Curly-Squeegee Process Book

Brian Kimmig & Jimmy Moore

November 30, 2015

Abstract

Curly-Squeegee (CS) is a tool for exploring and visualizing Actor filmographies in an interactive way. By enabling users to search for their favorite actors and see their entire body of work displayed a variety of ways, we hope it invites them to explore the different views, select and filter different sub-sets of actor filmography data, and find intersting or surprising hidden trends.

Contents

1 Background & Motivation

The idea for this project came from the fact that both developers watch a good amount of movies and enjoyed sites like IMDB and RottenTomatoes, but wanted something that focused on specific actors rather than being movie-centric. CS is their solution to needing to sift through lists and text to appreciate a given actors filmography. In three views, one can see their entire body of work as a timeline, with length and color encodings for film-output and film-quality, respectively, as well as career visualizations showing the breakdown of movie genre over the course of their career and a multi-axis interactive plot to explore an actors output as a function of date ranges, ratings, box office earnings, and directors.

CS is meant to provide a new way of viewing actor data, and seeks to facilitiate a fun and interactive web-based solution to questions like:

- How many movies has an actor acted in?
- Has an actor been type-cast to a specific genre?
- What is the best movie they have made? The worst?
- Do they consistently star in well-reviewed films?
- Has their career had a golden period in which they were particularly busy, or appeared in well-reviewed films?

2 Data Collection

This application readily takes advantage of several pre-packaged movie APIs, specifically My API Films and OMdb. We have set up a web framework using node.hs and Meteor which uses a RESTful architecture to gather API calls via GET requests. We store all data in a MongoDB database. This dataset is fairly dynamic, so we rely on user queries to pull the necessary information from the APIs.

2.1 Data Processing

API requests are returned in JSON format, so there is little clean-up beyond. Returned data is fairly detailed, so we selectively cull certain uncessary fields and aggregate filmography data based on what we want to visualize. We have two data structures in our databases:

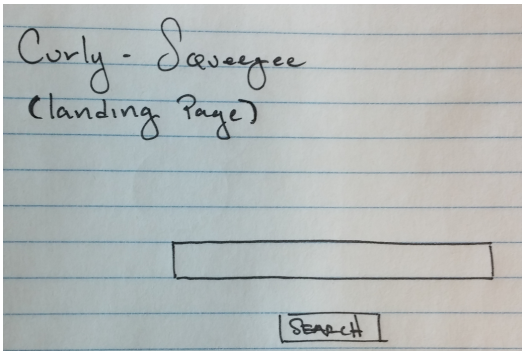
- Actor Table: This stores all the actor information of a selected actor, including the movies they've acted in.
- Movie Table: This contains all of the information for each movie we wish to plot or visualize.

The data collection and filtering is probably the most sophisticated portion of this project. With everything stored and readily accessible, we use built-in javascript math functions and agregate parameter counts of our actor data to illustrate actor filmographies.

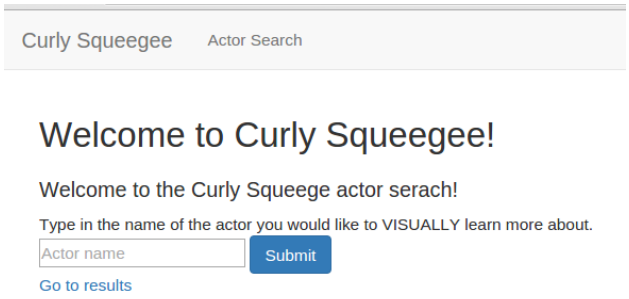
3 Project Evolution

This project has had an accelerated development cycle and we quickly settled on a fixed set of views. Details have not changed significantly from the proposal document, but we will compare and contrast any differences below.

3.1 Landing Page



(a) Original design sketch



(b) Current implementation

Figure 1: Landing page development

3.2 Loading Page

While the API calls are made and the film data is saved to the local database, we display a gif to indicate that the files are being loaded.

3.3 Display\Results Page

Once the filmography has been loaded, we load our results page, which shows a number of views and actor information

3.4 Actor Fact Box

This data serves as an orientation tool,indicating to the user whether the correct actor was returned. This is particularly important in cases when searching for actors with common names. Using the OMDB API data, we

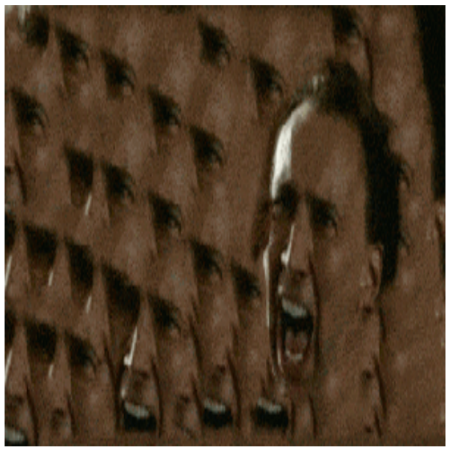
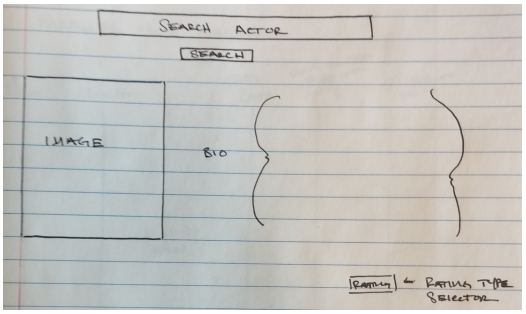
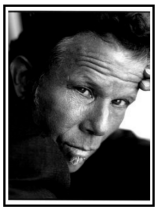


Figure 2: Once the user submits a search query, the application waits for the API calls to complete and fetch the actor's filmography. This can take upwards of 30 seconds

are able to display a photograph of the actor, their full name, date of birth, and other derived data from the API call. We currently display their total number of films. other useful information would be when they started acting \first credited role, years active, and highest or worst rated movie.



