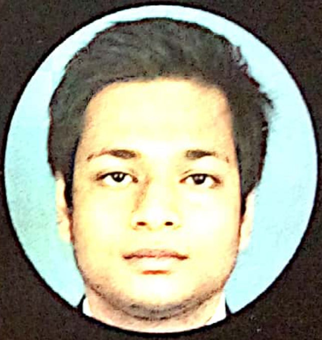
New Approach to Generate, Re-Generate, Encode, and Compress 3D Structures using Complex Functions

Jimut Bahan Pal
Department of Computer Science
Bachelor of Science
Batch of 2016-2019



Almost everything we see in this mundane world, can be represented using complex polynomial equations. Can't they? Most of the simple objects can be represented directly. Let's take an example of a simple heart in 3D as shown in Fig. 1, and look at the equation carefully. It represents a polynomial equation of 6th degree.

Let's talk about modelling an object in any standard animation software. There are several modeling softwares, like CAD which has .stl extension. It stores mainly the vertices. Another one being 3DS, now owned by Autodesk Maya. It stores data in the form of a binary chunk, similar to XML DOM tree with a .3ds extension. Flimbox or (.fbx) files are also modeled using Autodesk, another format to represent high quality models with textures.

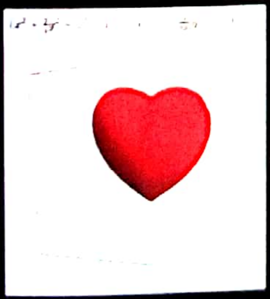


Fig. 1: A 3D model of a heart

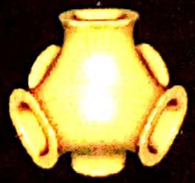


Fig. 2: A 3D model of a complex structure

Fig. 3: 3D representation of an Arena

Fig. 4: 3D representation of a chain like structure

Fig. 5: Composite structures in 3D

Fig. 6: A geode in 3D

Fig. 7: Implicit costa in 3D

Fig. 8: A representation of a lamp

Fig. 9: A broken vase represented in 3D

Fig. 10: A 3D representation of a virus

When we are modelling, we are working with vertices, edges and faces of the object being designed. Now, what is a face? A face is a 2D side of any 3D object. For example, a cube may have 6 faces. It is always necessary to have edges >2, i.e., 3 edges to have one face. Now if we draw a pentagon in a white sheet of paper, it will have one face but should have 5 edges, likewise if we draw a triangle it will have 3 edges and one face, and so on. Similarly, when we are working with any modelling software, we will work with face and edges and make necessary changes like a sculptor to change the shape and nature of the object in context. The more the no. of vertices/edges, the more heavy the exported object is. The more the number of faces, the more space the object will take to store it in the hard disk. Now, what are the .obj (object) files storing in it? It generally stores the position of edges, vertices in an optimised binary format, with some meta-data describing the software, time, etc. We can also easily analyse the object file by opening it in any text editor, like vim for opening large files in a short time.

Now let's come back to the example of the heart as shown in Fig. 1, if we sculpt it in Maya, a standard animation software using 400 faces and export it in an object file, we will find that it has a size of about 55 kb. What we will do here is, we will store the equation in a text format and it will be something like this: [(x^2 + (9/4) *(y^2) + z^2 - 1) ^3 - (x^2) *(z^3) - (9/200) *(y^2) *(z^3) = 0; -1<=x, y, z<=1], which will hardly represent 87 bytes in text. So, we have successfully reduced the size of the file, since it can be easily regenerated at any time, if we have the software.

There are a lot of advantages, firstly it reduces the size of the model, which makes it possible to carry and transport it very fast. Secondly, since it is a text file and an equation, it can be easily encoded in any format using standard cryptographic techniques, which means it can be broadly used to hide classified military secrets. One with a sharp memory can easily remember the equation and regenerate it in the software. All the objects in the above figures [Fig. 1, Fig. 2, Fig. 3, Fig. 4, Fig. 5, Fig. 6, Fig. 7, Fig. 8, Fig. 9, Fíg. 10] are represented using multiple set of equations, but it will be junk to show all of them here.

