Tcl/Tk on Android

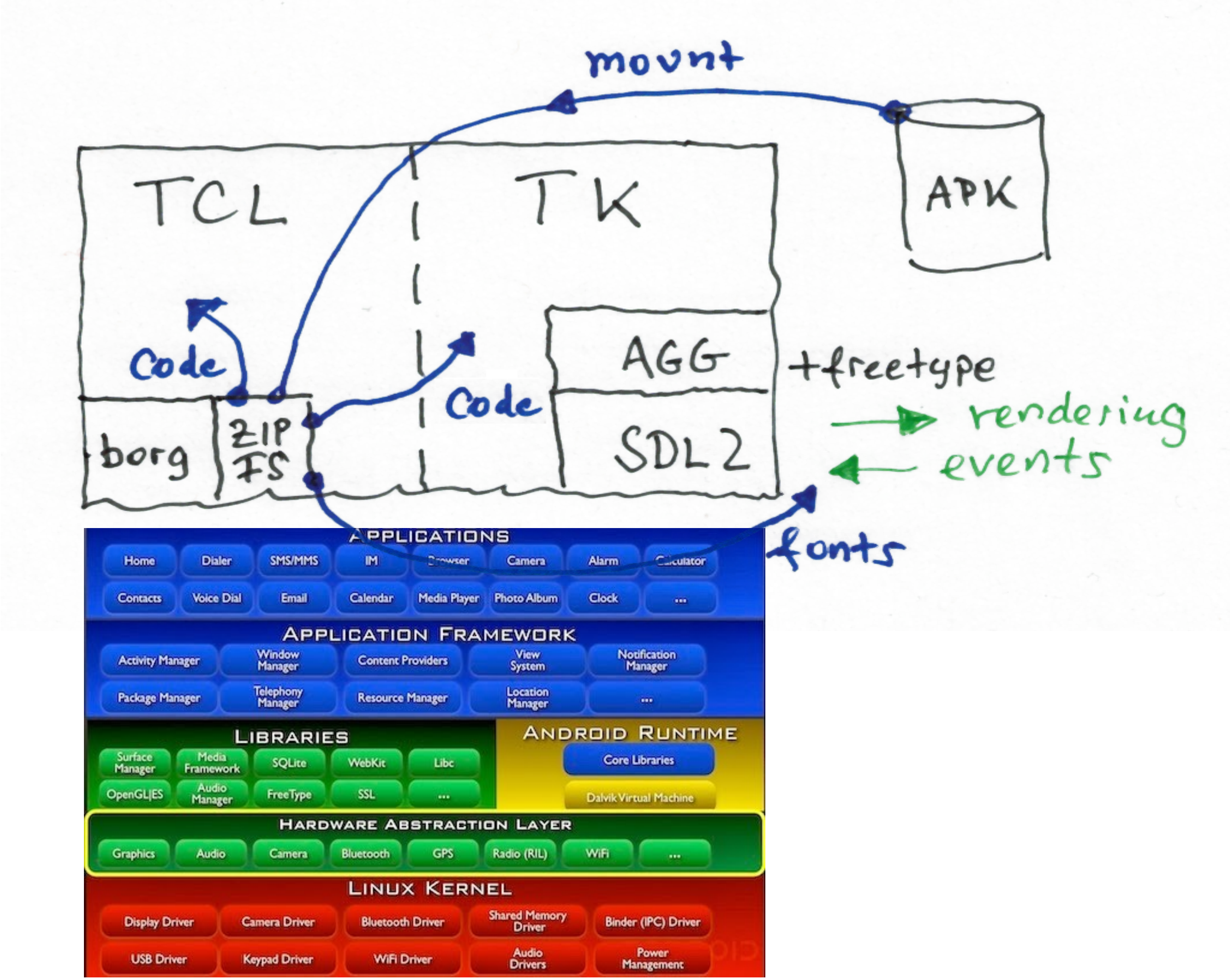
http://www.androwish.org

Icon kindly donated by Jorge Raul Moreno
Domain and web space kindly donated by D. Richard Hipp

