Input

5

Georgia Tech

(structure and 2D content based on CS4470/6456 slides by Keith Edwards)

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Where we are...

- Two largest aspects of building interactive systems: output and input
  - Have looked at basics of output
  - Now look at input

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Input

- Generally, input is somewhat harder than output
  - Less uniformity, more of a moving target
  - More affected by human properties
  - Not as mature
    - Very little standardized for mapping 2D inputs to 3D on flat screens
    - Every VR device has different controllers
    - Most companies striving for hand, voice, gesture, eye, …
    - AR (and soon VR) also depends on sensing the physical space and user's interactions with that space
- Will start with simple low level (devices) and work up to higher level

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Input devices

- Keyboard
  - Ubiquitous, but somewhat boring…
  - Quite mature design
- non-letter/number buttons as well
  - on keyboard, mouse, separate boxes, etc
- QWERTY key layout
  - Where did it come from?

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

QWERTY key layout

- Originally designed to spread out likely adjacent key presses to overcome jamming problem of very early mechanical typewriters
  - Often quoted as "intentionally slowing down" typing, but that's not true
    - Arrangement of letters to keep typebars from getting stuck
    - (Common letter pairs on alternating hands)

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

QWERTY keyboard layout

- Other layouts have been proposed
  - Dvorak is best known
  - Widely seen as better
  - Experimental and theoretical evidence casts doubt on this
    - Alternating hands of QWERTY are a win since fingers move in parallel

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Keyboards

- Repetitive Stress Injury
  - First comes up here, mouse tends to be a little worse for most people
- Take this seriously for yourself!
  - Can be a VERY bit deal
  - Biggest thing: adjust your work environment (e.g. chair height)

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Valuators

- Returns a single value in range
- Major impl. alternatives:
  - Potentiometer (variable resistor)
    - similar to typical volume control
  - Shaft encoders
    - sense incremental movements
- Differences?

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Valuator alternatives

- Potentiometer
  - normally bounded range of physical movement (hence bounded range of input values)
  - Keeps residual position in device
- Shaft encoder
  - Unbounded range of movement
  - No residual position in device

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Locators (AKA pointing devices)

- Returns a location (point)
  - two values in ranges
  - usually screen position
- Examples
  - Mice (current defacto standard)
  - Track balls, joysticks, tablets, touch panels, etc.
  - Analog sticks on controllers

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Locators

- Two major categories:
  - Absolute vs. Relative locators

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

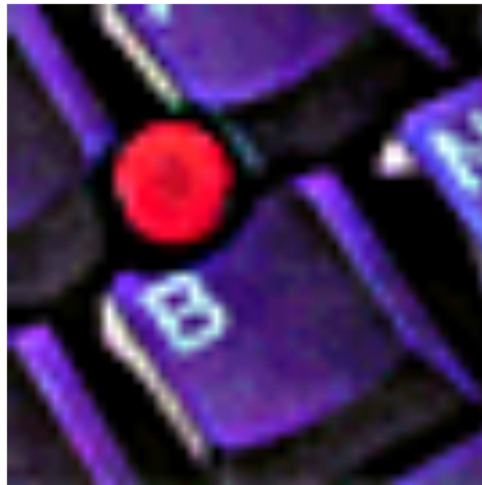
Absolute locators

- One-to-one mapping from device movement to input
  - e.g., tablet
  - Faster
  - Easier to develop motor skills
  - Doesn't scale past fixed distances
    - bounded input range
  - less accurate (for same range of physical movement)

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Relative locators

- Maps movement into rate of change of input
  - e.g., joystick (or TrackPoint)



Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Relative locators

- More accurate (for same range of movement)
- Harder to develop motor skills
- Not bounded (can handle infinite moves)

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Q: is a mouse a relative or absolute locator?

Georgia Tech

5

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Q: is a mouse a relative or absolute locator?

Georgia Tech

5

- Answer: No
- Third major type:
  "Clutched absolute"
  - Within a range its absolute
  - Can disengage movement (pick it up) to extend beyond range
    - picking up == clutch mechanism

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Georgia Tech

Clutched absolute locators

- Very good compromise
  - Get one-to-one mapping when "in range" (easy to learn, fast, etc.)
  - Clutch gives some of benefits of a relative device (e.g., unbounded)
- Trackballs also fall into this category
- As do many UI techniques that use 3D input with controllers

