物体をリアルに表現する方法

How to represent objects realistically

Abstract

We try to visualize 3D objects on a computer screen by using a programming language “decimal basic”. Then it‘s essential that we use vectors in our research. To express 3D objects 2D, we need the X,Y and Z coordinates of the objects and the points of view. Based on these conditions, we will make a formula that shows how the objects are expressed on the XY plane when we set the Z coordinate 0. So far, we have made several pots on which we have printed patterns on their surfaces using the two processes described above, in this way we can simulate how they are seen in the real world.

１．はじめに

近年、社会全体の IT 化に伴い、プログラミングの技術に注目が集まっている。そこで、私たちはプログラミング技術の中でもあまり研究が進んでいない部類である、高校数学を利用したプログラミングによる図形描画について興味を持ち、研究を行うことにした。目的は、様々な形状、色の物体を二次元画面上に表現することであり、本研究は前述の通り、あまり研究が行われていないため、先行研究などはほとんどない。担当の先生の指導の下、思考実験と実践を行うのが主な研究方法である。

２．研究方法

本研究では、プログラミング言語の一つである十進 BASIC を使用している。様々な形状の図形例えば立方体、円錐、球体、より複雑な回転曲面を含んだものなどを数式と命令を用いて表現するのである。物体をリアルに表現するに当たって、３つの重要な要素がある。それは

⑴ 物体の輪郭

⑵ 物体表面の陰

⑶ 物体が地面に作る影

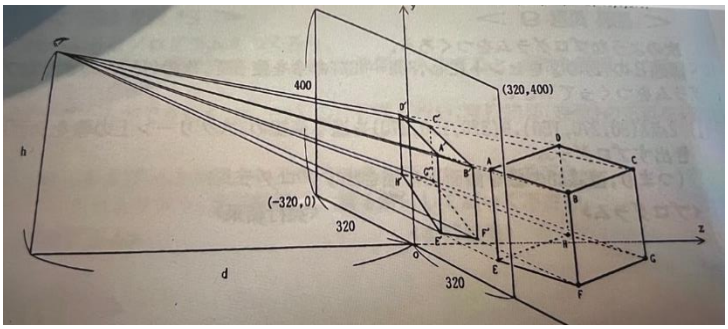
である。ここからは、これら３つの要素について１つ１つ解説していく。

⑴ 物体の輪郭

物体をパソコンの画面上に表現する上で、まず重要になってくるのは、三次元上の点を二次元で表現することである。あらゆる物体は、プログラミング上では無数の点の集合体である、と考えることができる。そして、三次元の点を二次元で表現すると言うことは、座標で考えると、X,Y,Z の３つの要素で表現されている点を Z 座標が０にした際にどこに対応するのかを考える、ということである。

下、図１にあるように物体を見る視点を決め、その視点と物体の各点をつなぐ直線を媒介変数を用いて表現し、その直線と Z 座標０の平面すなわちスクリーンとの交点が、三次元の点を二次元に移動させた際の点になるのである。

図１

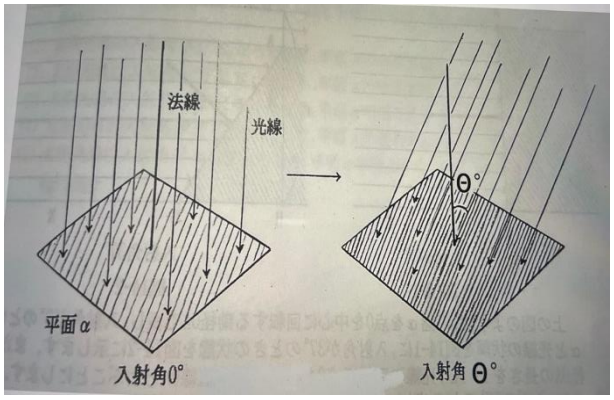


この一連の作業は、公式化することができる。視点の座標を（0,h,-d)、点の座標を(x,y,z)とおいたとき、その点を二次元化した際の座標は　(dx/(z+d),(zh+dy)/(z+d))となる。
公式の形上、情報として、物体のX,Y,Z座標を定義しなければならないので、一本の式のみでは表すことはできない。しかし、この公式を用いることによって、あらゆる図形を二次元の画面上に表現することができるようになる。

