CS 378: Computer Game Technology

SDL_net Client-Server Example

Spring 2012

SDL_net TCP Chat Server

```
if ( SDL_Init(0) < 0 ) {
        fprintf(stderr, "Couldn't initialize SDL: %s\n",SDL_GetError());
        exit(1);
}
/* Initialize the network */
if ( SDLNet_Init() < 0 ) {
        fprintf(stderr, "Couldn't initialize net: %s\n", SDLNet_GetError());
        SDL_Quit();
        exit(1);
}
/* Initialize the channels */
for ( i=0; i<CHAT_MAXPEOPLE; ++i ) {
        people[i].active = 0;
        people[i].sock = NULL;
}
/* Allocate the socket set */
socketset = SDLNet_AllocSocketSet(CHAT_MAXPEOPLE+1);
if ( socketset == NULL ) {
        fprintf(stderr, "Couldn't create socket set: %s\n", SDLNet_GetError());
        cleanup(2);
}
```

SDL_net TCP Chat Server

```
/* Create the server socket */
SDLNet_ResolveHost(&serverIP, NULL, CHAT_PORT);
printf("Server IP: %x, %d\n", serverIP.host, serverIP.port);
servsock = SDLNet_TCP_Open(&serverIP);
if ( servsock == NULL ) {
        fprintf(stderr, "Couldn't create server socket: %s\n", SDLNet_GetError());
        cleanup(2);
}
SDLNet_TCP_AddSocket(socketset, servsock);
/* Loop, waiting for network events */
for ( ; ; ) {
        /* Wait for events */
        SDLNet_CheckSockets(socketset, ~0);
        /* Check for new connections */
        if ( SDLNet_SocketReady(servsock) ) {
                HandleServer();
        }
        /* Check for events on existing clients */
        for ( i=0; i<CHAT_MAXPEOPLE; ++i ) {
                if ( SDLNet_SocketReady(people[i].sock) ) {
                HandleClient(i);
                }
        }
}
```

Handle New Connection

```
void HandleServer(void) {
        TCPsocket newsock;
        int which;
        unsigned char data;
        newsock = SDLNet_TCP_Accept(servsock);
        if ( newsock == NULL ) { return; }
        /* Look for unconnected person slot */
        for ( which=0; which<CHAT_MAXPEOPLE; ++which )
                if ( ! people[which].sock ) { break; }
        if ( which == CHAT_MAXPEOPLE ) {
                /* Look for inactive person slot */
                for ( which=0; which<CHAT_MAXPEOPLE; ++which ) {
                        if ( people[which].sock && ! people[which].active ) {
                                /* Kick them out.. */
                                data = CHAT_BYE;
                                SDLNet_TCP_Send(people[which].sock, &data, 1);
                                SDLNet_TCP_DelSocket(socketset, people[which].sock);
                                SDLNet_TCP_Close(people[which].sock);
                                break;
                        }
                }
        }
```

Handle New Connection

```
        if ( which == CHAT_MAXPEOPLE ) {
                /* No more room... */
                data = CHAT_BYE;
                SDLNet_TCP_Send(newsock, &data, 1);
                SDLNet_TCP_Close(newsock);
        } else {
                /* Add socket as an inactive person */
                people[which].sock = newsock;
                people[which].peer = *SDLNet_TCP_GetPeerAddress(newsock);
                SDLNet_TCP_AddSocket(socketset, people[which].sock);
        }
}
```

Chat

```
void HandleClient(int which) {
        char data[512];
        int i;
        /* Has the connection been closed? */
        if ( SDLNet_TCP_Recv(people[which].sock, data, 512) <= 0 ) {
                /* Notify all active clients */
                if ( people[which].active ) {
                        people[which].active = 0;
                        data[0] = CHAT_DEL;
                        data[CHAT_DEL_SLOT] = which;
                        for ( i=0; i<CHAT_MAXPEOPLE; ++i )
                            if ( people[i].active )
                                SDLNet_TCP_Send(people[i].sock,data,CHAT_DEL_LEN);
                }
                SDLNet_TCP_DelSocket(socketset, people[which].sock);
                SDLNet_TCP_Close(people[which].sock);
                people[which].sock = NULL;
```

