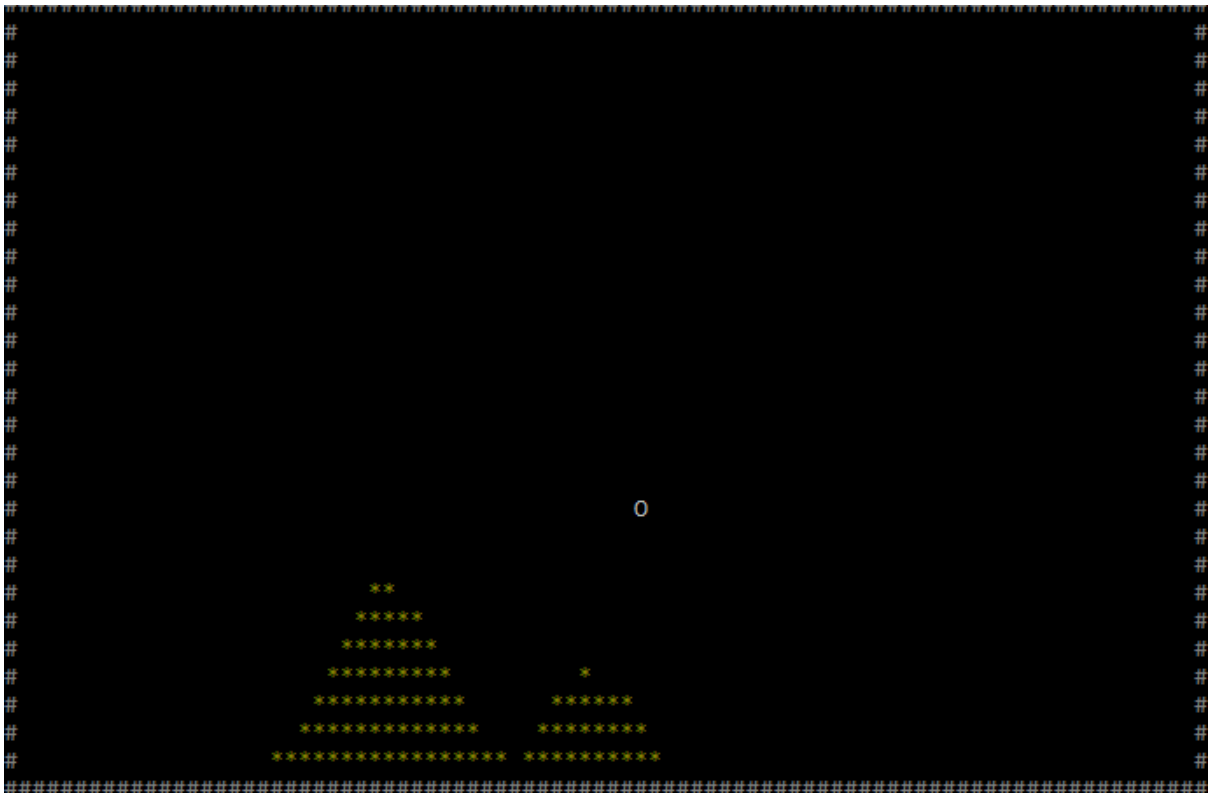
Documentație program “Nisip”.

De Novetschi Vlad



Acest program este un tip de simulare a particulelor. Acesta permit utilizatorului să plaseze particule de diferite elemente (solide, lichide). Particulele pot interacționa cu alte particule în diverse moduri și pot fi afectate de gravitație.

Taste utilizate –

Deplasare cursor – săgeți

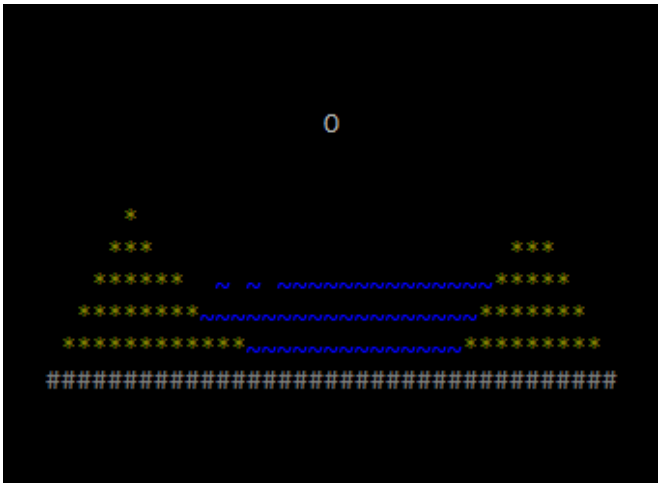
Ștergere particule – tasta ctrl dreapta

Plasare particule – nisip – numpad 1;

Plasare particule – apă – numpad 2;

Plasare pereți – numpad 0

Afișare



Informația despre particule este stocată în matricea type[poziţieY][pozitieX];

Type[x][y]= -1 sunt pereții;

Type[x][y]= 1 sunt particulele de nisip;

Type[x][y]= 2 sunt particulele lichide;

