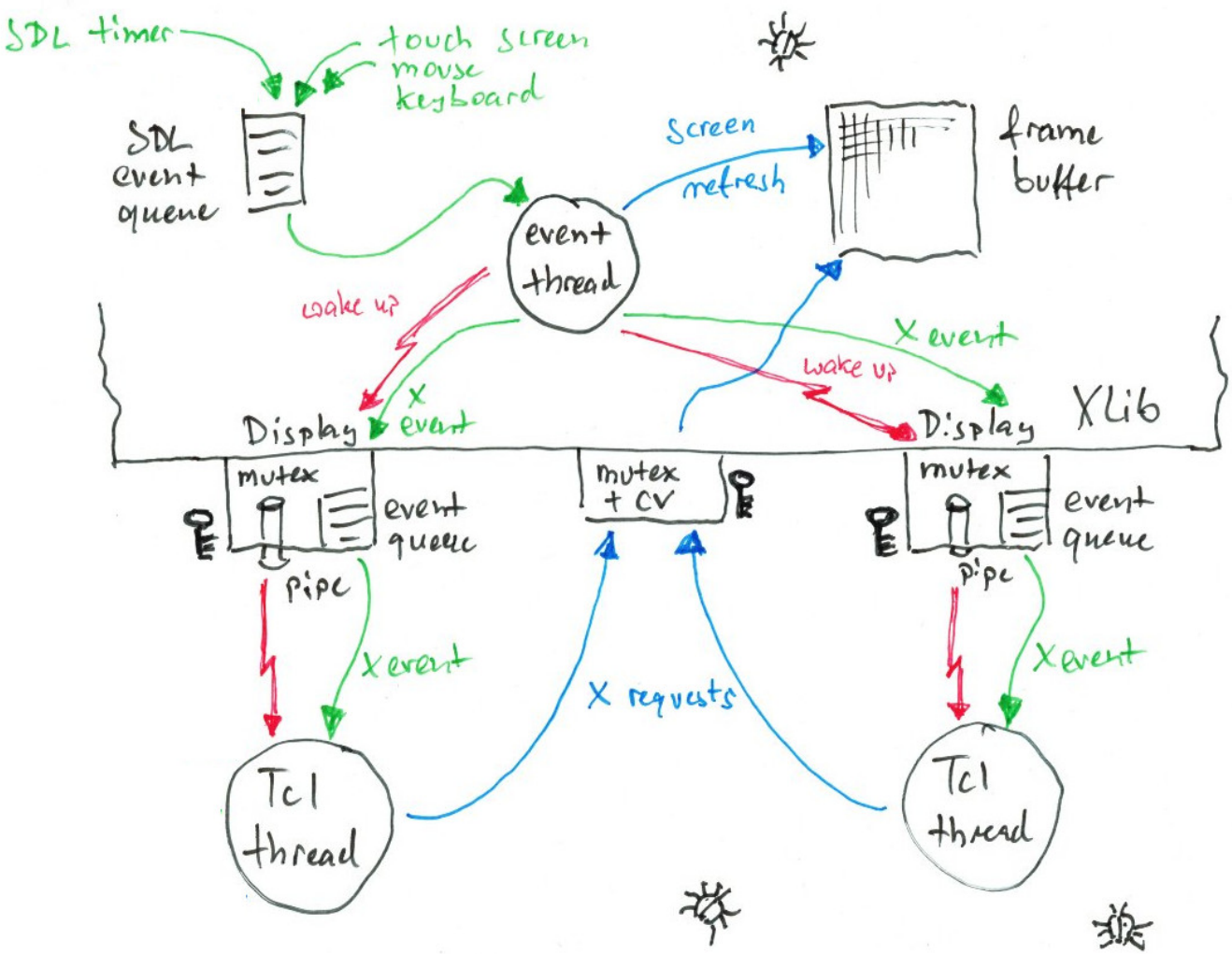
undroidwish on unusual platforms and other oddities

http://www.androwish.org

undroidwish

- Introduced at EuroTcl 2016
- AndroWish source tree plus few extras, minus Android specifics
- Uses same graphics/rendering libraries as AndroWish: SDL2, AGG, freetype
- Most SDL2 supported platforms are ported now: Linux, Windows (and ReactOS), MacOS, FreeBSD, OpenBSD, Illumos, and Haiku
- Wayland is now supported and tested on Debian 9, Fedora 28, and CentOS 7.5
- Experimental framebuffer mode (i.e. Tcl/Tk without display server/manager) is available for Raspberry and Linux 4.x (KMS, DRM)

