Object Design Description
for the
Track and Control System

Version 1.0

Robert Moss, Aaron Periera, Matthew Shrago

March 6, 2014

Contents

1 Introduction

1.1 Object Design Trade-offs

1.1.1 Buy vs. Build

There is no product on the market that allows a user to control certain features around their Windows environment simply by tracking ones head. For this reason, we will have to build such a system, because buying it is not an option.

1.1.2 Space vs. Speed

In TACS, space is not a major priority. With only settings being kept, speed is the most crucial; as a fluid motion experience is what it is striving for. The latency at which the program registers the users movement will rely heavily on speed. The efficiency of the implementation will be directly related to the user experience (pertaining to speed).

1.1.3 Delivery Time vs. Functionality

If the development of TACS is behind schedule, certain tracking algorithms and windows features can be scrapped. The modularity of the design allows for this. Also, it's worth mentioning that TACS only needs one method of tracking, yet it's nice to give the user options.

1.1.4 Delivery Time vs. Quality

If testing runs behind schedule, the software can still be released and updates can be released to fix bugs at a later date. As long as the core functionality and at least one tracking module has been implemented, the product can be delivered.

1.1.5 Files vs. Databases

Files would not be as beneficial as a database because the data for TACS is not voluminous. A database would provide a sufficient structure to organize user settings. In using a file, users can theoretically edit and thus corrupt the file and the data inside. An ”off site” database would be more secure and less likely to be modified.

1.2 Interface Documentation Guidelines

Classes*, *Interfaces* & *Packages: These names should be in *Pascal Case.*

- i.e.: ObjectTracker, TrackingModule, TrackingPackage

Constants: All constants should be entirely in *Upper Case.*

- i.e.: RESOLUTION

Identifiers* & *Methods: Identifier and Method names should be in *Camel Case.*

- i.e.: xPosition, yPosition, setWindow()

Local Variables: Variables should be in all *Lowercase.*

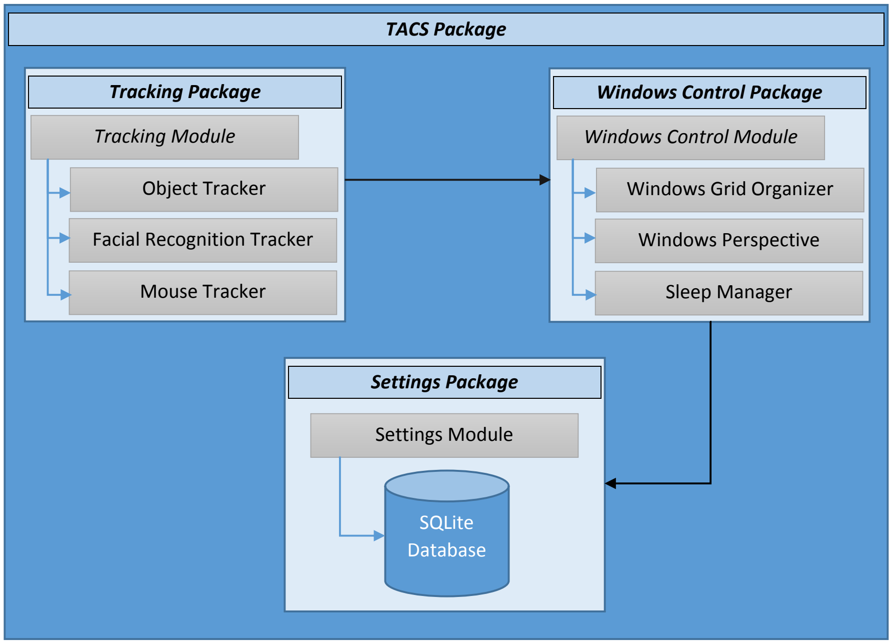
- i.e.: speed

1.3 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
  - ODD - Object Design Description
  - 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.

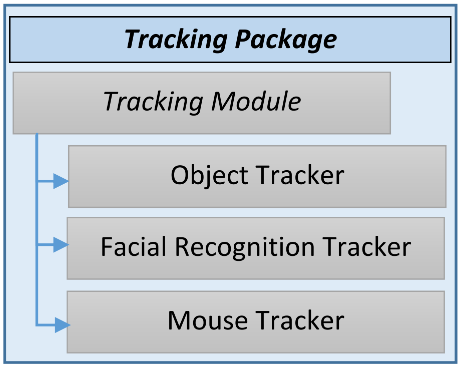
2 Packages

2.1 Package Diagram



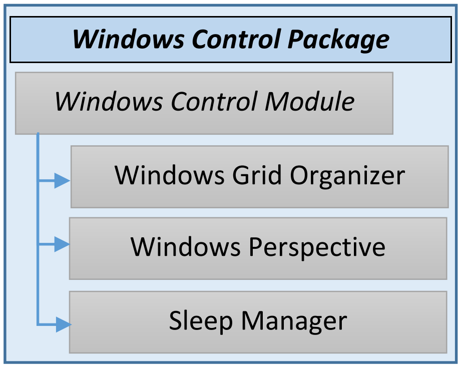
2.2 Package Definition

2.2.1 Tracking Package



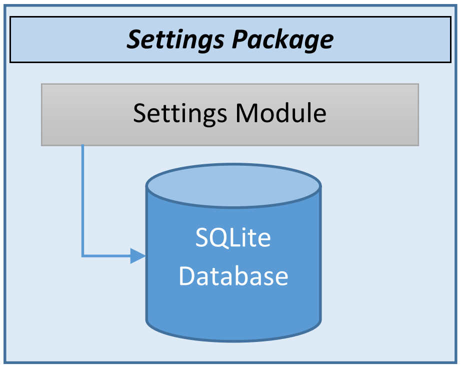
The tracking package will consist of the TrackingModule interface, and it's three sub-classes. The Tracking Module cannot be instantiated itself, only allowing the sub-class trackers to be initialized.

