ARTIFICE Construct

Public Release

Terminology has been deliberately obfuscated in this document. Certain keywords have been substituted. Full documentation will (most likely) not be made publicly available

1 Background

1.1 What is a Construct?

Put loosely, a construct is a system of people, locations, and objects that interact with each other over time.

1.2 What kind of people?

Each person has a main **type**. The types are:

- Primary: These people are the ones that we are primarily concerned about. Each day, their goal is to execute a Routine.
- Secondary: Think of these people as "supports". They help Primary people complete their Routine.
- Outsider: Any person who is not a Primary or Secondary, but occupies a location within the Construct. These people do not necessarily help Primary people execute their Routine, though depending on their proximity to a Primary person on the Social Network graph, they can.

1.2.1 Subtypes of Primary People

Primary people can be broken down into **sub-type**. We have four main subtypes:

- A
- S: Has further sub-subtype SM.
- X: Has further sub-subtypes XA and XC.
- H: Stands for "hybrid". A combination of two out of A, S, and X.

We call the four sub-types "subtypes", and any of the specific sub-subtypes **spec types**. The letters can be ambiguous (as for example, X could refer to either the specific spec type X, or the group X, XA, and XC in totality), so it is best to specify if we are referring to the subtype or spectype.

1.2.2 Subtypes of Secondaries

Secondaries have only three subtypes, which we refer to as **status**: Out of Network, Active and Inactive. These refer to the capacity of the secondary to do work, which we will explore later.

1.2.3 Subtypes of Outsiders

Outsiders are described by their proximity to a Primary person on the social network graph. We have:

- Parent Node
- Parallel Nodes: Parallel, Peer, Semi-Pair, Pair.
- Child Node
- True Outsider, where the Outsider in question has no connection to the Primary in question.

We will explain the differences in classification later.

1.3 Status, Mode

At any given time, a Primary person has a **mode** that they are in. We will explain all the functions available in each different mode, but here is a list of all modes and **sub-modes**:

- Zone: Sub-modes are Standard and Marine.
- Chair: Sub-modes are Analog, Power, Shell, Hollow, and Marine. More sub-types will be explored later.
- Anchor: Sub-modes are Rail, Marine, and Connector. More sub-types will be explored later.
- Struct: Sub-modes are Restricted, and Shell. More sub-types will be explored later.

We assign each person a **height** and **weight**, exactly the same as their standard definitions.

1.4 Multipliers

Given a person's height in centimeters h, let their n-multiplier ψ_n be defined as:

$$\psi_n = (\frac{h}{170})^n$$

2 APOS, BPOS

2.1 Coordinate System

We implement the standard Euclidean 3-dimensional coordinate system. With a frame of reference as a stationary person, define $+x$ as their right, $+y$ the direction they are looking, and $+z$ the direction of "up" (against gravity).

We work in units of centimeters. When we shift frames of reference, we aim to keep these directions consistent, and just translate the origin.

2.2 BPOS Points

A BPOS point can be defined as either **proximal** or **distal**.

2.2.1 Proximal Points

A proximal point has two values: r and θ, where $0 \leq r \leq \frac{850}{9} \approx 94.444$ and $-\pi \leq \theta \leq \pi$. $r = 0$ is typically the lowest z-value, though there are exceptions we will explore later. $+\theta$ is generally defined as the "right", $+x$, though r and θ are independent of the xyz global coordinate system.

2.2.2 Distal Points

A distal point has four values: A 0 or 1 indicating left or right, a 0 or 1 indicating lower or upper, then r and θ. Upper r ranges $0 \leq r \leq \frac{10115}{144} \approx 70.2430555$, and lower r ranges from $0 \leq r \leq \frac{680}{9} \approx 75.5555$.

For both proximal and distal points, $\theta = 0$ indicates a $+y$ value.

2.3 BPOS Angles

Within a BPOS there are several **joints** that connect **segments**, all with between 1 to 3 degrees of freedom and various ranges of motion. We describe this as a tree, since the point of rotation of certain joints are the endpoints of other segments.

Define the origin $(0, 0, 0)$ to be at proximal $(0, \pi) = (0, -\pi)$. Then we consider the following independent trees:

2.3.1 Node 1 and 2: LLA, RLA

The left LA and right LA nodes comprise nodes 1 and 2. They are symmetric, so let's consider them together. Both have 2 degrees of freedom.

Point of rotation: $(0, -\frac{\pi}{2}), (0, \frac{\pi}{2})$. Length $\frac{340}{9} \approx 37.7778$ (to xLB nodes).

θ_{XZ}: Ranges $(\frac{\pi}{6}, \frac{7\pi}{6})$. Defined as the angle that section xLA makes with the distal portion of the BPOS.

θ_{YZ}: Ranges $(-\frac{\pi}{9}, \frac{2\pi}{9})$. Defined as the angle with the YZ plane.