undroidwish X11 emulation



undroidwish X11 emulation

- Multi-threaded Tk applications are supported.
- SDL2 supported input devices work OOTB (joysticks, touch screens).
- Many (non-trivial) Tk extensions are working (platform dependent regarding SDL2 video drivers and OpenGL support): Canvas3D, tkpath, tkimg, TkZinc, tktable, BLT, tktreectrl.
- A server less static Tcl/Tk binary (in Linux KMSDRM framebuffer mode) can be made in about 6 Mbyte (excluding required shared libraries and fonts).

Canvas3D stereoscopic rendering

Generic new feature available on all platforms the Canvas3D runs on (POSIX, Windows, MacOS, undroidwish).

Three new widget options:

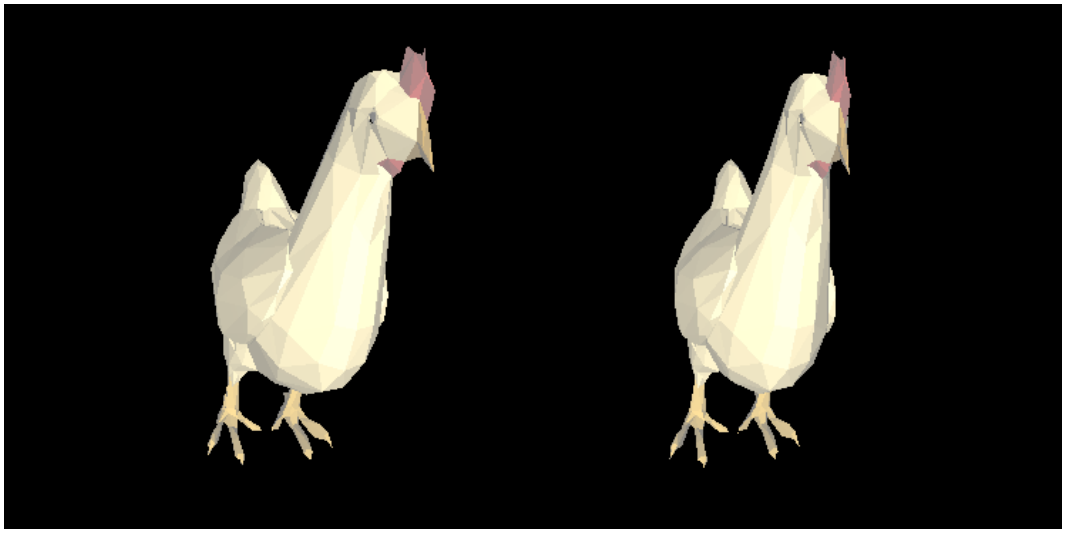
`-enablestereo`	boolean option to turn on stereoscopic rendering
`-eyedistance`	floating point value determining eye distance for left and right image
`-enableanaglyph`	boolean option to select between side-by-side images or a single image with anaglyph color filtering (red/cyan)

Canvas3D stereoscopic rendering

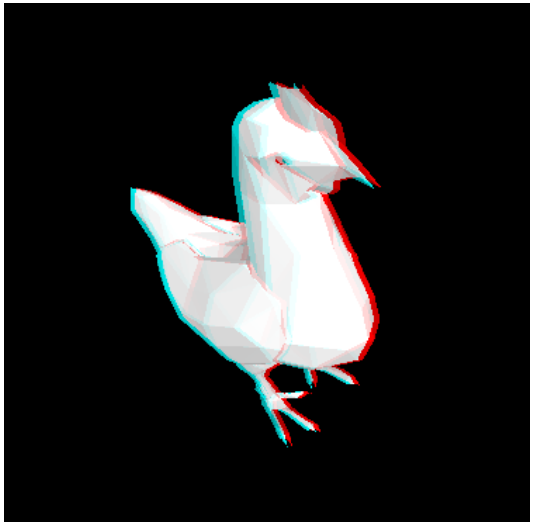
- If enabled, each image is rendered twice.
- In side-by-side mode, the viewport is switched between left and right halves of the widget’s drawing area.
- In anaglyph mode, a color mask (R for the left eye, GB for the right eye) is applied on each image and the combined RGB image is displayed in the widget’s drawing area.

