Coding the dimensional basic

The separate fundamental forces of nature: – the strong interaction, the electromagnetic interaction, the weak interaction and the gravitational interaction – are calculatable with one formula out of one principle. The statistical math of the quantum theory is set aside in favor of a goniometric approach. Gravitation is the only force that matters and the strong force, the electromagnetic force and the weak force can be explained out of gravitation, while gravity itself is only caused by the curvature of a mass, corresponding with a certain amount of bending of spacetime.

The axiom is that the most elementary particle in existence is the dimensional basic (db or 人). The 人 itself has no dimensions (no length, no width and no height). The 人 is found everywhere in the universe and is always moving through spacetime, where the speed of the movement of the 人, in respect to its surroundings, can have any value. The curvature of space on the location of the 人 is infinite while time on the location of the 人 stands still. The 人 behaves like a black hole without dimensions. The 人 is the building block of all that we perceive.

The formula for the extent of spacetime curvature around a 人 is:

$$\sqrt{x^2 + y^2 + z^2} \times Kr = 1 \quad (0) .$$

In the formula: x, y, z, are coordinates in spacetime [m], Kr = curvature [m^{-1}].

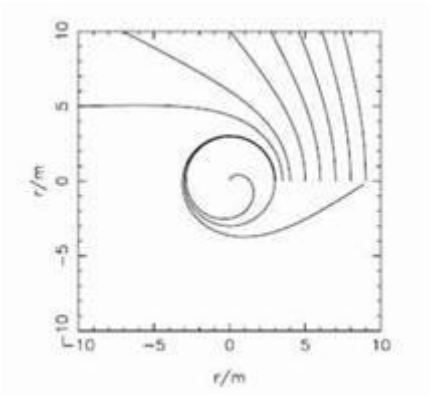
Formula (0) describes the relative lessened extent of curvature of spacetime surrounding the 人.

The distance between the various 人 s varies in time by movements relative to each other. The direction of movement is being influenced according to gravitational laws. The tracks of movement are being influenced by the curvature of spacetime caused by the 人s themselves. This means that spacetime surrounding a 人 gets smaller when the 人 s are approaching each other while spacetime surrounding a 人 gets bigger when the 人s move away from each other.

The ⅄ is different than other particles in that respect that other particles consist out of multiple ⅄s while the ⅄ itself is a singular particle. Each ⅄ is a singularity (infinite curvature) on itself while other particles than the ⅄ are a combination of multiple ⅄s and thus a system of multiple singularities.

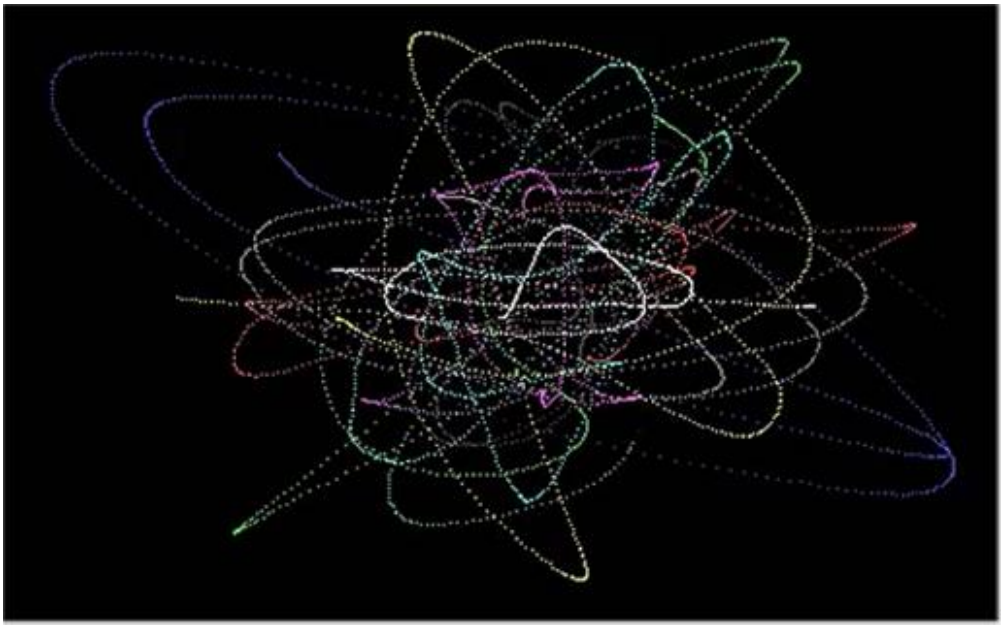
The observed forces (strong, electromagnetic, weak and gravitation) have the same origin. The cause of these forces are because of the characteristics of a singular ⅄. The observed forces are in fact a sum of circular movements that come to exist when multiple ⅄s interact with each other.

Figure 1: The tracks of two interacting ⅄s on different distances from each other (Original: Deflection of the tracks of a photon close to an object with a heavy mass).



In figure 1 is shown how the movement tracks of photons react to the event horizon of a black hole. The same regularity applies to a binary black hole system. This is equal to the movement tracks of two ⅄s in respect to each other with the difference that the two ⅄s have no event horizon. The Pauli principle is never violated because the ⅄s have no dimensions, they can approach each other, but can never touch each other. These movement tracks are equal in behavior to Newton's laws of gravity. On the basis of that information the Borland C computer program 'Newton' has been developed. This computer program shows the movement tracks of ⅄s in three dimensional spacetime, in which the movement tracks of the ⅄s follow the gravitational laws. A three dimensional snapshot with nine interacting ⅄s is shown in figure 2. In this figure the Einsteinian bending of spacetime has not been taken into account. The computer program 'Newton' gives the possibility to show the time delay in video, as seen by an outside observer thus making clear the principle of time delay.

Figure 2: The movement tracks of nine 人s during a random time.



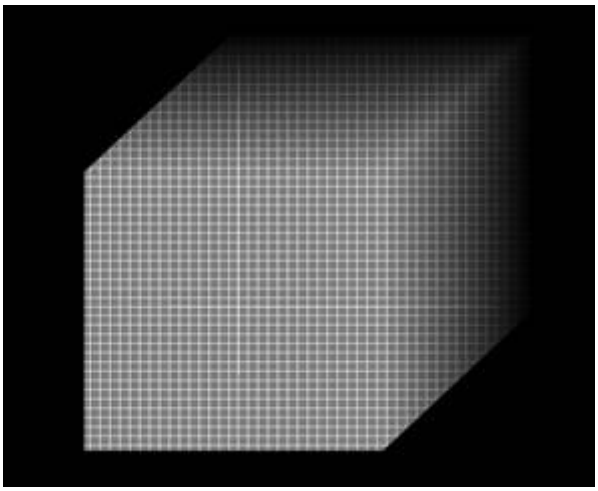
The second model that has been developed is the Borland C computer plot program 'Einstein'. This computer program has been developed to show how spacetime around a 人 is being bend as seen by an outside observer, the extent of bending calculated according to formula (0).

Just like one 人 as a singular singularity causes bending of spacetime because of an infinite curvature, a multitude of 人s will show a stronger bending of spacetime because of a sum of infinite curvatures. As Einstein made clear, we can speak of curbed spacetime instead of linear spacetime. The more mass an object has, the more spacetime bends. In fact mass is the sum of the curvatures of a certain amount of 人s close to each other. In case of for example three billion 人s one can speak of three billion times infinite curvature. This makes it possible to isolate infinite numbers in comparison equations and thus mass can be expressed as an absolute number. One can say that a cluster of a certain amount of 人s will have an absolute number of infinite curvatures. In this way one can speak of mass A with X times infinite curvatures, while mass B has Y times infinite curvatures. The infinities on both sides of the comparison can be done away with and only the absolute proportions of X and Y remain for the respective masses. A cluster of 人s with an absolute amount of 人s correlates with the mass of an object and thus a certain extent of bending of spacetime.

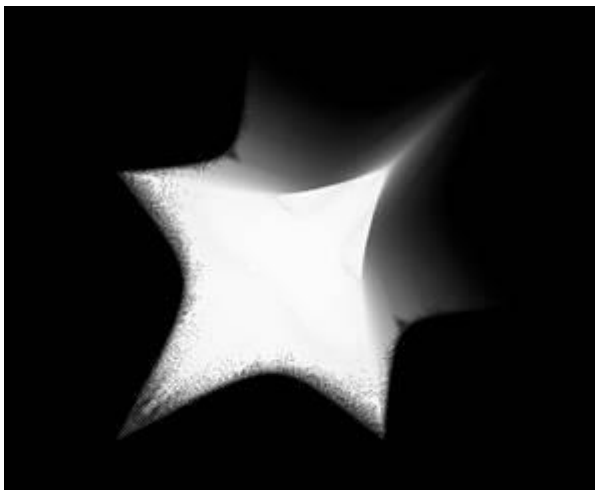
The extent of bending of spacetime is calculated using formula (0), where the extent of curvature on a specific position of spacetime is being calculated. A bigger curvature means that spacetime is more bended, whereas a smaller curvature means that spacetime is less bended.

