Exercises on Network Programming

Pipolis

Department of Electrical Engineering

Nara National College of Technology

1 Goals of this exercise

Some goals of this exercise are :

1. To know how to configure TCP/IP Network on linux desktop environment.
2. To get to know linux-based network programming.
3. To create and have fun with an original game by modifying the example codes for "Pipolis."

2 Preliminaries on Inter-Process Communication(IPC)

The communication method that is assumed in this exercise is :

(Communication Type) Socket-based communication

(Communication Protocol) TCP/IP

(Connection Type) Stream type (Connection oriented)

(Communication Mode) Client-Server Model

Typical procedure on the server side is as follows :

1. Create a socket by a function : socket(),
2. Establish a communication link by a function : bind(),
3. Start to accept a request from the client side by a function : listen(),
4. Issue a system call to establish a TCP/IP connection between a server and a client by a function : accept(),
5. After the connection is established, application packets can be sent bi-directionally by functions : read() and write().

Typical procedure on the client side is as follows :

1. Create a socket by a function : socket(),
2. Issue a system call to establish a TCP/IP connection between a server and a client by a function : connect(),
3. After the connection is established, application packets can be sent bi-directionally by functions : read() and write().

3 Example Codes for "Pipolis"

"Pipolis" is a network-based game, which is very similar to the famous but old-fasioned interactive puzzle game "Tetris." The example codes for "Pipolis" was created in our laboratory and is provided for whom want to learn the fundamentals of the game programming, X-Window programming, and network programming under the linux environment. Through the example codes for "Pipolis," you can learn to draw graphics, read inputs from a keyboard and a mouse under the X-Window environment. And of course, you can understand how the network-related functions are programmed in the actual application programs that operate in the same or another PC. Regarding "Pipolis," the example codes for server-side and client-side is prepared independently although their difference is small. The programs are shown in the appendix of this exercise notes.

4 How to Compile

To run the example codes, you need to complile the program as following :
(you may need to change the whereabouts of X11R5 library shown below).

[Server Side] gcc -o tserver tserver.c /usr/local/X11R5/lib/libX11.a -lm

[Client Cide] gcc -o tclient tclient.c /usr/local/X11R5/lib/libX11.a -lm

After the successful compilation, you can run the program :

[Server Side] tserver &

[Client Side] tclient [Host Name] &

5 How to Play

You need to find out how to play the game by analyzing the example codes. Once you find out what can be done by users in the programs, you can change the key allocation as you like. The followings are the operations that can be done by the users :

- Move the block to the left
- Move the block to the right
- Move the block downward (faster)
- Rotate the block by a right angle.
- Throw down the block

6 Home Works

Some of the example home works are :

[1] Play the game between distant PCs over the Local Area Network.

[2] Create original blocks and find out how it affects the interesting-ness of the game.

[3] Modify the communication type to data-gram socket and make it possible to play with more than two players.

In addition to the home works presented above , you need to create your original version of Pipolis and submit it as a report by the end of the first semester.