⑵ 物体表面の陰

物体をリアルに表現する重要な要素の一つとして表面色の濃淡がある。当然物体には色があり、私たちはそれを目で知覚しているが、同じ色で塗られたものであったとしても、点の位置によって、光の当たる角度、および受光量が変わり、濃淡は変わって見える。この違いもプログラミングで表現する必要がある。同一面上における色の濃淡の変化は、前述の通り受光量の違いによるものであり、入射角が面に対して垂直に近い角度であればあるほど単位面積あたりの受光量は大きくなって明るくなる。そこで、本研究では、その受光量の違いを、光源からの光を方向ベクトルで表現し、物体の１つ１つの点における法線ベクトルを導出し、それらの内積を用いることで、入射角θのCOSの値を出し、cosθの値を濃淡値とし、その値によって、色の濃淡を定義した。（図２）また、結果にて述べるが、絶対値命令とガウス記号を使用することでより、滑らかに色の濃淡変化を表現することに成功した。

図２



⑶ 物体が地面に作る影

物体に光が当たれば、当然影が地面に形成される。この影も表現しなければ、現実に近い形の物体の表現とは言えないだろう。影は、光が物体に遮られて届かなくなった地面にできるものであるので、プログラミング上では、物体のそれぞれの点を通る光の方向ベクトルと同じ方向、大きさの成分を持った直線と地面、すなわちｙ座標の値が０となる平面との交点がその点が生み出した影となる。（図３）

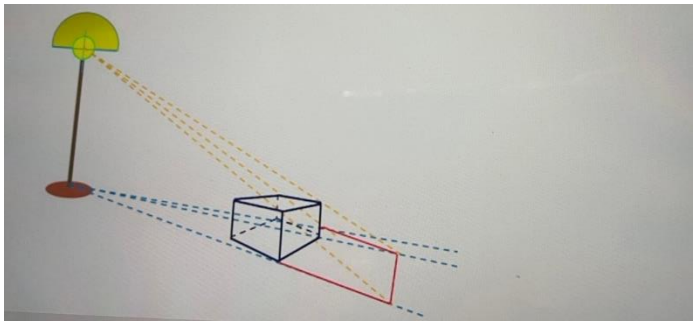


図３

この影をプロットする作業も公式化することができる。

太陽光線の方向ベクトル成分を(p, q, r)、視点の座標を(0, h, -d)、物体の点の座標を(a, b, c)とおいた際のスクリーン上に写る影の座標、(x, y, 0)の x, y は x=d(aq-bp)/(dq+cq-br)　y=h(cq-br)/(dq+cq-be)と表すことができる。この公式を用いることで、あらゆる物体の影を表現することができる。

続いて、物体を表現する上で重要な他の作業を説明して行く。

⑷ 陰面処理

実際の世界でもそうであるように、ある一点からだけでは、物体の全ての面を見ることはできない。必ず物体の別の面隠れて見えなくなっている面、陰面が存在する。二次元の画面上で表現する本研究のプログラミングにおいては、この面をプロットしないように命令しないと、同じ場所に二つの点が重なってしまい、図形を描く上で、不具合が生じるため、この作業は非常に重要である。

図４

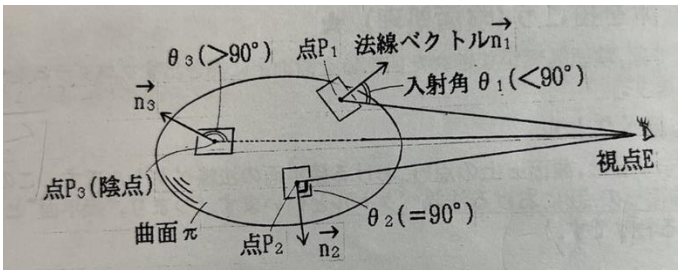


図４を例に説明していくと、人間の視界に入る物体の面の範囲は、視点と物体の点を結んだ直線の方向ベクトルと、その点の法線ベクトルがなす角度が９０度以下の面である。９０度を超えると、その場合視点とその点を結ぶ直線が物体の内部を通過していることになるので、実際には見えないのである。