An example of this is shown in figure 3. In figure 3 the plot of a cube of spacetime is shown. The Einsteinian bending of a cube of spacetime is made visual. While figure 3a shows no bending of spacetime because of the absence of a 人, the bending in a cube of spacetime, and thus deformed distances for an outside observer, in figure 3b have been calculated according to formula (0) because of the position of a 人 in the center of the cube of spacetime. At the center of the six surfaces of the cube of spacetime the distance to the 人 is the smallest, for the outside observer it appears that that piece of spacetime is closer to the 人 than it should be in linear (uncurbed) spacetime, this because of the bending of spacetime, made visual by formula (0). Hence the pointy form of the corners of the cube of spacetime, there the distance to the 人 is the biggest. Because of the bending of spacetime the distance is bigger for the outside observer than it should be according to a linear scale, this again made visual by calculating the extent of bending of spacetime according to formula (0). The closer spacetime is to a 人, the higher the curvature and the more spacetime will be bend.

Figure 3: The bending of a cube of spacetime under the influence of a 人.



3a. Uncurbed (linear) cube of spacetime



3b. Cube of spacetime curbed by the presence of a 人 in the center.

Conclusion: The Newtonian laws represent the straight movement paths as being caused by the bending of spacetime, just like Einstein made clear. Thus Newton's laws of gravity apply to the movement paths of the 人 or a multitude of 人s.

Both computer programs together represent the movement and character of the 人. The reality of the 人 can be simulated by computer programs according to gravitational laws, taking into account the reality of formula (0) and the thereby caused bending of spacetime, with, as seen by an outside observer, the observed bending of spacetime and time delay. A third model, combining the linear Newtonian laws of gravity with Einsteinian bending of space and delay of time should be able to simulate the universe as a whole. Whereas a model with an infinite amount of 人s is practically not possible, a model with a subset of a large number of 人s should be possible.

Below are shown the sources codes of the Borland C computer programs 'Newton' and 'Einstein', preceded by a MS Quick Basic example of 人 movement analysis with which figure 2 has been calculated.

1) db movement analysis (quick basic)

```
REM (C) G.J. Smit, Nijmegen, Nederland
REM This software is published under the GNU General Public License v3.0
REM www.dbphysics.org
REM Program purpose: db movement analysis

KEY(1) ON: ON KEY(1) GOSUB afrondschoonscherm
KEY(2) ON: ON KEY(2) GOSUB andermode
KEY(3) ON: ON KEY(3) GOSUB nieuwecoordinaten
KEY(4) ON: ON KEY(4) GOSUB windowgrootte
KEY(5) ON: ON KEY(5) GOSUB sterktezwaartekracht
KEY(6) ON: ON KEY(6) GOSUB nieuwaantaldeeltjes
KEY(7) ON: ON KEY(7) GOSUB lijnmetwis
KEY(8) ON: ON KEY(8) GOSUB lijnzonderwis
KEY(9) ON: ON KEY(9) GOSUB willekeuroud
KEY(10) ON: ON KEY(10) GOSUB willekeurnieuw
```

```
DIM x(100, 103), y(100, 103), z(100, 103), xfz(100), yfz(100), zfz(100)
DIM x2d(200), y2d(200)

SCREEN 12, 0:  CLS
xyz = 100
mfz = .1
aantal = 3
scherm = 1
begincord = 1
lijn = 0
willoud = 100
willnieuw = 1
wg = 3 * willoud
afrond = 0

WINDOW (-wg, wg)-(wg, -wg)

prog = 1

WHILE prog > 0

CLS

FOR tel = 0 TO aantal - 1
  x(tel, 0) = (RND(1) * 2 * willoud) - willoud: x(tel, 1) = x(tel, 0) + (RND(1) * 2 * willnieuw) - willnieuw
  y(tel, 0) = (RND(1) * 2 * willoud) - willoud: y(tel, 1) = y(tel, 0) + (RND(1) * 2 * willnieuw) - willnieuw
  z(tel, 0) = (RND(1) * 2 * willoud) - willoud: z(tel, 1) = z(tel, 0) + (RND(1) * 2 * willnieuw) - willnieuw
NEXT tel

IF begincord = 1 THEN GOSUB bcord
```

```
GOSUB status

prog = 2

WHILE prog > 1

FOR tel1 = 0 TO aantal - 1
  x(tel1, 2) = x(tel1, 1) - x(tel1, 0)
  y(tel1, 2) = y(tel1, 1) - y(tel1, 0)
  z(tel1, 2) = z(tel1, 1) - z(tel1, 0)
  FOR tel2 = tel1 TO aantal - 1
    x(tel1, 3 + tel1) = x(tel2, 1) - x(tel1, 1)
    y(tel1, 3 + tel1) = y(tel2, 1) - y(tel1, 1)
    z(tel1, 3 + tel1) = z(tel2, 1) - z(tel1, 1)
    x(tel2, 3 + tel2) = -x(tel1, 3 + tel1)
    y(tel2, 3 + tel2) = -y(tel1, 3 + tel1)
    z(tel2, 3 + tel2) = -z(tel1, 3 + tel1)
    x(tel1, 3 + aantal + tel1) = ABS(x(tel1, 3 + tel1))
    y(tel1, 3 + aantal + tel1) = ABS(y(tel1, 3 + tel1))
    z(tel1, 3 + aantal + tel1) = ABS(z(tel1, 3 + tel1))
    x(tel2, 3 + aantal + tel2) = ABS(x(tel2, 3 + tel2))
    y(tel2, 3 + aantal + tel2) = ABS(y(tel2, 3 + tel2))
    z(tel2, 3 + aantal + tel2) = ABS(z(tel2, 3 + tel2))
  NEXT tel2
NEXT tel1

FOR tel1 = 0 TO aantal - 1
  xfz(tel1) = 0
  yfz(tel1) = 0
  zfz(tel1) = 0
  FOR tel2 = 0 TO aantal - 1
```

```
    IF x(tel1, 3 + aantal + tel2) > 0 THEN xfz(tel1) = xfz(tel1) + x(tel1, 3 + tel2) * mfz / x(tel1, 3 + aantal + tel2)
    IF y(tel1, 3 + aantal + tel2) > 0 THEN yfz(tel1) = yfz(tel1) + y(tel1, 3 + tel2) * mfz / y(tel1, 3 + aantal + tel2)
    IF z(tel1, 3 + aantal + tel2) > 0 THEN zfz(tel1) = zfz(tel1) + z(tel1, 3 + tel2) * mfz / z(tel1, 3 + aantal + tel2)
  NEXT tel2
  x(tel1, 0) = x(tel1, 1)
  IF afrond = 0 THEN x(tel1, 1) = x(tel1, 0) + x(tel1, 2) + xfz(tel1) ELSE x(tel1, 1) = INT(x(tel1, 0) + x(tel1, 2) + xfz(tel1))
  y(tel1, 0) = y(tel1, 1)
  IF afrond = 0 THEN y(tel1, 1) = y(tel1, 0) + y(tel1, 2) + yfz(tel1) ELSE y(tel1, 1) = INT(y(tel1, 0) + y(tel1, 2) + yfz(tel1))
  z(tel1, 0) = z(tel1, 1)
  IF afrond = 0 THEN z(tel1, 1) = z(tel1, 0) + z(tel1, 2) + zfz(tel1) ELSE z(tel1, 1) = INT(z(tel1, 0) + z(tel1, 2) + zfz(tel1))
NEXT tel1

midx = 0
midy = 0
midz = 0

FOR tel = 0 TO aantal - 1
midx = midx + x(tel, 1)
midy = midy + y(tel, 1)
midz = midz + z(tel, 1)
NEXT tel

midx = midx / aantal
midy = midy / aantal
midz = midz / aantal


w2dx = midy - midx * .5
w2dy = midz - midx * .5

IF lijn = 2 THEN GOSUB wislijn:
```

```
FOR tel = 0 TO aantal - 1
  x2d(tel) = y(tel, 1) - x(tel, 1) * .5
  y2d(tel) = z(tel, 1) - x(tel, 1) * .5
NEXT tel

WINDOW (-wg + w2dx, wg + w2dy)-(wg + w2dx, -wg + w2dy)

IF lijn = 0 THEN GOSUB tekenpunt:  ELSE GOSUB tekenlijn:

WEND
WEND

andermode:
scherm = scherm + 1
IF scherm > 2 THEN scherm = 0
IF scherm = 0 THEN SCREEN 9, 0: WIDTH 80, 43: COLOR 1, 10
IF scherm = 1 THEN SCREEN 12: WIDTH 80, 60
IF scherm = 2 THEN SCREEN 13
GOSUB status
RETURN

afrondschoonscherm:
IF afrond = 0 THEN afrond = 1 ELSE afrond = 0
GOSUB status
RETURN

nieuwecoordinaten:
prog = 1
CLS
RETURN

sterktezwaartekracht:
```

```
PRINT "Mate van zwaartekracht is:"; mfz

INPUT "Nieuwe mate:", mfz

GOSUB status

RETURN


nieuwaantaldeeltjes:

PRINT "Aantal deeltjes is:"; aantal

INPUT "Nieuw aantal:", aantal

IF aantal < 1 THEN aantal = 1

IF aantal > 50 THEN aantal = 50

CLS

prog = 1

RETURN


windowgrootte:

PRINT "Windowgrootte is:"; wg

INPUT "Nieuwe grootte:", wg

IF wg < 10 THEN wg = 10

IF wg > 500 THEN wg = 500

GOSUB status:

RETURN


willekeuroud:

PRINT "Randomize oude co”rdinaat is:"; willoud

INPUT "Nieuwe randomize factor:"; willoud

IF willoud < 1 THEN willoud = 1

IF willoud > 10000 THEN willoud = 10000

wg = 3 * willoud

CLS

prog = 1

RETURN
```

```
willekeurnieuw:

PRINT "Randomize nieuwe co”rdinaat is:"; willnieuw

INPUT "Nieuwe randomize factor:"; willnieuw

IF willnieuw < .0000001 THEN willoud = .0000001

IF willnieuw > 1000 THEN willnieuw = 1000

CLS

prog = 1

RETURN



lijnzonderwis:

IF lijn = 1 THEN lijn = 0 ELSE lijn = 1

CLS

IF lijn = 0 THEN GOSUB status:

RETURN

lijnmetwis:

IF lijn = 2 THEN lijn = 0 ELSE lijn = 2

CLS

IF lijn = 0 THEN GOSUB status:

RETURN



bcord:

begincord = 0

x(0, 0) = xyz: x(0, 1) = xyz

y(0, 0) = 0: y(0, 1) = -.9

z(0, 0) = 0: z(0, 1) = .9

x(1, 0) = 0: x(1, 1) = .9

y(1, 0) = xyz: y(1, 1) = xyz

z(1, 0) = 0: z(1, 1) = -.9

x(2, 0) = 0: x(2, 1) = -.9
```

```
y(2, 0) = 0: y(2, 1) = .9

z(2, 0) = xyz: z(2, 1) = xyz


RETURN


tekenpunt:


FOR tel = 0 TO aantal - 1

  PSET (x2d(tel), y2d(tel)), 7 + tel

NEXT tel

RETURN


tekenlijn:

FOR tel1 = 0 TO aantal - 1

  FOR tel2 = tel1 TO aantal - 1

  LINE (x2d(tel1), y2d(tel1))-(x2d(tel2), y2d(tel2)), 2 + tel1 + tel2

  NEXT tel2

NEXT tel1

RETURN


wislijn:

CLS

RETURN


status:

CLS

IF scherm = 0 THEN PRINT "EGA (16k)"

IF scherm = 1 THEN PRINT "VGA (16k)"

IF scherm < 2 THEN PRINT "Window-grootte  :"; wg

IF scherm < 2 THEN PRINT "Sterkte Fzwaarte:"; mfz

IF scherm < 2 THEN PRINT "Aantal 1db's    :"; aantal

IF scherm < 2 THEN PRINT "r_oud           :"; willoud
```

```
IF scherm < 2 THEN PRINT "r_nieuw         :"; willnieuw

IF scherm < 2 THEN PRINT "Afronding c_oud :"; afrond

RETURN
```

