Software Design Description
for the
Track and Control System

Version 1.0

Robert Moss, Aaron Periera, Matthew Shrago

February 20, 2014

Contents

1 Introduction

1.1 Purpose

The Software Design Document (SDD) is intended to describe the software components of the Track and Control System (TACS). The SDD will include details about the software implementation, the database handling, and the overall design.

1.2 Scope

1. The software will track a user's movements and be able to control numerous features around their Desktop with the movement of their head.
   - The everyday computer user will utilize this software to organize their cluttered Desktop.
2. The application will also act as a security monitor to recognize when you're present at your computer and lock your screen accordingly.
   - A benefit of this will be a sense of security for the user when they're away from their computer.

1.3 Overview

The SDD will contain explicit details about the TACS software. Distinct acronyms and definitions will be clarified. References for our specified libraries will be included. A system overview of the software architectural design as well as the database design will be covered. Finally, the proposed interface design will also be established and talked about.

1.4 Reference Material

The list of references below are software documentation that we will be using:

1. OpenCV documentation: http://opencv.org/
   - FaceRecognizer API: http://docs.opencv.org/trunk/modules/contrib/doc/facerec/facerec_api.html
2. Windows API Index: http://msdn.microsoft.com/en-us/library/hh920508(v=vs.85).aspx
3. QT C++ documentation: http://qt-project.org/

1.5 Definitions, Acronyms, and Abbreviations

- Application Specific Definitions
  - TACS - Track and Control System
  - TM - Tracking Module
    - OT - Object Tracker
    - FRT - Facial Recognition Tracker
  - WCM - Windows Control Module
    - WGO - Windows Grid Organizer
    - WP - Windows Perspective

    - SM - Settings Module

- Industry Definitions
    - SRS - Software Requirements Specification
    - OpenCV - Open Computer Vision: An open source library for object tracking via the camera.
    - SQLite - A lightweight, low maintenance, self contained local database.
    - DB - Database
    - RGB - Red, Green, Blue color values.
    - HSV - Hue, Saturation, Value.
    - API - Application Programming Interface
    - C++ - An object oriented programming language.
    - GUI - Graphical User Interface
    - QT - An API for building GUIs

