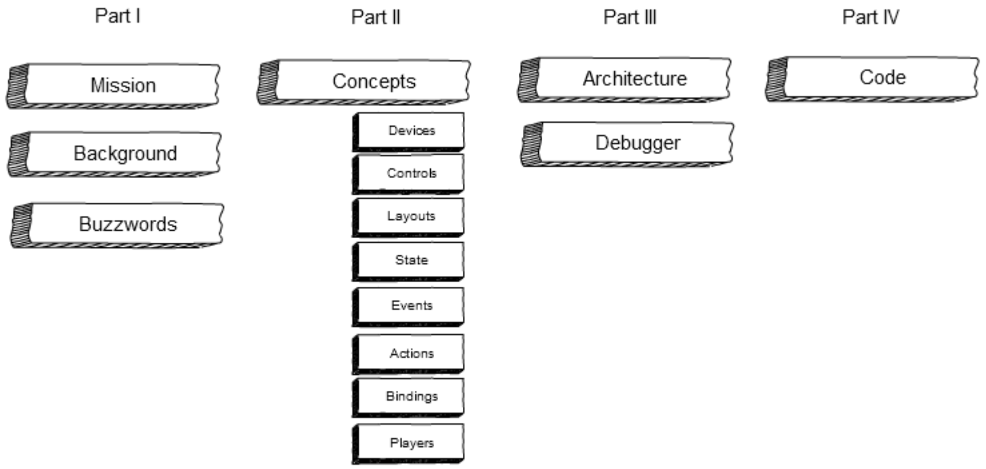
Input System

The Grand Tour

https://github.com/Unity-Technologies/InputSystem/Docs/Presentation/InputSystem-TheGrandTour.html

https://www.youtube.com/playlist?list=PLXbAKDQVwzta4J2Sbmjio2rTD6uO-phbR

Overview


Part I
Mission
Background
Buzzwords
Part II
Concepts
Devices
Controls
Layouts
State
Events
Actions
Bindings
Players
Part III
Architecture
Debugger
Part IV
Code

1. The Mission

Transmit, with *low overhead*,
input device activity from *all platforms*
through a *single API* catering to
both *low-level and high-level usage*.

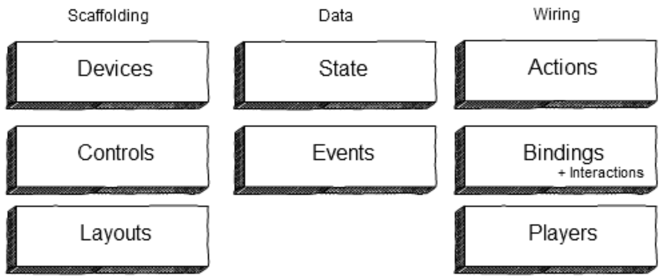
2. Background

- `InputManager`
  - 99% platform-dependent code
  - Diverging APIs: touch, XR, consoles
  - Non-extensible and largely inaccessible
- Push to user-land
  - Package

3: The Buzzwords

- *Game-oriented*
  - Frame-to-frame
- *Event-driven*
  - All input delivered in timestamped packets
- *Device-agnostic*
  - All devices represented the same way
- *Platform-agnostic*
  - All platforms go through the same API
- *Automation-ready*
  - Input can be fully driven from code
- *Cross-platform-consistent*
  - Same device, same input
- *User-extensible*
  - New devices and customizations