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Device specifics: joysticks

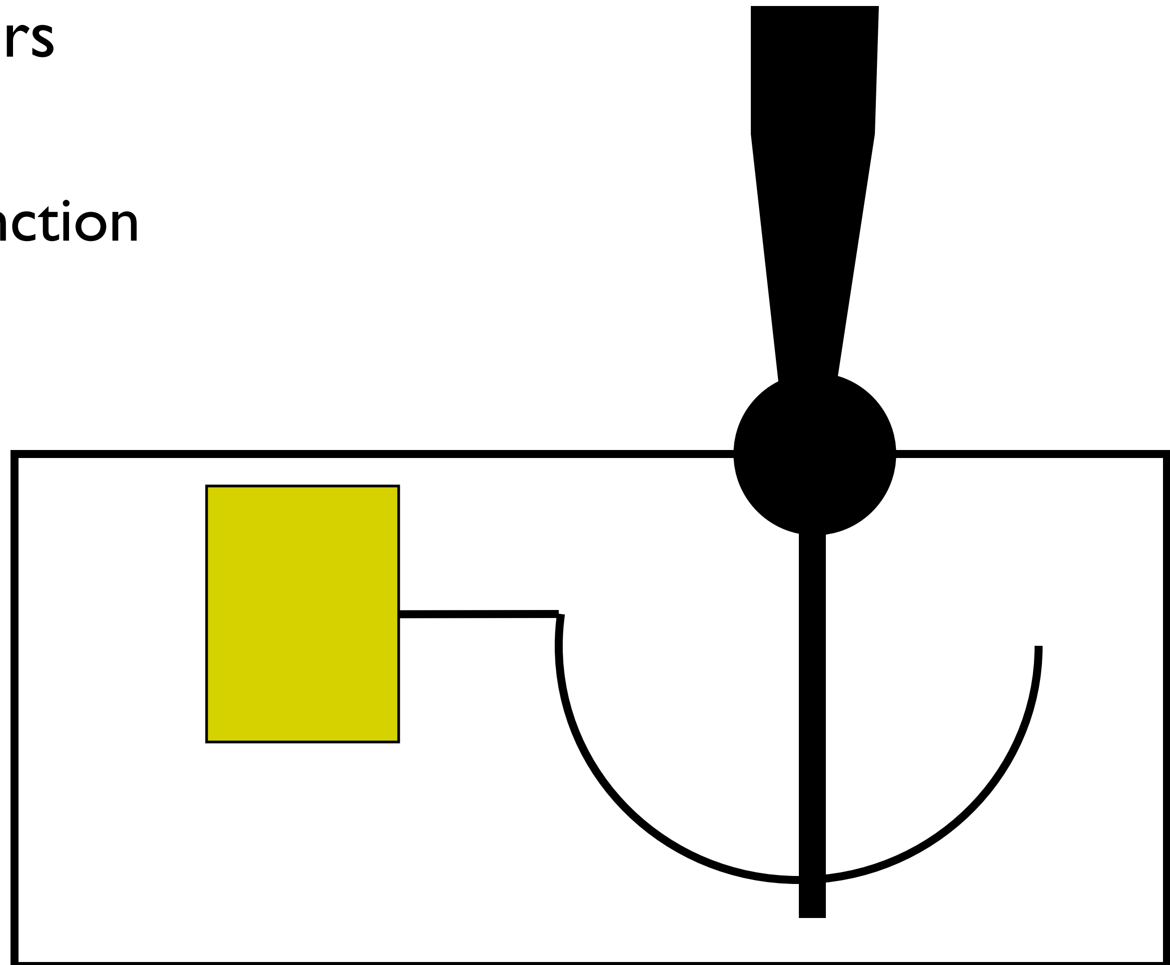
- self centering
- relative device
- possible to have absolute joysticks, but scaling is bad

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Georgia Tech

Joystick construction

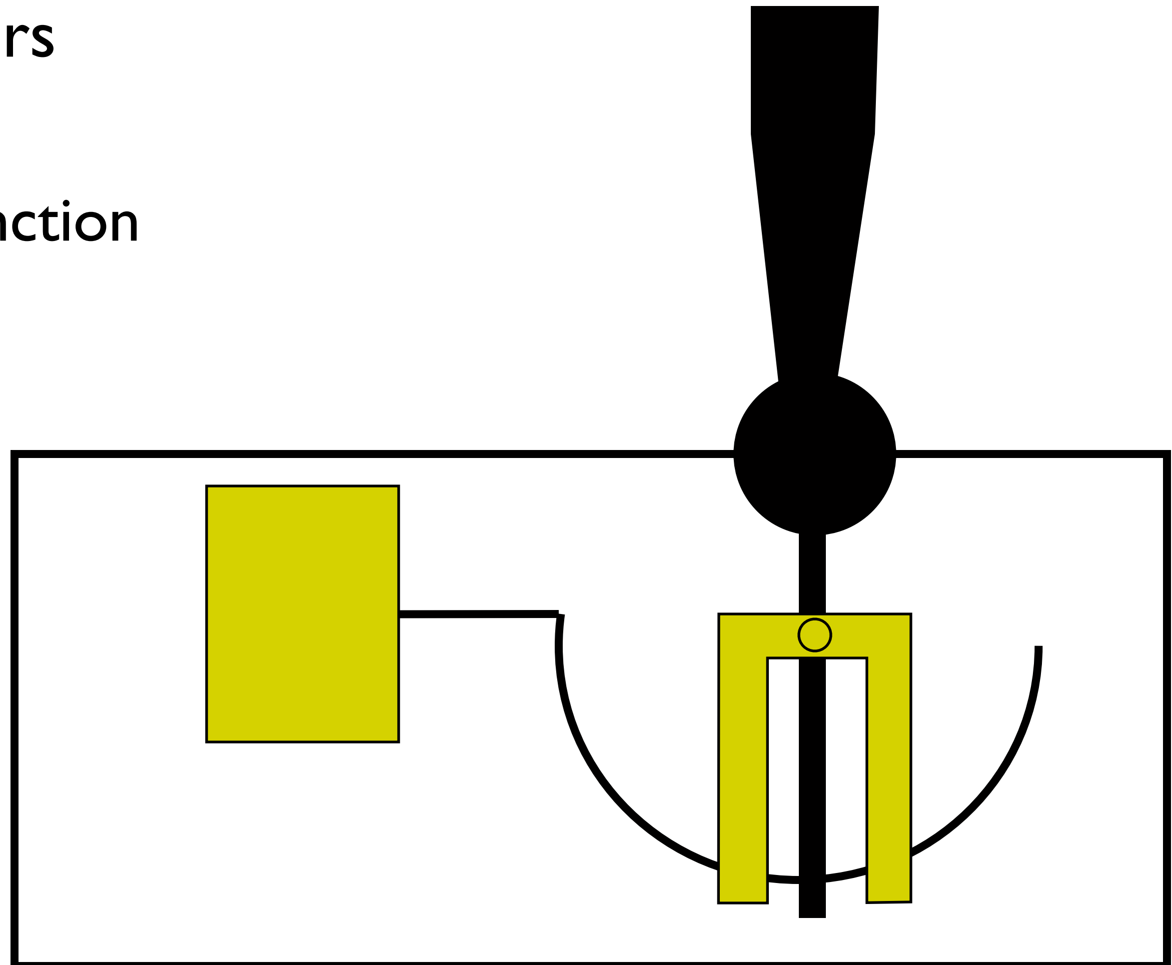
- Two potentiometers
  - x and y
  - resistance is a function of position



Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Joystick construction

- Two potentiometers
  - x and y
  - resistance is a function of position



Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

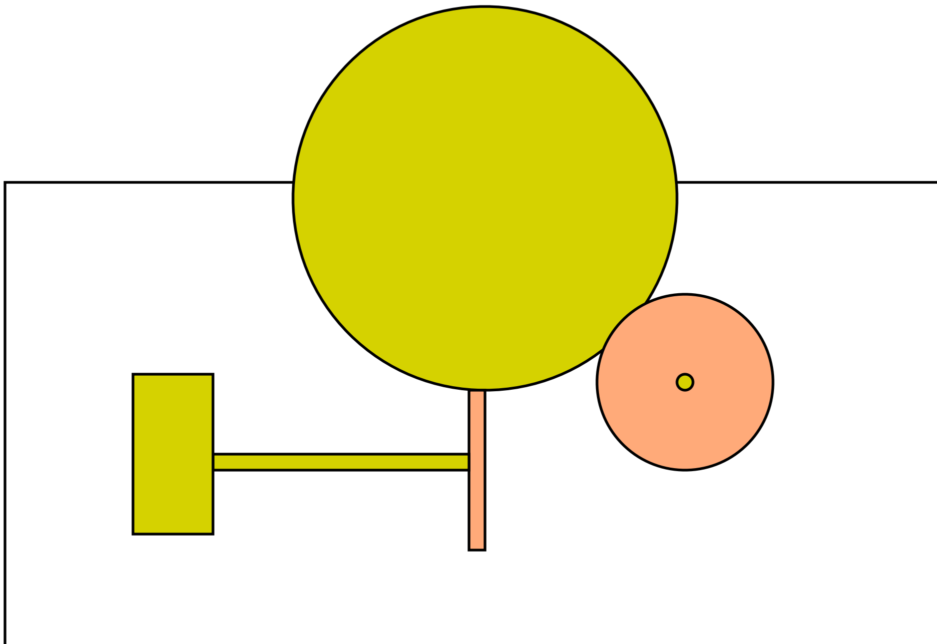
Joystick construction

- TrackPoint (IBM technology)
  - uses strain gauge sensors
- Also can be implemented with switches
  - one in each direction
  - Fixed speed of movement

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Trackballs

- (Typically large) ball which rolls over 2 wheels



Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Trackballs

- Clutched absolute
  - but with small movement range
- Infinite input range, etc.
- Properties vary quite a bit
  - scaling of movements
  - mass of ball
    - high mass ball can act as a relative device by spinning it

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Mouse

- Clutched absolute
  - infinite range, etc.
- How is it constructed?

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Mouse

- Clutched absolute
  - infinite range, etc.
- How is it constructed?
  - Turn a trackball upside down

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Mouse

- Current dominant 2D device
  - so much so that some people call any pointing device a "mouse"
  - overall a very good device

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Mouse

- Invented by Douglas Engelbart et al. ~1967

http://sloan.stanford.edu/MouseSite/Archive/AugmentingHumanIntellect62/Display1967.html

Cop

Touch panel

- What kind of a device?

