# Project 1: Content Creation and Processing
# Due April 16, Friday, 11:59PM via TURNIN

This project includes three components: (1) create media content and understand spatial, time and frequency representations; (2) experiment down- and up-sampling with and without filters; (3) understand the structure of a colored image. These three concepts have been covered in Lecture 2 and 3.

**Part 1. Create and visualize media content**

- Create a **color JPEG** image of your choice, **no smaller than 320x240**; you can use various tools to draw or merge images, or use your camera to capture one, or take an JPEG image you like. **The image size cannot be larger than 400Kbyte.**

- Download the funky.wav file from the course website: www.cs.ucsb.edu/~htzheng/teach/cs182/funky.wav

- Use matlab's "*imread*" and "*wavread*" to read each media file into a matrix;

  ```
  [audio,fs]=wavread('funky.wav');
  size(audio)
  image=imread('yourjpegimage.jpg');
  size(image)
  ```

  **check out the size of both your audio and image.**

- Visualize audio and image
  - For the audio, you can just use *plot(audio)*
  - For the image, you need to first convert *image* into double format, and then use mesh to plot (see Lecture 3, matlab demo example) *You can rotate the axes to obtain a better view of the image.*
  - Save each plot into a JPEG file.

**What you need to submit:** (a) your matlab file as part1.m; (b) your image file; (c) the matlab plots of the audio and image;
Creat a directory as part1 and put (a)(b)(c) files into this directory.

**Part 2. Experiment with down- and up-sampling**

- Take your funky.wav file, use the above wavread to extract the data; sub-sampling by 2, that is, take only the even indexed sample of the file, and then up-sampling by 2 using zero-padding.

```
[x,fs]=wavread('funky.wav');
sound(x)

%down-sampling
y=x(1:2:end);
k=length(y);
sound(y,fs/2)

%up-sampling
z=zeros(1, 2*k);
z(1:2:end)=y;
sound(z,fs)
```

Listen to these sounds, x, y, z, and report your observations.

- now try down- and up-sampling with filter; first apply a pre-filter to limit the bandwidth, then perform down-sampling by 2 followed by up-sampling by 2; finally apply a post-filter to interpolate the samples

```
[x,fs]=wavread('funky.wav');
sound(x,fs)

%pre-filter


%down-sampling
y=x(1:2:end);
k=length(y);
sound(y,fs/2);

%up-sampling
z=zeros(1, 2*k);
z(1:2:end)=y;
sound(z,fs);

%post-filter
Add your code here…..
```

**What you need to submit:** (a) your matlab file .m; (b) the media file after down- and up-sampling; (b) the media file after down- and up-sampling but with pre- and post-filtering. (c) your observation of the audio sound with and without filtering. Save these under a sub-directory called part2

**Part 3: Solve an Image Puzzle**

You will be provided an image whose content is being divided into 4 blocks of the same size. We shuffle the blocks in random orders to form the Puzzle. Using matlab, you will re-organize the image to solve the puzzle. Note that for each R, G, B layer, we use different shuffling. This means that **you need to reorganize each layer differently, and you might need to do this manually.**

Let's say the original puzzle's block order is

1 2
3 4

and after rendering you move the blocks as

1 3
2 4

We give different images to students. Your assigned image depends on your perm number. To access your quiz, download puzzle.zip. from the following location. If your perm # ends with k, your puzzle will be puzzlek.jpg
http://www.cs.ucsb.edu/~htzheng/teach/cs182/schedule/pdf/puzzle.zip

**What you need to submit:** (a) your matlab code that reads each of the 4 blocks on each layer, and assembles them into the right order; (b) the original and the rendered image; (c) the **rendered order** of the blocks in each R, G, B layer. Save these under a sub-directory called part3.

## Submission Instructions:

You will use turnin to submit your program. Do a **man turnin** to find more information about this program.

To submit your lab

1) Create a directory whose name is your CS/ECE account. For example user John Doe whose account is jdoe will do the following

```
%mkdir jdoe
```

2) Put in the directory the three subdirectories: part1, part2 and part3 for the lab
3) Execute turnin under the name of lab1. For example, jdoe will do

```
% turnin lab1@cs182 jdoe
```

Note : You can execute turnin upto 10 times per project. Earlier versions will be discarded. The timestamp of turnin has to be before the due date. I will close turnin by the due time.