Appendix 1 : Example Code for Server Side

```
#include <stdio.h>
#include <string.h>
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/keysym.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <fcntl.h>
#define USE_PORT 12322

/* スクリーン記憶用メモリ */
#define F_X 10
#define F_Y 24
char F[F_Y][F_X],F2[F_Y][F_X];

/* Xウィンドウ関連 */
Display *d;
Window w;
GC gc,gcfb,gcfs;
Colormap cmap;
XColor c,Col[20];
XEvent event;
KeySym key;
XGCValues gv;
Font fb,fs;

int sc,sct,rvflg,mc;
char MesBuf[80];
long level;

/* ブロックのパターン関連 */
struct BLOCK
{
  int x,y;
} Bpat[7][4][4]=
{ {0,0},{0,1},{1,1},{1,2} ,{0,1},{1,0},{1,1},{2,0}, /*blue    1,2*/
  {0,0},{0,1},{1,1},{1,2} ,{0,1},{1,0},{1,1},{2,0}, /*blue    3,4*/
  {0,0},{1,0},{1,1},{1,2} ,{0,1},{1,1},{2,0},{2,1}, /*red     1,2*/
  {1,0},{1,1},{1,2},{2,2} ,{0,1},{0,2},{1,1},{2,1}, /*red     3,4*/
  {0,1},{1,1},{2,1},{2,2} ,{0,2},{1,0},{1,1},{1,2}, /*magenta 1,2*/
  {0,0},{0,1},{1,1},{2,1} ,{1,0},{1,1},{1,2},{2,0}, /*magenta 2,3*/
  {0,1},{0,2},{1,0},{1,1} ,{0,0},{1,0},{1,1},{2,1}, /*green   1,2*/
  {0,1},{0,2},{1,0},{1,1} ,{0,0},{1,0},{1,1},{2,1}, /*green   3,4*/
  {0,1},{1,1},{1,2},{2,1} ,{0,1},{1,0},{1,1},{1,2}, /*cyan    1,2*/
  {0,1},{1,0},{1,1},{2,1} ,{1,0},{1,1},{1,2},{2,1}, /*cyan    3,4*/
  {0,0},{0,1},{1,0},{1,1} ,{0,0},{0,1},{1,0},{1,1}, /*yellow  1,2*/
  {0,0},{0,1},{1,0},{1,1} ,{0,0},{0,1},{1,0},{1,1}, /*yellow  3,4*/
  {0,0},{0,1},{0,2},{0,3} ,{0,0},{1,0},{2,0},{3,0}, /*white   1,2*/
  {0,0},{0,1},{0,2},{0,3} ,{0,0},{1,0},{2,0},{3,0} /*white   3,4*/
};

/* ソケット通信関連 */
struct sockaddr_in server,client;
struct hostent *host;
static char Buf[41];
int s,new_s,nbyte;

/************* Communication Related Functions ***********/
void ComInit()    /* Initialize Socket IPC */
{
  int clilen;

  /* Create a socket */
```

```
  if ((s=socket( AF_INET,SOCK_STREAM,0 ))== -1 )
  {
    perror("socket");
    exit( 1 );
  }

  /* Name and Bind the socket */
  bzero( (char *)&server,sizeof( server ) );
  server.sin_family = AF_INET;
  server.sin_addr.s_addr = INADDR_ANY;
  server.sin_port = htons( USE_PORT );
  if ( bind(s,&server,sizeof( server ))== -1 )
  {
    perror("bind");
    exit( 1 );
  }

  /* Listen */
  if ( listen( s,5 )== -1 )
  {
    perror("listen");
    exit( 1 );
  }

  /* Accept */
  clilen = sizeof( client );
  if ((new_s = accept( s,&client,&clilen ))== -1)
  {
    perror("accept");
    exit( 1 );
  }
  close( s );

  return;
}

void GetMessage()  /* クライアントからのメッセージを読み出す */
{
  if ((nbyte=read(new_s,Buf,40))== -1 )
  {
    perror("read");
    exit( 1 );
  }
  return;
}

void SendMessage( char *mes ) /* クライアントにメッセージを送る */
{
  if ((nbyte=write( new_s,mes,40)) == -1 )
  {
    perror("write");
    exit(1);
  }
  return;
}

/******************** MISC. Functions ******************/

double rnd()
{
      return (random() & 0xffffff)/(255*65535.);
}

void mesprintf( int x,int y,int c,char *s,GC g )
{
  XSetForeground( d,g,Col[(c==0 ? 0 : 5)] );
  XDrawString( d,w,g,x+1,y+1,s,strlen(s) );
  XSetForeground( d,g,Col[c] );
```

```
  XDrawString( d,w,g,x,y,s,strlen(s) );
  XFlush( d );
  return;
}

void mesdisp( int c )
{
  mc=2;
  XSetForeground( d,gcfb,Col[(c==0 ? 18 : 5 )] );
  XDrawString( d,w,gcfb,10+1,300+1,MesBuf,strlen(MesBuf) );
  XSetForeground( d,gcfb,Col[(c==0 ? 18 : 13 )] );
  XDrawString( d,w,gcfb,10,300,MesBuf,strlen(MesBuf) );
  XFlush( d );
  return;
}

