HUMAN-SYSTEMS INTERACTIONS

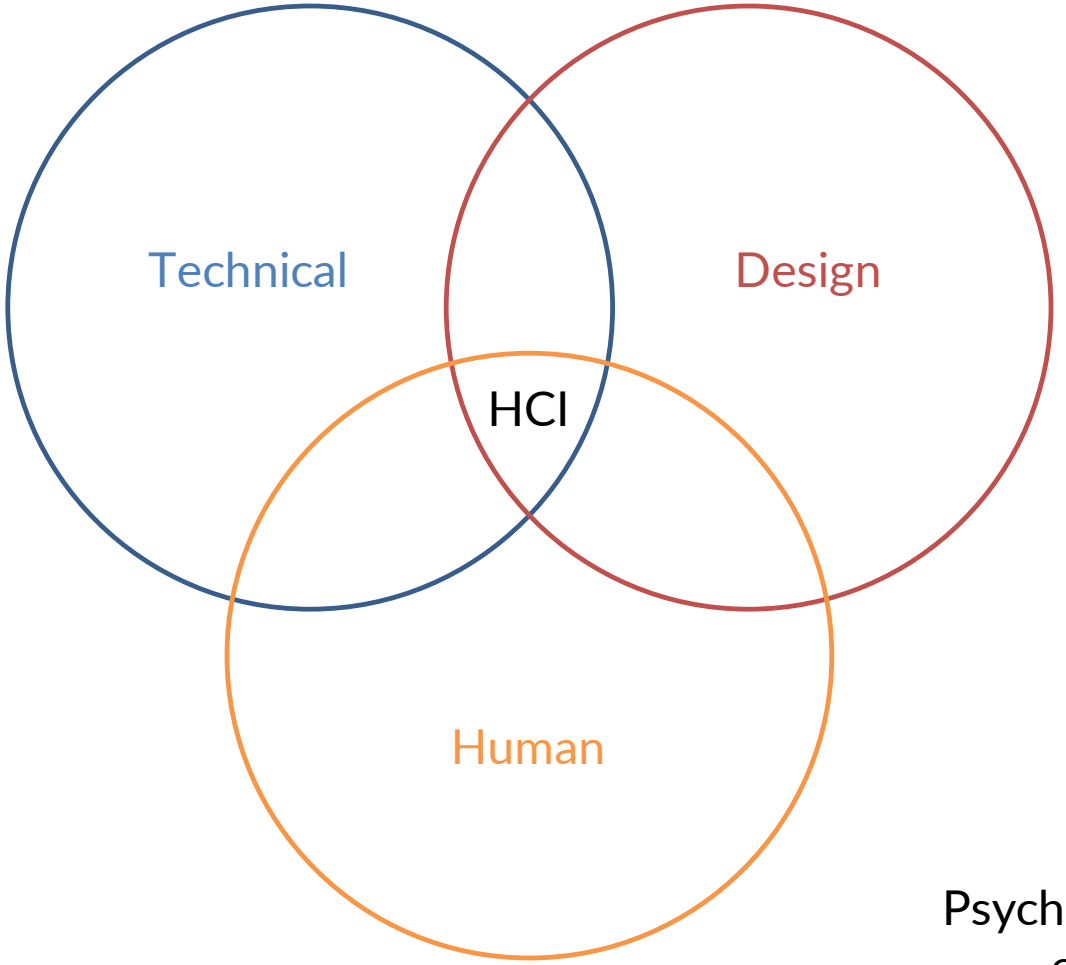
V2.0

guillaume.bouyer@ensiie.fr

www.ensiie.fr/~bouyer/

Human-Computer Interactions

Interfaces, software, implementation…



UX, UI, evaluation…

Psychology, physiology, ergonomics…

Objectives

Know how to design, program and evaluate HCIs for games

Practice in development (Unity), in search for information/inspiration, in taking into account the needs/context of the user and the ideas/constraints of the designer

Concepts also applicable in VR&AR and PFE

Methodology and planning:

- See project description

bouyer@ensiie.fr

http://www.ensiie.fr/~bouyer/JIN

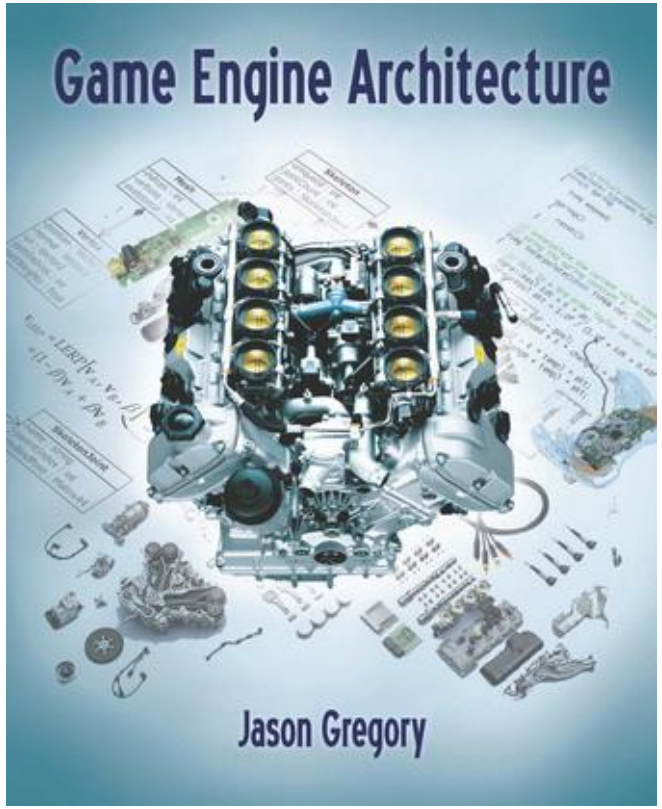
Office 111 @ ENSIIE

PART 1

HUMAN-GAMES INTERFACING

Main Reference

Game Engine Architecture, Jason Gregory, A K Peters/CRC Press, 2009 (http://www.gameenginebook.com/)



