# A Modern Array Language for Ar-

tificial Life Research

Noah Syrkis

October 6, 2025

1 | Motivation

2 | Glyphs

3 | The Stack

4 | Example: RHOS

5 | Example: Sortsol

#### 1 | Motivation

For legacy reasons ALife research overwhelmingly largely use Python. This talk presents an alternative: Uiua, a stack based array language written in Rust inspired by APL. Working with non-coders (architects and artists), I've found that they find Uiua intuitive. Further, Uiua's array orientedness and visual compactness makes it a winner (for me), and it's already used in neuroevolutiuon [1]

lambda x, y, z: 
$$x + y + z$$
 + +

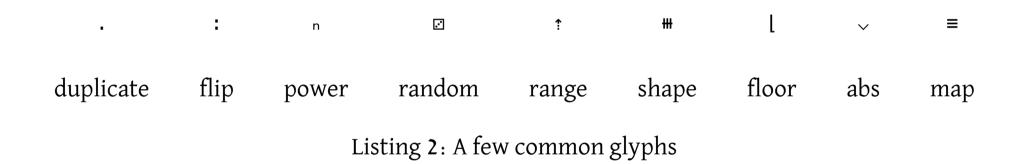
Python

Uiua

Listing 1: Two implementations of anonymous function that adds three numbers

# 2 | Glyphs

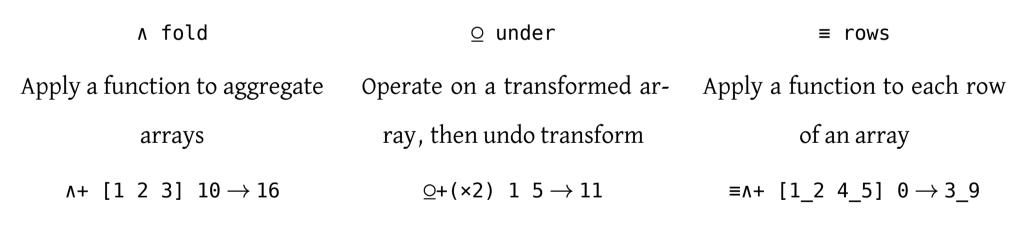
Uiua uses special symbols (glyphs). Two frequent initial reactions to these are intrigue (can you feel it?) and incredulity<sup>1</sup>. If you play with Uiua both quickly subside. A few common glyphs are:



<sup>&</sup>lt;sup>1</sup>I just got my drivers license and learning the signs of the road was harder than learning Uiua

# 2 | Glyphs

Some glyphs are more high-level. To visualize points, you could use  $\bigcirc$  (under), which transforms, applies a function, and undoes transform, on an  $n \times n$  array of zeros, by selecting the index of the particles in the array (transform), adding 1 (function), and putting the values back in the array.



Listing 3: Glyphs with descriptions and examples

1 2 3 # stack: 1 2 3

- ► Functions pop and push values on the stack
- ► Functions apply seamlessly across array dims

```
1 2 3 # stack: 1 2 3
+ + # stack: 6
```

- ► Functions pop and push values on the stack
- ► Functions apply seamlessly across array dims

- ► Functions pop and push values on the stack
- ► Functions apply seamlessly across array dims

- ► Functions pop and push values on the stack
- ► Functions apply seamlessly across array dims

```
1 2 3 # stack: 1 2 3
```

++ # stack: 6

\* # stack: [0 1 2 3 4 5]

+ 1 # stack: [1 2 3 4 5 6]

- ► Functions pop and push values on the stack
- ► Functions apply seamlessly across array dims

- ► Functions pop and push values on the stack
- ► Functions apply seamlessly across array dims

### 4 | Example: RHOS

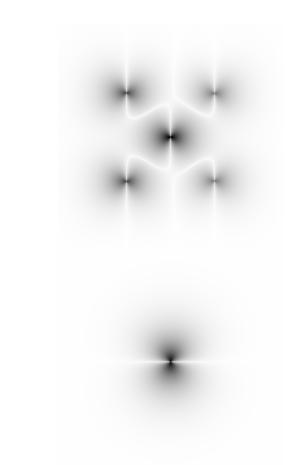


Figure 1: Two works from RHOS

### 5 | Example: Sortsol

```
Init \leftarrow + ÷ 2 W × S - 0.5 gen N_2

Step \leftarrow + × S -0.5 gen N_2

Draw \leftarrow \bigcirc(\boxdot|+1) 1 0 \mp -1 W [ : × 0 ^{\circ}\triangle W_W
```

Figure 2: 200 steps of a simple particle simulation with particles moving randomly

References

[1] Kai Schmidt, "Evonet: Basic Evolutionary Neural Network in Uiua." 2025.

#### A | Saving

```
≡Draw Λ(.Step) ↑ T Init R # run sim
&fwa "out.gif" gif 48 - : 1 # save gif
Listing 4: How Sortsol was run (top) and saved to gif (bottom)
```