Touch panel

- Absolute device
- Possible to do input and output together in one place
  - actually point at things on the screen
- Resolution limited by size of finger ("digital input")
  - Or requires a pen

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Touch panel construction

- Membrane
  - resistive, fine wire mesh
- Capacitive
- Optical
  - finger breaks light beam
- Surface acoustic waves

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

3D locators

- Can extend locators to 3 inputs
- Some fun older devices
  - 3D acoustic tablet
  - Wand on reels
  - Multi-axis joystick

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

3D locators

- Common pre-modern device for VR : Polhemus
  - 6D device (x,y,z + pitch, roll, yaw)
  - Magnetic sensing technology
    - Doesn't work well near metal
    - Doesn't work well near deflection coils of CRT

- Magic Leap ML1 uses similar tech, as do a number of stand-alone VR displays

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Georgia Tech

5

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Georgia Tech

5

Dealing with diversity

- Lots of diversity in devices
  - actual details of devices (e.g., device drivers) is a real pain
  - how do we deal with the diversity?
- Need a model (abstraction) for input
  - like file systems abstract disks
  - higher level & device independent

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Logical device approach

- One approach "logical devices"
  - A logical device is characterized by its software interface (only)
    - the set of values it returns
  - Rest of semantics (how it operates) fixed by category of device or left to the particular device

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Logical device approach

- Fixed set of categories
  - old "Core Graphics" standard had 6
    - keyboard, locator, valuator, button
    - pick, stroke
- If actual device is missing, device is simulated in software
  - valuator => simulated slider
  - 3D locator => 3 knobs
- 1st step towards today's interactors

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Logical device approach

- Abstraction provided by logical device model is good
- But… abstracts away too many details (some are important)
  - example: mouse vs. ipad stylus
    - Both are locators
    - What's the big difference?

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Not a success but..

- Still useful to think in terms of "what information is returned"
- Categorization of devices useful
  - Two broad classes emerged
    - Event devices
    - Sampled devices

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Georgia Tech

Categorization of devices

- Event devices
  - Time of input is determined by user
    - Best example: button
    - When activated, creates an “event record” (record of significant action)

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Categorization of devices

- Sampled devices
  - Time of input is determined by the program
    - Most valuator or locator devices
      - VR controllers and hand trackers, anything that provides continuously changing data
    - Value is constantly updated
      - Might best think of as continuous
    - Program retrieves current value when it needs it

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

OpenXR Input

Georgia Tech

Events

Events are messages sent from the runtime to the application. They're put into a queue by the runtime, and read from that queue by the application using `xrPollEvent`

Event	Description
XrEventDataEventsLost	event queue has overflowed and some events were lost
XrEventDataInstanceLossPending	application is about to lose the instance
XrEventDataInteractionProfileChanged	active input form factor for one or more top level user paths has changed
XrEventDataReferenceSpaceChangePending	runtime will begin operating with updated space bounds
XrEventDataSessionStateChanged	application has changed lifecycle state

