KineTcl

Andreas Kupries
ActiveState Software Inc.

19th Annual Tcl Conference
National Museum of Health + Medicine Chicago
Chicago, IL
Nov 12 – Nov 16 2012

KineTcl

- Background
- Architecture
- Tricks
- Tools
- Future
- Demo

KineTcl

- **Background**
- Architecture
- Tricks
- Tools
- Future
- Demo

KineTcl - Background

- NMHMC Project
  - Started January 2012
  - Working system wanted by May
  - Actually used since August
- Used in the Exhibition Hall
  - Detect people approaching a display
  - Send events to display controller

KineTcl

- Background
- **Architecture**
- Tricks
- Tools
- Future
- Demo

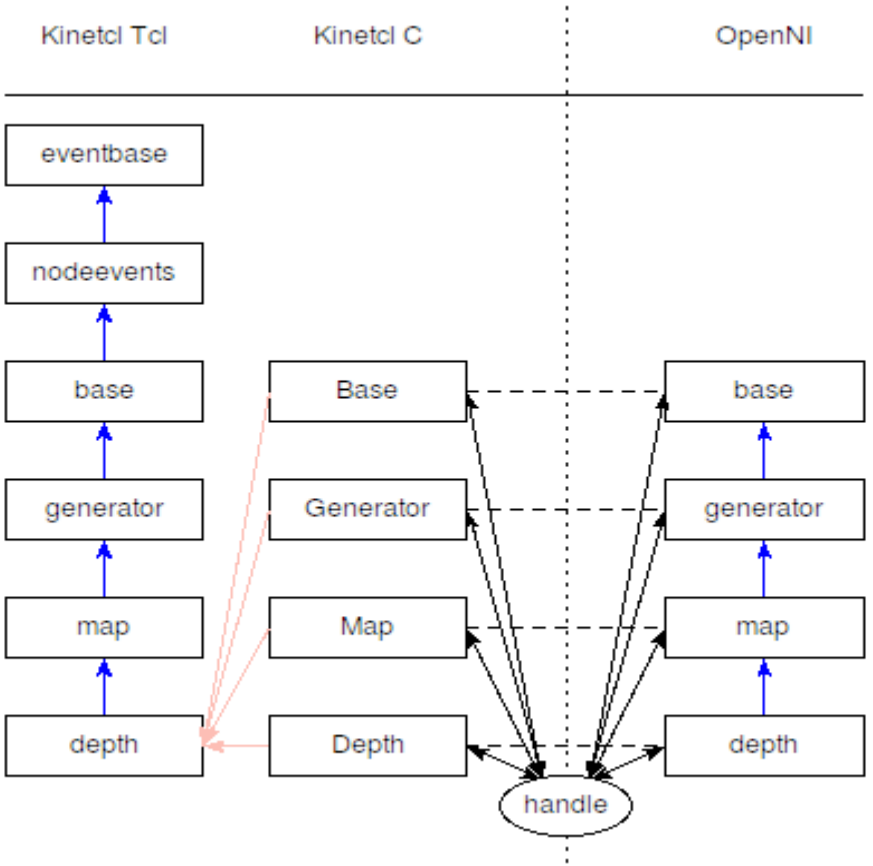
KineTcl - Architecture

- Existing software we could build on:
  - OpenKinect (aka libfreenect)
    - OSS Community
    - Lifts reverse engineered USB protocol into user space
    - Device access. No highlevel algorithms
  - OpenNI
    - PrimeSense (Depth Sensor manufacturer)
    - Open Framework
    - NITE middleware (user detection, skeleton tracking)
- Choose OpenNI for KineTcl, for NITE.

KineTcl - Architecture

- Layered, using C(riTcl) and Tcl(OO)
  - C code lifts OpenNI handles (objects) into Tcl
    - No (super)class hierarchy
  - Tcl glues the C classes into the proper hierarchy
    - And mix in the supported capability classes too.

KineTcl - Architecture



KineTcl

- Background
- Architecture
- **Tricks**
- Tools
- Future
- Demo

KineTcl - Tricks

- Instance Construction
  - Avoid passing a C pointer (the OpenNI handle) through Tcl.
  - Leaf classes (C layer) create OpenNI object, and store the resulting handle in a package-global, per-interp data structure.
  - Superclasses retrieve and use the handle instead of creating OpenNI objects.
  - Tcl layer (Base class) clears the communication storage.
    - **After** mixing the capability classes in.