LLB, RLB Nodes Point of rotation defined by LLA and RLA, angles $-\frac{\pi}{2}$, $\frac{\pi}{2}$. Length $\frac{935}{24} \approx 38.95833$. Two degrees of freedom.

θ_{XY}: ROM $(-\frac{\pi}{4}, \frac{\pi}{3})$, with distance away from the midline being positive.

θ_{XZ}: ROM $(-\frac{\pi}{6}, \pi)$. Defined as the angle xLB segment makes with its corresponding xLA segment.

LLC, RLC Nodes Point of rotation defined by LLB and RLB. Length $\frac{425}{16} = 26.5625$. Three degrees of freedom.

θ_{XY}: ROM $(-\frac{\pi}{9}, \frac{\pi}{4})$.

θ_{XZ}: ROM $(-\frac{\pi}{6}, \frac{3\pi}{4})$.

θ_{YZ}: ROM $(-\frac{\pi}{9}, \frac{\pi}{6})$. Angling $\theta = 0$ away from midline increases θ_{YZ}.

2.3.2 Node 3: TB

This is a proximal joint with three degrees of freedom, point of rotation $(\frac{595}{36}, -\pi) \approx (16.52778, \pi)$. Length $\frac{425}{9} \approx 47.2222$.

θ_{XY}: ROM $(-\frac{\pi}{6}, \frac{\pi}{6})$.

θ_{XZ}: ROM $(\frac{\pi}{6}, \frac{7\pi}{6})$. Linked with xLA's θ_{ZX} ROM.

θ_{YZ}: ROM $(-\frac{\pi}{6}, \frac{\pi}{6})$.

TA Node A proximal joint with three degrees of freedom, point of rotation defined at distal point $(\frac{425}{9}, -\pi)$ with respect to TB. Child of TB.

θ_{XY}: ROM $(-\frac{\pi}{2}, \frac{\pi}{2})$.

θ_{XZ}: ROM $(\frac{2\pi}{3}, \frac{4\pi}{3})$.

θ_{YZ}: ROM $(-\frac{\pi}{6}, \frac{\pi}{6})$.

HJ Node A proximal joint with one degree of freedom. Point of rotation defined at $(\frac{3995}{288}, -\pi) \approx (13.87152778, -\pi)$ with respect to the TA Node. ROM $\theta_{XZ} = (0, \frac{\pi}{6})$.

HL Node A proximal joint with one degree of freedom. Linked to HJ. ROM not defined as "angle", but rather distance between HL Node and the point diametrically opposite the HJ node. Ranges $(0, \frac{3}{2})$.

2.3.3 Node 4: LUA, RUA Nodes

Child nodes of TB node, points of rotation defined at distal point $(-\frac{425}{9}, \pm\frac{\pi}{2})$. Two degrees of freedom. Length $\frac{595}{24}$.

θ_{XZ}: ROM $(-\frac{5\pi}{18}, \pi)$.

θ_{YZ}: ROM $(-\frac{\pi}{6}, \frac{5\pi}{6})$.

LUB, RUB Nodes Child nodes of LUA, RUA. One degree of freedom. Length $\frac{425}{16}$.

θ_{XZ}: ROM $(0, \frac{5\pi}{6})$.

LUC, RUC Nodes Child nodes of LUB, RUB. Three degrees of freedom.

θ_{XY}: ROM $(-\frac{\pi}{2}, \frac{\pi}{2})$.

θ_{XZ}: ROM $(\frac{2\pi}{3}, \frac{4\pi}{3})$.

θ_{YZ}: ROM $(-\frac{\pi}{6}, \frac{\pi}{6})$.

LUD1, RUD1 Nodes Child nodes of LUC, RUC. $\theta_{YZ} \in (-\frac{\pi}{4}, \frac{\pi}{4})$; $\theta_{XZ} \in (-\frac{\pi}{3}, \frac{\pi}{3})$.

LUD2-5, RUD2-5 Nodes Child nodes of LUC, RUC. $\theta_{YZ} \in (-\frac{5\pi}{18}, \frac{5\pi}{18})$; $\theta_{XZ} \in (\frac{5\pi}{9}, \frac{7\pi}{6})$.

LUE2-5, RUE2-5 Nodes Child nodes of LUD2-5, RUD2-5. $\theta_{XZ} \in (\frac{\pi}{3}, \pi)$.

2.4 Subtype A

For those with subtype A (including any hybrids), we define the **extent** of their level A status by the set A of parent distal nodes removed, and the proportional lengths $A^{\top}$ of the adjacent segments.

2.5 Node Grading

We provide three components to grade a node - χ, ξ, and Ξ. Let χ and ξ both be ordered pairs (r, θ), and let Ξ be a set of reals in the range $[0, 1]$, corresponding to XY, XZ, YZ in that order.

For χ, let $r \in \{0, 1, 2, 3, 4, 5\}$, and θ some angle in the ROM range.

For ξ, let $r \in \{0, 1, 2, 3\}$, and θ some angle in the ROM range, less than χ_θ.

