BRYCE MORROW

bam4564@live.unc.edu – (239)233-4556 - Chapel Hill, NC - https://github.com/brycemorrow4564

If demo links are not active, you can download a hyperlinked copy of my resume at: http://bit.ly/resume-bryce-morrow

Education

University of North Carolina at Chapel Hill

Master of Science in Computer Science — ***Expected Graduation:*** *May 2020*

B.S. in Computer Science, Minor in Mathematics (3.53 GPA) — *August 2015 – May 2018*

Courses: Machine Learning (ML), Generative Methods in ML, Computer Vision, Robotics, Distributed Systems, Internet Services / Protocols, 2D Computer Graphics, Operating Systems, Files and Databases, Data Structures, Computer Architecture, Bioalgorithms, Combinatorics, Linear Algebra, Probability Theory, Multivariate Calculus, Numerical Analysis

Experience

UNC, Quantitative Methods for Biomedical Big Data Research Group — Chapel Hill, NC

Graduate Researcher / Full Stack Web Dev + Data Visualization — *August 2018 – Present*

- Prototyped a full-stack web app, *PrecisionVissta,* with a front-end interface, built using **JavaScript/D3/React/ Redux/Antd,** connected to a **RESTful API** backend created using **Python/Numpy/Pandas/Flask** to support visual analysis workflows of biomedical health data. Bundled with **Webpack**.
- Generalized core visualization algorithm from *PrecisionVissta* as a technique, called *PeripheryPlots* (REPO, DEMO), for multi-scale contextual visualization of multivariate temporal data (see publications). Developed an open-source implementation of the technique using **React/D3**. Authored documentation on component use and extension.

Sciome, LLC — Raleigh, NC

Software Engineer / Full Stack Web Dev + Data Visualization — *June – October 2018*

- Created a full-stack web component for visually exploring and querying 3D network graph structures.
- Implemented component with **server-side rendering** for quick page loading, synchronizing state between a **Polymer** component, containing a **Three.js** based visualization, and a **Java/Vaadin** server using a **Remote Procedure Call** interface. Queries were supported with **Elastic Search.**

SAS, Inc. — Cary, NC

Software Engineer / Web Dev — *May – December 2017*

- Created a client-facing web app with a **Model View Controller (MVC)** frontend built with **OpenUI5** connected to a **RESTful API** backend for compute cluster configuration and monitoring (SAS Grid Manager).

Publications

Bryce Morrow, Trevor Manz, Arlene Chung, Nils Gehlenborg, David Gotz. Periphery Plots for Contextualizing Heterogenous Time-Based Charts. *IEEE Visual Analytics Science and Technology (VAST)*, Vancouver, B.C (2019).

- *Selected for *Best Short Paper Award* from over 180 submissions*

Projects

Procedural Geometric System for Generating "Trippy" Visuals – (DEMO)

- Designed and implemented (using **Three.js**) a novel geometric algorithm that generates distributions for positions and surface normals for rendering 3D planar geometries along cylindrical tunnels.
- Developed an animation engine for creating a sequential chain of interpolations between different static configurations of this generative geometric system. Built GUI controls for the engine using **React**.

Multi Net-GAN - (PAPER)

- **Generative Adversarial Network** with a **Recurrent Neural Network** as both generator and discriminator built with **Python/TensorFlow/Numpy** that learns an implicit probability distribution of random walks over a multiplex network graph structure to model inter- and intra-layer connectivity structures. Applied model to a social network dataset and predicted the existence of unseen edges with a precision of 71%.

Cryptocurrency Market Analysis App (DEMO)

- A full-stack web app for cryptocurrency market data analysis supporting visual comparison and correlation analysis of financial features as well as providing text alerts for monitoring crypto-related subreddit growth. Frontend built with **OpenUI5/jQuery/HighCharts**. Server/Web Scraper implemented with **Node.js/SQLite/Request/Cheerio.**

Rock Stacking Algorithm (PAPER)

- A physics-based stochastic/greedy sequential target pose planning algorithm for constructing a stack of convex rigid body objects implemented with **Python/PyBullet/Scipy/Numpy/Docker.**