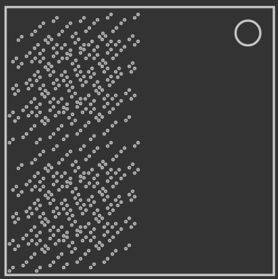
The Octadecayotton (10 Dimensions)



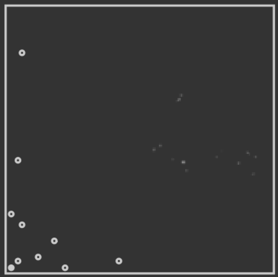
who'll be there when i'm gone?

By default 9 dimensions is used, but multiple Octadecayottons or mod settings can change it.

The 1024 spheres will start moving, resembling the rotations of a 10-dimensional cube. 3 rotations are shown, followed by a brief pause before starting over.

Identifying Dimensions

- There are 10 dimensions: X, Y, Z, W, V, U, R, S, T, and O. Each axis can either be positive, or negative.
- Any positive axis points to the right (+X), up (+Z), and/or towards (+Y) the viewer when viewed from the front.
- Any negative axis points to the left (-X), down (-Z), and/or away (-Y) the viewer when viewed from the front.
- Any given sphere has 10 neighbours, which are all 1 of each of the 10 axes, either positively or negatively.



- To the diagram to the right shows on example of a sphere's 10 neighbours as seen from the front. The unfilled spheres are neighbours of the filled sphere.
- In reading order (left-to-right then top-to-bottom), the spheres are O, S, Z, V, W, U, T, R, Y (stacked on top of the filled sphere), and X.
- In this example, going from the filled sphere to any unfilled is a positive axis, while going from an unfilled sphere to the filled sphere is a negative axis.

Identifying Rotations

- Look at 2 spheres that are neighbours (note down their initial axis that connects them together) and watch them transform.
- If these 2 spheres are now neighboured from a different axis, that implies a rotation. Record the new axis that they transformed into, and repeat this process starting with the new axis until the first initial axis is reached.
- For any negative initial axis, invert both the initial and new axis. (positive -> negative, negative -> positive)

Identifying Rotations (...continued)

- When the first initial axis has been reached, start from the bottom of the list, and working your way up; append all the new axes to a separate list.
- The resulting list is called a subrotation. In a given rotation there can be multiple simulatenous subrotations happening at the same time.

Primary Values

- All rotations and subrotations have a primary value. Within each subrotation, create a list of every possible pair of axes from itself and the next axis. (including the last and first axis as a pair*)
- If the subrotation contains exactly 1 axis, ignore the above rule and make only 1 pair, consisting of the axis repeated twice.
- For each pair, get the value from the table, using the first letter as the row and the second letter as the column.
- If 1 of the 2 axes are negative, multiply the pair's result with -1.

** Even with 2 axes, this rule still applies. For example, subrotation +R-T gives +R-T and -T+R.*

	X	Y	Z	W	V	U	R	S	T	O	
X	1	2	5	1	8	8	1	5	2	1	X
Y	2	3	6	2	9	9	2	6	3	2	Y
Z	9	1	4	9	7	7	9	4	1	9	Z
W	1	2	5	1	8	8	1	5	2	1	W
V	2	3	6	2	9	9	2	6	3	2	V
U	9	1	4	9	7	7	9	4	1	9	U
R	1	2	5	1	8	8	1	5	2	1	R
S	2	3	6	2	9	9	2	6	3	2	S
T	9	1	4	9	7	7	9	4	1	9	T
O	1	2	5	1	8	8	1	5	2	1	O
	X	Y	Z	W	V	U	R	S	T	O	

Primary Values (...continued)

- The absolute sum of all pairs on all subrotations is the primary value of that rotation. Later in this manual, whenever p_1, p_2, or p_3 is mentioned, it refers to the primary value of the 1st, 2nd, and 3rd rotation respectively.

The Anchor Sphere

- Create 4 codes, named a_0, a_1, a_2, and a_3, each starting with the value "0000000000". a_{1-3} represent rotations 1-3.
- Subtract the largest number equal or less than p_1 found in "Decimal <-> Binary" from it, then set a_1's Xth digit from the left to 1, where X is the position obtained from that same number that was subtracted with. Keep subtracting and setting digits to 1 until p_1 is 0.
- Repeat the above step using p_2 and a_2, as well as p_3 and a_3.

Decimal <-> Binary										
Subtract	512	256	128	64	32	16	8	4	2	1
Position	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th

- Look at a_1 and its rotation, for each axis, invert the number's position (0 -> 1, 1 -> 0) according to this the table below.
- Repeat this for a_2 and a_3, then set a_0 based on these 3 conditions:
- If the 6th digit of a_1 is 1, set the 1st, 2nd, and 3rd digits of a_0 to 1.
- If the 6th digit of a_2 is 1, set the 4th, 5th, and 6th digits of a_0 to 1.
- If the 6th digit of a_3 is 1, set the 7th, 8th, and 9th digits of a_0 to 1.

Axis <-> Position										
+Axis	+X	+Y	+Z	+W	+V	+U	+R	+S	+T	+O
-Axis	-O	-T	-S	-R	-U	-V	-W	-Z	-Y	-X
Position	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th

The Anchor Sphere (...continued)

a_{1-3} is now gray code, convert each one to binary:

1. The first binary digit will match the first digit of the gray code.
2. The next digit is a 1 if the sum of the previous digit of the binary code and the current position's gray code is exactly 1. Otherwise it's 0.
3. Repeat step 2 until 10 digits are obtained. This is the binary code.

- Starting from a_1, add the current a with the previous a, and then refer to the next a. Don't carry (1+1 ≠ 10) and replace 2's with 0's on each step.
- Replace every 0 with - and every 1 with +. This is now the anchor sequence. Whenever the anchor sphere is mentioned, it refers to the only 1 sphere that matches all positive/negative attributes of the anchor sequence's axes. The position of each character represents what axis they belong to, with the order being "XYZWVURSTO".

Pausing

- Interact anywhere on the module to pause it. The rotations will stop, and a sound cue is played to indicate that it is ready to be interacted with.
- Each time the module is paused, a random sphere is chosen. This is called the starting sphere. The starting sphere is white.
- The goal is to get the starting sphere to be on the same location as the anchor sphere.

Navigation

- When the module is interacted with during the last digit being 0-9, an axis is queued. Each submission from 0-9 represents an axis, though order is random.
- 3 axes need to be queued for a valid input. When the timer's last digit is a 9, it will try submitting the 3 axes. The queue is cleared if any other number of axes are queued.
- The starting sphere goes to the other side of all 3 axes that were submitted.
- During this submission, all axes can only be submitted up to 2 times.
- This rule can be violated up to four times. The 5th time causes a strike.
- When the starting sphere is in the same position as the anchor sphere, submit all 10 axes. The module will strike or solve accordingly.
- Striking the module will unpause the module.