User Input

Styles, Hardware, Unity

Tomáš Polášek ipolasek@fit.vutbr.cz

Brno University of Technology

Faculty of Information Technology
DCGM, CPhoto@FIT
Faculty of Fine Arts
Game Media Studio

5

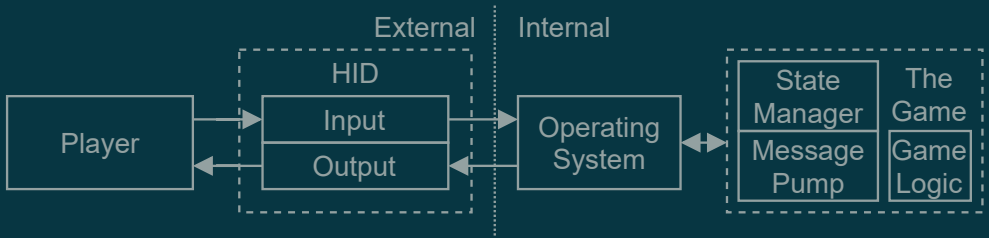
Human Interface Devices

Player Interaction

- HID = Human Interface Device
- Input & Output Interaction
- Input → Actuators
  - Movement
  - Sound
  - Motion
- Output → Senses
  - Sight
  - Hearing
  - Smell
  - Taste
  - Proprioception
  - Touch
- Control × Immersion

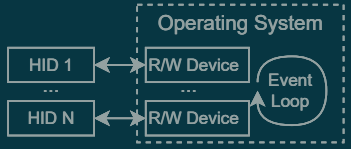
The Process

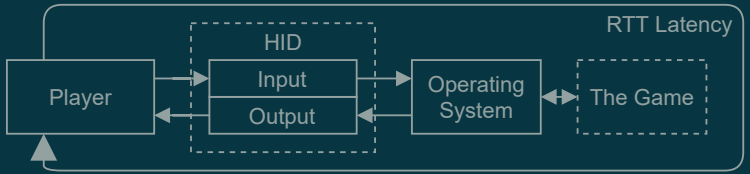
- **Player ↔ Game**
- Processing Chain
  1. Player Interaction
  2. HID Communication
  3. Game Behavior



HID Communication

- HID ↔ OS [2]
- The Channel: Wired × Wireless
- Communicating State
  - Polling
  - Interrupts
  - Packets
- I/O Latency = Round Trip Time





Player Inputs

Buttons

Positional

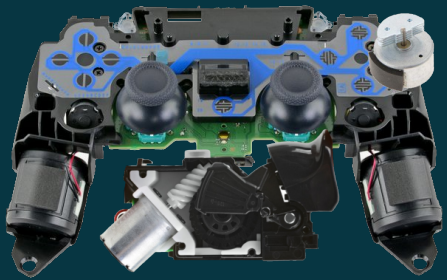
Player Inputs

Touch

Motion

Player Feedback

- Using Human **Senses**
- Visual (**Sight**) → Rendering
- Sound (**Hearing**) → Audio
- **Smell** & **Taste**
- Haptics (**Touch**):
  - Rumble
  - Force-Feedback
  - HD Haptics
  - HD Rumble
  - Haptic Bass
  - Resistance



Input Processing

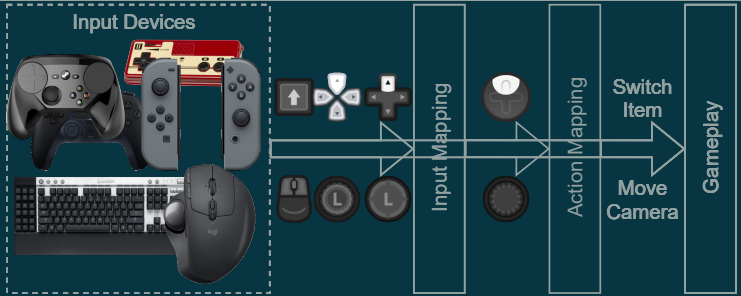
- Massaging **Raw Data** → **Gameplay**
- Getting **Data**: Polling × Messages
- Managing **State**
  - Simple Events
  - Composites & Modifiers
  - Sequences & Chords
  - Gestures
  - Value Axes
- Output API
- Dynamic Connections

```
void MainPoll()
{ while (mRunning) { InputProcessingPoll(); /* Update, Render, ... */ } }
void InputProcessingPoll()
{
    mNorthPressed = !mNorthDown && mGamePad.buttonNorth.isPressed;
    mNorthReleased = mNorthDown && !mGamePad.buttonNorth.isPressed;
    // ...
}

void MainMessages()
{ while (mRunning) { MessagePump(); /* Update, Render, ... */ } }
void MessagePump() { /* Reset Events, Serve Messages, ... */ }
void OnNorthPressed()
{ mNorthPressed = !mNorthDown; mNorthDown = true;}
void OnNorthReleased()
{ mNorthPressed = mNorthDown; mNorthDown = false;}
```