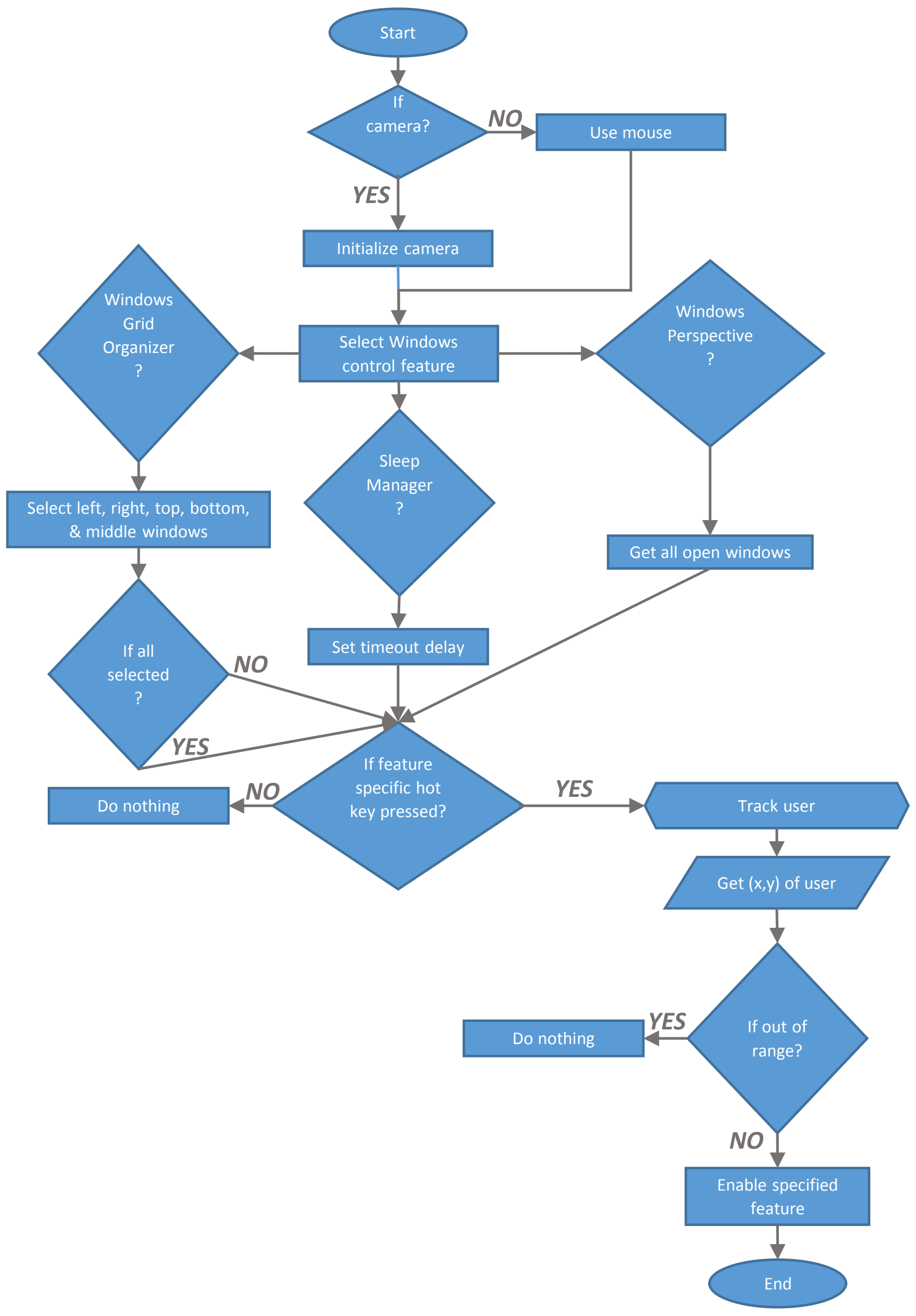
2 System Overview

There will be many available functions the users can utilize. The modular design of the software will allow for additional functions to be easily implemented. *It should be noted that the window functions will be activated with a user-defined hot key.*

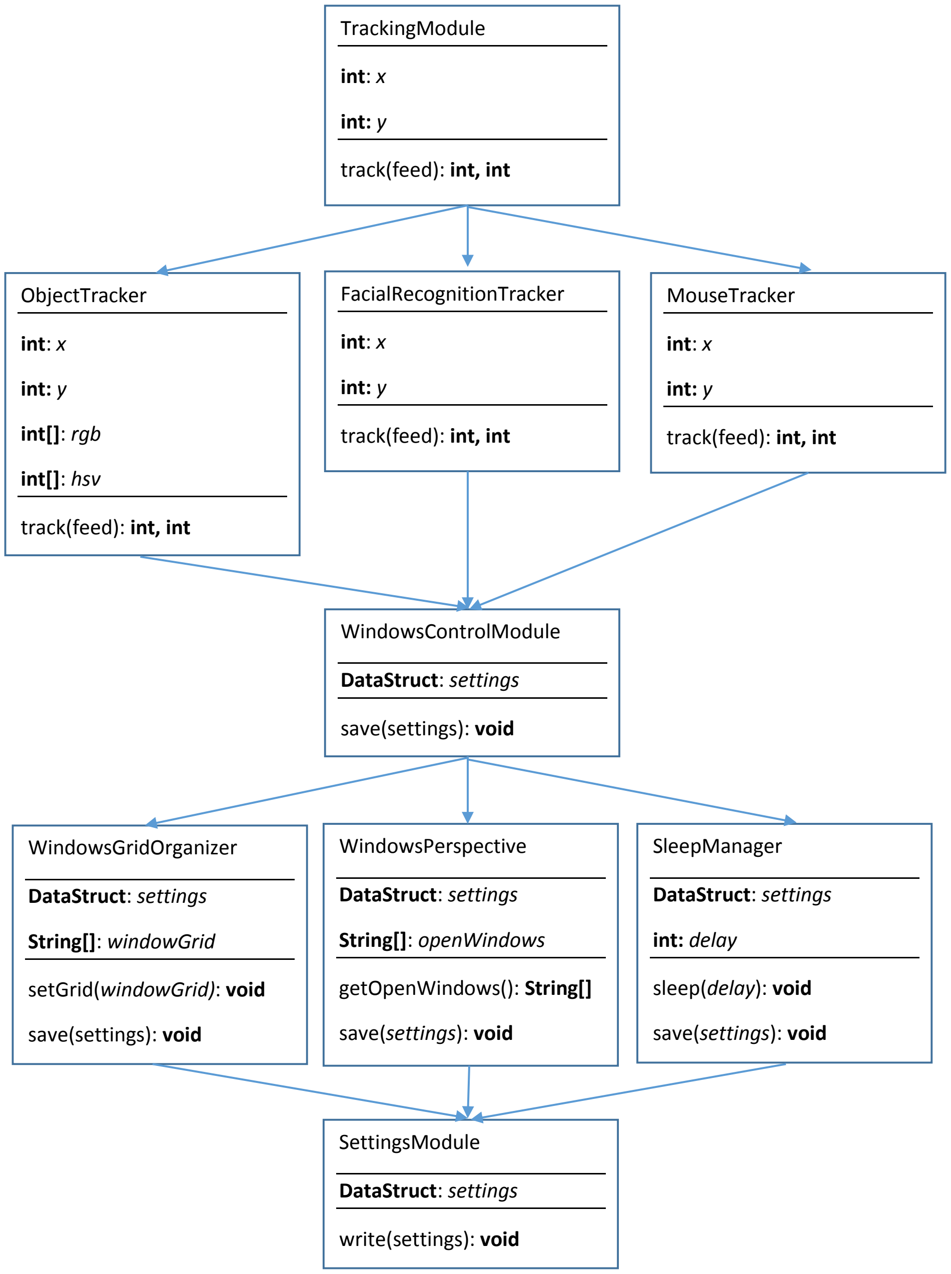
1. A proportional snap to grid set up for users to place desired windows in five different locations. Namely, left, right, top, bottom, and middle. Once all the windows are selected, the middle window will fill the screen and the user will be able to "peer" in the direction of the four other windows to show a preview of them.
2. The software will organize all open windows into a layered view (with the illusion of 3D windows) for the user to "look" around their desktop and see which window they want to select.
3. The software will be able to detect when the user is away from the screen, and with a user defined delay-time, be able to lock or put your computer to sleep (all settings can be changed from the SM).