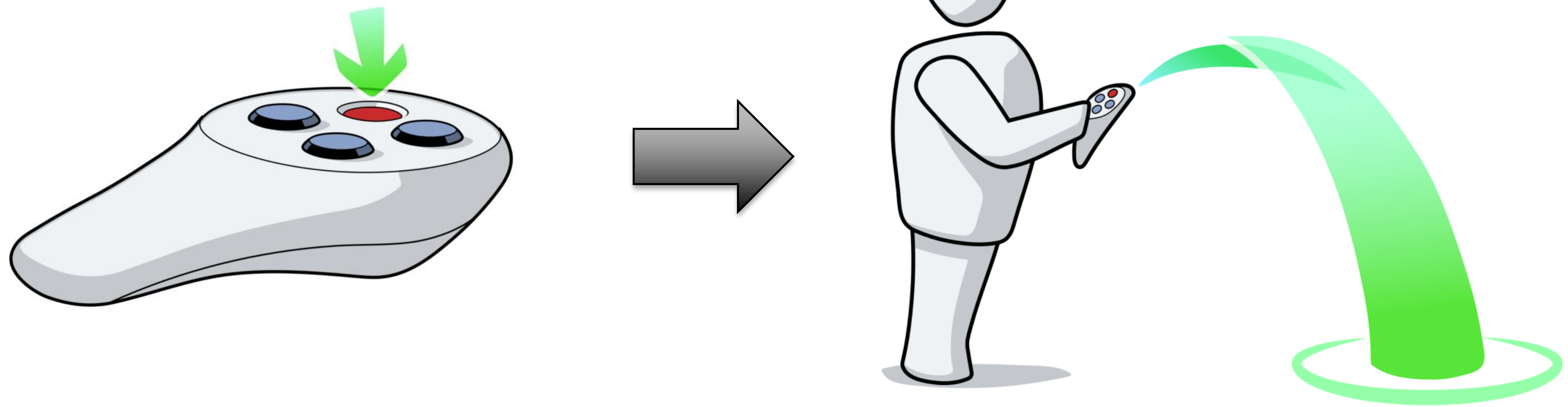
KHRONOS GROUP

© Khronos® Group Inc. 2019 - Page 27

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Georgia Tech

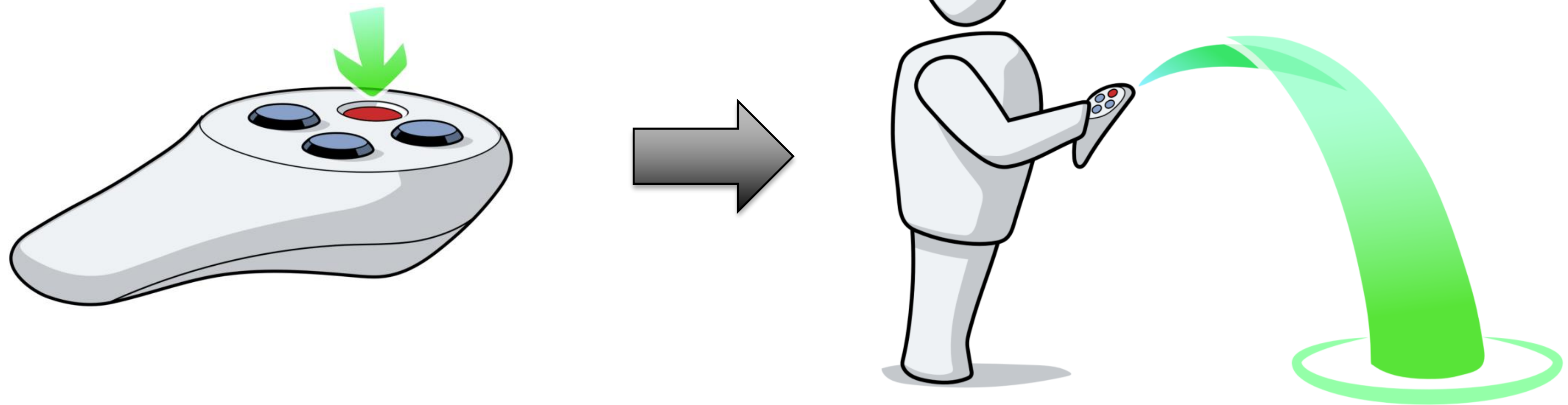
Input and Haptics



© Khronos® Group Inc. 2019 - Page 28

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Input and Haptics



When user clicks button "a" it results in the user teleporting

© Khronos® Group Inc. 2019 - Page 29

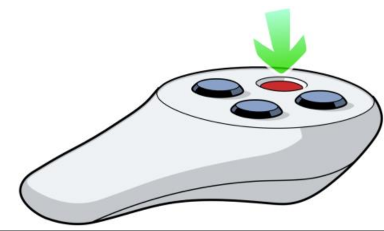
Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Georgia Tech 5

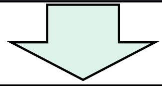
Input and Haptics

Input in OpenXR goes through a layer of abstraction built around Input Actions (`XrActions`). These allow application developers to define input based on resulting action (*e.g. "Grab", "Jump," "Teleport"*) rather than explicitly binding controls

While the application can suggest recommended bindings, it is ultimately up to the runtime to bind input sources to actions as it sees fit (application's recommendation, user settings in the runtime's UI, etc)

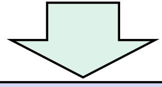


/user/hand/left/input/a/click
(/interaction_profile/ControllerCorp/fancy_controller/input/a/click)



OpenXR Runtime

.../input/trigger/click	Explode
.../input/a/click	Teleport



XrAction: "Teleport"

KHRONOS GROUP

© Khronos® Group Inc. 2019 - Page 30

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Georgia Tech

Input and Haptics - Interaction Profiles

- Collections of input and output sources on physical devices
- Runtimes can support multiple interaction profiles

ControllerCorp's Fancy_Controller:

- /user/hand/left
- /user/hand/right

- /input/a/click
- /input/b/click
- /input/c/click
- /input/d/click
- /input/trigger/click
- /input/trigger/touch
- /input/trigger/value
- /output/haptic

example

© Khronos® Group Inc. 2019 - Page 31

KHRONOS GROUP

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Input and Haptics - Predefined Interaction Profiles

- **Interaction profiles for many current products are predefined in the OpenXR specification including:**
  - Google Daydream* controller
  - HTC Vive and Vive Pro* controllers
  - Microsoft* Mixed reality motion controllers
  - Microsoft* Xbox controller
  - Oculus Go* controller
  - Oculus Touch* controllers
  - Valve Index* controllers

KHRONOS® GROUP

© Khronos® Group Inc. 2019 - Page 32

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Input and Haptics - Runtime Binding Decision – Why?

- **Runtime ultimately decides the bindings**
  - “dev teams are ephermal, games last forever”
  - More likely the runtime is updated than individual games
- **Reasons for selecting different bindings:**
  - 1. this runtime does not have ControllerCorp’s fancy_controller currently attached, but it knows how to map the inputs and outputs to the controllers that *are* attached
  - 2. Some runtimes can support user mapping of inputs such that the controls per game can be customized by the user, such as swapping trigger and button ‘a’, this enables customization without the original application knowing about it
  - 3. Some future controller is developed but the application is not updated for it, a new interaction profile can help map the actions to the new inputs
  - 4. Accessibility devices can be used and input mapped appropriately