/********************** 背景を描く ********************/
void DrawBackground( int bnext )
{
  int i,x,y;

  /* Nextブロックを描画する */
  XSetForeground( d,gc,Col[ 0 ] );
  XFillRectangle( d,w,gc,485,85,70,70 );
  for( i=0;i<4;++i )
  {
    x=Bpat[bnext][0][i].x;
    y=Bpat[bnext][0][i].y;
    XSetForeground( d,gc,Col[ bnext+1 ] );
    XFillRectangle( d,w,gc,485+x*16+16,85+y*16+8-(bnext==6)*8,15,15 );
    XSetForeground( d,gc,Col[ bnext+1+10 ] );
    XFillRectangle( d,w,gc,485+x*16+16+1,85+y*16+1+8-(bnext==6)*8,13,13 );
  }
  mesprintf( 497,190,6,"NEXT",gcfb );
  mesprintf( 440,335,4,"Made By",gcfb );
  mesprintf( 470,370,7,"T.Kirishima",gcfb );
  return;
}

/*************** 相手のスコアを表示する **************/
void RivScore( int flg )
{
  static int tmpsc= -1;
  char s[40];

  if ( !rvflg ) mesprintf( 10,260,12,"Your rival's DEAD!",gcfb );
  if ( tmpsc == sct && flg==0 ) return;
  XSetForeground( d,gc,Col[18] );
  XFillRectangle( d,w,gc,70,165,165,40 );
  sprintf( s,"%d",sct );
  mesprintf( 10,160,5,"Rival's Score",gcfb );
  mesprintf( 70,190,6,s,gcfb );
  tmpsc= sct;
  return;
}

/******************* ブロックを描く ******************/
void DrawBlocks( int flg )
{
  int i,i2;
  char s[40];
  XColor tmp1,tmp2;

  for( i=0;i<F_Y;++i )
  {
    for( i2=0;i2<F_X;++i2 )
    {
      if ( F[i][i2] != F2[i][i2] || flg  )
```

```
      {
F[i][i2]=F2[i][i2];
        tmp1=Col[ F[i][i2] ];
tmp2=Col[ F[i][i2]+10 ];
if ( flg == 2 ) /* Cover All Blocks with gray silouette */
{
  tmp1= Col[9];
  tmp2= Col[9+10];
  if ( F[i][i2]==0 ) continue;
}
XSetForeground( d,gc,tmp1 );
XFillRectangle( d,w,gc,240+i2*16,8+i*16,15,15 );
XSetForeground( d,gc,tmp2 );
XFillRectangle( d,w,gc,240+i2*16+1,8+i*16+1,13,13 );
      }
    }
  }
  /* スコアを表示する */
  sprintf( s,"%d",sc );
  mesprintf( 10,80,7,"Your Score",gcfb );
  mesprintf( 70,110,6,s,gcfb );
  RivScore( 0 );
  return;
}

/***************** 自分の位置にブロックをセット **************/
void SetBlock( int x,int y,int b1,int b2 )
{
  int i;

  for( i=0;i<4;++i )
  {
    F2[ y+Bpat[b1][b2][i].y ][ x+Bpat[b1][b2][i].x ]=b1 +1 ;
  }
  return;
}

/***************** 自分の位置のブロックをクリア **************/
void ClearBlock( int x,int y,int b1,int b2 )
{
  int i;

  for( i=0;i<4;++i )
  {
    F2[ y+Bpat[b1][b2][i].y ][ x+Bpat[b1][b2][i].x ]=0 ;
  }
  return;
}

/***************** 自分の位置のブロックをチェック **************/
CheckBlock( int x,int y,int b1,int b2 )
{
  int i,xx,yy,flg;

  flg=0;
  for( i=0;i<4;++i )
  {
    yy= y+Bpat[b1][b2][i].y;
    xx= x+Bpat[b1][b2][i].x;
    if ( xx<0 || xx>F_X-1 || yy<0 || yy>F_Y-1 ) {flg= -1;break;}
    else if ( F2[ yy ][ xx ] ) {flg= -1;break;}
  }
  return flg ;
}

/***************** 最下段にラインを挿入する *************/
void ScrollUp( int num )
{
```

```
  int l,i,i2;

  for( l=0; l<num ;++l )
  {
    for( i=0; i<F_Y-1 ;++i )
      for( i2=0; i2<F_X ;++i2 )
      {
F2[i][i2]=F2[i+1][i2];
      }
    for( i=0; i<F_X ;++i ) if ( F2[F_Y-1][i] ) F2[F_Y-1][i]= 8;
  }
  DrawBlocks( 0 );
  mesdisp( 0 );
  sprintf( MesBuf,"Received %d LINE!!",num );
  mesdisp( 1 );
  return;
}

/***************** 消せるラインの消去 ***************/
int DeletableLine()
{
  int i,i2,i3,i4,ln=0,sum,seed=0,cnt=0;
  char stmp[40];

  for( i=F_Y-1; i>=0 ;i-- )
  {
     sum=0;
     for( i2=0; i2<F_X ;i2++ ) sum+=( F2[i][i2]!=0 );
     if ( sum==F_X ) /* 1ライン消去可能 */
     {
       ln++;
       for( i3=i; i3>=1 ;i3-- )
       {
  for( i4=0; i4<F_X ;++i4 )
F2[ i3 ][ i4 ]=F2[ i3-1 ][ i4 ];
       }