Now we are representing very simple objects with equations. What about some complex objects? Yes, it too can be represented. For example, the Fig. 9, the broken vase can be represented in a series of equations:

```
isoTransform_2(x,y,-z,t)*isoTransform_6(x,y,-z,t)
Angle1=atan2(sqrt(x*x+y*y),(-z+cx))
Angle2=atan2(x,(y+cx))
CarvinCondition=abs((z-1)-0.6*cos(18*Ang
CarvinCondition
,x,z,t)/pi))<10.3|abs((z+3)-0.3*cos(18*Angle2(x,y,z,t)/pi+pi/4))<0.5
Iso=cos(x)*sin(y)+cos(y)*sin(z)+cos(z)*sin(x)
Torus=(sqrt(x*x+y*y)-3)^2+z*z-1
Bottom=(x*x+y*y+z*z-1)
IsoExterior=x*x/3,+y*y/3,-abs(1,5*sin(2*z/pi+0,3)+1,8)
Dfx2=((IsoExterior(x,y,z,t)-IsoExterior(x+cx,y,z,t))/cx)
Dfy2=((IsoExterior(x,y,z,t)-IsoExterior(x,y+cy,z,t))/cy)
Dfz2=((IsoExterior(x,y,z,t)-IsoExterior(x,y,z+cz,t))/cz)
Rapport2=(sqrt(DFx2(x,y,z,t)*Dfx2(x,y,z,t)+DFy2(x,y,z,t)*D-
fy2(x,y,z,t)+DFz2(x,y,z,t)*Dfz2(x,y,z,t)))
Iso2=(IsoExterior(x+t*Dfx2(x,y,z,t)*Thickness2/Rapport2(x-
,y,z,t),y+t*Dfy2(x,y,z,t)*Thickness2/Rapport2(x,y,z,t),z+t*Dfz2(x,y,z,t)*Thickness
2/Rapport2(x,y,z,t),t))
ThickIsoExterior=(Iso2(x,y,z,1)*Iso2(x,y,z,-1))
Iso=cos(x)*sin(y)+cos(y)*sin(z)+cos(z)*sin(x)
Dfx=((Iso(x,y,z,t)-Iso(x+cx,y,z,t))/cx)
Dfy=((Iso(x,y,z,t)-Iso(x,y+cy,z,t))/cy)
Dfz=((Iso(x,y,z,t)-Iso(x,y,z+cz,t))/cz)
Rapport=(sqrt(DFx(x,y,z,t)*Dfx(x,y,z,t)+DFy(x,y,z,t)*Dfy(x-
,y,z,t)+DFz(x,y,z,t)*Dfz(x,y,z,t)))
Iso4=(Iso(x+t*Dfx(x,y,z,t)*Thickness4/Rapport(x,y,z,t),y+t*D-
fy(x,y,z,t)*Thickness4/Rapport(x,y,z,t),z+t*Dfz(x,y,z,t)*Thickness4/Rapport(x,y,z
,t),t))
ThickIso2=(Iso4(x,y,z,-1)*Iso4(x,y,z,1))
isoTransform_2=if((CarvinCondition(x,y,z,t)=0),ThickIsoExterior(x,y,z,t),1)
Iso6=(Iso(x+t*Dfx(x,y,z,t)*Thickness6/Rapport(x,y,z,t),y+t*D-
fy(x,y,z,t)*Thickness6/Rapport(x,y,z,t),z+t*Dfz(x,y,z,t)*Thickness6/Rapport(x,y,z
,t),t))
isoTransform_6=if((CarvinCondition(x,y,z,t)&ThickIsoExteri-
or(x,y,z,t)<0),-ThickIso2(S*x,S*y,S*z,t)*Iso6(x*S,y*S,z*S,-1)*(Iso6(x*S,y*S,z*S,1))
,1)
```

with some additional parameters and boundary as :

x = -3.5 to 3.5

y = -3.5 to 3.5

z = -5 to 4.5

So, if we represent it in terms of a model in any animation software, it will be more than 2 MB for sure, but in this approach, we can represent it in a set of equations as shown above, which compresses it in a very compact format. The main advantage of this procedure is that it can represent complex shapes in a very compact manner and helping in transporting highly classified military secrets. We can also improve this method by applying multiple extra functions to describe the colour, which will generally take a lot of space for any texture based rendering software.

Now, if there is a complex structure, too complex to be represented in equations, what we can do with it is we can break it into multiple sub structures and represent each structure using a set of equations, like framing in networking. We can store the sub equations in some sequences in text files, so that during the time of generating it will regenerate the equation in such manner and sequences. Here the only problem is, it will take time to encode an existing 3D object to multiple set of equations, since it will have to fit every equation while partitioning it in several small objects. This method seems easy and interesting but is hard to implement, as polynomial equations generally represent continuous curves, obviously there will be some functions or equations that will be generating the object perfectly when represented in 3D space. So, in this way we can devise a unique way to hide, encode, compress, generate and regenerate 3D structures.