Cu Type[x][y]= 0 fiind notate spatiile libere

Afișarea in consolă se face prin utilizarea functiilor din stdlib.h, după fiecare cadru poziția cursorului din consolă este setată la 0,0 (coltul stanga sus al ecranului) prin instructiuniile:

HANDLE handle=GetStdHandle(STD_OUTPUT_HANDLE);

COORD zerozero={0,0};

SetConsoleCursorPosition(handle, zerozero); // inainte de randarea fiecarui cadru

Functia **display()** construieste un sir de caractere "**screen**" in functie de tipul de particule prezente pe ecran la o anumiă coordonată (simbolurile *, ~, # pentru tipurile de particule) , urmând ca la fiecare schimbare de culoare sirul de caractere sa fie afisat in intregime cu culoare respectiva prin funcția **display2(culoare)**

```
void display2(int color)
{
    SetConsoleTextAttribute(handle, color);
    cout<<screen;
    screen.clear();
}
```

Dacă coordonata **b** este egala cu latimea ecranului **lj** se introduce in sir caracterul '**\n'** pentru linie nouă dar nu se efectueaza incă afisarea sirului.

Cand coordonata **a** este egala cu inaltimea ecranului **li** se afișează ultiul sir de caractere construit.

```
void display()
{
     screen.clear();
     int clr=0;
     for (int a=0; a<=li+1;a++)
     {
          for (int b=0; b<=lj+1;b++)
          {

     if (b==crsx&&a==crsy)
     {
          if (7!=clr)display2(clr),clr=7;
          screen+="O";
     }
    else if (type[a][b]==-1)
     {
         if (8!=clr)display2(clr),clr=8;
         screen+="#";
    }
    else if (type[a][b]==1)
     {
         if (6!=clr)display2(clr),clr=6;
         screen+="*";
    }
     else if (type[a][b]==2)
     {
          if (9!=clr)display2(clr),clr=9;
     screen+="~";
       // screen+=wb[a][b]+('0'-0);
     }


          else if (type[a][b]==0)screen+=" ";

          }
     screen+="\n";
     }

     display2(clr);
}
```

Am utilizat afisarea prin siruri de caractere in locul afisarii individuale a fiecarui character pentru a accelera viteza cu care se pot afiseaza cadrele :

Astfel se pot obtine 140(min)-350(max) cadre pe secunda pe windows 10 intel i5-7500

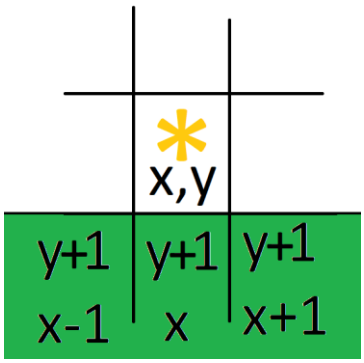
in practica insa aceasta rată de cadre pe secundă trebuie intarziată prin comanda **sleep()** pentru ca miscarile particulelor sa poata fi percepute pe ecrane cu rata de refresh de 60Hz.

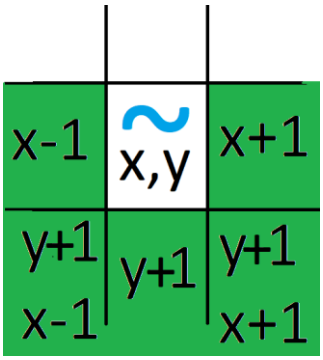
Simulare

Simularea particulelor se realizeaza in cadrul functiei **gravity ()**

La inceput se verifica ce particule trebuiesc actualizate (pot cadea) si pozitia acestora se marcheaza cu (1 pentru particule de nisip, 2 pentru lichide) in matricea **up[][];**

Particulele sunt actualizate daca exista celule valabile (goale) sub sau in lateralul lor:

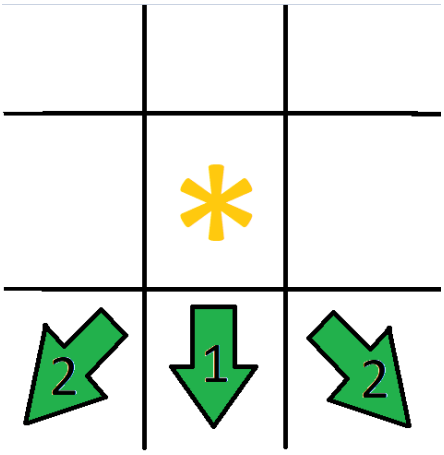


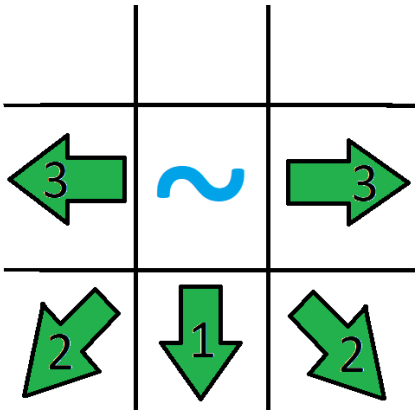


Pentru particulele de nisip sunt considerate celule valabile si spatiile acoperite de apă, (intrucat nisipul are densitatea mai mare decat apa si s-ar scufunda)

```
void gravity()
{for (int a=1; a<=li;a++) //update
    {
        for (int b=1; b<=lj;b++)
        {
            if (type[a][b]==1) // nisip
            {
                if (type[a+1][b]%2==0||type[a+1][b-1]%2==0||type[a+1][b+1]%2==0)up[a][b]=1;
            }

            if (type[a][b]==2) // apa
            {
                if (type[a+1][b]==0||type[a+1][b-1]==0||type[a+1][b+1]==0||type[a][b-1]==0||type[a][b+1]==0)up[a][b]=2;
            }
        }
    }
```