       for( i4=0; i4<F_X ;++i4 ) F2[ 0 ][ i4 ]=0;
       i++;
       seed++;
       cnt++;
       seed *= 2;
     }
  }
  if ( cnt )
  {
    sc+= seed*50;
    if ( level-seed*50 > 100 ) level -= seed*50;
    XSetForeground( d,gc,Col[ 18 ] );
    XFillRectangle( d,w,gc,50,70,180,40 );
    DrawBlocks( 0 );
    sprintf( stmp,"s%d",sc );
    SendMessage( stmp );
    sprintf( stmp,"l%d",cnt );
    SendMessage( stmp );
    mesdisp( 0 );
    sprintf( MesBuf,"Sent %d LINE!!",cnt );
    mesdisp( 1 );
  }

  return ln;
}

void SetUpWindows()
{
  /* ウィンドウを開く */
  d=XOpenDisplay( NULL );

  cmap=DefaultColormap( d,0 );
```

```
  XAllocNamedColor( d,cmap,"dark slate blue"  ,&Col[0],&c );
  XAllocNamedColor( d,cmap,"blue"         ,&Col[1],&c );
  XAllocNamedColor( d,cmap,"red"          ,&Col[2],&c );
  XAllocNamedColor( d,cmap,"magenta"      ,&Col[3],&c );
  XAllocNamedColor( d,cmap,"green"        ,&Col[4],&c );
  XAllocNamedColor( d,cmap,"cyan"         ,&Col[5],&c );
  XAllocNamedColor( d,cmap,"yellow"       ,&Col[6],&c );
  XAllocNamedColor( d,cmap,"white"        ,&Col[7],&c );
  XAllocNamedColor( d,cmap,"gold"         ,&Col[8],&c );
  XAllocNamedColor( d,cmap,"gray"         ,&Col[9],&c );
  XAllocNamedColor( d,cmap,"dark slate blue"  ,&Col[10],&c );
  XAllocNamedColor( d,cmap,"sky blue"    ,&Col[11],&c );
  XAllocNamedColor( d,cmap,"pink"         ,&Col[12],&c );
  XAllocNamedColor( d,cmap,"maroom"       ,&Col[13],&c );
  XAllocNamedColor( d,cmap,"yellow green",&Col[14],&c );
  XAllocNamedColor( d,cmap,"navy"         ,&Col[15],&c );
  XAllocNamedColor( d,cmap,"green yellow",&Col[16],&c );
  XAllocNamedColor( d,cmap,"wheat"        ,&Col[17],&c );
  XAllocNamedColor( d,cmap,"violet red"  ,&Col[18],&c );
  XAllocNamedColor( d,cmap,"dim gray"    ,&Col[19],&c );  /* dead */

  w=XCreateSimpleWindow( d,RootWindow(d,0),10,10, 640,400, 1,0,1 );
  XSelectInput( d,w,ExposureMask | KeyPressMask | ButtonPress );
  XMapWindow( d,w );

  /* 大きいサイズの文字 */
  gcfb=XCreateGC( d,w,0,0 );
  fb=XLoadFont( d,"r24" );
  XSetFont( d,gcfb,fb );

  /* 小さいサイズの文字 */
  gcfs=XCreateGC( d,w,0,0 );
  fs=XLoadFont( d,"variable" );
  XSetFont( d,gcfs,fs );

  /* 通常の用途向け GC */
  gc=XCreateGC( d,w,0,0 );

  return;
}

/******************** ゲーム画面の再描画 ******************/
void ExposeWindow( int x,int y,int bnext,int b1,int b2 )
{
   XSetForeground( d,gc,Col[ 18 ] );
   XFillRectangle( d,w,gc,0,0,640,400 );
   XSetForeground( d,gc,Col[0] );
   XFillRectangle( d,w,gc,240,8,160,384 );
   gv.line_width = 3;
   gc=XCreateGC( d,w,GCLineWidth,&gv );

   XSetForeground( d,gc,Col[2] );
   XDrawRectangle( d,w,gc,240-3,8-3,160+6,384+6 );
   XSetForeground( d,gc,Col[12] );
   XDrawRectangle( d,w,gc,480,80,80,80 ); /* Next 用ウィンドウ */
   gv.line_width = 1;
   gc=XCreateGC( d,w,GCLineWidth,&gv );

   DrawBackground( bnext );
   SetBlock( x,y,b1,b2 );  /* ブロックをセット */
   DrawBlocks( 0 );

   return;
 }

/******************** メインプログラム ********************/
main()
{
```

```
  int i,i2,i3,i4,
      x,y,b1,b2,bnext,
      newflg,strtflg,flg1,flg2,
      vx,vy,
      turube,sum,r,tmp;
  char string[10];
  long count;

  /* 各種変数の初期化 */
  strtflg=1;
  sc=0;
  sct=0;
  flg2=1;
  newflg=1;
  rvflg=1;
  turube=0;
  count=0;
  level=30000;
  memset( F ,0,sizeof( char )*F_X*F_Y );
  memset( F2,0,sizeof( char )*F_X*F_Y );

  bnext = (int)(rnd()*7.);

  SetUpWindows();  /* 初期画面の描画 */
  XClearWindow( d,w );

  while( flg2 )
