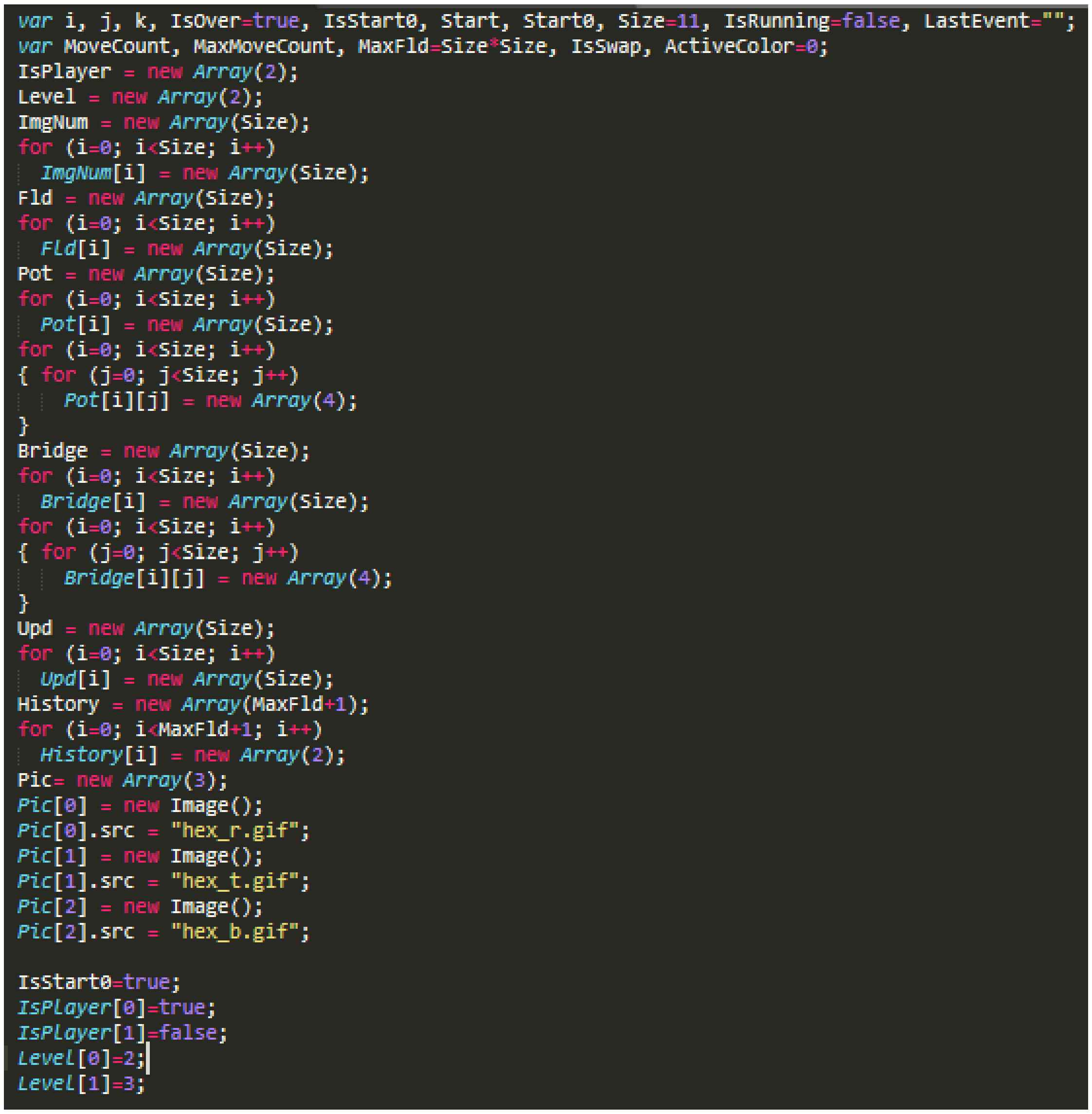
Codes utilisé pour jouer HEX

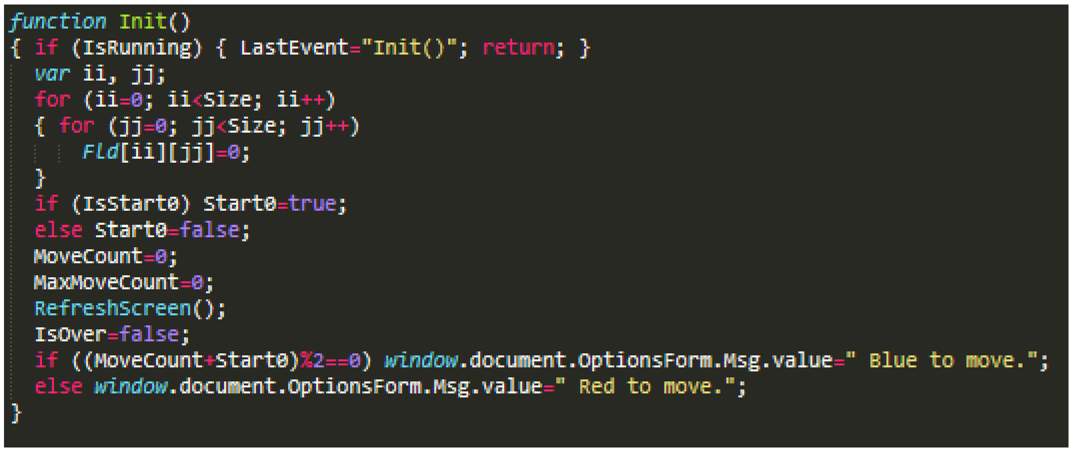
Walid BOURAYA et Youssef RKAISSI

Déclaration des variables



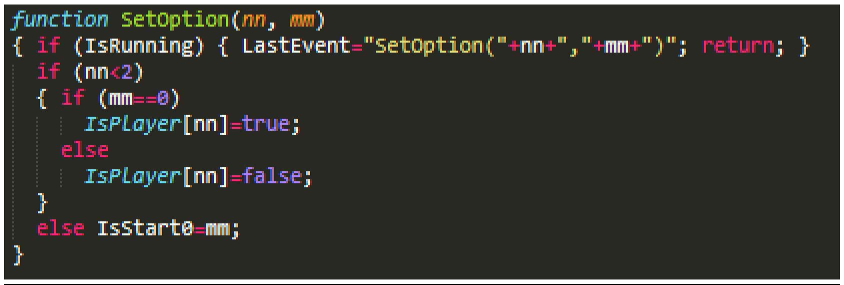
```
var i, j, k, IsOver=true, IsStart0, Start, Start0, Size=11, IsRunning=false, LastEvent="";
var MoveCount, MaxMoveCount, MaxFld=Size*Size, IsSwap, ActiveColor=0;
IsPlayer = new Array(2);
Level = new Array(2);
ImgNum = new Array(Size);
for (i=0; i<Size; i++)
  ImgNum[i] = new Array(Size);
Fld = new Array(Size);
for (i=0; i<Size; i++)
  Fld[i] = new Array(Size);
Pot = new Array(Size);
for (i=0; i<Size; i++)
  Pot[i] = new Array(Size);
for (i=0; i<Size; i++)
{ for (j=0; j<Size; j++)
    Pot[i][j] = new Array(4);
}
Bridge = new Array(Size);
for (i=0; i<Size; i++)
  Bridge[i] = new Array(Size);
for (i=0; i<Size; i++)
{ for (j=0; j<Size; j++)
    Bridge[i][j] = new Array(4);
}
Upd = new Array(Size);
for (i=0; i<Size; i++)
  Upd[i] = new Array(Size);
History = new Array(MaxFld+1);
for (i=0; i<MaxFld+1; i++)
  History[i] = new Array(2);
Pic= new Array(3);
Pic[0] = new Image();
Pic[0].src = "hex_r.gif";
Pic[1] = new Image();
Pic[1].src = "hex_t.gif";
Pic[2] = new Image();
Pic[2].src = "hex_b.gif";

IsStart0=true;
IsPlayer[0]=true;
IsPlayer[1]=false;
Level[0]=2;
Level[1]=3;
```

Fonction init pour initialiser le jeu



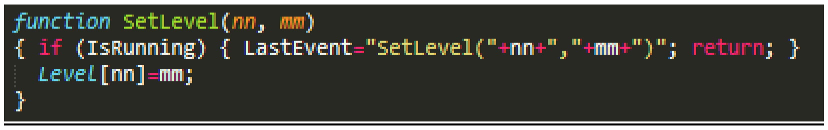
```
function Init()
{ if (IsRunning) { LastEvent="Init()"; return; }
  var ii, jj;
  for (ii=0; ii<Size; ii++)
  { for (jj=0; jj<Size; jj++)
      Fld[ii][jj]=0;
  }
  if (IsStart0) Start0=true;
  else Start0=false;
  MoveCount=0;
  MaxMoveCount=0;
  RefreshScreen();
  IsOver=false;
  if ((MoveCount+Start0)%2==0) window.document.OptionsForm.Msg.value=" Blue to move.";
  else window.document.OptionsForm.Msg.value=" Red to move.";
}
```

Fonction setoption pour choisir les types des joueurs et qui va commencer



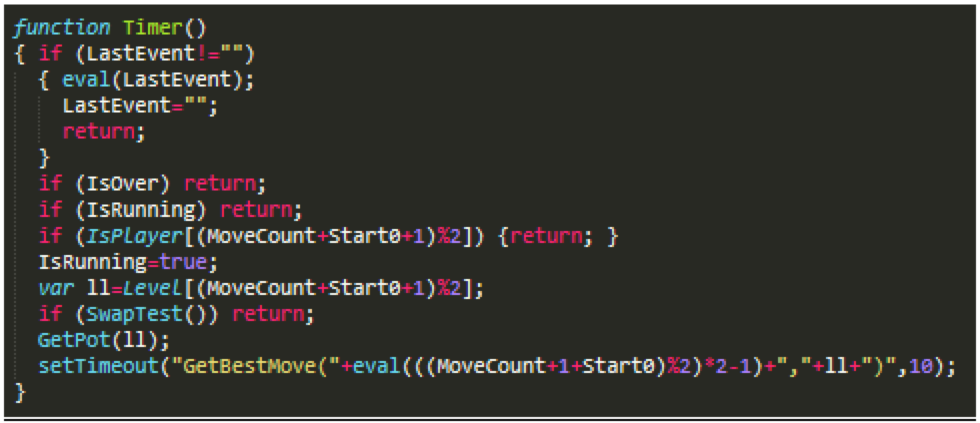
```
function SetOption(nn, mm)
{ if (IsRunning) { LastEvent="SetOption("+nn+","+mm+")"; return; }
  if (nn<2)
  { if (mm==0)
      IsPlayer[nn]=true;
    else
      IsPlayer[nn]=false;
  }
  else IsStart0=mm;
}
```

