Computing Forum MELK 0x07E3

Christian B. Hübschle

I‘m not a real programmer

- Real programmers write some code and debug it later.
- I‘m write bugs and decode them later.

float	char[4]	int	hex
789.145630	RIED	1145391442	0x44454952
841.020020	HARD	1146241352	0x44524148
53829.312500	PERG	1196574032	0x47524550
199957.187500	LECH	1212368204	0x4843454c
13387085.000000	MELK	1263289677	0x4b4c454d
53544224.000000	HALL	1280065864	0x4c4c4148
827479488.000000	WIEN	1313163607	0x4e454957
882102784.000000	HORN	1314017096	0x4e524f48
834336784384.000000	YBBS	1396851289	0x53424259
877336657920.000000	WELS	1397507415	0x534c4557
886076407808.000000	ENNS	1397640773	0x534e4e45
3630138916864.000000	IMST	1414745417	0x54534d49
3630678147072.000000	RUST	1414747474	0x54535552
13603783850328064.000000	GRAZ	1514230343	0x5a415247
14163177570828288.000000	WEIZ	1514751319	0x5a494557
14516108992184320.000000	LINZ	1515080012	0x5a4e494c
14937225166848000.000000	OETZ	1515472207	0x5a54454f
14937228388073472.000000	RETZ	1515472210	0x5a544552

```
          3428.956055 LOVE   1163284300
          3396.080078 HATE   1163149640
3320569397248.000000 WHAT   1413564503
          3205.257812  THE   1162368032
      12801350.000000 FUCK   1262703942
          3252.955566 JOKE   1162563402
          3252.079590 FAKE   1162559814
 924580708352.000000 NEWS   1398228302
      12928343.000000 WEEK   1262830935
     869520192.000000 MOON   1313820493
     894748672.000000  SUN   1314214688
          3300.956543 NONE   1162760014
 885926330368.000000 SENS   1397638483
3458007302144.000000 SHIT   1414088787
```

```c
#include <stdlib.h>
#include <stdio.h>
#include<string.h>

union test{
  float f;
  char ch[4];
  int ii;
}test;

int main(int argc, char **argv){
  test.ch[0]='E';
  test.ch[1]='S';
  test.ch[2]='E';
  test.ch[3]='L';
  int i=0;
  if ((argc>1)&&(strlen(argv[1])>3)){
    for (i=0; i<4; i++)test.ch[i]=argv[1][i];
  }
  printf("%25.6f %4s %11d %x\n",test.f,test.ch,test.ii,test.ii);
  return 0;
}
```

Language Perl

(bottled by Acme::EyeDrops)

Date: 06/04/05
Author: Andrew Savige
URL: n/a
Comments: 76
Info: http://search.cpan.org/dist/Acme-EyeDrops/lib/Acme/EyeDrops.pm
Score: ★★★½ (3.49 in 2780 votes)

```
''=~(     '(?{'     .('`'     |'&')     .('['     ^'-')
.('`'     |'!')     .('`'     |',')     .'"'.     '\$'
.'=='     .('['     ^'+')     .('`'     |'/')     .('['
^'+')     .'||'     .(';'     &'=')     .(';'     &'=')
.';-'     .'-'.     '\$'      .'=;'     .('['     ^'()')
.('['     ^'.')     .('`'     |'"')     .('!'     ^'+')
.'_\\{'   .'(\$'    .';=('.   '\$=|'    ."\|".(   '`'^'.'
).(('`')| '/').').' .'\"'.+(  '{'^'['). ('`'|'"') .('`'|'/'
).('['^'/') .('['^'/'). ('`'|',').( '`'|('&')). '\".\"'.( '['^('()).
'\"'.('['^  '#').'!!--' .'\$=.\"'  .('{'^'['). ('`'|'/').( '`'|"\&").(
'{'^"\[").( '`'|"\"").( '`'|"\&").( '`'|"\&").( '['^(')')). '\").\"'.
('{'^'['). ( '`'|"\/"). ( '`'|"\."). ( '{'^"\["). ( '['^"\/"). ( '`'|"\("). (
'`'|"\&"). ( '{'^"\["). ( '['^"\,"). ( '`'|"\!"). ( '`'|"\,"). ( '`'|(',')).
'\\"\\}'.+( '['^"\+"). ( '['^"\)"). ( '`'|"\)"). ( '`'|"\."). ( '['^('/')).
'+_,\\",'.( '{'^('[')). ('\\$;!'). ( '!'^"\+"). ( '{'^"\/"). ( '`'|"\!"). (
'`'|"\+"). ( '`'|"\&"). ( '{'^"\["). ( '`'|"\/"). ( '`'|"\."). ( '`'|"\&"). (
'{'^"\["). ( '`'|"\$"). ( '`'|"\/"). ( '['^"\,"). ( '`'|('.')). ',.'(('{')^
'['). ("\["^ '+'). ("\`"| '!'). ("\["^ '('). ("\["^ '('). ("\{"^ '['). ("\`"|
')'). ("\["^ '/'). ("\{"^ '['). ("\`"| '!'). ("\["^ ')'). ("\`"| '/'). ("\["^
'.'). ("\`"| '.'). ("\`"| '$'). "\,". ( '!'^('+')). '\\",_,\\"' .'!'. ("\!"^
'+'). ("\!"^ '+'). '\\"'. ('['^','). ( '`'|"\("). ( '`'|"\)"). ( '`'|"\,"). (
'`'|('&')). '++\\$="})' );$:=('.')^ '~';$~='@'| '(';$^=')'^ '[';$/='`';
```

Download Source | Write Comment

Alternative Versions

Version	Author	Date	Comments	Rate

Voting

Please select Vote

Category

real language

Bookmarking

Language Python

(Using a iterator class)

Date: 01/20/06
Author: Eric Moritz
URL: http://eric.themoritzfamily.com
Comments: 2
Info: http://www.python.org
Score: ★★★ (2.67 in 43 votes)

```
class Song:
    def __init__(self):
        self.n = 100

    def __iter__(self):
        return self

    def next(self):
        if self.n == 0:
            raise StopIteration
        self.n = self.n - 1
        return self._get_verse(self.n)

    def _get_verse(self,n):
        lines = []
        data  = []
        if n > 1:
            data.append(["%s bottles" % n,"%s bottles" % n])
            data.append(["Take one down pass it around","%s bottles" % (n-1)])
        elif n == 1:
            data.append(["%s bottle" % n,"%s bottle" % n])
            data.append(["Take one down pass it around","%s bottles" % "no more"])
        elif n == 0:
            data.append(["No more bottles","no more bottles"])
            data.append(["Go to the store and buy some more","%s bottles" % 99])

        lines.append("%s of beer on the wall, %s of beer" % tuple(data[0]))
        lines.append("%s, %s of beer on the wall" % tuple(data[1]))
        lines.append("")
        return "\n".join(lines)

song = Song()
for stanza in song:
    print stanza
```

Download Source | Write Comment

Voting

Please select Vote

Category

real language

Bookmarking

Language FORTRAN-IV

(This version really works.)

Date: 09/28/10
Author: Dave Pitts
URL: n/a
Comments: 0
Info: n/a
Score: ★★★ (3.00 in 24 votes)

```
C
C FORTRAN-IV VERSION OF 99 BOTTLES OF BEER
C DAVE PITTS, DPITTS AT COZX.COM
C
      DO 30 J = 1, 98
      I = 100 - J
      WRITE (6,100) I, I
      WRITE (6,110)
      I = I - 1
      IF (I .GE. 2) GO TO 20
      WRITE (6,125) I
      GO TO 30
20    WRITE (6,120) I
30    CONTINUE
      I = 1
      WRITE (6,105) I, I
      WRITE (6,110)
      WRITE (6,130)
      CALL EXIT
C
100   FORMAT (1H0,I2,30H BOTTLES OF BEER ON THE WALL,
     1,I2,16H BOTTLES OF BEER)
105   FORMAT (1H0,I2,29H BOTTLE OF BEER ON THE WALL,
     1,I2,15H BOTTLE OF BEER)
110   FORMAT (33H TAKE ONE DOWN AND PASS IT AROUND)
120   FORMAT (1H ,I2,17H BOTTLES OF BEER.)
125   FORMAT (1H ,I2,16H BOTTLE OF BEER.)
130   FORMAT (20H NO BOTTLES OF BEER.)
      END
```

Download Source | Write Comment

Alternative Versions

Voting

Please select Vote

Category

real language

Bookmarking

Language C++

(object-oriented version)

Date: 04/20/05
Author: Tim Robinson
URL: n/a
Comments: 4
Info: n/a
Score: ★★★ (2.95 in 20 votes)

```
// C++ version of 99 Bottles of Beer, object oriented paradigm
// programmer: Tim Robinson timtroyr@ionet.net
#include <fstream.h>

enum Bottle { BeerBottle };

class Shelf {
    unsigned BottlesLeft;
public:
    Shelf( unsigned bottlesbought )
        : BottlesLeft( bottlesbought )
        {}
    void TakeOneDown()
        {
        if (!BottlesLeft)
            throw BeerBottle;
        BottlesLeft--;
        }
    operator int () { return BottlesLeft; }
    };

int main( int, char ** )
    {
    Shelf Beer(99);
    try {
        for (;;) {
            char *plural = (int)Beer !=1 ? "s" : "";
            cout << (int)Beer << " bottle" << plural
                 << " of beer on the wall," << endl;
            cout << (int)Beer << " bottle" << plural
                 << " of beer," << endl;
            Beer.TakeOneDown();
            cout << "Take one down, pass it around," << endl;
            plural = (int)Beer !=1 ? "s":"";
            cout << (int)Beer << " bottle" << plural
                 << " of beer on the wall." << endl;
            }
        }
    catch ( Bottle ) {
        cout << "Go to the store and buy some more," << endl;
        cout << "99 bottles of beer on the wall." << endl;
        }
    return 0;
    }
```

Download Source | Write Comment

Voting

Please select Vote

Category

real language

Bookmarking

Chose your language…;

	Fortran	Python	Perl	C/C++
Whitespace matters? [space,tab,newline]	Yes	Yes	No	No
Keyword arguments	Yes	Yes	Optional	No
Semicolon as delimiter	No	No	Yes;	Yes;
Operator overloading	No	Somehow	Somehow	**Yes!!**

Operator overloading?

```
8  struct V3 {
9    double x//! x is the X coordinate
10       ,y//! y is the Y coordinate
11       ,z//! z is the Z coordinate
12       ;
13 //  int rc;
14   inline V3( void ){}
15   inline V3( const double& _x, const double& _y, const double& _z ) :
16   x(_x), y(_y), z(_z)//!< initializer
17         //,rc(0)
18   {
19     ;
20   }
21   inline V3& operator *= ( const double& d ){
22     x *= d;
23     y *= d;
24     z *= d;
25     return *this;
26   }//!< The *= operator to scale by a scalar
27   inline V3& operator += ( const V3& v ){
28     x += v.x;
29     y += v.y;
30     z += v.z;
31     return *this;
32   }//!< The += operator to add a V3
33   inline V3& operator += ( const double& v ){
34     x += v;
35     y += v;
36     z += v;
37     return *this;
38   }//!< The += operator to add a scalar
39 };
40 inline V3 operator + ( const V3& v1, const V3& v2 ) {
41   V3 t;
42   t.x = v1.x + v2.x;
43   t.y = v1.y + v2.y;
44   t.z = v1.z + v2.z;
45   return t;
46 }//!< The + operator to add two V3
47 inline V3 operator - ( const V3& v1, const V3& v2 ) {
48   V3 t;
49   t.x = v1.x - v2.x;
```