3 System Architecture

3.1 Architectural Design



3.2 Decomposition Description



3.3 Design Rationale

The design is highly modular. The justification for this is because there are several different ways to complete the tasks at hand. Tracking the user can be done in numerous ways, as long as it follows the TrackingModule interface and outputs the set of x and y coordinates. This is the same for the WindowsControlModule. New features can be added into the system due to the modularity. As long as the new feature follows the output data structure that the others obey, it can be easily incorporated into TACS. The SettingsModule is completely separate from the main functionality of TACS; allowing it to write to the database and solely worry about that process.

4 Data Design

4.1 Data Description

There will be a data structure, defined by the developers, that will hold all the necessary data to be written to the SQLite database. The data structure will be represented in the same way as the Data Dictionary below. Most components will have settings that need to be remembered. The dictionary below outlines which component will have the specified data associated with it.

4.2 Data Dictionary

name	type	size
	ObjectTracker	
red	int	8
green	int	8
blue	int	8
hue	int	8
saturation	int	8
value	int	8
	WindowsGridOrganizer	
left	String	32
right	String	32
top	String	32
bottom	String	32
middle	String	32
speed	float	32
left_right_ratio	int	32
top_bottom_ratio	int	32
	WindowsPerspective	
speed	float	32
	SleepManager	
delay	int	32
hibernate	bool	–

5 Component Design

5.1 TrackingModule

The TrackingModule will be the supper-class for the ObjectTracker, FacialRecognitionTracker, and MouseTracker. Each sub-class will hold the x and y values of the tracked user. The $track()$ function will be implemented differently for each sub-class. The ObjectTracker will need the RGB or the HSV values set by the user. The FRT won't need any extra variables, as it will recognize any face, not a specific face. Lastly, the MouseTracker will simply track the coordinates of the mouse and send them out as the x and y outputs.

5.2 WindowsControlModule

The WCM will act as a super-class for the WindowsGridOrganizer, WindowsPerspective, and SleepManager. All sub-classes will have the DataStruct, *settings*, to be updated and set out as output. The WGO will keep track of which window was selected for each grid section in a String array. Then it will set the grid positioning with $setGrid()$. The WP will gather all the open windows to be resized and projected. The SleepManager will run the $sleep()$ function with the user defined delay (defaulted to a specified value). Each sub-class will output the DataStruct directly to the SettingsModule.

5.3 SettingsModule

The SettingsModule will write the inputted DataStruct to the local SQLite database. Default values will be handled here.