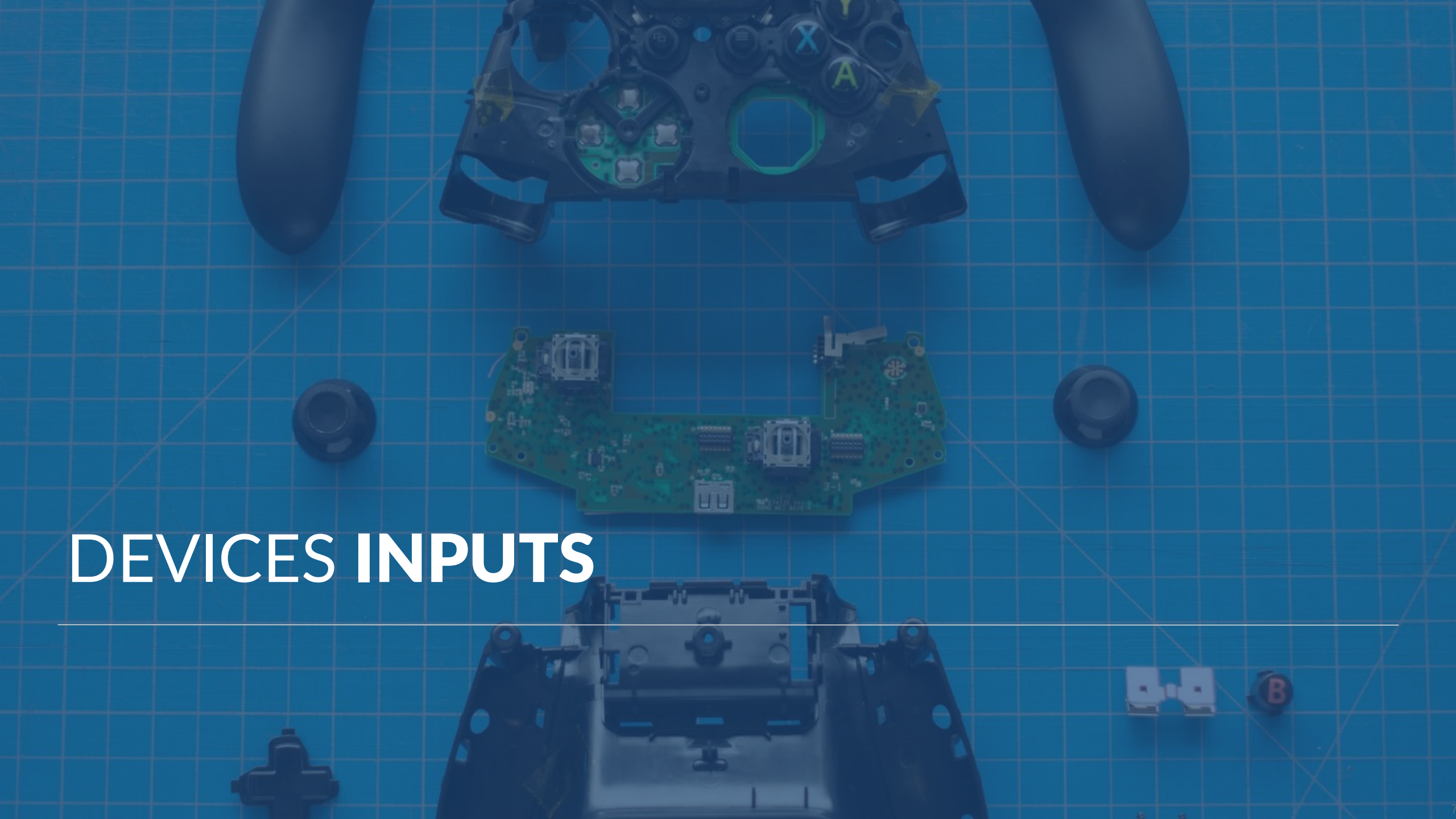
Types of devices

- Keyboard & mouse
- Joypad controller
- Gesture controller
  - Wiimote, Kinect, PS Move, touch surface
- Hybrid controller
  - Wii U, PS Vita, 3DS
- Built-in controllers
  - Arcade machines
- Specialized input devices and adapters
  - Music devices
  - Steering wheels
  - Dance pad

Various form factors and layouts

Common input types

Some kinds of outputs



DEVICES **INPUTS**

Digital Buttons

2 states:

pressed = down

not pressed = up

(cf. closed or open electrical flow)

1 button usually represented by a bit

0 = up

1 = down

→ states of all of the buttons on a device packed into a single unsigned integer value

Digital Buttons: Microsoft ’s XInput API

Struct contains a variable wButtons that holds the state of all buttons

```
typedef struct _XINPUT_GAMEPAD {
    // 16-bit unsigned integer
    WORD wButtons;
    // 8-bit unsigned integer
    BYTE bLeftTrigger;
    BYTE bRightTrigger;
    // 16-bit signed integer
    SHORT sThumbLX;
    SHORT sThumbLY;
    SHORT sThumbRX;
    SHORT sThumbRY;
} XINPUT_GAMEPAD;
```

Bit mask defines which physical button corresponds to each bit in the word

```
#define XINPUT_GAMEPAD_DPAD_UP      0x0001 // bit 0
#define XINPUT_GAMEPAD_DPAD_DOWN    0x0002 // bit 1
#define XINPUT_GAMEPAD_DPAD_LEFT    0x0004 // bit 2
#define XINPUT_GAMEPAD_DPAD_RIGHT   0x0008 // bit 3
#define XINPUT_GAMEPAD_START        0x0010 // bit 4
#define XINPUT_GAMEPAD_BACK         0x0020 // bit 5
#define XINPUT_GAMEPAD_LEFT_THUMB   0x0040 // bit 6
#define XINPUT_GAMEPAD_RIGHT_THUMB  0x0080 // bit 7
#define XINPUT_GAMEPAD_LEFT_SHOULDER  0x0100 // bit 8
#define XINPUT_GAMEPAD_RIGHT_SHOULDER  0x0200 // bit 9
#define XINPUT_GAMEPAD_A          0x1000 // bit 12
#define XINPUT_GAMEPAD_B          0x2000 // bit 13
#define XINPUT_GAMEPAD_X          0x4000 // bit 14
#define XINPUT_GAMEPAD_Y          0x8000 // bit 15
```

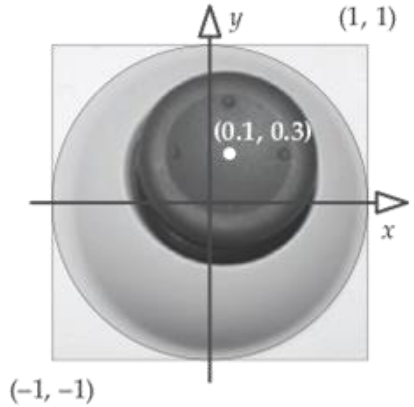
An individual button’s state can be read by masking the wButtons word with the appropriate bit mask

```
bool IsButtonADown(const XINPUT_GAMEPAD& pad){
    // Mask off all bits but bit 12 (the A button).
    return ((pad.wButtons & XINPUT_GAMEPAD_A) != 0);
}
```

Analog Axes and Buttons

Axes

- Range of values
- Used to represent the degree to which a trigger is pressed, or the 2D position of a joystick (two analog inputs x and y)



Buttons

- *Ex : Metal Gear Solid 2*
- Signals produced usually too noisy to be usable

Input signal usually digitized using integer or float

Analog Axes and Buttons: Microsoft ’s XInput API

16-bit signed integers for left and right thumb sticks

[-32768, 32767]