Canvas3D stereoscopic rendering

Side-by-side mode



Anaglyph mode

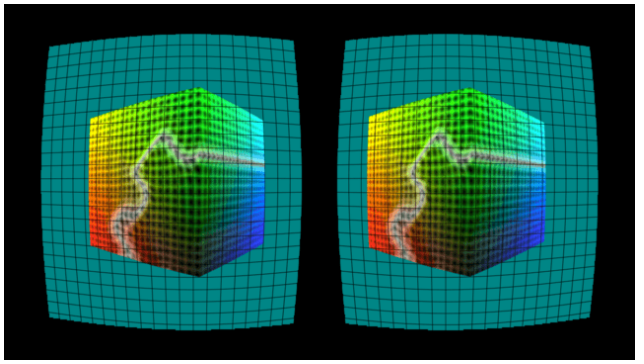


Stereoscopic rendering in AndroWish

When using a VR headset with a smartphone a lens correction is needed. This is achieved by an OpenGLES 2.0 shader in SDL's render driver and controlled by an `sdltk` minor command:

```
sdltk vrmode ?mode ?distortion rescale??
```

The parameter `mode` selects render mode, `distortion` and `rescale` control the lens correction.



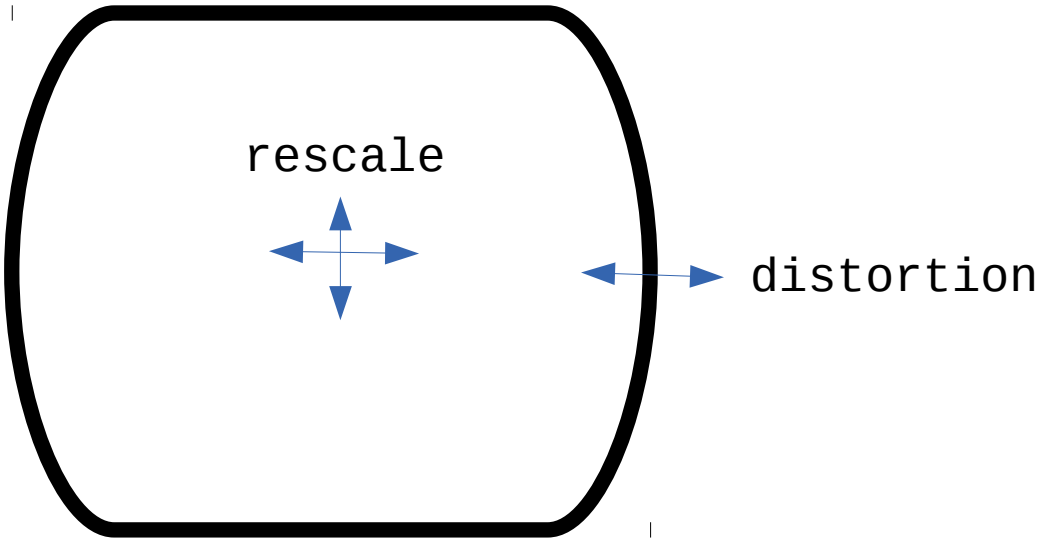
`sdltk vrmode` parameters in AndroWish

`vrmode 0` normal “un-virtual” operation

`vrmode 1` root window duplicated along horizontal axis and scaled by factor 0.5

`vrmode 2` root window treated as two halves, app must manage content

`vrmode 3` like mode 1 but without scaling



UVC stereo camera with tcluvc

- Tcl interface for UVC cameras called tcluvc is available in http://www.androwish.org/index.html/dir?name=jni/tcluvc
- Uses libusb and a modified version of libuvc and optionally libudev on Linux.
- Tested on Linux (x86, arm), Android (some devices work), and MacOS. *BSDs might work, too.
- For anaglyph images, the images of the left and right camera need be color filtered and mixed with `uvc mcopy`.

```
# called when a frame of a camera device is ready

proc img_callback {dev} {
    set imgname $::imgs($dev)    ;# photo image name assigned to camera
    uvc image $dev $imgname      ;# read frame to photo image L or R
    # combine to anaglyph, L is displayed
    uvc mcopy L R 0x0000ffff     ;# green/blue of R is copied into L
}
```