Fonction setlevel pour choisir le niveau du joueur AI



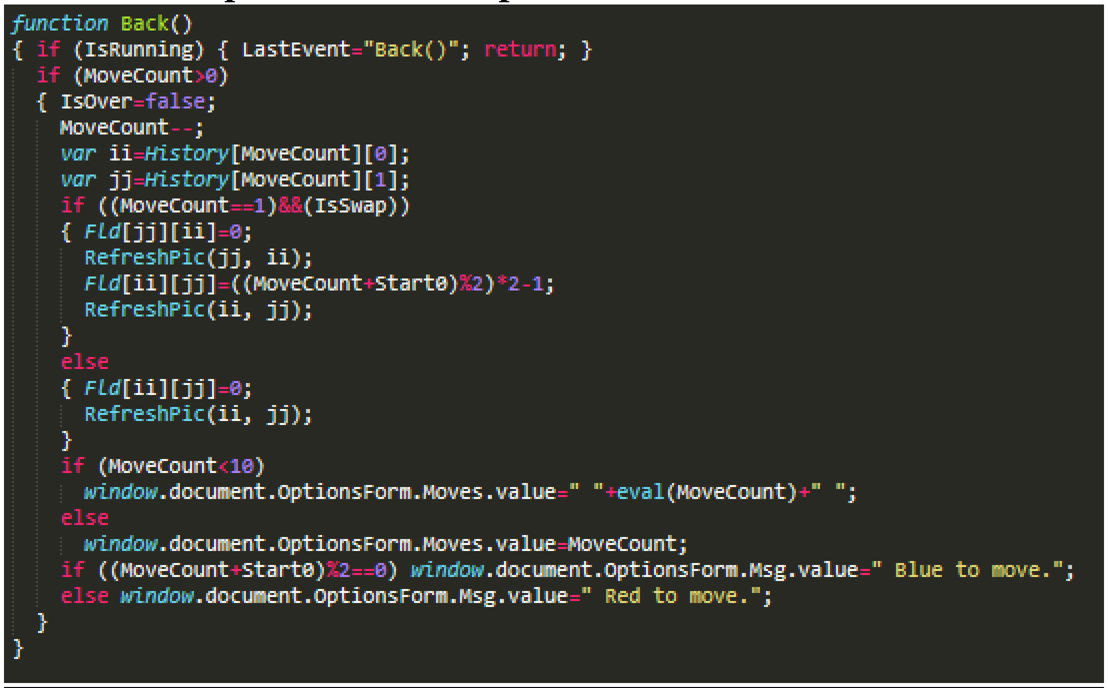
```
function SetLevel(nn, mm)
{ if (IsRunning) { LastEvent="SetLevel("+nn+","+mm+")"; return; }
  Level[nn]=mm;
}
```

Fonction timer pour initialiser le jeu



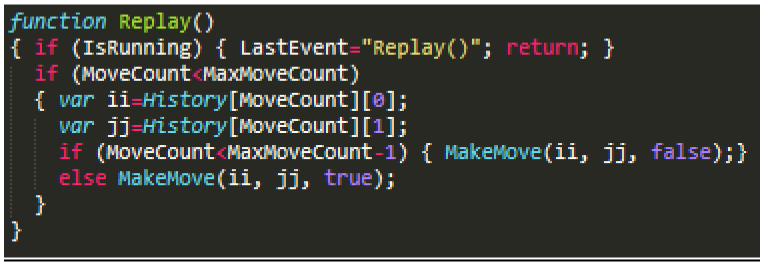
```
function Timer()
{ if (LastEvent!="")
  { eval(LastEvent);
    LastEvent="";
    return;
  }
  if (IsOver) return;
  if (IsRunning) return;
  if (IsPlayer[(MoveCount+Start0+1)%2]) {return; }
  IsRunning=true;
  var ll=Level[(MoveCount+Start0+1)%2];
  if (SwapTest()) return;
  GetPot(ll);
  setTimeout("GetBestMove("+eval(((MoveCount+1+Start0)%2)*2-1)+","+ll+")",10);
}
```

Fonction back pour revenir d'un pas



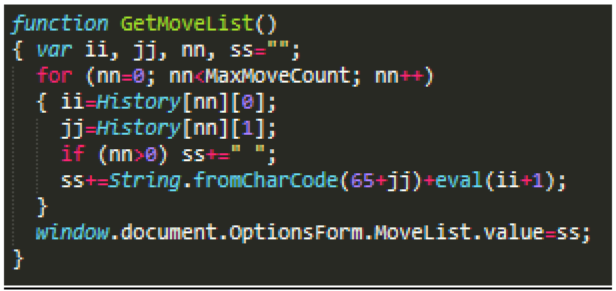
```
function Back()
{ if (IsRunning) { LastEvent="Back()"; return; }
  if (MoveCount>0)
  { IsOver=false;
    MoveCount--;
    var ii=History[MoveCount][0];
    var jj=History[MoveCount][1];
    if ((MoveCount==1)&&(IsSwap))
    { Fld[jj][ii]=0;
      RefreshPic(jj, ii);
      Fld[ii][jj]=((MoveCount+Start0)%2)*2-1;
      RefreshPic(ii, jj);
    }
    else
    { Fld[ii][jj]=0;
      RefreshPic(ii, jj);
    }
    if (MoveCount<10)
      window.document.OptionsForm.Moves.value=" "+eval(MoveCount)+" ";
    else
      window.document.OptionsForm.Moves.value=MoveCount;
    if ((MoveCount+Start0)%2==0) window.document.OptionsForm.Msg.value=" Blue to move.";
    else window.document.OptionsForm.Msg.value=" Red to move.";
  }
}
```

Fonction replay pour refaire un pas après utiliser la Fonction back



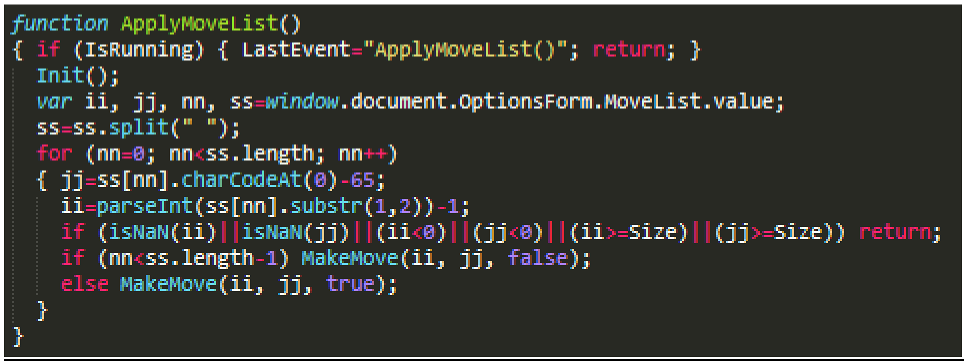
```
function Replay()
{ if (IsRunning) { LastEvent="Replay()"; return; }
  if (MoveCount<MaxMoveCount)
  { var ii=History[MoveCount][0];
    var jj=History[MoveCount][1];
    if (MoveCount<MaxMoveCount-1) { MakeMove(ii, jj, false);}
    else MakeMove(ii, jj, true);
  }
}
```

Fonction getmovelist pour avoir les mouvement faite par les joueurs



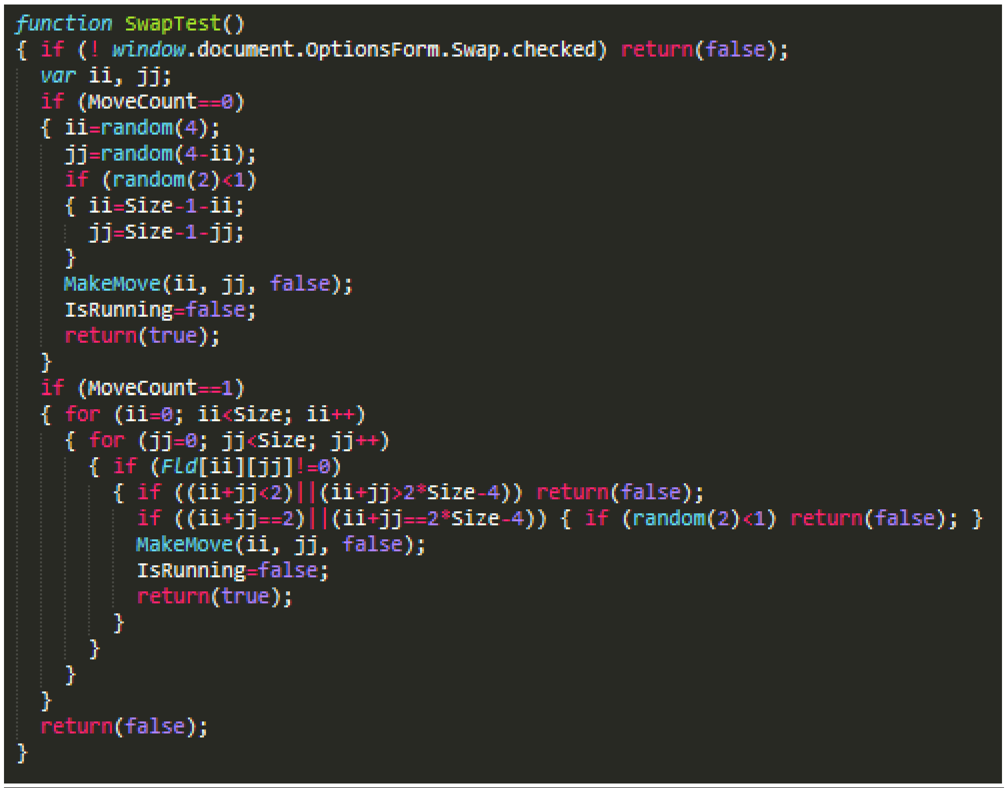
```
function GetMoveList()
{ var ii, jj, nn, ss="";
  for (nn=0; nn<MaxMoveCount; nn++)
  { ii=History[nn][0];
    jj=History[nn][1];
    if (nn>0) ss+=" ";
    ss+=String.fromCharCode(65+jj)+eval(ii+1);
  }
  window.document.OptionsForm.MoveList.value=ss;
}
```

