OnIt: Take Responsibility Of Your Own Focus

Competitive Analysis:

Terminology:
- Facial landmarks refer to landmark points assigned to X number of noticeable feature of the face
- Eye features refer to flattened matrix of pixels within a rectangular region of the eye (for WebGazer, it was a 6x10 pixel wide image patch flattened)

WebGazer:

WebGazer is an entirely browser-run, real-time gaze tracker. Users can either calibrate the prediction algorithm to fit their faces using a calibration interface, or skip this step entirely and allow the algorithm to self-calibrate as the user interacts with the web. In the former case, calibration is achieved by fitting parameters that 1) best map pupil locations (relative to rectangular "box" drawn around eyes using facial landmarks as reference points) to locations on-screen that user clicks 2) best map eye features itself to locations on-screen that user clicks. For the latter case of self-calibration, WebGazer incrementally collects "ground-truth" data about relationships between pupil locations, eye features and the locations on-screen that they map to. It does so by assuming the users' gaze always coincides with the mouse cursor when it is moving, and when it clicks on a location. Naturally, the internal parameters that predict the actual gaze are updated as new "ground truth" mappings between eye features, pupil locations and screen position are inputted through user interactions.

Despite using a Kalman filter and well-optimized algorithms, WebGazer's predicting power is limited (standard error of 72 pixels from actual gaze location). These shortcomings naturally arise from the fact that 1) web-cameras have lower resolution and are sensitive to background light and 2) as a web-application WebGazer sacrifices accuracy for speed. On the second point specifically, WebGazer utilizes straightforward linear ridge regression to compute parameters that are resilient to overfitting, yet is computationally cheap. As a point of contrast, desktop-application gaze predictors like PACE (Huang et al, 2016) and TukerGaze (Xu et al, 2015) that do not perform 3D calculations are able to achieve higher accuracy by using more powerful statistical models and by gathering greater training data for these models by lengthening calibration sessions. As a web-application aiming to be deployable anywhere on the web, WebGazer cannot afford to have the above factors that desktop gaze predictors.

Algorithmic Plan:

Most Complex/Algorithmic Solutions:

- Detecting pupil from eye features: use a summed area table to find the darkest area within the eye features, working under the assumption that the pupil of the eye is contrasted most sharply with its surrounding versus other eye parts.
- Training eye features to mouse-click locations: collect eye features corresponding to mouse-click points (perhaps use caching to save memory, as eye features are quite large and as it accumulates memory taxes) and train a linear ridge regression model using eye features as X and mouse-click as Y. Then retrieve the parameters from the training set, and average it out with other parameters (if the user chooses to do multiple calibration rounds) to reach the most general parameter values. Also be able to save/retrieve parameter values when closing/opening calibration stage based on user profile (login system).
- Calculating concentration feedback score/metric based on user settings: allow users to draw boxes/fields of workspace where they will be looking during work. Find the frequency the user's gaze deviates from the specified box while the session is in progress. Show users how often, for how long, and in what types of sessions they lose concentration from work the most.

Time-Line Plan:

Week 1 (Nov 17 - Nov 21)

- Home screen UI
- Calibration mode added to Home Screen
- (skeleton) Focus mode added
- (skeleton) Analytics mode added

Week 2 (Nov 22 - Nov 28) work like hell during thanksgiving break in my mom's basement (real plan, no joke).

- Theme: Finish Focus Mode
- Allow users to draw "focus" areas using mouse
- Allow user to input:
    - Name of Task
    - Duration of Task
    - Purpose of Task
- At the end of task duration, user required to input:
    - Reflections on focus
- At the end of task duration, user will receive:
    - Analytics that was undergoing in background

Final Stretch (Nov 29 - Dec 1)

- Theme: Save Parameters for Calibration / Focus Session Details
- Create user sign-in system
- Create pipeline for creating / storing user's personal parameters and records of focus sessions into auto-created folders
- Touch up UI.

Modules:

- dlib
- openCV
- scikit learn
- numpy
- PIL

StoryBoard:

