      HDC hDC = GetDC(hWnd);
358      HPEN hPen, hOldPen;
359      HBRUSH hBrush, hOldBrush;
360      int x, y;
361
362      hPen = CreatePen(PS_SOLID, 1, RGB(0, 255, 0));
363      hBrush = CreateSolidBrush(RGB(255, 255, 255));
364      hOldPen = (HPEN)SelectObject(hDC, hPen);
365      hOldBrush = (HBRUSH)SelectObject(hDC, hBrush);
366      for (int row = 0; row<4; row++)
367          for (int col = 0; col<4; col++)
368              if (Block[BlockNum][RotateNum][row][col] == 1)
369              {
370                  x = 110 + NowX * 10 + col * 10;
371                  y = NowY * 10 + row * 10;
372                  Rectangle(hDC, x, y, x + 10, y + 10);
373              }
374      SelectObject(hDC, hOldPen);
375      SelectObject(hDC, hOldBrush);
376      DeleteObject(hPen);
377      DeleteObject(hBrush);
378      ReleaseDC(hWnd, hDC);
379  }
```

EraseBlock

```
380
381 void EraseBlock( )
382 {
383     HDC hDC = GetDC(hWnd);
384     HPEN hPen, hOldPen;
385     HBRUSH hBrush, hOldBrush;
386     int x, y;
387
388     hPen = CreatePen(PS_SOLID, 1, RGB(0, 0, 0));
389     hBrush = CreateSolidBrush(RGB(0, 0, 0));
390     hOldPen = (HPEN)SelectObject(hDC, hPen);
391     hOldBrush = (HBRUSH)SelectObject(hDC, hBrush);
392
393     for (int row = 0; row<4; row++)
394         for (int col = 0; col<4; col++)
395             if (Block[BlockNum][RotateNum][row][col])
396             {
397                 x = 110 + NowX * 10 + col * 10;
398                 y = NowY * 10 + row * 10;
399                 Rectangle(hDC, x, y, x + 10, y + 10);
400             }
401     SelectObject(hDC, hOldPen);
402     SelectObject(hDC, hOldBrush);
403     DeleteObject(hPen);
404     DeleteObject(hBrush);
405     ReleaseDC(hWnd, hDC);
406 }
407
408
409
```

DrawNextBlock

```
410  void DrawNextBlock()
411  {
412      HDC hDC;
413      HPEN hPen, hOldPen;
414      HBRUSH hBrush, hOldBrush;
415      int x, y;
416      // 다음 블록의 배경을 그린다.
417      hPen = CreatePen(PS_SOLID, 1, RGB(0, 255, 0));
418      hBrush = CreateSolidBrush(RGB(0, 0, 0));
419      hDC = GetDC(hWnd);
420      hOldPen = (HPEN)SelectObject(hDC, hPen);
421      hOldBrush = (HBRUSH)SelectObject(hDC, hBrush);
422      Rectangle(hDC, 240, 45, 290, 90);
423      SelectObject(hDC, hOldPen);
424      SelectObject(hDC, hOldBrush);
425      DeleteObject(hPen);
426      DeleteObject(hBrush);
427      ReleaseDC(hWnd, hDC);
428      // 다음 블록을 그린다.
429      hPen = CreatePen(PS_SOLID, 1, RGB(0, 255, 0));
430      hBrush = CreateSolidBrush(RGB(255, 255, 255));
431      hDC = GetDC(hWnd);
432      hOldPen = (HPEN)SelectObject(hDC, hPen);
433      hOldBrush = (HBRUSH)SelectObject(hDC, hBrush);
434      for (int row = 0; row<4; row++)
435          for (int col = 0; col<4; col++)
436          {
437              if (Block[NextBlockNum][RotateNum][row][col])
438              {
439                  x = 250 + col * 10;
440                  y = 50 + row * 10;
441                  Rectangle(hDC, x, y, x + 10, y + 10);
442              }
443          }
444      SelectObject(hDC, hOldPen);
445      SelectObject(hDC, hOldBrush);
446      DeleteObject(hPen);
447      DeleteObject(hBrush);
448      ReleaseDC(hWnd, hDC);
449  }
```

Block-Move