$$\vec{c} = \vec{a} \times \vec{b}$$

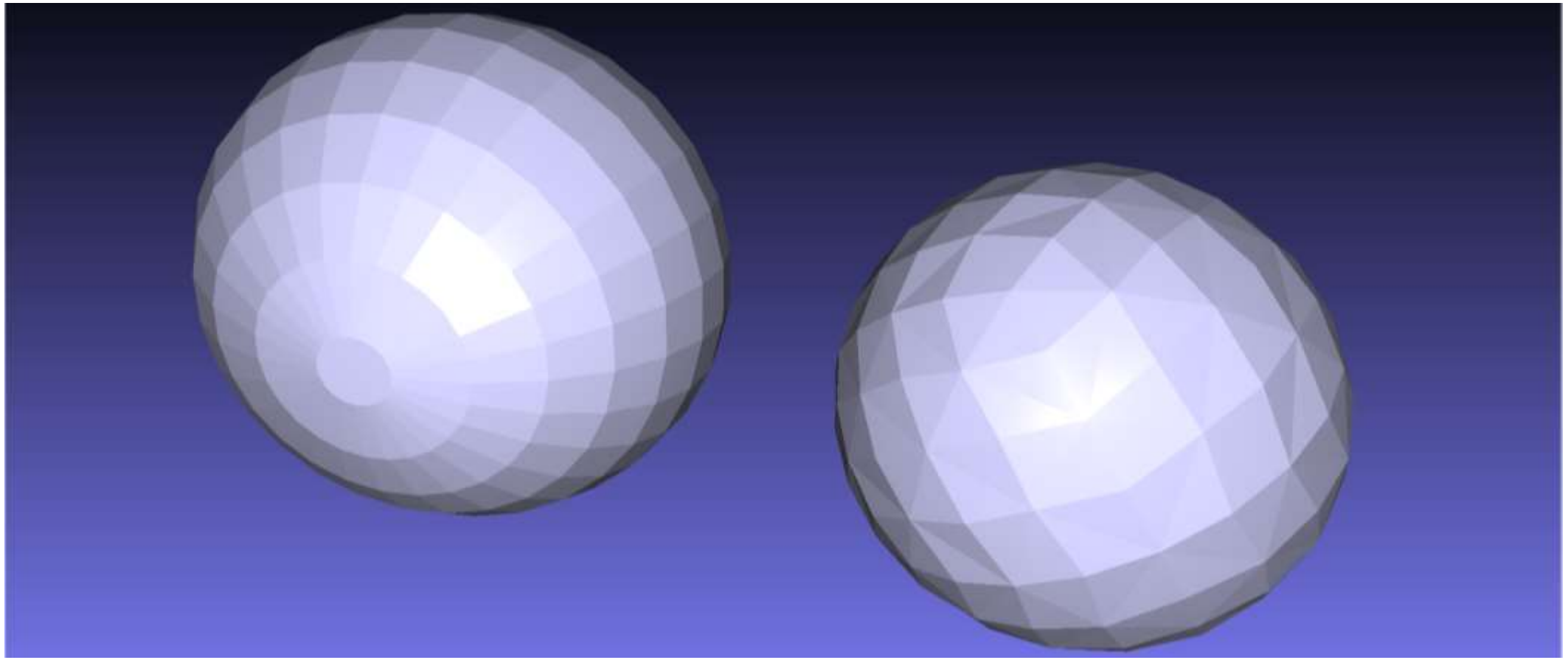
```
c = a % b;
```

$$d = \vec{a} \cdot \vec{b}$$

```
d = a * b;
```

```
47 inline V3 operator - ( const V3& v1, const V3& v2 ) {
48   V3 t;
49   t.x = v1.x - v2.x;
50   t.y = v1.y - v2.y;
51   t.z = v1.z - v2.z;
52   return t;
53 }//!< The + operator to subtract two V3
54 inline V3 operator * ( const V3& v, const double& d ) {
55   V3 t;
56   t.x = v.x*d;
57   t.y = v.y*d;
58   t.z = v.z*d;
59   return t;
60 }//!< The * to scale a V3
61 inline V3 operator * ( const double& d, const V3& v ) {
62   V3 t;
63   t.x = v.x*d;
64   t.y = v.y*d;
65   t.z = v.z*d;
66   return t;
67 }//!< The * to scale a V3
68 inline V3 operator % ( const V3& v1, const V3& v2 ) {
69   V3 t;
70   t.x = v1.y*v2.z - v2.y*v1.z;
71   t.y = v1.z*v2.x - v2.z*v1.x;
72   t.z = v1.x*v2.y - v2.x*v1.y;
73   return t;
74 }//!< The % operator the cross product of two V3
75 inline double operator * ( const V3& v1, const V3& v2 ) {
76   return v1.x*v2.x + v1.y*v2.y + v1.z*v2.z;
77 }//!< The * operator the scalar product of two V3
78 inline double Norm( const V3& v ) {
79   return v.x*v.x + v.y*v.y + v.z*v.z;
80 }//!< The squared lenght of a V3
81 inline double Distance( const V3& v1, const V3& v2 ) {
82   return Norm(v1 - v2);
83 }//!< The squared distance between two V3
84 inline bool operator == (const V3& v1, const V3& v2 ) {
85   //  return ((v1.x==v2.x)&&(v1.y==v2.y)&&(v1.z==v2.z));
86   return (Distance(v1,v2)<0.001);
87 }
```

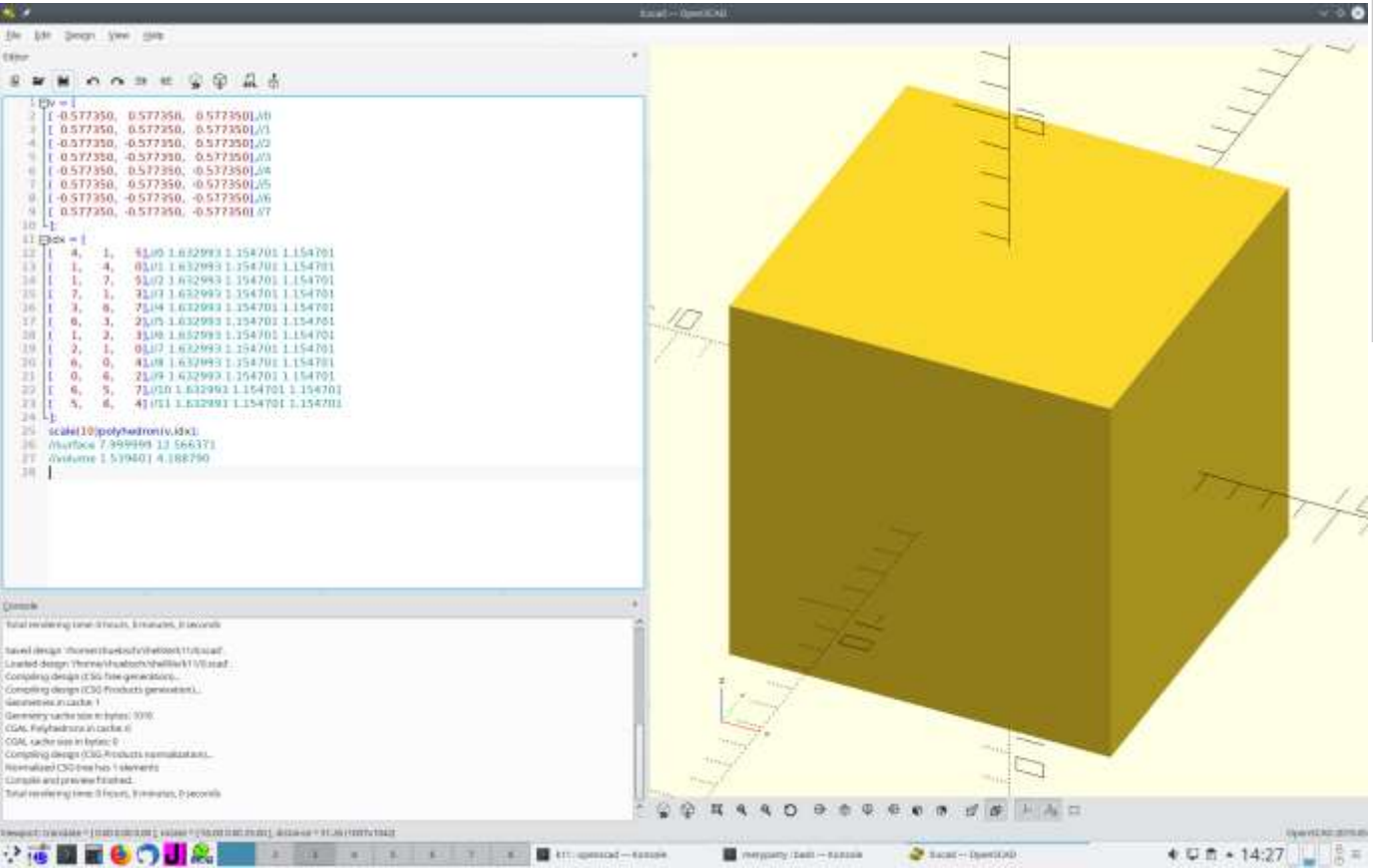
Sphere in 3D



Subdivison

```
QVector<QVector3D> vertz;
QVector<int> indexz;

void correctHand(){
  int dm;
  for (int i = 0; i < indexz.size(); i+=3){
    QVector3D a=a.crossProduct(vertz.at(indexz.at(i))-vertz.at(indexz.at(i+1)),vertz.at(indexz.at(i+2))-vertz.at(indexz.at(i+1)));
    float n=a.dotProduct(vertz.at(indexz.at(i)),a);
    if (n>0) {
      dm = indexz.at(i+1);
      indexz[i+1] = indexz.at(i);
      indexz[i] = dm;
    }
  }
}
```



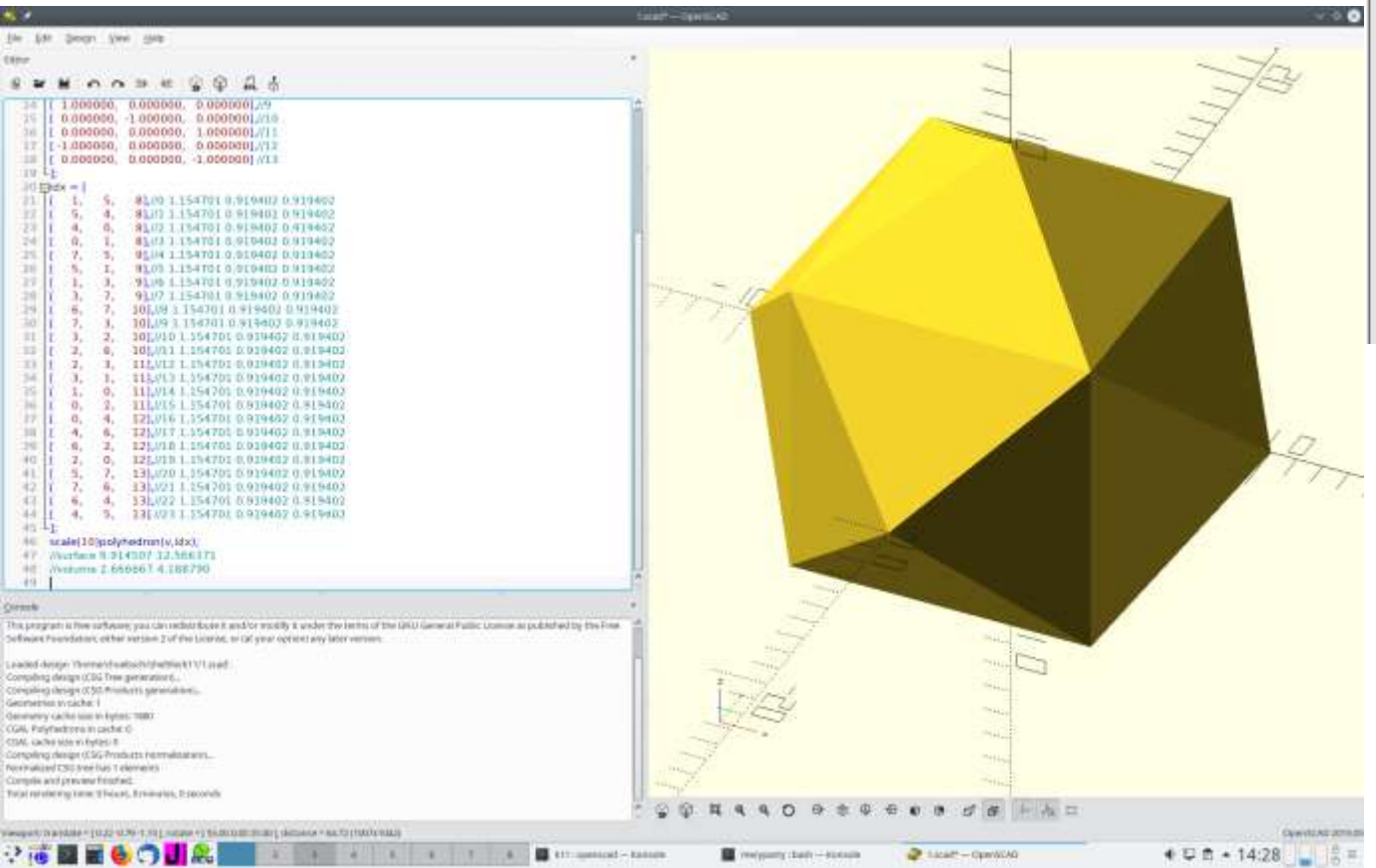
```
void subdivide(){
//  float l = vertz.at(0).length();
  int k = indexz.size();
  QVector<int> indexy;
  for (int i = 0; i<k; i+=3){
    int z=vertz.size();
    QVector3D mid =  (vertz.at(indexz.at(i)) + vertz.at(indexz.at(i+1)))* 0.5f;
    mid.normalize();
  //  mid*=l;
    if (mid == vertz.last()) {
      indexy<<indexz.at(i+1)<<indexz.at(i+2)<<z-1;
      indexy<<indexz.at(i)<<indexz.at(i+2)<<z-1;
    }else{
      indexy<<indexz.at(i+1)<<indexz.at(i+2)<<z;
      indexy<<indexz.at(i)<<indexz.at(i+2)<<z;
      vertz << mid;
    }
  }
  indexz.clear();
  indexz=indexy;
  correctHand();
}
```

```
  vertz << QVector3D(-0.5f,  0.5f,  0.5f)<<
    QVector3D( 0.5f,  0.5f,  0.5f)<<
    QVector3D(-0.5f, -0.5f,  0.5f)<<
    QVector3D( 0.5f, -0.5f,  0.5f)<<
    QVector3D(-0.5f,  0.5f, -0.5f)<<
    QVector3D( 0.5f,  0.5f, -0.5f)<<
    QVector3D(-0.5f, -0.5f, -0.5f)<<
    QVector3D( 0.5f, -0.5f, -0.5f);
  for (int i = 0; i < vertz.size(); i++){
    vertz[i]= vertz.at(i).normalized();
  }
  indexz<<
    1<<4<<5<<    //bef
    4<<1<<0<<    //eba
    1<<7<<5<<    //bhf
    7<<1<<3<<    //hbd
    3<<6<<7<<    //dgh
    6<<3<<2<<    //gdc
    1<<2<<3<<    //bcd
    2<<1<<0<<    //cba
    0<<6<<4<<    //age
    6<<0<<2<<    //gac
    5<<6<<7<<    //fgh
    6<<5<<4;     //gfe
  correctHand();
```