  {
    if ( newflg )              /* 新規にブロックを生成する */
    {
      x=4;
      y=0;
      b1 = bnext;
      bnext = (int)(rnd()*7.);
      DrawBackground( bnext );
      b2=(int)(rnd()*4.);
      if ( strtflg==0 )
      {
tmp=CheckBlock( x,y,b1,b2 );
SetBlock( x,y,b1,b2 );  /* ブロックをセット */
DrawBlocks( 0 );
if ( tmp == -1 ) break; /* ゲームオーバの判定 */
      }
      newflg=0;
    }
    vx=vy=0;

    /* イベントを獲得する */
    r=XCheckMaskEvent( d,ExposureMask | KeyPressMask | ButtonPress,&event );
    if ( r==False && count++ <level ) continue;

    if ( count>=level ) /* 時間切れで一個下にスライドする */
    {
      event.type = 0;
      vx=0;vy=1;
      count=0;

      if ( mc > 0 ) /* 時限メッセージ表示処理 */
      {
mc--;
if ( mc==0 ) mesdisp( 0 );
      }

      /* サーバからの情報を処理 */
      SendMessage("");
      GetMessage();
      if ( strcmp( Buf,"game over" )==0 && rvflg ) /* 相手が終了 */
      {
```

```
rvflg=0;
      }
      if ( Buf[0]=='s' ) /* 相手スコア */
      {
sct=atoi( &Buf[1] );
RivScore( 0 );
GetMessage();
if ( Buf[0]=='l' ) /* 相手が消したライン */
{
  ClearBlock( x,y,b1,b2 );
  DrawBlocks( 0 );
  ScrollUp( atoi( &Buf[1] ) );
}
      }
    }

    switch( event.type )
    {
    case KeyPress : /* キー入力 */
      if ( XLookupString( &event,string,10,&key,NULL )==1 )
      {
/*printf("%x\n",key);*/
switch( key )
{
case 0xffb4 : /* 左に移動 */
  vx= -1;vy=0;
  break;
case 0xffb6 : /* 右に移動 */
  vx=1;vy=0;
  break;
case 0xffb2 : /* 下に移動 */
  vx=0;vy=1;
  break;
case 0xffb8 : /* ブロックを回転 */
  ClearBlock( x,y,b1,b2 );
  DrawBlocks( 0 );
  flg1=0;
  if ( b2==3 ) flg1=0; else flg1=b2+1;
  if ( CheckBlock( x,y,b1,flg1 ) != -1 ) b2=flg1;
  SetBlock( x,y,b1,b2 );  /* 最新ブロックをセット */
  DrawBlocks( 0 );
  break;
case 0x20 : /* つるべ落し */
  vx=0;vy=1;
  turube=1;
  break;
case 0xff1b : /* 終了 */
  flg2=0;
  break;
} /* switch( key ) */
      } /* if */
      break;
    case Expose : /* EXPOSE 描画 */
      ExposeWindow( x,y,bnext,b1,b2 );
      DrawBlocks( 1 );
      RivScore( 1 );
      if ( strtflg==1 )
      {
mesprintf( 265,150,6,"Waiting..",gcfb );
ComInit();       /* ソケット通信初期化 */
mesprintf( 265,150,0,"Waiting..",gcfb );
mesprintf( 265,150,7,"Get Ready",gcfb );
sleep( 2 );
mesprintf( 265,150,0,"Get Ready",gcfb );
strtflg=0;
      }
      break;
    default : break;
```

```
    } /* switch */

    if ( vx || vy ) /* 実際に移動する */
    {
      do
      {
ClearBlock( x,y,b1,b2 );
if ( CheckBlock( x+vx,y+vy,b1,b2 ) != -1 )
{
  x+=vx;
  y+=vy;
} else { /* 底に足がついた */
  if ( turube==1 || vy==1 )
  {
    newflg=1;
    turube=0;
  }
} /* else */

SetBlock( x,y,b1,b2 );  /* ブロックをセット */
DrawBlocks( 0 );

      } while( turube );
      DeletableLine();     /* 消せるラインのチェック */

    } /* if ( vx || vy ) */

  } /* while */

  /* 自分がゲームオーバーしたことをサーバに伝える */
  DrawBlocks( 2 );
  mesprintf( 265,150,6,"GAME OVER",gcfb );

  /* 相手がゲームオーバーになるまで待つ */
  do {
    SendMessage("game over");
    GetMessage();
    if ( Buf[0]=='s' ) /* 相手スコア */
    {
      sct=atoi( &Buf[1] );
      RivScore( 0 );
      GetMessage();
    }
  } while( strcmp(Buf,"game over") && rvflg==1 );

  /* 勝敗判定 */
  if ( sc>sct )
  {
    mesdisp( 0 );
    sprintf( MesBuf,"You won!" );
    mesdisp( 1 );
  }
  else if ( sc<sct )
    {
    mesdisp( 0 );
    sprintf( MesBuf,"You lost!" );
    mesdisp( 1 );
  }
  else {
    mesdisp( 0 );
    sprintf( MesBuf,"EVEN GAME!!" );
    mesdisp( 1 );
  }

  close( new_s );
  sleep( 6 );
  return 0;
} /* main */
```