2.2.2 Windows Control Package



Similarly to the tracking package, the windows control package will consist of the WindowsControlModule interface, and it's three sub-classes. The WindowsControlModule cannot be instantiated itself, only allowing the sub-class features to be initialized.

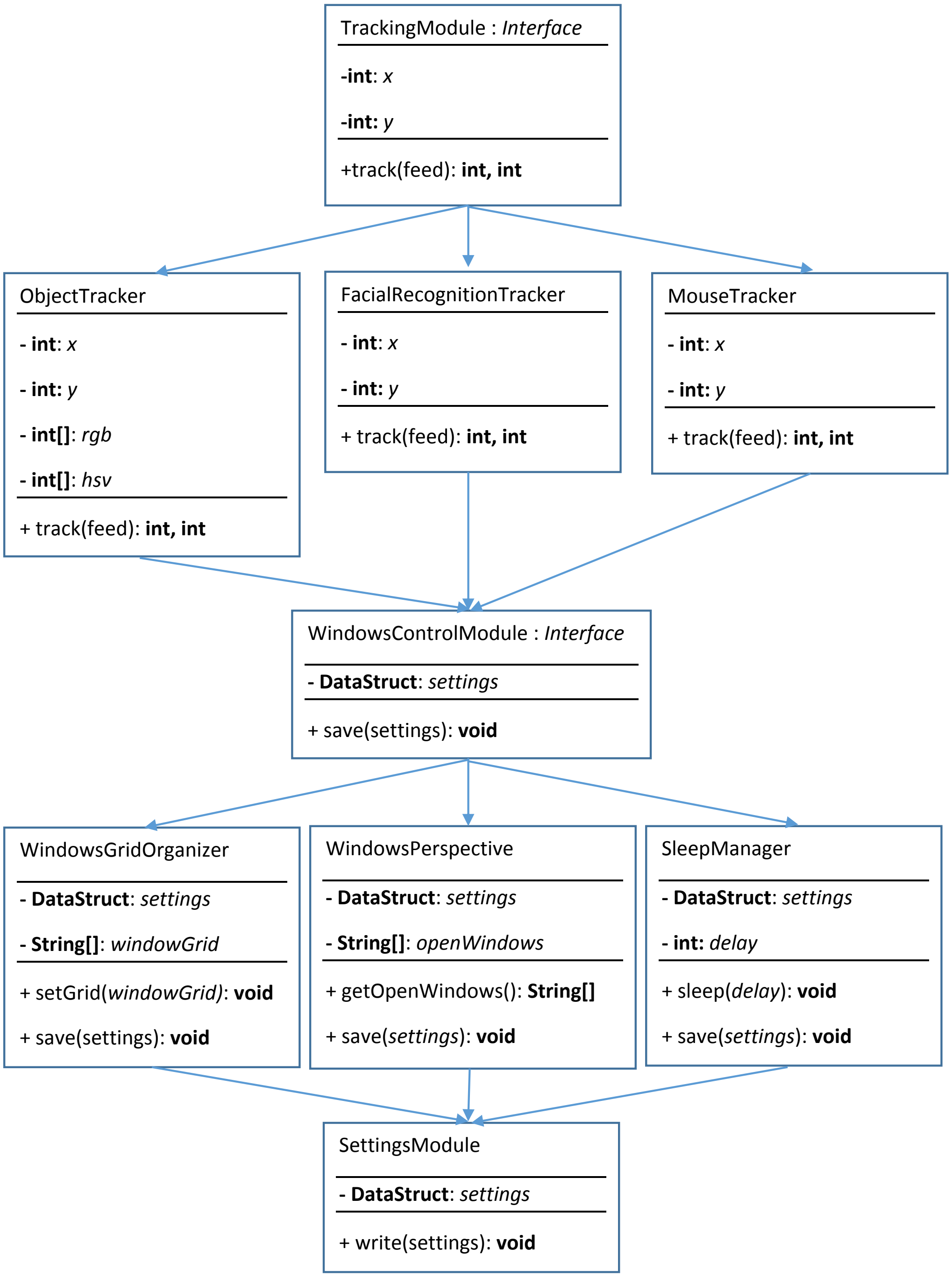
2.2.3 Settings Package



The settings package will contain the SettingsModule and the local SQLite database. The SettingsModule is the only class that will write directly to the database. All other classes will pass it's information that needs to be written straight to the SettingsModule.

3 Class Interface

3.1 Class Diagram



3.2 Class Definition

3.2.1 TrackingModule

The TrackingModule will be an interface and super-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.

3.2.2 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.

3.2.3 SettingsModule

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