Taking Action

- Goal: **Inputs → Actions**
- Action Mapping
  - Simple 1 : 1
  - Input Mapping
  - Action Mapping
- Platform Agnostic & Specifics
- Mapping, Remapping, Binding
- Context-Sensitive Input




```
void DoJump() { /* Make the Player jump... */ }

void ActionSimple()
{ if (mGamePad.crossButton) { DoJump(); } }
void ActionSimpleMultipleDevices()
{ if (mGamePad.crossButton || ...) { DoJump(); } }

void ActionMultiPlatform()
{ if (mGamePad.buttonSouth) { DoJump(); } }

void ActionMapped()
{ if (mPlayerInput.action["Jump"]) { DoJump(); } }
```

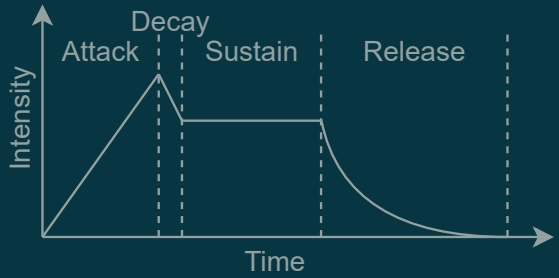
Interface Devices

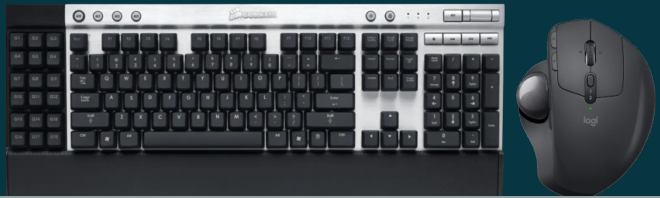
Input Hardware

Control Scheme

Character Controls

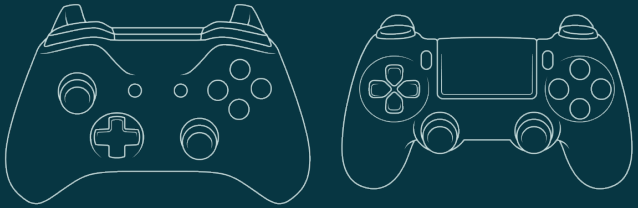
- **Learning** → Convention
- Conventional Control Schemes
  - Controller
  - Keyboard & Mouse
- Innovation × Tradition
- Humans are **Analog**
- Expectation → Filtering [1]
  - ADSR Curve
  - Control Assist





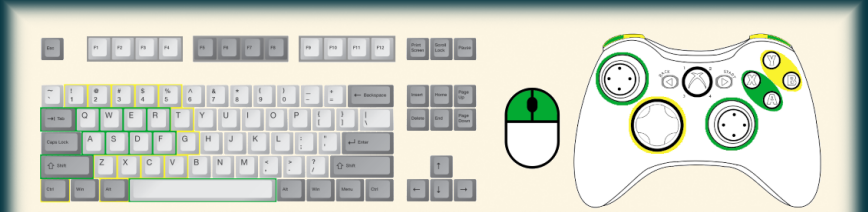
Feeling in Control

- HID = Means to an End
- Standard → Immersion
- Context Dependent
- Case Study: Camera Styles
  - Orbit
  - Top-Down
  - 1st Person
  - 3rd Person



Design Considerations

- 10 **Fingers**, 2 **Hands**, 1 **Head**
- Grouping & Neutral Position
- Accessibility Tiers [1]
- Special Circumstances
  - Casual Games
  - Disabilities
  - Virtual / Augmented Reality