4: The Concepts



4.1: Concept - Devices

A. Devices

- Receive input
- Expose controls
- Execute commands ("IOCTL")
- Built from layouts
- Numeric ID
  - Unique for application run
- Can be assigned "usages"
  - `LeftHand`, `Player1`, `Horizontal`
- Can be added and removed by anyone
- Can be polled by anyone at any time

https://docs.unity3d.com/Packages/com.unity.inputsystem@1.1/manual/Devices.html

4.2: Concept - Controls

A. Devices

B. Controls

- Provide values
- Form hierarchies
  - `Gamepad1/leftStick/x`
- Built from layouts
- Immutable internal name (`"buttonSouth"`)
- Mutable external name (`"é"`)
- Can have "usages" (`"Submit"`)
- Can be monitored for changes
  - Think "data breakpoint"

https://docs.unity3d.com/Packages/com.unity.inputsystem@1.1/manual/Controls.html

4.3: Concept - Layouts

- Data that describes the setup of controls/devices
- Collection of control items that in turn may reference other layouts
- Three ways to build
  1. Reflection
  2. JSON
  3. `InputControlLayout.Builder`
- Can be "precompiled"
- Can be added and modified on the fly
- Can be overridden in full or in part
- Are matched to devices by pattern matching on the device description

https://docs.unity3d.com/Packages/com.unity.inputsystem@1.1/manual/Layouts.html

4.4: Concept - State

- Raw blob of unmanaged memory holding input state
- Memcpy'able, no heap references
- Multiple copies
  - old&new, player&editor
- All devices share one big block of memory
- Each device and each control corresponds to a slice of memory
- State is updated by copying contents of events on top of it
  - May be intercepted by device to deal with complications such as `Pointer.delta` and touchscreens
- Turned into processed values by `InputControl.ReadValue()`

4.5: Concept - Events

- Anyone can feed `InputEvents`
- ATM every event is associated with a device
- All events are raw blittable memory chunks
- Central buffer in native
  - One main-thread-only buffer
  - One threaded buffer for background threads (blocks when full)
- Buffer flushed on each input update
  - In full except for `FixedUpdates` for which we do timeslicing
- Various types
  - State events (`StateEvent`, `DeltaStateEvent`)
  - Disconnect event (`DeviceRemoveEvent`)
  - Configuration change event (`DeviceConfigurationEvent`)
  - Text input (`TextEvent`)

https://docs.unity3d.com/Packages/com.unity.inputsystem@1.1/manual/Events.html

4.6: Concept - Bindings

- Establish **input** channel from 1+ controls to an action
- Can be grouped into control schemes
- Controls are addressed using a "path language":

  ```
  <XRController>{LeftHand}/trigger
  ```

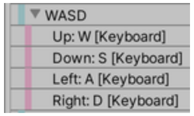
- Can apply processor stack to incoming values

  ```
  "invert,scale(factor=2)"
  ```

- Can apply "interactions"

  ```
  "multitap(tapCount=3)"
  ```

- Can use "composites" to source several bindings into one



WASD
Up: W [Keyboard]
Down: S [Keyboard]
Left: A [Keyboard]
Right: D [Keyboard]

https://docs.unity3d.com/Packages/com.unity.inputsystem@1.1/manual/ActionBindings.html

4.7: Concept - Actions

- Logical input: "jump", "move", "look"
- Can be on their own or grouped into maps
  - The latter can further be grouped into assets
- "Phased" interaction model
  - `Started`
  - `Performed`
  - `Canceled`
- Three types
  - Value
  - Button (press/release)
  - Pass-Through (input sink)
- Callback and polling APIs

https://docs.unity3d.com/Packages/com.unity.inputsystem@1.1/manual/Actions.html

4.8: Concept - Players

- Each player…
  - … is associated with 1+ devices
  - … may have an associated set of actions
  - … may have a current control scheme
  - … may be associated with a platform user account (consoles)
- Single-player: One player that can freely switch control schemes and devices
- Multiplayer: Multiple players each associated with a fixed set of devices

