19th European Tcl/Tk Conference
July 20th - 21st 2023, Vienna, Austria

https://rattleCAD.com/
... design your custom Bike

Manfred ROSENBERGER
Implementation of the Observer Pattern with TclOO in the CAD software rattleCAD

rattleCAD 4.0

- [https://rattlecad.com/](https://rattlecad.com/)

Context:

- User Interaction - GUI
  - manage various views
- Software Architecture
  - accelerate startup-process
  - independent packages
  - testability
  - improve quality!
    - start as small project
    - uncoordinated growth
    - clearup the mess!
rattleCAD - Vintage Versions
rattleCAD – History: 1.4.00 (2007)

Files: 46
Folder: 5
Size: 1,07 MB
rattleCAD – History: 2.8.03 (2009)

Files: 126
Folder: 12
Size: 1,16 MB
rattleCAD – History: 3.4.02 (2016)

Files: 453
Folder: 70
Size: 15 MB
rattleCAD 4.0: 4.3.00 (2023, comming soon)

Files: 1614
Folder: 189
Size: 14.6 MB

component files in an external library: 7.5 MB
Challenges of current Improvements
rattleCAD 4.0: 4.3.00

- synchronize views
- start-up process
rattleCAD 4.0: Challenge

Synchronize Views
- every view shows the same state

Accelerate the Start-up Process
- complex package dependencies
- fully loaded model required
- multiple view updates
Approach
**Approach**

- divide and conquer
  - independent packages
- separate data, GUI and logic
- Concepts & Strategies
  - MVC Pattern
  - Observer Pattern

**Refactor Code Base** (neverending story)

- improve Software architecture
- improve testability
  - test model independ from the GUI
  - initialize the GUI without loaded model
rattleCAD 4.0 - Packages
MVC - Pattern

- Actor
  - Consumer
    - User
    - Software component

- View
  - Interface to actor
  - Update views

- Controller
  - Glue code

- Model
  - Data
  - Domain logic

... where Tk comes in

Tcl only
... definitely no Tk
The Observer Pattern (Gang of Four)

Definition

• 1-to-n dependency between objects.
• A change in the state of an observed object causes all dependent objects to be notified and automatically updated. ([GoF], page 287)
• Objects (observer, observing object) can be registered by another object (subject, observed object) and henceforth be informed by the latter as soon as it changes.
• This registration can be cancelled.

https://www.philipphauer.de/study/se/design-pattern/observer.php#pdf
Manage View Updates with the Observer-Pattern
rattleCAD 4.0 - Views

Context:
- XZ
- XY
- Edit
- TubeMiter
Observer Pattern in the context of the rattleCAD project
Observer Pattern in the context of the rattleCAD project - Subject

**Subject:**

```tcl
oo::class create Subject {
    variable ObserverRegistry
    variable DictContext

    method subscribe {observer} {
        lappend ObserverRegistry $observer
    }
    # method unsubscribe {observer} {
    #    set idx [lsearch $ObserverRegistry $observer]
    #    set ObserverRegistry [lreplace $ObserverRegistry $idx $idx]
    #}
    method notify {{state {}}} {
        set StateNotification $state
        foreach subscriber $ObserverRegistry {
            $subscriber updateState [self]
        }
    }
    method setDictContext {dictContext} {
        set DictContext $dictContext
    }
    # method getContext {{key {}}} {
    #    if [dict exists $DictContext $key] {
    #        return [dict get $DictContext $key]
    #    }
    #}
}
```

![Diagram](image-url)
Observer Pattern in the context of the rattleCAD project - Concrete Subject

Concrete Subject:

```tcl
proc myGUI::modelBike::init () {
    # variable Object_BikeModel [IF_BikeModel new]
    variable State_BikeModel [State_BikeModel new]
    # variable Subject_ProjectInfo [Subject new]
    variable Subject_BikeModel [Subject new]
    variable Subject_BikeModel_Edit [Subject new]
    #
    myGUI::modelBike::model_Edit::setListBoxDict \$
       {[Object_BikeModel get_ListBoxValues]}
    
    $Subject_BikeModel setDictContext \$
       {xy ::myGUI::modelBike::model_XY
        xz ::myGUI::modelBike::model_XZ
        tubeMiter ::myGUI::modelBike::model_TubeMiter}
    
    $Subject_BikeModel_Edit setDictContext \$
       {edit ::myGUI::modelBike::model_Edit}
}
```
Observer Pattern in the context of the rattleCAD project - Observer

Observer:

```tcl
oo::class create Observer {

    method updateState {subject} {
        puts ""
        puts "-> [info object class [self]] -> updateState:" 
        puts " -> subject: $subject"
        set data [$subject getData]
        puts " -> data: $data"
        puts ""
        puts "... to be overwritten by custom observer class"
        puts ""
    }
}
```
Observer Pattern in the context of the rattleCAD project - Concrete Observer

Concrete Observer:

```tcl
oo::class create myGUI::viewDrawing::ObserverDrawingView {
}

oo::class create myGUI::viewEdit::ObserverEditView {
    superclass Observer
    method createState {subject args} {
        set contextSubject_Edit [subject getContext edit]
        set myGUI::viewEdit::SubjectModel_Edit $contextSubject_Edit
        myGUI::viewEdit::updateCurrentValues
    }
}
```
rattleCAD Startup Routine

Switch on the Update-Process late

```tcl
myGUI::main {rootDir baseDir {startupProject {}}} {
    # -- GUI

    myGUI:init $rootDir $baseDir
    myGUI:create_intro .intro
    myGUI:modelSession::update_windowTitle
    myGUI:init_GUI
    myGUI: update_GUI_Bindings
    myGUI:gui::set_AccordionMainTab frametubes
    myGUI:gui::show_CanvasLogo

    # -- Model

    myGUI:modelBike::init
    myGUI:modelBike::update_UserCompDirectories
    myGUI:modelBike::update_UserTemplateDirectory
    myGUI:modelBike::add_ComponentDir user ...
    myGUI:modelBike::add_ComponentLib rcLib ...

    # -- Model - Binding

    set observerDrawingView [myGUI::viewDrawing::ObserverDrawingView new
    set observerEditView [myGUI::viewEdit::ObserverEditView new]

    $::myGUI::modelBike::Subject_BikeModel subscribe $observerDrawingView
    $::myGUI::modelBike::Subject_BikeModel_Edit subscribe $observerEditView

    }```
View-Widgets in myGUI::viewEdit

treeview

canvas / tkpath

combobox

spinbox
Types of View-Widgets View Widgets in myGUI::viewEdit and update

Views to manipulate the model
Implementation in Detail: Context – viewEdit (manipulate the model)
Implementation in Detail: Context – Drawing XY, XZ (visualize data only)

Drawing views do not know about the MVC-Controller
Review & Summary
Results

Synchronize Views
• working

Accelerate the Start-up Process
• stable and working

Software Architecture
• documentation & overview
  • clearer structure
  • Independent packages

Improved testability
• modular testing
  • e.g: run the GUI without the domain model
Review: The Observer Pattern

Where do I get the Observer-Class libraries?

• Download:
  • https://github.com/level44/design-patterns-in-tcl
• Create yourself:
  • https://sourcemaking.com/design_patterns
rattleCAD 4.0: 4.3.00

coming soon!
Thanks for all of the support from the Tcl/Tk community!