

Question 1

Discuss the ways in which you might convert an audio signal into an animated 3D graphical display.

Assume that the audio source is a stereo (2 channel) signal and that you have analysis software that allows you to measure the sound level at any frequency and time. Describe the visual effects that you would like and outline algorithms for achieving them. You do not need to provide either detailed algorithms or source code.

Since this question requires a lot of creativity and imagination, the way I thought about it was rather simple and then ideas came and were piled on top of the solution. My first perception was to make a salad bowl that is filled up with various kinds of vegetables each according to some pattern or key. Then, I thought that this idea is too classical for the high-tech music we have today and that most of the people would find it rather ridiculous to use. So, I shifted my idea to include some more colors and visual effects.

If we take a semi-transparent navy blue bowl on a deep black background and make it move around so that it looks like it's dancing while the song is playing.

Suitable code for this effect would be:

```
pushMatrix()  
rotate(rand()%360,1,1,0) while giving values for Xb and Yb  
or any moving effect which sets Xb and Yb to be used later.  
popMatrix()
```

Then, 2 opposite jet cones defined by their position (x1,y1) and (x2,y2) respectively which are moving rather fast on a circular path, defined by $x=r\cos(\theta)$ and $y=r*\sin(\theta)$ where r is the radius of the circular path (may be elliptic in changing the value of r) and θ the angle of change. The angle θ can be changed as frequently as you wish but has an interval $[0-2\pi]$. These cones above the bowl would throw a stream of colored particles towards the moving bowl. This requires a particle projection dependent on the movement of the bowl which is at position*

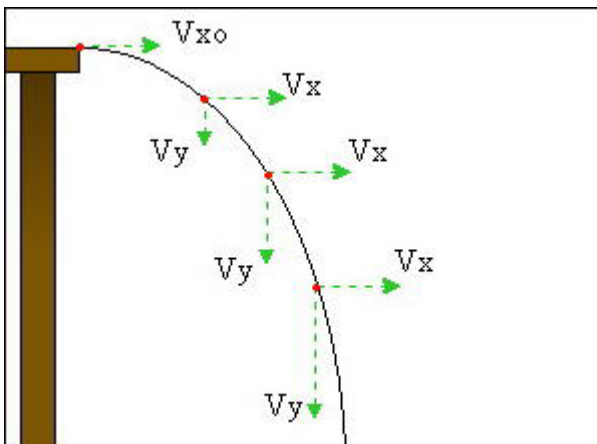
(X_b, Y_b) . Thus the movement of the bowl should be pre-calculated by some interval of time and (X_b, Y_b) changed accordingly.

```

Stops = 50;
Theta = 2PI/Stops;
R = 5;
for(int i=0;song is playing;i++)
{
    x=R*cos(i*theta);
    z=R*sin(i*theta);
    if(i<Stops/2) R+=0.1;//ecliptic effect
    else R-=0.1;
}

```

Projectile physics defined by this picture use the following functions:



$$g = 9.8m / s^2$$

$$V_y = -gt$$

$$V_x = V_{x0}$$

$$y = -\frac{1}{2}gt^2 \Rightarrow t = \sqrt{\frac{Y_b}{-4.9}}$$

$$x = V_x t$$

$$V_{x0} = \frac{x}{t} = \frac{X_b}{\sqrt{\frac{Y_b}{-4.9}}}$$

Using these formulas, use V_{x0} to get a value for x at a time slice $t_0 = t/\text{delay}$. Also get y at this same time t_0 . You will then have (x, y) for the next particle position as t_0 increases towards t . when $t_0 = t$, then the bowl is reached.

Since we have a dual channel stereo signal then, to interpret this, we could derive a color for the intensity (sound level) of each channel at the same frequency and time, therefore projecting 2 colored particles from the 2 cones, corresponding to the 2 channels, at one time each with its respective color. "The range of sound audible to the human ear falls roughly between 20 Hz and 20 kHz at typical

amplitudes with wide variations in response curves" (<http://wikipedia.com/>). Therefore, a set of 2 particles would be projected consequently for the 10 major frequencies at one time. This would make a projection of $2 \times 10 = 20$ particles at the same time filling the screen with a set of colors dependent on the sound intensity at the respective frequency. The major frequencies are selected to be: 31Hz, 62, 125, 250, 500, 1KHz, 2, 4, 8, and 16Khz. There is 1 octave difference between each of these globally used frequencies. Each frequency would be assigned a specific color, thus we need 10 colors. Suppose we split the color spectrum into 10 pieces having violet, dark blue, light blue, brown, dark green, bright green, yellow, orange, bright red, and dark red. Each of these would be assigned respectively to the up-mentioned frequencies.