Let $\Xi \geq 0$.

2.6 Contact Points and Leverage

To demonstrate **ability**, we evaluate the ability of the user to move a point on a given node to a certain point in Euclidean space. We must consider both Ξ's value and **leverage** - the position against gravity.

We first assign weights to each segment of BPOS defined above.

- TB: 0.508
- TA: 0.094
- xLA: 0.083
- xLB: 0.055
- xLC: 0.012
- xUA: 0.027
- xUB: 0.016
- xUC: 0.005

Suppose we are interested in moving node RUD2 to point $(0, 0, 80)$. We consider the set S of all its parent nodes - in this case, $\{TB, RUA, RUB, RUC\}$. We also consider set S_Ξ, the set of corresponding Ξ values to the nodes. In this case suppose they are

2.7 Latency, Uncertainty, and Randomness

2.8 Enhancements

3 Chair Mode

3.1 Sub-modes

Here we consider a few Chair sub-modes: Analog I, Analog II, Power I, and Shell I. There are more, but we will consider them later.

- Analog I, II: Has variables x, y, z representing length, width, and height in meters. Also has variable r for radius.
- Power I: Has variables:
  - (x_c, y_c, z_c)
  - (x_b, y_b, z_b)
  - (x_s, y_s, z_s)
  - h_a
  - l_f
  - r_1, r_2, r_3
  - A set S, where each element $i \in S$ has variables i_1, i_2 describing the diagonal of the minimal rectangular prism that encloses the object at CPOS preset Model(0).
- Shell I: Inherits all the variables from Power I, as well as d, representing depth.

3.2 CPOS

CPOS, short for "Chair Position", measures the position of the Chair object in space at some given time. Each different chair sub-mode has different degrees of freedom, which we will detail below.

- Analog: No degrees of freedom.
- Power: θ_H, θ_B, h, θ_K
- Shell: θ_B, h

We provide limits on the values these can take on:

- $\theta_H \in (\frac{\pi}{3}, \pi)$. By default $\frac{\pi}{2}$.
- $\theta_B \in (0, \frac{\pi}{2})$
- $h \in (0, \frac{\psi}{4})$
- $\theta_K \in (\frac{\pi}{2}, \pi)$. By default $\frac{\pi}{2}$.
- $\theta_H + \theta_B \leq \pi$

3.3 CPOS Presets

We can describe a CPOS configuration through the four variables listed above. To save ourselves time, we will describe some preset CPOS configurations that we will use extensively. Most Presets will take an argument as a variable, with 0 as default.

- Power Presets
  - Model(x): $(\theta_H, \theta_B, h, \theta_K) = (\frac{\pi}{2}, 0, x, \frac{\pi}{2})$
  - Default(x): $(\theta_H, \theta_B, h, \theta_K) = (\frac{\pi}{2}, \frac{\pi}{8}, x, \frac{\pi}{2})$
  - Flat(x): $(\theta_H, \theta_B, h, \theta_K) = (\pi, 0, x, \pi)$
  - Back(x): $(\theta_H, \theta_B, h, \theta_K) = (\frac{2\pi}{3}, \frac{\pi}{3}, x, \frac{2\pi}{3})$
- Shell Presets
  - Model(x): $(\theta_B, h) = (0, x)$
  - Default(x): $(\theta_B, h) = (\frac{\pi}{8}, x)$

3.4 Line of Sight

We consider that according to modern research, the range of motion of human eyes are $\pm\frac{\pi}{4}$ radians each direction horizontally, $\frac{\pi}{6}$ radians up, and $\frac{\pi}{4}$ radians down. Binocular vision (focus) covers an arc of length $\frac{2\pi}{3}$ horizontally and $\frac{\pi}{3}$ vertically, while peripheral vision covers an arc of length $\frac{7\pi}{6}$ horizontally and $\frac{3\pi}{4}$ vertically.

Certain tasks will require a possible binocular vision line of sight, such as reading text from a screen, while other tasks will only require peripheral vision, such as detecting motion.

We define the focal point of the eyes to be at proximal point $(12, 0)$ with respect to node TA. We additionally restrict vertical FOV downwards to less than $\frac{\pi}{2}$.

4 Primary Inputs

We define a primary input as some physical device the user interacts with, upon which their inputs are interpreted by the CPU. There are multiple types of "input readers", which can take a number of unique inputs.

4.1 Basic Readers

Here we outline the most common types of input readers.