Subdivison

```
QVector<QVector3D> vertz;
QVector<int> indexz;

void correctHand(){
  int dm;
  for (int i = 0; i < indexz.size(); i+=3){
    QVector3D a=a.crossProduct(vertz.at(indexz.at(i))-vertz.at(indexz.at(i+1)),vertz.at(indexz.at(i+2))-vertz.at(indexz.at(i+1)));
    float n=a.dotProduct(vertz.at(indexz.at(i)),a);
    if (n>0) {
      dm = indexz.at(i+1);
      indexz[i+1] = indexz.at(i);
      indexz[i] = dm;
    }
  }
}
```

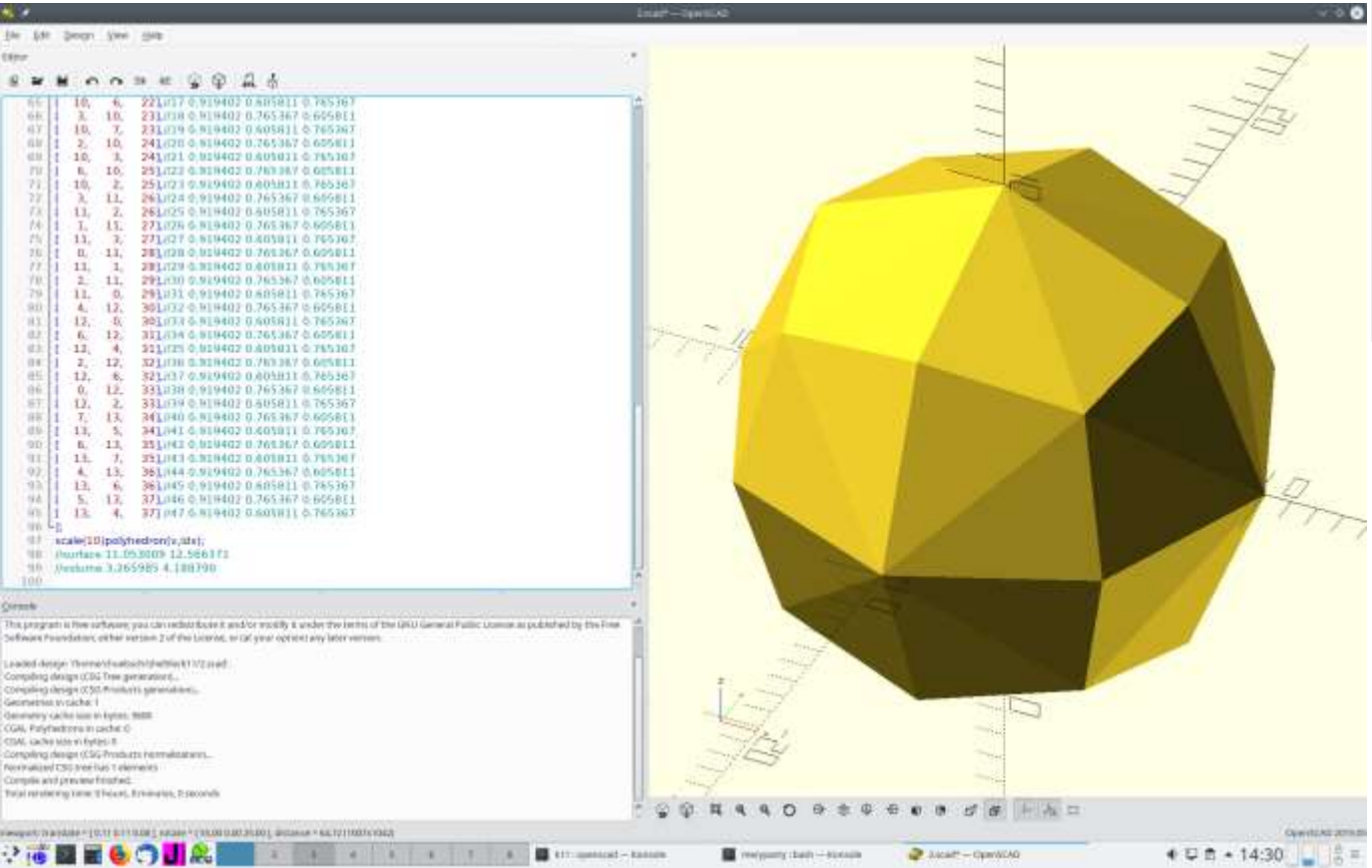


```
void subdivide(){
//  float l = vertz.at(0).length();
  int k = indexz.size();
  QVector<int> indexy;
  for (int i = 0; i<k; i+=3){
    int z=vertz.size();
    QVector3D mid =  (vertz.at(indexz.at(i)) + vertz.at(indexz.at(i+1)))* 0.5f;
    mid.normalize();
  //  mid*=l;
    if (mid == vertz.last()) {
      indexy<<indexz.at(i+1)<<indexz.at(i+2)<<z-1;
      indexy<<indexz.at(i)<<indexz.at(i+2)<<z-1;
    }else{
      indexy<<indexz.at(i+1)<<indexz.at(i+2)<<z;
      indexy<<indexz.at(i)<<indexz.at(i+2)<<z;
      vertz << mid;
    }
  }
  indexz.clear();
  indexz=indexy;
  correctHand();
}
```

```
  vertz << QVector3D(-0.5f,  0.5f,  0.5f)<<
    QVector3D( 0.5f,  0.5f,  0.5f)<<
    QVector3D(-0.5f, -0.5f,  0.5f)<<
    QVector3D( 0.5f, -0.5f,  0.5f)<<
    QVector3D(-0.5f,  0.5f, -0.5f)<<
    QVector3D( 0.5f,  0.5f, -0.5f)<<
    QVector3D(-0.5f, -0.5f, -0.5f)<<
    QVector3D( 0.5f, -0.5f, -0.5f);
  for (int i = 0; i < vertz.size(); i++){
    vertz[i]= vertz.at(i).normalized();
  }
  indexz<<
    1<<4<<5<<    //bef
    4<<1<<0<<    //eba
    1<<7<<5<<    //bhf
    7<<1<<3<<    //hbd
    3<<6<<7<<    //dgh
    6<<3<<2<<    //gdc
    1<<2<<3<<    //bcd
    2<<1<<0<<    //cba
    0<<6<<4<<    //age
    6<<0<<2<<    //gac
    5<<6<<7<<    //fgh
    6<<5<<4;     //gfe
  correctHand();
```

Subdivison

```
8   QVector<QVector3D> vertz;
9   QVector<int> indexz;
10
11  void correctHand(){
12    int dm;
13    for (int i = 0; i < indexz.size(); i+=3){
14      QVector3D a=a.crossProduct(vertz.at(indexz.at(i))-vertz.at(indexz.at(i+1)),vertz.at(indexz.at(i+2))-vertz.at(indexz.at(i+1)));
15      float n=a.dotProduct(vertz.at(indexz.at(i)),a);
16      if (n>0) {
17        dm = indexz.at(i+1);
18        indexz[i+1] = indexz.at(i);
19        indexz[i] = dm;
20      }
21    }
22  }
```



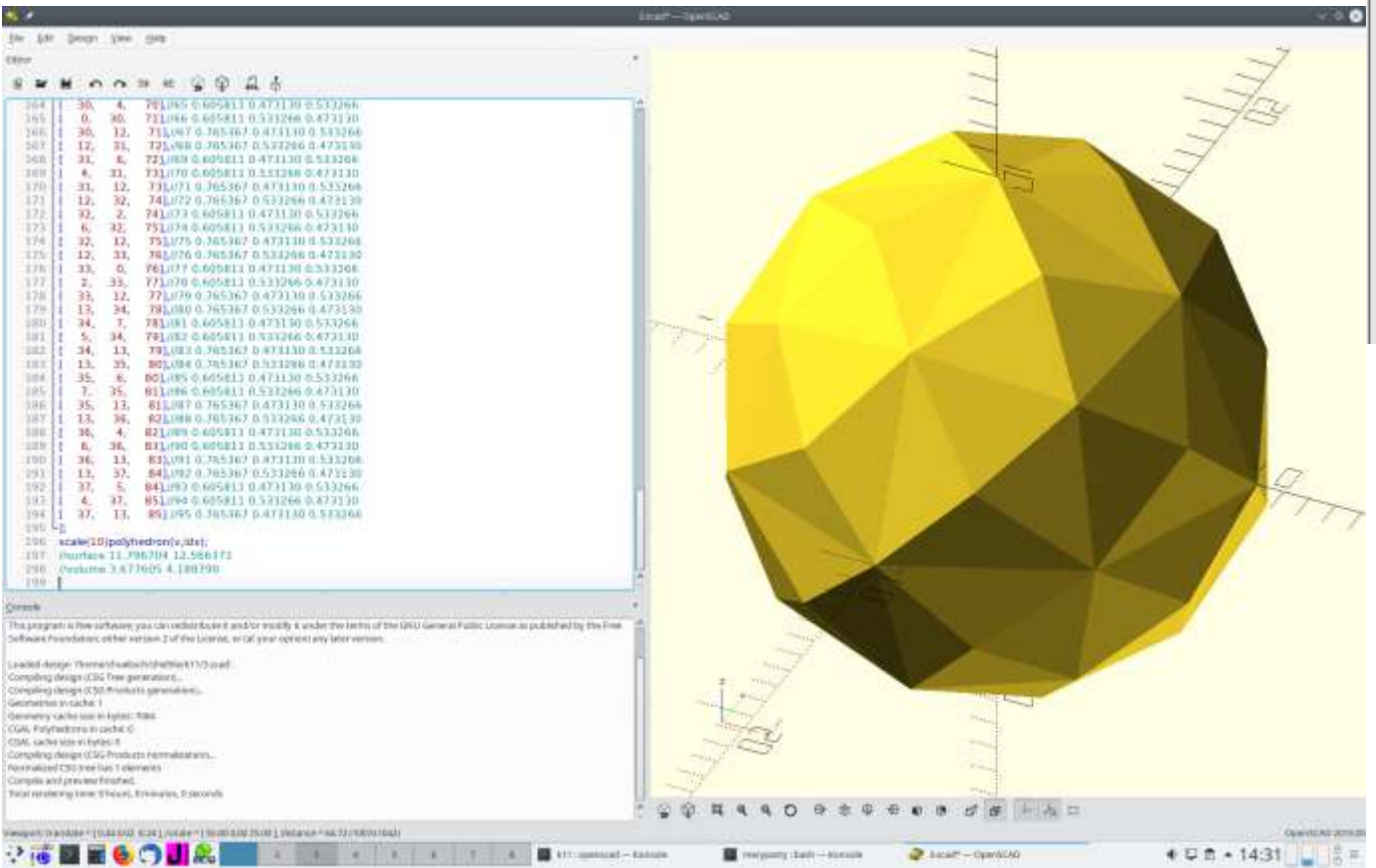
```
73  void subdivide(){
74  //  float l = vertz.at(0).length();
75    int k = indexz.size();
76    QVector<int> indexy;
77    for (int i = 0; i<k; i+=3){
78      int z=vertz.size();
79      QVector3D mid =  (vertz.at(indexz.at(i)) + vertz.at(indexz.at(i+1)))* 0.5f;
80      mid.normalize();
81    //  mid*=l;
82      if (mid == vertz.last()) {
83        indexy<<indexz.at(i+1)<<indexz.at(i+2)<<z-1;
84        indexy<<indexz.at(i)<<indexz.at(i+2)<<z-1;
85      }else{
86        indexy<<indexz.at(i+1)<<indexz.at(i+2)<<z;
87        indexy<<indexz.at(i)<<indexz.at(i+2)<<z;
88        vertz << mid;
89      }
90    }
91    indexz.clear();
92    indexz=indexy;
93    correctHand();
94  }
```

```
25    vertz << QVector3D(-0.5f,  0.5f,  0.5f)<<
26      QVector3D( 0.5f,  0.5f,  0.5f)<<
27      QVector3D(-0.5f, -0.5f,  0.5f)<<
28      QVector3D( 0.5f, -0.5f,  0.5f)<<
29      QVector3D(-0.5f,  0.5f, -0.5f)<<
30      QVector3D( 0.5f,  0.5f, -0.5f)<<
31      QVector3D(-0.5f, -0.5f, -0.5f)<<
32      QVector3D( 0.5f, -0.5f, -0.5f);
33    for (int i = 0; i < vertz.size(); i++){
34      vertz[i]= vertz.at(i).normalized();
35    }
36    indexz<<
37      1<<4<<5<<    //bef
38      4<<1<<0<<    //eba
39      1<<7<<5<<    //bhf
40      7<<1<<3<<    //hbd
41      3<<6<<7<<    //dgh
42      6<<3<<2<<    //gdc
43      1<<2<<3<<    //bcd
44      2<<1<<0<<    //cba
45      0<<6<<4<<    //age
46      6<<0<<2<<    //gac
47      5<<6<<7<<    //fgh
48      6<<5<<4;     //gfe
49    correctHand();
```