As said in the Wikipedia.com online encyclopedia, "Sounds above 85 dB are considered harmful, while 120 dB is unsafe and 150 dB causes physical damage to the human body. Windows break at 163 dB. Jet airplanes are 165 dB. Eardrums pop at 190 to 198 dB."

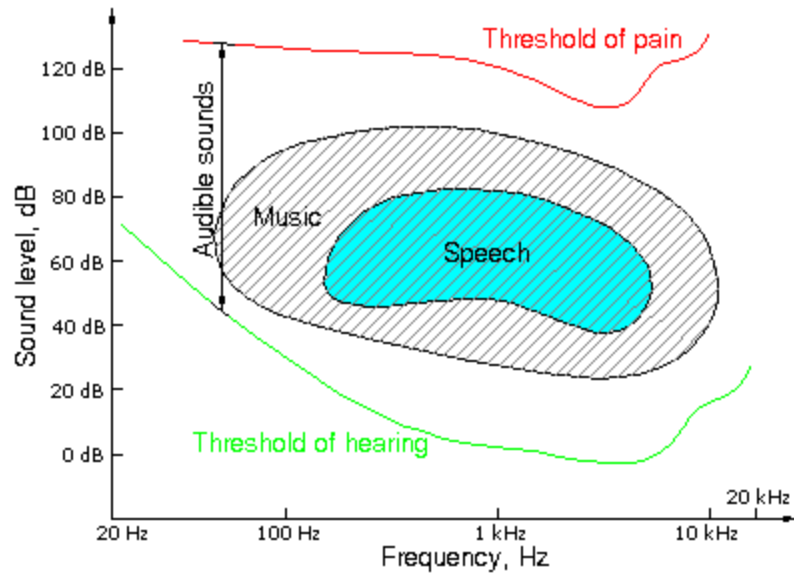
Each of these colors would represent 1 frequency and the sound level in decibels would concentrate that color accordingly. "In sound application the reference value is $20 \mu\text{Pa}$ for sound pressure or $10\text{E-}12 \text{ W/m}^2$ for sound intensity. These values give 0 dB sound level, and represent the human threshold of hearing (the lowest level that can be perceived). The threshold of pain or feeling (the level which causes pain in the ears) is about 120 dB as shown in the following drawing:" (<http://home.tir.com/~ms/concepts/concepts.html>)

Therefore, the sound intensity (SI) range for a certain frequency (F) at a specific time (t) is [0-120]. Our colors ranges are values between [0-1], the translation would be: $SI(F,t)/120$ which yields a value in the range [0-1].

```

At a time t
For(each channel C)(2)
  For(each frequency F)(10)
    Particlecolor = color(F) - (SI(F,t)/120);
    //color(F) gets the specified color for that frequency.
    Color(Particlecolor);
    drawParticle(x,y);

```



As an addition to this layout, the bowl is made to have a color fading effect from its bottom-end point up to its sides. This fading effect would be a blend of colors that reached their destination. To do this, the bowl should be done by constructing it from a set of points and then coloring these points consequently by setting the bottom point(s) to the new colors coming from the particles and updating all adjacent points by blending their own color with this color, thus creating a random color pattern to color the bowl dynamically.

```

For(all points on bowl)
  //average of all the surrounding points
  Pointcolor(r,g,b) = (eftptcolor + rightptcolor + topptcolor + downptcolor)/4;

```

The appearance of the fading color effect created on the shell of the bowl should be similar to the surface of a heavy liquid being colored by the arriving particles.

A music analysis software also can usually detect beats in bpm measure or as an onBeat() function which is made to do something when a beat is played. Beats are usually very low frequency and high intensity waves; therefore they can be detected by scanning the sudden increase in the 16Hz frequency at a specific time. Usually, music is processed some seconds before it is played to make-up for the delay to read, write, or display a certain effect. This enables us to scan some patterns like beats ahead of time. The beat here will be used to display a lighting effect coming from all sides of the blue colored circle with white nucleus behind the display as a background.

*onBeat() given time t:
lightning strip length = SI(16,t);//intensity of beat
create 6 lightning effects coming out from the circle symetrically (3 on each side)
rotate();//to give different positions to the lighnings at each beat.*

Some extensions of this idea would be to have a random movement of the bowl (Xb,Yb changing) or the jet cones after each time interval or even beat detection. Cones would look nice if they suddenly change direction and start going in the opposite sense after each beat (onbeat(): theta=-theta). Remember that this kind of visualization won't look at its best until many tries have been done and may be a lot of changes so that a lot of colors would reflect a nice eye-appealing visualisation. A basic drawing of the visual effect would look like this:

