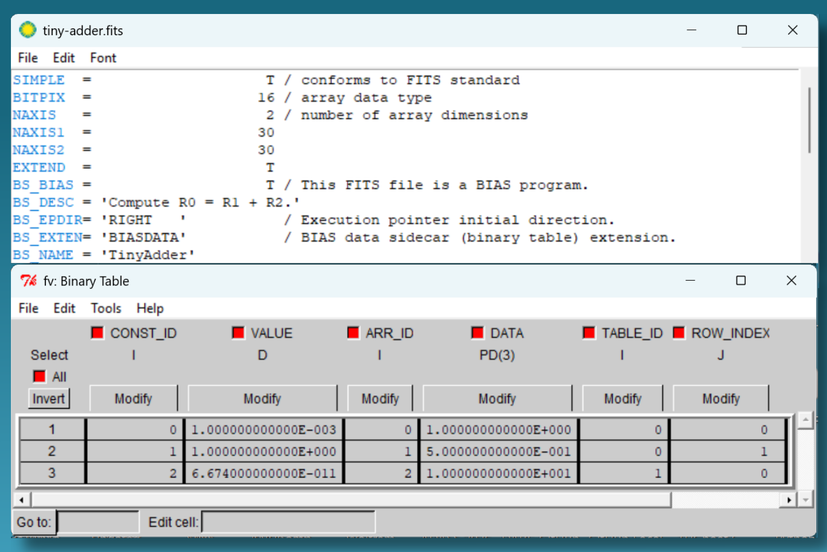
BIAS

Bias Images As Syntax



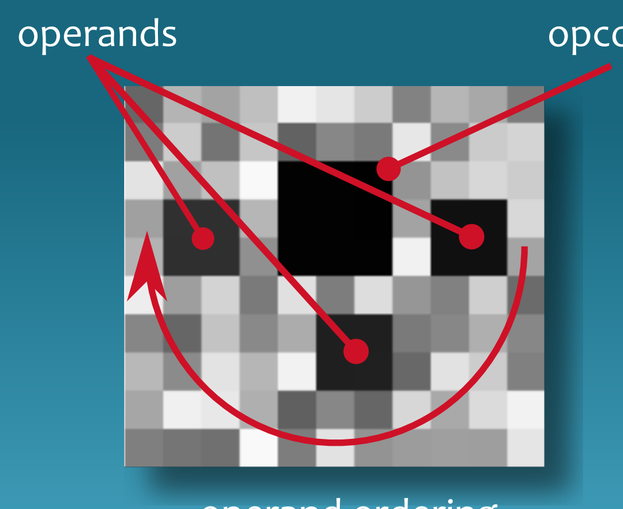
Hybrid Data Sidecar

BIAS programs support two data sidecars. FITS headers provide simple key-value data and metadata. A **BIASDATA** BINTABLE extension provides bulk numerical input such as constants, arrays and tables.

AI Co-designed Esolang

BIAS (Bias Images As Syntax) is a programming language that encodes logic as CCD bias frames. Inspired by Piet [1], it was created as an exercise in experimental visual programming.

BIAS is capable of general-purpose programming, but is classified here as an esoteric programming language (esolang) due to its unconventional visual syntax. The language was developed with assistance from Microsoft Copilot.



Opcodes and Operands

Opcodes specify operations. Operands provide instruction parameters. BIAS programs are organised so that opcodes appear as central pixel regions surrounded by satellite operand regions. Operands are ordered by clockwise angular displacement measured from the positive x-axis of a Cartesian plane centred on the opcode centroid.

Region Detection

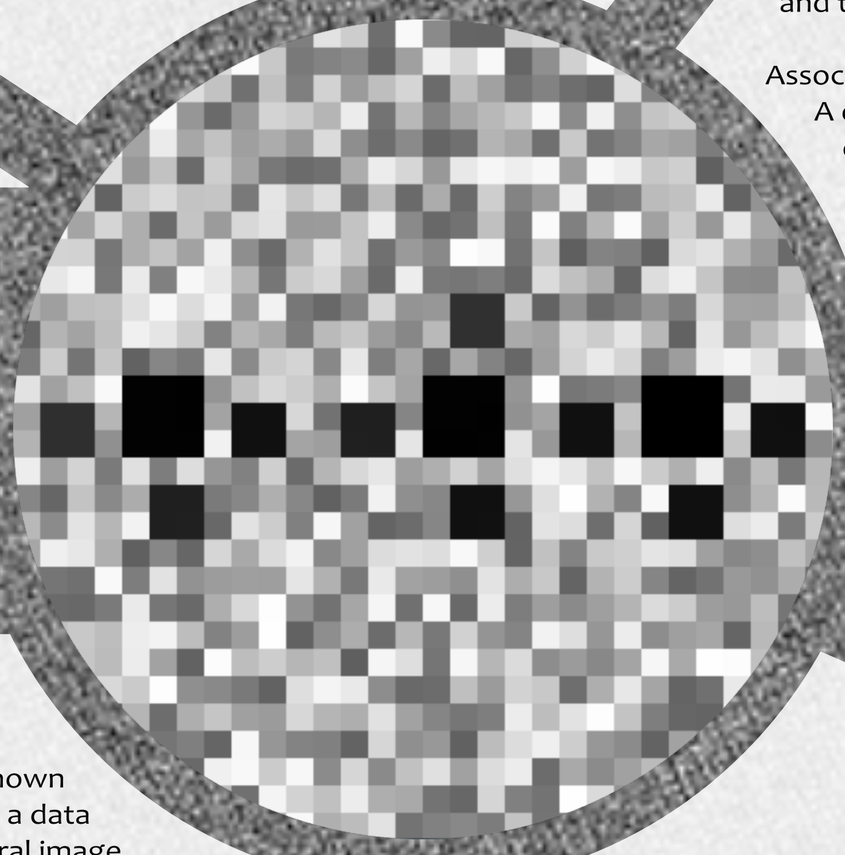
BIAS runs on FITS files with a 16-bit (0–65535) intensity range. Reserved bands in the intensity range are treated as BIAS syntax. Other pixels are ignored as background data. BIAS instructions comprise pixel regions of common mean pixel intensity.