Dupa generarea matricei pentru actualizare **up[][]** se va interschimba particula actualizata cu cea de sub ea (daca are densitatea mai mica decat aceasta), in caz contrar se va alege aleatoriu o celula din lateral folosind functia **rnd(int a, int b){return rand()%b+a);**

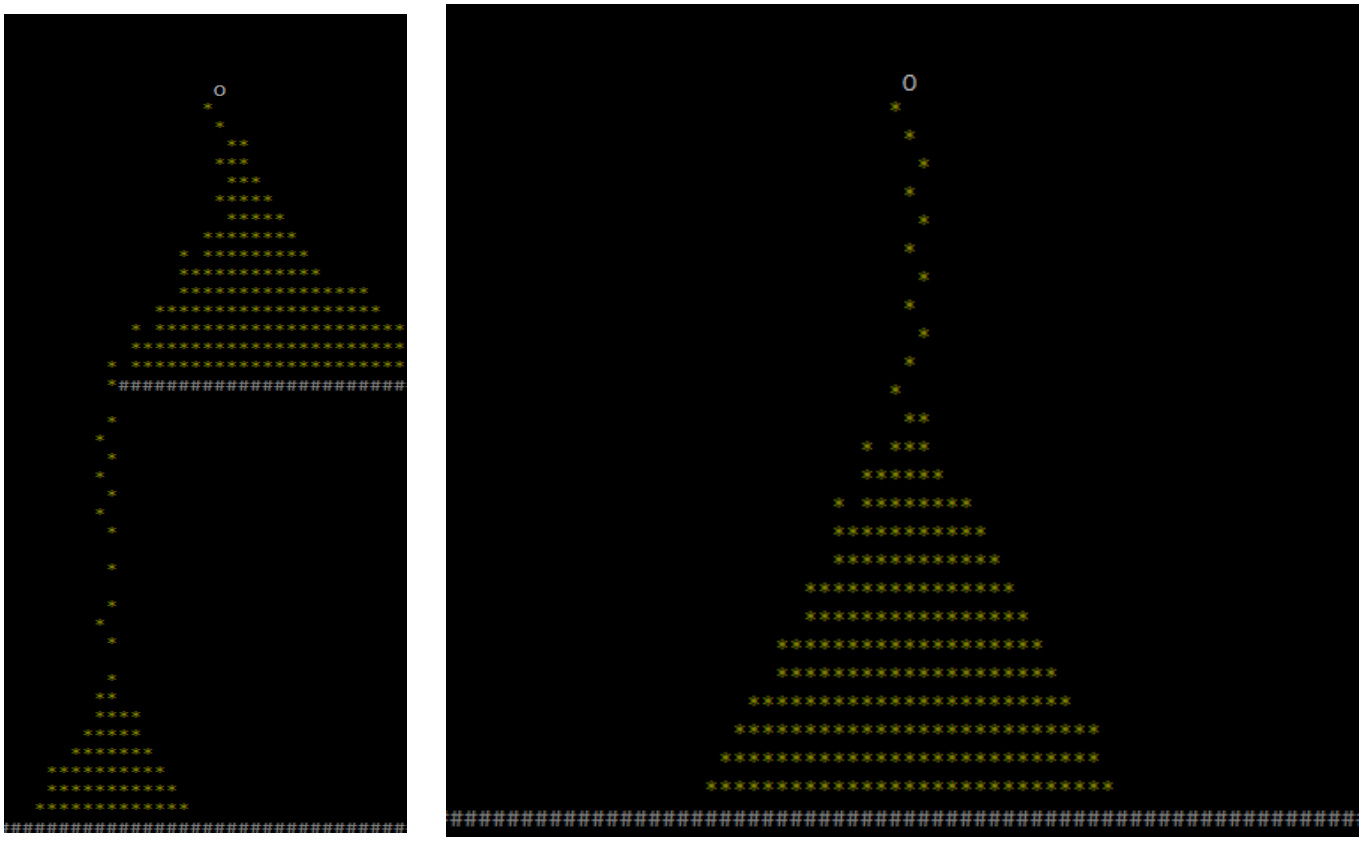


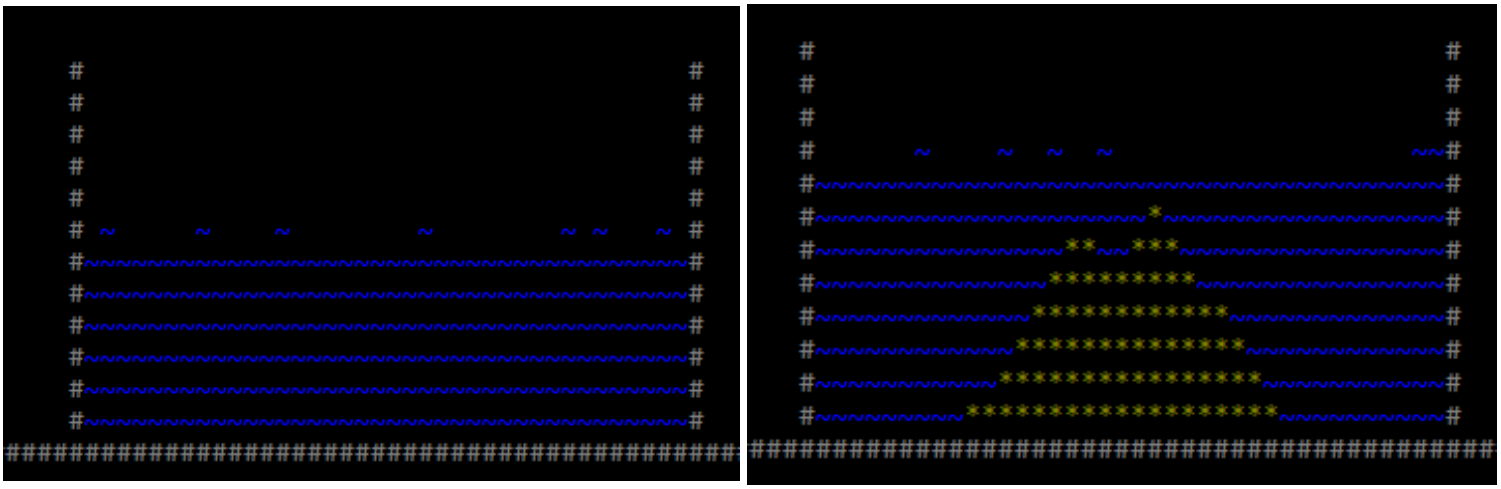



```
for (int a=1; a<=li;a++) //replace
{
    for (int b=1; b<=lj;b++)
    {
        if (up[a][b]==1) // sand
        {
            if (type[a+1][b]%2==0)swap(type[a+1][b],type[a][b]);
                else if (type[a+1][b-1]%2==0||type[a+1][b+1]%2==0)
            {
                if (rnd(0,2)==1&&type[a+1][b-1]%2==0||type[a+1][b+1]%2!=0)swap(type[a+1][b-1],type[a][b]);
                    else swap(type[a+1][b+1],type[a][b]);
            }
        }

        if (up[a][b]==2) // water