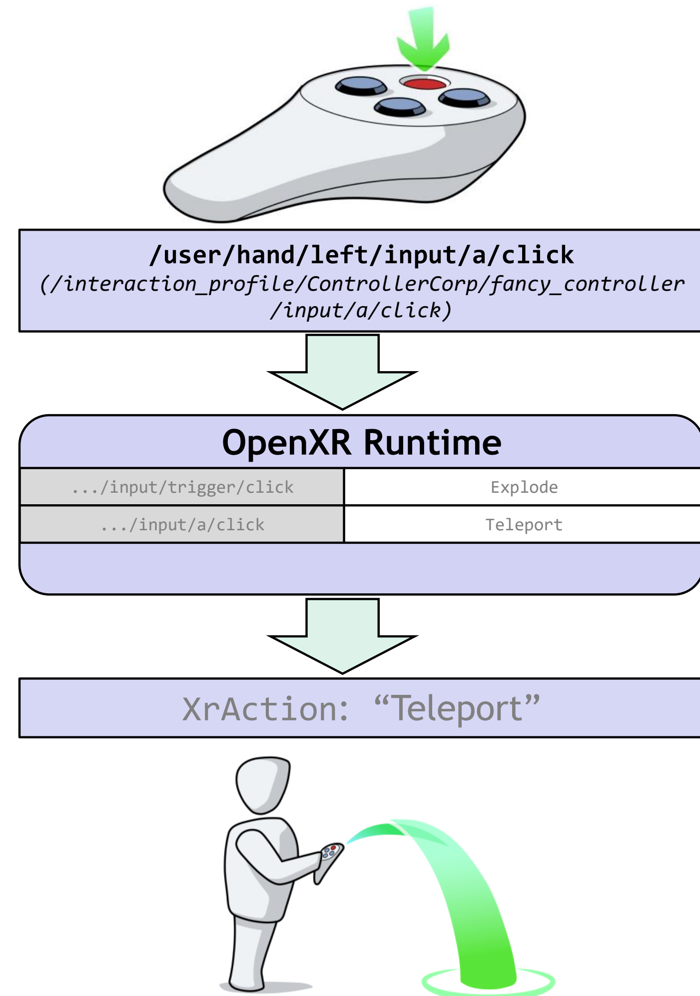
© Khronos® Group Inc. 2019 - Page 33

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Input and Haptics

Input in OpenXR goes through a layer of abstraction built around Input Actions (`XrActions`). These allow application developers to define input based on resulting action (*e.g. "Grab", "Jump," "Teleport"*) rather than explicitly binding controls

While the application can suggest recommended bindings, it is ultimately up to the runtime to bind input sources to actions as it sees fit (application's recommendation, user settings in the runtime's UI, etc)



KHRONOS GROUP
© Khronos® Group Inc. 2019 - Page 30
Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

A unified model

- Anybody see a way to do both major types of devices in one model?

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Georgia Tech

A unified model: the event model

- Model everything as events
  - Sampled devices are handled with “incremental change” events
  - Each measurable change in value produces an event containing the new value
  - Program can keep track of the current value if it wants to sample

- Popular on 2D systems, not generally used in immersive APIs
  - Doing desktop 3D requires you use this for keyboard and mouse, and deal with the potential issues

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Simulating sampling under the event model of input

- Can cause problems
  - lots of little events
    - Can fall behind if doing a lot of computation/redraw for every event
      - machines are fast, blah blah blah
      - but can get behind (sampling provided built in throttling)

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

The event input model

- Almost all systems now use this
- An “event” is an indication that “something potentially significant” has just happened
  - in our case user action on input device
  - but, can be generalized

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Georgia Tech

5

The event input model

- "Event records" are data structures (or objects) that record relevant facts about an event
  - generally just called "events"
- Event records often passed to an "event handler" routine
  - sometimes just encode relevant facts in parameters instead of event record

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Relevant facts

- What do we need to know about each event?

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Relevant facts

- What
- Where
- When
- Value
- Additional Context

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Georgia Tech

What

- What (exactly) caused the event
  - e.g., left mouse button went down
  - for "method based" systems this may be implicit in what handler gets called

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Where

- Where was the primary locator (mouse) when event happened
  - x,y position
  - also, inside what window, object, etc.
  - this is specific to GUIs, but it;s critical
    - e.g., can't tell what mouse button down means without this

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

When

- When did the event occur
  - Typically are dealing with events from the (hopefully recent) past
    - queued until program can get to them
  - In absolute time or relative to some start point
  - Hopefully at resolution of 10s of ms
    - important for e.g., double-clicks

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Value

- Input value
  - e.g., ASCII value of key press
  - e.g., value of valuator
  - some inputs don't have a value
    - e.g. button press

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Additional context

- Status of important buttons
  - shift, control, and other modifiers
  - possibly the mouse buttons

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Extending the event model

- Events can extend past simple user inputs
  - Extra processing of raw events to get "higher level" events
    - window / object enter & exit
    - list selection
    - rearrangement of the interactor hierarchy
  - Can extend to other "things of significance"
    - arrival of network traffic
    - gestures
    - voice input (sync results with time spoken in past)

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Georgia Tech

Extending the event model

- Window systems typically introduce a number of events
  - window enter/exit   region enter/exit
    - system tracks mouse internally so code acts only at significant points
  - Redraw / damage events
  - Resize & window move events
- 3D systems have some of these, but immersive UIs don't have notions like windows or "regions" for an application (yet)

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Georgia Tech

5

Synchronization and events

- The user and the system inherently operate in parallel
  - In 2D systems, everything happens asynchronously
    - Means different programming model for applications (asynchronous callbacks)
    - Means special work for toolkit/window system implementations
  - In 3D systems, more synchronized with the redraw method
    - Even though the *human* is still asynchronous
- This is a producer consumer problem
  - user produces events
  - system consumes them