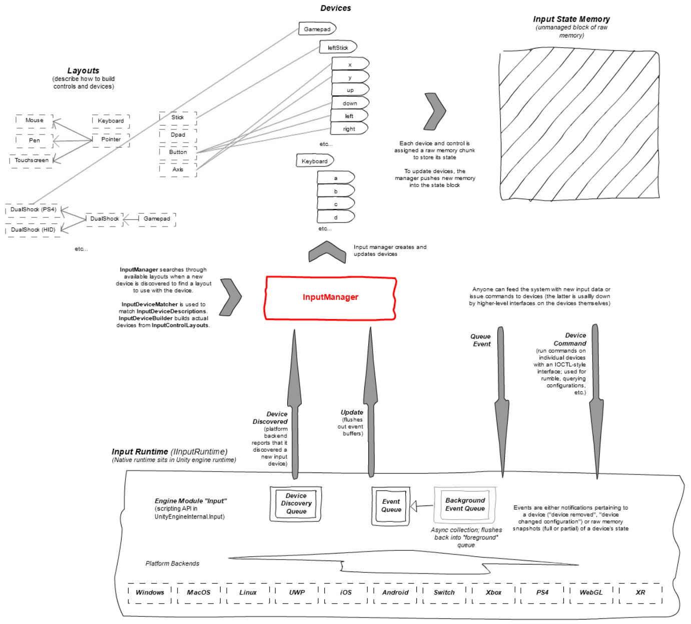
https://docs.unity3d.com/Packages/com.unity.inputsystem@1.1/manual/UserManagement.html

5: The Architecture

- Low-level: Events and devices
- High-level: Actions and players

https://docs.unity3d.com/Packages/com.unity.inputsystem@1.1/manual/Architecture.html

5.1: The Architecture - Low-Level



Each device and control is assigned a raw memory chunk to store its state

To update devices, the manager pushes new memory into the state block

Input manager creates and updates devices

InputManager searches through available layouts when a new device is discovered to find a layout to use with the device.

InputDeviceMatcher is used to match **InputDeviceDescriptions**. **InputDeviceBuilder** builds actual devices from **InputControlLayouts**.

Anyone can feed the system with new input data or issue commands to devices (the latter is usallly down by higher-level interfaces on the devices themselves)

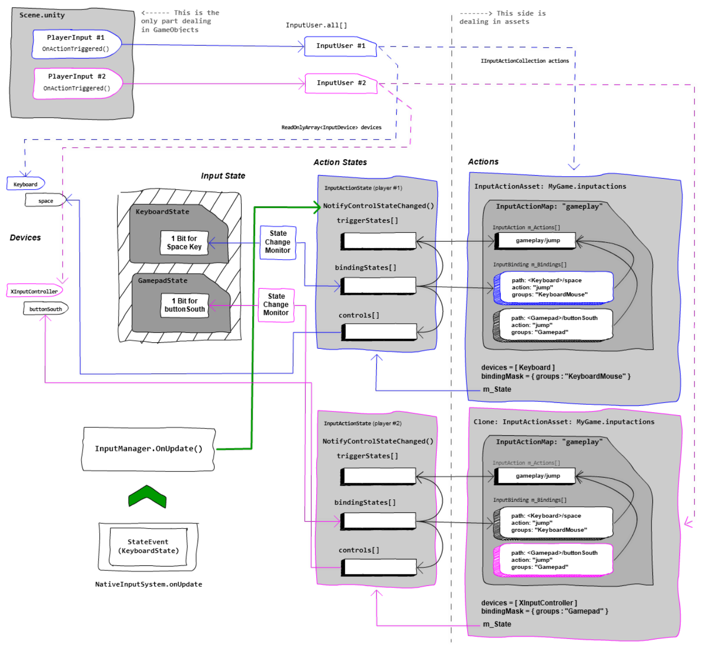
(run commands on individual devices with an IOCTL-style interface; used for rumble, querying configurations, etc.)