2) Program Newton (Borland C)

```
// (C) 1996 G.J. Smit, Nijmegen, Nederland

// This software is published under the GNU General Public License v3.0

// www.dbphysics.org

// The program 'Newton' is a n-body model where simultaneously for the positions of multiple dimensional basics (theoretical infinite curvature) moving through space time interact with each other according to newtonian laws.


#include "stdio.h"

#include "stdlib.h"

#include "conio.h"

#include "string.h"

#include "float.h"

#include "math.h"

#include "graphics.h"

#include "svga256.h"


FILE *bestand;            // Pointer voor geopend bestand.

char b_naam[12];          // Naam van actief bestand.

char b_test;              // Controle voor bestaan bestand.

char toets;               // Variabele voor ingedrukte toets.

int t1,t2,t3,t4;          // Tellers voor lussen.

int stap,dim,deel;        // Aantal stappen, dimensies en deeltjes.

int spoor;                // Lengte afbeelding in tijd per deeltje.

int prog;                 // Programma einde.

int i_temp;               // Tijdelijk opslag integer.

int midd;                 // Middelpunt in tekening aan/uit.
```

```
int modus;                // Kleur per deel/diepte.
float w_g, w_x, w_y;      // Windowgrootte, x en y co”rdinaatgrootte.
float x_max, y_max;       // Aantal pixels op beeldscherm.
float t_frag, grens;      // Tijdfragmentatie en grenswaarde ruimte.
float r_o, r_n;           // Bereik willekeurige beginco”rdinaten.
float fzx, fzy, fzz;      // Zwaartekracht per as.
float g_temp;             // Grenswaarde wisseling.
float f_temp;             // Tijdelijk opslag float.
float midx, midy;         // Middelpunt berekening-variabelen.
float diepte;             // Kleur-diepte variabele.
float huge x3[1851][4];   // Maximaal 250 x3d-co”rdinaten.
float huge y3[1851][4];   // Maximaal 250 y3d-co”rdinaten.
float huge z3[1851][4];   // Maximaal 250 z3d-co”rdinaten.
float huge x2[1851][30];  // Maximaal 30 x2d-co”rdinaten per deeltje.
float huge y2[1851][30];  // Maximaal 30 y2d-co”rdinaten per deeltje.

int huge DetectSvga256(){ int Vid; Vid=4; return Vid; }

void theorie(void) // Rekenkundig variabele verhoudingen.
{ t1=0; t2=2639;
  installuserdriver("Svga256",DetectSvga256);
  initgraph(&t1,&t2,"");
  setcolor(7);
  for(t1=0; t1<20; t1++)
  { line(x_max/2+200-t1*10, y_max/2, x_max/2, y_max/2+t1*10);
    line(x_max/2-t1*10, y_max/2, x_max/2, y_max/2+200-t1*10);
    line(x_max/2-200+t1*10, y_max/2, x_max/2, y_max/2-t1*10);
    line(x_max/2+t1*10, y_max/2, x_max/2, y_max/2-200+t1*10);
  }
  getch();
  closegraph();
}
```

```
void varbestand(void) // Variabele waarden bestand inlezen.
{ bestand=fopen(b_naam,"r");
  if(bestand==NULL) b_test=0;
  else
  { fscanf(bestand, "%d%d%f%f%f%f", &stap, &deel, &r_o, &r_n, &t_frag, &grens);
    if(grens!=0) w_g=1.5*grens;
    b_test=1;
  }
  fclose(bestand);
}

void menuopbeeld(void)
{ clrscr();
  textcolor(10);
  printf("[b]estandsnaam  ");
  if(b_test==0) printf("-"); else printf("+");
  printf("  : %s\n\n", b_naam);
  printf("[c]o”rdinaten\n");
  printf("[w]illekeur        : %f\n", r_o);
  printf("[r]ichting         : %f\n\n", r_n);
  printf("[t]ijdfragmentatie : %f\n", t_frag);
  printf("[g]rens            : %f\n\n", grens);
  printf("[s]tappen          : %d\n", stap);
  printf("[d]eel             : %d\n\n", deel);
  printf("[v]enster          : %f\n", w_g);
  printf("[p]rojectie        : %d\n", spoor);
  printf("M goedmaken voor 3d: ");
//printf("[m]iddelpunt       : ");
  if(midd==0) printf("uit\n"); else printf("aan\n");
  printf("[k]leurmodus       : ");
  if(modus==0) printf("deel\n\n\n"); else printf("diepte\n\n\n");
```

```
 printf("[R]ekenen [E]n [T]ekenen [S]toppen\n\n");
}

void menuvraag(void)
{ toets=getch();
 if(toets==98)  { printf("Nieuwe bestandsnaam? ");
                  scanf("%s", &b_naam); }
 if(toets==99)  { printf("Invoer co”rdinaten, nog programmeren...");
                  getch(); }
 if(toets==119) { f_temp=r_o; printf("Maximale willekeur? ");
                  scanf("%f", &r_o);
                  if(r_o<0||r_o==0||r_o>30000) r_o=f_temp; }
 if(toets==114) { f_temp=r_n; printf("Maximale richting? ");
                  scanf("%f", &r_n);
                  if(r_n<0||r_o==0||r_o>2500) r_o=f_temp; }
 if(toets==116) { f_temp=t_frag; printf("Nieuwe tijdfragmentatie? ");
                  scanf("%f", &t_frag);
                  if(t_frag<0||t_frag==0||t_frag>1) t_frag=f_temp; }
 if(toets==103) { f_temp=grens; printf("Grens van ruimte? ");
                  scanf("%f", &grens);
                  if(grens<0||grens>32500) grens=f_temp; }
 if(toets==115) { i_temp=stap; printf("Aantal stappen? ");
                  scanf("%d", &stap);
                  if(stap<1||stap>32500) stap=i_temp; }
 if(toets==100) { i_temp=deel; printf("Aantal deeltjes? ");
                  scanf("%d", &deel);
                  if(deel<2||deel>1850) deel=i_temp; }
 if(toets==118) { f_temp=w_g; printf("Venstergrootte? ");
                  scanf("%f", &w_g);
                  if(w_g<0||w_g==0||w_g>32500) w_g=f_temp; }
 if(toets==112) { i_temp=spoor; printf("Aantal fragmenten? ");
                  scanf("%d", &spoor);
```

```
                if(spoor<0||spoor>30) spoor=i_temp; }
  if(toets==109) { if(midd==0) midd=1; else midd=0; }
  if(toets==107) { if(modus==0) modus=1; else modus=0; }
}

void willekeur(void)
{ for(t1=0;t1<deel;t1++)
  { x3[t1][0]=2*(random(32767)*r_o/32767)-r_o;
    y3[t1][0]=2*(random(32767)*r_o/32767)-r_o;
    z3[t1][0]=2*(random(32767)*r_o/32767)-r_o;
    x3[t1][1]=x3[t1][0]+2*(random(32767)*r_n/32767)-r_n;
    y3[t1][1]=y3[t1][0]+2*(random(32767)*r_n/32767)-r_n;
    z3[t1][1]=z3[t1][0]+2*(random(32767)*r_n/32767)-r_n;
  }
}

void reken(void) // Kaal [R]ekenen, snelste routine.
{ // Bestand voor co”rdinaten openen.
  bestand=fopen(b_naam,"w");
  fprintf(bestand, "%d %d %f %f %f %f", stap, deel, r_o, r_n, t_frag, grens);
  for(t1=0;t1<deel;t1++)
  fprintf(bestand, " %f %f %f", x3[t1][1], y3[t1][1], z3[t1][1]);