8-bit unsigned integers for left and right shoulder triggers

[0 , 255]

```
typedef struct _XINPUT_GAMEPAD {
    // 16-bit unsigned integer
    WORD wButtons;
    // 8-bit unsigned integer
    BYTE bLeftTrigger;
    BYTE bRightTrigger;
    // 16-bit signed integer
    SHORT sThumbLX;
    SHORT sThumbLY;
    SHORT sThumbRX;
    SHORT sThumbRY;
} XINPUT_GAMEPAD;
```

Relative Axes

- The position of an analog button, trigger, joystick, or thumb stick is absolute
  - Clear zero value
- For relative devices
  - No clear location at which the input value should be zero
  - Zero input value = the position of the device has not changed
  - Non-zero input values = delta value from last time
  - *Ex: mice, mouse wheels, track balls…*

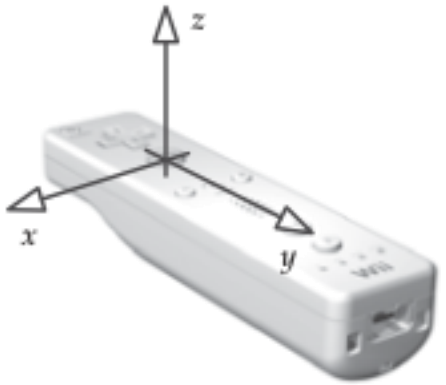
3D Orientation

3 accelerometers to estimate the orientation of the controller

Measure the acceleration along each axis based on constant downward gravity

- Quantized into three signed 8-bit integers (x, y, and z)

Ex: PS3 Sixaxis and DualShock 3, Nintendo WiiMote



Other Inputs

3D Position

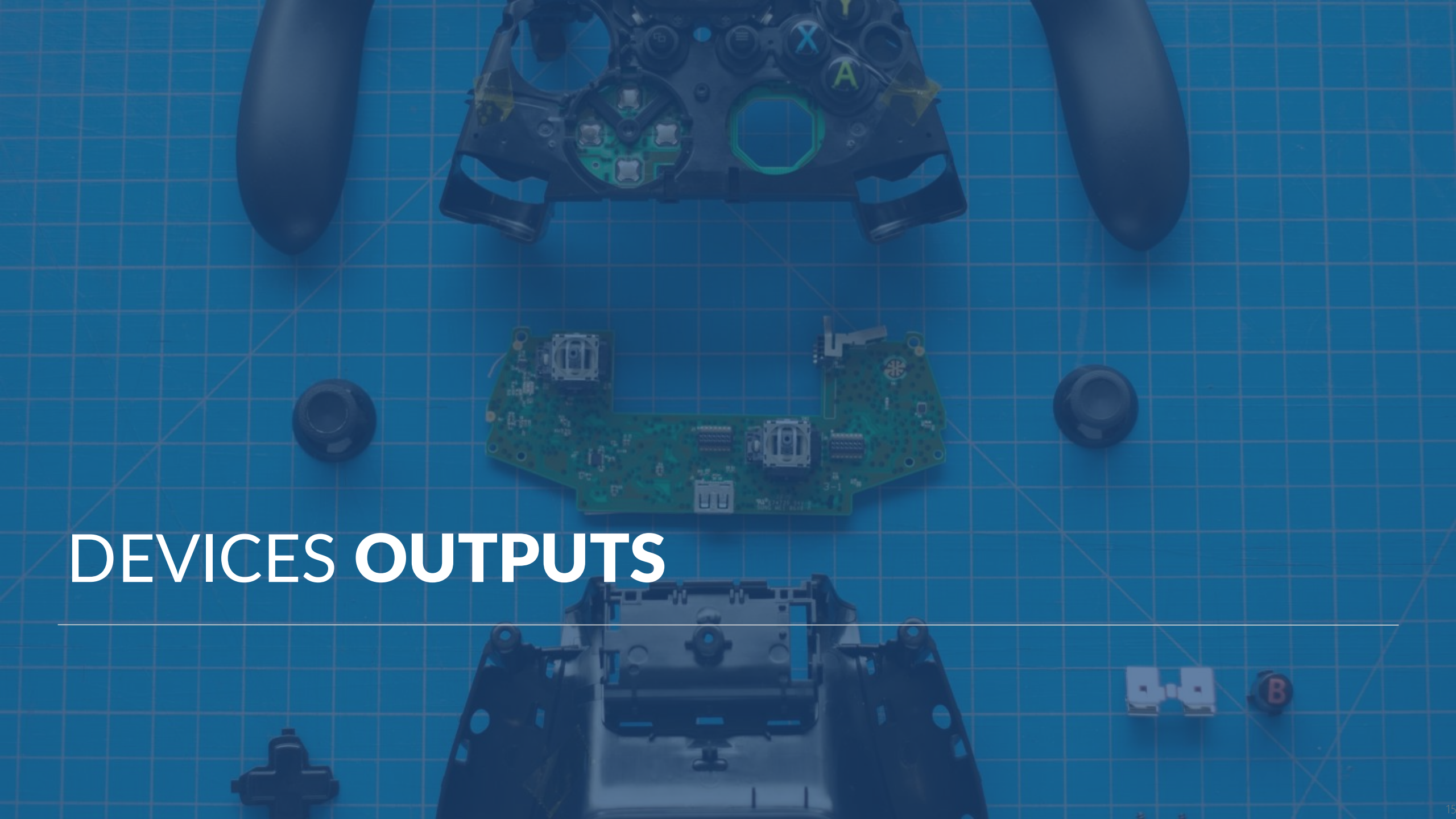
- IR Camera (Wiimote)
  - Location, size and distance of 2 fixed IR LEDs
- Camera (EyeToy)

Gestures (Kinect)

- Computer vision-based techniques (ex. structured light) -> depth map
- Infer body position: skeleton, joints
- Bodily movements, poses

Touch

- Single or multiple contact points
- Movements



DEVICES **OUTPUTS**

Rumble & Force-Feedback

Vibrations

One or more motors rotating a slightly unbalanced weight at various speeds

Game can control:

- motors on/off
- speeds -> different tactile effects

Ex. PS2, PS3, Xbox 360, Wii controllers

Force

Actuator(s) driven by a motor

Resist the motion of the player

Game can control:

- motors on/off
- strength and direction of the forces

Ex. arcade driving games: steering wheel resists the player's attempt to turn it, simulating difficult driving conditions or tight turns

Other Outputs

- Audio
  - Small speaker
  - Embedded USB audio I/O device
- Memory card slot on the pad (Dreamcast)
- LEDs
- Specific outputs for specific controller (music instruments…)

INTERFACING WITH A **DEVICE**

Interfacing

Game software reads devices inputs and writes outputs in various ways, depending on the specific design of the device

- Polling
- Interrupts
- Wireless

Polling

- Input = explicitly and periodically querying the state of the device
  - Usually once per iteration of the main game loop
  - By reading hardware registers directly
  - By reading a memory-mapped I/O port
  - Via a higher-level software interface
- Outputs = writing
  - Special registers, memory-mapped I/O addresses, via a higher-level API