        {
            if (type[a+1][b]==0)swap(type[a+1][b],type[a][b]);

            else if (type[a+1][b-1]==0||type[a+1][b+1]==0)
            {
                if (rnd(0,2)==1&&type[a+1][b-1]==0||type[a+1][b+1]!=0)swap(type[a+1][b-1],type[a][b]);
                    else swap(type[a+1][b+1],type[a][b]);
            }

            else if (type[a][b-1]==0||type[a][b+1]==0)
            {
                if (rnd(0,2)==1&&type[a][b-1]==0||type[a][b+1]!=0)swap(type[a][b-1],type[a][b]);
                    else swap(type[a][b+1],type[a][b]);
            }

        }
        up[a][b]=0;
    }
}
```

Utilizând aceste doua etape se pot obține fenomene precum:

Formarea grămezilor de nisip

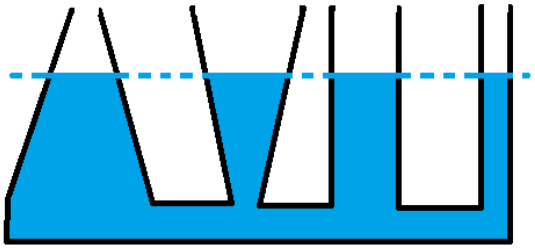




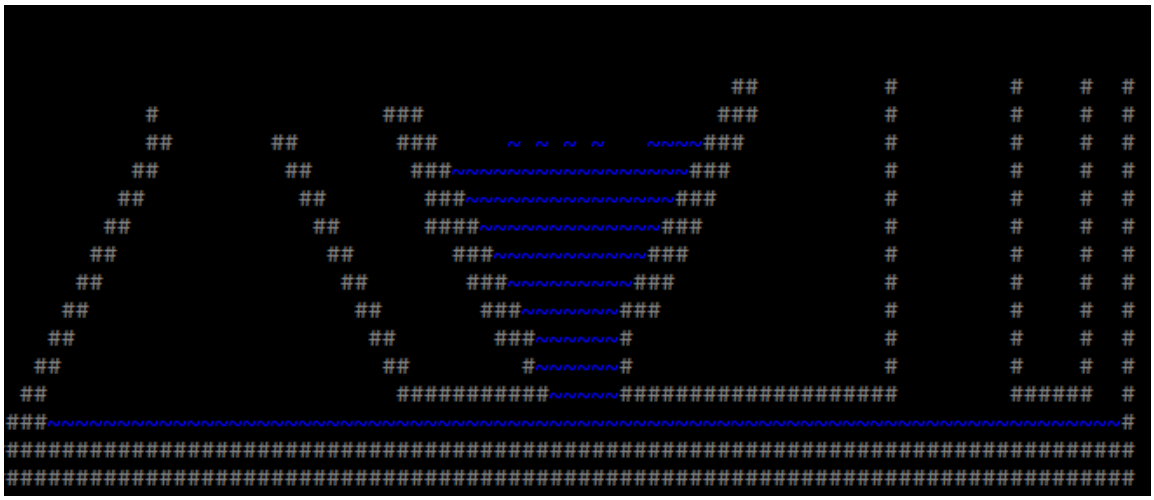
Interacțiunea intre particulele de nisip si de apă.

