Komunikator Internetowy

mgr inż. Krzysztof Wegner

Kontrolki

```
HWND hWndButtonSendMessage;
HWND hWndEditMessage;
switch (message)
{
case WM_CREATE:
 hWndButton = CreateWindow(TEXT("BUTTON"),TEXT("B1"),BS_FLAT | WS_VISIBLE |
 WS_CHILD ,10,20,50,20,hwnd,(HMENU)210,hInst,NULL);
 hWndButtonSendMessage = CreateWindow(TEXT("BUTTON"),TEXT("Wyślij
                         wiadomość"),BS_FLAT | WS_VISIBLE | WS_CHILD,
                         10,20,150,20,hwnd,
                         (HMENU)ID_SEND_MESSAGE,hInst,NULL);
 hWndEditMessage = CreateWindow(TEXT("EDIT"),TEXT("Moja Wiadomość"),WS_BORDER |
 WS_VISIBLE | WS_CHILD ,10,50,250,20,hwnd,(HMENU)ID_EDIT_MESSAGE,hInst,NULL);

  hBitmap = LoadBitmap(hInst,MAKEINTRESOURCE(FIRSTAPP_BMP));
  break;
case WM_PAINT:
  HDC hDC; // uchwyt do kontekstu urządzenia
  hDC = GetDC( hwnd ); // pobranie uchwytu do kontekstu obszaru roboczego
```

W pliku FirstApp.h

```
#define ID_SEND_MESSAGE 209
#define ID_EDIT_MESSAGE 212
```

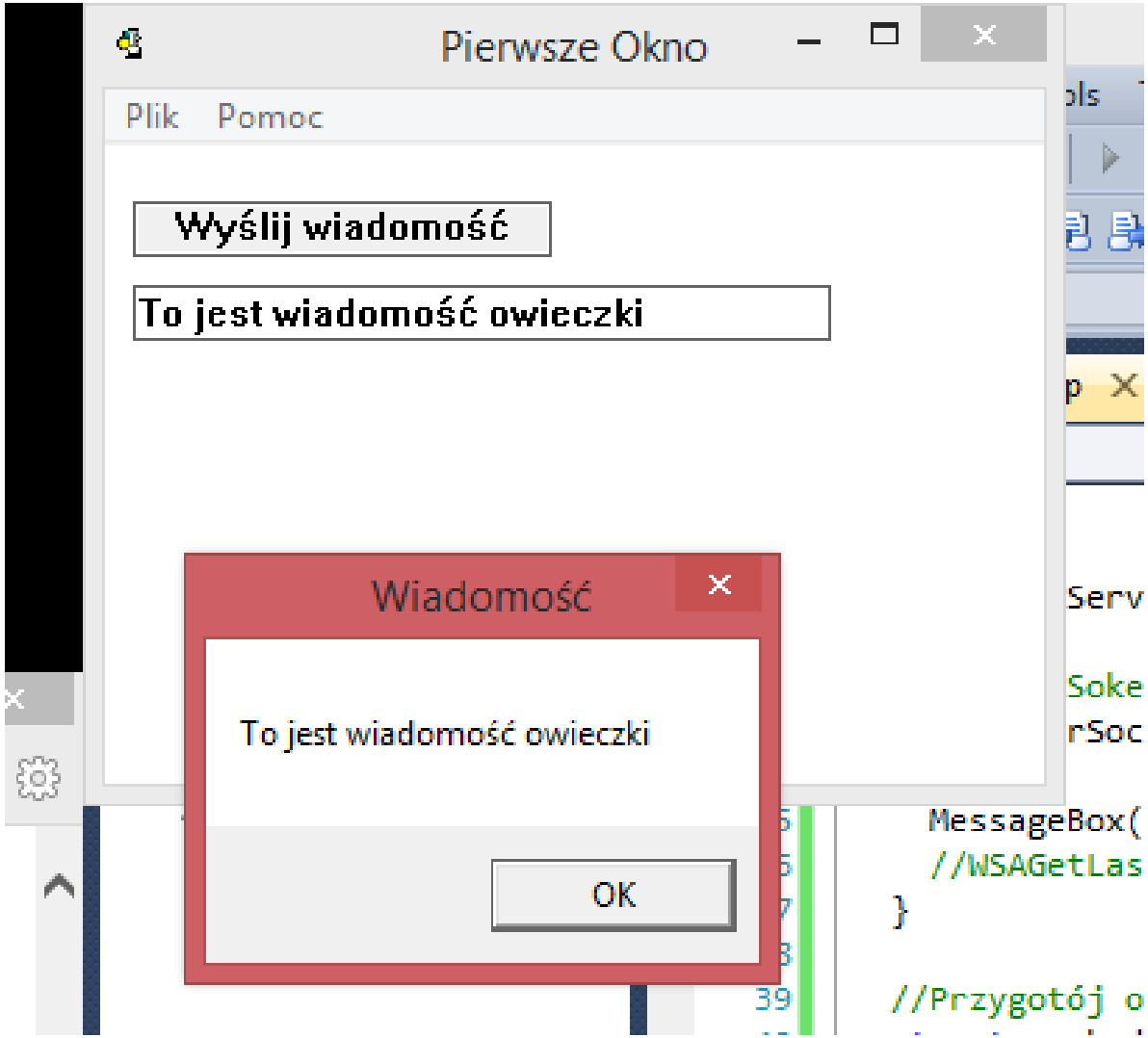
Obsługa przycisku

```
case WM_COMMAND:
    if(HIWORD(wParam)==0) //Sprawdzy czy identyfikator pozycji w menu
      switch(LOWORD(wParam))
      {
      case ID_MENU_EXIT:
        PostQuitMessage(0);
        break;
     case ID_SEND_MESSAGE:
        sendMessage(hwnd); //Wyslij wiadomość
        break;
    }
    break;
  case WM_DESTROY:
    DeleteObject(hBitmap);
    PostQuitMessage(0);
    break;
```