1. 1-Reader
    - Takes one single input. Consider it as a "button" of sorts. An example would be a Morse code reader.
2. n-Reader
    - Sends one input at a time, but has n different input values that can be passed.
    - Example: A Kahoot quiz would be a 4-reader. You can send 4 different inputs, but only one at a time.
3. (n, m)-Reader
    - Sends up to m inputs at a time, with n different input values. $m \leq n$.
    - Example: A full size piano would be a $(88, 10)$ reader, since there are 88 different keys, but we can only send 10 inputs at once since we (typically) have 10 fingers. (Obviously you can press more than one key with one finger, but we'll ignore that for the purpose of this example).
4. Voice
    - Voice commands can be thought of as a generalization of an n-Reader, where n is trivially large.
5. Joystick
    - Can be anywhere on a polar coordinate grid, with $r < 1$.
6. (n, m)-Touchscreen
    - Simply a touchscreen. Some position (x, y) given $0 \leq x \leq n$, $0 \leq y \leq m$. When the user is not actively touching the screen, the cursor disappears.
7. (n, m)-Dragscreen

- A $n \times m$-touchscreen where the cursor does not disappear when not being actively pressed. Consider something like a cursor on a computer screen; it does not disappear when not in use, but simply stays at the last point it was touched.

Of course we can generalize these to simply the (n, m)-Reader, (n, m)-Touchscreen, (n, m)-Dragscreen, and joystick.

To simplify notation, let $R(n, m)$, $T(n, m)$, and $D(n, m)$ represent the above three input types, respectively.

4.2 CPOS Mounts

Of course, such input readers do not exist in a vacuum. We mount them on the Chair object, relative to CPOS, and proceed from there. Here we summarize the most common positions for mounts, though other mount locations can be defined through CPOS coordinates.

1. RF, LF - Available for Power, Shell
2. RH, LH - Available for Analog II, Power, Shell
3. HD - Available for Power, Shell
4. SP - Available for Power, Shell
5. CT - Available for Shell
6. EE - Available for Power, Shell
7. Describe two inputs as **independent** if they do not share a mount location.

There are mount "sub-positions", which we will describe in detail in future sections.

4.3 N-Switches, Complexes

An n-switch is a collection of n independent 1-readers that are combined to form an n-reader, by mounting them all in the same location, with different sub-locations.

An n-complex is an n-switch and a joystick mounted in the same location.

This terminology will become useful later. For now, let ς_n denote an n-switch and ξ_n denote an n-complex.

4.4 Sequencers and Modifiers

For all (n, m)-Readers, we can define sequential inputs as unique inputs. Think of Morse code - there are technically only two input values (short and long), but they are transformed into letters based on sequencing.

A modifier is using a second, independent input to modify the value of an input. Think of the SHIFT key on the keyboard.

4.5 Information Conveyed

There are several types of information that can be passed to the CPU via an input, which we will detail below.

1. n, m-unary
    - The most basic type of input. Simply passes an event from n different possible events when triggered to the CPU.
    - Sequencing an R input will pass unary information.
2. n, m-unary OO
    - An n-unary input, which additionally passes information as to whether or not each possible input is being currently pressed at the current time.
    - Think of a 1-unary input as knocking on a door ("holding" down a knock won't change the knock), while a 1-unary OO input might be a buzzer (holding down the buzzer for some length of time will keep the buzz going longer).
3. Directional
    - Some input that can be translated into a position on a polar coordinate plane with $r \leq 1$.
    - This input will be used as "direction of motion" (think Mouse Keys but better).
    - Obviously, the joystick is the best fit for this input, though any 2-dimensional input will work.
4. (n, m)-Absolute
    - Some input that can be translated into a coordinate $(x, y) \in (\mathbb{Z}^+)^2$ on the Cartesian plane, given $0 < x < n$, $0 < y < m$.
    - This input will be used as "the cursor is here at this point".
    - A T or a D input would be the best here.

5 Secondary Command Mapping

A "secondary command" is some CPU-driven task triggered by some combination of primary inputs.

5.1 Cursor Position

Every user is equipped with a digital display. They must control the cursor position somehow. We assign a primary input based on the following criteria:

1. Assign the (n, m)-Absolute input with the greatest value of $n \times m$.
    - The cursor position is simply the value of (n, m) at some time t.
2. If no Absolute inputs exist, assign a Directional input.
    - The directional input will move the cursor around.
3. If no Directional inputs, assign an n, m-Unary OO input.
    - If $n \geq 4$, we can assign a WASD-type input. Preferably $m \geq 2$, though it is not mandatory.
    - Else if $n = 3$, we can assign W and D to two keys, and the third key to a local TOGGLE to switch the keys to S and A.
    - Else if $n = 2$, we can assign a key to "move" and a key to a local TOGGLE, toggling the 4 directions.
4. If no Unary OO inputs, assign an n, m-Unary input.
    - Such inputs will have a determined "step size", based on the amount of precision needed. Detailed in section 5.2
    - We assign such inputs for $n \geq 2$ identically to the Unary OO inputs, just instead of "move", we have "step".
    - If $n = 1$, a "scanner" will bounce up and down the screen, upon which the user will press the input at their desired Y coordinate. The process will repeat to select an X coordinate. The input can be re-pressed to start the process again.
    - RCLICK can be inherited for this purpose (see section 5.3).