Polling: Microsoft ’s XInput API

- **`XInputGetState()`** every frame
- Communicates with the hardware and/or drivers
- Reads the data in the appropriate way
- Packages it
- Returns pointer to an **`XINPUT_STATE`** struct
  - contains an embedded instance of an **`XINPUT_GAMEPAD`** struct

Interrupts

Controller sends data to the game engine when changing state via hardware interrupts

- Electronic signal causes the CPU to temporarily suspend execution of the main program
- Runs a routine which reads the state of the device, stores it off for later processing, and relinquish the CPU back to the main program
- The game engine can pick up the data at any time

Ex: mouse when it moves, or button pressed/released

Wireless Devices

I/O can't be read and written by simply accessing registers or memory-mapped I/O ports

Bluetooth protocol:

- Request the device to send input data back to the host
- Send output data to the device
- Handled by a thread separate from main loop, or encapsulated behind an interface that can be called from the main loop

Ex: WiiMote, DualShock 3, Xbox 360 wireless controller

GAME ENGINE **INTERFACE SYSTEMS**

Game Engine Interface Systems

- Inputs processing
  - From raw data to smooth, pleasing, intuitive behaviors in-game
  - Input management features
- Level of abstraction
  - Decouple raw inputs & logical game actions
  - Ex. button-mapping table

Dead Zone

Analog axis produce input values that range between predefined values I_{min} and I_{max}

Ideal: Control not touched -> should produce a steady and clear "undisturbed" value I_0:

$I_0 = 0$

$I_0 = (I_{min} + I_{max})/2$ or $I_0 = I_{min}$

In practice:

noisy voltage produced by the device

(no) inputs may fluctuate slightly around I_0

A dead zone might be defined as

Any input values within the dead zone are clamped to I_0

$[I_0 - \delta, I_0 + \delta]$ for a joystick

$[I_0, I_0 + \delta]$ for a trigger

Wide enough to account for the noisiest inputs generated by an undisturbed control

Small enough not to interfere with the player's sense of the HID's responsiveness

Analog Signal Filtering

A noise signal is usually of a high-frequency, relative to the signal produced by the player

Discrete 1st order low-pass filter

Combine the current unfiltered input value with last frame's filtered input

$$f(t) = (1 - a)\, f(t - \Delta t) + a.u(t)$$

$$with\ a = \frac{\Delta t}{RC + \Delta t}\ \ and\ RC\ constant$$

Average on n frames

Store the input data in a n-element circular buffer

Detecting Button Up and Down

Bit-wise operators to compare buttons' state bits between frames

- Previous XOR Current => 1 for changed buttons
- Result AND Current => button-down event
- Result AND NOT Current => button-up event

Detecting Chords

Group of buttons pressed at the same time

- Watch the states of the buttons
- Perform the requested operation when all of them are down

Problem 1: Ambiguities if the chord includes a button assigned to an action

- Do to perform both
- Check that the other chord buttons are not down when detecting the individual button-presses

Detecting Chords

Problem 2: Humans rarely press buttons in the same frame

- Game design such that a chord does all the actions of the individual buttons + additional action
- Delay between individual button-down event detection and valid game event
  - During the delay period, a detected chord prevails over the individual button-down events
- Detect the chord when buttons pressed, but trigger effect when released
- Begin the single-button move immediately and preempt it by the chord

Gesture

Sequence of actions performed over a period of time

Only valid if performed within maximum time-frame

Implemented via a history of actions

- 1st detected component stored in the history buffer, with a time stamp
- Each subsequent detected component -> time difference
  - If within the allowable time-frame: added to the history buffer
- If the entire sequence is completed within the time-frame (i.e. history buffer filled) -> event generated
- If any non-valid intervening inputs detected, or if any component outside valid time window -> history buffer reset

Robustness

- Managing multiple devices for multiple players
  - One-to-one mapping between controller index and player index
  - Assigning controllers to players when start
- Detecting low-battery conditions
  - Game or OS
  - Unobtrusive on-screen message and/or a sound effect
- Lost connection
  - *Ex: controller being unplugged or running out of batteries*
  - Usually pause gameplay, display a message, and wait for the controller to be reconnected
  - Multiplayer games
    - Suspend or temporarily remove the avatar corresponding to a removed controller
    - Allow the other players to continue playing the game
    - The removed/suspended avatar might reactivate when the controller is reconnected

Cross-Platform

Conditional compilation directives in the code, wherever device interactions

Hardware abstraction layer

- Abstract button and axis ids -> `enum AbstractControlIndex`
- Translate between raw control ids on the current target hardware and abstract control ids
  - Ex.: bit-swizzling operation to rearrange the bits of the buttons 32-bits word, separate the only trigger axis into two distinct L&R axes on the Xbox, scaling the values appropriately so the range of valid values is the same on all platforms…

Functional approach

- Naming the abstract controls according to their function in the game, rather than their physical locations on the joypad
- Introduce higher-level functions that detect abstract gestures, with custom detection code on each platform

Platform-specific versions of all of the game code that requires device I/O

Input Re-Mapping

Many games allow to choose the controls

Ex: sense of the vertical axis for camera, predefined button mappings, full control of individual keys

Level of indirection

Table to maps each control index to a logical function via a unique id

Different controls produce different kinds of input data: only permit logic control mappings

Normalize all the inputs and group into classes

Digital buttons: states packed into a 32-bit word, one bit per button

Unidirectional absolute axes: produce floating-point input values in [0, 1]

Bidirectional absolute axes: produce floating-point input values in [−1, 1]

Relative axes: produce floating-point input values in [−1, 1]

Context-Sensitive Controls

A single physical control can have different functions depending on context

- *“Use” button -> open, pick up…*
- *Modal control -> navigate and control camera, steer a vehicle…*

Problem: how to decide the state given the context

- *Ex: equidistance between 2 items*

State machine, priority system…

Lots of trial-and-error

Control Ownership

Controls might be "owned" by different parts of the game

- *Player control*
- *Camera control*
- *Menu system (pausing…)*

"Logical devices"

- Subsets of the inputs on the physical device
- Each assigned to a system (camera, player…)

Disabling Inputs

Ex.:

Disable all player controls during cinematic

Disable free camera rotation when walking through a doorway

Use a bit mask to disable individual controls on the input device

When needed, neutral or zero value returned instead of the actual value read

Put the logic in the player or camera code itself

Interfacing in practice

Interactions are the basis of the player mechanics: correct and smooth handling is an important part of any good game