Appendix 2 : Example Code for Client Side

```
#include <stdio.h>
#include <string.h>
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/keysym.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <fcntl.h>
#define USE_PORT 12322

/* スクリーン記憶用メモリ */
#define F_X 10
#define F_Y 24
char F[F_Y][F_X],F2[F_Y][F_X];

/* Xウィンドウ関連 */
Display *d;
Window w;
GC gc,gcfb,gcfs;
Colormap cmap;
XColor c,Col[20];
XEvent event;
KeySym key;
XGCValues gv;
Font fb,fs;

int sc,sct,rvflg,mc;
char MesBuf[80];
long level;

/* ブロックのパターン関連 */
struct BLOCK
{
  int x,y;
} Bpat[7][4][4]=
{ {0,0},{0,1},{1,1},{1,2} ,{0,1},{1,0},{1,1},{2,0}, /*blue    1,2*/
  {0,0},{0,1},{1,1},{1,2} ,{0,1},{1,0},{1,1},{2,0}, /*blue    3,4*/
  {0,0},{1,0},{1,1},{1,2} ,{0,1},{1,1},{2,0},{2,1}, /*red     1,2*/
  {1,0},{1,1},{1,2},{2,2} ,{0,1},{0,2},{1,1},{2,1}, /*red     3,4*/
  {0,1},{1,1},{2,1},{2,2} ,{0,2},{1,0},{1,1},{1,2}, /*magenta 1,2*/
  {0,0},{0,1},{1,1},{2,1} ,{1,0},{1,1},{1,2},{2,0}, /*magenta 2,3*/
  {0,1},{0,2},{1,0},{1,1} ,{0,0},{1,0},{1,1},{2,1}, /*green   1,2*/
  {0,1},{0,2},{1,0},{1,1} ,{0,0},{1,0},{1,1},{2,1}, /*green   3,4*/
  {0,1},{1,1},{1,2},{2,1} ,{0,1},{1,0},{1,1},{1,2}, /*cyan    1,2*/
  {0,1},{1,0},{1,1},{2,1} ,{1,0},{1,1},{1,2},{2,1}, /*cyan    3,4*/
  {0,0},{0,1},{1,0},{1,1} ,{0,0},{0,1},{1,0},{1,1}, /*yellow  1,2*/
  {0,0},{0,1},{1,0},{1,1} ,{0,0},{0,1},{1,0},{1,1}, /*yellow  3,4*/
  {0,0},{0,1},{0,2},{0,3} ,{0,0},{1,0},{2,0},{3,0}, /*white   1,2*/
  {0,0},{0,1},{0,2},{0,3} ,{0,0},{1,0},{2,0},{3,0} /*white   3,4*/
};

/* ソケット通信関連 */
struct sockaddr_in server;
struct hostent *host;
static char Buf[41];
int s,nbyte;

/************* Communication Related Functions ***********/

void ComInit( char *HostName )    /* Initialize Socket IPC */
{
  /* Create a socket */
  if ((s=socket(AF_INET,SOCK_STREAM,0))== -1 )
```

```
  {
    perror("socket");
    exit(1);
  }

  /* Demand the connection */
  bzero((char *)&server,sizeof(server));
  server.sin_family = AF_INET;

  if ((host=gethostbyname(HostName))==NULL)
  {
    fprintf(stderr,"%s: Unknown host\n",HostName);
    exit(1);
  }
  bcopy( host->h_addr,&server.sin_addr,host->h_length);
  server.sin_port = htons( USE_PORT );
  if ( connect(s,&server,sizeof(server))== -1 )
  {
    perror("connect");
    exit( 1 );
  }

  return;
}

void GetMessage()  /* クライアントからのメッセージを読み出す */
{
  if ((nbyte=read( s,Buf,40 ))== -1 )
  {
    perror("read");
    exit( 1 );
  }
  return;
}

void SendMessage( char *mes ) /* クライアントにメッセージを送る */
{
  if ((nbyte=write( s,mes,40)) == -1 )
  {
    perror("write");
    exit(1);
  }
  return;
}

/******************** MISC. Functions ******************/

double rnd()
{
      return (random() & 0xffffff)/(255*65535.);
}

void mesprintf( int x,int y,int c,char *s,GC g )
{
  XSetForeground( d,g,Col[(c==0 ? 0 : 5)] );
  XDrawString( d,w,g,x+1,y+1,s,strlen(s) );
  XSetForeground( d,g,Col[c] );
  XDrawString( d,w,g,x,y,s,strlen(s) );
  XFlush( d );
  return;
}

void mesdisp( int c )
{
  mc=2;
  XSetForeground( d,gcfb,Col[(c==0 ? 18 : 5 )] );
  XDrawString( d,w,gcfb,10+1,300+1,MesBuf,strlen(MesBuf) );
  XSetForeground( d,gcfb,Col[(c==0 ? 18 : 13 )] );
```

```
  XDrawString( d,w,gcfb,10,300,MesBuf,strlen(MesBuf) );
  XFlush( d );
  return;
}

/********************** 背景を描く ********************/
void DrawBackground( int bnext )
{
  int i,x,y;

  /* Nextブロックを描画する */
  XSetForeground( d,gc,Col[ 0 ] );
  XFillRectangle( d,w,gc,485,85,70,70 );
  for( i=0;i<4;++i )
  {
    x=Bpat[bnext][0][i].x;
    y=Bpat[bnext][0][i].y;
    XSetForeground( d,gc,Col[ bnext+1 ] );
    XFillRectangle( d,w,gc,485+x*16+16,85+y*16+8-(bnext==6)*8,15,15 );
    XSetForeground( d,gc,Col[ bnext+1+10 ] );
    XFillRectangle( d,w,gc,485+x*16+16+1,85+y*16+1+8-(bnext==6)*8,13,13 );
  }
  mesprintf( 497,190,6,"NEXT",gcfb );
  mesprintf( 440,335,4,"Made By",gcfb );
  mesprintf( 470,370,7,"T.Kirishima",gcfb );
  return;
}

/****************** 相手のスコアを表示する ***************/
void RivScore( int flg )
{
  static int tmpsc= -1;
  char s[40];

  if ( !rvflg ) mesprintf( 10,260,12,"Your rival's DEAD!",gcfb );
  if ( tmpsc == sct && flg==0 ) return;
  XSetForeground( d,gc,Col[18] );
  XFillRectangle( d,w,gc,70,165,165,40 );
  sprintf( s,"%d",sct );
  mesprintf( 10,160,5,"Rival's Score",gcfb );
  mesprintf( 70,190,6,s,gcfb );
  tmpsc = sct;
  return;
}

/****************** ブロックを描く ******************/
void DrawBlocks( int flg )
{
  int i,i2;
  char s[40];
  XColor tmp1,tmp2;

  for( i=0;i<F_Y;++i )
  {
    for( i2=0;i2<F_X;++i2 )
    {