  // Co”rdinaten berekenen, schrijven naar disk en naar tekst-beeldscherm.
  for(t1=0;t1<stap;t1++)
  { for(t2=0;t2<deel;t2++)
    { x3[t2][3]=0;
      y3[t2][3]=0;
      z3[t2][3]=0;
    }
    for(t2=0;t2<deel;t2++)
    { x3[t2][2]=x3[t2][1]-x3[t2][0];
```

```
      y3[t2][2]=y3[t2][1]-y3[t2][0];
      z3[t2][2]=z3[t2][1]-z3[t2][0];
      for(t3=t2;t3<deel;t3++)
      { fzx=x3[t3][1]-x3[t2][1];
        fzy=y3[t3][1]-y3[t2][1];
        fzz=z3[t3][1]-z3[t2][1];
        if(fzx!=0) { fzx=1/fzx; x3[t2][3]=x3[t2][3]+fzx;
                     x3[t3][3]=x3[t3][3]-fzx; }
        if(fzy!=0) { fzy=1/fzy; y3[t2][3]=y3[t2][3]+fzy;
                     y3[t3][3]=y3[t3][3]-fzy; }
        if(fzz!=0) { fzz=1/fzz; z3[t2][3]=z3[t2][3]+fzz;
                     z3[t3][3]=z3[t3][3]-fzz; }
      }
      x3[t2][0]=x3[t2][1];
      y3[t2][0]=y3[t2][1];
      z3[t2][0]=z3[t2][1];
      x3[t2][1]=x3[t2][0]+x3[t2][2]+x3[t2][3];
      y3[t2][1]=y3[t2][0]+y3[t2][2]+y3[t2][3];
      z3[t2][1]=z3[t2][0]+z3[t2][2]+z3[t2][3];
    }
    for(t2=0;t2<deel;t2++)
    fprintf(bestand, " %f %f %f", x3[t2][1], y3[t2][1], z3[t2][1]);
    putchar(13); printf("%d",t1+1);
  }
  fclose(bestand);
  b_test=1;
}

void rekenmetopties(void) // [R]ekenen met grens en/of t_frag aan.
{ // Bestand voor co”rdinaten openen.
  bestand=fopen(b_naam,"w");
  fprintf(bestand, "%d %d %f %f %f %f", stap, deel, r_o, r_n, t_frag, grens);
```

```
for(t1=0;t1<deel;t1++)
fprintf(bestand, " %f %f %f", x3[t1][1], y3[t1][1], z3[t1][1]);


// Co”rdinaten berekenen, schrijven naar disk en naar tekst-beeldscherm.
for(t1=0;t1<stap;t1++)
{ for(t2=0;t2<deel;t2++)
  { x3[t2][3]=0;
    y3[t2][3]=0;
    z3[t2][3]=0;
  }
  for(t2=0;t2<deel;t2++)
  { x3[t2][2]=x3[t2][1]-x3[t2][0];
    y3[t2][2]=y3[t2][1]-y3[t2][0];
    z3[t2][2]=z3[t2][1]-z3[t2][0];
    for(t3=t2;t3<deel;t3++)
    { fzx=x3[t3][1]-x3[t2][1];
      fzy=y3[t3][1]-y3[t2][1];
      fzz=z3[t3][1]-z3[t2][1];
      if(fzx!=0) { fzx=1/fzx; x3[t2][3]=x3[t2][3]+fzx;
                   x3[t3][3]=x3[t3][3]-fzx; }
      if(fzy!=0) { fzy=1/fzy; y3[t2][3]=y3[t2][3]+fzy;
                   y3[t3][3]=y3[t3][3]-fzy; }
      if(fzz!=0) { fzz=1/fzz; z3[t2][3]=z3[t2][3]+fzz;
                   z3[t3][3]=z3[t3][3]-fzz; }
    }
    // Bewerk co”rdinaten als t_frag ongelijk aan 1.
    if(t_frag!=1)
    { x3[t2][2]=x3[t2][2]*t_frag; x3[t2][3]=x3[t2][3]*t_frag;
      y3[t2][2]=y3[t2][2]*t_frag; y3[t2][3]=y3[t2][3]*t_frag;
      z3[t2][2]=z3[t2][2]*t_frag; z3[t2][3]=z3[t2][3]*t_frag;
    }
    // Bepaal de nieuwe co”rdinaten.
```

```
      x3[t2][0]=x3[t2][1];
      y3[t2][0]=y3[t2][1];
      z3[t2][0]=z3[t2][1];
      x3[t2][1]=x3[t2][0]+x3[t2][2]+x3[t2][3];
      y3[t2][1]=y3[t2][0]+y3[t2][2]+y3[t2][3];
      z3[t2][1]=z3[t2][0]+z3[t2][2]+z3[t2][3];
      // Test grensoverschrijding.
      if(grens>0)
      { if(x3[t2][1]<-grens||x3[t2][1]>grens)
        { g_temp=x3[t2][1]; x3[t2][1]=-x3[t2][0]; x3[t2][0]=-g_temp; }
        if(y3[t2][1]<-grens||y3[t2][1]>grens)
        { g_temp=y3[t2][1]; y3[t2][1]=-y3[t2][0]; y3[t2][0]=-g_temp; }
        if(z3[t2][1]<-grens||z3[t2][1]>grens)
        { g_temp=z3[t2][1]; z3[t2][1]=-z3[t2][0]; z3[t2][0]=-g_temp; }
      }
    }
    for(t2=0;t2<deel;t2++)
    fprintf(bestand, " %f %f %f", x3[t2][1], y3[t2][1], z3[t2][1]);
    putchar(13); printf("%d",t1+1);
  }
  fclose(bestand);
  b_test=1;
}

void rekenenteken(void) // [E]n.
{ t1=0; t2=2639;
  installuserdriver("Svga256",DetectSvga256);
  initgraph(&t1,&t2,"");
  w_x=(x_max+1)/(w_g*2); w_y=(y_max+1)/(w_g*2);
  printf("     |%d|%d|%f|%f|%f|%f|%f|%d|%s",
         stap, deel, r_o, r_n, t_frag, grens, w_g, spoor, b_naam);
  gotoxy(0,0);
```

```
midd=0; spoor=0; modus=0;


// Bestand voor co”rdinaten openen.
bestand=fopen(b_naam,"w");
fprintf(bestand, "%d %d %f %f %f %f", stap, deel, r_o, r_n, t_frag, grens);
for(t1=0;t1<deel;t1++)
fprintf(bestand, " %f %f %f", x3[t1][1], y3[t1][1], z3[t1][1]);


// Co”rdinaten berekenen, schrijven naar disk en naar grafisch beeldscherm.
for(t1=0;t1<stap;t1++)
{ for(t2=0;t2<deel;t2++)
  { x3[t2][3]=0;
    y3[t2][3]=0;
    z3[t2][3]=0;
  }
  for(t2=0;t2<deel;t2++)
  { x3[t2][2]=x3[t2][1]-x3[t2][0];
    y3[t2][2]=y3[t2][1]-y3[t2][0];
    z3[t2][2]=z3[t2][1]-z3[t2][0];
    for(t3=t2;t3<deel;t3++)
    { fzx=x3[t3][1]-x3[t2][1];
      fzy=y3[t3][1]-y3[t2][1];
      fzz=z3[t3][1]-z3[t2][1];
      if(fzx!=0) { fzx=1/fzx; x3[t2][3]=x3[t2][3]+fzx;
                   x3[t3][3]=x3[t3][3]-fzx; }
      if(fzy!=0) { fzy=1/fzy; y3[t2][3]=y3[t2][3]+fzy;
                   y3[t3][3]=y3[t3][3]-fzy; }
      if(fzz!=0) { fzz=1/fzz; z3[t2][3]=z3[t2][3]+fzz;
                   z3[t3][3]=z3[t3][3]-fzz; }
    }
    // Bewerk co”rdinaten als t_frag ongelijk aan 1.
    if(t_frag!=1)
```

```
  { x3[t2][2]=x3[t2][2]*t_frag; x3[t2][3]=x3[t2][3]*t_frag;
    y3[t2][2]=y3[t2][2]*t_frag; y3[t2][3]=y3[t2][3]*t_frag;
    z3[t2][2]=z3[t2][2]*t_frag; z3[t2][3]=z3[t2][3]*t_frag;
  }
  // Bepaal de nieuwe co”rdinaten.
  x3[t2][0]=x3[t2][1];
  y3[t2][0]=y3[t2][1];
  z3[t2][0]=z3[t2][1];
  x3[t2][1]=x3[t2][0]+x3[t2][2]+x3[t2][3];
  y3[t2][1]=y3[t2][0]+y3[t2][2]+y3[t2][3];
  z3[t2][1]=z3[t2][0]+z3[t2][2]+z3[t2][3];
  // Test grensoverschrijding.
  if(grens>0)
  { if(x3[t2][1]<-grens||x3[t2][1]>grens)
    { g_temp=x3[t2][1]; x3[t2][1]=-x3[t2][0]; x3[t2][0]=-g_temp; }
    if(y3[t2][1]<-grens||y3[t2][1]>grens)
    { g_temp=y3[t2][1]; y3[t2][1]=-y3[t2][0]; y3[t2][0]=-g_temp; }
    if(z3[t2][1]<-grens||z3[t2][1]>grens)
    { g_temp=z3[t2][1]; z3[t2][1]=-z3[t2][0]; z3[t2][0]=-g_temp; }
  }
 }
 for(t2=0;t2<deel;t2++)
 fprintf(bestand, " %f %f %f", x3[t2][1], y3[t2][1], z3[t2][1]);
 for(t2=0;t2<deel;t2++)
 { x2[t2][0]=y3[t2][1]-x3[t2][1];
   y2[t2][0]=-z3[t2][1]+x3[t2][1]/2;
   x2[t2][0]=x_max/2-w_x*x2[t2][0];
   y2[t2][0]=y_max/2-w_y*y2[t2][0];
   putpixel(x2[t2][0],y2[t2][0],2+t2);
 }
 putchar(13); printf("%d",t1+1);
}
```

```
  putchar(13); printf("Klaar");
  fclose(bestand);
  b_test=1;
  getch();
  closegraph();
}

void geendisk(void) // [A]lleen rekenen en tekenen.
