CSE 340
Winter Quarter 2020
2/3/2020

Lecturer: Dr. Jennifer Mankoff
Note-taker email: jial8@uw.edu

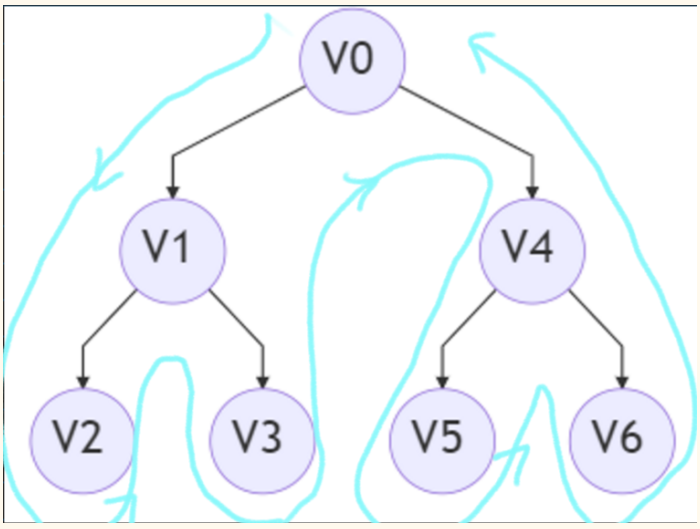
Link to Notes on Google Docs:
https://docs.google.com/document/d/1zJ_QvvwNytcAK0qD733xIRVGp1qyfSuac4IElI4vyEY/edit?usp=sharing

Week 5 Lecture 1

Essential Geometry and View Updates

Input Dispatching Process

- *Input Thread:* when user interacts → events are created, events go in a **queue**
- *Dispatch Thread:* front event removed from queue
    - How toolkit decides where to send events?
    - Input Process - Picking (see below diagram)



Traverse the tree **pre-order(?)** (see blue line, professor Mankoff was unsure, need to clarify later) looking for an event to send to.

Can send to **multiple** events, or none! Views **without listeners for that event won't be considered** at all.

Order matters!

Dispatch asks: will you **consume** this event?

    - True → event propagation **stops**, event consumed, don't search tree anymore
    - False → Move to the next element in the View list... why?
        - Want to log all the views, keep track of all clicks
        - Click within bounding box of circular color picker, but is not actually clicking the circle
- Some kinds of interactors want to only capture an event after seeing that the other views did not use it
    - Ex: log things that weren't consumed
    - Outer ScrollView can safely scroll (no inner ScrollView to be scrolled by user)
    - **Capture Order** (order in which events are sent in the tree traversal)
    - **Bubbling Order** (ask if apply events, opposite of capture order)
        - Android backtracks through the list asking if they actually want to consume the event
- Callbacks handle *application* response to events
    - Update application model

Implementing Drag and Drop

- Drag+Drop function → Focus or Positional?
    - Uses both! Shifts to either when necessary.

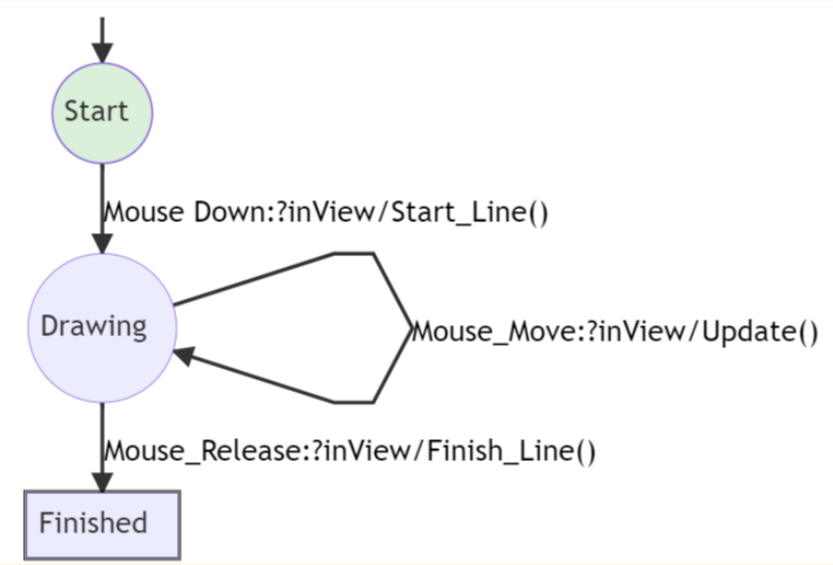
Overall: how does Android decide where to send events?

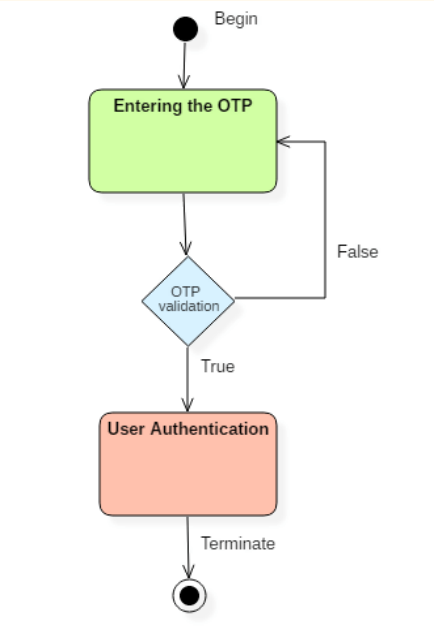
1. **Capture** (top to bottom traversal, most things don't capture the event)
2. **Pick**/focus on objects of interest
    a. buildTouchDispatchChildList(), happens *only after Capture!*

3. **Bubble** (bottom to top)
    a. onTouchEvent()
4. Invoke **Callback** (wait until complete)
    a. Custom listener in Android

Interaction Technique

- Method for carrying out specific interactive task
- Ex: user specify endpoints for a line
    - Click start click end....
    - OR better method = "rubber-banding", stretch the line out when user drags mouse further away from start
- **State Machine:** models transitions → see right
    - Lines represent actions





Essential Geometry

- **Propositional Production System** (aka "guards")
- Adds extra conditions before you can apply event
- Ex: color picker, did the user actually click on the circle or just its bounding box?

- Typical notation = **event: pred?: action**
    - Ex: **MouseDown : InsideAboveThumb? Scrollup()**