Simularea Presiunii

Folosind doar instructiunile de mai sus însă putem observa lipsa unui fenomen prezent în natură: egalizarea nivelului unui fluid dintr-un recipient datorată presiunii egale la adâncimi egale indiferent de forma recipientului



Principiul vaselor comunicante

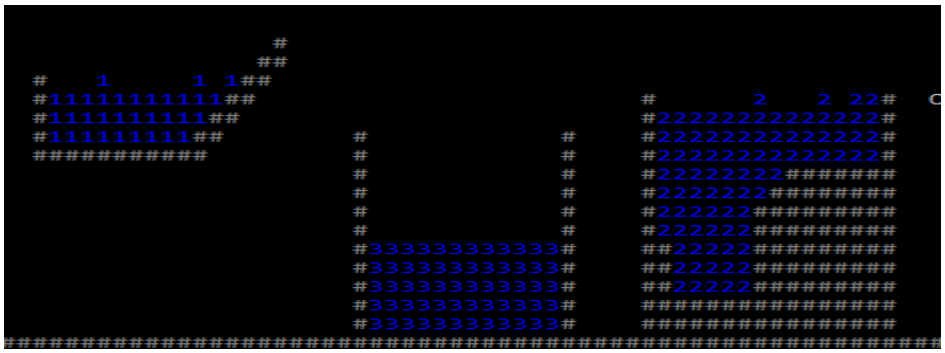


Simulare incorectă

Pentru a rezolva aceasta problema am introdus urmatoarea reozolvare:

Prima etapă este împățrirea volumelor de apă

– fiecărui volum de apă ii se asociază un număr, iar fiecare particulă de apă dintr-un anumit volum va avea aceeiași valoare in matricea **wb[x][y]**, această asociere este realizată utilizând un algoritm iterativ de umplere;



```
    int wk=1;
    int wk2=1;
    int wk3=1;

while (wk2==1)
{
        wk2=0;
        wk=1;
    while (wk==1)
    {
            wk=0;
    for (int al=1; al<=li;al++)
        for (int bl=1; bl<=lj;bl++)
        {
            if (type[al][bl]==2)
                {
            if (wb[al][bl]==0 && wk3==1)wb[al][bl]=wen,wk3=0;
            if (wb[al][bl]==wen)
            {
                for (i=-1;i<=1;i++)
                    for (j=-1;j<=1;j++)
                    {
                    if (type[al+i][bl+j]==2 && wb[al+i][bl+j]!=wen) wb[al+i][bl+j]=wen,wk=1;
                    }
            }
            if (wb[al][bl]==0)wk2=1;
                }

        }
    }
    wen++;
    wk3=1;
}
```

Apoi Pentru fiecare volum de apă se gasește punctul cel mai înalt (cu presiunea cea mai mică)

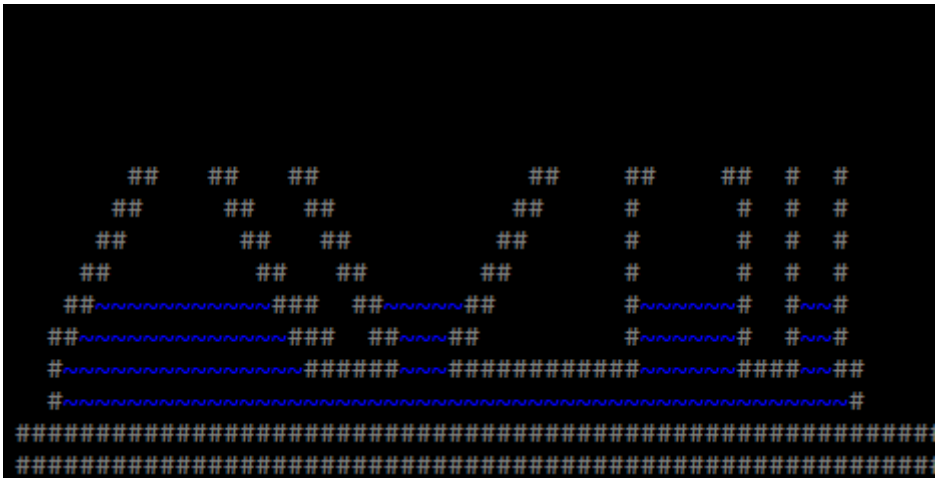
Si punctul cel mai jos (cu presiunea cea mai mare), dar cu spațiu liber deasupra; coordonatele acestora se noteaza in vectorii **(wba[],wqa[]),(wbb[],wqb[])** iar presiuniile sunt retinute temporar in variabilele **difw1** si **difw1**

```
  int difw1;
  int difw2;
for (int a=1; a<=li;a++)
    for (int b=1; b<=lj;b++)
    {
        difw1=0;
        difw2=0;
        if (type[a][b]==2 && wba[wb[a][b]]==0)wba[wb[a][b]]=a,wbb[wb[a][b]]=b;
        if (type[a][b]==2&&(type[a-1][b-1]==0||type[a-1][b]==0||type[a-1][b+1]==0))
        {
            if (wbb[wb[a][b]]-b>0)difw1=wbb[wb[a][b]]-b;
            else difw1=b-wbb[wb[a][b]];

            if (wbb[wb[a][b]]-wqb[wb[a][b]]>0)difw2=wbb[wb[a][b]]-wqb[wb[a][b]];
            else difw2=wqb[wb[a][b]]-wbb[wb[a][b]];

         if (a-1>wba[wb[a][b]]&&(kwqb==0||difw1<difw2))wqa[wb[a][b]]=a,wqb[wb[a][b]]=b,kwqb=1;

        }
    }
```