Funkcja wyślij wiadomość

```
void sendMessage(HWND hwnd)
{
  char sMessage[1024];
  HWND hWndEditMessage;
  hWndEditMessage=GetDlgItem(hwnd, ID_EDIT_MESSAGE);
  GetWindowText(hWndEditMessage,sMessage,1024);
  MessageBox(0,sMessage,TEXT("Wiadomość"),MB_OK);
}
```

Program


Pierwsze Okno
Plik
Pomoc
Wyślij wiadomość
To jest wiadomość owieczki
Wiadomość
To jest wiadomość owieczki
OK
MessageBox(
//WSAGetLas
}
//Przygotój o

Bufor na wiadomości

```
char asMessageList[LINES][1024];
```

Wyświetlanie wiadomości

```
char asMessageList[LINES][1024];

void displayMessageList(HDC hDC)
{
  SelectObject( hDC, GetStockObject( ANSI_VAR_FONT ) );
  SetTextColor( hDC, RGB( 255, 0, 0 ) );

  for(int i=0;i<LINES;i++)
    TextOut( hDC, 0, 70+i*20, asMessageList[i], strlen(asMessageList[i]));

}
```

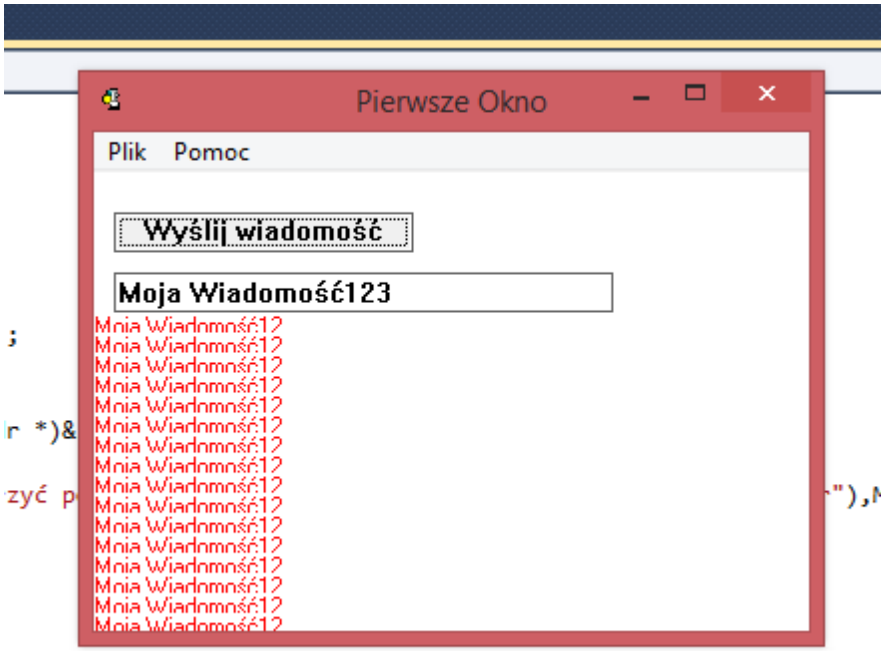
Wywołanie w obsługie komunikatu odrysowania okna

```
case WM_PAINT:
    HDC hDC; // uchwyt do kontekstu urządzenia
    hDC = GetDC( hwnd ); // pobranie uchwytu do kontekstu obszaru roboczego

    //BitBlt(hDC,0,0,800,600,GetDC(0),000,0,SRCCOPY); //skopiowanie obrazu pulpitu
   do obszaru okna
    displayMessageList(hDC);

    ReleaseDC( hwnd, hDC ); // Zwolnienie kontekstu urządzenia
    break;
```