この作業を公式化し、これによって、実際に二次元の画面上に点をプロットするか、判別することができるようになる。式の形としては、上記の２種類のベクトルと、内積の公式を用いて、cos の値を求め、その値が０以下であれば、プロットしない、というものである。

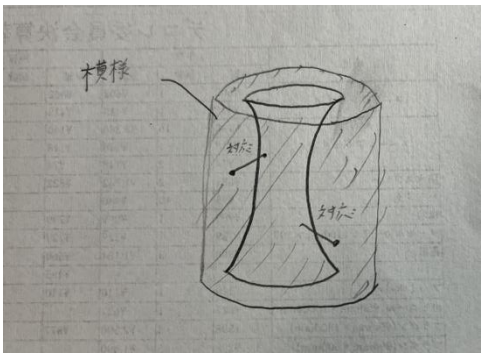
⑸ 模様

様々な物体を表現するに当たって、表面の模様も任意のものを貼り付けることができれば、表現の幅が広がると考えた。この作業は前述全ての作業ができた上での応用であり、難易度は非常に高い。

表現方法としては、プログラミングコードとアプリを使って任意の模様をプログラミング画面上に出し、その模様を表現したい物体がちょうど収まるような大きさの円柱型に各点を対応させ、その上で、物体に色をつける際に、その点の色を１つ１つ対応する模様の点のものにする、といったものである。

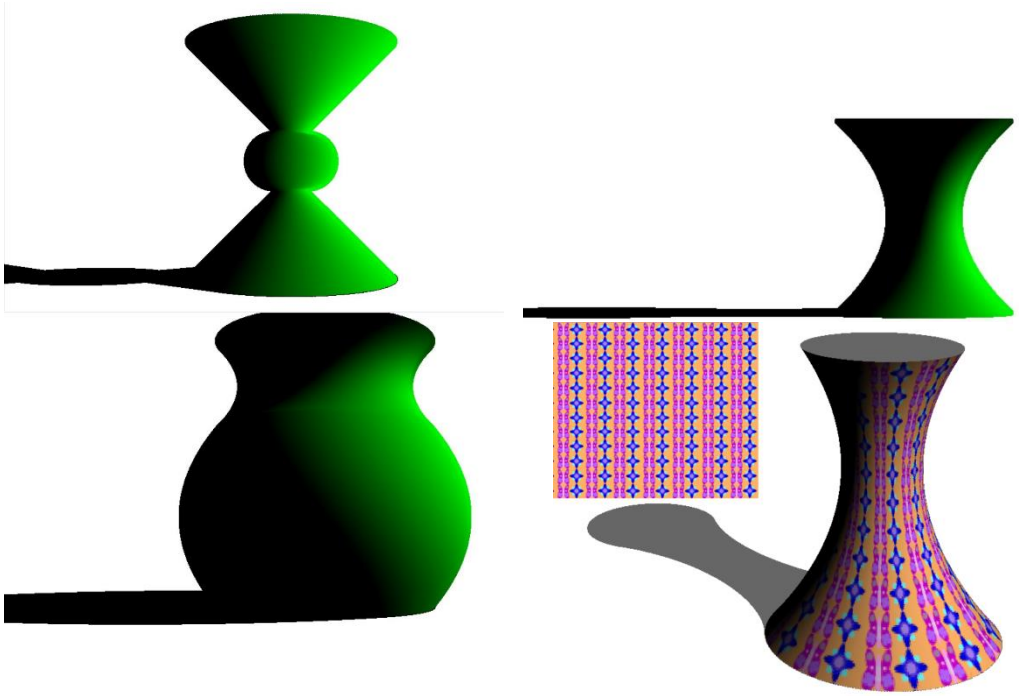
（図５）

図５



３．実験結果

上記の技術を利用し、様々な形の壺などを作った。



４．考察

結果から、プログラミングによって様々な物質を任意の模様で表現できる、ということが分かった。
現実世界の一部ではあるが、数式によって、二次元上に作り出すことができた、と言えるだろう。

５．今後の課題

今後の課題としては、より複雑な形をした物体の表現が挙げられる。複雑な凹凸がある面の表現方法などについては、さらに研究を重ねることで、公式などを作り出すことができるようになるだろう。

６．参考文献

物体表面色の濃淡解析　大石明徳