```
450
451 BOOL BlockCanMove(int x, int y)
452 {
453     int check = 0;
454     int row, col;
455     for (row = 0; row<4; row++)
456         for (col = 0; col<4; col++)
457             if (Block[BlockNum][RotateNum][row][col])
458                 check += BackGround[y + row][x + col + 1];
459     if (check == 0)
460         return YES;
461     else
462         return NO;
463 }
464
465 void LeftMove()
466 {
467     if (PlayerState == DEAD)
468         return;
469     if (BlockCanMove(NowX - 1, NowY))
470     {
471         EraseBlock();
472         NowX--;
473         DrawBlock();
474     }
475 }
476
477 void RightMove()
478 {
479     if (PlayerState == DEAD)
480         return;
481     if (BlockCanMove(NowX + 1, NowY))
482     {
483         EraseBlock();
484         NowX++;
485         DrawBlock();
486     }
487 }
488
489
```

Block-Move

```
490  void Rotate( )
491  {
492      if (PlayerState == DEAD)
493          return;
494      int temp = RotateNum;
495      RotateNum++;
496      RotateNum %= 4;
497      if (BlockCanMove(NowX, NowY))  // RotateNum 값이 1증가하면
498      {
499          RotateNum = temp;
500          EraseBlock( );
501          RotateNum++;
502          RotateNum %= 4;
503          DrawBlock( );
504      }
505      else
506          RotateNum = temp;
507  }
508
509  BOOL DownMove( )
510  {
511      if (PlayerState == DEAD) return FAIL;
512      if (BlockCanMove(NowX, NowY + 1)) // 블록이 아래로 내려 갈수 있다면
513      {
514          EraseBlock( );
515          NowY++;
516          DrawBlock( );
517          return SUCCESS;
518      }
519      else // 블록이 아래로 내겨 갈수 없다면
520      {
521          UpdateBackGround( );
522          CheckFullLine( );
523          SendMessage(hWnd, WM_NewBlock, 0, 0);
524          return FAIL;
525      }
526  }
527
528
529
```

Line

```
530  void UpdateBackGround( )
531  {
532      int element;
533      for (int row = 0; row<4; row++)
534          for (int col = 0; col<4; col++)
535          {
536              element = Block[BlockNum][RotateNum][row][col];
537              if (element)
538                  BackGround[NowY + row][NowX + col + 1] = element;
539          }
540  }
541
542  void CheckFullLine( )
543  {
544      int row, col, line;
545      int elementNum;
546      for (row = 19; row >= 0; row--)
547      {
548          elementNum = 0;
549          for (col = 1; col <= 10; col++)
550              elementNum += BackGround[row][col];
551          if (elementNum == 10) // 블록이 가득 채워져 있는지 검사한다.
552          {
553              for (line = row; line>0; line--)
554                  for (col = 1; col <= 10; col++)
555                      BackGround[line][col] = BackGround[line - 1][col];
556              for (col = 1; col <= 10; col++)
557                  BackGround[0][col] = 0;
558              EraseFullLine(row); // 블록이 가득찬 줄을 삭제한다.
559              FullLineNum++;
560              row++; // 2줄 이상이 있을 경우를 대비
561          }
562      }
563  }
564
```

Line

```
565  void EraseFullLine(int row)
566  {
567      HDC hDC, hMemDC;
568      HBITMAP hBmp;
569      int Xlen = 209 - 110 + 1;
570      /* 복사할 영역의 Ylen은 제일 위에서 가득찬 라인 바로 위까지
571      의 길이이다.
572      row는 0부터 시작하므로 row*10은 full line의 바로 위까지의 길이
573      이다.
574      즉! full line이 10번째 줄이라면 이 라인의 y 영역은
575      90~99이다. 이때 row=9이고 복사할 부분의 y영역은 0~89까지 이다.*/
576      int Ylen = row * 10;
577
578      hDC = GetDC(hWnd);
579      hMemDC = CreateCompatibleDC(hDC);
580      hBmp = CreateCompatibleBitmap(hDC, 100, 200); // 충분한 크기로 잡는다.
581      SelectObject(hMemDC, hBmp);
582
583      //화면의 일부를 메모리로 복사한다.
584      BitBlt(hMemDC, 0, 0, Xlen, Ylen, hDC, 110, 0, SRCCOPY);
585
586      // 메모리에서 화면의 다른  부분으로 복사한다.
587      // 즉! 두번째 줄부터 복사한다.
588      BitBlt(hDC, 110, 10, Xlen, Ylen, hMemDC, 0, 0, SRCCOPY);
589
590      DeleteDC(hMemDC);
591      ReleaseDC(hWnd, hDC);
592      DeleteObject(hBmp);
593  }
594
```