AndroWish

- Native Tcl/Tk 8.6 port for Android ≥ 2.3.3 on ARM and x86 processors. Top goal: execute existing Tcl/Tk scripts on Android without modification.
- Uses Tim Baker's SDLTK project for graphics rendering, see http://members.shaw.ca/tnbaker/SDLTk
  - X11 emulation based on AGG (Anti-Grain-Geometry, http://www.antigrain.com) and SDL 2.0 (http://libsdl.org)
  - freetype font engine (http://www.freetype.org)
- Mounts its constituting APK (Android Package, i.e. the app) using a built-in ZIP file system based on mmap(2)
- “Batteries included” like TclKits (i.e. many ready-to-use Tcl extensions already bundled)
- Tcl commands available to use specific Android facilities

AndroWish, the big picture



AndroWish, limitations

- Due to Android process start up with respect to the window system **`exec(n)`** is limited to non-Tk processes, i.e. you can't **`exec wish`**.
- The bandwidth of device screen resolutions is broad (140 dpi ... 500 dpi) compared to usual desktop systems. But many elements of Tk widgets are pixel based.

SDLTK

- Partial Xlib replacement to implement rendering of standard Tk widgets using SDL and AGG and to support most features of extensions (treeview, tktable, TIX, BLT).
- Event handling by translating SDL events to X11 events plus additional virtual events
  - App life cycle (`<<WillEnterBackground>>`, `<<WillEnterForeground>>`, etc.)
  - Accelerometer is handled by SDL and mapped to an SDL joystick, and translated to Tk virtual event `<<Accelerometer>>`
  - Multi-touch events `<<FingerUp>>`, `<<FingerDown>>`, `<<FingerMotion>>`, `<<PinchToZoom>>`
- Other SDL goodies available through `sdltk` command
- Can be built standalone on other platforms (currently tested on Linux x86/Raspberry with X11 and DirectFB), see http://www.ch-werner.de/sdltk

`sdltk` command

- Obtain power management information (`sdltk powerinfo`)
- Control accelerometer (`sdltk accelerometer`)
- Control screensaver (`sdltk screensaver`)
- Show/hide virtual keyboard (`sdltk textinput`)
  - built into standard bindings of entry and text widgets
- Control emulation of middle and right mouse buttons for context menus and panning/scrolling (`sdltk touchtranslate`)

borg [ˈbɔʁk]

- German imperative of “borgen” (to borrow)
- Inspired by Scripting Layer for Android (SL4A) and PhoneGap (now Apache Cordova)
- Junction to connect the (native) Tcl/Tk with the (Java) Android universe
- Use Android facilities from Tcl scripts
  - Information about device
  - Activities (Android apps)
  - Alarms, Notifications
  - Content (Android databases)
  - Location (GPS)
  - Text to speech and speech recognition
- Report device events using Tk virtual events (network status, location updates, sensor events)

borg command examples

Obtain information about the display

```
borg displaymetrics
density 1.33125 densitydpi 213 \
width 800 height 1216 \
xdpi 216.17021 ydpi 215.31126 \
scaleddensity 1.33125
```

borg command examples

Use Android's text-to-speech facility for speech output

```
borg speak “resistance is futile” \
en_US
```

borg command examples

Add an icon to Android's desktop, which launches a Tcl script contained in AndroWish's ZIP mounted `/assets` directory

```
 borg shortcut add \
 file://assets/sdl2tk8.6/demos/widget
```

borg command examples

Use other Android activities (parts of other apps). Example: read a bar code using the http://code.google.com/p/zxing bar code scanner app.

```
proc barcode_read {code action uri type cat data} {
    array set result $data
    if {[info exists result(SCAN_RESULT)]} {
        # that is the barcode
        # result(SCAN_RESULT_FORMAT) is the barcode format
    }
}

borg activity com.google.zxing.client.android.SCAN \
    {} {} {} {} {} barcode_read
```

`borg` virtual events

- `<<LocationUpdate>>`: location information is available and can be read using

```
borg location get
```

- `<<SensorUpdate>>`: a sensor can be read using

```
borg sensor get ...
```

- `<<NetworkInfo>>`: network status was updated and can be read using

```
borg networkinfo
```