(platform backend reports that it discovered a new input device)

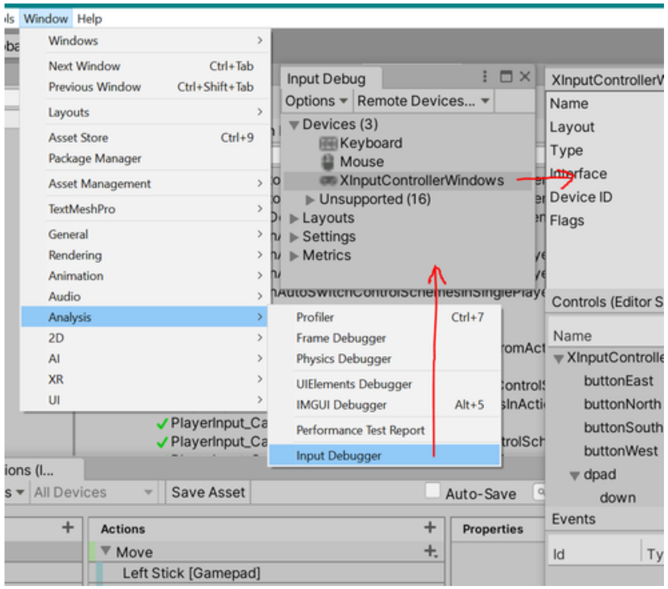
Async collection; flushes back into "foreground" queue

Events are either notifications pertaining to a device ("device removed", "device changed configuration") or raw memory snapshots (full or partial) of a device's state

5.2: The Architecture - High-Level

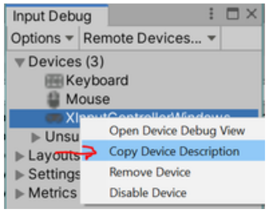


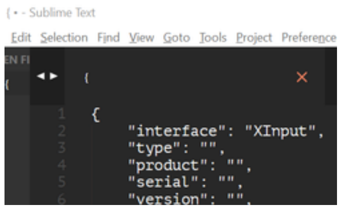
6: The Debugger



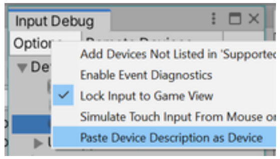
https://docs.unity3d.com/Packages/com.unity.inputsystem@1.1/manual/Debugging.html

6.1: The Debugger - Device Descriptions

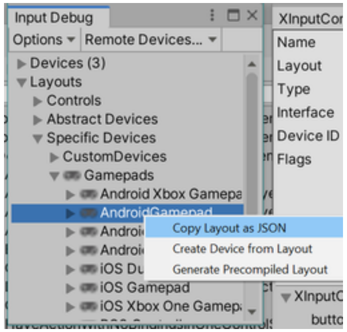




```
{
    "interface": "XInput",
    "type": "",
    "product": "",
    "serial": "",
    "version": "",
```

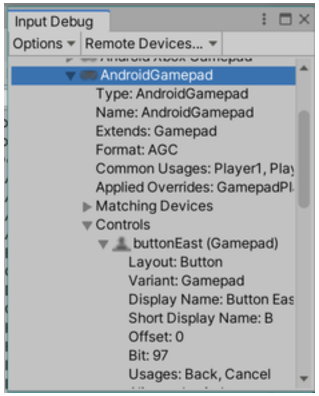

Input Debug
Option
De
Add Devices Not Listed in 'Supported
Enable Event Diagnostics
✓ Lock Input to Game View
Simulate Touch Input From Mouse or
Paste Device Description as Device