Dodawanie wiadomości do listy

```
void sendMessage(HWND hwnd)
{
  for(int i=1;i<LINES-1;i++)
    strcpy(asMessageList[i],asMessageList[i+1]);

  //char sMessage[1024];
  HWND hWndEditMessage;
  hWndEditMessage=GetDlgItem(hwnd, ID_EDIT_MESSAGE);
  GetWindowText(hWndEditMessage,asMessageList[LINES-1],1024);
  //MessageBox(0,sMessage,TEXT("Wiadomość"),MB_OK);
}
```

Program

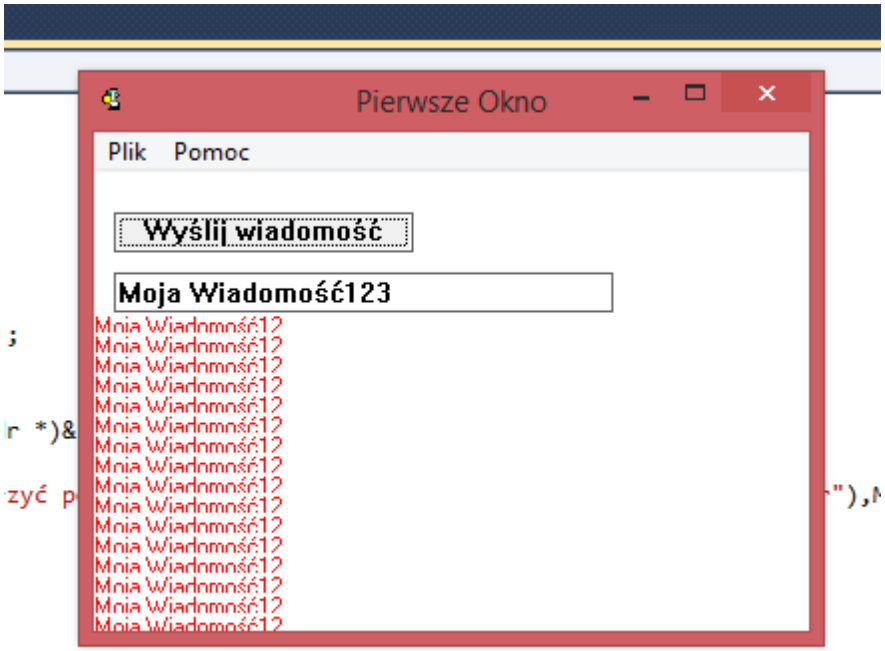


Lista odświeża się tylko jak rozciągamy okienko

Wymuszenie odmalowania okna

```
void sendMessage(HWND hwnd)
{
  for(int i=1;i<LINES-1;i++)
    strcpy(asMessageList[i],asMessageList[i+1]);

  //char sMessage[1024];
  HWND hWndEditMessage;
  hWndEditMessage=GetDlgItem(hwnd, ID_EDIT_MESSAGE);
  GetWindowText(hWndEditMessage,asMessageList[LINES-1],1024);
  //MessageBox(0,sMessage,TEXT("Wiadomość"),MB_OK);

  //Wymusznie odmalowania okna
  RECT rc;
  GetClientRect(hwnd, &rc); //Pobranie obszaru klienciego
  InvalidateRect(hwnd, NULL, false); //Uniewaznienie tresci
  RedrawWindow(hwnd,&rc,NULL,RDW_ERASE); //Wymuszenie odmalowania nieważnej
}
```