Subdivison

```
8   QVector<QVector3D> vertz;
9   QVector<int> indexz;
10
11  void correctHand(){
12    int dm;
13    for (int i = 0; i < indexz.size(); i+=3){
14      QVector3D a=a.crossProduct(vertz.at(indexz.at(i))-vertz.at(indexz.at(i+1)),vertz.at(indexz.at(i+2))-vertz.at(indexz.at(i+1)));
15      float n=a.dotProduct(vertz.at(indexz.at(i)),a);
16      if (n>0) {
17        dm = indexz.at(i+1);
18        indexz[i+1] = indexz.at(i);
19        indexz[i] = dm;
20      }
21    }
22  }
```

```
73  void subdivide(){
74  //  float l = vertz.at(0).length();
75    int k = indexz.size();
76    QVector<int> indexy;
77    for (int i = 0; i<k; i+=3){
78      int z=vertz.size();
79      QVector3D mid =  (vertz.at(indexz.at(i)) + vertz.at(indexz.at(i+1)))* 0.5f;
80      mid.normalize();
81    //  mid*=l;
82      if (mid == vertz.last()) {
83        indexy<<indexz.at(i+1)<<indexz.at(i+2)<<z-1;
84        indexy<<indexz.at(i)<<indexz.at(i+2)<<z-1;
85      }else{
86        indexy<<indexz.at(i+1)<<indexz.at(i+2)<<z;
87        indexy<<indexz.at(i)<<indexz.at(i+2)<<z;
88        vertz << mid;
89      }
90    }
91    indexz.clear();
92    indexz=indexy;
93    correctHand();
94  }
```

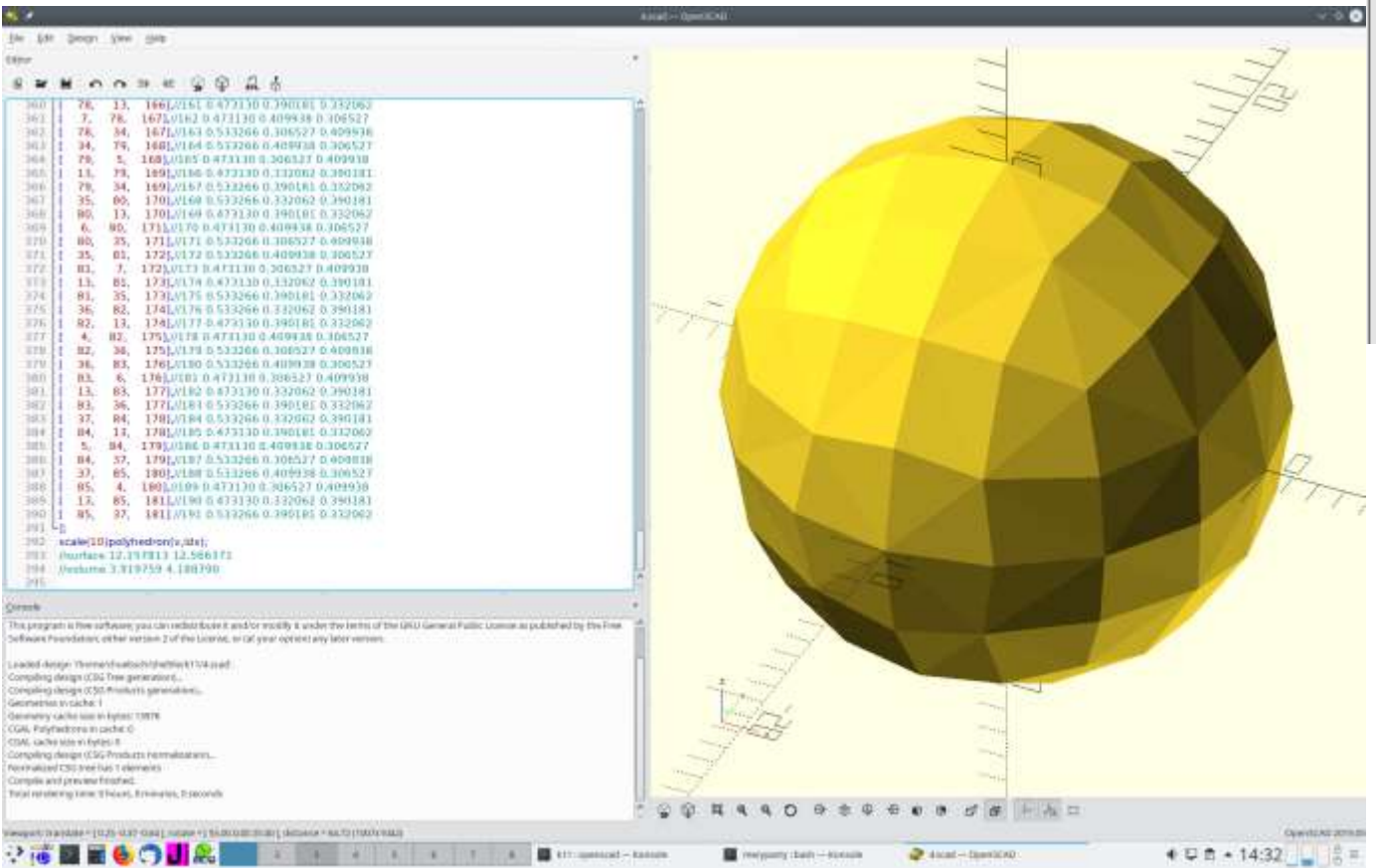


```
25    vertz << QVector3D(-0.5f,  0.5f,  0.5f)<<
26      QVector3D( 0.5f,  0.5f,  0.5f)<<
27      QVector3D(-0.5f, -0.5f,  0.5f)<<
28      QVector3D( 0.5f, -0.5f,  0.5f)<<
29      QVector3D(-0.5f,  0.5f, -0.5f)<<
30      QVector3D( 0.5f,  0.5f, -0.5f)<<
31      QVector3D(-0.5f, -0.5f, -0.5f)<<
32      QVector3D( 0.5f, -0.5f, -0.5f);
33    for (int i = 0; i < vertz.size(); i++){
34      vertz[i]= vertz.at(i).normalized();
35    }
36    indexz<<
37      1<<4<<5<<    //bef
38      4<<1<<0<<    //eba
39      1<<7<<5<<    //bhf
40      7<<1<<3<<    //hbd
41      3<<6<<7<<    //dgh
42      6<<3<<2<<    //gdc
43      1<<2<<3<<    //bcd
44      2<<1<<0<<    //cba
45      0<<6<<4<<    //age
46      6<<0<<2<<    //gac
47      5<<6<<7<<    //fgh
48      6<<5<<4;     //gfe
49    correctHand();
```

Subdivison

```
8  QVector<QVector3D> vertz;
9  QVector<int> indexz;
10
11 void correctHand(){
12   int dm;
13   for (int i = 0; i < indexz.size(); i+=3){
14     QVector3D a=a.crossProduct(vertz.at(indexz.at(i))-vertz.at(indexz.at(i+1)),vertz.at(indexz.at(i+2))-vertz.at(indexz.at(i+1)));
15     float n=a.dotProduct(vertz.at(indexz.at(i)),a);
16     if (n>0) {
17       dm = indexz.at(i+1);
18       indexz[i+1] = indexz.at(i);
19       indexz[i] = dm;
20     }
21   }
22 }
```

```
73 void subdivide(){
74 //  float l = vertz.at(0).length();
75   int k = indexz.size();
76   QVector<int> indexy;
77   for (int i = 0; i<k; i+=3){
78     int z=vertz.size();
79     QVector3D mid =  (vertz.at(indexz.at(i)) + vertz.at(indexz.at(i+1)))* 0.5f;
80     mid.normalize();
81   //  mid*=l;
82     if (mid == vertz.last()) {
83       indexy<<indexz.at(i+1)<<indexz.at(i+2)<<z-1;
84       indexy<<indexz.at(i)<<indexz.at(i+2)<<z-1;
85     }else{
86       indexy<<indexz.at(i+1)<<indexz.at(i+2)<<z;
87       indexy<<indexz.at(i)<<indexz.at(i+2)<<z;
88       vertz << mid;
89     }
90   }
91   indexz.clear();
92   indexz=indexy;
93   correctHand();
94 }
```