Fonction applymovelist pour appliquer des mouvements au jeu



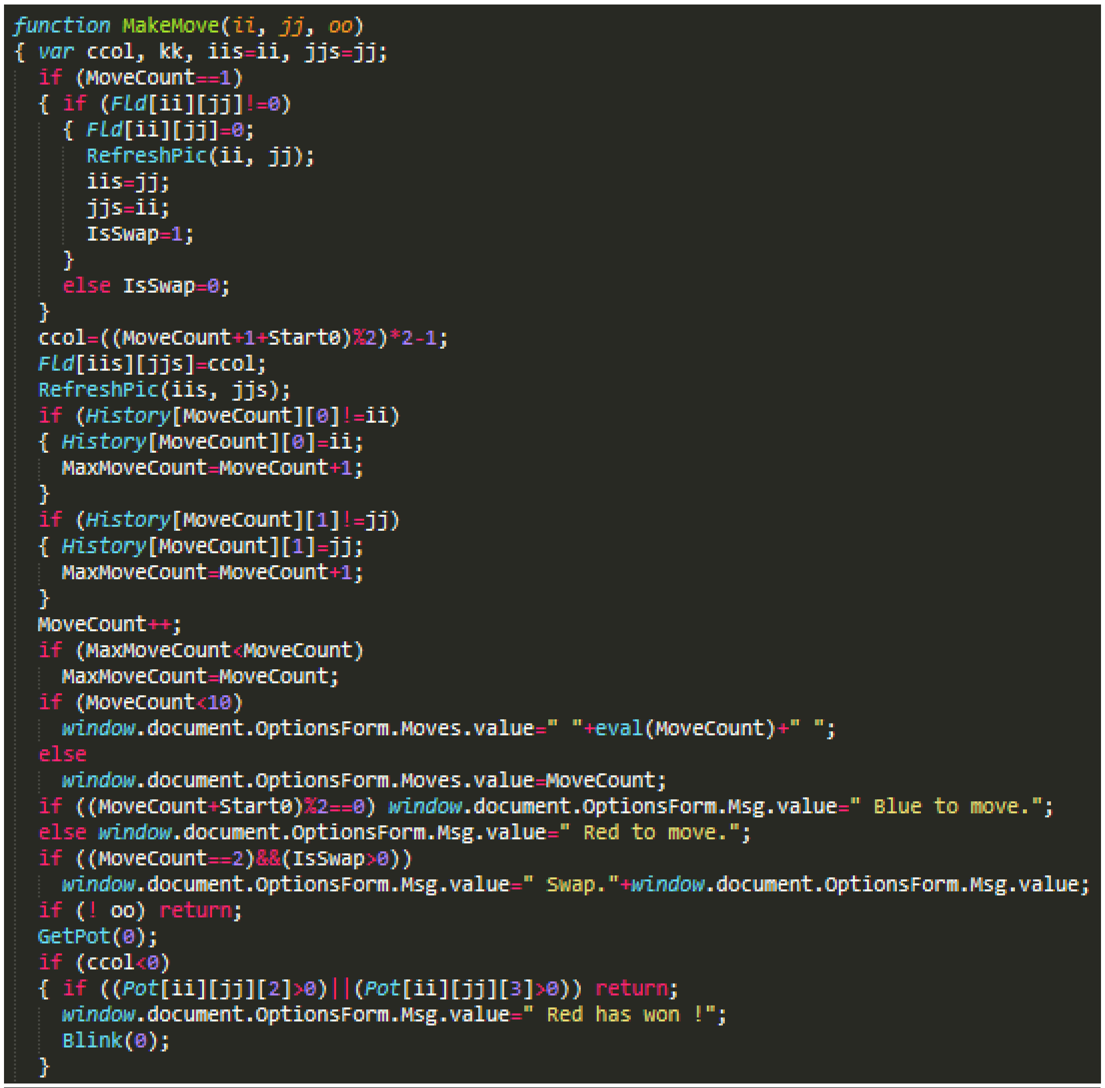
```
function ApplyMoveList()
{ if (IsRunning) { LastEvent="ApplyMoveList()"; return; }
  Init();
  var ii, jj, nn, ss=window.document.OptionsForm.MoveList.value;
  ss=ss.split(" ");
  for (nn=0; nn<ss.length; nn++)
  { jj=ss[nn].charCodeAt(0)-65;
    ii=parseInt(ss[nn].substr(1,2))-1;
    if (isNaN(ii)||isNaN(jj)||(ii<0)||(jj<0)||(ii>=Size)||(jj>=Size)) return;
    if (nn<ss.length-1) MakeMove(ii, jj, false);
    else MakeMove(ii, jj, true);
  }
}
```

Fonction swaptest pour voir si il y a un swap au début ou non

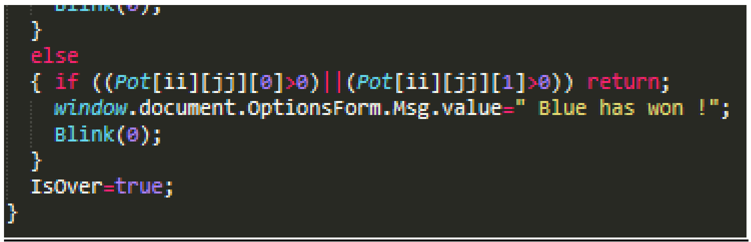


```
function SwapTest()
{ if (! window.document.OptionsForm.Swap.checked) return(false);
  var ii, jj;
  if (MoveCount==0)
  { ii=random(4);
    jj=random(4-ii);
    if (random(2)<1)
    { ii=Size-1-ii;
      jj=Size-1-jj;
    }
    MakeMove(ii, jj, false);
    IsRunning=false;
    return(true);
  }
  if (MoveCount==1)
  { for (ii=0; ii<Size; ii++)
    { for (jj=0; jj<Size; jj++)
      { if (Fld[ii][jj]!=0)
        { if ((ii+jj<2)||(ii+jj>2*Size-4)) return(false);
          if ((ii+jj==2)||(ii+jj==2*Size-4)) { if (random(2)<1) return(false); }
          MakeMove(ii, jj, false);
          IsRunning=false;
          return(true);
        }
      }
    }
  }
  return(false);
}
```

Fonction makemove pour appliquer un mouvement dans le jeu

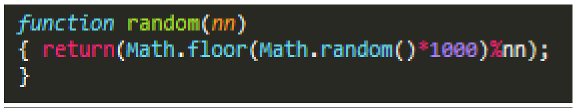


```
function MakeMove(ii, jj, oo)
{ var ccol, kk, iis=ii, jjs=jj;
  if (MoveCount==1)
  { if (Fld[ii][jj]!=0)
    { Fld[ii][jj]=0;
      RefreshPic(ii, jj);
      iis=jj;
      jjs=ii;
      IsSwap=1;
    }
    else IsSwap=0;
  }
  ccol=((MoveCount+1+Start0)%2)*2-1;
  Fld[iis][jjs]=ccol;
  RefreshPic(iis, jjs);
  if (History[MoveCount][0]!=ii)
  { History[MoveCount][0]=ii;
    MaxMoveCount=MoveCount+1;
  }
  if (History[MoveCount][1]!=jj)
  { History[MoveCount][1]=jj;
    MaxMoveCount=MoveCount+1;
  }
  MoveCount++;
  if (MaxMoveCount<MoveCount)
    MaxMoveCount=MoveCount;
  if (MoveCount<10)
    window.document.OptionsForm.Moves.value=" "+eval(MoveCount)+" ";
  else
    window.document.OptionsForm.Moves.value=MoveCount;
  if ((MoveCount+Start0)%2==0) window.document.OptionsForm.Msg.value=" Blue to move.";
  else window.document.OptionsForm.Msg.value=" Red to move.";
  if ((MoveCount==2)&&(IsSwap>0))
    window.document.OptionsForm.Msg.value=" Swap."+window.document.OptionsForm.Msg.value;
  if (! oo) return;
  GetPot(0);
  if (ccol<0)
  { if ((Pot[ii][jj][2]>0)||(Pot[ii][jj][3]>0)) return;
    window.document.OptionsForm.Msg.value=" Red has won !";
    Blink(0);
  }
```



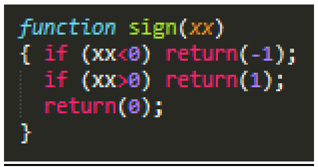
```
  }
  else
  { if ((Pot[ii][jj][0]>0)||(Pot[ii][jj][1]>0)) return;
    window.document.OptionsForm.Msg.value=" Blue has won !";
    Blink(0);
  }
  IsOver=true;
}
```

