**Design Proposal**

* **Project Proposal**
  + **Project Description**:
  + The name of this project is *DJ Simulator.* I want to build something that can bring the experience of being a DJ to the players. The works of a DJ is simulated, though slightly simplified, and composed as a game: players can earn score and other things by getting good at djing. Another important feature is a career mode, in which the player earn fame and money and opportunities through djing and producing music.
  + **Competitive Analysis**:
  + The idea of this project is similar to many “simulator” types of game, though the subject is, I believe, rather unique. The first thing I think of is *PewDIePie’s Tuber Simulator*(partially because the ads just stuck inside my head all the time). I gained some artistic inspiration from that game. My project share the same basic idea of making a career, earn money and popularity. The difference is the major way of interacting with the game and the “works” that actually compose the career: *Tuber Simulator* simulate YouTuber’s works, which is simplified into clicking item and organizing studio(room); *DJ Simulator* simulate real DJing, while though simplified, players still need to sync with beat, switch songs between valid transition, etc.
  + Another similar existing project is the actual dj simulator program, which isn’t a game, rather, is an fully function simulation of djing, which support everything a dj controller can do. I can’t name the exact program, but there are many similar applications. I gained a lot of inspiration from them, but my project is not a completely accurate simulation; instead, I’m trying to approach an enjoyable and fun version of entertainment.
  + **Structural Plan** :
  + The final project will contain a source folder, containing all image files, and a music folder, containing all music available in the game. There will also be a musicInfo.csv file containing all information related to each track.
  + The code is spited into many separate files: There will be a controller class, which contain each item in a dj controller that player can interact with. The controller is composed from Turntable, Display, Fader, Effects Units, and Equalizers. Each item is a separate class. Turntable stores another class called Channel, and Display stores another class called InfoTable.
  + Controller
    - Turntable
      * Channel
        + Display Music Bar
        + Cue locations in music
        + Switch between musics
    - Display
      * InfoTable
        + Choose music from local music list
        + Load Music into channel
    - Fader
      * setVolume
    - Effects
      * playEffects
    - EQ
      * setEQ
  + Setting
    - Speaker
    - ScoreBoard
    - Lights
  + People
    - tasteOfMusic
    - dance()
    - response()
  + StudioSetting:
    - Computer
      * Produce music
    - Calendar
      * Plan future events and performance
    - Phone
      * Sign up with record company
      * Contact with manage/booking agency
  + There might be other classes I haven’t come up with at this moment.
  + **Algorithmic Plan** :
  + The trick part in this project is switching between different modes, scenes, and interfaces; and handling information among so many classes.
  + I’ll try to separate codes and functions for different game modes (which I have already tried before in Tetris bonus). Set Game mode variable to switch between modes.
  + For handling information among classes, I need to create separate function for calling different items(e.g. turntableAct()), and store every operation into data(maybe create a log variable), which other classes can access to response to current state of the game
  + **Timeline Plan**
  + Mon Aug 5:
    - Complete DJ Interface
    - Including fader and effects units
    - Complete scoring
    - Start establishing crowd class
  + Tue Aug 6:
    - Reach MVP
    - Start making studio interface
    - Start incorporating music module into the game
  + Wed Aug 7:
    - Complete most of studio interface
    - Complete making music sync up with djing
  + Thu Aug 8:
    - Complete studio interface
    - Complete saving process function
    - Starting making more detailed scoring
  + Fri Aug 9:
    - Implement more ways of djing
  + **Version Control Plan**
  + I will store the code in different location inside my computer, and upload the code to GitHub.
  + The version before MVP will have suffix of 0.X.X(e.g. 0.3.2) to keep track of each version
  + The version after MVP will have suffice of 1.X.x(e.g. 1.2.5) to keep track of each version