(a) Original design sketch



- **Name: Thomas Alan Waits**
- **Birthday: 7 December 1949**
- **Number of Films: 130**
- **etc:**

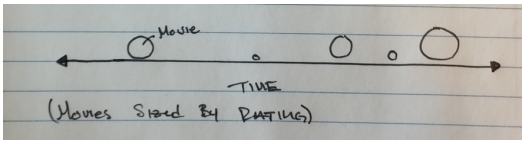
(b) Current implementation

Figure 3: Actor 'Fact Box' development

3.5 Visualizations

Talk a bit about the different views we chose, and why. This can be a cut/paste job from certain parts of our proposal.

3.5.1 Filmography Timeline

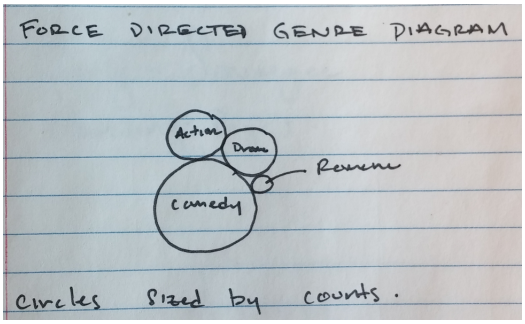


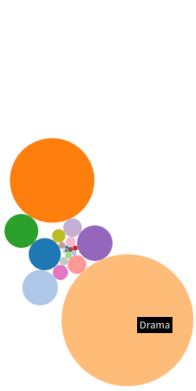
(a) Original design sketch

(b) Current implementation

Figure 4: Timeline development

3.5.2 Genre Visualization



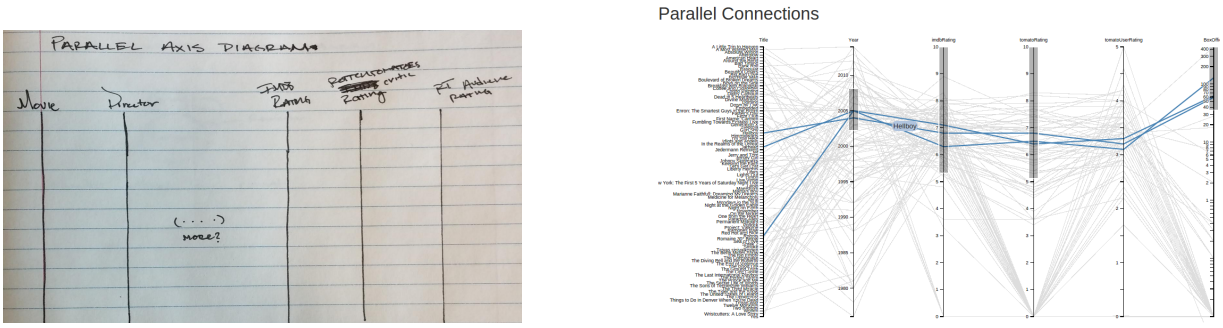


(a) Original design sketch

(b) Current implementation

Figure 5: Genre visualization development

3.5.3 Parallel Axis Coordinate Visualization



(a) Original design sketch

(b) Current implementation

Figure 6: Parallel Axis Coordinate visualization development

4 Project Feedback

After we settled on our design and project implementation, we had the opportunity to present a "sales pitch" for Curly-Squeegee to another project team to get their feedback. We met with Phil Cutler, Ariel Herbert-Voss, and Ian Sohl of the "Legion Profiling Visualization" team. They provided the following feedback:

- **Phil Cutler** (u0764757@utah.edu)

  Provided good feedback and constructive criticism. Raised concerns about Parallel axis plot readability in the limit of a long, active acting career as well as a lack of information on new actors. Liked the idea of the filmography visualization, but recommended using bar charts as opposed to circles anchored to a timeline. He thought it was a neat idea, but did not see it's utility. He admitted he does not like watching movies.

- **Ariel Herbert-Voss** (u0591949@utah.edu)

  Overall very positive and excited reaction. she loved the parallel axis plot idea and also agreed with Phil that a bar chart for the filmography timeline would be more effective. She suggested scaling bars either by film rating or number of films in a given period (for a drill-down style barchart), as well as shading a given bar to convey additional information. Ariel is a film buff and saw a great deal of utility in this visualization

- **Ian Sohl** (u0445696@utah.edu) No additional feedback beyond what Phil and Ariel had to suggest.

5 Team Evaluation

Brian Kimmig: Brian was responsible for the API calls, data collection, and database wrangling. His experience as a web developer was very helpful for making this portion of the project go very smoothly. Brian also created the project framework, and parallel coordinate view.

Jimmy Moore: Jimmy was the lead scribe for the group and was responsible for project documentation including the proposal and process book. He also coded the Actor photo and display box features and filmography timeline visualization.