{ t1=0; t2=2639;
  installuserdriver("Svga256",DetectSvga256);
  initgraph(&t1,&t2,"");
  w_x=(x_max+1)/(w_g*2); w_y=(y_max+1)/(w_g*2);
  if(modus!=0) diepte=255/(w_g*2);

  printf("     |%d|%d|%f|%f|%f|%f|%f|%d|XXXXXX",
        stap, deel, r_o, r_n, t_frag, grens, w_g, spoor);
  gotoxy(0,0);

  // Co”rdinaten berekenen, schrijven naar grafisch beeldscherm.
  for(t1=0;t1<stap;t1++)
  { for(t2=0;t2<deel;t2++)
    { x3[t2][3]=0;
      y3[t2][3]=0;
      z3[t2][3]=0;
    }
    putchar(13);printf("%d", t1);
    if(midd>0) { midx=0; midy=0; }
    for(t2=0;t2<deel;t2++)
    { x3[t2][2]=x3[t2][1]-x3[t2][0];
      y3[t2][2]=y3[t2][1]-y3[t2][0];
      z3[t2][2]=z3[t2][1]-z3[t2][0];
      for(t3=t2;t3<deel;t3++)
```

```
{ fzx=x3[t3][1]-x3[t2][1];
  fzy=y3[t3][1]-y3[t2][1];
  fzz=z3[t3][1]-z3[t2][1];
  if(fzx!=0) { fzx=1/fzx; x3[t2][3]=x3[t2][3]+fzx;
                        x3[t3][3]=x3[t3][3]-fzx; }
  if(fzy!=0) { fzy=1/fzy; y3[t2][3]=y3[t2][3]+fzy;
                        y3[t3][3]=y3[t3][3]-fzy; }
  if(fzz!=0) { fzz=1/fzz; z3[t2][3]=z3[t2][3]+fzz;
                        z3[t3][3]=z3[t3][3]-fzz; }
}
// Bewerk co”rdinaten als t_frag ongelijk aan 1.
if(t_frag!=1)
{ x3[t2][2]=x3[t2][2]*t_frag; x3[t2][3]=x3[t2][3]*t_frag;
  y3[t2][2]=y3[t2][2]*t_frag; y3[t2][3]=y3[t2][3]*t_frag;
  z3[t2][2]=z3[t2][2]*t_frag; z3[t2][3]=z3[t2][3]*t_frag;
}
// Bepaal de nieuwe co”rdinaten.
x3[t2][0]=x3[t2][1];
y3[t2][0]=y3[t2][1];
z3[t2][0]=z3[t2][1];
x3[t2][1]=x3[t2][0]+x3[t2][2]+x3[t2][3];
y3[t2][1]=y3[t2][0]+y3[t2][2]+y3[t2][3];
z3[t2][1]=z3[t2][0]+z3[t2][2]+z3[t2][3];
// Test grensoverschrijding.
if(grens>0)
{ if(x3[t2][1]<-grens||x3[t2][1]>grens)
  { g_temp=x3[t2][1]; x3[t2][1]=-x3[t2][0]; x3[t2][0]=-g_temp; }
  if(y3[t2][1]<-grens||y3[t2][1]>grens)
  { g_temp=y3[t2][1]; y3[t2][1]=-y3[t2][0]; y3[t2][0]=-g_temp; }
  if(z3[t2][1]<-grens||z3[t2][1]>grens)
  { g_temp=z3[t2][1]; z3[t2][1]=-z3[t2][0]; z3[t2][0]=-g_temp; }
}
```

```
}
for(t2=0;t2<deel;t2++)
{ x2[t2][0]=y3[t2][1]-x3[t2][1];
  y2[t2][0]=-z3[t2][1]+x3[t2][1]/2;
  if(midd>0)
  { midx=midx+x2[t2][0];
    midy=midy+y2[t2][0];
  }
}
if(midd>0) { midx=midx/deel; midy=midy/deel; }
for(t2=0;t2<deel;t2++)
{ if(midd>0)
  { x2[t2][0]=x_max/2+midx-w_x*x2[t2][0];
    y2[t2][0]=y_max/2+midy-w_y*y2[t2][0];
  }
  else
  { x2[t2][0]=x_max/2-w_x*x2[t2][0];
    y2[t2][0]=y_max/2-w_y*y2[t2][0];
  }
  if(modus==0) putpixel(x2[t2][0],y2[t2][0],2+t2);
  else putpixel(x2[t2][0],y2[t2][0],1+(x3[t2][1]+w_g)*diepte);
  if(spoor>0)
  { if(t1<spoor-1)
    { x2[t2][spoor-1-t1]=x2[t2][0]; y2[t2][spoor-1-t1]=y2[t2][0]; }
    else
    { putpixel(x2[t2][spoor-1],y2[t2][spoor-1],0);
      for(t3=spoor-1;t3>0;t3--)
      { x2[t2][t3]=x2[t2][t3-1]; y2[t2][t3]=y2[t2][t3-1]; }
    }
  }
}
putchar(13); printf("%d",t1+1);
```

```
  }
  putchar(13); printf("Klaar");
  getch();
  closegraph();
}

void teken(void)
{ t1=0; t2=2639;
  installuserdriver("Svga256",DetectSvga256);
  initgraph(&t1,&t2,"");

  // Berekenen hoeveelheid pixels per ,,nheid.
  w_x=(x_max+1)/(w_g*2); w_y=(y_max+1)/(w_g*2);

  bestand=fopen(b_naam,"r");
  fscanf(bestand, "%d%d%f%f%f%f", &stap, &deel, &r_o, &r_n, &t_frag, &grens);
  printf("     |%d|%d|%f|%f|%f|%f|%f|%d|%s",
         stap, deel, r_o, r_n, t_frag, grens, w_g, spoor, b_naam);

  if(modus!=0) diepte=255/(w_g*2);
  for(t1=0;t1<stap+1;t1++)
  { putchar(13);printf("%d", t1);
    if(midd>0) { midx=0; midy=0; }
    for(t2=0;t2<deel;t2++)
    fscanf(bestand, "%f%f%f", &x3[t2][1], &y3[t2][1], &z3[t2][1]);
    for(t2=0;t2<deel;t2++)
    { x2[t2][0]=y3[t2][1]-x3[t2][1];
      y2[t2][0]=-z3[t2][1]+x3[t2][1]/2;
      if(midd>0)
      { midx=midx+x2[t2][0];
        midy=midy+y2[t2][0];
      }
```

```
    }
    if(midd>0) { midx=midx/deel; midy=midy/deel; }
    for(t2=0;t2<deel;t2++)
    { if(midd>0)
      { x2[t2][0]=x_max/2+midx-w_x*x2[t2][0];
        y2[t2][0]=y_max/2+midy-w_y*y2[t2][0];
      }
      else
      { x2[t2][0]=x_max/2-w_x*x2[t2][0];
        y2[t2][0]=y_max/2-w_y*y2[t2][0];
      }
      if(modus==0) putpixel(x2[t2][0],y2[t2][0],2+t2);
      else putpixel(x2[t2][0],y2[t2][0],1+(x3[t2][1]+w_g)*diepte);
      if(spoor>0)
      { if(t1<spoor-1)
        { x2[t2][spoor-1-t1]=x2[t2][0]; y2[t2][spoor-1-t1]=y2[t2][0]; }
        else
        { putpixel(x2[t2][spoor-1],y2[t2][spoor-1],0);
          for(t3=spoor-1;t3>0;t3--)
          { x2[t2][t3]=x2[t2][t3-1]; y2[t2][t3]=y2[t2][t3-1]; }
        }
      }
    }
  }
  putchar(13); printf("Klaar");
  fclose(bestand);
  getch();
  closegraph();
}

void main(void)
{ // Beginwaarden zetten.
```

```
stap=250; deel=3; r_o=100; r_n=.0001; t_frag=1; grens=0; w_g=500; spoor=0;
midd=0; spoor=0; modus=0;
clrscr(); textcolor(10);

// Grafische modus bepalen.
t1=0; t2=2639;
installuserdriver("Svga256",DetectSvga256);
initgraph(&t1,&t2,"");
x_max=getmaxx(); y_max=getmaxy();
closegraph();

// Test of standaard bestand bestaat.
strcpy(b_naam, "bestand.xyz");
b_test=0;
varbestand();

// Begin programma-lus.
prog=1;
do
{ menuopbeeld();
  toets=0;
  menuvraag();
  // [b]estandsnaam.
  if(toets==98) varbestand();
  // [R]eken.
  if(toets==82)
  { if(b_test==1)
    { printf("Bestand %s overschrijven? [j/n] ", b_naam);
      i_temp=getch();
      putchar(13); printf("                                        ");
      putchar(13);
      if(i_temp==106) b_test=0;
```

```
    }
    if(b_test==0)
    { willekeur();
      if(grens>0||t_frag!=1) rekenmetopties(); else reken();
    }
  }
  if(toets==69)
  { if(b_test==1)
    { printf("Bestand %s overschrijven? [j/n] ", b_naam);
      i_temp=getch();
      putchar(13); printf("                                        ");
      putchar(13);
      if(i_temp==106) b_test=0;
    }
    if(b_test==0)
    { willekeur();
      rekenenteken();
    }
  }
  // [T]eken.
  if(toets==84)  teken();
  if(toets==65)  { willekeur(); geendisk(); }
  if(toets==81)  theorie();
  if(toets==63)  { printf("Bedacht en geschreven door G.J.Smit.");
                   getch(); }
  if(toets==83)  prog=0;
 } while(prog>0);
}
```