Program


Pierwsze Okno
Plik
Pomoc
Wyślij wiadomość
Moja Wiadomość123
Moja Wiadomość12

Inicjalizacja Sieci - Biblioteki

```
// Niezbędne biblioteki do obsługi sieci
#pragma comment (lib, "Ws2_32.lib")
#pragma comment (lib, "Mswsock.lib")
#pragma comment (lib, "AdvApi32.lib")
```

Inicializacja Sieci

```
if ( hWnd == NULL ) return( FALSE );

  ShowWindow(hWnd,iCmdShow);

  int iResult;
  WSADATA wsaData;
  // Initializacja Winsock
  iResult = WSAStartup(MAKEWORD(2,2), &wsaData);
  if (iResult != NO_ERROR)
  {
    MessageBox(0,TEXT("Initialization error."),TEXT("WinSock"),MB_OK);
    return 1;
  }

  ...

  // Posprzątanie Winsock
  closesocket(serverSocket);
  WSACleanup();

  MessageBox(0,TEXT("Hejka"),TEXT("Informacja"),MB_OK);
  return 0;
```

Serwer

```
void createServer()
{
  //Utwórz Soket Serwera
  if((serverSocket = socket(AF_INET , SOCK_DGRAM , 0 )) == INVALID_SOCKET)
  {
    MessageBox(0,TEXT("NIe można utworzyć Soketa"),TEXT("WinSock Error"),MB_OK);
    //WSAGetLastError()
  }
  //Przygotój opis adresu serwera
  struct sockaddr_in server, si_other;
    server.sin_family = AF_INET;
  server.sin_addr.s_addr = INADDR_ANY;
  server.sin_port = htons( SERVER_PORT );
  //Podpięcie adresy pod socket
  if( bind(serverSocket ,(struct sockaddr *)&server , sizeof(server)) ==
   SOCKET_ERROR)
  {
     MessageBox(0,TEXT("NIe można utworzyć podpiąć soketa pod wybrany
   adress"),TEXT("WinSock Error"),MB_OK);
     return;
  }
}
```

Przycisk – tworzenia serwera

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
  HWND hWndButton;
  HWND hWndButtonCreateServer;
  HWND hWndButtonSendMessage;
...
case WM_CREATE:
    hWndButton = CreateWindow(TEXT("BUTTON"),TEXT("B1"),BS_FLAT | WS_VISIBLE |
   WS_CHILD ,10,20,50,20,hwnd,(HMENU)210,hInst,NULL);

    hWndButtonCreateServer = CreateWindow(TEXT("BUTTON"),TEXT("Stwórz
   Serwer"),BS_FLAT | WS_VISIBLE | WS_CHILD
   ,170,20,150,20,hwnd,(HMENU)ID_CREATE_SERVER,hInst,NULL);
    hWndButtonSendMessage = CreateWindow(TEXT("BUTTON"),TEXT("Wyślij
   wiadomość"),BS_FLAT | WS_VISIBLE | WS_CHILD
   ,10,20,150,20,hwnd,(HMENU)ID_SEND_MESSAGE,hInst,NULL);
    hWndEditMessage = CreateWindow(TEXT("EDIT"),TEXT("Moja Wiadomość"),WS_BORDER |
   WS_VISIBLE | WS_CHILD ,10,50,250,20,hwnd,(HMENU)ID_EDIT_MESSAGE,hInst,NULL);

    hBitmap = LoadBitmap(hInst,MAKEINTRESOURCE(FIRSTAPP_BMP));
    break;
```

W pliku FirstApp.h dodajemy