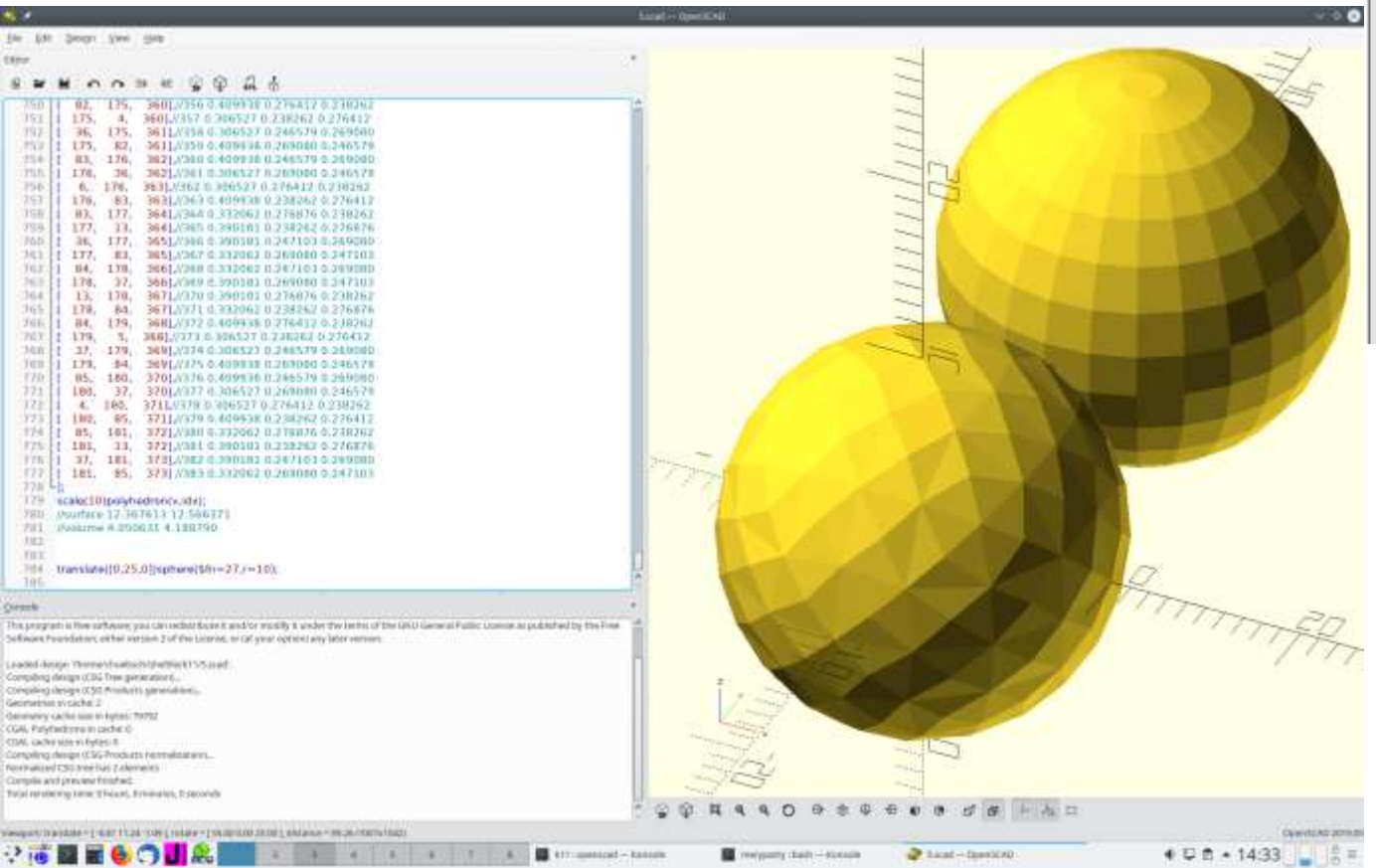


```
25   vertz << QVector3D(-0.5f,  0.5f,  0.5f)<<
26     QVector3D( 0.5f,  0.5f,  0.5f)<<
27     QVector3D(-0.5f, -0.5f,  0.5f)<<
28     QVector3D( 0.5f, -0.5f,  0.5f)<<
29     QVector3D(-0.5f,  0.5f, -0.5f)<<
30     QVector3D( 0.5f,  0.5f, -0.5f)<<
31     QVector3D(-0.5f, -0.5f, -0.5f)<<
32     QVector3D( 0.5f, -0.5f, -0.5f);
33   for (int i = 0; i < vertz.size(); i++){
34     vertz[i]= vertz.at(i).normalized();
35   }
36   indexz<<
37     1<<4<<5<<    //bef
38     4<<1<<0<<    //eba
39     1<<7<<5<<    //bhf
40     7<<1<<3<<    //hbd
41     3<<6<<7<<    //dgh
42     6<<3<<2<<    //gdc
43     1<<2<<3<<    //bcd
44     2<<1<<0<<    //cba
45     0<<6<<4<<    //age
46     6<<0<<2<<    //gac
47     5<<6<<7<<    //fgh
48     6<<5<<4;     //gfe
49   correctHand();
```

Subdivison

```
8   QVector<QVector3D> vertz;
9   QVector<int> indexz;
10
11  void correctHand(){
12    int dm;
13    for (int i = 0; i < indexz.size(); i+=3){
14      QVector3D a=a.crossProduct(vertz.at(indexz.at(i))-vertz.at(indexz.at(i+1)),vertz.at(indexz.at(i+2))-vertz.at(indexz.at(i+1)));
15      float n=a.dotProduct(vertz.at(indexz.at(i)),a);
16      if (n>0) {
17        dm = indexz.at(i+1);
18        indexz[i+1] = indexz.at(i);
19        indexz[i] = dm;
20      }
21    }
22  }
```

```
73  void subdivide(){
74  //  float l = vertz.at(0).length();
75    int k = indexz.size();
76    QVector<int> indexy;
77    for (int i = 0; i<k; i+=3){
78      int z=vertz.size();
79      QVector3D mid =  (vertz.at(indexz.at(i)) + vertz.at(indexz.at(i+1)))* 0.5f;
80      mid.normalize();
81    //  mid*=l;
82      if (mid == vertz.last()) {
83        indexy<<indexz.at(i+1)<<indexz.at(i+2)<<z-1;
84        indexy<<indexz.at(i)<<indexz.at(i+2)<<z-1;
85      }else{
86        indexy<<indexz.at(i+1)<<indexz.at(i+2)<<z;
87        indexy<<indexz.at(i)<<indexz.at(i+2)<<z;
88        vertz << mid;
89      }
90    }
91    indexz.clear();
92    indexz=indexy;
93    correctHand();
94  }
```



```
25    vertz << QVector3D(-0.5f,  0.5f,  0.5f)<<
26      QVector3D( 0.5f,  0.5f,  0.5f)<<
27      QVector3D(-0.5f, -0.5f,  0.5f)<<
28      QVector3D( 0.5f, -0.5f,  0.5f)<<
29      QVector3D(-0.5f,  0.5f, -0.5f)<<
30      QVector3D( 0.5f,  0.5f, -0.5f)<<
31      QVector3D(-0.5f, -0.5f, -0.5f)<<
32      QVector3D( 0.5f, -0.5f, -0.5f);
33    for (int i = 0; i < vertz.size(); i++){
34      vertz[i]= vertz.at(i).normalized();
35    }
36    indexz<<
37      1<<4<<5<<     //bef
38      4<<1<<0<<     //eba
39      1<<7<<5<<     //bhf
40      7<<1<<3<<     //hbd
41      3<<6<<7<<     //dgh
42      6<<3<<2<<     //gdc
43      1<<2<<3<<     //bcd
44      2<<1<<0<<     //cba
45      0<<6<<4<<     //age
46      6<<0<<2<<     //gac
47      5<<6<<7<<     //fgh
48      6<<5<<4;      //gfe
49    correctHand();
```

Qt: code less, create more

Qt is a well written widget toolkit for creating graphical user interfaces which run on various hardware plattforms like Linux, Windows, macOS.
It comes with Creator,a decent IDE, and Assistant a well made documentation.

ASCII Art (AA)
Colored AA (CaCa)


```cpp
#include <QtGui>
#include <stdlib.h>
#include <string.h>

int maxcol=90;

void printicon(QString s,int flip,char x){
  QImage II=QImage(s);
  if ((II.width()>maxcol)||(maxcol!=90)) II=II.scaledToWidth(maxcol);
//  QImage II=QPixmap(":/icons/cbh.png" ).toImage();
  int r, g, b, c216,gray,k=0;
  char X[13]="ShelXle.org ";
  for (int j=0; j<II.height(); j++){
  for (int i=0; i<II.width(); i++){
    //printf("%2x",qGray(II.pixel(i,j)));a
    double al=qAlpha(II.pixel(i,j))/256.0;
    r=(int) (qRed(II.pixel(i,j))/48.0*al);
    g=(int) (qGreen(II.pixel(i,j))/48.0*al);
    b=(int) (qBlue(II.pixel(i,j))/48.0*al);
    gray=qGray(II.pixel(i,j))/11+232;
    c216 = 16 + 36 * r + 6 * g + b;

    gray=(int) (al*gray + (1.0-al)*238);
   // c216=(int) (al*c216 + (1.0-al)*102);
    if (flip) printf("\e[48;5;%dm\e[38;5;%dm%c%c",gray,c216,x,x);
    else  printf("\e[48;5;%dm\e[38;5;%dm%c%c",c216,gray,X[k%12],X[(k+1)%12]);
    k++;
    k++;
    //i++;

  }
  printf("\e[0;0;0m\n");
  //j++;
  }
```

256-color mode — foreground: ESC[38;5;#m background: ESC[48;5;#m [hide]