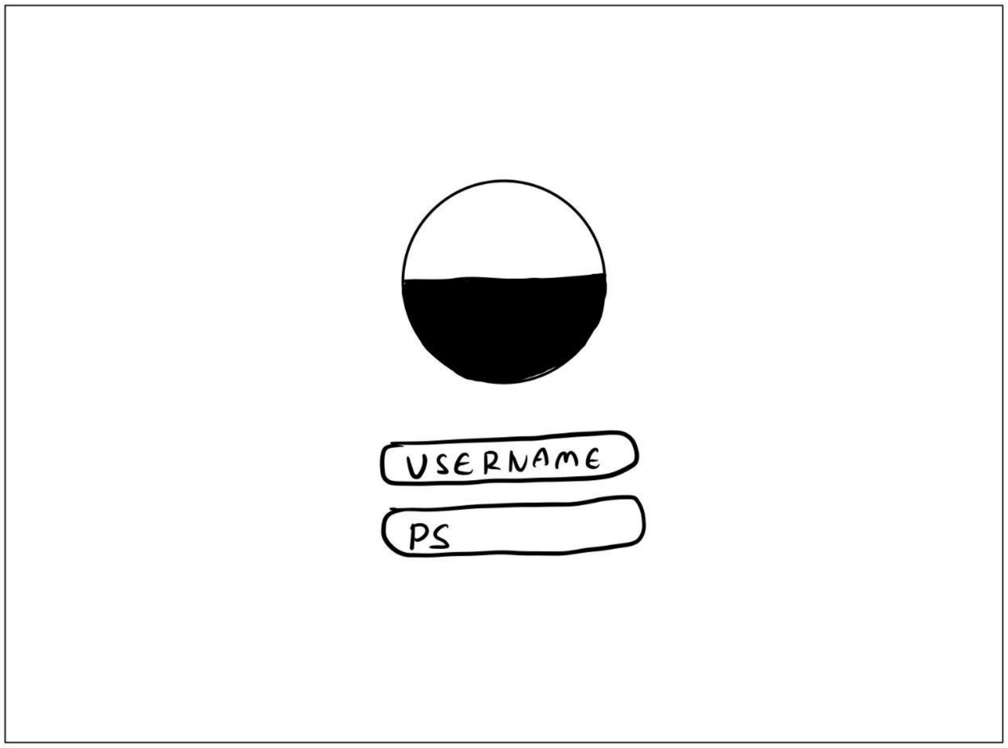
TP2, Nov 23 Updates

- Add box-drawing mode, where users can specify the area on screen that they aim to focus on. Will be a separate window part of the create focus session screen.
- Analytics based on proportion of time during which the user's predicted gaze was outside of user-drawn focus area.

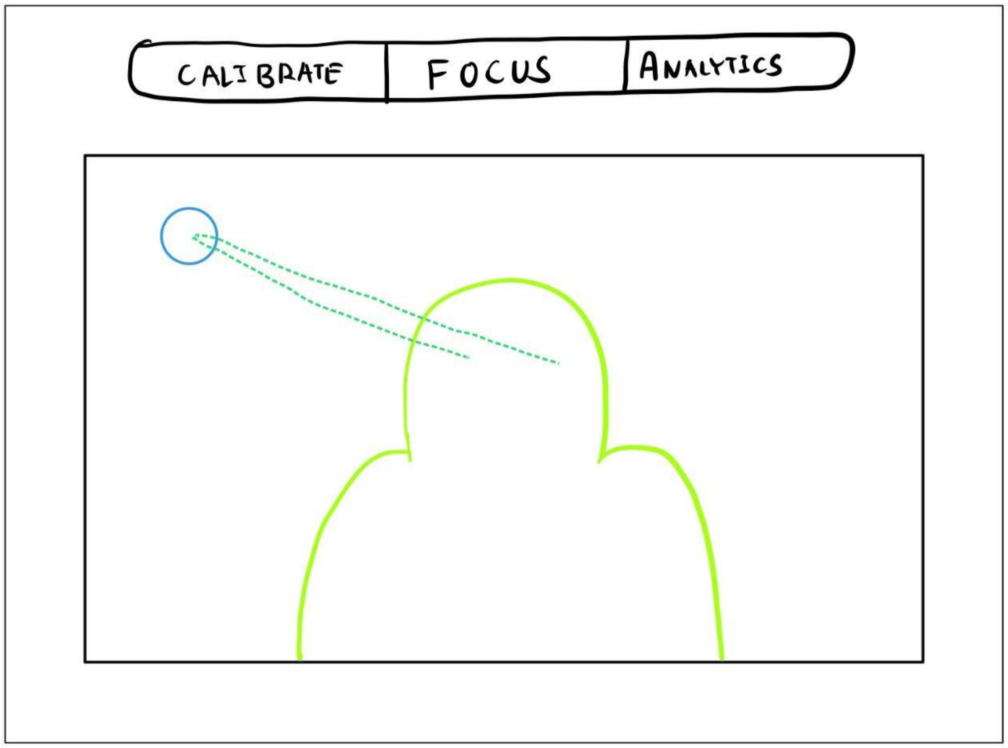
TP3, Nov 30 Updates

- Box-drawing mode added as 'palette' mode, users can now specify which area there gaze will be tracked in.
- Analytics visual system added. x-axis = time, y-axis = ratio of focus between 0-10 (Gaze in Box per minute / 60 sec * 33 fps) * 10