```
#define ID_CREATE_SERVER 211
```

Przycisk – obsługa

```
case WM_COMMAND:
   if(HIWORD(wParam)==0) //Sprawdzy czy identyfikator pozycji w menu
     switch(LOWORD(wParam))
     {
     case ID_MENU_EXIT:
       PostQuitMessage(0);
       break;
     case ID_SEND_MESSAGE:
       sendMessage(hwnd);
       break;
     case ID_CREATE_SERVER:
       createServer();
       break;
   }
   break;
```

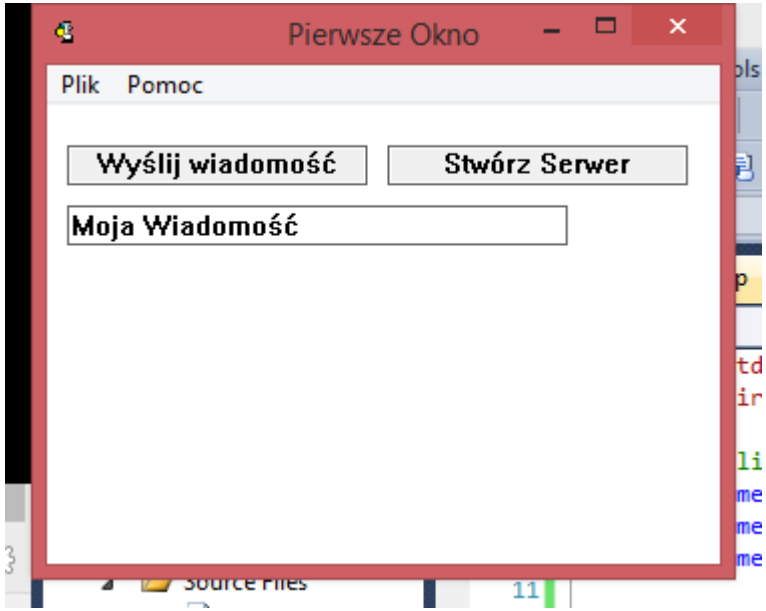
Odbierz dane

```
bool gNotExit=true;

void recieveMessage()
{
  struct sockaddr_in si_other;
  int slen , recv_len;
  char buf[1024];

  while(gNotExit)
  {
    //Wyzerowanie buffora
    memset(buf,'\0', 1024);

    //Odbierz dane
    if ((recv_len = recvfrom(serverSocket, buf, 1024, 0,
         (struct sockaddr *) &si_other, &slen)) == SOCKET_ERROR)
    {
      MessageBox(0,TEXT("Nie udało sie odebrać danych bład"),TEXT("WinSock
                 Error"),MB_OK);
      return;
    }
```

Odbierz dane - wywołanie

```
void createServer()
{
  ...

  //Podpięcie adresy pod socket
  if( bind(serverSocket ,(struct sockaddr *)&server , sizeof(server)) ==
   SOCKET_ERROR)
  {
      MessageBox(0,TEXT("NIe można utworzyć podpiąć soketa pod wybrany
   adress"),TEXT("WinSock Error"),MB_OK);
      return;
  }

  //Odbierz dane
  recieveMessage();
}
```

Program



Powinien zawiesić się po kliknięciu stwórz serwer

Utwórz klienta

```
void createClient()
{
  //Utwórz Soket Klienta
  if((clientSocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == INVALID_SOCKET)
  {
    MessageBox(0,TEXT("NIe można utworzyć Soketa"),TEXT("WinSock Error"),MB_OK);
    //WSAGetLastError()
  }
}
```

Utwórz klienta - Przycisk

```
HWND hWndButton;
  HWND hWndButtonCreateServer;
  HWND hWndButtonCreateClient;
  HWND hWndButtonSendMessage;
  HWND hWndEditMessage;
  HWND hWndAbout;
  switch (message)
  {
  case WM_CREATE:
    hWndButton = CreateWindow(TEXT("BUTTON"),TEXT("B1"),BS_FLAT | WS_VISIBLE |
   WS_CHILD ,10,20,50,20,hwnd,(HMENU)210,hInst,NULL);