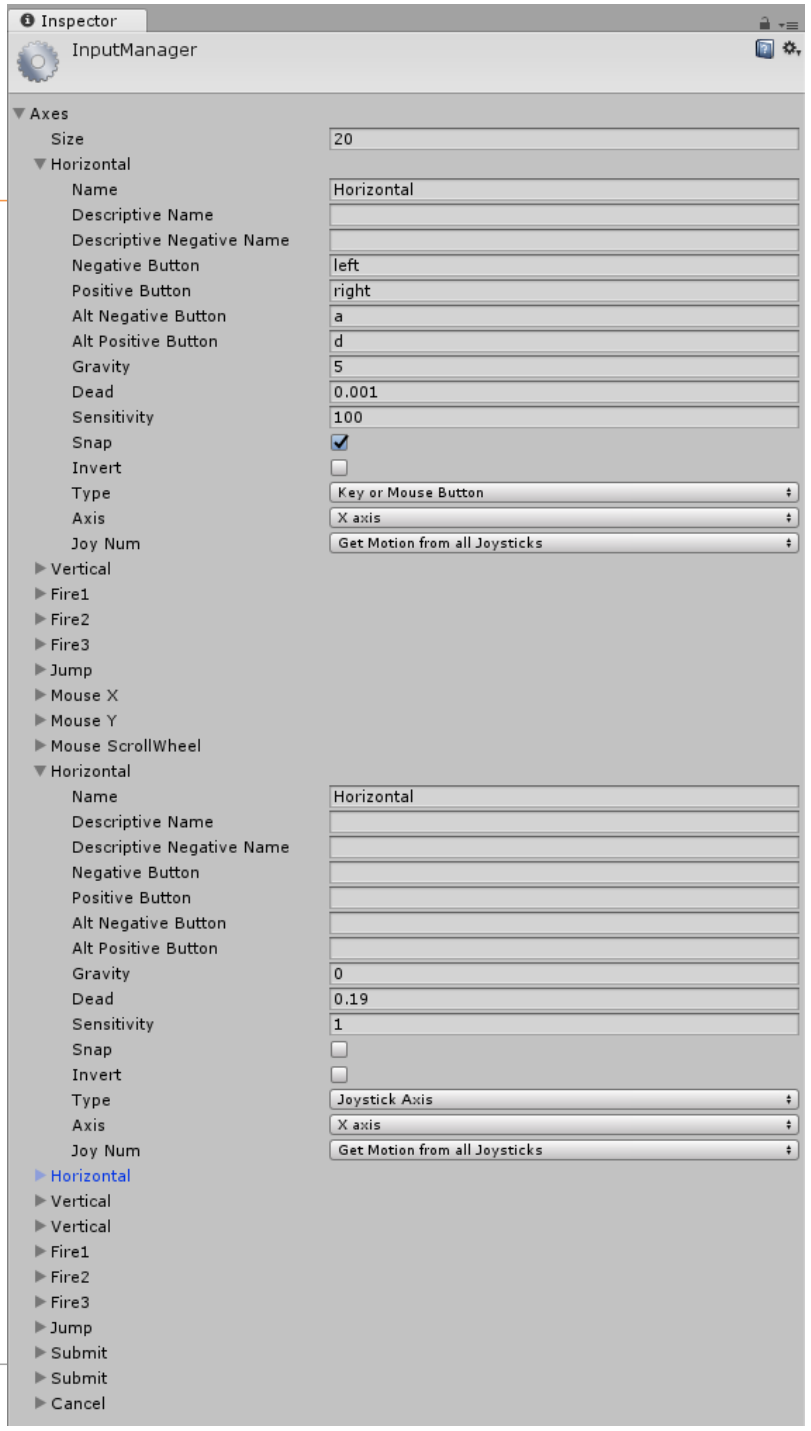
Deal with

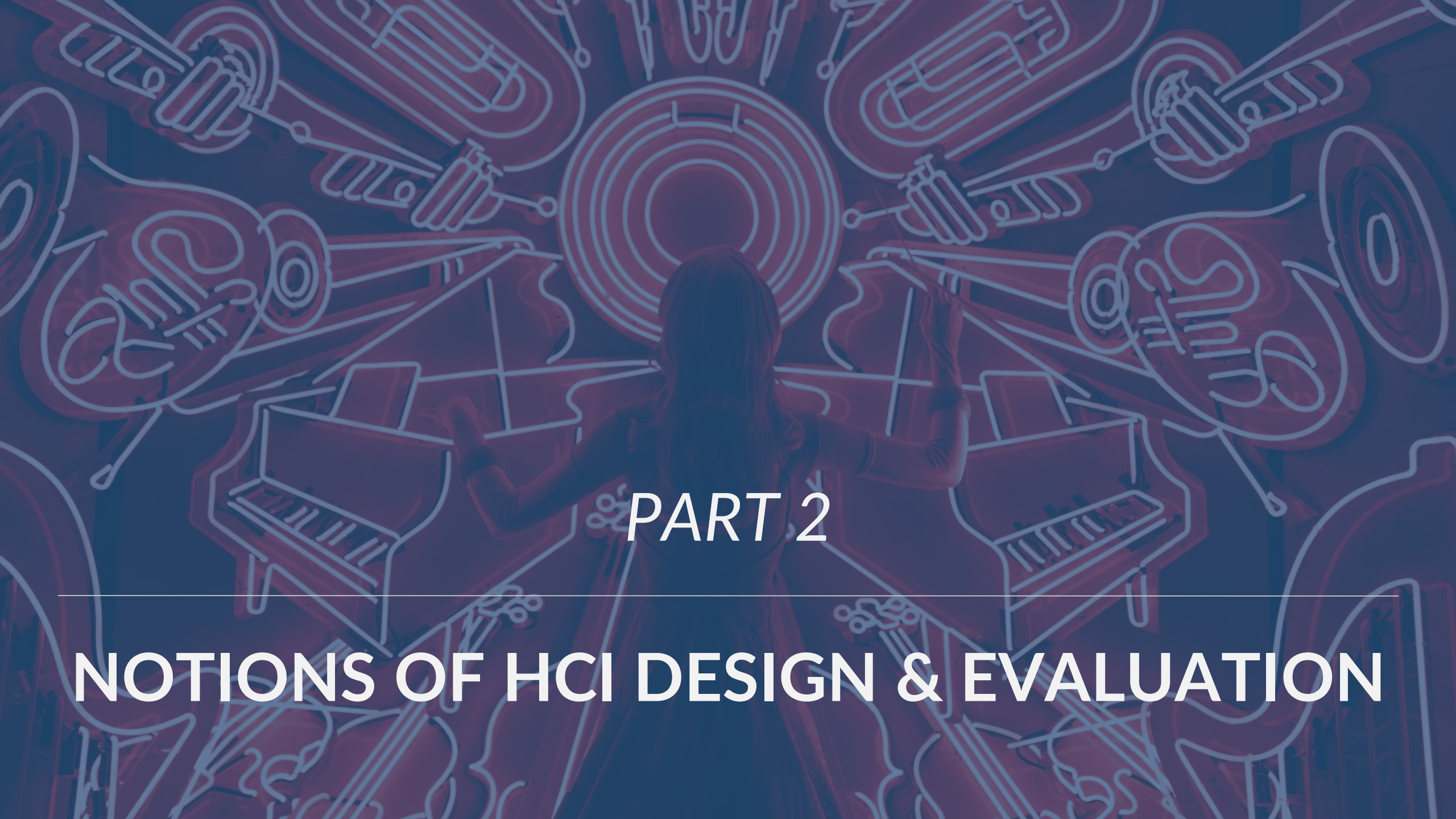
- Variations between different input devices
- Filtering
- Command mappings
- Achieving the right “feel”
- Limitations from manufacturers (technical requirements checklists TRCs)