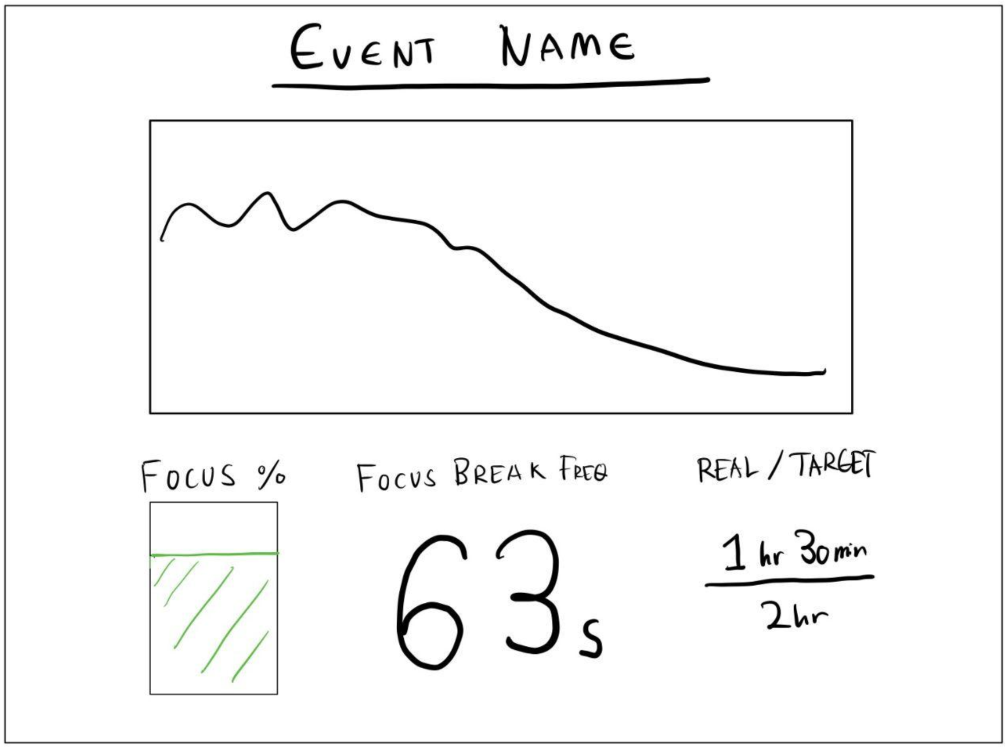
LOGIN SCREEN



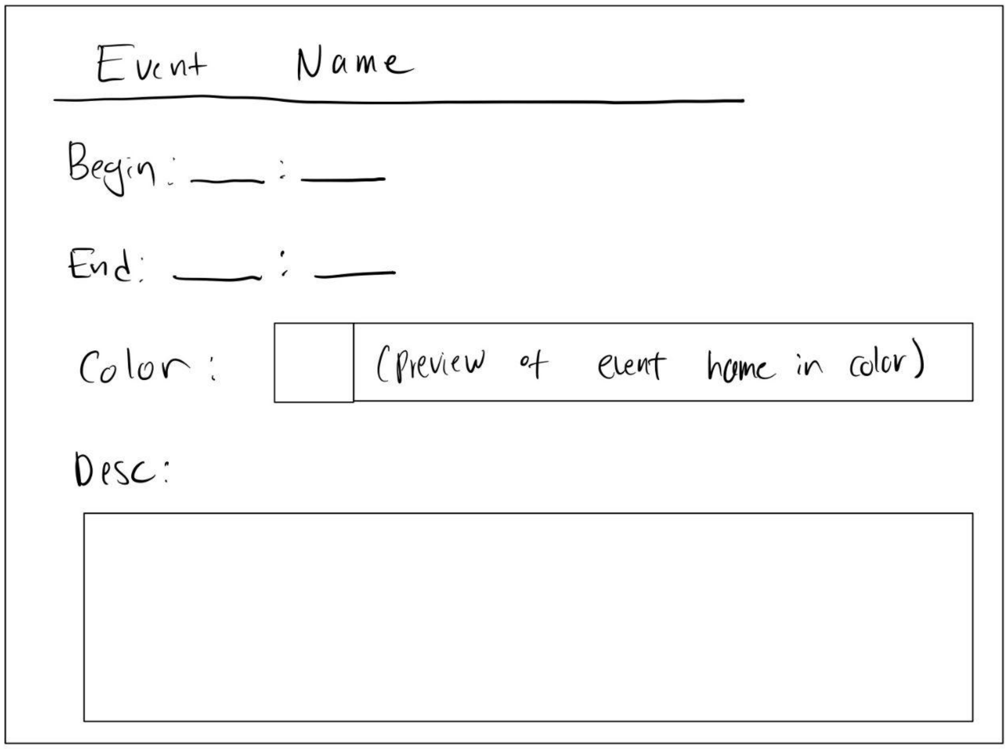
Home Mode



ANALYTICS MODE



Focus Session Creation Mode



Calibration Mode