Standard colors

0	1	2	3	4	5	6	7

High-intensity colors

8	9	10	11	12	13	14	15

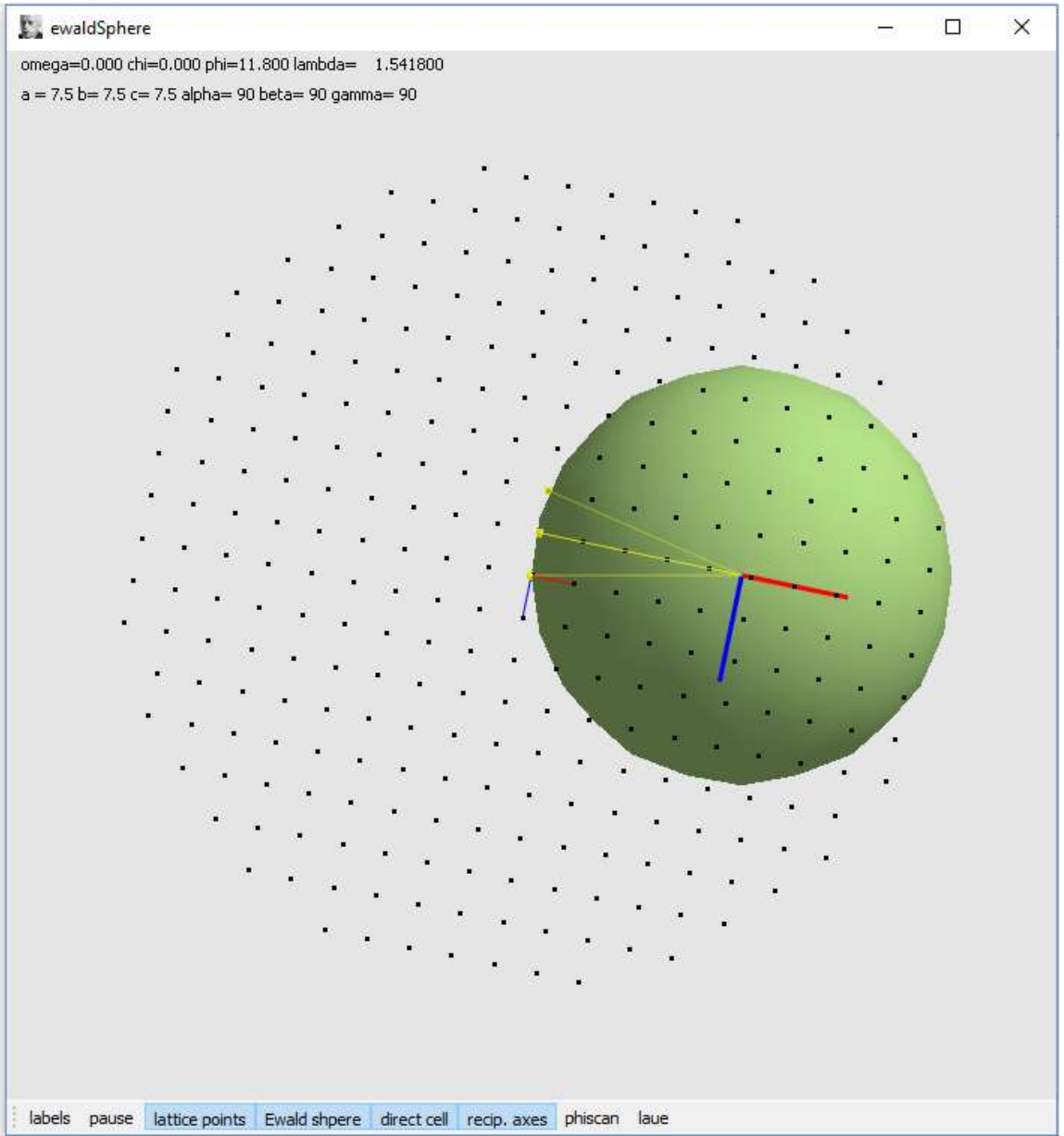
216 colors

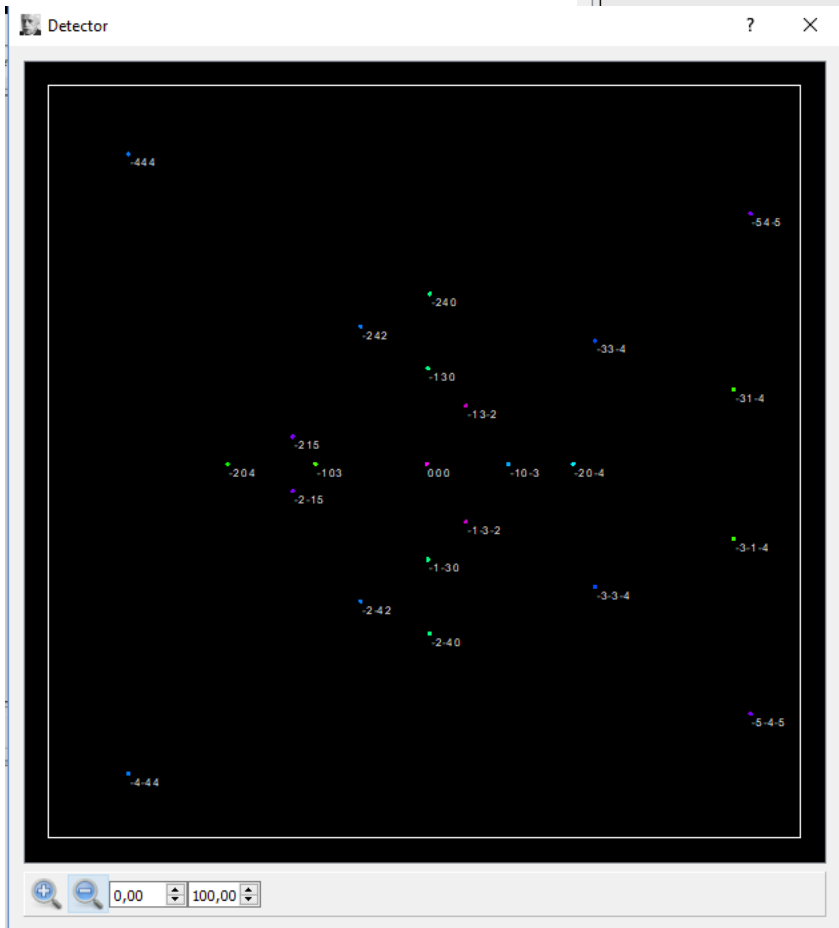
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87
88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123
124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195
196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231

Grayscale colors

232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Ewald sphere simulator





OpenScad + 3D-printing = programming based physical Objects

OpenScad CheatSheet

OpenSCAD v2019.05

Syntax

var = value;
var = cond ? value_if_true : value_if_false;
module name(…) { … }
name();
function name(…) = …
name();
include <….scad>
use <….scad>

Constants

undef undefined value
PI mathematical constant π (~3.14159)

Special variables

$fa minimum angle
$fs minimum size
$fn number of fragments
$t animation step
$vpr viewport rotation angles in degrees
$vpt viewport translation
$vpd viewport camera distance
$children number of module children
$preview true in F5 preview, false for F6

Modifier Characters

* disable
! show only
highlight / debug
% transparent / background

2D

circle(radius | d=diameter)
square(size,center)
square([width,height],center)
polygon([points])
polygon([points],[paths])
text(t, size, font,
halign, valign, spacing,
direction, language, script)
import("….ext")
projection(cut)

3D

sphere(radius | d=diameter)
cube(size, center)
cube([width,depth,height], center)
cylinder(h,r|d,center)
cylinder(h,r1|d1,r2|d2,center)
polyhedron(points, faces, convexity)
import("….ext")
linear_extrude(height,center,convexity,twist,slices)
rotate_extrude(angle,convexity)
surface(file = "….ext",center,convexity)

Transformations

translate([x,y,z])
rotate([x,y,z])
rotate(a, [x,y,z])
scale([x,y,z])
resize([x,y,z],auto)
mirror([x,y,z])
multmatrix(m)
color("colorname",alpha)
color("#hexvalue")
color([r,g,b,a])
offset(r|delta,chamfer)
hull()
minkowski()

Boolean operations

union()
difference()
intersection()

List Comprehensions

Generate [for (i = *range*|*list*) i]
Generate [for (*init*;*condition*;*next*) i]
Flatten [each i]
Conditions [for (i = …) if (condition(i)) i]
Conditions [for (i = …) if (condition(i)) x else y]
Assignments [for (i = …) let (assignments) a]

Flow Control

for (i = [start:end]) { … }
for (i = [start:step:end]) { … }
for (i = […,…,…]) { … }
for (i = …, j = …, …) { … }
intersection_for(i = [start:end]) { … }
intersection_for(i = [start:step:end]) { … }
intersection_for(i = […,…,…]) { … }
if (…) { … }
let (…) { … }

Type test functions

is_undef
is_bool
is_num
is_string
is_list

Other

echo(…)
render(convexity)
children([idx])
assert(condition, message)
assign (…) { … }

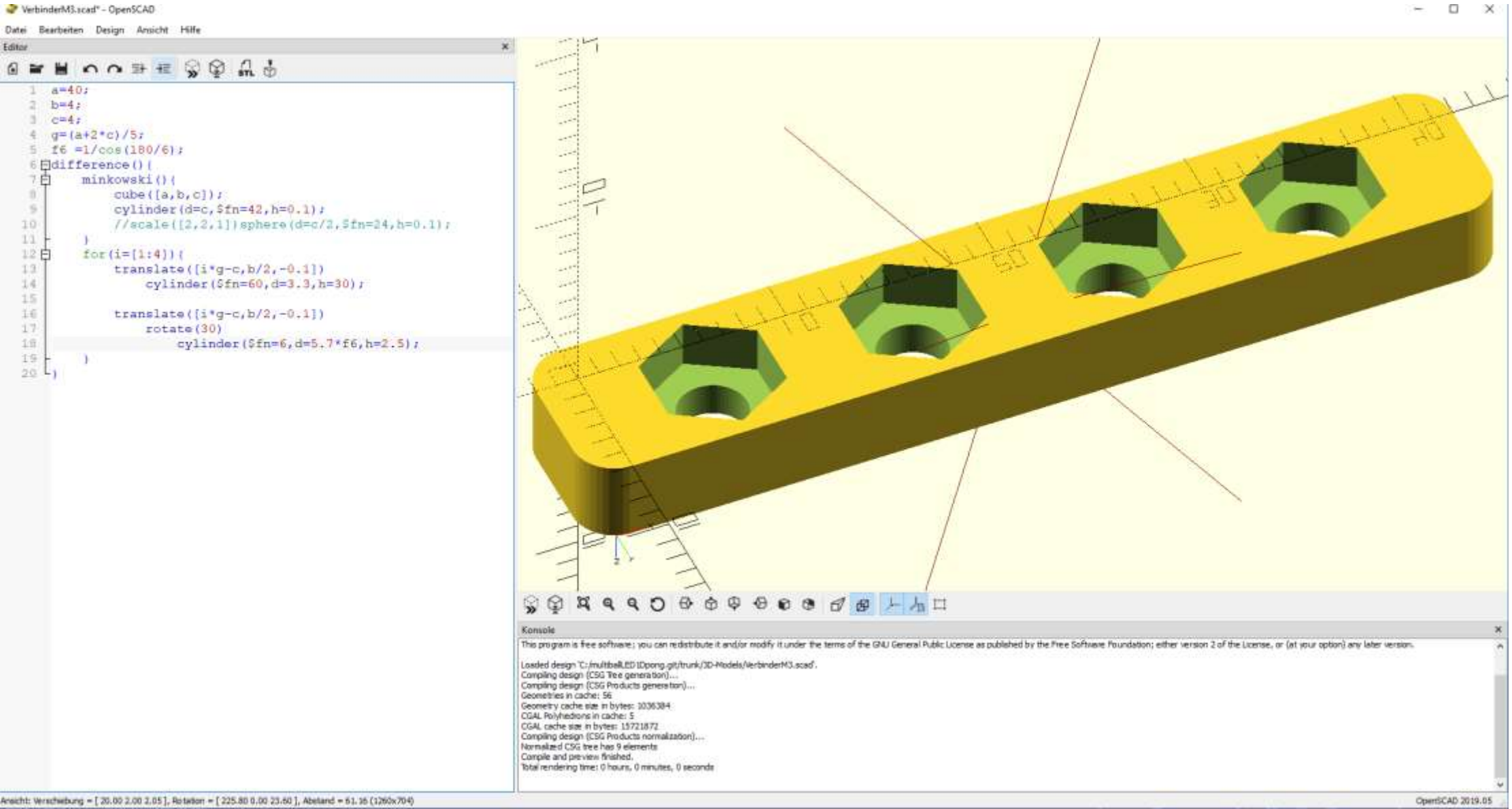
Functions

concat
lookup
str
chr
ord
search
version
version_num
parent_module(idx)

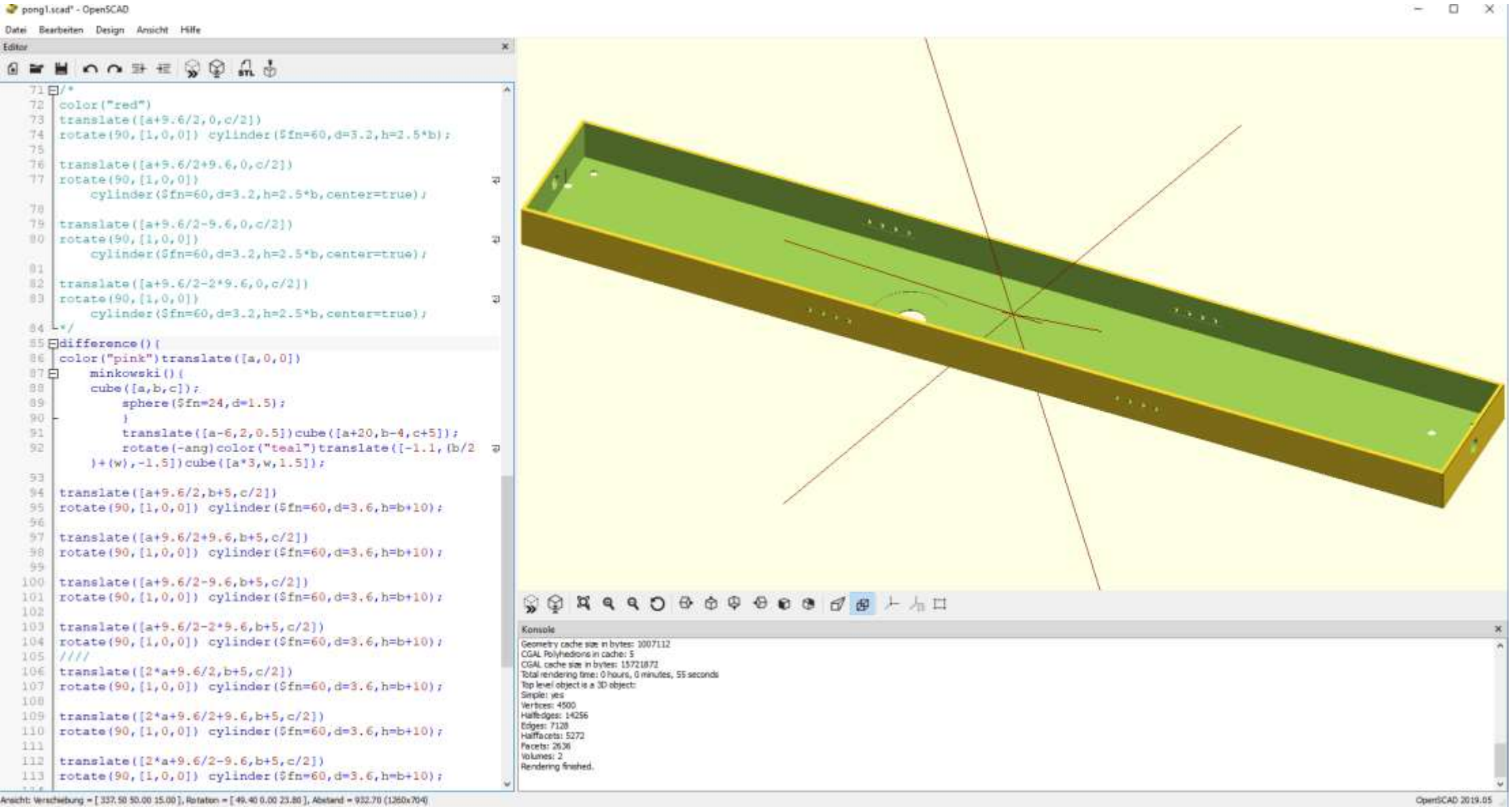
Mathematical

abs
sign
sin
cos
tan
acos
asin
atan
atan2
floor
round
ceil
ln
len
let
log
pow
sqrt
exp
rands
min
max
norm
cross