3) Program Einstein (Borland C)

// www.dbphysics.org

// The program 'Einstein' photographs (plots) a piece of einsteinian space time where individual and multiple dimensional basics can be seen, showing the deformation of space time as seen for an outside observer.

```
#include "conio.h"
#include "graphics.h"
#include "math.h"
#include "process.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"

FILE *vkini;                    // Actieve rekenvariabelen.
FILE *vkxyz;                    // Krommingssterkte en virtuele 3D-co”rdinaten.
FILE *vkfilm;                   // Film krommingsverloop.
char fiotest;                   // Menu +/- controle op bestaande bestanden.
char prog;                      // Stuurt programmaverloop.
float xd[24],yd[24],zd[24];     // Co”rdinaten van maximaal 24 1db's.
char deel,dtel,ctel;            // Actieve hoeveelheid 1db's en teller daarvoor en teller voor invoer coordinaten.
float bereik,stap;              // Formaat en resolutie van berekende ruimte-kubus.
float kromming,afstand;         // Sterkte en spreiding van de zichtbare kromming.
char ruimte;                    // Al of niet afbeelden als gekromde ruimte.
float schaal;                   // Grootte van afbeelding op scherm.
char kl_modus;                  // Kleurenpalette/kleurmodus.
float dummy,begin,eind;         // Waarden voor film.
float frag;                     // Voor film krommingsverloop.
char film;                      // Bepaalt film aan/uit in tekenfunctie.
int ftel,fx,fy;                 // Besturing film.
unsigned far fk;                // Besturing film.
char toets;                     // Test op toetsaanslag in menu.
float x,y,z;                    // Actieve rekenco”rdinaten.
```

```
float afx,afy,afz,afs,krm;      // Berekenen krommingssterkten.
float xtot,ytot,ztot;           // Berekenen visuele co”rdinaten.
float ktot;                     // Berekenen totale krommingssterkte per co”rdinaat.
int v1,v2;                      // Instellen video-mode.
float x2,y2;                    // 2D co”rdinaten voor het beeldscherm.
float kleur,midx,midy;          // Kleur van te tekenen pixel + relocatie.
float afst,kl_w;                // Voor tekenen in kl_modus=2.
char c_invoer;                  // Voor invoer co”rdinaten.
char bnaam[13],tnaam[13];       // Voor variabele bestandsnaam

int huge DetectSvga256() { int vid; vid=4; return vid; }

void kleur_mod(void)
{ v1=0;v2=2341;
  installuserdriver("Svga256",DetectSvga256);
  initgraph(&v1,&v2,"");
  midx=getmaxx()/2;midy=getmaxy()/2;
  if(kl_modus==0) { for(dtel=0;dtel<63;dtel++) setrgbpalette(32+dtel,dtel,dtel,dtel); }
  if(kl_modus==1)
  { for(dtel=0;dtel<64;dtel++) setrgbpalette(128+dtel,63-dtel,dtel,0);
    for(dtel=0;dtel<64;dtel++) setrgbpalette(192+dtel,0,63-dtel,dtel);
  }
  if(kl_modus==2)
  { for(dtel=0;dtel<32;dtel++) setrgbpalette(32+dtel,2*dtel,2*dtel,2*dtel);
    for(dtel=0;dtel<32;dtel++) setrgbpalette(64+dtel,2*dtel,0,0);
    for(dtel=0;dtel<32;dtel++) setrgbpalette(96+dtel,0,2*dtel,0);
    for(dtel=0;dtel<32;dtel++) setrgbpalette(128+dtel,0,0,2*dtel);
    for(dtel=0;dtel<32;dtel++) setrgbpalette(160+dtel,2*dtel,2*dtel,0);
    for(dtel=0;dtel<32;dtel++) setrgbpalette(192+dtel,0,2*dtel,2*dtel);
    for(dtel=0;dtel<32;dtel++) setrgbpalette(224+dtel,2*dtel,0,2*dtel);
  }
  setfillstyle(1,0);
```

```
}

void reken(void)
{ _setcursortype(_NOCURSOR);
  strcpy(tnaam,bnaam); strcat(tnaam,".ini");
  vkini=fopen(tnaam,"wb");
  fprintf(vkini,"%d %f %f ",deel,bereik,stap);
  for(dtel=0;dtel<deel;dtel++) fprintf(vkini," %f %f %f",xd[dtel],yd[dtel],zd[dtel]);
  fclose(vkini);

  // Bestand openen voor krommingssterkte en visuele 3D co”rdinaten.
  strcpy(tnaam,bnaam); strcat(tnaam,".xyz");
  vkxyz=fopen(tnaam,"wb");

  // Berekenen krommingssterkten per co”rdinaat per deeltje in kubus.
  for(x=-bereik;x<bereik;x+=stap)
  { gotoxy(14,19); printf(": %5.3f procent",x/(2*bereik)*100+50);
    for(y=-bereik;y<bereik;y+=stap)
    { for(z=-bereik;z<bereik;z+=stap)
      { ktot=0; xtot=0; ytot=0; ztot=0;
        for(dtel=0;dtel<deel;dtel++)
        { afx=(x-xd[dtel])*(x-xd[dtel]);    // Afstand per x,y,z as.
          afy=(y-yd[dtel])*(y-yd[dtel]);
          afz=(z-zd[dtel])*(z-zd[dtel]);
          afs=sqrt(afx+afy+afz);            // Afstand co”rdinaat tot deeltje.
          if(afs!=0) krm=1/(afs*afs); else krm=1000000;  // Krommingssterkte bepalen.
          // Bepalen co”rdinaten voor representatie van visuele ruimte door krommingssterkte.
          ktot+=krm;
          xtot+=(x-xd[dtel])/krm;
          ytot+=(y-yd[dtel])/krm;
          ztot+=(z-zd[dtel])/krm;
        }
```

```
        fprintf(vkxyz,"%f %f %f %f ",ktot,xtot,ytot,ztot);
      }
    }
    if(kbhit()!=0)
    { if(getch()==27) x=bereik;
    }
  }
  fclose(vkxyz);
  if(fiotest==0||fiotest==1) fiotest=1; else fiotest=3;
  if(toets==27)
  { unlink(tnaam);
    if(fiotest==3) fiotest=2; else fiotest=0;
  }
  toets=32;
}

void teken(void)
{ if(film==0)
  { strcpy(tnaam,bnaam); strcat(tnaam,".ini");
    vkini=fopen(tnaam,"rb");
    fscanf(vkini,"%d%f%f",&deel,&bereik,&stap);
    for(dtel=0;dtel<deel;dtel++) fscanf(vkini,"%f%f%f",xd[dtel],yd[dtel],zd[dtel]);
    fclose(vkini);
    printf("Deel:%d Bereik:%f Stap:%f Kromming:%f Afstand:%f Ruimte:%d Kleur:%d Schaal:%f",
           deel,bereik,stap,kromming,afstand,ruimte,kl_modus,schaal);
  }
  else
  { bar(0,0,1023,767);
    gotoxy(1,1);
    printf("FC : Begin:%f Eind:%f Fragment:%f Kromming:%f Afstand:%f Ruimte:%d Kleur:%d Schaal:%f",
           begin,eind,frag,kromming,afstand,ruimte,kl_modus,schaal);
  }
```

```
setcolor(7);line(0,18,1023,18);setcolor(10);
strcpy(tnaam,bnaam); strcat(tnaam,".xyz");
vkxyz=fopen(tnaam,"rb");

for(x=-bereik;x<bereik;x+=stap)
{ line((x/(2*bereik)*100+50)*10.24,18,((x+stap)/(2*bereik)*100+50)*10.24,18);
  for(y=-bereik;y<bereik;y+=stap)
  { for(z=-bereik;z<bereik;z+=stap)
    { fscanf(vkxyz,"%f %f %f",&ktot,&xtot,&ytot,&ztot);
      if(ktot>kromming-afstand&&ktot<kromming+afstand)
      { if(ruimte==0) { xtot=x; ytot=y; ztot=z; }
        x2=(ytot-.5*xtot)*schaal;
        y2=(ztot-.5*xtot)*schaal;
        if(kl_modus==0) kleur=(x+bereik)/(2*bereik)*62+1;
        if(kl_modus==1)
        { if(ktot>kromming&&ktot<(kromming+afstand)) kleur=160+64*(ktot-kromming)/afstand;
          else kleur=160+64*(ktot-kromming)/afstand;
        }
        if(kl_modus==2)
        { afst=1000000;
          kl_w=32;
          for(dtel=0;dtel<deel;dtel++)