=> Devote significant time and engineering to a careful and complete implementation of the interface system

Unity

- Input
  - http://docs.unity3d.com/ScriptReference/Input.html
- Input Manager
  - Custom axis and buttons, dead zone, gravity, sensitivity, key binding…
- Time





PART 2

NOTIONS OF HCI DESIGN & EVALUATION

DEFINITIONS

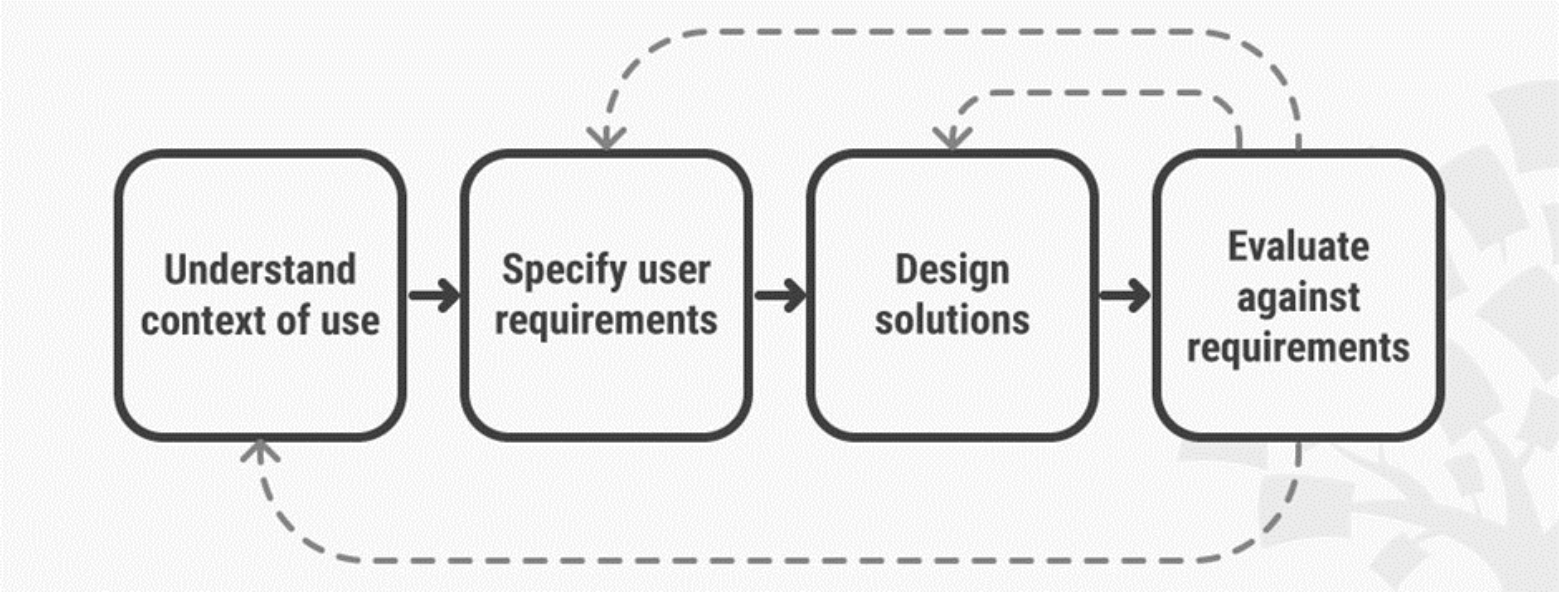
User-Centered Design

Norman & Draper, 80's

Iterative design process in which designers focus early on the **users, their needs,** their tasks and their environment in each phase of the design process

Active participation of users

Iteration of solutions, until the needs and requirements expressed by users are fulfilled



User Experience (UX)

Norman, Miller & Henderson (95)

- “user interface” and “usability” were too restricted
- wanted to cover all aspects of a person's experience with a system, including industrial design, graphic elements, interface, physical interaction and instructions for use

Popularized by Merhloz (98) et Garrett (02), esp. for web design

“User experience encompasses all aspects of the end-user's interaction with the company, its services, and its products.”

Don Norman and Jakob Nielsen

User Experience (UX)

Requirements

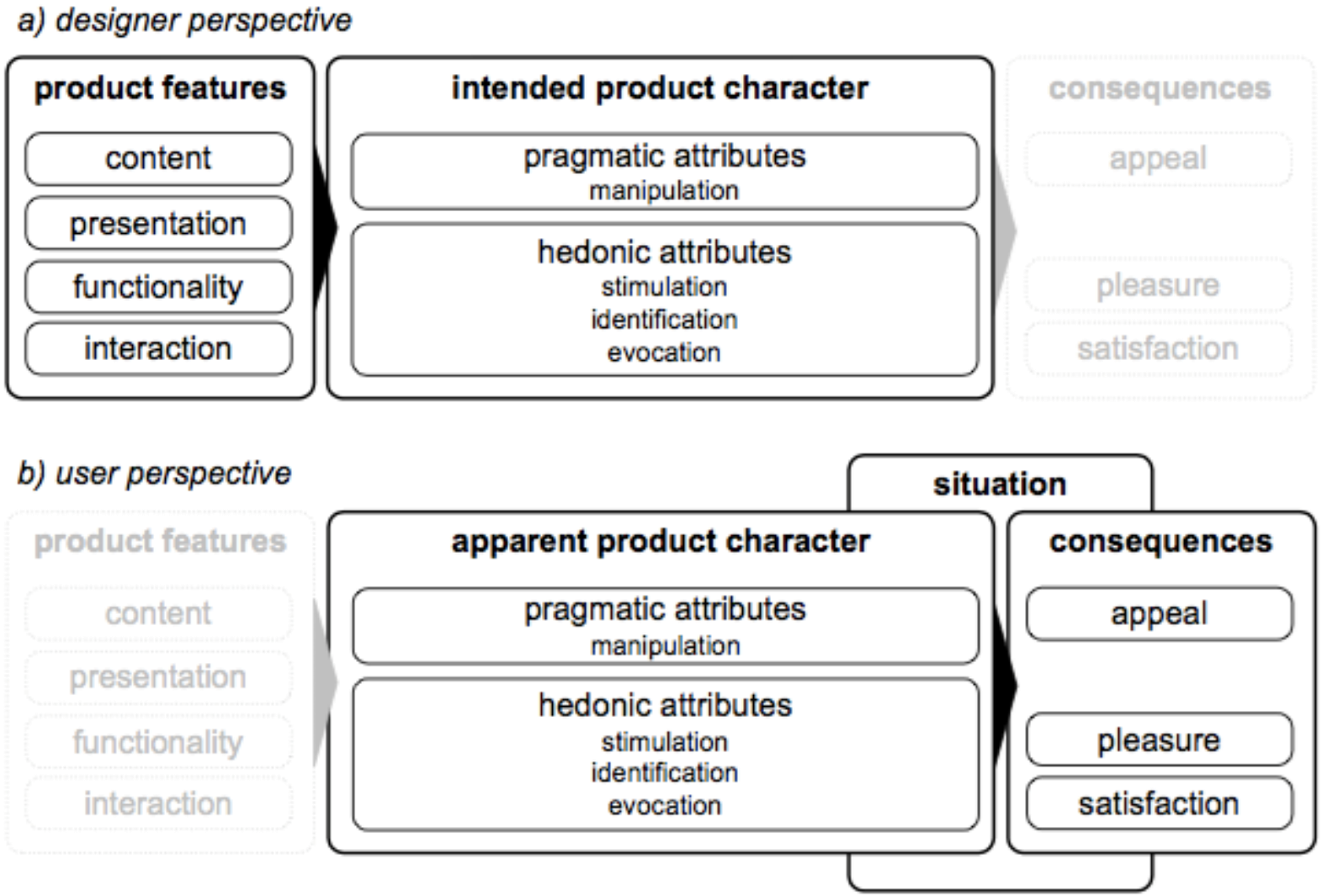
1. Meet the exact needs of the customer
2. Simplicity and elegance, products that are a joy to own/use