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

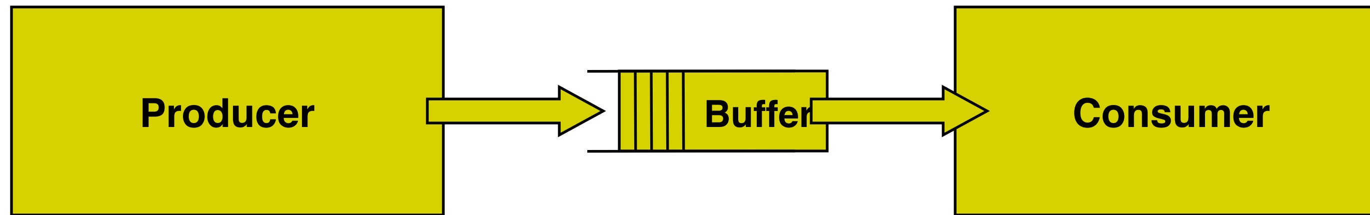
Synchronization and events

- Need to deal with asynchrony
  - both parties need to operate when they can
  - but can’t apply concurrency control techniques to people
- How do we handle this?

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Synchronization and events

- Use a queue (buffer) between



  - As long as buffer doesn't overflow, producer does not need to block
  - Consumer operates on events when it can

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Implications of queued events

- We are really operating on events from the past
  - hopefully the recent past
- But sampled input is from the present
  - mixing them can cause problems
  - e.g. inaccurate position at end of drag

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Using events from an event queue

- Basic paradigm of event driven program can be summed up with one prototypical control flow
  - Will see several variations, but all on the same theme

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Using events from an event queue in 2D systems

Georgia Tech

```
Main_event_loop()
    init();
    set_input_interests();
    repeat
        evt = wait_for_event();
        case evt of
            … dispatch evt -- send to some object
        end case;
        redraw_screen();
    until done;
```

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Using events from an event queue

- Very stylized code
  - in fact, generally you don't even get to write it
  - often only provide system with routines/methods to call for "dispatch"

```
repeat
   evt = wait_for_event();
   user_object.handle_event(evt);
   redraw_screen();
until done;
```

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Using events from an event queue

- Two big questions:
  - What object(s) gets the event?
  - What does it do with it?
    - Interpret it based on what the event is, what the object is, and what state the object is in

Georgia Tech

5

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Dispatch strategies: what object gets the event

- Simple approach
  - lowest object in interactor tree that overlaps the position in event gets it
    - if that object doesn't want it, try its parent, etc.
  - "Bottom first" dispatch strategy

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Dispatch strategies: what object gets the event

Georgia Tech

5

- Can also do "top-first"
  - root gets it
  - has chance to act on it, or modify it
  - then gives to overlapping child
  - has another chance to act on it if child (and its children) doesn't take it

➔ more flexible (get top-first & bottom-first)

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

But… a problem with fixed dispatch strategies like this

- Does this work for everything?

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

But… a problem with fixed dispatch strategies like this

- Does this work for everything?
  - What about key strokes?
  - Should these be dispatched based on cursor location?
    - Probably not
    - Probably want them to go to "current text focus"

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Two major ways to dispatch events

- Positional dispatch
  - Event goes to an object based on position of the event
- Focus-based dispatch
  - Event goes to a designated object (the current focus) no matter where the mouse is pointing

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Question

- Would mouse events be done by focus or positional dispatch?

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Georgia Tech

5

Question & answer

- Would mouse events be done by focus or positional dispatch?
- It depends…
  - painting: use positional
  - dragging an object: need focus (why?)

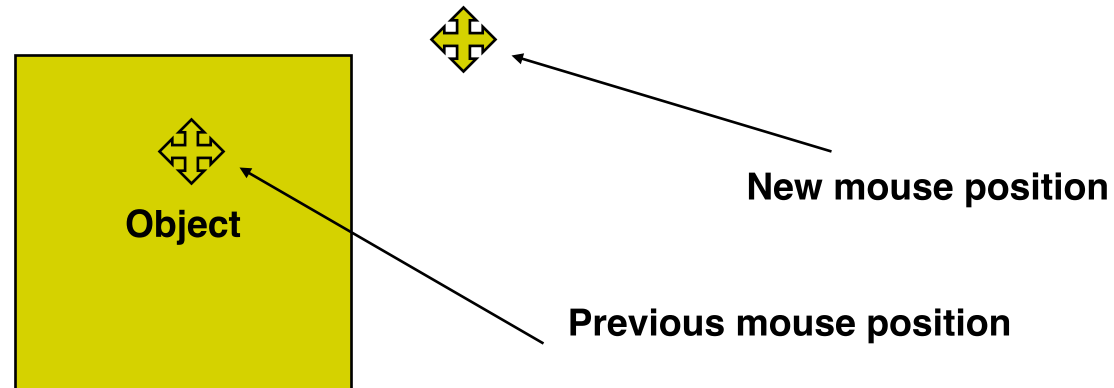
Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Dragging an object needs focus dispatch

Georgia Tech

5

- Why? What if we have a big jump?