Fonction random pour avoir un numéro au Hazard



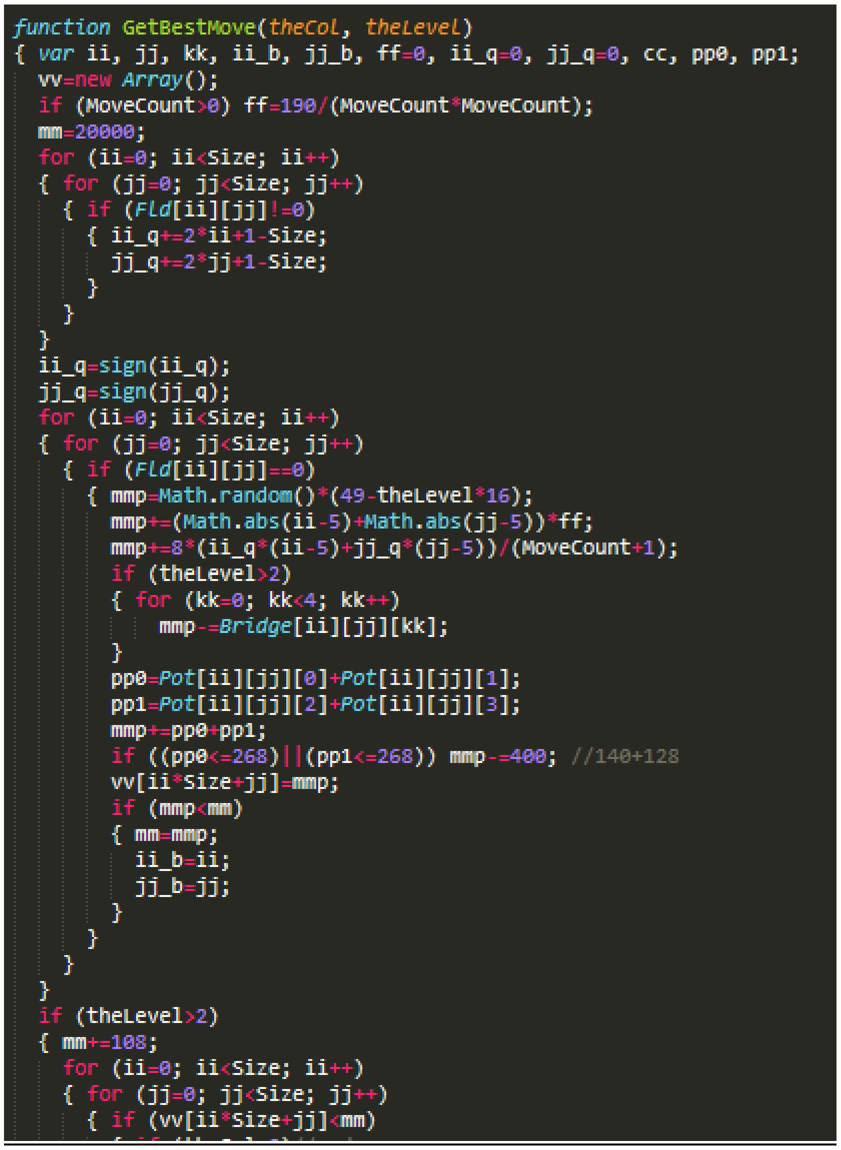
```
function random(nn)
{ return(Math.floor(Math.random()*1000)%nn);
}
```

<u>Fonction sign pour avoir le signe</u>

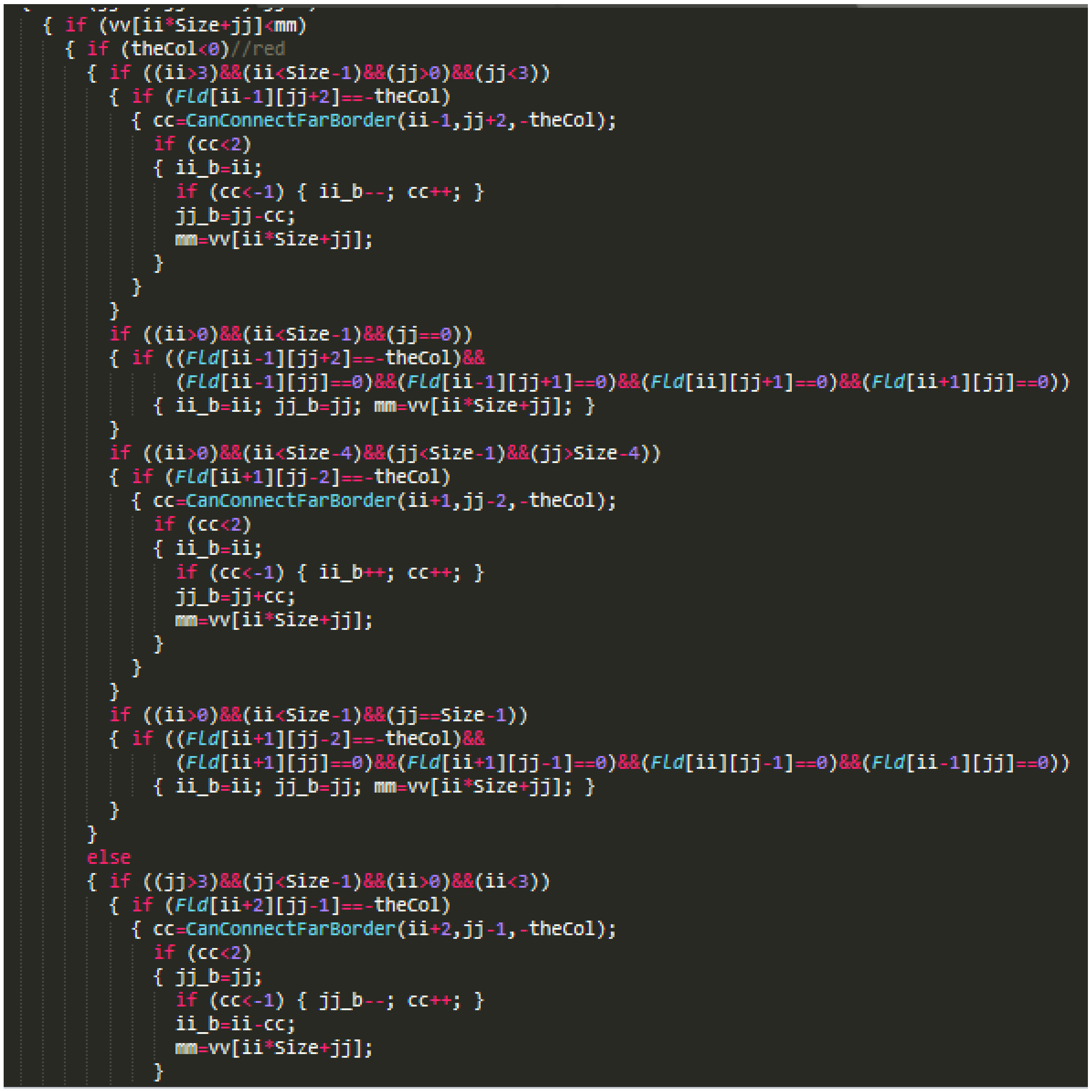


```
function sign(xx)
{ if (xx<0) return(-1);
  if (xx>0) return(1);
  return(0);
}
```

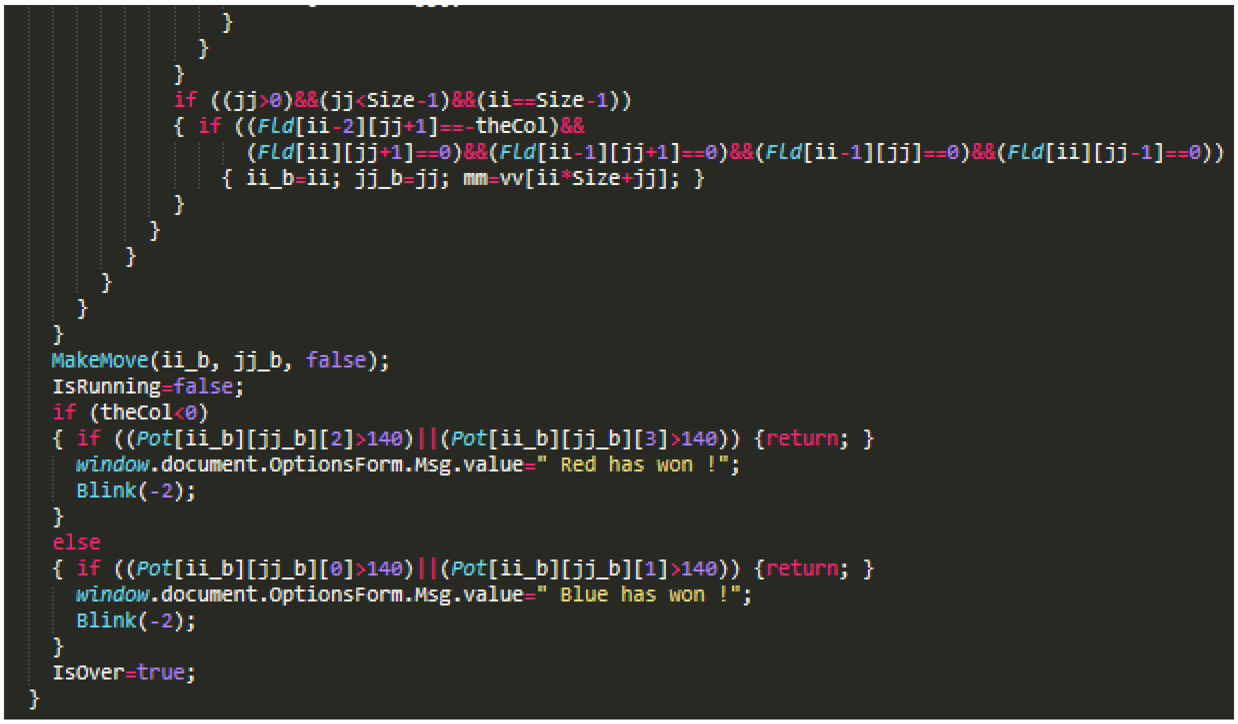
<u>Fonction getbestmove pour avoir le meilleur choix de mouvement possible selon le niveau choisi</u>



```
function GetBestMove(theCol, theLevel)
{ var ii, jj, kk, ii_b, jj_b, ff=0, ii_q=0, jj_q=0, cc, pp0, pp1;
  vv=new Array();
  if (MoveCount>0) ff=190/(MoveCount*MoveCount);
  mm=20000;
  for (ii=0; ii<Size; ii++)
  { for (jj=0; jj<Size; jj++)
    { if (Fld[ii][jj]!=0)
      { ii_q+=2*ii+1-Size;
        jj_q+=2*jj+1-Size;
      }
    }
  }
  ii_q=sign(ii_q);
  jj_q=sign(jj_q);
  for (ii=0; ii<Size; ii++)
  { for (jj=0; jj<Size; jj++)
    { if (Fld[ii][jj]==0)
      { mmp=Math.random()*(49-theLevel*16);
        mmp+=(Math.abs(ii-5)+Math.abs(jj-5))*ff;
        mmp+=8*(ii_q*(ii-5)+jj_q*(jj-5))/(MoveCount+1);
        if (theLevel>2)
        { for (kk=0; kk<4; kk++)
            mmp-=Bridge[ii][jj][kk];
        }
        pp0=Pot[ii][jj][0]+Pot[ii][jj][1];
        pp1=Pot[ii][jj][2]+Pot[ii][jj][3];
        mmp+=pp0+pp1;
        if ((pp0<=268)||(pp1<=268)) mmp-=400; //140+128
        vv[ii*Size+jj]=mmp;
        if (mmp<mm)
        { mm=mmp;
          ii_b=ii;
          jj_b=jj;
        }
      }
    }
  }
  if (theLevel>2)
  { mm+=108;
    for (ii=0; ii<Size; ii++)
    { for (jj=0; jj<Size; jj++)
      { if (vv[ii*Size+jj]<mm)
```