    hWndButtonCreateServer = CreateWindow(TEXT("BUTTON"),TEXT("Stwórz
   Serwer"),BS_FLAT | WS_VISIBLE | WS_CHILD
   ,170,20,150,20,hwnd,(HMENU)ID_CREATE_SERVER,hInst,NULL);
    hWndButtonSendMessage = CreateWindow(TEXT("BUTTON"),TEXT("Wyślij
   wiadomość"),BS_FLAT | WS_VISIBLE | WS_CHILD
   ,10,20,150,20,hwnd,(HMENU)ID_SEND_MESSAGE,hInst,NULL);
    hWndEditMessage = CreateWindow(TEXT("EDIT"),TEXT("Moja Wiadomość"),WS_BORDER |
   WS_VISIBLE | WS_CHILD ,10,50,250,20,hwnd,(HMENU)ID_EDIT_MESSAGE,hInst,NULL);
    hWndButtonCreateClient = CreateWindow(TEXT("BUTTON"),TEXT("Stwórz
   Klienta"),BS_FLAT | WS_VISIBLE | WS_CHILD
   ,350,20,150,20,hwnd,(HMENU)ID_CREATE_CLIENT,hInst,NULL);
```

Przycisk - obsługa

```
case WM_COMMAND:
   if(HIWORD(wParam)==0) //Sprawdzy czy identyfikator pozycji w menu
     switch(LOWORD(wParam))
     {
     case ID_MENU_EXIT:
       PostQuitMessage(0);
       break;
     case ID_SEND_MESSAGE:
       sendMessage(hwnd);
       break;
     case ID_CREATE_SERVER:
       createServer();
       break;
     case ID_CREATE_CLIENT:
       createClient();
       break;
   }
```

Wysyłanie wiadomości

```
#define SERVER_ADDR "127.0.0.1"
void sendMessage(HWND hwnd)
{
...


  //Przygotój opis adresu serwera
  struct sockaddr_in si_other;
  memset((char *) &si_other, 0, sizeof(si_other));
  si_other.sin_family = AF_INET;
  si_other.sin_port = htons(SERVER_PORT);
  si_other.sin_addr.S_un.S_addr = inet_addr(SERVER_ADDR);

  int slen=sizeof(si_other);
  //send the message
  if (sendto(clientSocket, asMessageList[LINES-1], strlen(asMessageList[LINES-1]) ,
   0 , (struct sockaddr *) &si_other, slen) == SOCKET_ERROR)
  {
      MessageBox(0,TEXT("Nie udało sie wysłać danych bład"),TEXT("WinSock
   Error"),MB_OK);
      return;
  }
}
```

Server

```
void createServer()
{
 ...

  //Odbierz dane
  //recieveMessage();
  DWORD dwThreadId ;
  CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)&recieveMessage,NULL, 0,
   &dwThreadId);
}
```

```
void recieveMessage()
{
...
sprintf(asMessageList[LINES-1],"%s:%d %s",inet_ntoa(si_other.sin_addr),
   ntohs(si_other.sin_port),buf);
}
```

- Nie odrysowuje okienko po otrzymaniu wiadomości

Nowy typ danych

```
typedef struct
{
  HWND hwnd;
  SOCKET socket;
} sData;
```

Server – Odrysowanie okienka

```
void recieveMessage(sData* data)
{
  struct sockaddr_in si_other;
  int slen=sizeof(si_other), recv_len;
  char buf[1024];
...
  for(int i=1;i<LINES-1;i++)
      strcpy(asMessageList[i],asMessageList[i+1]);

    sprintf(asMessageList[LINES-1],"%s:%d %s",inet_ntoa(si_other.sin_addr),
   ntohs(si_other.sin_port),buf);

    //Wymusznie odmalowania okna
    RECT rc;
    GetClientRect(data->hwnd, &rc); //Pobranie obszaru klienciego
    InvalidateRect(data->hwnd, NULL, false); //Uniewaznienie tresci
     RedrawWindow(data->hwnd,&rc,NULL,RDW_ERASE); //Wymuszenie odmalowania niewaznej
  }
}
```

Server – Odrysowanie okienka

```
void createServer(HWND hwnd)
{
 ...

  //Odbierz dane
  //recieveMessage();
  Data* data = new sData;
  data->hwnd = hwnd;
  data->socket = serverSocket
  DWORD dwThreadId ;
  CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)&recieveMessage,data, 0,
    &dwThreadId);
}
```

- Działa

Kanał zwrotny

```
typedef struct
{
  HWND hwnd;
  SOCKET socket;
  bool server;
} sData;
```

Kanał zwrotny

```
void createServer(HWND hwnd)
{
 ...