          { afx=(x-xd[dtel])*(x-xd[dtel]);
            afy=(y-yd[dtel])*(y-yd[dtel]);
            afz=(z-zd[dtel])*(z-zd[dtel]);
            afs=sqrt(afx+afy+afz);
            if(afst>afs)
            { afst=afs;
              kl_w=32*dtel;
            }
          }
```

```
          kleur=(x+bereik)/(2*bereik)*30+1+kl_w;
        }
        putpixel(midx+x2,midy-y2,32+kleur);
      }
      if(kbhit()!=0)
      { if(getch()==27)
        { if(ruimte==0)
          printf("\n\nOnderbroken  k:%f x:%f y:%f z:%f x2:%f y2:%f",ktot,x,y,z,x2,y2);
          else
          printf("\n\nOnderbroken  k:%f x:%f y:%f z:%f x2:%f y2:%f",ktot,xtot,ytot,ztot,x2,y2);
          x=bereik; y=bereik; z=bereik;
          if(film!=0)  // Als film cre‰ren dan einde beeld-lus.
          { kromming=eind;
            film=0;
          }
        }
      }
    }
  }
}
fclose(vkxyz);
if(film==0)
{ getch();
  toets=32;
}
else
{ for(fy=19;fy<768;fy++)
  { for(fx=0;fx<1024;fx++)
    { fk=getpixel(fx,fy);
      if(fk!=0) fprintf(vkfilm,"%d %d %u ",fx,fy,fk);
    }
  }
```

```
  }
}

void animatie(void)
{ putchar(13);
  printf("KF : Begin:%f Eind:%f Fragment:%f Ruimte:%d Kleur:%d Schaal:%f",
              begin,eind,frag,ruimte,kl_modus,schaal);

  vkfilm=fopen("vkfilm.xyz","rb");

  do
  { fscanf(vkfilm,"%d",&ftel);
    if(ftel<0)
    { if(ftel!=-1)
      { if(ftel==-1000) printf("                    Laatste beeld");
        toets=getch();
        if(toets!=27) bar(0,19,1023,767);
        else toets=32;
      }
      fscanf(vkfilm,"%d",&fx);
    }
    else fx=ftel;
    fscanf(vkfilm,"%d",&fy);
    fscanf(vkfilm,"%u",&fk);
    putpixel(fx,fy,fk);
  } while(ftel>-999);
  fclose(vkfilm);
}

void cord_in(void)  // Nu 24 deeltjes in te voeren
{
  for(dtel=0;dtel<deel;dtel++)
```

```
{ clrscr();
  printf("Bestandsnaa[m] : %s\n\n",bnaam);
  printf("[T]ekenen ");
  if(fiotest==1||fiotest==3) printf("+"); else printf("-");
  printf("  [R]ekenen\n");
  printf("[F]ilm ");
  if(fiotest==2||fiotest==3) printf("+"); else printf("-");
  printf("     [C]re%oren\n");
  printf("[w]illekeur  [i]nvoeren\n");
  printf("[d]eel       : %d\n",deel);
  printf("[b]ereik     : %f\n",bereik);
  printf("[s]tap       : %f\n",stap);
  printf("[k]romming   : %f\n",kromming);
  printf("[a]fstand    : %f\n",afstand);
  printf("[r]uimte     : ");
  if(ruimte==0) printf("lineair\n"); else printf("gekromd\n");
  printf("s[c]haal     : %f\n",schaal);
  printf("k[l]eur      : ");
  if(kl_modus==0) printf("3D grijswaarden\n");
  if(kl_modus==1) printf("2D krommingssterkte\n");
  if(kl_modus==2) printf("3D per deeltje\n");
  printf("b[e]gin      : %f\n",begin);
  printf("ei[n]d       : %f\n",eind);
  printf("[f]ragment   : %f\n",frag);

  for(ctel=0;ctel<dtel;ctel++)
  { gotoxy(38,1+ctel);
    printf("%d : %f %f %f",ctel,xd[ctel],yd[ctel],zd[ctel]);
  }

  gotoxy(1,19); printf("Deel : %d\n",dtel);
  printf("x > "); scanf("%f",&xd[dtel]);
```

```
   printf("y > "); scanf("%f",&yd[dtel]);
   printf("z > "); scanf("%f",&zd[dtel]);
  }
  toets=32;
}

void main(void)
{ // Bepalen van beginwaarden voor actieve rekenvariabelen.
  strcpy(bnaam,"vkveld");
  strcpy(tnaam,bnaam); strcat(tnaam,".ini");
  vkini=fopen(tnaam,"rb");
  if(vkini==NULL)
  { fiotest=0;
    deel=7; bereik=10; stap=.25;
    for(dtel=0;dtel<deel;dtel++)
    { xd[dtel]=((random(32767)*bereik)/32767-bereik/2)*2;
      yd[dtel]=((random(32767)*bereik)/32767-bereik/2)*2;
      zd[dtel]=((random(32767)*bereik)/32767-bereik/2)*2;
    }
  }
  else
  { fiotest=1;
    fscanf(vkini,"%d%f%f",&deel,&bereik,&stap);
    for(dtel=0;dtel<deel;dtel++)
    { fscanf(vkini,"%f%f%f",&xd[dtel],&yd[dtel],&zd[dtel]);
    }
  }
  fclose(vkini);

  // Testen of vkveld.xyz bestaat.
  strcpy(tnaam,bnaam); strcat(tnaam,".xyz");
  vkxyz=fopen(tnaam,"rb");
```

```
if(vkxyz==NULL) fiotest=0;
fclose(vkxyz);

// Bepalen van beginwaarden voor actieve tekenvariabelen.
kromming=1; afstand=.25; schaal=10; ruimte=0; kl_modus=0;

// Bepalen van beginwaarden voor actieve filmvariabelen.
vkini=fopen("vkfilm.ini","rb");
if(vkini==NULL)
{ begin=.5; eind=1.5; frag=.25;
}
else
{ if(fiotest==0) fiotest=2; else fiotest=3;
  fscanf(vkini,"%f%f%f%d",&begin,&eind,&frag,&kl_modus);
}
fclose(vkini);

// Testen of vkfilm.xyz bestaat.
vkfilm=fopen("vkfilm.xyz","rb");
if(vkfilm==NULL) { if(fiotest==1||fiotest==3) fiotest=1; else fiotest=0; }
fclose(vkfilm);

// Menu -> begin programma-lus.
prog=1; do
{ // Menu op het scherm.
  _setcursortype(_NOCURSOR);
  clrscr();
  printf("Bestandsnaa[m] : %s\n\n",bnaam);
  printf("[T]ekenen ");
  if(fiotest==1||fiotest==3) printf("+"); else printf("-");
  printf("  [R]ekenen\n");
  printf("[F]ilm ");
```

```
if(fiotest==2||fiotest==3) printf("+"); else printf("-");

printf("     [C]re‰ren\n");

printf("[w]illekeur  [i]nvoeren\n");

printf("[d]eel       : %d\n",deel);

printf("[b]ereik     : %f\n",bereik);

printf("[s]tap       : %f\n",stap);

printf("[k]romming   : %f\n",kromming);

printf("[a]fstand    : %f\n",afstand);

printf("[r]uimte     : ");

if(ruimte==0) printf("lineair\n"); else printf("gekromd\n");

printf("s[c]haal     : %f\n",schaal);

printf("k[l]eur      : ");

if(kl_modus==0) printf("3D grijswaarden\n");

if(kl_modus==1) printf("2D krommingssterkte\n");

if(kl_modus==2) printf("3D per deeltje\n");

printf("b[e]gin      : %f\n",begin);

printf("ei[n]d       : %f\n",eind);

printf("[f]ragment   : %f\n",frag);


// Bestandsnaam afdrukken




// Co”rdinaten op het scherm.

for(dtel=0;dtel<deel;dtel++)

{ gotoxy(38,1+dtel);

  printf("%d : %f %f %f",dtel,xd[dtel],yd[dtel],zd[dtel]);

}


// Keuze voor functie en afhandeling daarvan.

toets=getch(); gotoxy(1,19); _setcursortype(_NORMALCURSOR);

if(toets==27) prog=0;

if(toets==100)
```

```
{ printf("Aantal 1db's ? "); scanf("%d",&deel);
  if(deel<1) deel=1; if(deel>24) deel=24;
  if(kl_modus==2) kl_modus=0;
}
if(toets==98)
{ printf("Maximale co”rdinaten ? "); scanf("%f",&bereik);
  if(bereik==0) bereik=10; if(bereik<0) bereik=-bereik;
}
if(toets==115)
{ gotoxy(1,19); printf("Resolutie ? "); scanf("%f",&stap);
  if(stap==0) stap=.25; if(stap<0) stap=-stap;
}
if(toets==119)
{ for(dtel=0;dtel<deel;dtel++)
  { xd[dtel]=((random(32767)*bereik)/32767-bereik/2)*2;
    yd[dtel]=((random(32767)*bereik)/32767-bereik/2)*2;
    zd[dtel]=((random(32767)*bereik)/32767-bereik/2)*2;
  }
}
if(toets==105) cord_in();
if(toets==107)
{ printf("Zichtbare kromming ? "); scanf("%f",&kromming);
  if(kromming==0) kromming=1; if(kromming<0) kromming=-kromming;
