

Writing A New Widget Class Using C and Tk

John Ousterhout

Computer Science Division
Department of EECS

University of California at Berkeley

Outline

- What does Tk do for widget writers?
- Widget basics: data structures, etc.
- Six procedures to write:
 - Create
 - Configure
 - Display
 - Widget command
 - Event handler
 - Destroy
- Example: trivial “square” widget.

What Does Tk Do For You?

1. **Window names:** `.a.b.c`
2. **Caching (efficiency, convenience):**
 - Window information (size, parent, etc.).
 - X resources (colors, fonts, GCs, etc.).
3. **Protocol intermediary:**
 - Event dispatching.
 - Geometry management.
 - Selection protocols.
 - Keyboard focus.
 - Window manager.
 - Error handling.

Tk C Interfaces, slide 3.

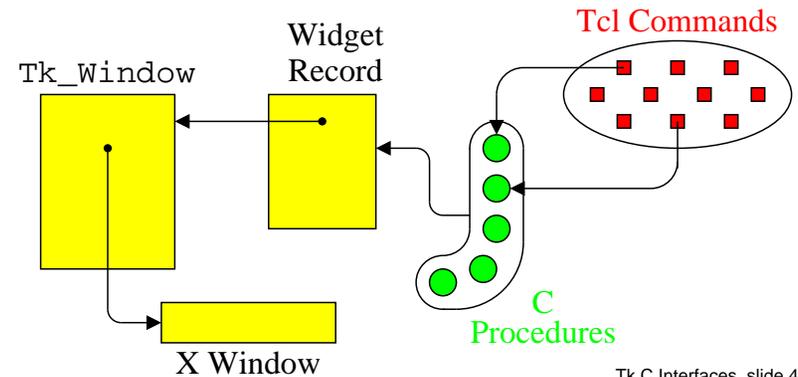
The Structure of a Widget

Data structures for each widget:

- `Tk_Window` managed by Tk.
- Widget record managed by widget code.

C code for widget class:

- Tcl **class command** to create widgets.
- Tcl **widget command** to manipulate widgets.
- Supporting C procedures (e.g. redisplay).



Tk C Interfaces, slide 4.

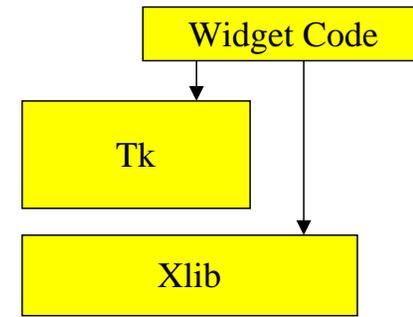
Philosophy: Widgets Are Reactive

- User is in control, not Tk or widget or application.
- Widget code is **event-driven**: responds to events around it.
- Procedures in widget are invoked when events occur, e.g.:
 - Tcl command invoked.
 - Window needs to be redrawn.
 - Window destroyed.

Tk C Interfaces, slide 5.

Tk and Xlib

When should widget call Tk, when Xlib?



Call Xlib only to draw on screen.

Call Tk for everything else:

- Creating windows.
- Manipulating windows (map, resize, etc.).
- Managing events.
- Allocating resources (colors, GCs, etc.).

Tk C Interfaces, slide 6.

Example: Square Widget

- Displays colored square on background.
- Widget command allows square to be moved, resized:

```
.s position 20 30  
.s size 10
```
- Can write fancier behaviors in Tcl:
 - Drag square with mouse.
 - Animate.

Creating a Widget

- Procedure **Tk_SquareCmd**.
- Invoked with Tcl command named after class:

```
square .s -fg RoyalBlue1
```
- Class command registered in main program:

```
Tcl_CreateCommand(interp, "square",  
Tk_SquareCmd, ...);
```
- Create **Tk_Window** object (variable **tkwin** holds handle).
- Initialize widget record (**squarePtr**), set up callbacks.
- Register widget command.
- Configure widget using command-line arguments.
- Don't map window: geometry manager will do it.