În cele din urmă sunt particula de apă cu presiunea cea mai mică este plasată în spațiul liber adiacent particulei cu cea mai mare presiune.

```
for (i=1;i<=wen;i++)
{
    if (wqa[i]!=0)
        for (j=-1;j<=1;j++)
            if (type[wqa[i]-1][wqb[i]+j]==0)
            {
                swap(type[wqa[i]-1][wqb[i]+j],type[wba[i]][wbb[i]]);
                wba[i]=0;wqa[i]=0;wbb[i]=0;wqb[i]=0;
                swap(wb[wqa[i]-1][wqb[i]+j],wb[wba[i]][wbb[i]]);
                j=1;
            }
}
```

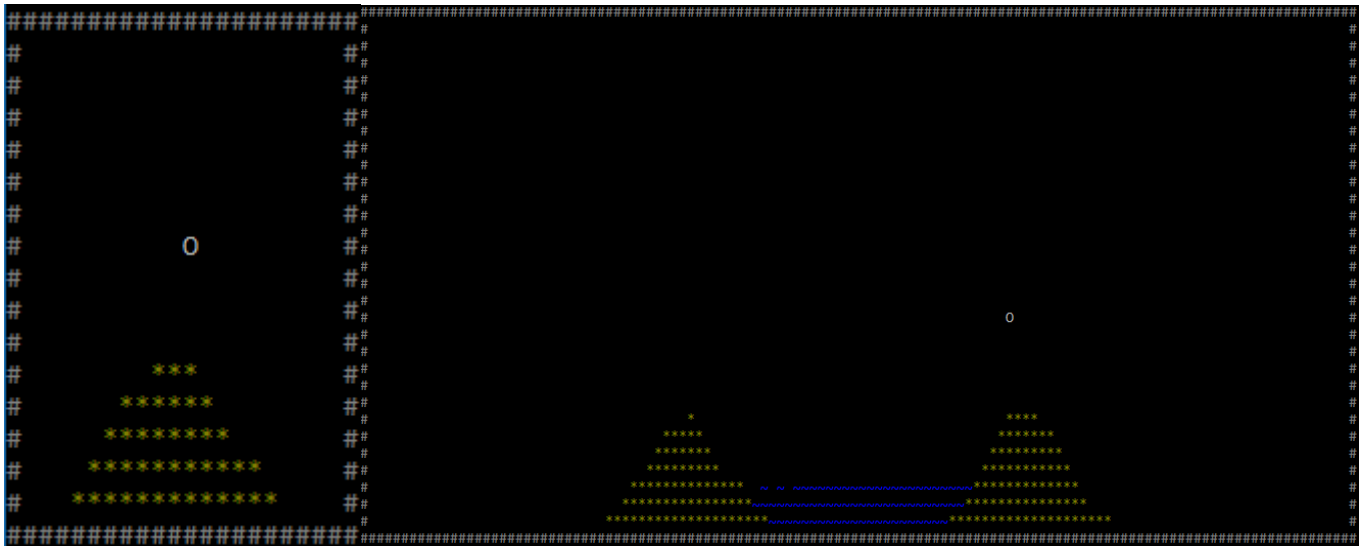
Rezultat



Interacțiunea

Înainte de compliare, utilizatorul poate modifica fară griji parametrii următori:

Rezoluția consolei (prin variabililele **li** si **lj** , maxim 90*210**)** și viteza de simulare (variabila **simspd**)



*Rezolutia 15*20* *Rezolutia 30*120*

Pentru a se realiza interacțiunea in timp real cu programul se utilizează **funcția GetAsyncKeyState(VK_CODE)** din libraria **windows.h** (funcția returnează true atunci când tasta specificată este apăsată la momentul apelării);