- Cursor now outside the object and it doesn't get the next event!

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Positional and focus based dispatch

- Will need both
- Will need a flexible way to decide which one is right
  - will see this again later, for now just remember that sometimes we need one, sometimes another

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Positional dispatch

- If we are dispatching positionally, need a way to tell what object(s) are "under" a location
- "Picking"

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Picking

- Probably don't want to pick on the basis of a point (single pixel) or infinitely thin line/ray
  - Why?

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

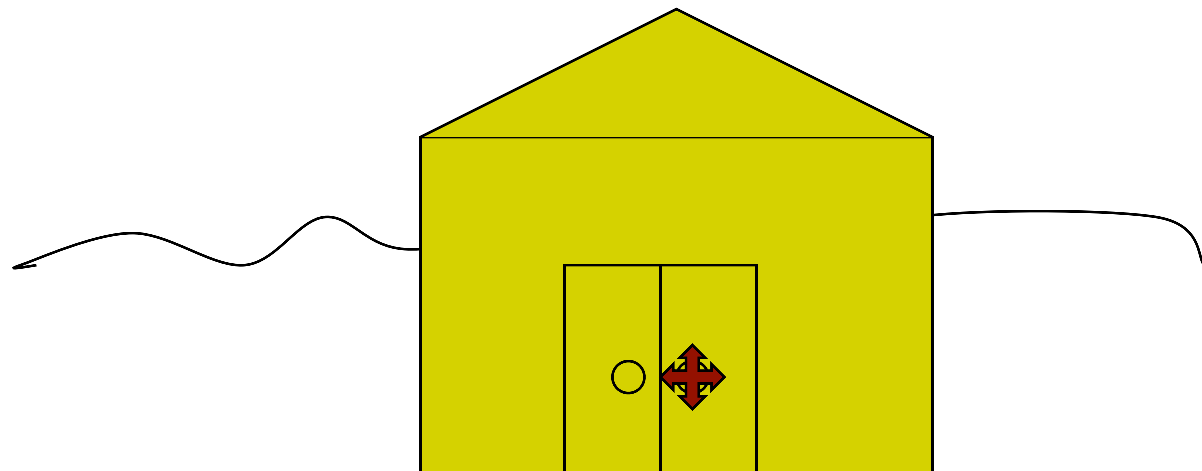
Picking

- Probably don't want to pick on the basis of a point (single pixel) or infinitely thin line/ray
  - Why?
  - Because it requires a lot of accuracy
- Instead may want to pick anything within a small region around the cursor or ray

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Pick ambiguity

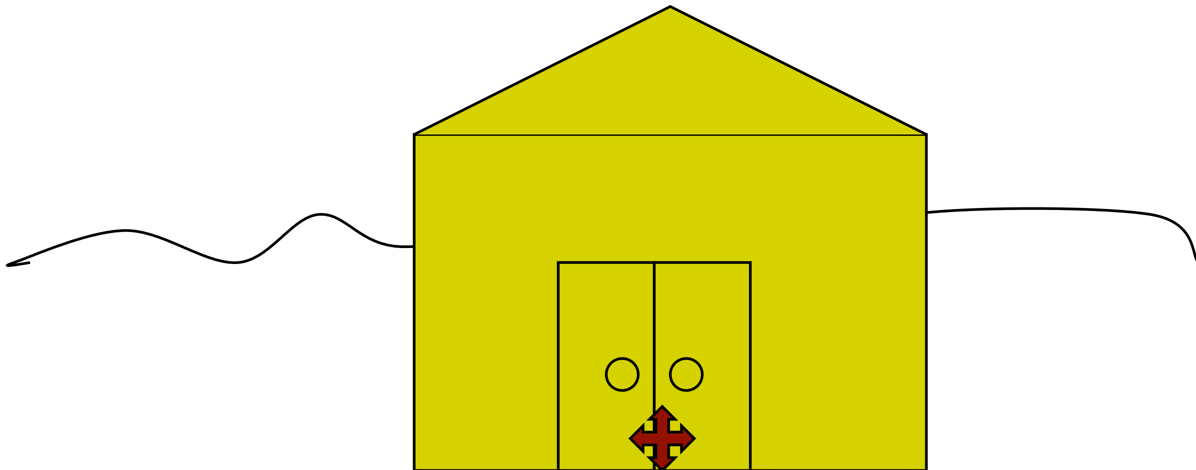
- Classic problem, what if multiple things picked?
  - Two types
  - Hierarchical ambiguity
    - are we picking the door knob, the door, the house, or the neighborhood?



Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Pick ambiguity

- Spatial ambiguity
  - Which door are we picking?



Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Solutions for pick ambiguity

- No "silver bullet", but two possible solutions
  - "Strong typing" (use dialog state)
    - Not all kinds of objects make sense to pick at a given time
      - Turn off "pickability" for unacceptable objects
        - reject pick during traversal

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Solutions for pick ambiguity

- Get the user involved
  - direct choice
    - typically slow and tedious
  - pick one, but let the user reject it and/or easily back out of it
    - often better
    - feedback is critical

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))

Georgia Tech

5

Copyright 2020 Blair MacIntyre ((CC BY-NC-SA 4.0))