True user experience >> giving customers what they say they want, or providing checklist features

Seamless merging of the services of multiple disciplines: engineering, marketing, graphical, industrial and interface design.

User Experience (UX)

Designer vs. User



Marc Hassenzhal

UX facets

Usable

Simple and easy to use, familiar and easy to understand. Learning curve as short and painless as possible.

Useful

Fill a need. Overwise no real purpose for the product itself.

Desirable

Aesthetics attractive and easy to translate.

Findable

Information easy to navigate. Able to quickly find a solution to a problem.

Accessible

Users with disabilities can have the same user experience as others.

Credible

The company and its products or services need to be trustworthy.

Peter Morville's honeycomb

UX vs Usability

Usability = how easy & pleasant these features are to use

Learnability: How easy to accomplish basic tasks the first time?

Efficiency: Once users have learned, how quickly can they perform tasks?

Memorability: When users return after a period of non-use, how easily can they restore their skills?

Errors: How many errors, how severe, how easily can they recover?

Satisfaction: How pleasant?

UX vs User Interface

UI = anything a user may interact with to use a product or service

Screens, touchscreens, keyboards, sounds, lights…

UX focuses on the user's journey through the product to solve a problem

UI focuses on how a product's surfaces look and function, a series of snapshots in time

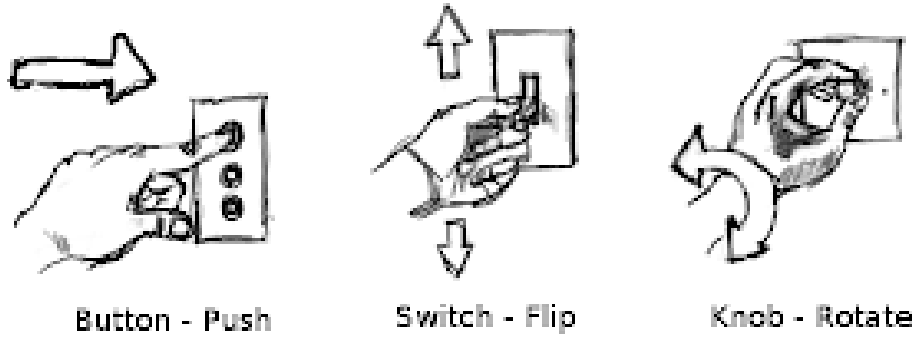
Affordance (Gibson 86)

An action a user perceives as possible

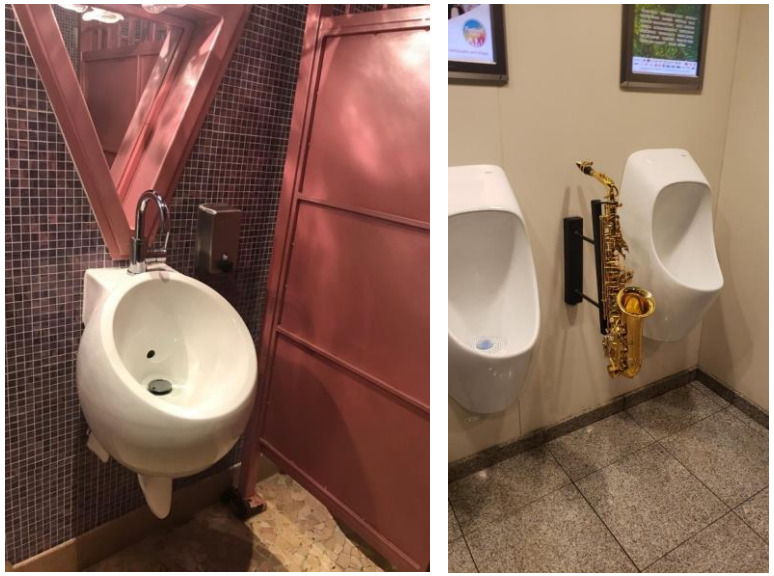
The quality of an object that communicates a way to use it

Hyperlink, drawer handle…

In game design will mean that the players know how to interact with items without significant instruction



Be careful of misleading affordance…



Dark patterns

Elements of product design created to make users do things they might not want to do—actions that benefit the business, not users

SOME **EVALUATION METHODS**

User Experience Research

Practice of analysing a user's interaction with the product to find insights and identify weaknesses

Qualitative methods

- Focus on observation rather than gathering numerical data
- Suitable for answering questions about the reasons that caused a problem
- *Interview, diary studies, usability testing…*

Quantitative methods

- Suitable for measuring success or discovering shortcomings
- Best to answer questions like "how many," "how much," "how often,"…
- *Clickstream analytics, A/B testing, survey…*

5 seconds test

- Measure what information users take away and what impression they get within the first 5s after viewing a design
- Commonly used to check whether web pages effectively communicate their intended message
  - *Ex: What is the purpose of the page? What are the main elements you recall? Who do you think the intended audience is? Did the design/brand appear trustworthy?*
- Not suited to measure comprehension of complex information
  - A page that requires lots of reading (prefer a design survey)
  - Predicting user behavior (prefer a click test or navigation test)
  - Asking complex questions (prefer a design survey)