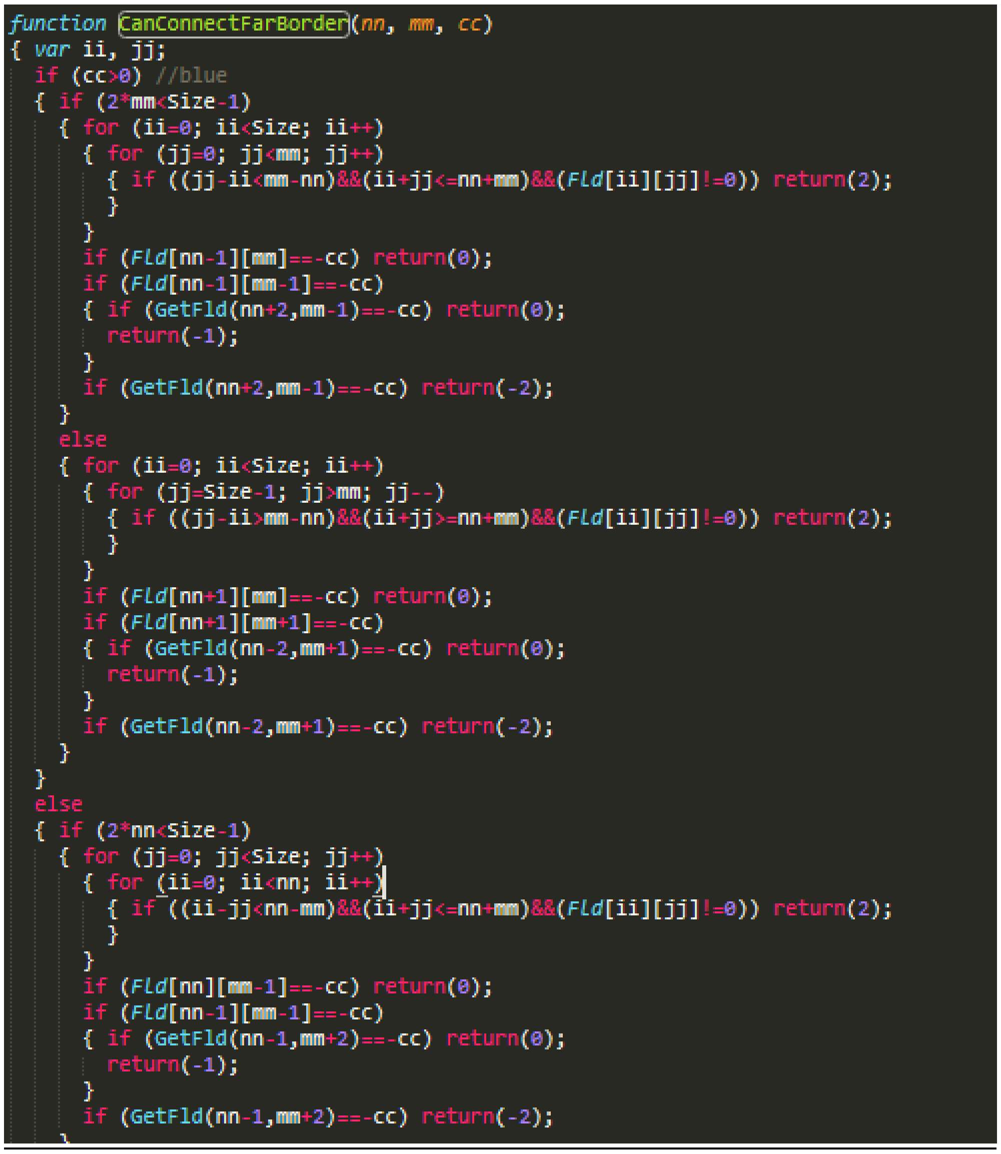


```
  { if (vv[ii*Size+jj]<mm)
    { if (theCol<0)//red
      { if ((ii>3)&&(ii<Size-1)&&(jj>0)&&(jj<3))
        { if (Fld[ii-1][jj+2]==-theCol)
          { cc=CanConnectFarBorder(ii-1,jj+2,-theCol);
            if (cc<2)
            { ii_b=ii;
              if (cc<-1) { ii_b--; cc++; }
              jj_b=jj-cc;
              mm=vv[ii*Size+jj];
            }
          }
        }
        if ((ii>0)&&(ii<Size-1)&&(jj==0))
        { if ((Fld[ii-1][jj+2]==-theCol)&&
              (Fld[ii-1][jj]==0)&&(Fld[ii-1][jj+1]==0)&&(Fld[ii][jj+1]==0)&&(Fld[ii+1][jj]==0))
            { ii_b=ii; jj_b=jj; mm=vv[ii*Size+jj]; }
        }
        if ((ii>0)&&(ii<Size-4)&&(jj<Size-1)&&(jj>Size-4))
        { if (Fld[ii+1][jj-2]==-theCol)
          { cc=CanConnectFarBorder(ii+1,jj-2,-theCol);
            if (cc<2)
            { ii_b=ii;
              if (cc<-1) { ii_b++; cc++; }
              jj_b=jj+cc;
              mm=vv[ii*Size+jj];
            }
          }
        }
        if ((ii>0)&&(ii<Size-1)&&(jj==Size-1))
        { if ((Fld[ii+1][jj-2]==-theCol)&&
              (Fld[ii+1][jj]==0)&&(Fld[ii+1][jj-1]==0)&&(Fld[ii][jj-1]==0)&&(Fld[ii-1][jj]==0))
            { ii_b=ii; jj_b=jj; mm=vv[ii*Size+jj]; }
        }
      }
      else
      { if ((jj>3)&&(jj<Size-1)&&(ii>0)&&(ii<3))
        { if (Fld[ii+2][jj-1]==-theCol)
          { cc=CanConnectFarBorder(ii+2,jj-1,-theCol);
            if (cc<2)
            { jj_b=jj;
              if (cc<-1) { jj_b--; cc++; }
              ii_b=ii-cc;
              mm=vv[ii*Size+jj];
            }
```

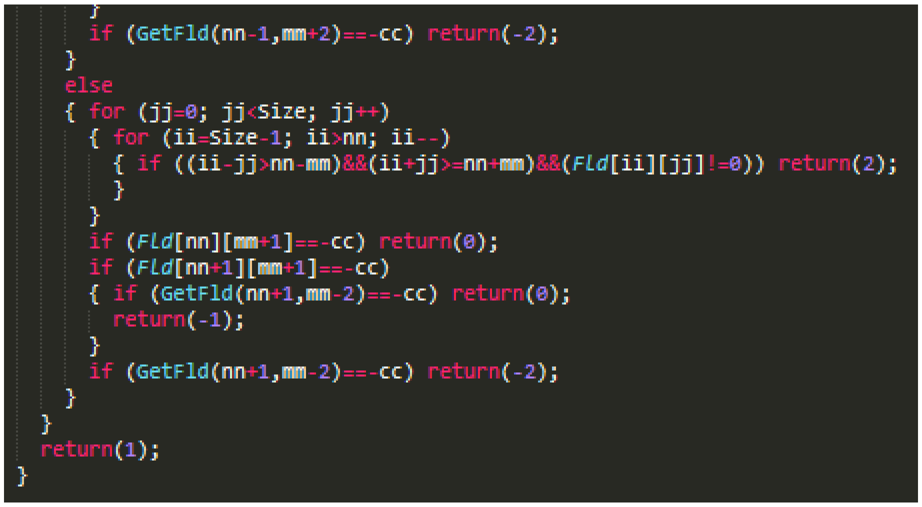


```
                }
              }
            }
            if ((jj>0)&&(jj<Size-1)&&(ii==Size-1))
            { if ((Fld[ii-2][jj+1]==-theCol)&&
                  (Fld[ii][jj+1]==0)&&(Fld[ii-1][jj+1]==0)&&(Fld[ii-1][jj]==0)&&(Fld[ii][jj-1]==0))
                { ii_b=ii; jj_b=jj; mm=vv[ii*Size+jj]; }
            }
          }
        }
      }
    }
  }
  MakeMove(ii_b, jj_b, false);
  IsRunning=false;
  if (theCol<0)
  { if ((Pot[ii_b][jj_b][2]>140)||(Pot[ii_b][jj_b][3]>140)) {return; }
    window.document.OptionsForm.Msg.value=" Red has won !";
    Blink(-2);
  }
  else
  { if ((Pot[ii_b][jj_b][0]>140)||(Pot[ii_b][jj_b][1]>140)) {return; }
    window.document.OptionsForm.Msg.value=" Blue has won !";
    Blink(-2);
  }
  IsOver=true;
}
```

Fonction canconnectfarborder permet de savoir si le mouvement choisie par le AI est possible

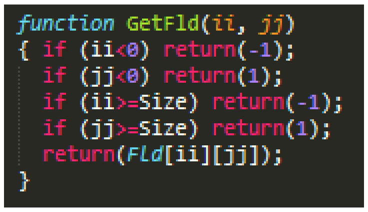


```
function CanConnectFarBorder(nn, mm, cc)
{ var ii, jj;
  if (cc>0) //blue
  { if (2*mm<Size-1)
    { for (ii=0; ii<Size; ii++)
      { for (jj=0; jj<mm; jj++)
        { if ((jj-ii<mm-nn)&&(ii+jj<=nn+mm)&&(Fld[ii][jj]!=0)) return(2);
        }
      }
      if (Fld[nn-1][mm]==-cc) return(0);
      if (Fld[nn-1][mm-1]==-cc)
      { if (GetFld(nn+2,mm-1)==-cc) return(0);
        return(-1);
      }
      if (GetFld(nn+2,mm-1)==-cc) return(-2);
    }
    else
    { for (ii=0; ii<Size; ii++)
      { for (jj=Size-1; jj>mm; jj--)
        { if ((jj-ii>mm-nn)&&(ii+jj>=nn+mm)&&(Fld[ii][jj]!=0)) return(2);
        }
      }
      if (Fld[nn+1][mm]==-cc) return(0);
      if (Fld[nn+1][mm+1]==-cc)
      { if (GetFld(nn-2,mm+1)==-cc) return(0);
        return(-1);
      }
      if (GetFld(nn-2,mm+1)==-cc) return(-2);
    }
  }
  else
  { if (2*nn<Size-1)
    { for (jj=0; jj<Size; jj++)
      { for (ii=0; ii<nn; ii++)
        { if ((ii-jj<nn-mm)&&(ii+jj<=nn+mm)&&(Fld[ii][jj]!=0)) return(2);
        }
      }
      if (Fld[nn][mm-1]==-cc) return(0);
      if (Fld[nn-1][mm-1]==-cc)
      { if (GetFld(nn-1,mm+2)==-cc) return(0);
        return(-1);
      }
      if (GetFld(nn-1,mm+2)==-cc) return(-2);
    }
```



