Top Level design

Thursday, 20 October 2011
8:10 p.m.

Thread ONE (Activity Monitor)

Primary functions

- Runs continuously
- Records system activity state changes by placing them on a shared FIFO buffer
- The system is considered to be in the IDLE state if it is beyond the min idle time amount

Secondary functions

- Records system state to file every 10 sec
    - Records as ACTIVE if the system has had activity within that time slice
    - Records as INACTIVE if no activity in the time slice
    - Records as IDLE if the idle state has been reached
    - The associated time should be included with each entry
    - File should be in CSV format using a TAB as the delimiter
    - File should be saved using the current days date
    - One file per day

Thread TWO (Real time GUI)

Primary Functions

- Responsible for maintaining the total task time
- Consumes the buffer produced by the activity monitor thread
- Displays a prompt to the user for every IDLE->ACTIVE state transition
    - Should offer a selectable list of options to the user
        i. Add the idle time to the total
        ii. Add the idle time to the total less *x* hours:mins
        iii. Add *x* amount to the total time (should be less than the total idle time)
    - If incorrect amounts are entered by the user a msg indicating how they can correct it should be displayed on the prompt, and they should not be able to continue until they have done so
    - Should show the idle time period being dealt with
    - Should show the current total task time leading up to the start of the IDLE state
    - Should show the total time after the selected option has been applied
    - Should have two submission buttons
        - **Wipe Idle Time**
        - **Add Idle Time** (has focus by default)
    - A third button **Snooze** should hide the window until the next IDLE->ACTIVE transition happens or until a timer expires (time TBD)
    - Closing the window should add the idle time to the total, but should warn the user first

Structures

Thursday, 27 October 2011
7:35 p.m.

System Activity State

The state of the system for a given time slice.
One time slice is the period of time examined in one iteration of the activity monitoring loop.
Activity states:

- Active - System has had activity
- Inactive - System has had no activity
- Idle - System has been in the Inactive state for at least the minimum idle time (configurable)

Activity Trace

A trace of system activity state changes.
Will take the form of a FIFO buffer.

Produced by the ActivityMonitor thread.
Consumed by the RealTimeGui thread.

Available states to be traced:

- Idle
- Active

Data stored:

- The state that the system has changed ***to***
- The wall time that the state change occurred

Time Slot

A generic slice of time.
Has a start and end time (and implicitly a corresponding duration).

Unique Time Slot (Extends TimeSlot)

A *unique* slice of time.
Uniqueness is to be managed by a TimeSpace instance.
Each UniqueTimeSlot will have a reference to the time space that created it.
All operations that modify the UniqueTimeSlot will be done via the manager.

Time Space (Implements UniqueTimeSlotFactory)

A factory object for owning UniqueTimeSlot objects - namely to maintain their uniqueness.
A single TimeSpace should be created to ensure that all UniqueTimeSlots created from it are garrenteed to be unique always. Beyond creating new UniqueTimeSlots, the interface should not be used directly, other than by UniqueTimeSlots themselves.

All operations should be thread safe.

Methods:

- Allocates new UniqueTimeSlot objects
- Authorises changes to a UniqueTimeSlot start time and/or end time
- Releases a UniqueTimeSlot to another manager (not in version 1.x)

Data:

- An ordered list of used time allocations (prob a list of standard TimeSlot objects)

Unique Time Slot Factory (Pure interface)

Has a single method:

- GetNewUniqueTimeSlot()

Task

Consists of a series of UniqueTimeSlots that when combined, form the total time spent on the task.
May also have any number of child Tasks (forming a tree structure with a single task at the root).
Note: In version 1.x child tasks will not be implemented, only the root task.

Methods:

- GetTotalTime()
- AddTimeSlot()
- GetTimeSlots()
  - WithinDateRange()
  - WithinTimeRange()
  - ParentTaskOnly()
  - ChildrenOnly()

Data:

- List of UniqueTimeSlot objects - these should not be allowed to overlap