5.2 Cursor Precision

As mentioned above, some degree of precision is needed. It would be quite hard to select items on a tiny screen, so we must introduce a few features:

- Zoom
- Velocity, Step Size
- User Precision: Position

- User Precision: Time
- Desired Precision

Denote "User Precision Position" by ϵ_x and ϵ_y, which represent the number of unique x and y positions that the user can hit reliably. Trivially, for $T(n, m)$ and $D(n, m)$, $(\epsilon_x, \epsilon_y) = (n, m)$. The precision of a joystick might vary from user to user.

User Precision Time is denoted by ϵ_t, and describes the precision of time interval that a user can reliably hit for an unary OO or directional input. A precision of 0.01 seconds is most likely unattainable for humans, while a precision of 1 second is trivially attainable.

Denote "Desired Precision" by δ_x and δ_y - the required precision of the cursor (or any object being moved in general, when we generalize this later).

For Directional inputs, we introduce Velocity. This is the speed at which the cursor moves within an $n \times m$ screen, measured in units per second. We have:

$$\vec{\delta} \geq v \cdot \epsilon_t \cdot \frac{2}{\vec{\epsilon} - 1}$$

For example, if our joystick has precision (3,3), time precision 1/4 seconds, and we desire an accuracy of 1×1, we have:

$$1 \geq \frac{v}{4} \implies v = 4$$

meaning that our cursor will move at a speed of 4 units per second at top speed on an $n \times m$ grid.

For Unary OO inputs, we can use the same formula for velocity.

We introduce Step Size for unary to cursor mappings. Let Δ represent the step size in units on an $n \times m$ grid; trivially for cursor position, step size should be $\Delta = 1$.

Finally, we introduce Zoom to certain Absolute-cursor mappings. Suppose we have desired precision δ_x, δ_y and actual precision n, m. We assign a minimum Unary input to a local TOGGLE variable to toggle the zoom level around the current cursor position. The minimum number of zooms is:

$$\min(\lceil \log_{\epsilon_x} \delta_x \rceil, \lceil \log_{\epsilon_y} \delta_y \rceil)$$

In the event that the number of zooms is greater than 2, it is recommended but not required to assign inherit two TOGGLE-like commands, for zoom-in and zoom-out.

5.3 Universal Commands

We must first discuss "sub-modes". There are several sub-modes that a CPU can be in at a given time; think of them as program windows. Pressing Ctrl+T on Chrome has a different function than if you were to press Ctrl+T on Discord. In this case, Chrome and Discord would both be sub-modes.

We now introduce the concept of "Universal Commands", which are input-to-secondary mappings valid in every sub-mode (example Ctrl+Alt+Delete). As seen above, TOGGLE is one of the Universal Commands, and can take on a variety of useful functions within certain control configurations. The universal commands are as follows:

1. TOGGLE: Switches "axis" within a given control scheme.
2. LCLICK: A Left Click function - typically confirms the current selection.
3. SWITCH: Returns to the homepage, where the user can select a mode.
4. RCLICK: Right Click - some alternate function from the current selection.
5. BACK: Goes back by one menu.
6. STOP: Erases the current call stack and returns the Chair to resting.
7. EMERGENCY: In case of emergency. We will not be examining this function in detail, but it is important to have.

All these commands are assigned to Unary inputs - in the event that the primary inputs aren't unary, they will be reduced to unary anyway. In the event that the user cannot map all seven commands, the above commands will be assigned in that order of priority.

It serves to reason that the minimum amount of unique unary inputs for any Primary person within an Artificer Construct is 4, since TOGGLE, LCLICK, SWITCH, and RCLICK are absolutely necessary. Keep in mind that a single unary input can be transformed into 4 unique unary inputs through sequencing.

5.4 Drive

We go into our first sub-mode: Drive. The most basic of sub-modes, this sub-mode essentially only needs to cover four functions.

1. Go Forward
2. Go Backwards
3. Turn Right
4. Turn Left

Of course, there are numerous ways to achieve this, which I will detail in order from most desired to least desired. Please keep in mind that inputs used here may not overlap with the Universal Commands, except to explicitly call a Universal Command (this is true for all sub-modes). Also note that when I say that a (n, m)-input is needed, I mean that any (x, y)-input must satisfy $x \geq n$ and $y \geq m$.

Keep in mind that a new non-independent, identical Unary OO input can be created by modifying an existing one with another Unary OO input.

1. A Directional input.
2. A single $(4, 2)$-Unary OO input.
3. Two independent $(2, 1)$-Unary OO inputs, with n maximized. One is mapped to Forward-Backwards, while the other is mapped to Left-Right.
4. A $(2, 1)$ unary input to increase/decrease the current speed by a Step Size and a $(2, 1)$-Unary OO input for Left-Right. Not necessarily independent, though recommended.
5. Two $(2, 1)$ unary inputs for increase/decrease speed and turn left/right. Not necessarily independent.
6. A $(2, 1)$ unary input for step forward and step right. Inherit TOGGLE to switch to backwards and left.
7. A $(1, 1)$ unary input to step forward. TOGGLE to switch WASD axes.
8. LCLICK to step. TOGGLE to switch.