- `<<Bluetooth>>`: Bluetooth status was updated and can be read using

```
borg bluetooth state
```

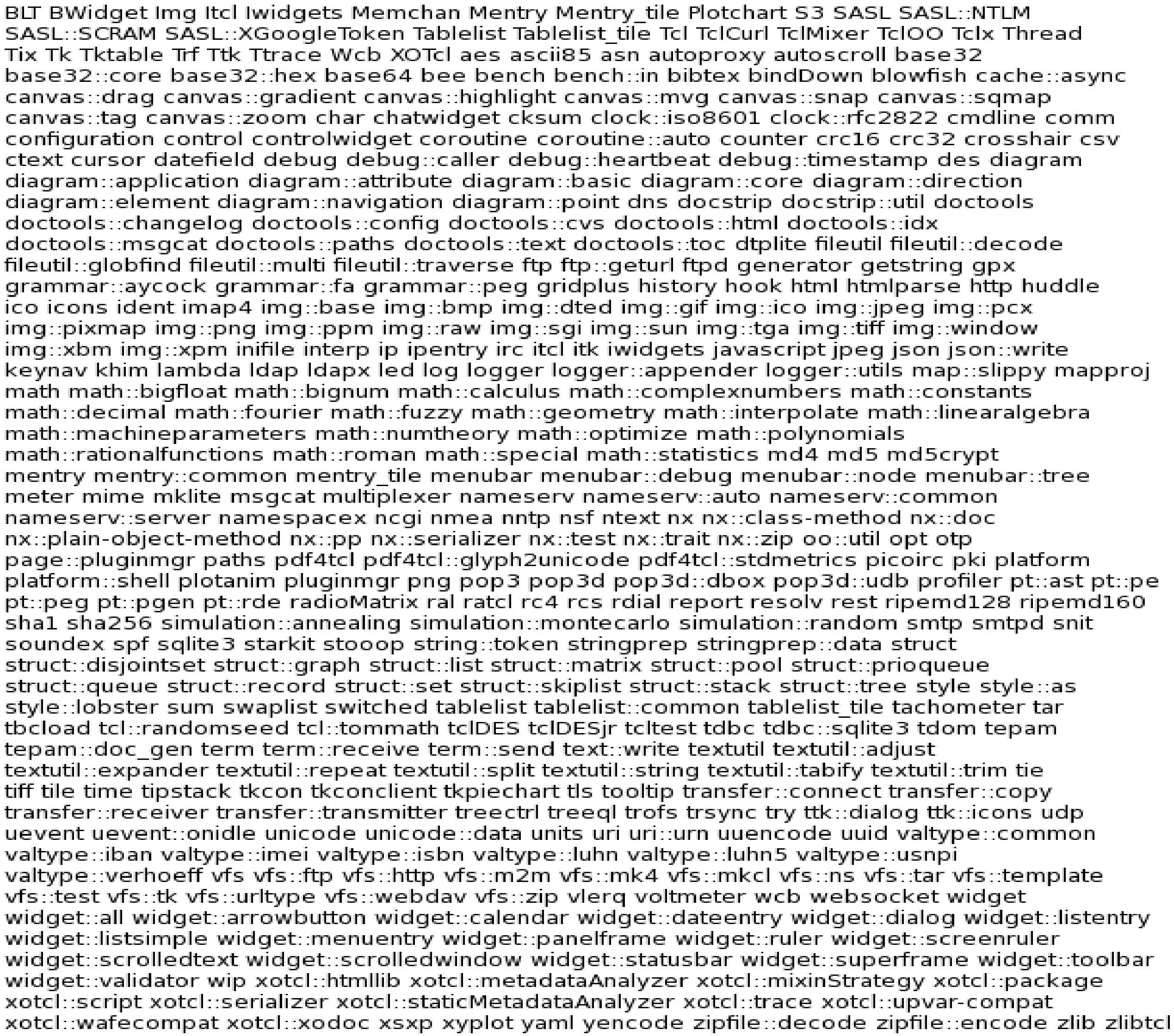
Other AndroWish goodies

- `bluetooth` command allows to create Bluetooth client and server sockets providing the serial port profile (SPP). These sockets are normal Tcl channels.
- `usbserial` command allows to use certain USB to serial converters (FTDI, Prolific) similar to normal Tcl serial channels, i.e. `fconfigure` can be used to control the baud rate etc.