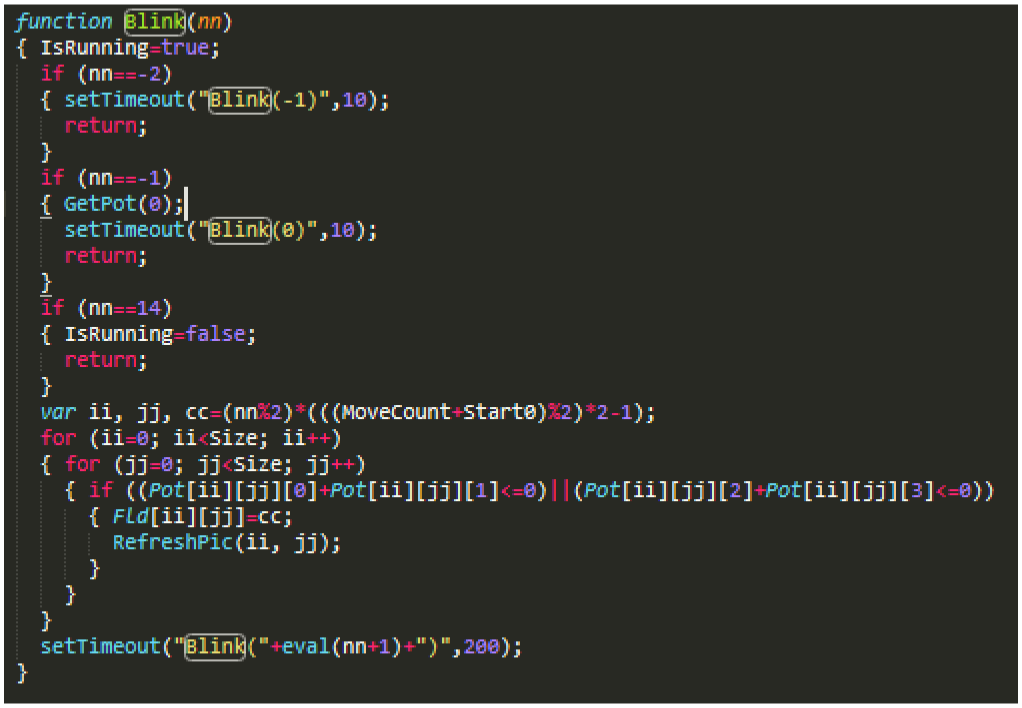
```
      }
      if (GetFld(nn-1,mm+2)==-cc) return(-2);
    }
    else
    { for (jj=0; jj<Size; jj++)
      { for (ii=Size-1; ii>nn; ii--)
        { if ((ii-jj>nn-mm)&&(ii+jj>=nn+mm)&&(Fld[ii][jj]!=0)) return(2);
        }
      }
      if (Fld[nn][mm+1]==-cc) return(0);
      if (Fld[nn+1][mm+1]==-cc)
      { if (GetFld(nn+1,mm-2)==-cc) return(0);
        return(-1);
      }
      if (GetFld(nn+1,mm-2)==-cc) return(-2);
    }
  }
  return(1);
}
```

Fonction getfld donne une position



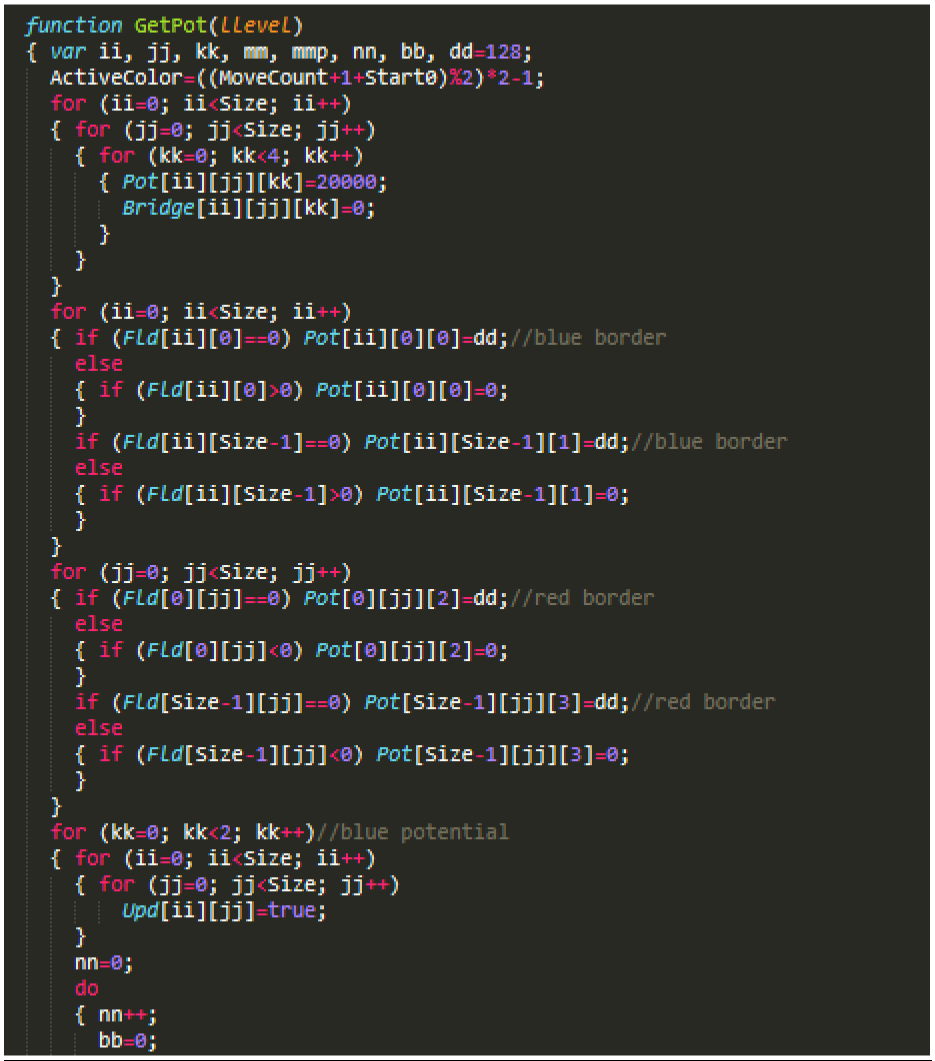
```
function GetFld(ii, jj)
{ if (ii<0) return(-1);
  if (jj<0) return(1);
  if (ii>=Size) return(-1);
  if (jj>=Size) return(1);
  return(Fld[ii][jj]);
}
```

Fonction blink ajoute un effet de flash à la fin

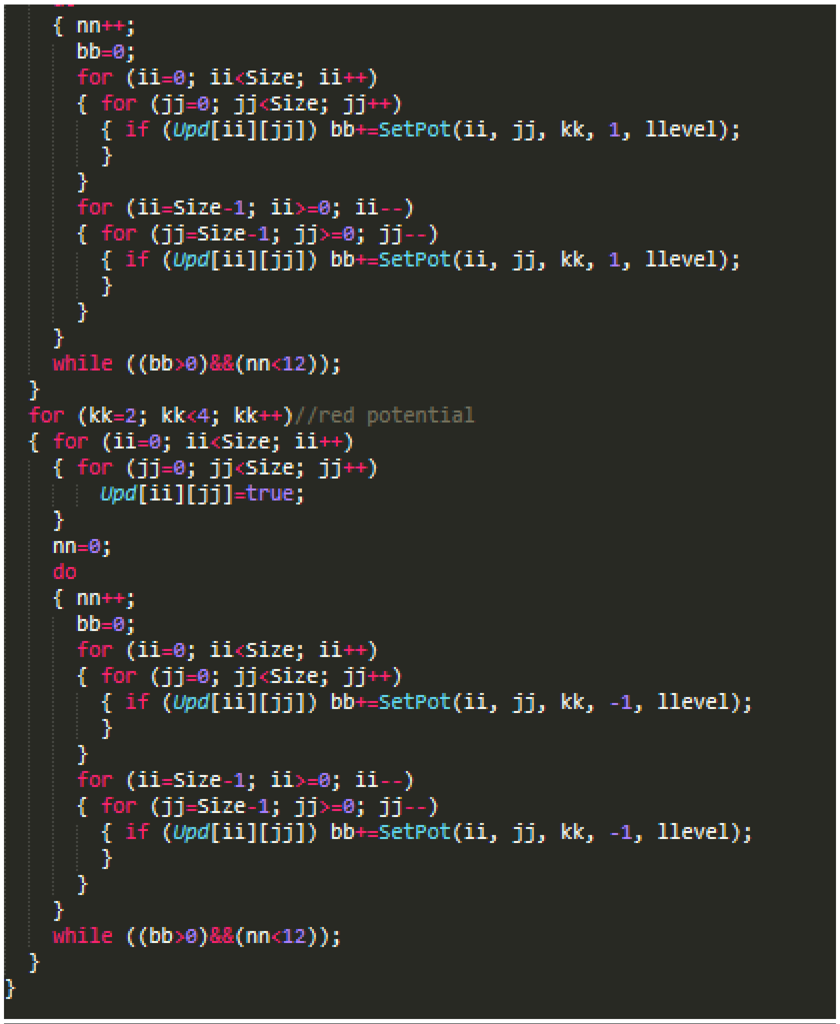


```
function Blink(nn)
{ IsRunning=true;
  if (nn==-2)
  { setTimeout("Blink(-1)",10);
    return;
  }
  if (nn==-1)
  { GetPot(0);
    setTimeout("Blink(0)",10);
    return;
  }
  if (nn==14)
  { IsRunning=false;
    return;
  }
  var ii, jj, cc=(nn%2)*(((MoveCount+Start0)%2)*2-1);
  for (ii=0; ii<Size; ii++)
  { for (jj=0; jj<Size; jj++)
    { if ((Pot[ii][jj][0]+Pot[ii][jj][1]<=0)||(Pot[ii][jj][2]+Pot[ii][jj][3]<=0))
      { Fld[ii][jj]=cc;
        RefreshPic(ii, jj);
      }
    }
  }
  setTimeout("Blink("+eval(nn+1)+")",200);
}
```