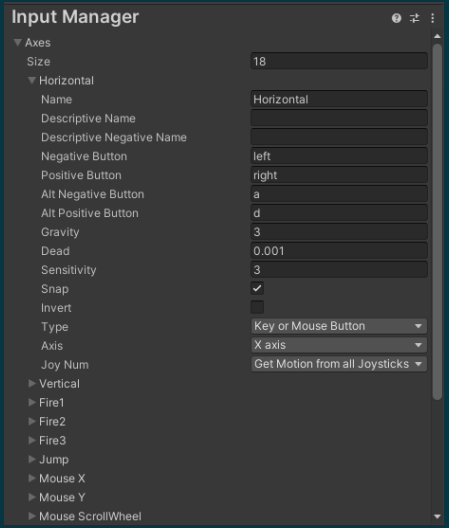
Source: Andrew Dotsenko's Game Design Framework

Input in Unity

Input Module

- Default **Input** Method
- “The Old Way”
- Stateless & Axis-Based
- Input Manager

```
void ProcessInput()
{
    if (Input.GetKeyDown(KeyCode.W)) { MoveForward(); }
    if (Input.GetButtonDown("Jump")) { DoJump(); }
    if (Input.GetAxis("Horizontal") > 0.0) { MoveCamera(); }
}
```

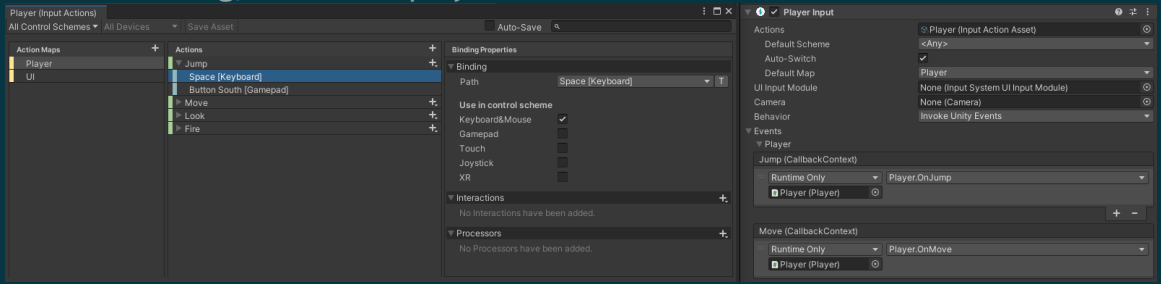


Input System

- **Input System** Package
- “The New Way”
- Stateless & Action-Based
- Mapping, Bindings, Virtual
- Higher Complexity
- Rebinding, Local Multiplayer

```
public void OnMove(InputAction.CallbackContext ctx)
{ DoMove( ctx.ReadValue<Vector2>()); }

public void OnJump(InputAction.CallbackContext ctx)
{ if (ctx.started) { DoJump(); } }
```



Debug and Profiling


CPU Usage
Rendering
Scripts
Physics
Animation
GarbageCollector
VSync
Global Illumination
UI
Others
33ms (30FPS)
16ms (60FPS)
Rendering
Batches Count
SetPass Calls Count
Triangles Count
Vertices Count
Memory
Total Used Memory
Texture Memory
Mesh Memory
Material Count
Object Count
GC Used Memory
GC Allocated In Frame
Audio
Playing Audio Sources
Audio Voices
Total Audio CPU
Total Audio Memory
Video
Total Video Sources
Playing Video Sources
Pre-buffered frames
Total Video Memory
Physics
Physics Used Memory
Active Dynamic Bodies
Active Kinematic Bodies
Dynamic Bodies
Active Constraints
Overlaps
Trigger Overlaps
Discreet Overlaps
Continuous Overlaps
Physics Queries
Timeline
Live
CPU:1,811.35ms GPU:--ms
Main Thread
Render Thread
Job
Loading
Scripting Threads
Background Job
Profiler
Other Threads
Audio
AssetDatabase

Additional Resources

- [YouTube] Evolution of Video Game Controllers
- [Article] John Harris: 20 Unusual Control Schemes
- [Product] Olorama: Smells in Virtual Reality
- [Article] Andrew Dotsenko: Designing Game Controls

Source: Designing Game Controls


Thanks For
Your Attention!
Baldur's Gate
Minsc: Minsc and Boo stand ready.
111/111
108/108
95/95
86/86
45/45
49/49

References I

[1] Andrew Dotsenko. ***Designing Game Controls.*** https://gamedesignframework.net/designing-game-controls/. 2018.

[2] Jason Gregory. ***Game Engine Architecture, Second Edition.*** 3rd. USA: A. K. Peters, Ltd., CRC Press, 2018. ISBN: 1351974288.