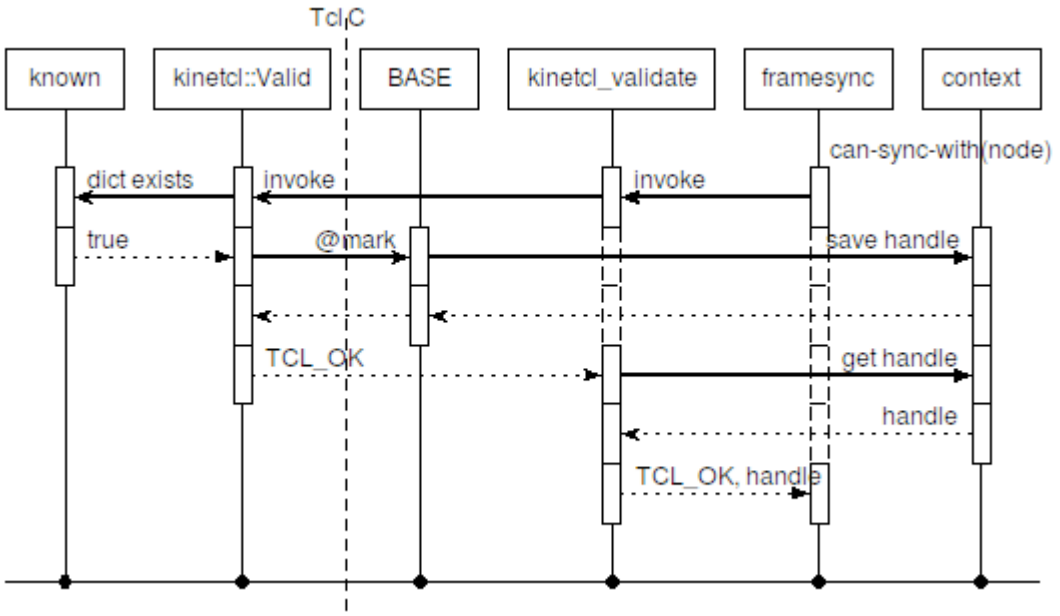
KineTcl - Tricks

- Tcl Object → OpenNI Handle
  - Done in cooperation between Tcl and C layers.
  - C layer calls up with the Tcl_Obj* to convert.
  - Tcl layer database of active objects can validate.
  - Tcl layer knows the internals, invokes the special methods to save handle information.
  - C layer retrieves then uses the stored handle.

KineTcl - Tricks

- Object → Handle conversion

  Sequence Diagram



KineTcl - Tricks

- Callbacks
  - Called by OpenNI threads → Can't call Tcl directly.
    - Solution: Convert Callbacks to Events, use Tcl_ThreadQueueEvent(), Tcl_ThreadQueueAlert()
  - Problem: A high-rate 'new frame' signal (@ 30 fps)
    - Solution: Event-coalescing (like for Mouse Motion).
  - Events delivered while Tcl processes events. Queue is never empty, processing never ends.
    - Solution: Defer delivery, save in spill-queue.
    - Exposed to Tcl level: **Hack**.
    - Future: Research Tcl “Event Sources” as means of hiding

KineTcl

- Background
- Architecture
- Tricks
- **Tools**
- Future
- Demo

KineTcl - Tools

- Critcl @ http://jcw.github.com/critcl/
  - Specifically 3.1 because of
  - critcl::class – Code generator package.
    - Takes a TclOO-like class definition
    - And generates all the C boilerplate needed for
      - Class and instance data structures
      - Class and instance Tcl commands.
      - Method dispatch
    - kinetcl::map : 4 KB critcl → 25 KB C
- CRIMP for images.

KineTcl - Tools

- Example:

```
critcl::class def ::kinetcl::CapFramesync {
    ::kt_abstract_class

    method can-sync-with proc {XnNodeHandle other} bool {
        return xnCanFrameSyncWith (instance->handle, other);
    }

    method start-sync-with proc {XnNodeHandle other} XnStatus {
        return xnFrameSyncWith (instance->handle, other);
    }

    method stop-sync-with proc {XnNodeHandle other} XnStatus {
        return xnStopFrameSyncWith (instance->handle, other);
    }

    method synced-with proc {XnNodeHandle other} bool {
        return xnIsFrameSyncedWith (instance->handle, other);
    }

    kt_callback framesync \
        xnRegisterToFrameSyncChange \
        xnUnregisterFromFrameSyncChange \
    {} {}
}

critcl::argtype XnNodeHandle {
    if (kinetcl_validate(interp, @@, &@A) != TCL_OK) return TCL_ERROR;
}

critcl::resulttype XnStatus {
    if (rv != XN_STATUS_OK) {
        Tcl_AppendResult (interp, xnGetStatusString (rv), NULL);
        return TCL_ERROR;
    }
    return TCL_OK;
}
```

KineTcl

- Background
- Architecture
- Tricks
- Tools
- **Future**
- Demo

KineTcl - Future

- Research into gesture recognition.
  - Example: FAAST
    - ICT http://projects.ict.usc.edu/mxr/faast/

      Flexible Action & Articulated Skeleton Toolkit
- User recognition (geometric user hash)
- Tcl “Event Sources”
  - Less exposure of event innards
- Implement things not used here at NMHMC
  - Player, Record, Script?, Audio, Motor (Sensor Pan)
  - Introspection, node stacks, non-default instances

KineTcl

- Background
- Architecture
- Tricks
- Tools
- Future
- **Demo**

KineTcl – Location

Where ?

- http://chiselapp.com/user/andreas_kupries/ \
repository/KineTcl
- On the USB-Stick