6.2: The Debugger - Layouts



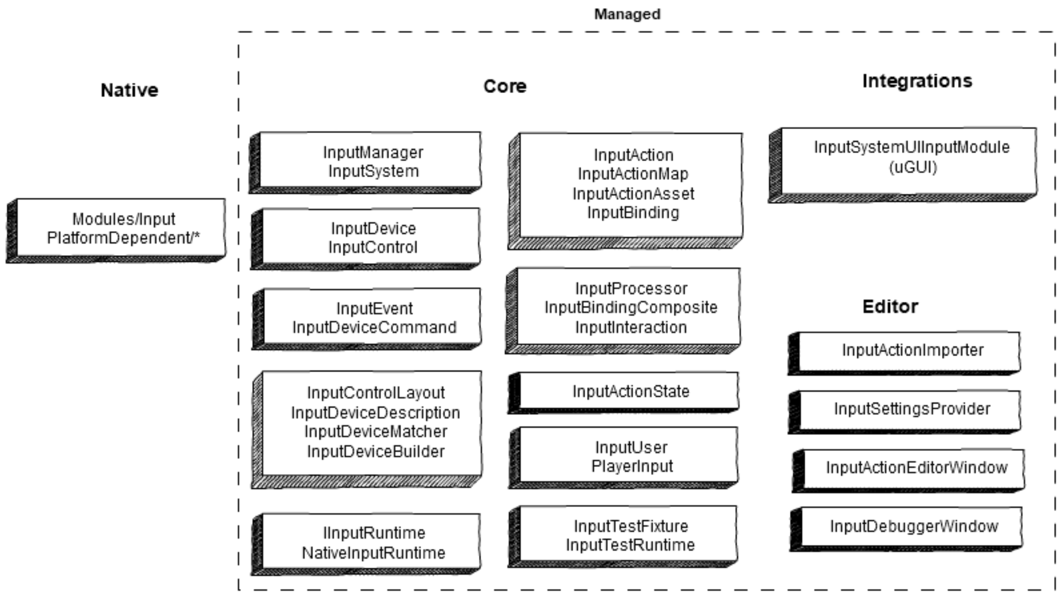
```
▶ Devices (3)
▼ Layouts
  ▶ Controls
  ▶ Abstract Devices
  ▼ Specific Devices
    ▶ CustomDevices
    ▼ Gamepads
      ▶ Android Xbox Gamepa
      ▶ AndroidGamepad
      ▶ Androi
      ▶ Androi
      ▶ iOS Du
      ▶ iOS Gamepad
      ▶ iOS Xbox One Gamep:
```

```
Name
Layout
Type
Interface
Device ID
Flags
▼ XInputC
   butto
```



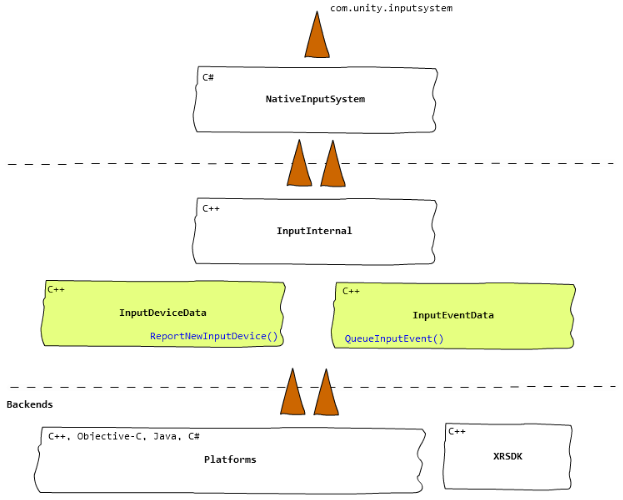
```
▼ AndroidGamepad
    Type: AndroidGamepad
    Name: AndroidGamepad
    Extends: Gamepad
    Format: AGC
    Common Usages: Player1, Play
    Applied Overrides: GamepadPl
  ▶ Matching Devices
  ▼ Controls
    ▼ buttonEast (Gamepad)
        Layout: Button
        Variant: Gamepad
        Display Name: Button Eas
        Short Display Name: B
        Offset: 0
        Bit: 97
        Usages: Back, Cancel
```

7: The Code


Managed
Native
Modules/Input PlatformDependent/*
Core
InputManager InputSystem
InputDevice InputControl
InputEvent InputDeviceCommand
InputControlLayout InputDeviceDescription InputDeviceMatcher InputDeviceBuilder
IInputRuntime NativeInputRuntime
InputAction InputActionMap InputActionAsset InputBinding
InputProcessor InputBindingComposite InputInteraction
InputActionState
InputUser PlayerInput
InputTestFixture InputTestRuntime
Integrations
InputSystemUIInputModule (uGUI)
Editor
InputActionImporter
InputSettingsProvider
InputActionEditorWindow
InputDebuggerWindow

