Proposal: Paint with sound

Team member: Jingwei Fan (jingweif)
Jing Xu (jxu2)

Project description:

Our web application can transform sound into images.

We can help people who's not talented in painting but want to show their feelings in paints. Or if people want to attach a image with their blog or post but can find a fit one. Or just relax yourself and have fun with our web application.

User can choose from 3 different modes: 1.The user can sing, speak, scream, shout or make whatever sound they like through a microphone, our website will paint an image according to the user input. 2.The user can upload an audio file and a more complicated and artistic image will be generated and returned to the client. 3.The user can speak a few sentences or words, and the server will choose an image somehow related to the content of the user's speech and return it to the client. Users can edit(like add some text or filters) and save their work at Paint With Sound. They can share with other users as well as post it on social network.

Planned technologies :

Google speech and text-to-speech
D3.js
Web Audio API
Web Speech API
Deep Art Effects style transfer API

Functionality:

Module 1: "What You Sing Is What You See"

The user can sing, speak, scream, shout or make whatever sound they like through a microphone, our website will paint an image according to the user input.

Description:

1. Frontend:
    a. Html design --- JX
    b. Sound Input: Record the sound of the user through microphone through Web Audio API.--- JX
    http://www.cnblogs.com/Wayou/p/3543577.html
    c. Realtime painting: Use D3 to draw a picture in frontend html. --- JWF

2.Backend

    d. Build Django and web server. --- JWF
    e. Generate output: Analysis data collected from Web Audio API in backend and send back response.--- JX

Module 2: "It Takes A Song To Be An Artist"

The user can upload an audio file and a more complicated and artistic image will be generated and returned to the client.

Description:

1. Frontend:
    a. Html design: Provide a UI for user to upload audio files. --- JWF
2. Backend analysis:
    a. Collect data: Deploy datasert on AWS server
    https://www.kaggle.com/c/painter-by-numbers/data ---JX
    b. Analyse the user-uploaded audio file in the server end through classification, pattern recognition and other data analysing techniques. ---JX
    c. Generate output: Generate an image according to the analysis results and send back response to the client.---JX
    d. Filtering: User can apply different filters to their output to make the color and style more satisfied. Using Deep Art Effects style transfer API. ---JWF
    https://github.com/chuanli11/MGANs
    https://developer.deeparteffects.com/

Module 3: "Is This What You Want?"

The user can speak a few sentences or words, and the server will choose an image somehow related to the content of the user's speech and return it to the client.

Description:

1. Frontend:
    a. Record the speech of the user through microphone using Web Audio API. --JWF
2. Backend:
    a. Backend analysis: Analysis the input using Google speech recognition API, match the text with our pre-defined class of the content and style of images. --- JX https://bam-dataset.org
    b. Generate output: Select an image from the database according to the analysis results and send back response to the client. --- JWF

Module 4: Light-weight Social Network Extention

User can interact with others by sharing, creating an album and post their image in gallery.

Description:

1. Album: ---JX
    a. Registered users can create and manage their own album, and make pictures public to have fun with other users.
    b. All users can share their picture on their social network such as Twitter, whether they have registered or not.
2. Gallery: ---JWF
    a. Users can choose to share their pictures generated with module 1 or module 2 and corresponding songs in the gallery.
    b. When sharing the picture, user can add description and/or tags to their picture, and search pictures with specific keyword in the gallery.
    c. Users can browse the Gallery according to images popularity and like others song or picture.
    d. When browsing the gallery, user can listen to the original sound of the corresponding picture.

Model:

Our application uses 3 data models except the Django's built-in User model:

1. **Painting**
   The painting created by the user along with other information.
2. **Audio**
   The audio file recorded or uploaded by the user. Each audio file is corresponding to exactly one painting.
3. **Album**
   Users can create albums to manage their paintings.

The draft implementation of the data models in Django are as follows:

```
from django.db import models
from django.utils import timezone

# Use Django's built-in User class for our users
from django.contrib.auth.models import User

# Create your models here.

# User can create albums to manage their paintings
class Album(models.Model):
    # The user to whom this album belongs to
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    # The name of album
    album_name = models.CharField(max_length=100)
    # The time at which this album is created
    time = models.DateTimeField(default=timezone.now)

# The audio file uploaded by the user
class Audio(models.Model):
    # The user who uploads this painting
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    # The time at which this audio file is uploaded
    time = models.DateTimeField(default=timezone.now)
    audio_file = models.FileField(upload_to'audio/')
```

```python
# The paintings created by users
class Painting(models.Model):
    # The user who creates this painting
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    # The album to which this painting belongs to
    album = models.ForeignKey(Album, on_delete=models.CASCADE)
    # The time at which this painting is created
    time = models.DateTimeField(default=timezone.now)
    # The audio file which cooresponding to this image
    audio = models.OneToOneField(Audio, on_delete=models.CASCADE)
    image = models.ImageField(upload_to='img/')
```