      if ( F[i][i2] != F2[i][i2] || flg  )
      {
F[i][i2]=F2[i][i2];
tmp1=Col[ F[i][i2] ];
tmp2=Col[ F[i][i2]+10 ];
if ( flg == 2 ) /* Cover All Blocks with gray silouette */
{
  tmp1= Col[9];
  tmp2= Col[9+10];
  if ( F[i][i2]==0 ) continue;
}
XSetForeground( d,gc,tmp1 );
```

```
XFillRectangle( d,w,gc,240+i2*16,8+i*16,15,15 );
XSetForeground( d,gc,tmp2 );
XFillRectangle( d,w,gc,240+i2*16+1,8+i*16+1,13,13 );
      }
    }
  }
  /* スコアを表示する */
  sprintf( s,"%d",sc );
  mesprintf( 10,80,7,"Your Score",gcfb );
  mesprintf( 70,110,6,s,gcfb );
  RivScore( 0 );
  return;
}

/***************** 自分の位置にブロックをセット **************/
void SetBlock( int x,int y,int b1,int b2 )
{
  int i;

  for( i=0;i<4;++i )
  {
    F2[ y+Bpat[b1][b2][i].y ][ x+Bpat[b1][b2][i].x ]=b1 +1 ;
  }
  return;
}

/***************** 自分の位置のブロックをクリア **************/
void ClearBlock( int x,int y,int b1,int b2 )
{
  int i;

  for( i=0;i<4;++i )
  {
    F2[ y+Bpat[b1][b2][i].y ][ x+Bpat[b1][b2][i].x ]=0 ;
  }
  return;
}

/***************** 自分の位置のブロックをチェック **************/
CheckBlock( int x,int y,int b1,int b2 )
{
  int i,xx,yy,flg;

  flg=0;
  for( i=0;i<4;++i )
  {
    yy= y+Bpat[b1][b2][i].y;
    xx= x+Bpat[b1][b2][i].x;
    if ( xx<0 || xx>F_X-1 || yy<0 || yy>F_Y-1 ) {flg= -1;break;}
    else if ( F2[ yy ][ xx ] ) {flg= -1;break;}
  }
  return flg ;
}

/************* 最下段にラインを挿入する ************/
void ScrollUp( int num )
{
  int l,i,i2;

  for( l=0; l<num ;++l )
  {
    for( i=0; i<F_Y-1 ;++i )
      for( i2=0; i2<F_X ;++i2 )
      {
F2[i][i2]=F2[i+1][i2];
      }
    for( i=0; i<F_X ;++i ) if ( F2[F_Y-1][i] ) F2[F_Y-1][i]= 8;
  }
```

```
  DrawBlocks( 0 );
  mesdisp( 0 );
  sprintf( MesBuf,"Received %d LINE!!",num );
  mesdisp( 1 );
  return;
}

/***************** 消せるラインの消去 ***************/
int DeletableLine()
{
  int i,i2,i3,i4,ln=0,sum,seed=0,cnt=0;
  char stmp[40];

  for( i=F_Y-1; i>=0 ;i-- )
  {
     sum=0;
     for( i2=0; i2<F_X ;i2++ ) sum+=( F2[i][i2]!=0 );
     if ( sum==F_X ) /* 1ライン消去可能 */
     {
       ln++;
       for( i3=i; i3>=1 ;i3-- )
       {
  for( i4=0; i4<F_X ;++i4 )
F2[ i3 ][ i4 ]=F2[ i3-1 ][ i4 ];
       }
       for( i4=0; i4<F_X ;++i4 ) F2[ 0 ][ i4 ]=0;
       i++;
       seed++;
       cnt++;
       seed *= 2;
     }
  }
  if ( cnt )
  {
    sc+= seed*50;
    if ( level-seed*50 > 100 ) level -= seed*50;
    XSetForeground( d,gc,Col[ 18 ] );
    XFillRectangle( d,w,gc,50,70,180,40 );
    DrawBlocks( 0 );
    sprintf( stmp,"s%d",sc );
    SendMessage( stmp );
    sprintf( stmp,"l%d",cnt );
    SendMessage( stmp );
    mesdisp( 0 );
    sprintf( MesBuf,"Sent %d LINE!!",cnt );
    mesdisp( 1 );
  }

  return ln;
}

void SetUpWindows()
{
  /* ウィンドウを開く */
  d=XOpenDisplay( NULL );

  cmap=DefaultColormap( d,0 );
  XAllocNamedColor( d,cmap,"dark slate blue"  ,&Col[0],&c );
  XAllocNamedColor( d,cmap,"blue"        ,&Col[1],&c );
  XAllocNamedColor( d,cmap,"red"         ,&Col[2],&c );
  XAllocNamedColor( d,cmap,"magenta"     ,&Col[3],&c );
  XAllocNamedColor( d,cmap,"green"       ,&Col[4],&c );
  XAllocNamedColor( d,cmap,"cyan"        ,&Col[5],&c );
  XAllocNamedColor( d,cmap,"yellow"      ,&Col[6],&c );
  XAllocNamedColor( d,cmap,"white"       ,&Col[7],&c );
  XAllocNamedColor( d,cmap,"gold"        ,&Col[8],&c );
  XAllocNamedColor( d,cmap,"gray"        ,&Col[9],&c );
  XAllocNamedColor( d,cmap,"dark slate blue"  ,&Col[10],&c );
```

```
  XAllocNamedColor( d,cmap,"sky blue"    ,&Col[11],&c );
  XAllocNamedColor( d,cmap,"pink"        ,&Col[12],&c );
  XAllocNamedColor( d,cmap,"maroom"      ,&Col[13],&c );
  XAllocNamedColor( d,cmap,"yellow green",&Col[14],&c );
  XAllocNamedColor( d,cmap,"navy"        ,&Col[15],&c );
  XAllocNamedColor( d,cmap,"green yellow",&Col[16],&c );
  XAllocNamedColor( d,cmap,"wheat"       ,&Col[17],&c );
  XAllocNamedColor( d,cmap,"violet red"  ,&Col[18],&c );
  XAllocNamedColor( d,cmap,"dim gray"    ,&Col[19],&c );  /* gray */

  w=XCreateSimpleWindow( d,RootWindow(d,0),10,10, 640,400, 1,0,1 );
  XSelectInput( d,w,ExposureMask | KeyPressMask | ButtonPress );
  XMapWindow( d,w );

  /* 大きいサイズの文字 */
  gcfb=XCreateGC( d,w,0,0 );
  fb=XLoadFont( d,"r24" );
  XSetFont( d,gcfb,fb );

  /* 小さいサイズの文字 */
  gcfs=XCreateGC( d,w,0,0 );
  fs=XLoadFont( d,"variable" );
  XSetFont( d,gcfs,fs );

  /* 通常の用途向け GC */
  gc=XCreateGC( d,w,0,0 );

  return;
}

/******************** ゲーム画面の再描画 ******************/
void ExposeWindow( int x,int y,int bnext,int b1,int b2 )
{
   XSetForeground( d,gc,Col[ 18 ] );
   XFillRectangle( d,w,gc,0,0,640,400 );