Velocity and Step Size calculations are exactly the same as Cursor Position, but precision should be 15 degrees on Left-Right orientation and 0.5 meters for Forward-Backwards direction.

5.5 CPOS

Recall from above that CPOS has up to four arguments - $\theta_H, \theta_B, \theta_K, h$. We have the following options:

1. Assign a $(2, 1)$-Unary OO switch to all of them.
2. Assign some amount of $(2, 1)$-Unary OO switches, and use the TOGGLE function as needed.
3. Assign some amount of $(2, 1)$-Unary switches to step, and use the TOGGLE function as needed.
4. Assign some amount of $(1, 1)$-Unary switches to step, and use the TOGGLE function as needed.

5. Use LCLICK and TOGGLE.

CPOS is typically not time-sensitive, so we do not care about being fast. Additionally, we can select CPOS Presets (detailed above) using the cursor and LCLICK functionality.

5.6 Text Input

At its most raw, text can be input through an on-screen keyboard using the Cursor Position and LCLICK functionalities.

One option is to use some form of physical keyboard. We assign a $(n, 2)$-Unary input and some amount of toggles if we like the shift key, or a $(n, 1)$-Unary input to simulate a mobile keyboard.

Another option is to use a common word/phrase selector, where the screen shows $n \times m$ possible phrases or words to select, and the user selects them using LCLICK, RCLICK (zoom), and the cursor position. One must consider case-by-case whether or not this form of input is faster than a physical keyboard - the average phrase length is 2, and there must be at least 5000 words and phrases in the bank (a 71×71 grid).

5.7 Camera

There are several cameras placed around the Chair, which we will detail later. What we are interested in right now is the functionality of them. Here are the functions:

- Zoom In, Zoom Out
- Turn Left, Turn Right
- Turn Up, Turn Down
- Take Picture

In this case, RCLICK is reserved to toggle between cameras. The first 3 commands can be assigned similarly to CPOS, using TOGGLE, while Take Picture only requires a $(1, 1)$-Unary.

5.8 Basic Schemes

A **scheme** is the control mapping for all sub-modes. We will cover sample schemes for common Primary archetypes later.

6 Hooks

A hook is a mechanical arm. While human arms can be described as hooks, the difference is that hooks are not mounted on the user, but rather on a Chair, Zone, or Struct object.

6.1 Body Types

The body (the arm of the mechanical arm) can be described by a sequence J of triples (f, k, r) and an origin point $\vec{o} = (x, y, z)$ where:

- f represents the rotational degrees of freedom in binary. 1 = flexion/extension; 2 = abduction/adduction; 4 = internal/external rotation.
- k is a real number between 0 and 1, representing what proportion of the length of the segment is adjustable.
- r is the distance from this joint to either the next joint in the sequence, or the hook head if the joint is the last in the sequence.
- The origin point (x, y, z) represents the point on the Chair where the first joint is mounted. We will deal with alternate-mounted hooks later.

For example, a human arm can be generalized to a hook where:

$$J = \{(7, 0, \frac{595}{24}\psi), (5, 0, \frac{425}{16}\psi), (3, 0, 0)\}$$

Continuing the arm analogy, we describe the last joint in the sequence of arm joints as the "wrist". The first joint is referred to as the "shoulder", and all middle joints are referred to as "elbows". Let J_0 denote the shoulder, J_n the n-th elbow, and J_w the wrist.

6.2 Head Types

Every head has a "smart gripper", which will attempt to grip an object by scanning it using radar and determining its dimensions. Of course, sometimes we want to grab two things at once, so we define a head with an ordered pair (n, k_1, k_2), where n is the number of "fingers", k_1 the number of degrees of freedoms (using the binary formula) of the primary finger joint, and k_2 the number of additional joints per finger. Let F_n denote the n-th finger, and $F_n(0)$ its primary joint.

A human hand can be generalized to a hook hand of $(5, 3, 2)$ (ignoring the fact that a thumb does not have a little finger).

We make some gross assumptions and assume that a hook hand has all their finger primary joints evenly spaced about a circle perpendicular to J_w, total length 20 cm, and has all their joints evenly spaced.

6.3 Primary Mapping

A hook body has two possible configurations. The simple one, we call **Camera-Driven**, where:

- This inherits all existing Camera controls, and adds some new ones.
- Move hook Towards/Away from Camera Central Crosshair
- Rotate Wrist CW/CCW

This body configuration uses smart technology to position the hook joints in a way that the head can reach the object, while avoiding contact. However, this has its shortcomings - what if we want to manipulate the middle joint positions as well (what if a shopping basket is placed on one of the joints and we want to hold it in place)? Then we must move to an advanced configuration which we call **User-Driven**, where we have:

- All current Camera controls
- A Length Increase/Decrease command for every joint where $J_n(k) > 0$
- A command for every degree of freedom in every joint. Maintain the same $(2, 1)$ Unary OO hierarchy as Drive and CPOS.