7.1: The Code - Native

A. Modules/Input



7.2: The Code - Native

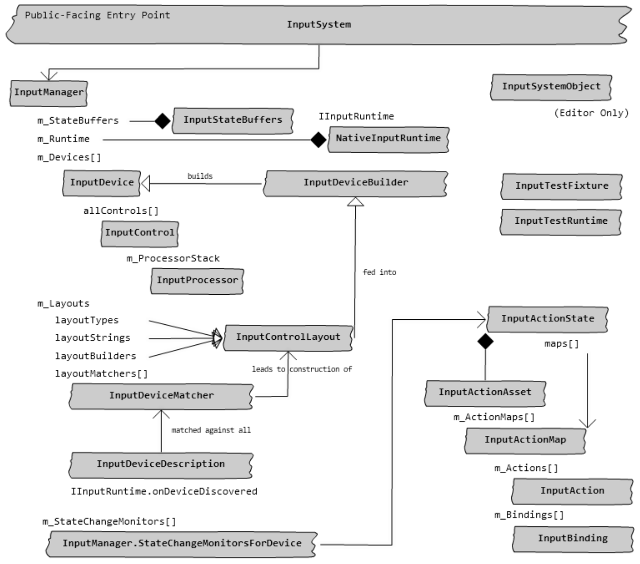
A. Modules/Input

B. PlatformDependent

- One backend for each platform we support
  - Win32, Mac, Linux, UWP, Android, iOS, WebGL, etc.
  - XR is orthogonal to this and feeds data in parallel from XRSDK subsystem
  - Future: Unity/XR Remote
- Some platforms run UI on thread separate from app/Unity thread
  - `QueueInputEvent()` may block and thus deadlock UI thread
  - Use of separate, platform-specific input buffer

7.3: The Code - Managed

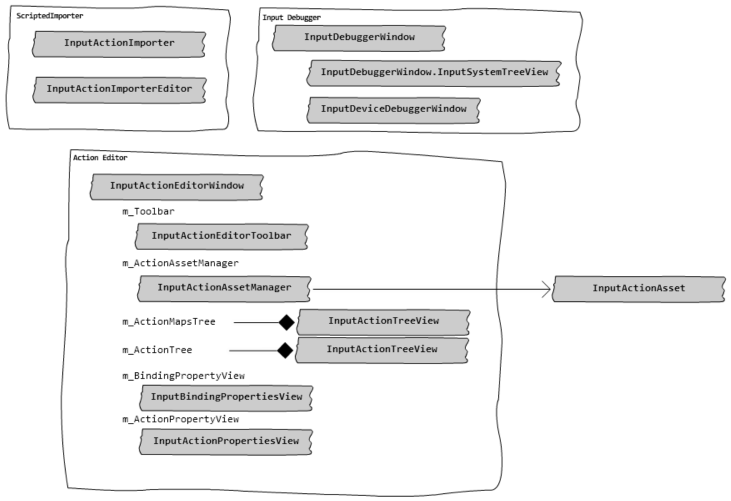
A. Core



7.4: The Code - Managed

A. Core

B. Editor



7.5: The Code - Managed

A. Core

B. Editor

C. "Plugins"

- Integrations
  - uGUI (`Plugins/UI`)
- Additions
  - `MonoBehaviour` wrapper: `Plugins/PlayerInput`
  - Touch polling: `Plugins/EnhancedTouch` (... wtf)
  - On-screen controls: `Plugins/OnScreen`
  - HID layout builder: `Plugins/HID`
- Platform/hardware-specific

8: The Problems

- Susceptible to event load
- System-wide buffer instead of per-device buffer
- Expensive `ReadValue()`
- Expensive layout system

9: Resources

- Introducing the new Input System - Unite Copenhagen 2019
- Custom Devices with Unity's Input System
- DOTS Input Prototype