Anatomy of AndroWish's APK

Directory within APK	Description	Size uncompressed
assets/	Application auxiliary files, Tcl libraries	38 MByte
tcl8.6/		
sdltk8.6/		
fonts/	Default fonts, Deja Vu	
...	“Batteries included” (SQLite, tcllib, tls, treectrl, ...)	
app/		
lib/	Native code, shared libraries	
armeabi/	... for ARM processors	12 MByte
x86/	... for x86 processors	18 MByte
res/	Android Resources	few kByte
drawable/	... icons	
layout/	... layout, styles	
resources.asrc	Application resources	few kByte
AndroidManifest.xml	Application descriptor	few kByte
classes.dex	Compiled Java code for Android's Dalvik VM	< 1 MByte
META-INF/	JAR Manifest, signatures of all files in APK	< 1 MByte

Batteries included



```
BLT BWidget Img Itcl Iwidgets Memchan Mentry Mentry_tile Plotchart S3 SASL SASL::NTLM
SASL::SCRAM SASL::XGoogleToken Tablelist Tablelist_tile Tcl TclCurl TclMixer TclOO Tclx Thread
Tix Tk Tktable Trf Ttk Ttrace Wcb XOTcl aes ascii85 asn autoproxy autoscroll base32
base32::core base32::hex base64 bee bench bench::in bibtex bindDown blowfish cache::async
canvas::drag canvas::gradient canvas::highlight canvas::mvg canvas::snap canvas::sqmap
canvas::tag canvas::zoom char chatwidget cksum clock::iso8601 clock::rfc2822 cmdline comm
configuration control controlwidget coroutine coroutine::auto counter crc16 crc32 crosshair csv
ctext cursor datefield debug debug::caller debug::heartbeat debug::timestamp des diagram
diagram::application diagram::attribute diagram::basic diagram::core diagram::direction
diagram::element diagram::navigation diagram::point dns docstrip docstrip::util doctools
doctools::changelog doctools::config doctools::cvs doctools::html doctools::idx
doctools::msgcat doctools::paths doctools::text doctools::toc dtplite fileutil fileutil::decode
fileutil::globfind fileutil::multi fileutil::traverse ftp ftp::geturl ftpd generator getstring gpx
grammar::aycock grammar::fa grammar::peg gridplus history hook html htmlparse http huddle
ico icons ident imap4 img::base img::bmp img::dted img::gif img::ico img::jpeg img::pcx
img::pixmap img::png img::ppm img::raw img::sgi img::sun img::tga img::tiff img::window
img::xbm img::xpm inifile interp ip ipentry irc itcl itk iwidgets javascript jpeg json json::write
keynav khim lambda ldap ldapx led log logger logger::appender logger::utils map::slippy mapproj
math math::bigfloat math::bignum math::calculus math::complexnumbers math::constants
math::decimal math::fourier math::fuzzy math::geometry math::interpolate math::linearalgebra
math::machineparameters math::numtheory math::optimize math::polynomials
math::rationalfunctions math::roman math::special math::statistics md4 md5 md5crypt
mentry mentry::common mentry_tile menubar menubar::debug menubar::node menubar::tree
meter mime mklite msgcat multiplexer nameserv nameserv::auto nameserv::common
nameserv::server namespacex ncgi nmea nntp nsf ntext nx nx::class-method nx::doc
nx::plain-object-method nx::pp nx::serializer nx::test nx::trait nx::zip oo::util opt otp
page::pluginmgr paths pdf4tcl pdf4tcl::glyph2unicode pdf4tcl::stdmetrics picoirc pki platform
platform::shell plotanim pluginmgr png pop3 pop3d pop3d::dbox pop3d::udb profiler pt::ast pt::pe
pt::peg pt::pgen pt::rde radioMatrix ral ratcl rc4 rcs rdial report resolv rest ripemd128 ripemd160
sha1 sha256 simulation::annealing simulation::montecarlo simulation::random smtp smtpd snit
soundex spf sqlite3 starkit stooop string::token stringprep stringprep::data struct
struct::disjointset struct::graph struct::list struct::matrix struct::pool struct::prioqueue
struct::queue struct::record struct::set struct::skiplist struct::stack struct::tree style style::as
style::lobster sum swaplist switched tablelist tablelist::common tablelist_tile tachometer tar
tbcload tcl::randomseed tcl::tommath tclDES tclDESjr tcltest tdbc tdbc::sqlite3 tdom tepam
tepam::doc_gen term term::receive term::send text::write textutil textutil::adjust
textutil::expander textutil::repeat textutil::split textutil::string textutil::tabify textutil::trim tie
tiff tile time tipstack tkcon tkconclient tkpiechart tls tooltip transfer::connect transfer::copy
transfer::receiver transfer::transmitter treectrl treeql trofs trsync try ttk::dialog ttk::icons udp
uevent uevent::onidle unicode unicode::data units uri uri::urn uuencode uuid valtype::common
valtype::iban valtype::imei valtype::isbn valtype::luhn valtype::luhn5 valtype::usnpi
valtype::verhoeff vfs vfs::ftp vfs::http vfs::m2m vfs::mk4 vfs::mkcl vfs::ns vfs::tar vfs::template
vfs::test vfs::tk vfs::urltype vfs::webdav vfs::zip vlerq voltmeter wcb websocket widget
widget::all widget::arrowbutton widget::calendar widget::dateentry widget::dialog widget::listentry
widget::listsimple widget::menuentry widget::panelframe widget::ruler widget::screenruler
widget::scrolledtext widget::scrolledwindow widget::statusbar widget::superframe widget::toolbar
widget::validator wip xotcl::htmllib xotcl::metadataAnalyzer xotcl::mixinStrategy xotcl::package
xotcl::script xotcl::serializer xotcl::staticMetadataAnalyzer xotcl::trace xotcl::upvar-compat
xotcl::wafecompat xotcl::xodoc xsxp xyplot yaml yencode zipfile::decode zipfile::encode zlib zlibtcl
```