Tcl interface to Linux SocketCAN

- Linux provides a socket address family AF_CAN which implements a message based interface to CAN bus.
- AF_CAN raw sockets are similar to UDP (datagram) sockets.
- The message format is fixed due to the frame format of CAN bus.
- Messages have an 11 or 29 bit identifier and zero to 8 bytes payload data.
- A CAN bus (physically one or two wires plus ground) can be seen as a distributed memory (CAN identifier serves as address).
- CAN frames are sent with data rates ranging from some 10 kBit/s up to 1 Mbit/s.

```
struct can_frame {
    unsigned int  can_id;  /* CAN identifier, flags */
    unsigned char can_dlc; /* data length code: 0 .. 8 */
    unsigned char data[8]; /* message payload */
};
```

Tcl interface to Linux SocketCAN

- Tclcan is a Tcl interface to AF_CAN raw sockets written in C, see http://www.androwish.org/index.html/dir?name=undroid/tclcan
- The latest CAN bus spec defines CAN_FD (Flexible Datarate) which increases bandwidth up to factor 8 by packing up to 8 times more bits into the payload field of a frame (up to 64 byte). This is currently not supported by the Tcl interface.
- Some SOCs with Linux support have a CAN controller built in, e.g. the Beaglebone Black.
- Inexpensive CAN interfaces are available for the Raspberry Pi, google for CANberry.
- A cheap generic CAN interface for about €20 can be built with an Arduino and a MCP2515 breakout board, using the CAN serial line discipline, see instructions in http://www.androwish.org/index.html/dir?name=undroid/tclcan/arduino

tclcan commands (subset)

`can open` *`ifname`*	Opens a channel on interface *`ifname`* and returns a channel handle.
`can read` *`chan`*	Reads a CAN message from channel *`chan`* as a list of CAN identifier, payload as byte array, interface index, and a “more data” flag.
`can write` *`chan id data ?ifindex?`*	Writes a CAN message to channel *`chan`* with CAN identifier *`id`* and payload *`data`* as byte array.
`can dump chan`	Like `can read` but pretty prints message for debugging.
`can interfaces`	Returns list of CAN interfaces as interface name/index pairs.
`can start` *`ifname`*	Starts the CAN interface *`ifname`*.
`can stop` *`ifname`*	Stops the CAN interface *`ifname`*.
`can devstats` *`ifname`*	Retrieves device statistic information of CAN interface *`ifname`*.
`can bitrate` *`ifname ?rate ...?`*	Sets or retrieves the bitrate of CAN interface *`ifname`*.

tclcan channels

- Channels returned by `can open` are normal Tcl channels but do not support `gets`, `puts`, and `read`.
  Use `can read|dump` and `can write` instead.
- Use `chan event` or `fileevent` for a script to be invoked on readability of the channel.
- Special options for `chan configure` or `fconfigure` are

  `-error` to retrieve last system/socket error

  `-loopback` and `-ownmsgs` for controlling reflection of locally sent messages

  `-filter` to apply filter rules (masks and ranges) on CAN identifiers to be received.

Tcl interface to libmodbus

- http://libmodbus.org provides a library for the Modbus-RTU (typically over RS-485 serial lines) and Modbus-TCP protocols to connect to PLCs, sensors, actors.
- A proof of concept Tcl interface using Ffidl and TclOO is in the AndroWish source tree.
- Modbus registers can be read and written as boolean or integer values, see http://www.androwish.org/index.html/dir?name=assets/modbus0.1

```
package require modbus

modbus::new MB /dev/ttyS0 19200 N 8 10 ;# Modbus-RTU slave 10
MB connect                             ;# opens tty
MB serial_mode 1                       ;# sets RS-485 mode
MB write_bit 1000 1                    ;# single bit
MB read_registers 2000 10              ;# read 10*16bit
MB write_register 2002 42              ;# write 1*16bit
MB disconnect                          ;# closes tty
MB destroy                             ;# cleanup
```

Tcl interface to snap7

- snap7 is a library written in C++ to communicate with S7 PLCs over TCP/IP. It supports POSIX and Windows OSes. See http://snap7.sourceforge.net
- A proof of concept Tcl interface using Ffidl and TclOO is in the AndroWish source tree, see http://www.androwish.org/index.html/dir?name=assets/snap70.1
- PLC data blocks can be read and written as bytes or Tcl byte arrays.

```
package require snap7

snap7::new S7

S7 connect 1.2.3.4 102 0 2  ;# connect IP port rack slot
S7 dbread 1 0 10            ;# read DB 1, offset 0, 10 bytes
S7 dbwrite 1 0 1 2 3 4 5    ;# write DB 1, offset 0, 5 bytes
S7 disconnect
S7 destroy
```

Questions?