Chat

```
        } else {
                switch (data[0]) {
                        case CHAT_HELLO: {
                                /* Yay!  An active connection */
                                memcpy(&people[which].peer.port, &data[CHAT_HELLO_PORT], 2);
                                memcpy(people[which].name, &data[CHAT_HELLO_NAME], 256);
                                people[which].name[256] = 0;
                                /* Notify all active clients */
                                for ( i=0; i<CHAT_MAXPEOPLE; ++i )
                                        if ( people[i].active ) { SendNew(which, i); }
                                /* Notify about all active clients */
                                people[which].active = 1;
                                for ( i=0; i<CHAT_MAXPEOPLE; ++i )
                                        if ( people[i].active ) { SendNew(i, which); }
                        }
                        break;
                        default: { /* Unknown packet type?? */; }
                        break;
                }
        }
}
```

Chat

```
/* Send a "new client" notification */
void SendNew(int about, int to) {
        char data[512];
        int n;
        n = strlen((char *)people[about].name)+1;
        data[0] = CHAT_ADD;
        data[CHAT_ADD_SLOT] = about;
        memcpy(&data[CHAT_ADD_HOST], &people[about].peer.host, 4);
        memcpy(&data[CHAT_ADD_PORT], &people[about].peer.port, 2);
        data[CHAT_ADD_NLEN] = n;
        memcpy(&data[CHAT_ADD_NAME], people[about].name, n);
        SDLNet_TCP_Send(people[to].sock, data, CHAT_ADD_NAME+n);
}
```

SDL_net TCP Chat Client

```
int main(int argc, char* argv[]) {
      /* Initialize SDL */
       if ( SDL_Init(SDL_INIT_VIDEO) < 0 ) {
              fprintf(stderr, "Couldn't initialize SDL: %s\n",SDL_GetError());
              exit(1);
       }
       /* Initialize the network */
       if ( SDLNet_Init() < 0 ) {
                fprintf(stderr, "Couldn't initialize net: %s\n", SDLNet_GetError());
                SDL_Quit();
                exit(1);
       }
       /* Allocate a vector of packets for client messages */
       packets = SDLNet_AllocPacketV(4, CHAT_PACKETSIZE);
       if ( packets == NULL ) {
                fprintf(stderr, "Couldn't allocate packets: Out of memory\n");
                cleanup(2);
                }
```

SDL_net TCP Chat Client

```
/* Connect to remote host and create UDP endpoint */
server = argv[1];
termwin->AddText("Connecting to %s ... ", server);
gui->Display();
SDLNet_ResolveHost(&serverIP, server, CHAT_PORT);
if ( serverIP.host == INADDR_NONE ) {
        termwin->AddText("Couldn't resolve hostname\n");
} else {
        /* If we fail, it's okay, the GUI shows the problem */
        tcpsock = SDLNet_TCP_Open(&serverIP);
        if ( tcpsock == NULL ) {
                termwin->AddText("Connect failed\n");
        } else {
                termwin->AddText("Connected\n");
        }
}
```

SDL_Net TCP Chat Client

```
/* Try ports in the range {CHAT_PORT - CHAT_PORT+10} */
for ( i=0; (udpsock == NULL) && i<10; ++i )
        udpsock = SDLNet_UDP_Open(CHAT_PORT+i);
if ( udpsock == NULL ) {
        SDLNet_TCP_Close(tcpsock);
        tcpsock = NULL;
        termwin->AddText("Couldn't create UDP endpoint\n");
}

/* Allocate the socket set for polling the network */
socketset = SDLNet_AllocSocketSet(2);
if ( socketset == NULL ) {
        fprintf(stderr, "Couldn't create socket set: %s\n",
                                                SDLNet_GetError());
        cleanup(2);
}
SDLNet_TCP_AddSocket(socketset, tcpsock);
SDLNet_UDP_AddSocket(socketset, udpsock);

/* Run the GUI, handling network data */
SendHello(argv[2]);
```