AttrakDiff

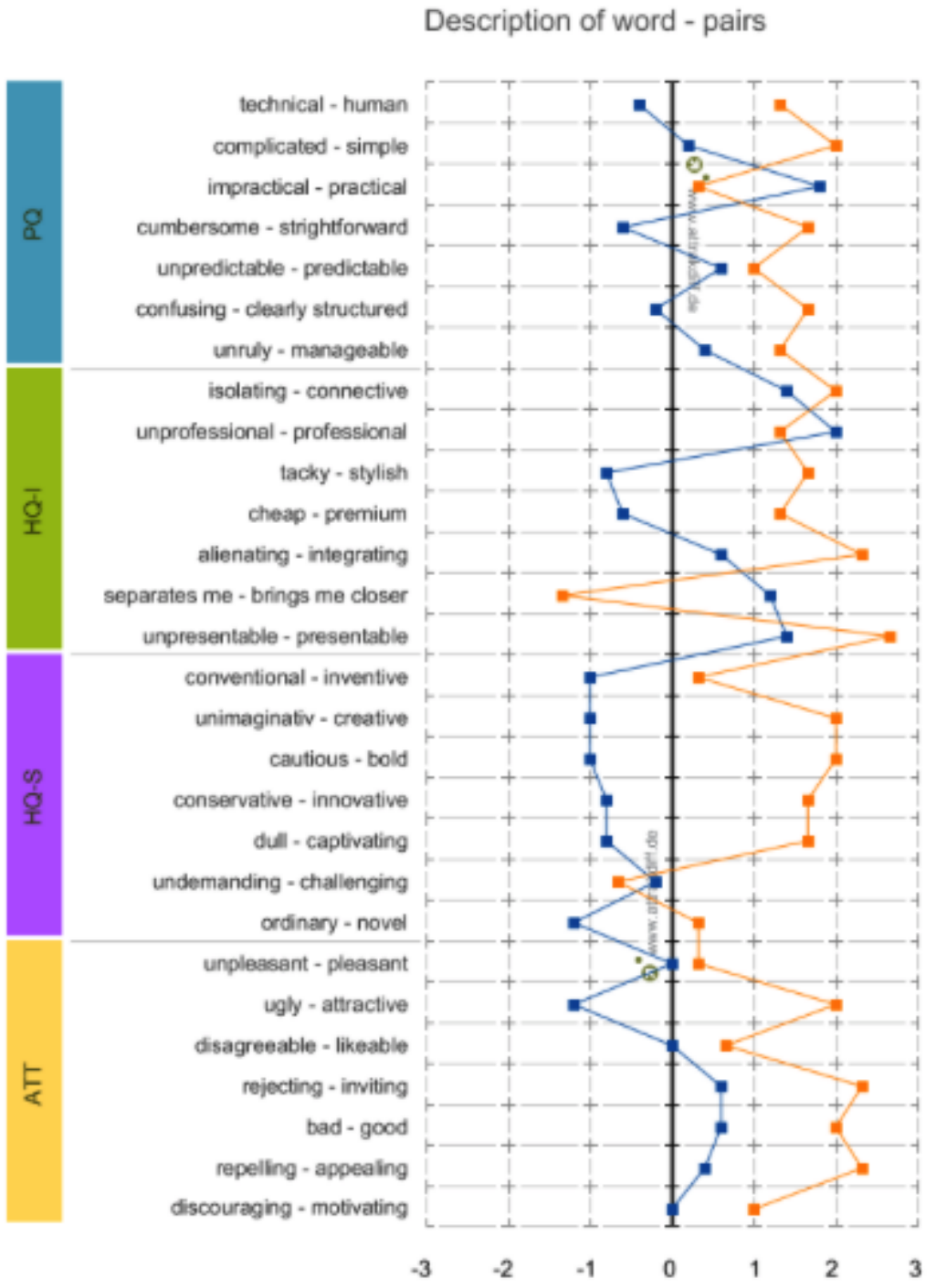
Hassenzahl, Burmester & Koller, 2003

Helps to understand how users personally rate the usability and design of the product.

- Single Evaluation
- A/B Comparison
- Before-After Comparison

Questionnaire

- Pragmatic quality
- Hedonic quality (identity - stimulation)
- Attractiveness



Heuristic evaluation

Small set of evaluators examine the interface and judge its compliance with recognized usability principles (the "heuristics")

Advantages

- Low cost, quick and easy to apply
- Can obtain feedback early in the design process
- Heuristics can be used both as a design and evaluation support

Heuristic evaluation

10 Usability Heuristics for User Interface Design (Jakob Nielsen 1994)

1: Visibility of system status
2: Match between system and the real world
3: User control and freedom
4: Consistency and standards
5: Error prevention
6: Recognition rather than recall
7: Flexibility and efficiency of use
8: Aesthetic and minimalist design
9: Help users recognize, diagnose, and recover from errors
10: Help and documentation

Applicable to video games

Heuristic evaluation

10 Heuristics for an optimal User Experience (Colombo & Pasch, 2012)

Derived from the flow theory (Csíkszentmihályi, 1975)

1. Clear Goals
2. Appropriate Feedback
3. Focused Concentration
4. Ergonomical Transparency
5. Technology Appropriation
6. Challenges/Skills Balance
7. Potential control
8. Follow the Rhythm
9. Know Thy User's Motivations
10. Conservative Innovation

Heuristic evaluation

Others

Ergonomic criteria for the evaluation of human-computer interfaces (Bastien & Scapin 92)

Ergonomic criteria for Human-Virtual Environments Interations (Bach & Scapin 2005)

Playability heuristics for mobile games

…

Heuristic evaluation

Drawbacks

- Requires knowledge and experience to apply the heuristics effectively
  - Judgment often based on expertise rather than heuristics
- Trained usability experts are sometimes hard to find and expensive
- May identify more minor issues and fewer major issues, or even false issues
- Non-exhaustiveness of the dimensions covered by the heuristics
- The heuristics are often vague, no precise recommendations or evaluation grid/criteria

=> Limited validity and reliability, recommended to use it in combination with other user-centered methods

Expert review

- Less formal evaluation
- Experts base their report not only on heuristics, but rather on their knowledge of user tasks, HCI guidelines and standards, and personal experience

Usability testing

Iterative method of testing few functionalities of a digital product by observing **real users** as they attempt to complete tasks on it

Goals

- Get user reactions and feedback
- Check if the user can perform the tasks proposed
- See if product meet user's expectations
- Check if the design is matching business decision to real world use

Usability testing

During the tests

- Do not lead the users
  - It's not about what you think of a good user experience, but how the user perceives the solution. so shut up, relax and listen.
- Ask users to externalize thoughts and feelings
- Keep test environment as realistic as possible
- Takes notes & record/log the session
  - Focus on what users do
  - Quantitative information: time on tasks, success and failure rates, effort (#clicks, perception of progress)
  - Qualitative information: stress responses, subjective satisfaction, perceived effort or difficulty
- Do not jump into any conclusions during the session

Further readings

https://uxdesign.cc/
https://www.smashingmagazine.com/usability-and-user-experience/
https://www.nngroup.com/articles/
https://medium.com/topic/ux
https://uxplanet.org
http://www.allaboutux.org/