}
if(toets==97)
{ printf("Afstand tot zichtbare kromming ? "); scanf("%f",&afstand);
  if(afstand==0) afstand=.25; if(afstand<0) afstand=-afstand;
}
if(toets==114)
{ if(ruimte==0) ruimte=1; else ruimte=0;
}
if(toets==99)
```

```
{ printf("Grootte (op 1024x768) ? "); scanf("%f",&schaal);
  if(schaal==0) schaal=10; if(schaal<0) schaal=-schaal;
}
if(toets==108)
{ if(kl_modus==0) kl_modus=1;
  else
  { if(kl_modus==1)
    { if(deel<8) kl_modus=2; else kl_modus=0; }
    else
    { if(kl_modus==2) kl_modus=0; }
  }
}
if(toets==101)
{ printf("Beginkromming ? "); scanf("%f",&begin);
  if(begin<0) begin=-begin;
}
if(toets==110)
{ printf("Eindkromming ? "); scanf("%f",&eind);
  if(eind<0) eind=-eind;
}
if(toets==102)
{ printf("Fragmentatie ? "); scanf("%f",&frag);
}

if(toets==109)
{ printf("Nieuwe bestandsnaam ? "); scanf("%s",&bnaam);

  // Testen of bnaam.ini bestaat.
  strcpy(tnaam,bnaam); strcat(tnaam,".ini");
  vkini=fopen(tnaam,"rb");
  if(vkini==NULL)
  { fiotest=0;
```

```
   // deel=7; bereik=10; stap=.25;
   // for(dtel=0;dtel<deel;dtel++)
   // { xd[dtel]=((random(32767)*bereik)/32767-bereik/2)*2;
   //   yd[dtel]=((random(32767)*bereik)/32767-bereik/2)*2;
   //   zd[dtel]=((random(32767)*bereik)/32767-bereik/2)*2;
   // }
  }
  else
  { fiotest=1;
    fscanf(vkini,"%d%f%f",&deel,&bereik,&stap);
    for(dtel=0;dtel<deel;dtel++)
    { fscanf(vkini,"%f%f%f",&xd[dtel],&yd[dtel],&zd[dtel]);
    }
  }
  fclose(vkini);

  // Testen of bnaam.xyz bestaat.
  strcpy(tnaam,bnaam); strcat(tnaam,".xyz");
  vkxyz=fopen(tnaam,"rb");
  if(vkxyz==NULL) fiotest=0;
  fclose(vkxyz);
 }
if(toets==82)
{ if(fiotest==1||fiotest==3)
  { printf("Bestand overschrijven (j/n) ? ");
    toets=getch();
    if(toets==106)
    { putchar(13); printf("                              ");
      putchar(13); printf("Rekenen");
      reken(); toets=32;
    }
  }
```

```
  else
  { putchar(13); printf("Rekenen");
    reken();
  }
 }
if(toets==84)
{ if(fiotest==0||fiotest==2)
  { printf("Bestand bestaat niet ! "); getch();
  }
  else
  { film=0;
    kleur_mod();
    teken();
    closegraph();
  }
 }
if(toets==67)  // Film cre‰ren.
{ if(fiotest==3)  // Test of vkfilm.xyz bestaat.
  { printf("Bestand overschrijven (j/n) ? ");
    toets=getch();
    if(toets==106) fiotest=1;
  }
  if(fiotest==1)
  { // Kijk naar variabelen voor bereik & stap.
    vkini=fopen("vkveld.ini","rb");
    fscanf(vkini,"%d%f%f",&deel,&bereik,&stap);
    for(dtel=0;dtel<deel;dtel++)
    { fscanf(vkini,"%f%f%f",&xd[dtel],&yd[dtel],&zd[dtel]);
    }
    fclose(vkini);

    // Begin beeld-lus.
```

```
    kleur_mod();
    vkini=fopen("vkfilm.ini","wb");
    fprintf(vkini,"%f %f %f %d",begin,eind,frag,kl_modus);
    fclose(vkini);
    vkfilm=fopen("vkfilm.xyz","wb");
    film=1; ftel=1;
    for(kromming=begin;kromming<eind;kromming+=frag)
    { fprintf(vkfilm,"-%d ",ftel);
      teken();
      ftel++;
    }
    fprintf(vkfilm,"-1000");
    fclose(vkfilm);
    fiotest=3;
    if(film==0)
    { fiotest=1;
      unlink("vkfilm.xyz");
    }
    closegraph();
  }
  if(fiotest==0||fiotest==2)
  { printf("Bestand bestaat niet ! "); getch();
  }
  toets=32;
 }
if(toets==70)
{ if(fiotest==3)
  { vkini=fopen("vkveld.ini","rb");
    fscanf(vkini,"%d%f%f",&deel,&bereik,&stap);
    for(dtel=0;dtel<deel;dtel++)
    { fscanf(vkini,"%f%f%f",&xd[dtel],&yd[dtel],&zd[dtel]);
    }
```

```
   fclose(vkini);
   kleur_mod();
   animatie();
   closegraph();
  }
  if(fiotest==2)
  { kleur_mod();
   animatie();
   closegraph();
  }
  if(fiotest<2)
  { printf("Bestand bestaat niet ! "); getch();
  }
  toets=32;
 }
if(toets==63)
{ for(dtel=1;dtel<15;dtel++)
  { gotoxy(1,dtel); printf("                              ");
  }
  clrscr();
  printf("( 04-96, Nijmegen -> G.J.Smit  Geb:08-01-68, Veendam )\n");
  printf("      De theorie in hoeverre het betrekking heeft op dit programma. Dit pro-\n");
  printf("gramma geeft een stilstaande foto van een aantal 1db's in een bepaalde ruimte,\n");
  printf("op een bepaalde afstand van elkaar. Er is geen beweging, alleen hun onderlinge\n");
  printf("positie. Dit programma berekend een kubus ruimte met een bepaalde grootte met\n");
  printf("als middelpunt x,y,z-co”rdinaat (0,0,0). De 1db's hebben een bepaalde locatie\n");
  printf("met elk zijn specifieke co”rdinaten in die kubus. Op het punt waar de 1db zich\n");
  printf("bevindt is de ruimte oneindig sterk gekromd (de 1db • s de gekromde ruimte!).\n");
  printf("Op een bepaalde afstand van een 1db is de ruimte de in die mate gekromd als\n");
  printf("bepaalt door 1/afstand (of waarschijnlijker 1/afstand^2 zoals wij dat waar-\n");
  printf("nemen bij o.a. de zwaartekracht) (en logisch met een 3-dimensionaal heelal\n");
  printf("gezien de 'localiteit' van energie (energie=mate van kromming t.o.v. omgeving)?\n");
```

```
    printf("Een 1db is net als alle omliggende ruimte, alleen is in dat punt de ruimte ten\n");
    printf("opzichte van de omgeving een factor oneindig sterker gekromd. Het is allemaal\n");
    printf("puur relatief. Want ook al is de omliggende ruimte op een bepaalde afstand\n");
    printf("bijvoorbeeld de helft van de 1db kromming, dan is die ruimte daar ook oneindig\n");
    printf("gekromd (0.5*i=i). Er is dus alleen sprake van een relatief sterkere kromming,\n");
    printf("slechts absoluut, en eindig in verhouding tot andere 1db's. Dat wil zeggen;\n");
    printf("meetbare parameters zijn onderlinge afstand en onderlinge snelheid. Ze zijn\n");
    printf("allen oneindig, doch meetbaar eindig in verhouding tot elkaar Met welke\n");
    printf("snelheid naderen ze elkaar en welke afstand hebben ze tot elkaar? In principe\n");
    printf("meetbaar, in elk geval berekenbaar. ");
    getch(); clrscr();
    printf("Technische opmerkingen:\n\n");
    printf("Bereik: Kleiner dan .01 en groter dan 100 levert onbetrouwbare resultaten op.\n");
    printf("       Dit door het maximum bereik van 'float-type' variabelen.\n\n");
    printf("Schaal: De schaal is alleen re‰el in lineaire-ruimte afbeelding. In gekromde-\n");
    printf("       ruimte afbeelding is de grootte het resultaat van een algoritme dat\n");
    printf("       bepaalt in hoeverre de kromming van de 1db's de in 3D berekende kubus\n");
    printf("       van vlakke ruimte vervormd. ");

    getch();
   }
  } while (prog>0);
}
```

Gerhard Jan Smit, Jelle Ebel van der Schoot, 23 November 2017, Nijmegen, Nederland

www.dbphysics.org