Universal Design Language

*Written by Aaron Vontell - **Draft v1***

Shoutout to Tej Patel, Eric Wadkins, and Jack Spilecki for feedback and input

The goal of Universal Design Language (UDL) is to provide a way to represent and define user interfaces in a platform-agnostic way. In other words, a user interface on a website, mobile device, or even physical object can be defined using UDL. It primarily presents a language to talk about user interfaces.

The Gap - Why do current descriptions not work?

The current way to describe user interfaces are known to developers and programmers. In a nutshell, it revolves around:

- Views, components, or elements - basically objects that a user interface can be composed of. These objects have styles and properties.
- Hierarchy and structure - the organization of the views, components, and elements above.
- State machines - A way to represent the state of a user interface at any given moment, and define how a user may move from one state to another.

However, these common descriptions omit a lot of key points about user interfaces - what if the user interface, for instance, speaks out audio when the interface is displayed to the user - this is not usually considered a component of the “user interface,” but it is may be integral part of the interface. How about vibrations when a notification is received on the phone? How do we tie this in to describe the user interface of the device OS? Finally, where does the user come in? If an elderly person who has never used a smartphone before is shown a virtual button, do they view this component in the same way as an experienced smartphone user? How do we compare a mobile user interface to a web interface in the same format? These are the issues we hope to address with UDL.

Terminology

Part of UDL is a new way and new terminology to talk about user interfaces. Here are the main components and terms behind the language:

- **Perceptifers**
  Latin roots: PERCEPT - something to be perceived, FER - bearer

Definition - A bearer of something that is to be perceived by a user.
Example - A button is a perceptifer - it is a visual interface component that can be perceived by a user (if the user has a visual input channel (eyes) and the interface has a visual output channel (screen))
An audio clip being played from the device is also a perceptifer, since it displays a sound that can be perceived (heard) by the user

- **Interactifers**
  Latin roots: INTERACT - to take action against, FER - bearer
  Definition - A bearer of something that can be interacted with
  Example - A button perceptifer can have an interactifer - it is a visual interface component that can be touched by the user (if the user has a motor channel (hands) and the interface has a motor channel (a touchscreen)). **Important:** An interactifer is only created by a user's interpretation of a perceptifer, and a perceptifer may have more than one interactifer attached to it. However, an interactifer may also be freestanding, not attached to a perceptifer. In this case, we say that the interactifer is attached to the **empty perceptifer**. Additionally, if the user knows that something is an interactifer, but does not know the result of interacting with that interactifer, we call that an **unspecified interactifer.** Finally, a perceptifer may have multiple interactifers attached to it, but each interactifer must have a unique action.

  Further Details
  A part of a user interface can be just a perceptifer, just an interactifer, or both. A perceptifer is essentially a way to describe user interfaces without being bound to the restrictions that views and components have, in that views are only visual. Think of a perceptifer as a view, but it is also a way to describe non-visual user interface components. Perceptifers can therefore be of different types:
    - Visual Perceptifers: A UI component displayed through a visual channel, such as button through a screen
    - Audio Perceptifers: A UI component displayed through an audio channel, such as an audio instruction through a speaker
    - Motor Perceptifers: A UI component displayed through motor channels, such as an electric shock, or a vibration

- **Percept**
  A percept (which is already an existing word) means *an object of perception;*

something that is perceived.[1]
Perceptifers will **display** (defined below) percepts along their associated output channels on a devices (i.e. visual, audio, and motor channels). These percepts are then perceived and processed by a user. A percept is the actual properties or information about that perceptifer in the user interface. A user only becomes aware of perceptifers via its displayed percepts.

- **Action**
  Interactifers will receive actions along their associated input channels on a device (i.e. visual, audio, and motor channels). These actions are then processed by the driver behind the literal interface.

- **Display**
  Perceptifers are displayed along different mediums. Where display usually refers to displaying visually, in this language we use the word *display* to mean to allow a percept to be perceived through any of the main channels.