6 Human Interface Design

6.1 Overview of User Interface

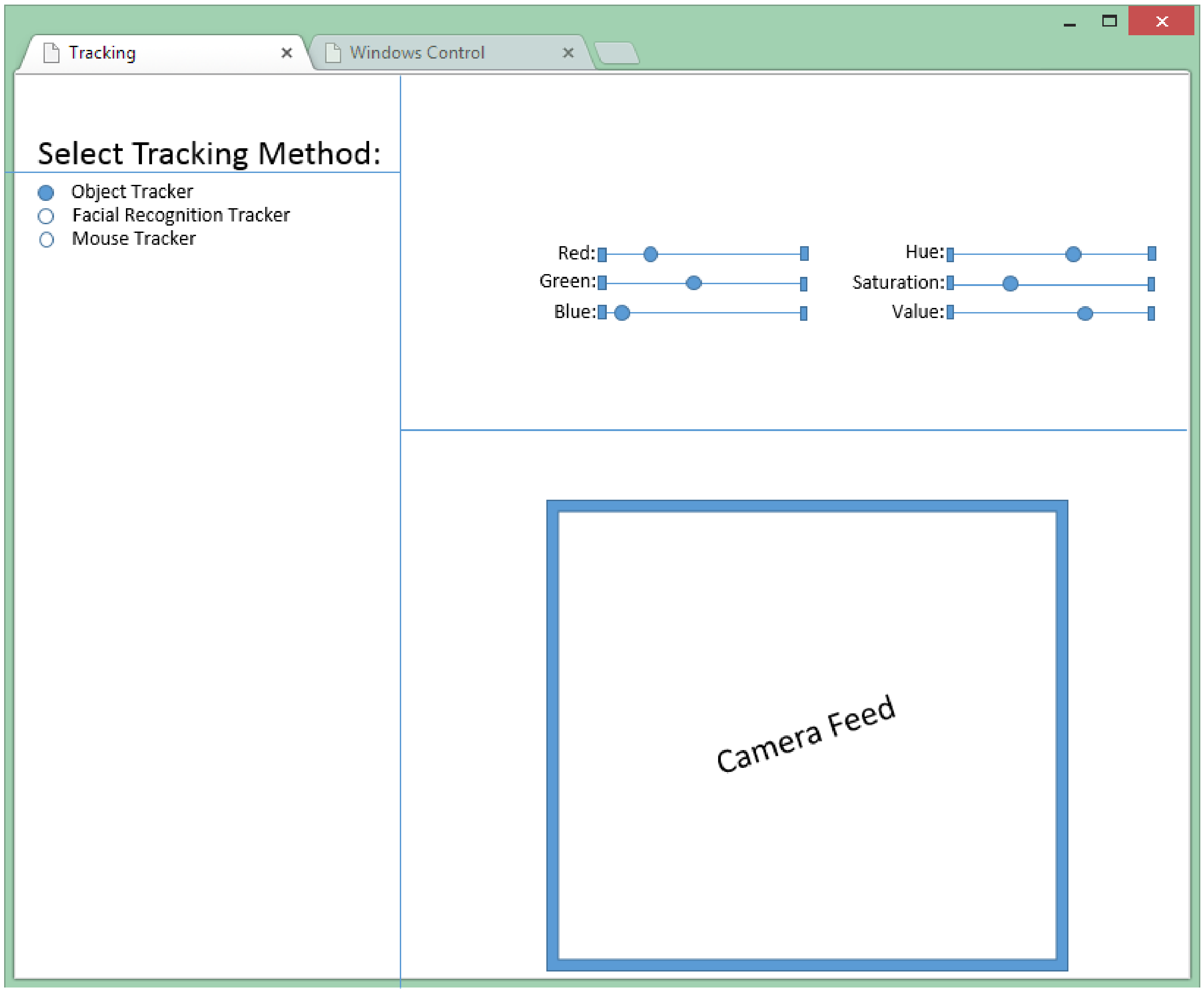
The GUI will be very minimal, seeing as most of the software features will utilize the open windows. A tabbed view of the Tracking options and the Windows Control option will be present.

Within the Tracking tab, the user will be able to change the settings for which tracking method they prefer. They will also be able to see the camera and how it's tracking.