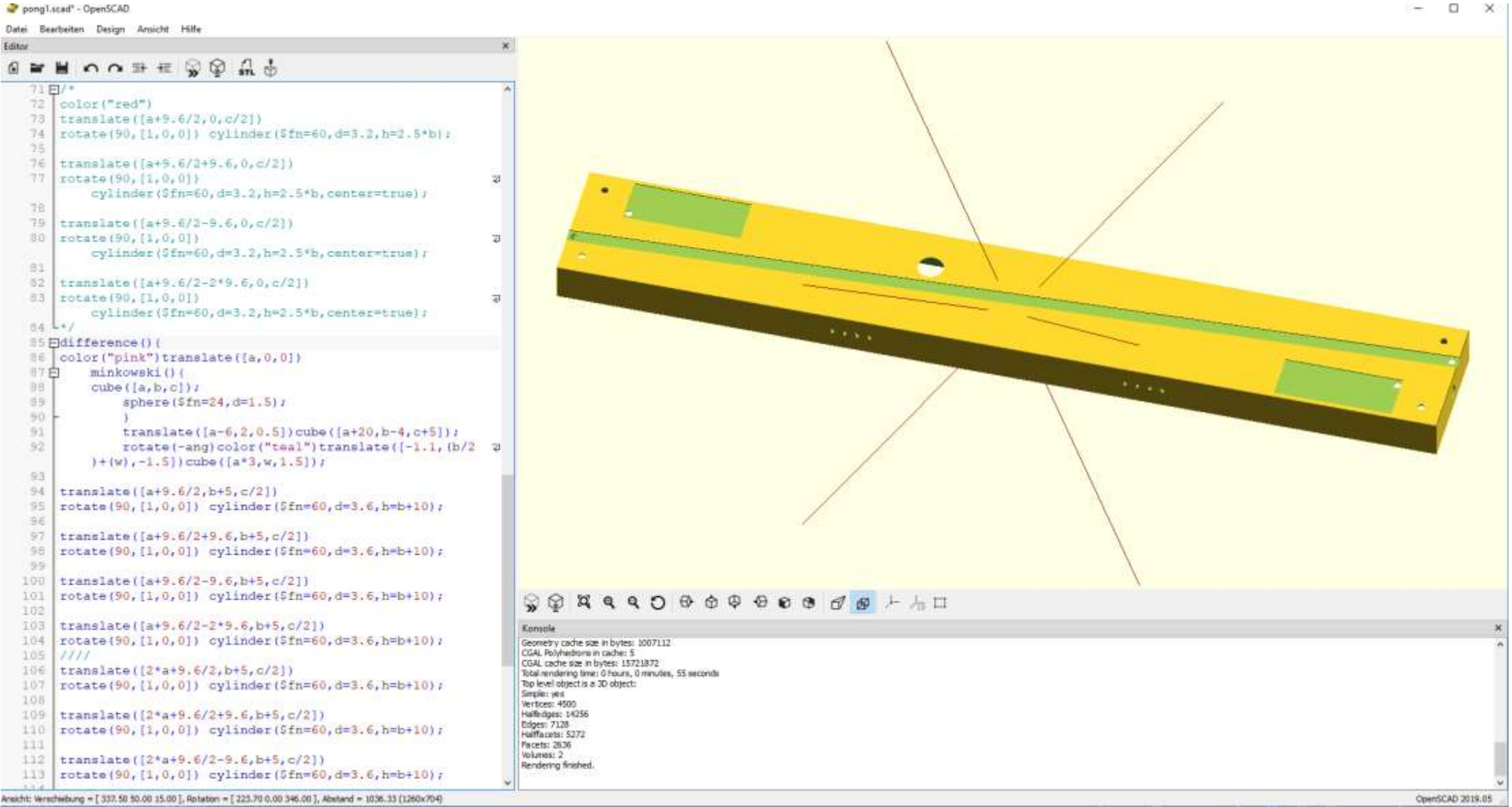
OpenScad + 3D-printing = programming based physical Objects



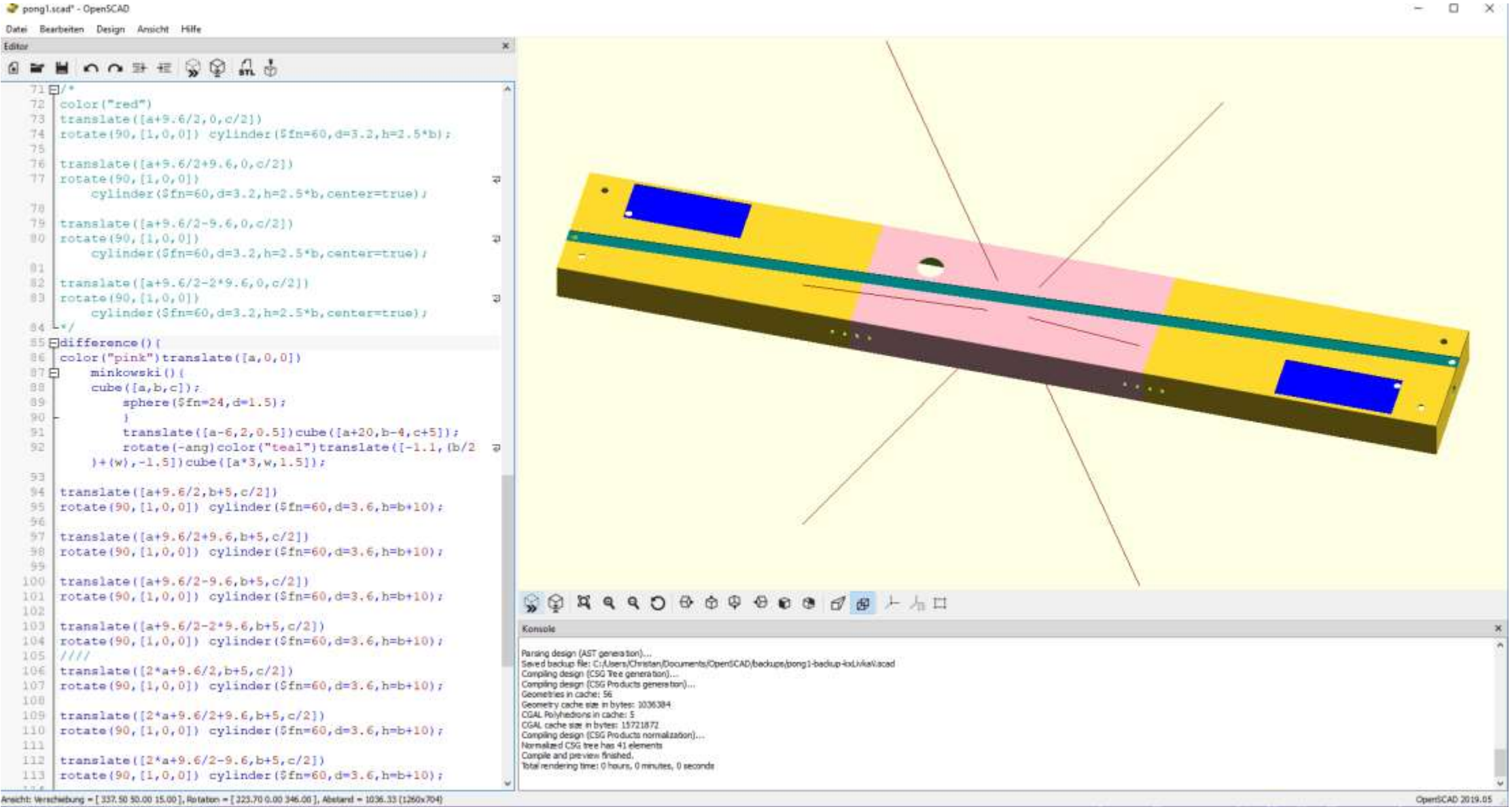
OpenScad + 3D-printing = programming based physical Objects



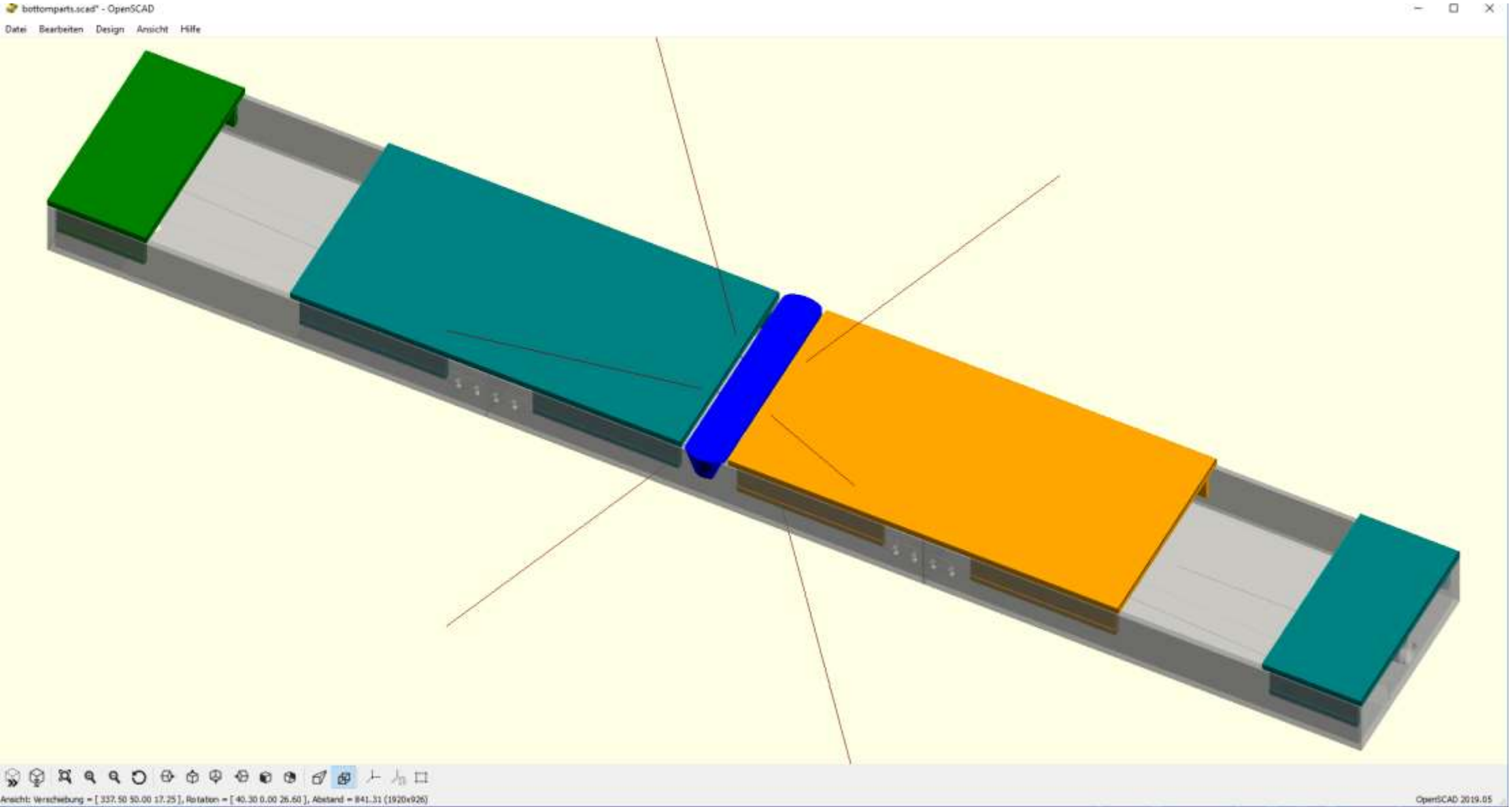
OpenScad + 3D-printing = programming based physical Objects