  //Odbierz dane
  //recieveMessage();
  Data* data = new sData;
  data->hwnd = hwnd;
  data->socket = serverSocket
  data->server = true;
  DWORD dwThreadId ;
  CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)&recieveMessage,data, 0,
    &dwThreadId);
}
```

Server – Kanał zwrotny

```
void recieveMessage(sData* data)
{
...
    //Odbierz dane
    if ((recv_len = recvfrom(data->socket, buf, 1024, 0, (struct sockaddr *)
   &si_other, &slen)) == SOCKET_ERROR)
    {
      MessageBox(0,TEXT("Nie udało sie odebrać danych bład"),TEXT("WinSock
   Error"),MB_OK);
      return;
    }
    if(data->server){
    //Prześlij do innych osób
    int slen=sizeof(si_other);
    //send the message
    if (sendto(data->socket,buf, strlen(buf) , 0 , (struct sockaddr *) &si_other,
   slen) == SOCKET_ERROR)
    {
      MessageBox(0,TEXT("Nie udało sie wysłać danych bład"),TEXT("WinSock
   Error"),MB_OK);
      return;
    }
    }
```

Server – Kanał zwrotny

```
void createClient(HWND hwnd)
{
  //Utwórz Soket Klienta
  if((clientSocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == INVALID_SOCKET)
  {
    MessageBox(0,TEXT("NIe można utworzyć Soketa"),TEXT("WinSock Error"),MB_OK);
    //WSAGetLastError()
  }

  sendMessage(hwnd);

  sData* data = new sData;
  data->hwnd = hwnd;
  data->socket = clientSocket;
  data->server = false;
  DWORD dwThreadId;
  CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)&recieveMessage,data, 0,
   &dwThreadId);
}
```

Server – Kanał zwrotny

```
void sendMessage(HWND hwnd)
{
  //for(int i=1;i<LINES-1;i++)
  //  strcpy(asMessageList[i],asMessageList[i+1]);
  char sMessage[1024];
  HWND hWndEditMessage;
  hWndEditMessage=GetDlgItem(hwnd, ID_EDIT_MESSAGE);
  GetWindowText(hWndEditMessage,sMessage,1024);
  //MessageBox(0,sMessage,TEXT("Wiadomość"),MB_OK);
...
  //Przygotuj opis adresu serwera
  struct sockaddr_in si_other;
  memset((char *) &si_other, 0, sizeof(si_other));
  si_other.sin_family = AF_INET;
  si_other.sin_port = htons(SERVER_PORT);
  si_other.sin_addr.S_un.S_addr = inet_addr(SERVER_ADDR);

  int slen=sizeof(si_other);
  if (sendto(clientSocket, sMessage, strlen(sMessage) , 0 , (struct sockaddr *)
   &si_other, slen) == SOCKET_ERROR) {
      MessageBox(0,TEXT("Nie udało sie wysłać danych bład"),TEXT("WinSock
   Error"),MB_OK);
      return; }
```

- Spr

Wielu klientów

```
struct sockaddr_in siClients[100];
int iCountConnectedClients = 0;

void recieveMessage(sData* data)
{
...
    if(data->server)
    {
      //Sprawdz czy już kiedyć widziano tego klienta
      bool bFound=false;
      for(int i=0;i<iCountConnectedClients;i++)
      {
        if(siClients[i].sin_addr.S_un.S_addr==si_other.sin_addr.S_un.S_addr)
        {
          bFound = true;
          break;
        }
      }
      if(!bFound) //Jeśli nie było jeszcze nigdy to dodaj do listy
         memcpy(&siClients[iCountConnectedClients++],&si_other,sizeof(si_other));

      for(int i=0;i<iCountConnectedClients;i++)
      {
```

Wielu klientów

```
   for(int i=0;i<iCountConnectedClients;i++)
   {
   //Prześlij do innych osób
     int slen=sizeof(si_other);
     //send the message
     if (sendto(serverSocket,buf, strlen(buf) , 0 , (struct sockaddr *)
&siClients[i], slen) == SOCKET_ERROR)
     {
       MessageBox(0,TEXT("Nie udało sie wysłać danych bład"),TEXT("WinSock
Error"),MB_OK);
       return;
     }
   }
 }
```

- Działa