- ***Example Use Case - Button***
  Let's put this all together - imagine a button, it can be broken down into our defined language:
    - A button is a **perceptifer**
    - It's percepts are displayed through a visual channel (the screen)
    - The percepts that this perceptifer bears are:
      - Text - what text does the button have
      - Position - where is the button placed within the screen? Relative to the other percepts
      - Style - what color is the button? What is its width? Height? Border radius? Border width and color? Text color?
    - IMPORTANT NOTE: Perceptifers and percepts are only defined by the literal things that they contain, with no interpreted information or meaning. For instance, an impossible percept would be "affords clickability" - this is because this affordance only becomes relevant when a percept is processed by a user, and the user realizes that this is a button and should be clicked. In other words, when describing a UI component in this

[1] There is a special type of percept, called **virtual percepts** - these percepts are things that would not be perceived by a real user in real life, but can be detected by human simulators. For instance, an image having alternative text for screen-readers may be virtual percept, and can be detected by our AI / simulators. The point of these percepts is to take a "shortcut" to detected some interface properties. Instead of having to go through the interface without a screen reader and then with a screen-reader, we can do it it one go, sending this extra information as a virtual percept.

language, the descriptions should truly be separate from a user's interpretation, and should only contain things that can be perceived.

- ***Example Use Case - Audio Instruction***
  Let's see this language used in the context of an audio instruction displayed to a user
    - An audio clip is a **perceptifer**
    - Its percepts are displayed through an audio channel (the speakers)
    - The percepts that this perceptifer bears are
        - Content - What are the signals / important speech waves present? What are the frequencies at that moment in time?
        - Volume - How loud or quiet is a percept?
    - Note how the audio percepts never contain information about what is said, or things like gender and accent - these are all part of the users interpretation

IMPORTANT CONCEPT HERE: We now break "User Interface" into three components: The Literal Interface, the User, and the User Interface. In a nutshell, the Literal Interface is the actual non-interpreted composition of the displayed user interface, the User is the entity interacting with that interface, and the User Interface is the entire interpretation of the Literal Interface as seen by the User.

- **Literal Interface**
    - A set of perceptifers and the percepts they display
    - A set of output channels that can display the percepts
    - A set of input channels that can take actions from the user
- **User**
  Describing a user is obviously a difficult task, but that won't stop us from trying
    - A memory / history unit
    - A context, or their current environment and goals within this environment
    - A set of input channels for receiving percepts
    - A set of output channels for interacting with
- **User Interface**
    - A set of perceptifers and the percepts that they display
    - A set of output channels that can display the percepts
    - A set of input channels that can take actions from the user
    - A set of interactifers, which are mapped to perceptifers or or the empty perceptifer. These interactifers can be specified (i.e. actions known) or unspecified (i.e. actions unknown)

        - The user's internal memory and context, which may include objects such as:
            - An automaton representing the flow of the interface

Principles / Big Ideas

Here are some of the big ideas and concepts that drive UDL:

- Separate the **user** from the **interface** in **user interfaces**.
- Perceptifers become apparent to a user once its corresponding percepts have been displayed and perceived by a user
- A perceptifer may be transient or persistent - i.e. a visual perceptifer such as a button is primarily persistent, while an audio perceptifer is often transient
- The state of a literal interface only changes with a perceptifer changes, but the state of the user interface can change at any time, based on the internal state of the user
- In rare cases, interactifers can be created in the absence of a perceptifer - for instance, from a user context, they may derive the interactifer that they can snap their fingers to open an app on their phone. There is nothing perceived here to facilitate this, other than the user's own internal memory of the action.
- A user may create and attach an interactifer to a perceptifer without knowing this their truly is an interactifer (i.e. you may see something that looks kind of like a button, and may think that you can click it).
- A user's action is most often related to an interactifer, but user's may also take freeform actions (for instance, waiting)

The User Interface Interaction Loop (UIIL)

Now that we have built out this language, what does the process for interpreting and interacting with a user interface look like? Also known as the Display-Perceive-Process-Action (DPPA loop))

1. The User is defined with memory and context
2. A Literal Interface is presented (**Display Step)**
    a. All perceptifers of the user interface will display their percepts
    b. If an output channel for that percept is not available, that percept is destroyed

3. A User received all percepts along input channels that they have access to (**Perceive Step**)
    a. The user perceives all percepts that are sent along these IO channels
4. The User processes all percepts that have been perceived (**Process Step**)
    a. The user chooses what to do with each percept - they may be ignored, stored in short term memory, or heavily focused on
    b. The user uses these percepts to reconstruct a model of the Literal Interface (i.e. constructs a set of perceptifers)
    c. The user uses memory, context, and perceptifers to construct and map interactifers.
5. The User decides on an action (**Action Step**)
    a. Given the set of interactifers, the User's context, and the output channels available to them, the user takes an action, possibly against an interactifer.
6. Move to step 2