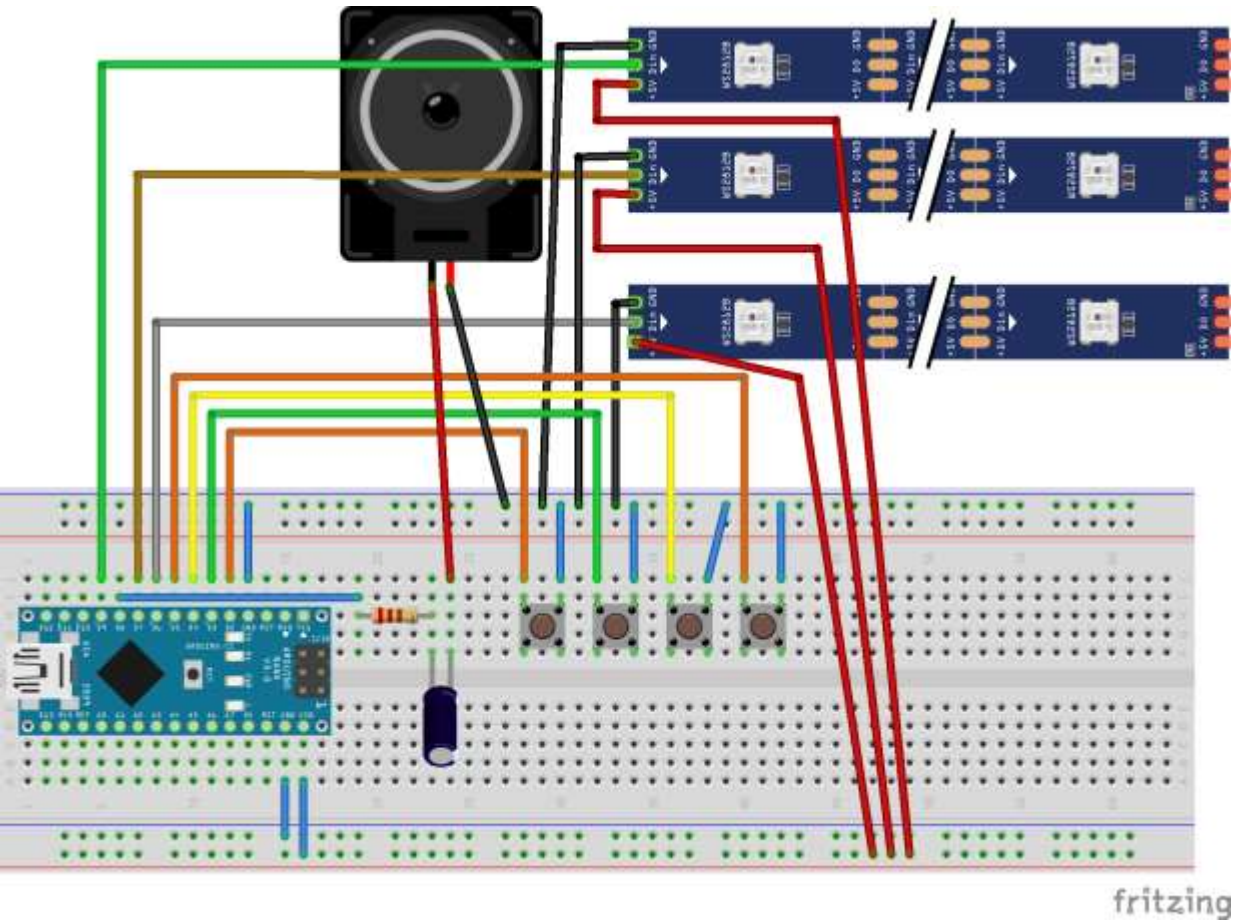
OpenScad + 3D-printing = programming based physical Objects

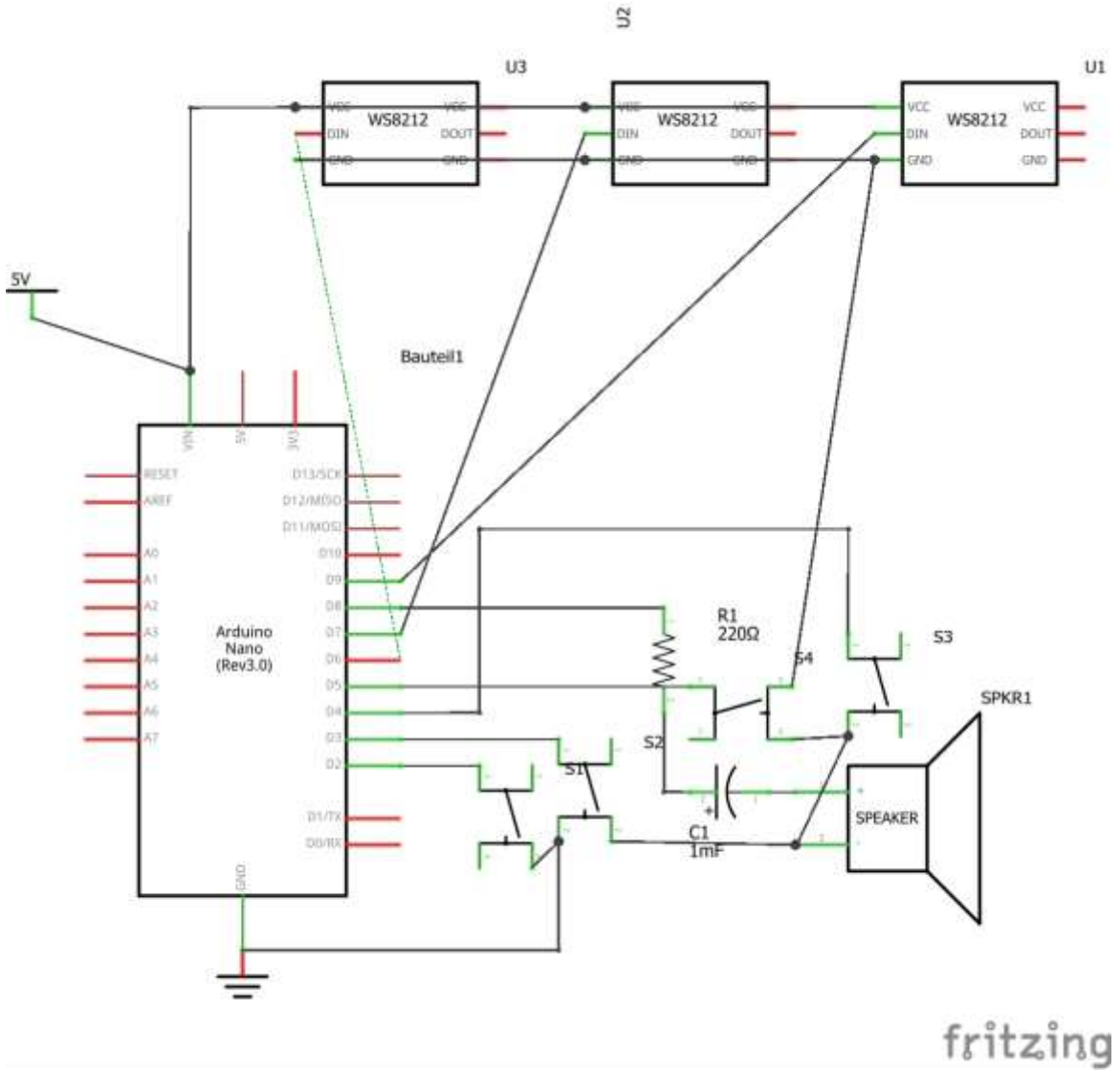


OpenScad + 3D-printing = programming based physical Objects



Fritzing and Arduino





Fritzing and Arduino

1D-multiball-pong.ino | Arduino 1.8.9 (Windows Store 1.8.21.0)

Datei Bearbeiten Sketch Werkzeuge Hilfe

1D-multiball-pong.ino

```
    activePlayers = 1;
    activeBalls = 1;
    twoPlays = true;
    showScore2();
    balls[ab].color = cols[ab];
    balls[ab].dir = -1;
    balls[ab].pos = 39;
    balls[ab].speed = 0;
  } else if ((activeBalls) && (balls[0].speed == 0)) {
    if ((activePlayers == 1) && (balls[0].dir == -1)) {
      activeBalls = (activeBalls < 3) ? activeBalls + 1 : activeBalls;
      ab = activeBalls - 1;
      balls[ab].color = cols[ab];
      balls[ab].dir = -1;
      balls[ab].pos = 39;
      balls[ab].speed = 0;
    } else if ((activePlayers == 1) && (balls[0].dir == 1)) {
      activePlayers = 2;
      twoPlays = true;
      showScore2();
    }
  }
}

uint8_t calls = 0;
void doballs() {
  calls++;
  calls %= 5;
  trackoff();
  for (int i = 0; i < activeBalls; i++) {
    /*
      Serial.print("Ball# "); Serial.print(i + 1);
      Serial.print(" pos=  "); Serial.print(balls[i].pos);
      Serial.print(" speed= "); Serial.print(balls[i].speed);
      Serial.print(" dir= "); Serial.println(balls[i].dir);
      // */
    if (balls[i].speed > calls) {
      balls[i].pos += balls[i].dir; //speed == 0 never moves, speed == 1 moves once in 5 calls, speed 5 or higher always moves
      //Serial.print(" Pos=  "); Serial.println(balls[i].pos);
      if (balls[i].pos < 0) player2scores(i);
      else if (balls[i].pos > 39) player1scores(i);
    }
    track.setPixelColor(balls[i].pos, track.getPixelColor(balls[i].pos) | balls[i].color); //mix colors if two are at same position
  }
  track.show();
}
```

1D-multiball-pong.ino | Arduino 1.8.9 (Windows Store 1.8.21.0)

Datei Bearbeiten Sketch Werkzeuge Hilfe

1D-multiball-pong.ino

```
#include <Adafruit_NeoPixel.h>

#define PLAYER1_RACKET_PIN 2
#define PLAYER2_RACKET_PIN 3
#define PLAYER1_FUNC_PIN 4
#define PLAYER2_FUNC_PIN 5
#define GAME_TRACK_PIN 6
#define PLAYER1_DISPLAY_PIN 7
#define SPEEKER_PIN 8
#define PLAYER2_DISPLAY_PIN 9

const byte N0[12] = {0, 9, 10, 1, 11, 2, 12, 3, 13, 4, 5, 14};
const byte N1[6] =  {5, 6, 7, 8, 9, 13};
const byte N2[11] = {0, 9, 10, 11, 2, 7, 12, 3, 4, 5, 14};
const byte N3[11] = {0, 1, 2, 3, 4, 5, 7, 9, 10, 12, 14};
const byte N4[9] = {0, 1, 2, 3, 4, 7, 12, 13, 14};
const byte N5[11] = {0, 9, 10, 1, 2, 7, 12, 13, 4, 5, 14};
const byte N6[12] = {0, 1, 2, 4, 5, 7, 9, 10, 11, 12, 13, 14};
const byte N7[7] = {0, 1, 2, 3, 4, 5, 14};
const byte N8[13] = {0, 9, 10, 1, 11, 2, 7, 12, 3, 13, 4, 5, 14};
const byte N9[12] = {0, 9, 10, 3, 1, 2, 7, 12, 13, 4, 5, 14};


Adafruit_NeoPixel track(40, GAME_TRACK_PIN, NEO_GRB + NEO_KHZ800);
Adafruit_NeoPixel display_1(15, PLAYER1_DISPLAY_PIN, NEO_GRB + NEO_KHZ800);
Adafruit_NeoPixel display_2(15, PLAYER2_DISPLAY_PIN, NEO_GRB + NEO_KHZ800);
unsigned long last = 0, nowis = 0;
struct Ball {
  uint32_t   color;
  int8_t pos;
  int8_t dir;
  uint16_t speed;
};
Ball balls[3];
uint8_t activeBalls = 0;
uint8_t activePlayers = 0;
uint8_t scorePlayer1 = 0;
uint8_t scorePlayer2 = 0;
bool onePlays = false;
bool twoPlays = false;

uint32_t cols[3] = {0x00ff0000, 0x0000ff00, 0x000000ff};

void func1() { // Player one pressed their button
  tone(SPEEKER_PIN, 880, 100);
  uint8_t ab = 0;
  if (activePlayers == 0) {
```

Hochladen abgeschlossen.

```
Der Sketch verwendet 9290 Bytes (30%) des Programmspeicherplatzes. Das Maximum sind 30720 Bytes.
Globale Variablen verwenden 438 Bytes (21%) des dynamischen Speichers, 1610 Bytes für lokale Variablen verbleiben. Das Maximum sind 2048 Byte
```

Credits

- Thanks to Lukas Eiter for playing in the video
- Developers of:
  - Qt
  - openScad
  - Cura
  - Marlin
  - Fritzing
  - Arduino, sure I forgot a lot, sorry!
- Google, Wikipedia, Thingiverse, GitHub

- You for you patience!