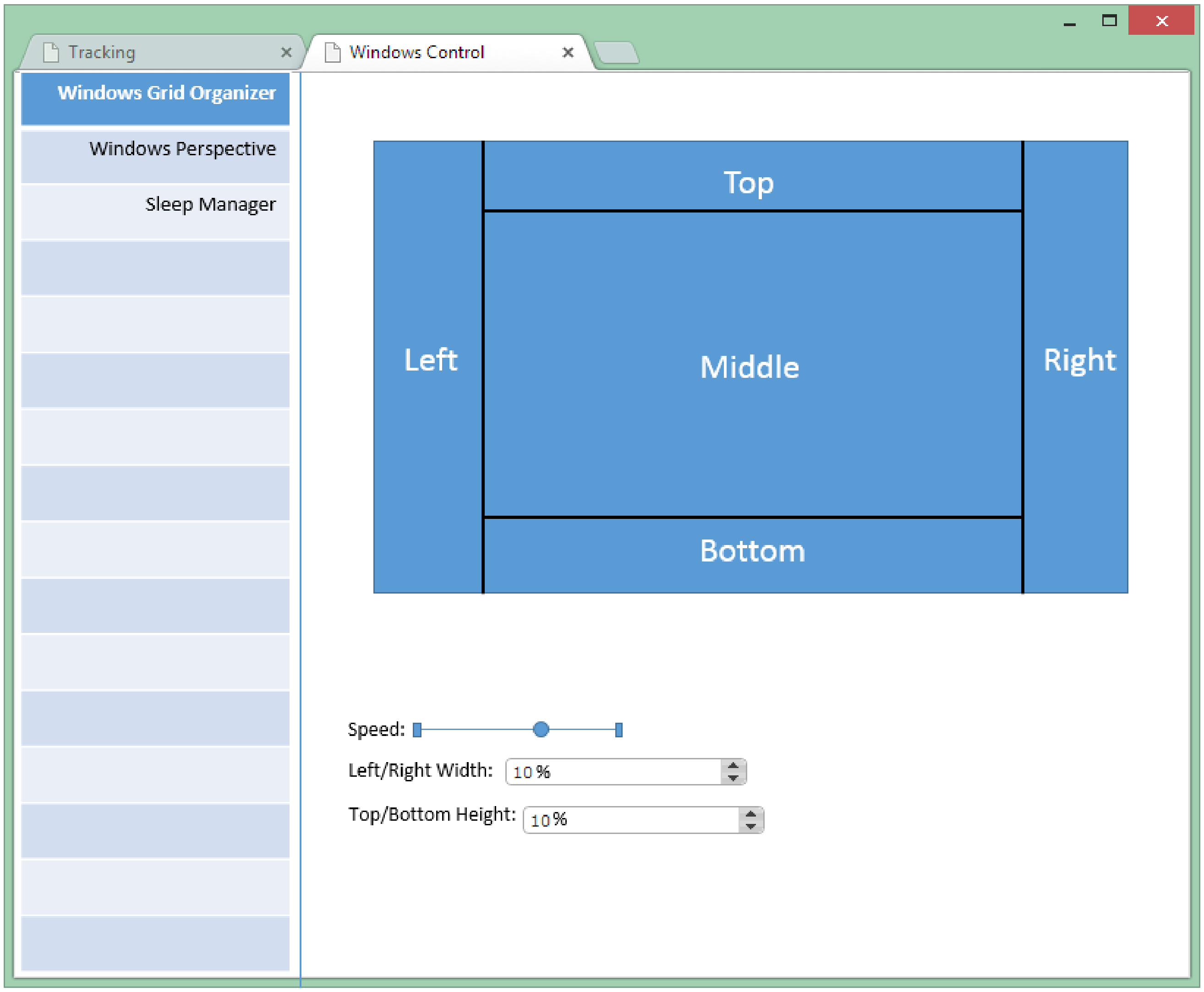
The Windows Control tab will have a list of available features. When clicking on an item in this list, the user will have the options for that feature presented to them. Keeping each features options separated in the list will enable the addition of features to be easily implemented.

6.2 Screen Images

6.2.1 Tracking Tab



6.2.2 Windows Control Tab



Speed:

Left/Right Width: 10 %

Top/Bottom Height: 10 %

6.3 Screen Objects and Actions

Only one tracking method can be selected at a time. The settings for that method will appear to the right when it's selected (Object Tracking shown).

The Windows Control tab will include the list of features. The spinners for the Left/Right Width and Top/Bottom Height will be a percentage from 0-100 of the screen width and height, respectively.

7 Requirements Matrix

SRS	Functional Requirement	SDD
3.2.1	Tracking Module	5.1
3.2.2	Windows Control Module	5.2
3.2.3	Settings Module	5.3