How to roll your own app

- It is possible to re-use AndroWish's infrastructure (`/assets` directory with Tcl runtime, native shared libraries) from other apps.
- A slightly modified Java glue is required (about 300 kByte compressed Java classes, compared to ≥ 24 MByte of the complete AndroWish APK).
- Due to APK building (various Android tools and Java `jarsigner` needed) this must be done using Android's SDK and optionally Eclipse.
- A demo project using this approach is in the source tree of AndroWish (subdirectory `hellotcltk`)

Dive into the hive: Preparations

0. AndroWish's APK is installed and launched.

1. The static constructor of AndroWish's activity loads various shared libraries including `System.loadLibrary(“main”);`

2. The `JNI_OnLoad()` function of `libmain.so` remembers a pointer to the JVM among other initialization steps.

3. The constructor of AndroWish's activity calls a static native method in `libmain.so`, which keeps a Java object reference on the activity for later. The activity object is the main entry point to carry out Android-specific functions.

Dive into the hive: let the `borg` bark

1. Tcl evaluates `borg beep`.
2. The native code in `libmain.so` invokes a static method on the activity object obtained during startup:

```
JNIEnv *env = GetJNIEnv();
jmethodID mid = (*env)->GetStaticMethodID(env, jactivity, "beep", "()V");
(*env)->CallStaticVoidMethod(env, jactivity, mid);
```

3. In the activity class the JVM executes this static method:

```
public static void beep() {
  Runnable beeper = new Runnable() {
    public void run() {
      Uri ringuri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
      Ringtone tone = RingtoneManager.getRingtone(mSingleton.getBaseContext(), ringuri);
      if ((tone != null) && (!tone.isPlaying())) {
        tone.play();
      }
    }
  };
  mSingleton.runOnUiThread(beeper);
}
```