   XSetForeground( d,gc,Col[0] );
   XFillRectangle( d,w,gc,240,8,160,384 );
   gv.line_width = 3;
   gc=XCreateGC( d,w,GCLineWidth,&gv );

   XSetForeground( d,gc,Col[2] );
   XDrawRectangle( d,w,gc,240-3,8-3,160+6,384+6 );
   XSetForeground( d,gc,Col[12] );
   XDrawRectangle( d,w,gc,480,80,80,80 ); /* Next 用ウィンドウ */
   gv.line_width = 1;
   gc=XCreateGC( d,w,GCLineWidth,&gv );

   DrawBackground( bnext );
   SetBlock( x,y,b1,b2 );  /* ブロックをセット */
   DrawBlocks( 0 );

   return;
 }

/********************** メインプログラム **********************/
main( int argc,char **argv )
{
  int i,i2,i3,i4,
      x,y,b1,b2,bnext,
      newflg,strtflg,flg1,flg2,
      vx,vy,
      turube,sum,r,tmp;
  char string[10];
  long count;

  /* 引数のチェック */
  if ( argc != 2 )
  {
```

```
    fprintf( stderr,"Argument Error. You have to specify HostName.\n");
    exit( 1 );
  }

  /* 各種変数の初期化 */
  strtflg=1;
  sc=0;
  sct=0;
  flg2=1;
  newflg=1;
  rvflg=1;
  turube=0;
  count=0;
  level=30000;
  memset( F ,0,sizeof( char )*F_X*F_Y );
  memset( F2,0,sizeof( char )*F_X*F_Y );

  bnext = (int)(rnd()*7.);

  SetUpWindows();  /* 初期画面の描画 */
  XClearWindow( d,w );

  while( flg2 )
  {
    if ( newflg )              /* 新規にブロックを生成する */
    {
      x=4;
      y=0;
      b1 = bnext;
      bnext = (int)(rnd()*7.);
      DrawBackground( bnext );
      b2=(int)(rnd()*4.);
      if ( strtflg==0 )
      {
tmp=CheckBlock( x,y,b1,b2 );
SetBlock( x,y,b1,b2 );  /* ブロックをセット */
DrawBlocks( 0 );
if ( tmp == -1 ) break; /* ゲームオーバの判定 */
      }
      newflg=0;
    }
    vx=vy=0;

    /* イベントを獲得する */
    r=XCheckMaskEvent( d,ExposureMask | KeyPressMask | ButtonPress,&event );
    if ( r==False && count++ <level ) continue;

    if ( count>=level ) /* 時間切れで一個下にスライドする */
    {
      event.type = 0;
      vx=0;vy=1;
      count=0;

      if ( mc>0 ) /* 時限メッセージ表示処理 */
      {
mc--;
if ( mc==0 ) mesdisp( 0 );
      }
      /* クライアントからの情報を処理 */
      SendMessage("");
      GetMessage();
      if ( strcmp( Buf,"game over" )==0 && rvflg ) /* 相手が終了 */
      {
rvflg=0;
      }
      if ( Buf[0]=='s' ) /* 相手スコア */
      {
sct=atoi( &Buf[1] );
```

```
RivScore( 0 );
GetMessage();
if ( Buf[0]=='l' ) /* 相手が消したライン */
{
  ClearBlock( x,y,b1,b2 );
  DrawBlocks( 0 );
  ScrollUp( atoi( &Buf[1] ) );
}
      }
    }

    switch( event.type )
    {
    case KeyPress : /* キー入力 */
      if ( XLookupString( &event,string,10,&key,NULL )==1 )
      {
/*printf("%x\n",key);*/
switch( key )
{
case 0xffb4 : /* 左に移動 */
  vx= -1;vy=0;
  break;
case 0xffb6 : /* 右に移動 */
  vx=1;vy=0;
  break;
case 0xffb2 : /* 下に移動 */
  vx=0;vy=1;
  break;
case 0xffb8 : /* ブロックを回転 */
  ClearBlock( x,y,b1,b2 );
  DrawBlocks( 0 );
  flg1=0;
  if ( b2==3 ) flg1=0; else flg1=b2+1;
  if ( CheckBlock( x,y,b1,flg1 ) != -1 ) b2=flg1;
  SetBlock( x,y,b1,b2 );  /* 最新ブロックをセット */
  DrawBlocks( 0 );
  break;
case 0x20 : /* つるべ落し */
  vx=0;vy=1;
  turube=1;
  break;
case 0xff1b : /* 終了 */
  flg2=0;
  break;
} /* switch( key ) */
      } /* if */
      break;
    case Expose : /* EXPOSE 描画 */
      ExposeWindow( x,y,bnext,b1,b2 );
      DrawBlocks( 1 );
      RivScore( 1 );
      if ( strtflg==1 )
      {
ComInit( argv[1] );       /* ソケット通信初期化 */
        mesprintf( 265,150,7,"Get Ready",gcfb );
sleep( 2 );
mesprintf( 265,150,0,"Get Ready",gcfb );
strtflg=0;
      }
      break;
    default : break;
    } /* switch */

    if ( vx || vy ) /* 実際に移動する */
    {
      do
      {
ClearBlock( x,y,b1,b2 );
```

```
if ( CheckBlock( x+vx,y+vy,b1,b2 ) != -1 )
{
  x+=vx;
  y+=vy;
} else { /* 底に足がついた */
  if ( turube==1 || vy==1 )
  {
    newflg=1;
    turube=0;
  }
} /* else */

SetBlock( x,y,b1,b2 );  /* ブロックをセット */
DrawBlocks( 0 );

      } while( turube );
      DeletableLine();     /* 消せるラインのチェック */

    } /* if ( vx || vy ) */

  } /* while */

  /* 自分がゲームオーバーしたことをクライアントに伝える */
  DrawBlocks( 2 );
  mesprintf( 265,150,6,"GAME OVER",gcfb );

  /* 相手がゲームオーバーになるまで待つ */
  do {
    SendMessage("game over");
    GetMessage();
    if ( Buf[0]=='s' ) /* 相手スコア */
    {
      sct=atoi( &Buf[1] );
      RivScore( 0 );
      GetMessage();
    }
  } while( strcmp(Buf,"game over") && rvflg==1 );

  /* 勝敗判定 */
  if ( sc>sct )
  {
    mesdisp( 0 );
    sprintf( MesBuf,"You won!" );
    mesdisp( 1 );
  }
  else if ( sc<sct )
    {
    mesdisp( 0 );
    sprintf( MesBuf,"You lost!" );
    mesdisp( 1 );
  }
  else {
    mesdisp( 0 );
    sprintf( MesBuf,"EVEN GAME!!" );
    mesdisp( 1 );
  }

  close( s );
  sleep( 6 );
  return 0;
} /* main */
```