BIAS scans FITS images top-to-bottom and left-to-right. Background pixels are skipped. A 4-connectivity flood fill bounds each instruction region. Region extent and centroid are calculated, and the region classified as an **opcode** or **operand**. Scanning continues until all instruction regions have been identified.

An example BIAS program is shown in the central image, below.

FITS Image

BIAS is an esoteric programming language that encodes programs as astronomical CCD bias frames.



Instruction Construction

Each BIAS opcode and operand has a dedicated mean region intensity band. Opcode regions may have associated operand regions. E.g., 100 + 17 may use an ADD opcode and two operand regions.

Associating operands to opcodes is done by adjacency. A one-pixel gap associates an operand with an opcode. To avoid ambiguity, regions may not directly abut one another.

Operands are positional parameters for opcodes. Ordering is determined geometrically. Operands are ordered by the clockwise angular displacement of their centroids, measured from the positive x-axis centred on the opcode centroid.

Opcodes and their ordered operands are then saved as BIAS Instructions.

Program Output

Example output from a Python interpreter is shown below. It reads two floating point values from a data sidecar and sums them. BIAS code is the central image, above right.

```
(.venv) PS C:\bias-images-as-syntax> python src/python/
bs_interpreter.py examples/tiny-adder.fits
BIAS program result: 100.0 + 17.0 = 117.0.
```

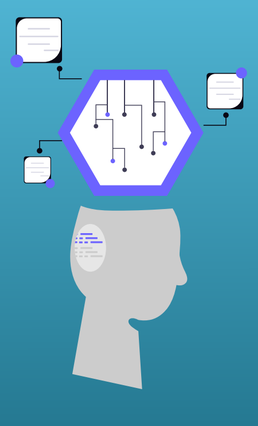
BIAS example program output.
Floating point addition is demonstrated:
100.0 + 17.0 = 117.0.

Execution Pointer

BIAS maintains an execution pointer that traverses program pixels. Pointer state includes the current pixel, active region and traversal direction.

The entry opcode is the region whose centroid is nearest the image's top-left corner. Ties are resolved using the operand-ordering geometry.

After executing the entry instruction, the execution pointer traverses the image. Instructions execute when opcode regions are encountered. Flow may be altered by traversal-direction changes or JUMP instructions. Execution halts at the image boundary.



Microsoft Copilot

BIAS was conceived by the author and developed with assistance from the Microsoft Copilot desktop application. Conversational engineering was used to develop architectural elements and associated source code. Microsoft Copilot is freely available to anyone with a Microsoft account.

Discussion

BIAS was not intended to be a parody (e.g., INTERCAL), a work of software art (e.g., Piet), or a deliberate obfuscation (e.g., MALBOLGE) [2, 3, 4]. BIAS was a low-stakes project for testing the capabilities of Microsoft Copilot, an AI tool needing no investment of time or money before use, in contrast to locally-trained large language models or fee-based services.

AI systems can struggle to generate working esolang programs. Performance on BIAS program-generation tasks may help distinguish between AI systems capable of reasoning about novel programming scenarios and those that rely on memorised patterns or retrieval-augmented generation.

THE UNIVERSITY OF SYDNEY

Travis Stenborg

[1] Buda, A. G., & Primiero, G. 2024, Minds and Machines, 34, S139.
[2] Mateas, M., & Montfort, N. 2005, in Proc. 6th Digital Arts & Culture Conference (IT Univ. Copenhagen), 144.
[3] McLean, A., & Wiggins, G. 2012, in Computers and Creativity, eds. J. McCormack, & M. d'Inverno (Berlin: Springer), 235–252.
[4] DuPont, Q. 2020, in Second International Handbook of Internet Research, eds. J. Hunsinger, M. M. Allen, & L. Klastrup (Dordrecht: Springer), 691–705.

The poster background CCD bias frame was taken from the 2.3 m Advanced Technology Telescope.