Delayed Window Creation

- X window isn't created immediately by **Tk_CreateWindowFromPath**.
- **Tk_WindowId(tkwin)** returns **None**.
- Window creation occurs when window mapped by **Tk_MapWindow**.
- Delay saves overhead (can reconfigure without involving X server).
- Can force creation of X window with **Tk_MakeWindowExist**.

Configure Procedure

- Procedure **ConfigureSquare**.
- Processes **argc/argv**, modifies widget record, schedules widget redisplay.
- Called from both class command and widget command.
- Almost all of work done by Tk library procedure **Tk_ConfigureWidget**.
- Class provides table of configuration options: **configSpecs**.
- **ConfigureSquare** must also call **Tk_GeometryRequest** to set desired size.

Display Procedure

- Procedure `DisplaySquare`.
- Redisplay is delayed:
 - Don't redisplay immediately (could result in multiple redisplays).
 - Instead, record what must be redrawn (for simple widgets, all or nothing).
 - Do actual redisplay when all pending work is finished: use `Tk_DoWhenIdle()`.
- Tk provides support for 3-D effects:
 - `Tk_3DBorder` data type.
 - `Tk_Fill3DRectangle()`, etc.
- Double-buffering with pixmap to avoid flashing.

Widget Command Procedure

- Procedure `SquareWidgetCmd`.
- Decodes `argv[1]`, executes one of several commands.
- Potentially modifies widget record.
- Arranges for redisplay if necessary.

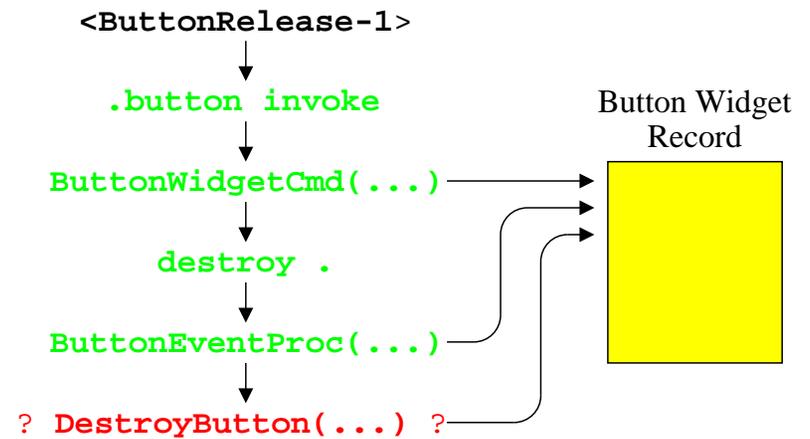
Only provides primitive operations; complex features are implemented with Tcl scripts.

Event-Handling Procedure

- Procedure `SquareEventProc`.
- Most events (e.g. `ButtonPress`) handled with Tcl bindings (more flexible).
- Most widgets need C code only for `Expose`, `DestroyNotify`, maybe `ConfigureNotify`.
- Handler set up during widget creation by calling `Tk_CreateEventHandler`.

Destroy Procedure

- Procedure `DestroySquare`.
- **WARNING!** Can't always clean up immediately: widget record may be in use by nested procedure.



- Solution: must delay destruction.

Delayed Window Destruction

- Tk implements short-term reference counts.
E.g. in `ButtonWidgetCmd`:

```
Tk_Preserve((ClientData) butPtr);  
...  
Tcl_Eval(interp, butPtr->cmd, ...);  
...  
Tk_Release((ClientData) butPtr);  
return result;
```
- Don't call destruction procedure directly:
E.g. in `ButtonEventProc`:

```
if (eventPtr->type == DestroyNotify) {  
    ...  
    Tk_EventuallyFree((ClientData)  
        butPtr, DestroyButton);  
}
```
- `DestroyButton(butPtr)` is invoked:
 - Immediately if (no `Tk_Preserve`'s pending).
 - During last `Tk_Release` call.

Tk C Interfaces, slide 15.

Things To Remember

- Event-driven style of programming.
- Delayed operations:
 - Creation of X window.
 - Redisplay.
 - Destruction of widget record.
- Work within Tcl framework: focus on primitives.
- Use Tk caches, configuration support.

Don't build from scratch: modify an existing widget.

Tk C Interfaces, slide 16.