Fonction Getpot pour voir les places possible du jeu

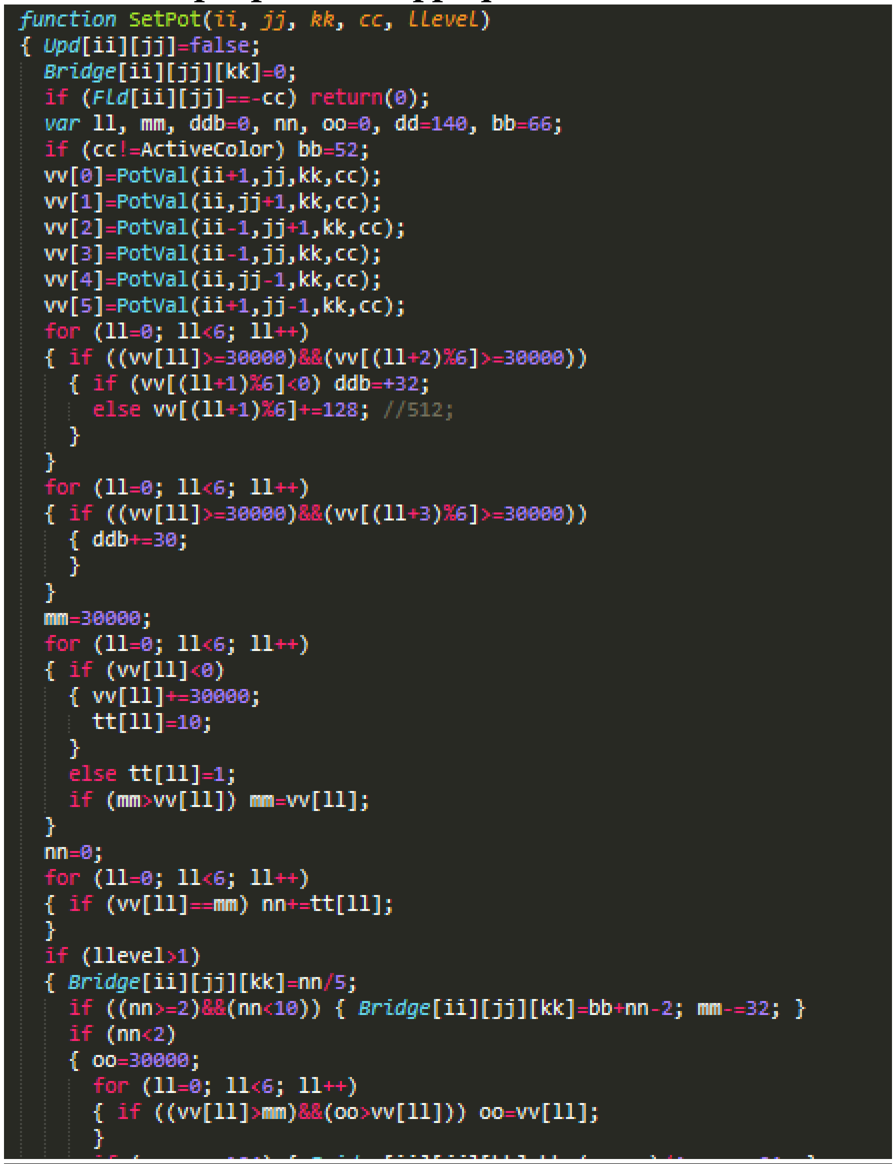


```
function GetPot(llevel)
{ var ii, jj, kk, mm, mmp, nn, bb, dd=128;
  ActiveColor=((MoveCount+1+Start0)%2)*2-1;
  for (ii=0; ii<Size; ii++)
  { for (jj=0; jj<Size; jj++)
    { for (kk=0; kk<4; kk++)
      { Pot[ii][jj][kk]=20000;
        Bridge[ii][jj][kk]=0;
      }
    }
  }
  for (ii=0; ii<Size; ii++)
  { if (Fld[ii][0]==0) Pot[ii][0][0]=dd;//blue border
    else
    { if (Fld[ii][0]>0) Pot[ii][0][0]=0;
    }
    if (Fld[ii][Size-1]==0) Pot[ii][Size-1][1]=dd;//blue border
    else
    { if (Fld[ii][Size-1]>0) Pot[ii][Size-1][1]=0;
    }
  }
  for (jj=0; jj<Size; jj++)
  { if (Fld[0][jj]==0) Pot[0][jj][2]=dd;//red border
    else
    { if (Fld[0][jj]<0) Pot[0][jj][2]=0;
    }
    if (Fld[Size-1][jj]==0) Pot[Size-1][jj][3]=dd;//red border
    else
    { if (Fld[Size-1][jj]<0) Pot[Size-1][jj][3]=0;
    }
  }
  for (kk=0; kk<2; kk++)//blue potential
  { for (ii=0; ii<Size; ii++)
    { for (jj=0; jj<Size; jj++)
        Upd[ii][jj]=true;
    }
    nn=0;
    do
    { nn++;
      bb=0;
```

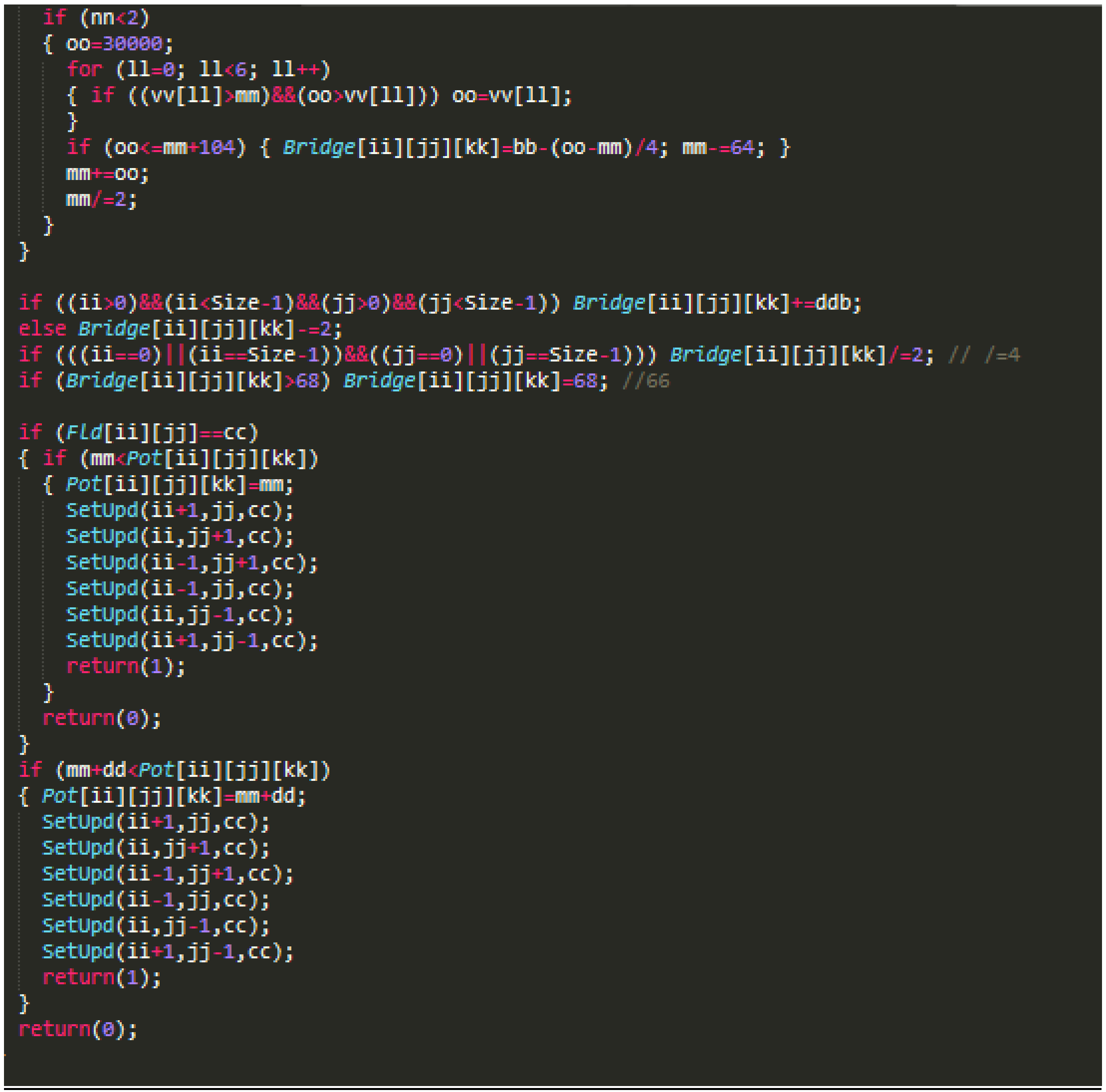


```
    { nn++;
      bb=0;
      for (ii=0; ii<Size; ii++)
      { for (jj=0; jj<Size; jj++)
        { if (Upd[ii][jj]) bb+=SetPot(ii, jj, kk, 1, llevel);
        }
      }
      for (ii=Size-1; ii>=0; ii--)
      { for (jj=Size-1; jj>=0; jj--)
        { if (Upd[ii][jj]) bb+=SetPot(ii, jj, kk, 1, llevel);
        }
      }
    }
    while ((bb>0)&&(nn<12));
  }
  for (kk=2; kk<4; kk++)//red potential
  { for (ii=0; ii<Size; ii++)
    { for (jj=0; jj<Size; jj++)
        Upd[ii][jj]=true;
    }
    nn=0;
    do
    { nn++;
      bb=0;
      for (ii=0; ii<Size; ii++)
      { for (jj=0; jj<Size; jj++)
        { if (Upd[ii][jj]) bb+=SetPot(ii, jj, kk, -1, llevel);
        }
      }
      for (ii=Size-1; ii>=0; ii--)
      { for (jj=Size-1; jj>=0; jj--)
        { if (Upd[ii][jj]) bb+=SetPot(ii, jj, kk, -1, llevel);
        }
      }
    }
    while ((bb>0)&&(nn<12));
  }
}
```