Finally, we move to the head. We have the commands:

- Smart Open / Close
- A command for every degree of freedom in every joint.

This is obviously incredibly inefficient. In section 10, we will go over macros and manipulation to more efficiently use a hook.

6.4 Capabilities

A hook has incredible grip strength and precision, limited only by the user. Of course, this is not without its limitations, but we will claim that a hook can hold any item in which one of the following criteria are met:

- Contact from the head is made on two diametrically opposite points of the object,
- There are three points of contact made "under" the object - that is, the object is resting on at least three points of the hook, whether it be the arm or the head.

We will do precision and velocity calculations later, to determine if a hook can do things like unscrew a bottle cap or cut your hair.

7 Basic Routine Components, Anchor Mode

7.1 Anchor Mode: Connector

For the Connector sub-mode of Anchor, we use a Cartesian coordinate system with +z as up. (x, y) are nonnegative reals, bound by some positive (x_k, y_k) such that $y_k > 170\psi$ and $x_k > \frac{2y_k}{3}$.

A Connector environment contains both External and Internal Connectors, which constrain BPOS.

An External Connector fixes a BPOS coordinate to an environment point. An Internal Connector fixes a BPOS coordinate relative to another BPOS coordinate. We can then re-evaluate the set of "reachable" points from BPOS given these Connector restrictions. As such, connectors can be described by an ordered pair of coordinates $\vec{r}_1, \vec{r}_2$.

7.2 Self-Enhance

There are two components to "self-enhancement". The first is the ability of a user to attach and detach a given External Connector at will, while the second is the ability of a user to attach and detach a given Internal Connector at will. We refer to Internal Connectors as "enhancements" when outside the context of the Anchor Connector mode. We will describe the conditions for self-enhancement in detail in the Articles section.

7.3 Manual

Outside the confines of Anchor Mode, we denote a user's Manual stat by M. Let M be an ordered pair of ordered sextuples $(F, \delta, \theta_x, z, z_x, m)$, one for the left and the right, according to the following criteria:

- F: Force output. According to the following formula:

$$16 \cdot UD1(\Xi_{XZ}) + 4 \cdot UD2(\Xi_{XZ} \cdot \Xi_{YZ}) + 4 \cdot UD3(\Xi_{XZ} \cdot \Xi_{YZ})$$

$$+2 \cdot (UD2(\Xi_{YZ}) + UD3(\Xi_{YZ}) + UD4(\Xi_{YZ}) + UD5(\Xi_{YZ}))$$

$$+2 \cdot UD4(\Xi_{XZ} \cdot \Xi_{YZ}) + UD5(\Xi_{XZ} \cdot \Xi_{YZ})$$

- δ: We use error propagation and quaternions to determine the error of node UC at some point (too lazy right now).
- θ_x: Minimum of $UB(\chi(\theta))$ and $\frac{\pi}{2}$.
- z: This is expressed as a multiple of the $UA - UB$ node length, assuming $UA - UB = UB - UC$.

- The "negative" ROM of the UB node (anything below $\frac{\pi}{2}$) can be converted to z as follows: Let the minimal UB angle reached be θ_z, then:

$$z = \cos(\theta_z)$$

- Then we consider azimuth angle. Consider the UA node.

There are a few "landmark" Manual levels. They are below:

- Bread: $z \geq \frac{1}{2}$
- Markdown: $z \geq \frac{3}{4}$
- Top: $z \geq 1$
- Stylus: $F > 0, z > 0, \theta_x > \frac{\pi}{4}, \delta <$
- Touchscreen: $\theta_x \delta$

7.4 OHP

Not sure why I made this its own section. Essentially we want $z = 2$, the maximum.

7.5 Swing

8 Blazy Seeds and Configs

8.1 Introduction

As a user provides inputs, they produce two kinds of waste. Call these L and W, measured in $cm^3 = cc = mL = g$.

8.2 Void

8.3 Blazy Seed

8.4 L Stride

8.5 W Stride

8.6 WL Collectors

8.7 L Collectors

8.8 Invokers

8.9 Information Theory

9 Articles

An Article is anything bound to the user's BPOS in some way. We define an Article with a number of criteria.

9.1 Zippers, Buttons, Loops

Define a zipper Z by an ordered pair of two BPOS coordinates, representing the bottom and the top of the zipper (not necessarily +z or -z, just the direction it "opens").

Define a button by two BPOS coordinates (the center of the button and the button "hole") and the fastening type. The types are below:

- A: Magnetic. Fastens on contact.
- V: Velcro. Requires correct orientation.
- P: Pull-through (think a dress shirt button).
- L: Latch.

Define a loop by an ordered triple, along with any Button objects it inherits. The ordered triple $(r, \vec{x}_1, \vec{x}_2)$ is defined as: r the thickness in meters, $\vec{x}_1, \vec{x}_2$ two diametrically opposite points on the loop.

9.2 Describing an Article

An article is either a "Pull" article or a "Wrap" article.