Pentru miscarea cursorului variabilele **crsy**, **crsx** cresc sau scad (doar dacă cursorul va rămâne vizibil în fereastra programului) in funcție de săgetile apăsate

```
if( GetAsyncKeyState(VK_LEFT))
{
    if (crsx==1)crsx=lj;
    else crsx--;
}
if( GetAsyncKeyState(VK_RIGHT))
{
    if (crsx==lj)crsx=1;
    else crsx++;
}
if( GetAsyncKeyState(VK_UP))
{
    if (crsy==1)crsy=li;
    else crsy--;
}
if( GetAsyncKeyState(VK_DOWN))
{
    if (crsy==li)crsy=1;
    else crsy++;
}
```

Plasarea respectiv stergerea particulelor se realizează schimbând valoarea matricei type[][] pe poziția cursorului in cea dorită

```
if( GetAsyncKeyState(VK_NUMPAD1))
{
    type[crsy][crsx]=1;
}
if( GetAsyncKeyState(VK_NUMPAD2))
{
    type[crsy][crsx]=2;
}
if( GetAsyncKeyState(VK_RCONTROL))
{
    type[crsy][crsx]=0;
}
if( GetAsyncKeyState(VK_NUMPAD0))
{
    type[crsy][crsx]=-1;
}
```

Iar prin apăsarea tastei *'inmulțire * '* de pe numpad se pot sterge toate particulele de pe ecran prin instructiunea

for (int a=1; a<=li;a++) for (int b=1; b<=lj;b++) type[a][b]=0;

După citirea tastelor apăsate se apelează funcțile **gravity()** si **display()**, funcția **sleep()** representând finalul cadrului ;

La inceputul următorului cadru, toate variabilele temporare sunt resetate și se repetă procesul.

