The Application Stack

Application Program → · Doodle, Facebook...
High Level Tools
Toolkit → · graphics, animation, event handling...
Window System → · window size/visibility · clipping, redraw
Operating System
Hardware → · macOs, Linux...etc

Developer Roles

- Interface Programmer → · combine libr. elements + toolkit rules + arch. constrain.
- Component Developer
- Library Extender → new input types, layouts, styling
- Architecture Ext.
- Toolkit Builder → · Physical Interf. · Android Studio

· ex: Lineview
· new, re-usable
· supported by very few toolkits
· modifies flow of information w/ toolkit to create entirely new effects
· animation

Drawing on Canvas

PRE-ORDER

· onDraw() → automatically called by toolkit → triggered whenever pixel dirty
· Raster: array of bits within rect. grid of pixels or dots, more scalable than vector
· Vector: lines + shapes (math statements)
  - Paths
  - more "crisp"-looking
· How to rotate about center of an object? [Translate, Rotate, Translate] relative to entire screen.

Visual Perception

"smooth pursuit" = "constant move" = focus jumping everywhere

- Rods (night vision, many > cones, movement)
- Cones (color, only 3 types RGB) → 24 bit + 8 bit alpha

Hue: Dominant wavelength (ex: green)
Saturation: How much color vs. white / black (pink vs. red)
Value: How dark (ex: black vs. white, dark red vs. red)

culturally defined

★ > 50 flashing images/sec begins to look "static" to humans

★ 1 2 seconds 10 15
good response time — bad response

Layout

can define specific layout properties, ex: LinearLayout

ViewGr.
- View (image, text... etc)
- View
- ViewGr. ...

Poor BORDER measure

Overlapping Elements

ex: modify ScrollView, constrain top of ScrollView to Bot. of "Route"

Accessibility and Inclusive Design...

Disability ---→ meaningful Tab orders, labels, contrast, target size, helpful event notification
- context
- mismatched interaction
- spectrum

Event Record

- what (Key pushed)
- where (input component)
- when (timest.)
- Value (mouse coord, which key...)
- context (CTRL+A, # of clicks)

Input dispatch

Logical Device Approach

Valuator: scalar val.
Button: int value
Locator: position on logical view surface
Keyboard: charact. string
Stroke: sequence of points

★ Model View Controller

· model state: Adam
· Event Handling: Password Entry

Knows curr. state, updates model
- tells model how to change

User interacts w/ View, creates event → trigger redraw and show

View → (input) Controller → (output to view) View
Controller → (state change) model

★ Input Types

· mouse → relative positioning
· touchscreen → absolute positioning

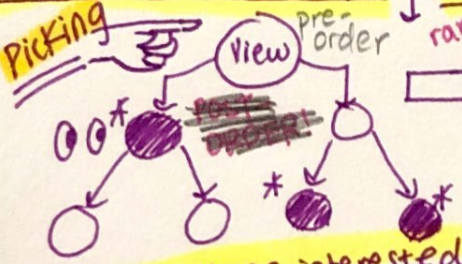
① Queue (input thread) — event
② Remove Front
· where to? → Focus (interest) → no picking, globally created
→ Positional (z-order under cursor)
· Bottom-first + Top-Down

③ callbacks handle application response to events, ex: action_listener(Action)

Interface of View class must be attached to particular View
register by creating anon. class or implem. interface

Picking

View (pre-order)



rarely used by comp. dev.

Capture! or Bubbling → post-order (reverse order)

Will you consume event???
A: True → ~~event~~ PROP.STOP

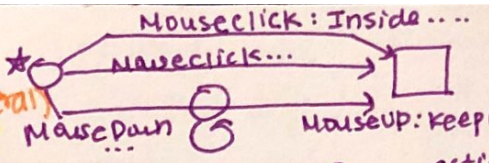
· onTouchEvent()
· used by component devs.
· default order

★ pick only those interested (not inside View, doesn't have listener for that input)

★ Note: CTRL != event, = modifier

state machine: handles event driven code, maintain state between events

very good at handling multiple actions, don't handle independ. act well, useful for smaller things



- based on region (general)
- **Essential Geometry**
  - PPS (propositional production system)
    - MouseDown: InsideAboveThumb? scrollUp()
    - extra conditions required to fire
    - aka "guards"
    - control flow in event-driven programs

action

predicate

- outside vs. inside

Design Tips!

Gestalt Psychology

blue+small

contrast

↓↓ saturated colors

redundant cues ★★★★

distinct!

small multip.

manage expectations...

Obvious cues!!!

well-tested grouping

- option 1
- option 2
- option 3
- option 10000000

recognition

recall

more on state machine / Handle Dispatch

- implement event handler
- update state/model
- call invalidate()

don't directly call onDraw

more Quiz content

Application *

- setListener() ← called by
- is listening for offRoute

ToolKit Arch.

- onOffRoute() ← called by (doesn't know when)

implement by

implem. in a View or some other thing outside Application/View

- button pressed

ex: library

* **Application** (tells when to use it)

- when Button pressed (no need to make multiple)

- CircleView:
  - center (100,100)
  - radius = 10
  - cx, cy = (10,10)

- focus - used during event-handling → mouse moves off scrollbar