A "Pull" article consists of some number of "holes". We can describe each hole as an ordered triple (x, δ, Δ), where:

- x represents a BPOS point (the r component). A hole stretches the entire θ.
- δ represents the ratio of the radius of the unstretched hole to the radius of the corresponding node segment.
- Δ represents the ratio of the radius of the stretched hole to the radius of the corresponding node segment.

A "Wrap" article can be described as a rectangle in the xy-plane created by the Anchor Connector environment.

Both "Pull" and "Wrap" articles must be followed up by listing all the Zippers, Buttons, and Loops (ZBL for short) that they inherit.

10 Chair Mode II

10.1 Bands, Inputs, Enhancements

10.2 Advanced Inputs

10.3 Precision and Velocity

10.4 Inventory

10.5 Mounts

10.6 Screen UI

10.7 Settings Manipulation

11 Routines, Dependencies

11.1 Anchor Mode: Connector II

11.2 Blazy Invokers, Chair Mode: Hollow

11.3 Emptying Collectors

11.4 Anchor Mode: Marine, Chair Mode: Marine

11.5 Manual, OHP, Dead Time

11.6 Equipping and Removing Articles

11.7 Swing, Anchor Mode: Rail

11.8 Chair Bands

11.9 Bread Considerations

11.10 Non-Standard Events

11.11 Gym

12 Drivers, PT

12.1 True Driver

12.2 Modified Driver

12.3 Linked Driver

12.4 SAE 1 Driver

12.5 Other Driver Conditions

12.6 SPT and GPT Conditions

12.7 Scalability

13 Secondaries, Residencies, Time Complexity

13.1 Optional Features in Residencies

13.2 Structs

13.3 Time Complexity as Primary

13.4 Time Complexity as Secondary

13.5 Travel Time

13.6 Bread

14 Outsiders, Social Network

14.1 Pairs

14.2 Semi-Pairs

14.3 Base Nodes

15 Zone Mode

15.1 Available Inputs

15.2 PLC

15.3 ZPOS

15.4 Active Hook

15.5 Camera

15.6 Inventory

15.7 Torrent

15.8 Desert

15.9 Entering and Exiting Zone: Rail, Secondary, Direct

15.10 Procedures

16 Tech 3.0 and Beyond

16.1 CPOS+, Attachments

16.2 Remote Hooks

16.3 Remote Input

16.4 SAE 2, 3 Drivers

16.5 Other Tech

16.6 Macros

17 Blazy II

17.1 Primary

17.2 Travel Time and Availability

17.3 Risk Minimization

17.4 Known Information

18 Dependencies II

18.1 Bread

18.2 Blazy

18.3 Non-Standard Self

18.4 Non-Standard Environment

19 Struct Mode

19.1 Available Inputs

19.2 MAG, Shell, Restricted

19.3 Entering and Exiting Structs

19.4 Articles with Structs

20 Growth Curves

20.1 Height and Weight

20.2 Blazy Seeds

20.3 Latency

20.4 BPOS Progression

21 Residency Rings, Chained Coverage

21.1 Available Modes

21.2 In-Residency Groups

21.3 Multi-Residency Groups

21.4 Drivers, PT

21.5 Scalability

21.6 Native Language

22 Whitewater, Outsiders II

22.1 Research Groups: STEM

22.2 Combat Groups: Structs

23 Construct Creation

23.1 Level

23.2 Initial Conditions

23.3 World Initialization

23.4 Fixed Points in Time

23.5 World Events

23.6 Superficial Characteristics

24 Construct Auto-Population

24.1 Blazy

24.2 Assigning Levels

24.3 Assigning Residencies

24.4 Assigning Secondaries

24.5 Filling Outsiders

25 Vacations, Swaps

25.1 Scalability

25.2 Available Modes

25.3 Time Zone

26 Pushing Updates, Research

26.1 Update Waves

26.2 New Invokers

27 Social Networks II

27.1 Node Distance

27.2 Enterprise

28 Child Nodes II

28.1 Blazy

28.2 Sub-Routines

29 Schedule Generator

29.1 Perspective vs Omniscient

29.2 Global vs Local

30 Construct Creation II

30.1 Global Vars, Consts

30.2 Weighted Distributions

30.3 Global Events

31 Experimental Technology

32 Glossary

32.1 Terminology

- Type: The type of person we are referring to. Can be Primary, Secondary, or Outsider.
- Subtypes and Spec Types: Primary has sub types A, S, X, and H, and spec types A, S, SM, X, XA, XC, HAS, HAX, HSX.
- Secondary Status: Out of Network, Active, and Inactive.
- Outsider subtypes: Parent, Parallel, Peer, Semi-Pair, Pair, Child, True Outsider
- Mode: Zone, Chair, Anchor, Struct.
- Submode: Standard/Marine Zone; Analog/Power/Shell/Hollow/Marine Chair; Rail/Marine/Connector Anchor; Restricted/Shell Struct.