Sursă

```
#include <iostream>
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

using namespace std;

int type[90][210],crsx,crsy,i,j,up[90][210],li,lj,wba[2000],wqa[2000],wbb[2000],wqb[2000],wb[90][210],wen,wk,wk2,wk3,a,b;

string screen;

HANDLE handle=GetStdHandle(STD_OUTPUT_HANDLE);

int rnd(int a, int b)
{
   return rand()%b+a;
}

void display2(int color)
{
   SetConsoleTextAttribute(handle, color);
   cout<<screen;
   screen.clear();
}

void display()
{
   screen.clear();
   int clr=0;
   for (int a=0; a<=li+1;a++)
   {
      for (int b=0; b<=lj+1;b++)
      {

   if (b==crsx&&a==crsy)
   {
      if (7!=clr)display2(clr),clr=7;
      screen+="O";
   }
  else if (type[a][b]==-1)
  {
     if (8!=clr)display2(clr),clr=8;
     screen+="#";
  }
  else if (type[a][b]==1)
  {
     if (6!=clr)display2(clr),clr=6;
     screen+="*";
  }
   else if (type[a][b]==2)
   {
      if (9!=clr)display2(clr),clr=9;
   screen+="~";
     // screen+=wb[a][b]+('0'-0);
   }
       else if (type[a][b]==0)screen+=" ";
       }
   screen+="\n";
   }

   display2(clr);
}
```

```
void gravity()
{for (int a=1; a<=li;a++) //update
  {
    for (int b=1; b<=lj;b++)
    {
      if (type[a][b]==1) // sand
      {
        if (type[a+1][b]%2==0||type[a+1][b-1]%2==0||type[a+1][b+1]%2==0)up[a][b]=1;
      }

      if (type[a][b]==2) // water
      {
        if (type[a+1][b]==0||type[a+1][b-1]==0||type[a+1][b+1]==0||type[a][b-1]==0||type[a][b+1]==0)up[a][b]=2;
      }
    }
  }


  for (int a=1; a<=li;a++) //replace
  {
    for (int b=1; b<=lj;b++)
    {
      if (up[a][b]==1) // sand
      {
        if (type[a+1][b]%2==0)swap(type[a+1][b],type[a][b]);
          else if (type[a+1][b-1]%2==0||type[a+1][b+1]%2==0)
        {
          if (rnd(0,2)==1&&type[a+1][b-1]%2==0||type[a+1][b+1]%2!=0)swap(type[a+1][b-1],type[a][b]);
            else swap(type[a+1][b+1],type[a][b]);
        }
      }

      if (up[a][b]==2) // water
      {
        if (type[a+1][b]==0)swap(type[a+1][b],type[a][b]);

        else if (type[a+1][b-1]==0||type[a+1][b+1]==0)
        {
          if (rnd(0,2)==1&&type[a+1][b-1]==0||type[a+1][b+1]!=0)swap(type[a+1][b-1],type[a][b]);
            else swap(type[a+1][b+1],type[a][b]);
        }

        else if (type[a][b-1]==0||type[a][b+1]==0)
        {
          if (rnd(0,2)==1&&type[a][b-1]==0||type[a][b+1]!=0)swap(type[a][b-1],type[a][b]);
            else swap(type[a][b+1],type[a][b]);
        }

      }
      up[a][b]=0;
    }
  }

  wen=1; //water b
  for (a=1; a<=li;a++)
    for (b=1; b<=lj;b++)
      wb[a][b]=0;

    int wk=1;
    int wk2=1;
    int wk3=1;

  while (wk2==1)
  {
      wk2=0;
      wk=1;
    while (wk==1)
    {
        wk=0;
    for (int a1=1; a1<=li;a1++)
      for (int b1=1; b1<=lj;b1++)
```

```
        {
          if (type[a1][b1]==2)
            {
          if (wb[a1][b1]==0 && wk3==1)wb[a1][b1]=wen,wk3=0;
          if (wb[a1][b1]==wen)
          {
            for (i=-1;i<=1;i++)
              for (j=-1;j<=1;j++)
              {
              if (type[a1+i][b1+j]==2 && wb[a1+i][b1+j]!=wen) wb[a1+i][b1+j]=wen,wk=1;
              }
          }
          if (wb[a1][b1]==0)wk2=1;
            }

        }
      }
      wen++;
      wk3=1;
    }

     for (i=1;i<=wen;i++)wba[i]=0,wqa[i]=0,wbb[i]=0,wqb[i]=0;  //wequal

     int kwqb=0;
     int difw1;
     int difw2;
    for (int a=1; a<=li;a++)
      for (int b=1; b<=lj;b++)
      {
        difw1=0;
        difw2=0;
        if (type[a][b]==2 && wba[wb[a][b]]==0)wba[wb[a][b]]=a,wbb[wb[a][b]]=b;
        if (type[a][b]==2&&(type[a-1][b-1]==0||type[a-1][b]==0||type[a-1][b+1]==0))
        {
          if (wbb[wb[a][b]]-b>0)difw1=wbb[wb[a][b]]-b;
          else difw1=b-wbb[wb[a][b]];

          if (wbb[wb[a][b]]-wqb[wb[a][b]]>0)difw2=wbb[wb[a][b]]-wqb[wb[a][b]];
          else difw2=wqb[wb[a][b]]-wbb[wb[a][b]];

         if (a-1>wba[wb[a][b]]&&(kwqb==0||difw1<difw2))wqa[wb[a][b]]=a,wqb[wb[a][b]]=b,kwqb=1;

        }
      }

    for (i=1;i<=wen;i++)
    {
      if (wqa[i]!=0)
        for (j=-1;j<=1;j++)
          if (type[wqa[i]-1][wqb[i]+j]==0)
          {
            swap(type[wqa[i]-1][wqb[i]+j],type[wba[i]][wbb[i]]);
            wba[i]=0;wqa[i]=0;wbb[i]=0;wqb[i]=0;
            swap(wb[wqa[i]-1][wqb[i]+j],wb[wba[i]][wbb[i]]);
            j=1;
          }
    }

}

void walls ()
{
for (i=0; i<=li+1;i++)
    {
      for (j=0; j<=lj+1;j++)
      {  if (i==li+1||j==0||j==lj+1||i==0) type[i][j]=-1;}
    }
}
```

```
int main()
{
   srand (time(NULL));
   li=9*3;
   lj=21*4;
   int simspd =50;
   crsx=1;
   crsy=1;
   walls();

   COORD zerozero={0,0};
   CONSOLE_CURSOR_INFO crsvis;
   crsvis.bVisible = 0,crsvis.dwSize = 1;
   SetConsoleCursorInfo(handle, &crsvis);

   while (true)
   {
      SetConsoleCursorPosition( handle, zerozero );

      if( GetAsyncKeyState(VK_LEFT)) {
         if (crsx==1)crsx=lj;
         else crsx--;
      }
      if( GetAsyncKeyState(VK_RIGHT)) {
         if (crsx==lj)crsx=1;
         else crsx++;
      }
      if( GetAsyncKeyState(VK_UP)) {
         if (crsy==1)crsy=li;
         else crsy--;
      }
      if( GetAsyncKeyState(VK_DOWN)) {
         if (crsy==li)crsy=1;
         else crsy++;
      }
      if( GetAsyncKeyState(VK_NUMPAD1)) {type[crsy][crsx]=1;
      }
      if( GetAsyncKeyState(VK_NUMPAD2))  {type[crsy][crsx]=2;
      }
      if( GetAsyncKeyState(VK_RCONTROL)) {  type[crsy][crsx]=0;
      }
      if( GetAsyncKeyState(VK_NUMPAD0))  {  type[crsy][crsx]=-1;
      }
      if( GetAsyncKeyState(VK_MULTIPLY)) {
         for (int a=1; a<=li;a++)
      for (int b=1; b<=lj;b++)
      {
         type[a][b]=0;
      }
      }
      if( GetAsyncKeyState(VK_SUBTRACT))
      {
         for (int a=0; a<=li+1;a++)for (int b=0; b<=lj+1;b++)
            if (b==0||b==lj+1||a==0)
               type[a][b]=0;
      }

      if( GetAsyncKeyState(VK_ADD))
      {
         walls();
      }

         for (int a=0; a<=li+1;a++)for (int b=0; b<=lj+1;b++)
            if (type[a][b]!=-1&&(a==li+1||b==0||b==lj+1||a==0))
               type[a][b]=0;

      gravity();
      display();
      Sleep(simspd);
   }
}
```

Mulțumesc pentru atenția acordată

-Program și prezentare scrise de Novetschi Vlad,

elev al Colegiului Național Unirea Focșani, clasa 12B , profesor Oprea Marilena.