4. The notification sound is played back (more or less barking due to microscopic speakers).

Dive into the hive: borg sensors

1. Tcl evaluates the command `borg sensor enable 42` (that is our fictional sense of life sensor).
2. The native code in `libmain.so` invokes a static method, which switches that sensor on.
3. The AndroWish activity includes a supplemental class, which implements a `SensorEventListener` interface. When new sensor data is available, this method is invoked:

   ```
   public void onSensorChanged(SensorEvent event) ...
   ```

   It stores sensor data in a synchronized field of the supplemental class. Finally this native method is called:

   ```
   mAndroWish.nativeTriggerSensor(42);
   ```

4. This carries out the following code in `libmain.so`:

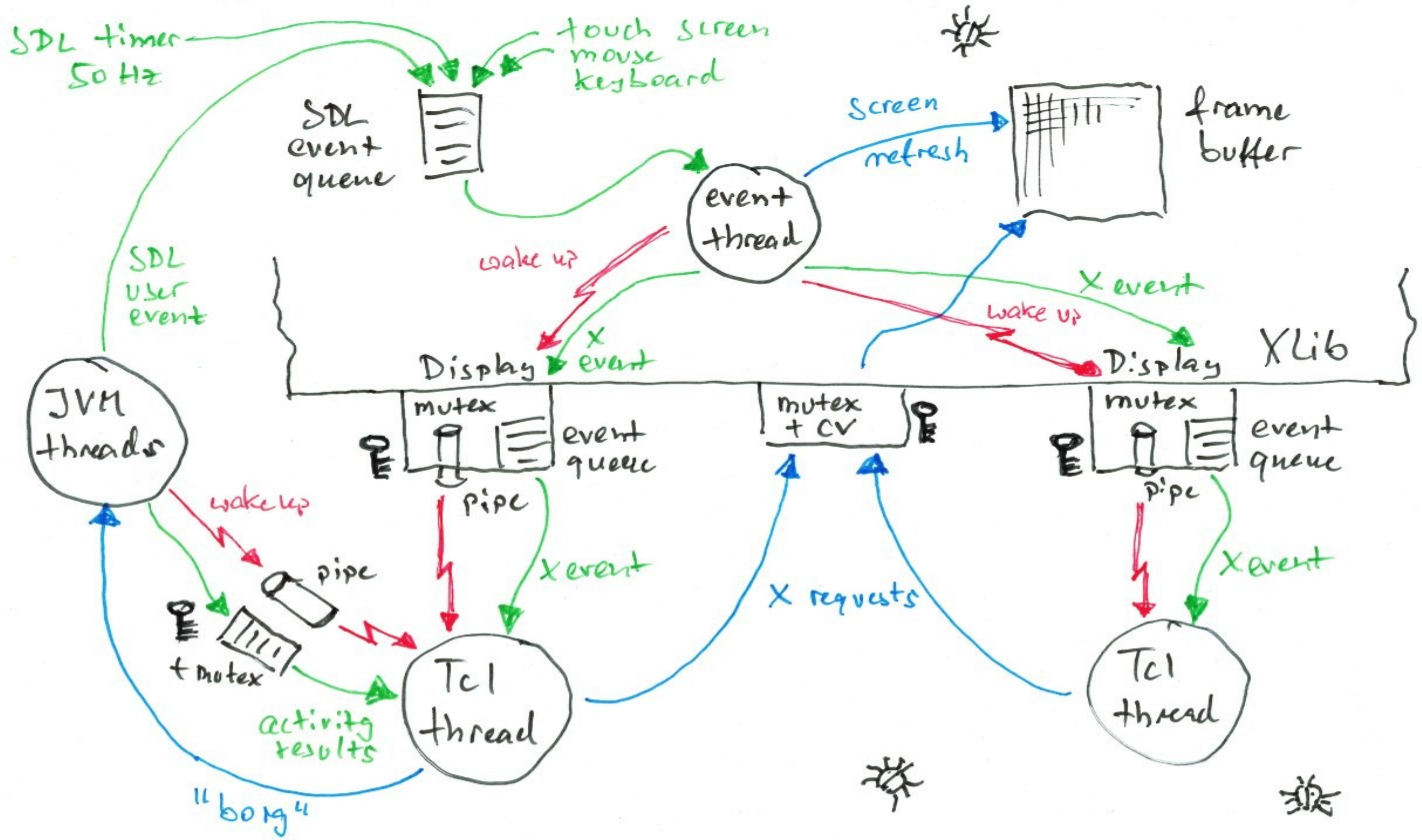
   ```
   SDL_Event event;
   event.type = SDL_USEREVENT;
   event.user.code = 42;
   event.user.data1 = (char *) “SensorUpdate”;
   if (SDL_PeepEvents(&event, 1, SDL_ADDEVENT, 0, 0) <= 0) return -1;
   return 1;
   ```