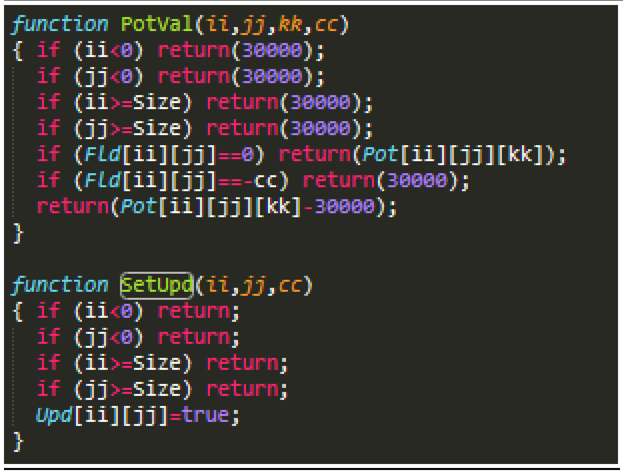
Fonction Setpot permet d'appliquer des slots vide



```
function SetPot(ii, jj, kk, cc, llevel)
{ Upd[ii][jj]=false;
  Bridge[ii][jj][kk]=0;
  if (Fld[ii][jj]==-cc) return(0);
  var ll, mm, ddb=0, nn, oo=0, dd=140, bb=66;
  if (cc!=ActiveColor) bb=52;
  vv[0]=PotVal(ii+1,jj,kk,cc);
  vv[1]=PotVal(ii,jj+1,kk,cc);
  vv[2]=PotVal(ii-1,jj+1,kk,cc);
  vv[3]=PotVal(ii-1,jj,kk,cc);
  vv[4]=PotVal(ii,jj-1,kk,cc);
  vv[5]=PotVal(ii+1,jj-1,kk,cc);
  for (ll=0; ll<6; ll++)
  { if ((vv[ll]>=30000)&&(vv[(ll+2)%6]>=30000))
    { if (vv[(ll+1)%6]<0) ddb=+32;
      else vv[(ll+1)%6]+=128; //512;
    }
  }
  for (ll=0; ll<6; ll++)
  { if ((vv[ll]>=30000)&&(vv[(ll+3)%6]>=30000))
    { ddb+=30;
    }
  }
  mm=30000;
  for (ll=0; ll<6; ll++)
  { if (vv[ll]<0)
    { vv[ll]+=30000;
      tt[ll]=10;
    }
    else tt[ll]=1;
    if (mm>vv[ll]) mm=vv[ll];
  }
  nn=0;
  for (ll=0; ll<6; ll++)
  { if (vv[ll]==mm) nn+=tt[ll];
  }
  if (llevel>1)
  { Bridge[ii][jj][kk]=nn/5;
    if ((nn>=2)&&(nn<10)) { Bridge[ii][jj][kk]=bb+nn-2; mm-=32; }
    if (nn<2)
    { oo=30000;
      for (ll=0; ll<6; ll++)
      { if ((vv[ll]>mm)&&(oo>vv[ll])) oo=vv[ll];
      }
```
]

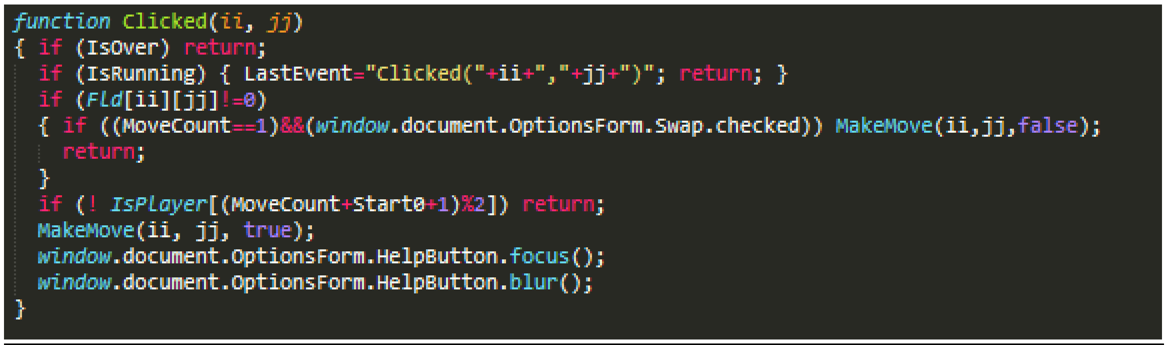


```
  if (nn<2)
  { oo=30000;
    for (ll=0; ll<6; ll++)
    { if ((vv[ll]>mm)&&(oo>vv[ll])) oo=vv[ll];
    }
    if (oo<=mm+104) { Bridge[ii][jj][kk]=bb-(oo-mm)/4; mm-=64; }
    mm+=oo;
    mm/=2;
  }
}

if ((ii>0)&&(ii<Size-1)&&(jj>0)&&(jj<Size-1)) Bridge[ii][jj][kk]+=ddb;
else Bridge[ii][jj][kk]-=2;
if (((ii==0)||(ii==Size-1))&&((jj==0)||(jj==Size-1))) Bridge[ii][jj][kk]/=2; // /=4
if (Bridge[ii][jj][kk]>68) Bridge[ii][jj][kk]=68; //66

if (Fld[ii][jj]==cc)
{ if (mm<Pot[ii][jj][kk])
  { Pot[ii][jj][kk]=mm;
    SetUpd(ii+1,jj,cc);
    SetUpd(ii,jj+1,cc);
    SetUpd(ii-1,jj+1,cc);
    SetUpd(ii-1,jj,cc);
    SetUpd(ii,jj-1,cc);
    SetUpd(ii+1,jj-1,cc);
    return(1);
  }
  return(0);
}
if (mm+dd<Pot[ii][jj][kk])
{ Pot[ii][jj][kk]=mm+dd;
  SetUpd(ii+1,jj,cc);
  SetUpd(ii,jj+1,cc);
  SetUpd(ii-1,jj+1,cc);
  SetUpd(ii-1,jj,cc);
  SetUpd(ii,jj-1,cc);
  SetUpd(ii+1,jj-1,cc);
  return(1);
}
return(0);
```



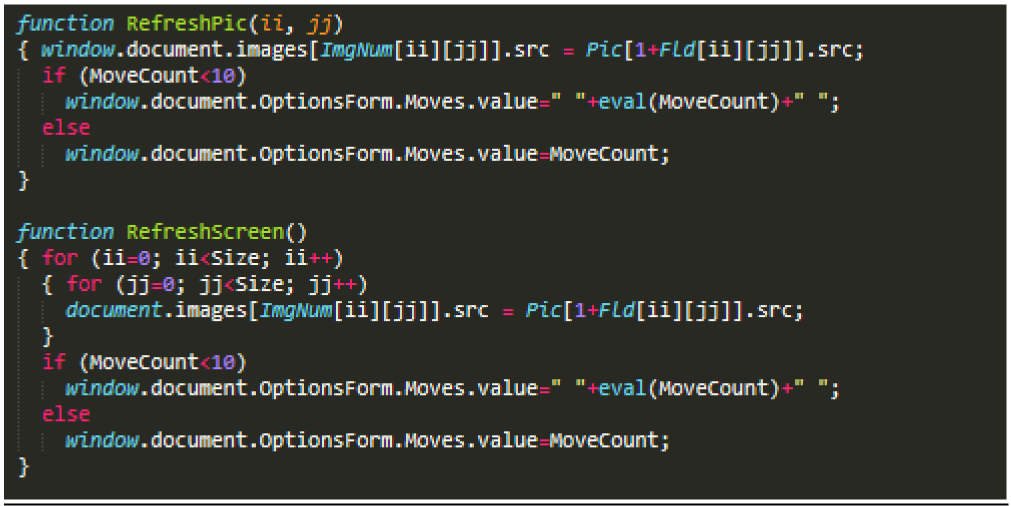
```
function PotVal(ii,jj,kk,cc)
{ if (ii<0) return(30000);
  if (jj<0) return(30000);
  if (ii>=Size) return(30000);
  if (jj>=Size) return(30000);
  if (Fld[ii][jj]==0) return(Pot[ii][jj][kk]);
  if (Fld[ii][jj]==-cc) return(30000);
  return(Pot[ii][jj][kk]-30000);
}

function SetUpd(ii,jj,cc)
{ if (ii<0) return;
  if (jj<0) return;
  if (ii>=Size) return;
  if (jj>=Size) return;
  Upd[ii][jj]=true;
}
```

Fonction clicked qui applique le choix du joueur



```
function Clicked(ii, jj)
{ if (IsOver) return;
  if (IsRunning) { LastEvent="Clicked("+ii+","+jj+")"; return; }
  if (Fld[ii][jj]!=0)
  { if ((MoveCount==1)&&(window.document.OptionsForm.Swap.checked)) MakeMove(ii,jj,false);
    return;
  }
  if (! IsPlayer[(MoveCount+Start0+1)%2]) return;
  MakeMove(ii, jj, true);
  window.document.OptionsForm.HelpButton.focus();
  window.document.OptionsForm.HelpButton.blur();
}
```

Fonctions refreshpic et refreshscreen



```
function RefreshPic(ii, jj)
{ window.document.images[ImgNum[ii][jj]].src = Pic[1+Fld[ii][jj]].src;
  if (MoveCount<10)
    window.document.OptionsForm.Moves.value=" "+eval(MoveCount)+" ";
  else
    window.document.OptionsForm.Moves.value=MoveCount;
}

function RefreshScreen()
{ for (ii=0; ii<Size; ii++)
  { for (jj=0; jj<Size; jj++)
    document.images[ImgNum[ii][jj]].src = Pic[1+Fld[ii][jj]].src;
  }
  if (MoveCount<10)
    window.document.OptionsForm.Moves.value=" "+eval(MoveCount)+" ";
  else
    window.document.OptionsForm.Moves.value=MoveCount;
}
```