Embedding an SDL Surface in a Tk Window

by Lawrence Woodman 26 June 2012
Tags: Programming SDL Tcl/Tk Tutorial

Tk is great, but sometimes it just isn't fast enough. SDL is fast, but has no support for input dialogs and other GUI conventions. By embedding an SDL surface in a Tk window you get the best of both worlds. Whether you want to use Tk to add a nice GUI to an SDL app or want to access SDL via Tcl/Tk, this article will show you how.

Embedding the SDL Surface

To embed an SDL surface in another window you have to alter the `SDL_WINDOWID` environmental variable so that it matches the ID of the window that you want the SDL surface embedded in. This must be done after the main window is displayed and before SDL_Init() is called.

To ensure that the Tk window is displayed you need to call something like:

```
Tcl_Eval(interp, "update");
```

Then to get the window ID and set `SDL_WINDOWID`:

```
int
setSDLWindowID(Tcl_Interp *interp)
{
  Tcl_Obj *result;
  char *windowID;
```

```
  char envBuf[50];

  /* .screen here is the name of the widget that you want to overwrite,
     this is normally a frame */
  if (Tcl_Eval(interp, "winfo id .screen") == TCL_ERROR)
    return 0;

  result = Tcl_GetObjResult(interp);
  Tcl_GetStringFromObj(windowID, NULL);

  snprintf(envBuf, 50, "SDL_WINDOWID=%s", windowID);
  SDL_putenv(buf);

  return 1;
}
```

The above uses `SDL_putenv()` rather than `putenv()` as this is recommended by an old SDL GUI FAQ

From this point you can call `SDL_Init()` and `SDL_SetVideoMode()`, but do remember to use the `SDL_NOFRAME` attribute:

```
SDL_Surface *
initScreenSurface(int width, int height, int depth)
{
  SDL_Surface *sfScreen;

  if (SDL_Init(SDL_INIT_EVERYTHING) < 0) {
    fprintf(stderr, "Couldn't initialize SDL: %sWn", SDL_GetError());
    return NULL;
  }

  sfScreen = SDL_SetVideoMode(width, height, depth,
                              SDL_NOFRAME|SDL_SWSURFACE|SDL_ANYFORMAT);
  if (sfScreen == NULL) {
    fprintf(stderr, "Couldn't initialize SDL: %sWn", SDL_GetError());
    return NULL;
  }

  return sfSreen;
}
```

The Event Loop

You must have an event loop that calls both SDL_PollEvent() and Tk_DoOneEvent(). The events will be handled mostly by Tk. However, you do need to detect SDL_QUIT from SDL_PollEvent() because SDL converts SIGTERM to this.

```
void
event_loop()
{
  SDL_Event event;

  while (!(SDL_PollEvent(&event) && event.type == SDL_QUIT)) {
    Tk_DoOneEvent(TK_ALL_EVENTS|TK_DONT_WAIT);
  }
}
```

Handling Key Release Events

From Tcl you can handle whichever events you need to detect. For example to bind the

<KeyRelease> event to a key handler:

```
proc handleKey {key} {
  switch -regexp -- $key {
    .*Up$ {ball up}
    .*Down$ {ball down}
    .*Left$ {ball left}
    .*Right$ {ball right}
  }
}
bind all <KeyRelease> {handleKey %K}
```

Handling Focus Events

SDL also needs to know when the screen is put into focus. From Tcl:

```
bind . <FocusIn> {screen_refresh}
```

And to provide the screen_refresh command:

```
static SDL_Surface *sfScreen = NULL;

void
screenRefresh(void)
{
  if (sfScreen != NULL)
    SDL_Flip(sfScreen);
}

static int
ScreenRefreshCmd(ClientData clientData, Tcl_Interp *interp,
                 int objc, Tcl_Obj *CONST objv[])
{
  if (objc != 1) {
    Tcl_WrongNumArgs(interp, 1, objv, "");
  }

  screenRefresh();
  return TCL_OK;
}

void createCommands(Tcl_Interp *interp)
{
  Tcl_CreateObjCommand(interp, "screen_refresh", ScreenRefreshCmd,
                       (ClientData) NULL,
                       (Tcl_CmdDeleteProc *) NULL);
}
```

A Small Demonstration

I have created the sdl_and_tk_demo on github to demonstrate how to put this altogether. The README contains information on how to compile and run the demo. This demo was inspired by Kent Mein's SDL and Tk MDI demo.

Related

- Running 4K FORTRAN on a DEC PDP-8
- Installing the HI-TECH Z80 C Compiler for CP/M
- Installing ZDE 1.6, a programmers editor for CP/M
- Getting Colour ANSI Emulation to Work Properly When Connecting to a BBS With Telnet Under Linux
- Setting up a Beowulf Cluster Using Open MPI on Linux