5. The SDL_USEREVENT is processed like other SDL events (touchscreen, mouse, keyboard) in the Xlib emulation layer and transformed into a virtual Tk event <<`SensorUpdate`>>, which is sent to all Tk toplevel windows.
6. Zero or more Tcl event handlers bound to <<`SensorUpdate`>> evaluate `borg sensor get 42` in order to read out the sensor data from the synchronized field in the supplemental class.

Dive into the hive: Activities

0. Activities come in two forms: one-shots without synchronization and a kind of remote procedure calls delivering results back to the caller. Execution is always asynchronous, i.e. the results are returned through callback functions. `borg activity` supports both modes.
1. `borg activity ... -callback ...` invokes native code in `libmain.so`, which translates the (many) parameters to their Java counterparts and invokes a static method in the AndroWish activity object which starts the activity.
2. This is remembered in a callback structure within `libmain.so` to match with the result later and to transport it back to Tcl.
3. The Android system carries out the activity and returns the result to the AndroWish activity object's `onActivityResult()` method.
4. In `onActivityResult()` the result is preformatted and presented to the native method `nativeIntentCallback()` of `libmain.so`.
5. In `nativeIntentCallback()` the matching callback structure is looked up, the preformatted results are translated to be appended to the `-callback` argument within the callback structure. That structure is finally queued to a thread-specific callback queue and a single byte is written into the write end of the pipe associated with the queue.
6. The file event handler on the read end of the pipe (running in the thread, which issued `borg activity` in step 1.) is invoked due to readability of the pipe. It empties the pipe and invokes all callbacks of the callback structures read out from the queue.
7. One-shot activities omit steps 2. to 6.
8. For speech recognition results the same mechanism (callback structure, queue, and pipe) is used.

X11 Emulation: The Puzzle



X11 Emulation: The Pieces

- SDL: device events, low-level timer, Android glue, frame buffer, low-level drawing surfaces, screen refresh functions.
- AGG: anti-aliased rendering (e.g. XDraw*() functions).
- freetype: low-level font support (loading TTF files, basic rendering).
- Event thread: transformation of SDL events to X events, screen updates from frame buffer (rate limited to 50 fps), event translation (pinch-to-zoom, middle/right mouse button emulation).
- Display: X11 display structure used to transfer X events to Tcl threads (one Tcl thread opens one Display like in Tk on X11).
- Big Xlib lock: Mutex plus condition variable to serialize X requests and to deal with server grabs like a real X server does.
- Pipes: OS level unidirectional communication channels to indicate queue not-empty state on display's X event queue similar to Tk on X11.

X11 Emulation: Claims & Reality

- If the behavior of the emulation is near real X11 not much additional code is needed in Tk.
- Porting existing Tk extensions becomes easy.
- Window background Pixmaps, clipping by Pixmaps, and rarely used GC functions are not implemented.
- Window manager functions are a bare minimum.
- Design inherently does not scale well on multiple cores.

20 minutes into the future

Tcl/Tk is quite easily portable and SDL already runs on many platforms. So let's think of ...

Windows ≥ XP
Windows RT
Linux ≥ 2.6 (X11 & DirectFB)
Mac OS X ≥ 10.5
iOS ≥ 5.1.1
Android ≥ 2.3.3
(Free|Open|Net)BSD
Solaris
Haiku
PSP
Mir
Wayland

(list of more or less supported platforms/window systems on SDL 2.0.3)

Thank you.

Questions?